Manual

RuSseL: A Self-Consistent Field Theory Code for Inhomogeneous Polymer Interphases

Contents

1. Introduction 4

2. Setup 5

2.1. Version 5

2.2. Download 5

2.3. Compilation 5

2.4. Integrity test 6

2.5. How to Run 6

3. Structure of input files 7

4. Commands 8

4.1. Domain setup 8

4.2. Thermodynamics 9

4.3. Polymer characteristics 9

4.4. nonbonded interactions and free energy density 10

4.5. Matrix chains 11

4.6. Grafted chains 13

4.7. Walls 14

4.8. Field and convergence parameters 18

4.9. Solvers and Boundary Conditions 20

4.10. Exports frequency 23

4.11. Computes 24

5. Master table 28

6. Demo input files 30

6.1. Vacuum-matrix interface (VM) 30

6.2. Solid-polymer interface: square well potential 31

6.3. Solid-polymer interface (SM): tabulated potential 31

6.4. Solid-matrix-solid geometry (SMS) 32

6.5. Grafted-matrix-grafted geometry (GMG) 33

6.6. Grafted-matrix-vacuum geometry (GMV) 34

6.7. Polymer-grafted nanoparticle in vacuum 35

6.8. Polymer-grafted nanoparticle in melt 36

7. References 38

1. Introduction

The present document aims to serve as a comprehensive manual for anyone who desires to use or extend the functionalities of the *RuSseL1D* code. It gives detailed instructions for the compilation and the execution of the code and then it describes the structure of the input file, which fully controls the code execution. Afterwards, it offers a thorough description of all the commands that the user can issue through this input file to control the calculations. At the moment, the code addresses homopolymer melts in contact with bare or grafted solid surfaces of planar or spherical geometry and computes a series of thermodynamic and structural properties of the interfacial system. When grafted chains are present in the system, these may interact with matrix chains or may interact exclusively with the solid wall and with each other (i.e., matrix chains do not exist in the system).

The code has been written in Fortran90 in a modular fashion so that anyone can easily contribute by extending its functionalities according to his needs, e.g., wall potentials, equations of state, solvers, copolymer systems. The document concludes with a master lookup table summarizing all user options and a series of example input files reproducing published results regarding systems addressed by the authors in the past, along with the corresponding references. As it is also mentioned in the main text, the source code of *RuSseL1D* is uploaded on *Github*, where it is publicly accessible for viewing, downloading and editing.

2. Setup

2.1. Version

The current version of the code is the first one to be publicly distributed and it is given the number `1.0.0`.

2.2. Download

You can find and download the source code of *RuSseL1D* on the following Github repository:
`https://github.com/cjrevelas/RuSseL1D`
In this repository, the user will find:

- a `README` file containing a short description of the code along with a few compilation instructions
- the `LICENSE` file of the code
- the `CMakeLists.txt` file controlling the compilation process
- a `clean.sh` bash script removing all CMake built files
- a `doc/` directory containing the code documentation files
- an `examples/` directory containing demo input files
- an `src/` directory containing the fortran source files
- a `run/`directory containing the generated executable file (`RSL1D`)
- a `tools/` directory containing pre- and post-processing scripts
- a `test_integrity/`directory containing integrity tests

By default, the input files start with the "`in.`" prefix (e.g., `in.input`, `in.field`, `in.table`), whereas the output files start with the "`o.`" prefix.

2.3. Compilation

The Fortran source code can be compiled with `GNU Fortran (GCC) 5.1.0` or `ifort (IFORT) 13.0.1 20121010` using the `Makefile` in the root of the Github repository. The user selects to compile the code in `debug` or `release` configuration by selecting the appropriate compilation flags. The following table presents the compiler flags for each one of the two aforementioned compilers and each configuration.

Table 1: RuSseL1D compilation flags.

Compilation flags	GNU Fortran (GCC)	ifort (IFORT)
Debug	-m64 -g -O0 -pedantic-errors -frepack-arrays -fdump-core -fbounds-check -fimplicit-none -fbacktrace -ffree-line-length-none -frange-check -Wall -Waliasing –Wampersand -Wsurprising -Wunderflow -W	-O0 -g -traceback -fpe0 -fp-stack-check -heap-arrays -ftrapuv -check pointers -check bounds -warn all
Release	-O3 -ffree-line-length-none -m64 -mtune=native -march=native	-O2 -prec-sqrt -prec-div -align -static -ip -ipo -heap-arrays

When setting the desired compilation flags in the `CMakeLists.txt`, then the compilation of the code is performed by simply running the following commands:

```
cmake .
make
```

After the successful compilation of the code, an executable will be generated by the name `RSL1D` inside the `run/` directory.

2.4. Integrity test

The repository contains a `test_integrity/` directory, which in turn contains a series of directories with templated output of the code under various input settings. When the user decides to make changes in the code for his own purposes, then it is highly recommended to run these tests to ensure that no bugs were accidentally inserted in the code. The comparison with respect to the test files is performed via the bash script `test_integrity.sh`, which is also contained in the same directory. The user has also the option of replacing the aforementioned tests using the same script by just swapping the variables of the following variables of the bash script:

```
replace_log=true
inspect_log=false
```

In order to inspect or replace the integrity tests, the user can simply go to the root directory of the code (i.e., the one containing the `Makefile)` and run the following command:

```
make rsltest
```

2.5. How to Run

To run the executable `RSL1D`, the user must specify the path of this executable file and the path of the input file, as follows:

```
path_to_RuSseL_directory/run/RSL1D Path_to_input_file
```

This means that the location of the input file is an argument of the execution command. In case the path of the input file is not specified, the code will attempt by default to read the file from the present directory. For example, in case the executable (`RSL1D)` and the input file (`in.input`) are in the same directory, the user simply has to type:

```
./RSL1D
```

In case the input file does not exist at all, the code will exit after printing a relevant error message to the standard error output (`console`).

3. Structure of input files

The input files include all the necessary user inputs regarding the domain, the convergence parameters and the polymer properties for which the calculations are to be conducted. All input files are read *once* during the initialization of the program by the `parser.f90` subroutine. This subroutine is designed to parse the input file line-by-line and whenever it spots specific flag identifiers, it attempts to record the corresponding user input(s).

The format of a valid user command is the following:

```
ARGUMENT(S) ! IDENTIFIER (COMMENTS)
```

The first column(s) of the input file includes the user specified `ARGUMENT(S)`. If a several number of arguments are required, they can be separated by whitespace or tab. The `IDENTIFIER` must be located after the user arguments. Its precise location, and whether there are comments before or after it, does not matter, since the parser only checks if the `IDENTIFIER` is included within the line. With a few exceptions, the order of the issued commands does not matter either.

The parser will skip every line that:

- does not include an `IDENTIFIER`
- starts with "`#`" or "`!`"

If the input file includes more than one line with the same `IDENTIFIER`, the last one overwrites the preceding ones.

In situations where the user specifies unphysical values (e.g., negative temperature) or omits to specify the value of a necessary parameter, the parser will issue an error and terminate the program after printing a relevant exception message to the standard error output (console).

4. Commands

4.1. Domain setup

4.1.1. Geometry

Syntax*:*

- identifier: “domain geometry”
- args = 0 or 1 (kind: integer)
  0 → planar geometry
  1 → spherical geometry

Examples:

```
0 ! domain geometry # Planar geometry
1 ! domain geometry # Spherical geometry
```

Description:

Specifies the geometry of the system.

Default:

None

4.1.2. Sphere radius

Syntax:

- identifier: “sphere_radius”
- arg = radius of nanoparticle/spherical cavity, R_{NP} (kind: positive real, units: Å)

Examples:

```
20 ! sphere_radius # Specify a NP with RNP = 20 Å
```

Description:

Set the radius of the nanoparticle. It is always required for spherical geometries.

Default:

None

4.1.3. Size

Syntax:

- identifier: “domain lx”
- arg = size of the domain, l_x (kind: positive real, units: Å)

Examples:

```
100 ! domain lx # Set the length of the domain to lx = 100 Å
```

Description:

Set the length of the domain.

Default:

None

4.1.4. Discretization

Syntax:

- identifier: “domain dx”
- arg = discretization of the domain, Δh (kind: positive real, units: Å)

Examples:

```
0.5 ! domain dx # Set the spatial discretization to 0.5 Å
```

Description:

Sets the discretization of the domain.

Default:

None

4.2. Thermodynamics

4.2.1. Temperature

Syntax:

- identifier: "system temperature"
- arg = system temperature, T (kind: positive real, units: Kelvin)

Examples:

```
500 ! system temperature # Set the temperature to 500 K
```

Description:

Sets the temperature of the system.

Default:

None

4.2.2. Pressure

Syntax:

- identifier: "system pressure"
- arg = system pressure, P (kind: real, units: atm)

Examples:

```
0 ! system pressure # Set the pressure to 0 atm
```

Description:

Sets the pressure of the system.

Default:

0 atm

4.3. Polymer characteristics

4.3.1. mass density across the bulk phase

Syntax:

- identifier: "polymer mass_density"
- arg = mass density in the bulk phase, $\rho_{mass,bulk}$ (kind: positive real, units: g/cm^3)

Examples:

```
52 ! polymer monomer mass # Set the monomer mass to 52 g/mol
```

Description:

Sets the mass density of the polymer across the bulk phase.

When using HFD EoS, $\rho_{mass,bulk}$ must be specified.

When using SL-EoS, $\rho_{mass,bulk}$ is recomputed based on the vapor-liquid equilibria[1].

Default:

None

4.3.2. Monomer mass

Syntax:

- identifier: "polymer monomer mass"
- arg = monomer mass, $m_{monomer}$ (kind: positive real, units: g/mol)

Examples:

```
52 ! polymer monomer mass # Set the monomer mass to 52 g/mol
```

Description:

The monomer mass is required for the segment density in the bulk to be calculated: $\rho_{seg,bulk} = N_A \rho_{mass,bulk} / m_{monomer}$.

Default:

none

4.3.3. Bond length

Syntax:

- identifier: "polymer bond_length"
- arg = the bond length among consecutive chain segments, $l_{c\text{-}c}$ (kind: positive integer, units: Å)

Examples:

```
1.54 ! polymer bond_length # Set the bond length to 1.54 Å.
```

Description:

Sets the value of the bond length between consecutive polymer segments.

The bond length coupled with the characteristic ratio of the polymer chains define the squared radius of gyration per monomer as: $R_g^2 / N_c = C_\infty \, l_{C-C}^2 / 6$

Default:

none

4.3.4. Characteristic ratio

Syntax:

- identifier: "polymer C_inf"
- arg = characteristic ratio, C_∞, (kind: positive real)

Examples:

```
10 ! characteristic ratio # Set the characteristic ratio to 10
```

Description:

The characteristic ratio, coupled with the bond length set the radius of gyration per chain length as: $R_g^2 / N_c = C_\infty \, l_{C-C}^2 / 6$

Default:

None

4.4. nonbonded interactions and free energy density

4.4.1. EoS type

Syntax

- identifier: "EOS type"
- *args* = 0 or 1 (kind: integer)
  0 → Helfand
  1 → Sanchez-Lacombe

Examples:

```
0 ! EOS type # Activate the Helfand EoS
1 ! EOS type # Activate the SL EoS
```

Description:

Sets the equation of state that will be used for the description of the nonbonded interactions among the polymer segments.

Default:

none

4.4.2. Helfand coefficients

Syntax:

- identifier: "EOS coeffs"
- arg = isothermal compressibility, κ_T (kind: positive real, units: GPa^{-1})

Examples:

```
10 ! EOS coeffs # Set the isothermal compressibility of HFD EoS to 10 GPa^-1
```

Description:

Sets the isothermal compressibility for HFD EoS.

Default:

None

Restrictions:

Must be defined *after* EOS type

4.4.3. Sanchez-Lacombe coefficients

Syntax

- identifier: "EOS coeffs"
- arg 1 = characteristic SL density, ρ^* (kind: positive real, units: kg/m^3)
- arg 2 = characteristic SL temperature, T^* (kind: positive real, units: Kelvin)
- arg 3 = characteristic SL pressure, P^* (kind: positive real, units: Pa)

Examples:

```
1105.0 735.0 3.57D+08 ! EOS coeffs # Set the characteristic density, temperature and
                        pressure for SL EoS
```

Description:

Sets the coefficients (ρ^*, T^* and P^*) for SL EoS.

Default:

None

Restrictions:

Must be defined *after* EOS type

4.4.4. Square gradient

Syntax 1:

- identifier: "real_influence_parameter"
- arg = influence parameter, κ (kind: positive real, units: J m^5/mol^2)

Syntax 2:

- identifier: "influence_parameter"
- arg = reduced influence parameter, $\tilde{\kappa}$ (kind: positive real)

Examples:

```
0.223344930D-66 ! EOS real_influence_parameter # Set the influence parameter in units
                                                (J m^5/mol^2)
0.55            ! EOS influence_parameter      # Set the influence parameter in
                                                dimensionless units
```

Description:

The first syntax sets the influence parameter (κ) in real (J m^5/mol^2) units. The alternative one sets it in reduced units ($\tilde{\kappa}$) and can only be used when the SL-EoS is activated. The conversion between reduced and real units is performed with the following expression:

$$\kappa = 2(r_{SL} / N)^2 P^* \left(\upsilon^*\right)^{8/3} \tilde{\kappa}$$

with r_{SL}, P^*, and υ^* derived from the SL-EoS.

Default:

None

Restrictions:

Syntax 2 can only be used when the SL-EoS is activated.

4.5. Matrix chains

4.5.1. Enable matrix chains

Syntax:

- identifier: "matrix set"
- arg = True or False (kind: logical)

Examples:

```
True  ! matrix set # Activate the matrix chains
False ! matrix set # Deactivate the matrix chains
```

Description:

In case matrix chains are not set or deactivated, the subsequent commands regarding the chain length, contour discretization are ignored.

Default:

None

4.5.2. Chain length

Syntax:

- identifier: "matrix chain_length"
- arg = chain length of matrix chains, N_m (kind: positive real, units: number of monomers/segments)

Examples:

```
100 ! matrix chain_length # Set the chain length of matrix chains to 100
```

Description:

Sets the length of matrix chains in terms of monomer units.

Default:

None

4.5.3. Contour discretization

Syntax:

- identifier: "matrix ds"
- arg = Contour length discretization of matrix chains, ΔN_m (kind: positive real)

Examples:

```
0.25 ! matrix ds # Set the contour discretization of matrix chains to 0.25
```

Description:

Sets the contour discretization of matrix chains.

Default:

None

4.5.4. Critical adsorption distance

Syntax:

- identifier: "chain r_ads_lo" and "chain r_ads_hi"

- arg = Critical adsorption distance, r_{ads} (kind: positive real, units: Å)

Examples:

```
10 ! chain r_ads_lo # Set the critical adsorption distance at the lo bound to 10 Å
5 ! chain r_ads_hi # Set the critical adsorption distance at the hi bound to 5 Å
```

Description:

Sets the critical adsorption distance of matrix chains.

Default:

0 (no adsorption)

Restrictions:

Currently, this calculation is supported only for matrix chains.

4.6. Grafted chains

4.6.1. Enable grafted chains at the bottom/top side

Syntax:

- identifier: "grafted lo set" or "grafted hi set"
- arg = True or False (kind: logical)

Examples:

```
True  ! grafted lo set # Activate the grafted chains at the bottom side
True  ! grafted hi set # Activate the grafted chains at the top side
False ! grafted lo set # Deactivate the grafted chains at the bottom side
False ! grafted hi set # Deactivate the grafted chains at the top side
```

Description:

Activates/deactivates the presence of grafted chains.
In case the grafted chains at the bottom side are not set or deactivated, the subsequent commands regarding the chain length, contour discretization and the grafting density are ignored.

Default:

None

4.6.2. Chain length

Syntax:

- identifier: "grafted lo chain_length" or "grafted hi chain_length"
- arg = chain length of g± chains, $N_{g^{\pm}}$ (kind: positive real)

Examples:

```
100 ! grafted lo chain_length # Set the chain length chains grafted at the
                                bottom side to 100 Å
```

Description:

Sets the length of chains grafted at the bottom/top side in terms of monomers units.

Default:

None

4.6.3. Contour discretization

Syntax:

- identifier: "grafted lo ds" or "grafted hi ds"
- arg = Contour length discretization of g± chains, $\Delta N_{g^{\pm}}$ (kind: positive real)

Examples:

```
0.25 ! grafted lo ds # Set the contour discretization of chains grafted at the
                       bottom side to 100 Å
```

Description:

Sets the contour discretization of chains grafted at the bottom/top side

Default:

None

4.6.4. Grafting density

Syntax:

- identifier: "grafted lo grafting_density" or "grafted hi grafting_density"
- arg = Grafting density of g± chains, $\sigma_{g^{\pm}}$ (kind: positive real, units: chains/ Å^2)

Examples:

```
0.004 ! grafted lo ds # Set the grafting density at the bottom wall to 0.001 chains / Å^2
```

Description:

Sets the grafting density (σ_g) at the bottom/top side

Default:

None

4.6.5. Distance of grafted chains from the walls

Syntax:

- identifier: "grafted distance_from_solid"
- arg = grafted point-wall distance, h_g (kind: positive real, units: Å)

Examples:

```
5 ! grafted distance_from_solid # Set the distance of the grafting point
                                  from the solid was to 5 Å
```

Description:

Sets the distance of the grafting point from the solid wall to 5 Å

Default:

By default the grafting points are set to the second node of the domain ($h_g = \Delta h + h_{HS}$).

Restrictions:

In case the distance is set manually, it must be larger than the distance of the hard-sphere wall from the solid surface.

Currently, the distance of the grafting point from the walls applies both for g^- and g^+ chains.

4.7. Walls

4.7.1. Type

Syntax:

- identifier: "wall type"
- arg = –1,0,1,2,3, 9, 10 (kind: integer)
  –1→ hybrid
  0 → vacuum
  1 → Hamaker
  2 → square well
  3 → ramp
  9 → custom
  10 → table

Examples:

```
-1 ! wall type # Set the wall type to hybrid
 0 ! wall type # Set the wall type to vacuum
 1 ! wall type # Set the wall type to Hamaker
```

Description:

Sets the wall type

Default:

By default the wall type is set to "0" (vacuum)

4.7.2. Coeffs → Hamaker

Syntax:

- identifier: "wall coeffs"
- arg 1 = collision diameter of polymer monomers, σ_{pol} (kind: positive real, units: Å)
  arg 2 = collision diameter of wall monomers, σ_{sol} (kind: positive real, units: Å)
  arg 3 = Hamaker constant of polymer, A_{pol}(kind: real, units: 10^{-20} J)
  arg 4 = Hamaker constant of solid, A_{sol} (kind: real, units: 10^{-20} J)

Examples:

```
3.7 3.0 5.84 6.43 ! wall coeffs # set σpol, σsol, Apol and Asol
```

Description:

Sets the necessary parameters for the calculation of the Hamaker potential.[2–4]

● For spherical geometries, the segment-wall interactions are described by the following equations

$$u_{\mathrm{A}} = -\frac{A_{12}}{6}\left[\frac{2a_1a_2}{{r_{12}}^2-\left(a_1+a_2\right)^2}+\frac{2a_1a_2}{{r_{12}}^2-\left(a_1-a_2\right)^2}+\ln\left(\frac{{r_{12}}^2-\left(a_1+a_2\right)^2}{{r_{12}}^2-\left(a_1-a_2\right)^2}\right)\right] \quad \text{S1}$$

$$u_{\mathrm{R}} = \frac{A_{12}}{37800}\frac{{\sigma_{\mathrm{sm}}}^6}{r_{12}}\left[\begin{array}{l}\dfrac{{r_{12}}^2-7r_{12}\left(a_1+a_2\right)+6\left(a_1^2+7a_1a_2+{a_2}^2\right)}{\left(r_{12}-a_1-a_2\right)^7}+\dfrac{{r_{12}}^2+7r_{12}\left(a_1+a_2\right)+6\left(a_1^2+7a_1a_2+{a_2}^2\right)}{\left(r_{12}+a_1+a_2\right)^7}\\ -\dfrac{{r_{12}}^2+7r_{12}\left(a_1-a_2\right)+6\left(a_1^2-7a_1a_2+{a_2}^2\right)}{\left(r_{12}+a_1-a_2\right)^7}-\dfrac{{r_{12}}^2-7r_{12}\left(a_1-a_2\right)+6\left(a_1^2-7a_1a_2+{a_2}^2\right)}{\left(r_{12}-a_1+a_2\right)^7}\end{array}\right] \quad \text{S2}$$

where $A_{12}=A_{\mathrm{SM}}=\sqrt{A_{\mathrm{pol}}A_{\mathrm{sol}}}$, $a_1=a_{\mathrm{M}}=\sqrt[3]{3/\left(4\pi\rho_{\mathrm{seg,bulk}}\right)}$, $a_2 = R_{\mathrm{NP}}$, $\sigma_{\mathrm{SM}}=0.5\left(\sigma_{\mathrm{pol}}+\sigma_{\mathrm{sol}}\right)$, and r_{12} is the distance between the centers of the spheres.

● For planar geometries, the segment-wall interactions are described by the following equations:

$$u_{\mathrm{A}}^{\mathrm{SM}} = -\frac{A_{\mathrm{SM}}}{6}\left(\frac{1}{r'}+\frac{1}{2+r'}+\ln\left(\frac{r'}{2+r'}\right)\right) \quad \text{S3}$$

$$u_{\mathrm{A}}^{\mathrm{SM}} = -\frac{A_{\mathrm{SM}}}{6}\left(\frac{1}{r'}+\frac{1}{2+r'}+\ln\left(\frac{r'}{2+r'}\right)\right) \quad \text{S4}$$

$r' = d_{12} / a_{\mathrm{M}}$ with d_{12} being the distance between the surface of the sphere from the solid surface.

In situations where the Hamaker potential has been applied to the opposing surfaces [5], an additional term is activated that described the solid-solid interactions:

$$U_{\mathrm{ss}}(h_{\mathrm{ss}}) = S_{\mathrm{solid}}\frac{A_{\mathrm{SMS}}}{\pi}\left[\frac{{\sigma_{\mathrm{S}}}^6}{360{h_{\mathrm{ss}}}^8}-\frac{1}{12{h_{\mathrm{ss}}}^2}\right] \quad \text{S5}$$

where $A_{\mathrm{SMS}} = A_{\mathrm{sol}}$ and h_{ss} is the solid-solid distance.

Restrictions

Must be defined *after* the wall type.

4.7.3. Coeffs → square well

Syntax:

- identifier: "wall coeffs"
- arg 1 = range of the square well potential, σ_{well} (kind: positive real, units: Å)
  arg 2 = depth of the square well potential, v_{well} (kind: positive real, units: 10^{-20} J)

Examples:

```
10.0 -1.0 ! wall coeffs # set σwell and uwell
```

Description:

The polymer-solid interactions are described by a square well potential of the form:

$$u_{\mathrm{square\_well}} = v_{\mathrm{square\_well}} \quad \forall h < \sigma_{\mathrm{square\_well}} \quad \text{S6}$$

Default:

none

Restrictions

Must be defined *after* the wall type.

4.7.4. Coeffs → ramp

Syntax:

- identifier: "wall coeffs"
- arg 1 = range of the ramp potential, σ_{ramp} (kind: positive real, units: Å)
  arg 2 = depth of the ramp potential, v_{ramp} (kind: positive real, units: 10^{-20} J)

Examples:

```
10.0 -1.0 ! wall coeffs # set σramp and uramp
```

Description:

The polymer-solid interactions are described by a square well potential of the form:

$$u_{ramp} = v_{ramp} \max\left(\frac{\sigma_{ramp} - h}{\sigma_{ramp}}, 0\right) \qquad \text{S7}$$

where σ_{ramp} is the range and v_{ramp} is the depth of the potential.

Default:

none

Restrictions

Must be defined *after* the wall type.

4.7.5. Coeffs → custom wall potential

Syntax:

- identifier: "wall coeffs"
- arg 1 = number of coefficients for the custom wall potential (kind: integer)
  arg 2 = first argument for the custom wall potential
  arg 3 = second argument for the custom wall potential
  arg n+1 = n^{th} argument for the custom wall potential

Examples:

```
5 -1.86388E-19  4.0 5.0 -1.863875502E-20 2.5 ! wall coeffs
```

Description:

The polymer-solid interactions are described by a custom wall potential whose functional form is specified by the user during compile time. The form of the potential is specified in the subroutine `init_solid.f90`, inside the conditional:

```
   if (wall_custom) then
      Uatt = Uatt + exp(-xx/wall_custom_vars(2)) * &
&            ( wall_custom_vars(4) + wall_custom_vars(1) * &
&             sin((2 * PI / wall_custom_vars(3)) * &
&            (xx+ wall_custom_vars(5)) ))
   endif
```

The example above implements a custom potential of the form:

$$u_{custom}(r) = e^{-\frac{r}{u_2}}\left(u_4 + u_1 \sin\left(\frac{2\pi}{u_3}\left[r + u_5\right]\right)\right) \qquad \text{S8}$$

where $u_1 = -1.86388 \cdot 10^{-19}$ J, $u_2 = 4.0$ Å, $u_3 = 5.0$ Å, u4 = $-1.863875502 \cdot 10^{-20}$ J, and $u_5 = 2.5$ Å.

Default:

none

Restrictions

Must be defined *after* the wall type.

4.7.6. Coeffs → tabulated potential

Syntax:

- identifier: "wall coeffs"
- arg = no coefficients are required

Description:

The tabulated potential is read from the input file: "`in.table`" The first column displays the distance from the wall in Angstrom units and the second one the interaction energy in Joules.

Default:

none

4.7.7. Coeffs → hybrid

Syntax:

- identifier: "wall coeffs"
- arg 0 = identified of the wall type (e.g., 1, 2, 3)
  args = arguments of the corresponding wall type

Examples:

```
1   3.7 3.0 5.84 6.43 ! wall coeffs # set σpol, σsol, Apol and Asol
3   10.0 -1.0         ! wall coeffs # set σramp and uramp
```

Description:

The hybrid wall style allows the user to activate multiple potentials at the same time. The syntax in this style differs than the syntax in the other wall coeff styles in that the list of coefficients if precided by the identified of the corresponding wall type. In the example above, both the Hamaker and ramp potential are activated at the same time.

Default:

none

Restrictions

Must be defined after wall type
Each line must be preceded by the corresponding wall type.

4.7.8. Side

Syntax:

- identifier: "wall side"
- arg = –1,0,1 (kind: integer)
  –1→ bottom side
  0 → both sides
  1 → top side

Examples:

```
-1  ! wall side # Enable a wall at the bottom side
 0  ! wall side # Enable walls at both the bottom and top side
```

Description:

This command allows the used to choose the side(s) of the wall(s).

Default:

0

Restrictions

Presently, the user cannot assign different potentials for each side.

4.7.9. Position of the hard-sphere wall

Syntax:

- identifier: "wall pos set"

- arg = distance of the hard sphere wall from the solid, h_{HS} (kind: positive real, units: Å)

Examples:

```
5 ! wall pos set # Set the reflective wall h_HS = 5 Å from the surface
```

Description:

Sets the coordinates of the hard-sphere wall. Setting its position to h_{HS} = 0 Å should be avoided when using the Hamaker potential or any other potential with a strongly repulsive term, since this may affect convergence.

Default:

h_{HS} = 0 Å

Restrictions

The hard-sphere wall must be located below the grafting points.

4.7.10. Automatic calibration of the position of the hard-sphere wall

Syntax:

- identifier: "wall pos auto"
- arg = target polymer-solid interaction (kind: positive real, units: k_BT)

Examples:

```
5.0 ! wall pos auto # set the maximum interaction u_max, for the hard-sphere wall.
```

Description:

Activates an automatic recalibration scheme that sets the position of the hard-sphere wall at a distance where the polymer-solid interaction becomes u_{max}.

Default:

none

4.8. Field and convergence parameters

4.8.1. Number of iterations for convergence

Syntax:

- identifiers: "field iterations"
- arg = number of iterations (kind: positive integer)

Examples:

```
100 ! field iterations # number of field iterations
```

Description:

Sets the number of field iterations

Default:

500000

4.8.2. Field initialization

Syntax:

- identifiers: "field read"
- arg = True/False
  True → Read field from the binary file in.field.bin
  False → initialize field to zero everywhere across the domain, $\mathbf{w}'_{ifc,init} = \mathbf{0}$

Examples:

```
True  ! field read # Read field from the binary file in.field.in
False ! initialize field to zero everywhere across the domain,
```

Description:

Choose between reading the field or initializing it to zero

Default:

In case the "`field read`" flag is set to `False`, the field is initialized to zero across the domain.

By setting the "`field read`" flag to `True`, the code will attempt to read the field from the file `in.field.bin`.

Restrictions

The user must make sure that a valid binary field file is located at the run directory. In case the parser is unable to find the file it will issue a relevant error message and terminate the program.

4.8.3. Maximum field error for convergence

Syntax:

- identifiers: "field max_error"
- arg = maximum error of the field, $\Delta w_{\mathrm{ifc}}^{\prime \max}$ (kind: positive real, units: $k_B T$)

Examples:

```
1.e-7   ! field max_error # the tolerance error from the maximum field for convergence
                          is set to w_max = 10^-7 k_B T.
```

Description:

Set the field tolerance for convergence.

Default:

$\Delta w_{\mathrm{ifc}}^{\prime \mathrm{tol}} = 0.0 k_{\mathrm{B}} T$

4.8.4. Field mixing fraction

Syntax:

- identifiers: "field mixing_fraction"
- arg = maximum error of the field, $\Delta w_{\mathrm{ifc}}^{\prime \mathrm{mix}}$ (kind: positive real)

Examples:

```
0.0001   ! field mixing_fraction : the field mixing fraction is set to wmix = 0.0001 kBT.
```

Description:

Set the field mixing fraction. The user must set it to allow enough value that prevents the iterative scheme from diverging. For more details see ref (Supplementary Information Section S7).

Default:

None

4.8.5. Contour type discretization

Syntax:

- identifiers: "discret contour"
- arg = 0 or 1 (kind: positive integer)
  0 → uniform
  1 → nonuniform

Examples:

```
0 ! discret contour # the chain contour discretization is set to uniform
1 ! discret contour # the chain contour discretization is set to nonuniform
```

Description:

Choose the chain contour discretization scheme

Default:

0: uniform

4.8.6. Integration method

Syntax:

- identifiers: "integ contour"

- arg = 0 or 1 (kind: positive integer)
  0 → Rectangle integration
  1 → Simpson's Rule

Examples:

```
0 ! integ contour # the rectangle integration method is chosen
1 ! integ contour # Simpson's integration method is chosen
```

Description:

Choose the chain contour integration scheme

Default:

1: Simpson rule

4.9. Solvers and Boundary Conditions

4.9.1. Edwards solver

Syntax:

- identifiers: "edwards solver"
- arg = 0 or 1 (kind: positive integer)
  0 → implicit scheme
  1 → semi-implicit scheme

Examples:

```
0   ! edwards solver # the implicit scheme is chosen
```

Description:

Semi-implicit method

In several works [6–8], the Crank-Nicholson contour discretization is implemented, which is a semi-implicit scheme (aka *central differences*), in that the unknown solution, q_h^N, is expressed in our implementation by means of a *central differences* scheme, averaged between two successive contour points, N and $N+\Delta N$, as shown in the following eq S9.

$$\frac{\partial^2 q}{\partial h^2} = \frac{1}{2}\frac{{q_{h+1}}^{N+1} - 2{q_h}^{N+1} + {q_{h-1}}^{N+1}}{\Delta h^2} + \frac{1}{2}\frac{{q_{h+1}}^{N} - 2{q_h}^{N} + {q_{h-1}}^{N}}{\Delta h^2} \qquad \text{S9}$$

while the first derivative of the solution, q, with respect to the contour variable N is given by eq S10.

$$\frac{\partial q}{\partial N} = \frac{{q_h}^{N+1} - {q_h}^{N}}{\Delta N} \qquad \text{S10}$$

Therefore, the matrix form of the Edwards partial differential equation becomes as follows:

$$-D{q_{h-1}}^{N+1} + \left(1 + 2D + \frac{\Delta N \beta w_{\mathrm{ifc},h}}{2}\right){q_h}^{N+1} - D{q_{h+1}}^{N+1} = D{q_{h-1}}^{N} + \left(1 - 2D - \frac{\Delta N \beta w_{\mathrm{ifc},h}}{2}\right){q_h}^{N} + D{q_{h+1}}^{N} \qquad \text{S11}$$

where $D = \dfrac{{R_{\mathrm{G},c}}^2 \Delta N}{2N_c \Delta h^2}$. In matrix-vector notation, eq S11 can be written as presented in the following eq S12.

$$\left(\mathbf{I} - D\mathbf{T} + \frac{\Delta N}{2}\mathbf{W}\right)\mathbf{q}^{N+1} = \left(\mathbf{I} + D\mathbf{T} - \frac{\Delta N}{2}\mathbf{W}\right)\mathbf{q}^{N} \qquad \text{S12}$$

where **I** is the identity matrix and we have defined the tridiagonal matrix **T** and the diagonal matrix **W**, as shown below:

$$\mathbf{T} = \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & .. & .. & .. & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix} \quad \text{S13}$$

$$\mathbf{W} = \begin{bmatrix} w(\Delta x) & & & & & \\ & w(2\Delta x) & & & & \\ & & w(3\Delta x) & & & \\ & & & .. & & \\ & & & & w(L_x - 1) & \\ & & & & & w(L_x) \end{bmatrix} \quad \text{S14}$$

We observe that in the semi-implicit scheme, the right hand side vector is the solution vector at the previous contour-step but weighted with the matrix: $\mathbf{I} + D\mathbf{T} - \frac{\Delta N}{2}\mathbf{W}$.

Implicit method

A more stable, but computationally more demanding, way to solve the time-dependent partial differential equation is to express the unknown solution, q_h^N, in terms of the next contour step, N+ΔN, according to the eq S15 presented below.

$$\frac{\partial^2 q}{\partial h^2} = \frac{{q_{h+1}}^{N+1} - 2{q_h}^{N+1} + {q_{h-1}}^{N+1}}{\Delta h^2} \quad \text{S15}$$

In this case, a linear system of equations needs to be solved to determine the solution at each contour step, but this implicit contour stepping method (aka *backward differences*) allows for larger contour steps without reaching the numerical stability limits. The first derivative of the chain contour is still given by eq S10.

Adopting this discretization scheme, we end up with the following matrix-form of the partial differential equation to be solved:

$$-2D{q_{h-1}}^{N+1} + \left(1 + 4D + \Delta N \beta w_{\text{ifc},h}\right){q_h}^{N+1} - 2D{q_{h+1}}^{N+1} = {q_h}^N \quad \text{S16}$$

where again the diffusion coefficient is given by the expression, $D = \frac{{R_{\text{G},c}}^2 \Delta N}{2N_c \Delta h^2}$. In matrix-vector notation, eq S16 is written as:

$$\left(\mathbf{I} - 2D\mathbf{T} + \Delta N\mathbf{W}\right)\mathbf{q}^{N+1} = \mathbf{q}^N \quad \text{S17}$$

where **I** is the identity matrix and the matrices **T** and **W** are again those presented in eq S13-S14. In contrast to the semi-implicit scheme developed in the previous section, in the implicit one, the right hand side is just the solution vector of the previous contour-step.

Default:

0: implicit scheme

4.9.2. Setting up the boundary conditions of matrix and grafted chains

Syntax:

- identifiers: "boundary_condition lo matrix", "boundary_condition hi matrix", "boundary_condition lo grafted" and "boundary_condition hi grafted"
- arg = –1,0,1,2 (kind: integer)
  –1→ $\nabla_r q_{\text{m}}(r_{\text{lo/hi}}, N) = 0$

$0 \rightarrow q_{\mathrm{m}}(r_{\mathrm{lo/hi}}, N) = 0$

$1 \rightarrow q_{\mathrm{m}}(r_{\mathrm{lo/hi}}, N) = 1$

$2 \rightarrow$ periodic BC

Examples:

```
0 ! boundary_condition lo matrix
1 ! boundary_condition hi matrix
0 ! boundary_condition lo grafted
0 ! boundary_condition hi grafted
2 ! boundary_condition hi matrix
```

Description:

Dirichlet-Dirichlet system

$$\begin{bmatrix} 1 & 0 & & & & 0 \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & .. & .. & .. & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ .. \\ q_{n-1} \\ q_n \end{bmatrix} = \begin{bmatrix} R_1^{\mathrm{DIR}} \\ R_2 \\ R_3 \\ .. \\ R_{n-1} \\ R_n^{\mathrm{DIR}} \end{bmatrix} \qquad \text{S18}$$

Sets the boundary conditions at the polymer/solid and polymer/vacuum ("0": $q_{\mathrm{m}}(r_{\mathrm{lo/hi}}, N) = 0$), and the polymer/polymer ("1": $q_{\mathrm{m}}(r_{\mathrm{lo/hi}}, N) = 1$ or "–1": $\nabla_r q_{\mathrm{m}}(r_{\mathrm{lo/hi}}, N) = 0$) interfaces. In the above example for a solid-polymer interface, all chains experience *Dirichlet* BC, $q_{\mathrm{m}}(r_{\mathrm{lo/hi}}, N) = 0$, at the solid surfaces. In the where *Neumann* boundary conditions are imposed on the bottom side to mathematically represent the matrix chains of the bulk polymer, $\nabla_r q_{\mathrm{m}}(r_{\mathrm{lo}}, N) = 0$, then the matrices **I** and **W** remain the same, but the matrix **T** for the implicit scheme is modified as follows:

$$\mathbf{T} = \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & .. & .. & .. & \\ & & & 1 & -2 & 1 \\ & & & & 2 & -2 \end{bmatrix} \qquad \text{S19}$$

While for the semi-implicit scheme, both matrix **T** and the right-hand side vector **r** are modified as presented below:

$$\mathbf{T} = \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & .. & .. & .. & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -1 \end{bmatrix} \qquad \text{S20}$$

Default:

None

Restrictions

Systems with periodic boundary conditions cannot be solved with Thomas algorithm. Instead, the user should choose the Gauss elimination method (see section 4.9.3).

4.9.3. Linear solvers

Syntax:

- identifiers: "linear solver"
- arg = 0 or 1 (kind: positive integer)
  0 → Thomas algorithm for tridiagonal matrices
  1 → Gauss elimination method

Examples:

```
0   ! linear solver # Thomas algorithm
1   ! linear solver # Gauss elimination method
```

Description:

Unless the user required to solve the system with periodic boundary conditions the Thomas algorithm should be preferred, since it is computationally inexpensive.

Default:

0

4.10. Exports frequency

4.10.1. Thermodynamics

Syntax:

- identifiers: "thermo every"
- arg = export frequency of thermodynamics (kind: positive integer)

Examples:

```
1000 ! thermo every # information regarding the thermodynamics is
                      exported every 1000 steps
```

Description:

Set the frequency export for the thermodynamic quantities of the system.

Default:

1000

4.10.2. Binary field

Syntax:

- identifiers: "field every"
- arg = export frequency of binary field (kind: positive integer)

Examples:

```
1000 ! field every # output a binary field file every 1000 steps
```

Description:

Set the frequency export for the binary field files.

Default:

1000

4.10.3. Computes

Syntax:

- identifiers: "compute every"
- arg = export frequency of computes (kind: positive integer)

Examples:

```
1000 ! compute every # compute and export every 1000 steps
```

Description:

Sets the computing frequency of the thermodynamic and structural properties of the system.

Default:

50000

4.11. Computes

4.11.1. Density profiles

Syntax:

- identifiers: “export phi”
- arg = True/False (kind: logical)
  True → export
  False → do not export

Examples:

```
True ! export phi # compute and export the phi profiles
```

Description:

Activate/deactivate the computation of the density profiles.

The density profiles of matrix and grafted chains are computed by the following equations respectively:

$$\varphi_{\mathrm{m}}(\mathbf{r}) = \frac{1}{N_{\mathrm{m}}} \int_0^{N_{\mathrm{m}}} \mathrm{d}N \; q_{\mathrm{m}}(\mathbf{r}, N) \; q_{\mathrm{m}}(\mathbf{r}, N_{\mathrm{m}} - N)$$

$$\varphi_{\mathrm{g}^{\mp}}(\mathbf{r}) = \frac{1}{N_{\mathrm{g}^{\mp}}} \int_0^{N_{\mathrm{g}^{\mp}}} \mathrm{d}N \; q_{\mathrm{g}^{\mp}}(\mathbf{r}, N) \; q_{\mathrm{m}}(\mathbf{r}, N_{\mathrm{g}^{\mp}} - N) \qquad \text{S21}$$

The density profiles are exported in the `o.phi` output file by the structure of the following columns:

```
r    phi_matrix    phi_gr_lo    phi_gr_hi    phi_tot
```

Default:

True

4.11.2. Field

Syntax:

- identifiers: “export field”
- arg = True/False (kind: logical)
  True → export
  False → do not export

Examples:

```
True ! export field # export the profiles of the field
```

Description:

Activate/deactivate the export of ascii field files.

The field is exported in the `o.field` output file by the structure of the following columns:

```
r    wa_old    wa_new
```

Default:

True

4.11.3. Restricted partition function

Syntax:

- identifiers: “export q”
- arg = True/False (kind: logical)
  True → export
  False → do not export

Examples:

```
True ! export q # compute and export the profiles of the partition function
```

Description:

Activate/deactivate the export of the restricted partition function of matrix and grafted chains.

The restricted partition functions of matrix and grafted chains are exported in the o.q_matrix and o.q_gra output files, respectively, by the structure of the following columns:

r q_0 q_1 q_2 q_{N-1} .. q_N

Default:

True

4.11.4. Chains/area

Syntax:

- identifiers: "export shape"
- arg = True/False (kind: logical)
  True → export
  False → do not export

Examples:

```
True ! export shape # compute and export the chains/area profiles
```

Description:

Activate/deactivate the export of the chains/area computations

The number of chains per area of either matrix or grafted chains is calculated via the following equations:

$$p_{\mathrm{int},c}(h_0) = 1 - \frac{\int_{\mathcal{R}} q_{c,h_0}^{\mathrm{shape}}(\mathbf{r}, N_c)\mathrm{d}\mathbf{r}}{\int_{\mathcal{R}} q_c(\mathbf{r}, N_c)\mathrm{d}\mathbf{r}}$$

$$n_{\mathrm{ch},c}(h_0) = p_{\mathrm{int},c}(h_0)\frac{1}{S_{h_0}}\frac{1}{N_c}\int_{\mathcal{R}} \rho_c(\mathbf{r})\mathrm{d}\mathbf{r}$$

The number of chains per area of matrix and grafted chains is exported in the `o.chainshape_matrix` and `o.chainshape_gra` output files, respectively, by the structure of the following columns:

Z z_Rg phi pcross phi/n_shape q1/p_cross n_shape

Default:

True

4.11.5. Density profiles of adsorbed/free matrix segments

Syntax:

- identifiers: "export ads_free"
- arg = True/False (kind: logical)
  True → export
  False → do not export

Examples:

```
True ! export ads_free # compute and export the density profiles of
                         adsorbed and free chains
```

Description:

Activate/deactivate the export of the profiles of adsorbed and free chains.

The density profiles are exported in the `o.chain_states_[chain kind],` output files, which are structured as follows:

```
r          distance from lo bound)
phi_c      profile of kind-c chains
f          free chains
```

```
bridge     bridges
!a-        not adsorbed to lo surface
a-         adsorbed to lo surface
afull-     fully adsorbed to lo surface
apart-     partially adsorbed to lo surface
loop_f-    loop outside the lo surface
tail_f-    tails outside the lo surface
tail_a-    tails inside the lo surface
!a+        not adsorbed to hi surface
a+         adsorbed to hi surface
afull+     fully adsorbed to hi surface
apart+     partially adsorbed to hi surface
loop_f+    loop outside the hi surface
tail_f+    tails outside the hi surface
tail_a     tails inside the hi surface
```

Default:

True

4.11.6. Brush thickness

Syntax:

- identifiers: "export brush"
- arg = True/False
  True → export
  False → do not export

Examples:

```
True ! export brush # compute and export the brush thickness
```

Description:

Activate/deactivate the computation of brush thickness.

The thickness of the brush is estimated by means of the $\left\langle h_{\mathrm{g}}{}^{2}\right\rangle^{1/2}$ and $h_{99\%}$ measures, which are calculated by the following equations, respectively:

$$\left\langle h_{\mathrm{g}}{}^{2}\right\rangle^{1/2}=\left[\frac{\int_{\mathcal{R}}\mathrm{d}\mathbf{r}\,\left[h(\mathbf{r})\right]^{2}\rho_{\mathrm{g}}(\mathbf{r})}{\int_{\mathcal{R}}\mathrm{d}\mathbf{r}\rho_{\mathrm{g}}(\mathbf{r})}\right]^{1/2} \qquad \text{S22}$$

$$\int_{\mathcal{R}_{99\%}}\mathrm{d}\mathbf{r}\rho_{\mathrm{g}}(\mathbf{r})=0.99N_{g}n_{\mathrm{g}} \qquad \text{S23}$$

The thickness of the brush (grafted chains) is exported in the `o.brush_thickness_gra` output file.

Default:

True

4.11.7. Density profiles of individual segments

Syntax:

- identifiers: “export phi_seg”
- arg = id of specified chain segment (kind: integer)

Examples:

```
 0 ! export phi_seg # export the density profile of the first segment.
-1 ! export phi_seg # export the density profile of all segments
```

Description:

Activate/deactivate the computation and export of the profiles of end- and middle segments.

The segmental density profiles are exported in the `o.phi_segs_[chain kind]` files.

Default:

0

5. Master table

The table below includes the syntax and value range of the commands issued by the input files.

type	flag identifier	value(s)	default value	comment	units
system setup	! domain geometry	0 / 1	none	cylindrical /spherical	-
	! domain lx	positive real	none	domain length	Å
	! domain dx	positive real	none	domain discretization (Δh)	Å
	! sphere_radius	positive real	none	sphere_radius	Å
	! system temperature	positive real	none	temperature	K
	! system pressure	positive real	0	pressure	atm
polymer parameters	! polymer C_inf	positive real	none	characteristic ratio	
	! polymer monomer_mass	positive real	none	monomer mass	g/mol
	! polymer mass_density	positive real	none	mass density	g/mol
	! polymer bond length	positive real	none	bond length	Å
equation of state	! EOS type	integer	none	EoS type (see 4.4)	-
	! EOS coeffs	list	none	EoS coefficient(s) (see 4.4)	-
square gradient	! EOS real influence parameter	positive real	0	influence parameter	J m^5/mol^2
	! EOS influence parameter	positive real	0	influence parameter	-
matrix chains	! matrix set	True/False	False	enable matrix chains	-
	! matrix chain length	positive real	none	N_m	-
	! matrix ds	positive real	none	ΔN_m	-
	! chain r_ads_lo/chain r_ads_hi	positive real	0	$h_{ads}^{\mp}$	Å
grafted lo chains	! grafted lo set	True/False	False	enable grafted lo chains	-
	! grafted lo chain_length	positive real	none	N_{g-}	-
	! grafted lo ds	positive real	none	ΔN_{g-}	-
	! grafted lo grafting_density	positive real	0	σ_{g-}	chains/Å^2
grafted hi chains	! grafted hi set	True/False	False	enable grafted hi chains	-
	! grafted hi chain_length	positive real	none	N_{g+}	-
	! grafted hi ds	positive real	none	ΔN_{g+}	-
	! grafted hi grafting_density	positive real	0	σ_{g+}	chains/Å^2
	! grafted distance from solid	positive real	Δh	h_g	Å
wall coefficients	! wall type	integer	0	wall type (see 4.7)	-
	! wall_coeffs	list	none	wall_coeff(s)	
	! wall pos auto	real	none	automatic recalibration	k_BT
	! wall pos set	positive real	0	position of the refl wall	Å
	! wall_side	-1 / 0 / 1	0	lo / both / hi	-
boundary condi-	! boundary_condition lo matrix	-1 / 0 / 1	none	Newman / dir0 / dir1	-

tions	! boundary_condition hi matrix	-1 / 0 / 1	none	Newman / dir0 / dir1	-
	! boundary condition lo grafted	-1 / 0 / 1	none	Newman / dir0 / dir1	-
	! boundary condition hi grafted	-1 / 0 / 1	none	Newman / dir0 / dir1	-
solution parameters	! field iterations	positive int	500000	number of iterations	-
	! field read	False / True	False	field initialization	
	! field max_error	positive real	0	wifc,tol	$k_B T$
	! field maxing_fraction	positive real	none	wifc,mix	-
	! edwards solver	0 / 1	0	implicit / semi-implicit	-
	! discret contour	0 / 1	0	uniform / nonuniform	-
	! integ contour	0 / 1	1	rectangle / Simpson rule	-
output	! thermo every	positive int	1000	thermo export frequency	-
	! field every	positive int	1000	field export frequency	-
	! compute every	int	50000	compute frequency	-
	! export phi	True/False	True	phi export frequency	-
	! export field	True/False	True	field export frequency	-
	! export q	True/False	True	q export frequency	-
	! export shape	True/False	True	chain/area export frequency	-
	! export ads free	True/False	True	ρ_{ads}/ρ_{free} export frequency	-
	! export brush	True/False	True	$\langle h^2 \rangle$, $h_{99\%}$ export frequency	-
	! export phi seg	–1 or int ≥ 0	True	$\rho_{c,i}$ export frequency	-

6. Demo input files

The present section includes the input files of representative examples to help the user get started. The examples are located at the `/examples` folder in the source directory.

Each example folder includes the following:

- `in.input`: file containing a the commands
- `in.field.bin`: binary file that includes the converged field of the example
- `README`: file that provides a short description about the example
- `out/` directory containing the output of the calculation

By default, the calculation is restarted from the converged field in the `in.field.bin` file. The calculation can be instead performed from scratch by changing the flag "`! field read`" from `True`, to `False`.

6.1. Vacuum-matrix interface (VM)

Description:

A planar vacuum/matrix interphase (VM), SL-SGT EoS. [1]
The system corresponds to the one in Figure 1(VM), in the main document.

Parameters:

```
Folder name:                    planar_VM_SLSGT_Nm100
Geometry:                       Planar, VM
EoS:                            SL-SGT
Chain length of m chains:       100
```

Input file:

```
 # system setup
 0                          ! domain geometry (Film)
 2.000000000e+02            ! domain lx (Angstrom)
 0.5                        ! domain dx
 5.000000000e+02            ! system temperature (K)
 0.000000000D+00            ! system pressure (atm)

 # polymer parameters
 0.985300000D+01            ! polymer C_inf
 0.520000000D+02            ! polymer monomer_mass (g/mol)
 0.953000000D+00            ! polymer mass_density (g/cm3)
 1.540000000D+00            ! polymer bond_length (Angstrom)

 # simulation parameters
 1000000                    ! field iterations
 True                       ! field read
 0.100000000D-06            ! field max_error (J/k_BT)
 0.000850000e-00            ! field mixing_fraction
 0                          ! discret contour (uniform)
 0                          ! discret spatial (uniform)
 1                          ! integr contour (Simpson rule)
 0                          ! integr spatial (Rectangle rule)

 # matrix chains
 True                       ! matrix set
 100.0                      ! matrix chain_length
 0.25                       ! matrix ds

 # boundary conditions
 0                          ! boundary_condition lo matrix (Dirichlet q=0)
 1                          ! boundary_condition hi matrix (Dirichlet q=1)

 # equation of state
 1                          ! EOS type (Sanchez-Lacombe)
 1105.0 735.0 3.57D+08      ! EOS coeffs (rho_star,T_star, P_star)
 0.550000000D+00            ! EOS influence_parameter (reduced units)
```

6.2. Solid-polymer interface: square well potential

Description:

A planar solid/matrix interface (SM), with the presence of the HFD EoS and a square well potential. [8] The system corresponds to the one in Figure 3a(θ = 45.3°) in the main document.

Parameters:

```
Folder name:                    planar_SM_HFD_Nm100_sqwell_kT1.65
Geometry:                       Planar, SM
EoS:                            HFD
Chain length of m chains:       100
Polymer-solid potential:        square well
```

Input file:

```
# system setup
0                             ! domain geometry (Film)
0.500000000e+02               ! domain lx (Angstrom)
0.5                           ! domain dx
0.450000000D+03               ! system temperature (K)
0.000000000D+00               ! system pressure (atm)

# polymer parameters
0.826563810E+01               ! polymer C_inf
0.140000000D+02               ! polymer monomer_mass (g/mol)
0.766000000D+00               ! polymer mass_density (g/cm3)
1.540000000D+00               ! polymer bond_length (Angstrom)

# simulation parameters
10000000                      ! field iterations
True                          ! field read
0.100000000D-07               ! field max_error (J/k_BT)
0.001000e-00                  ! field mixing_fraction
0                             ! discret contour (uniform)
0                             ! discret spatial (uniform)
1                             ! integr contour (Simpson rule)
0                             ! integr spatial (Rectangle rule)

# computes
False                         ! export q
False                         ! export ads_free
False                         ! export shape

# wall parameters
2                             ! wall type (square well)
6.500000000E+00 -1.025131527  ! wall coeffs
2.000000000D+00               ! wall pos set
-1                            ! wall side (lo)

# matrix chains
True                          ! matrix set
100.0                         ! matrix chain_length
0.25                          ! matrix ds
1.280000000D+01               ! matrix r_adsorbed (Angstrom)

# boundary conditions
0                             ! boundary_condition lo matrix (Dirichlet q=0)
-1                            ! boundary_condition hi matrix (Neumann dq/dr=0)

# equation of state
0                             ! EOS type (Helfand)
1.430000000D-09               ! EOS coeffs (kappa_T)
```

6.3. Solid-polymer interface (SM): tabulated potential

Description:

A planar solid/matrix interface (SM), with the tabulated potential in Figure 4a,c in the main document, that reproduces the profile of PE/graphite interface from molecular dynamics . The tabulated potential is read from the file `in.table`.

Parameters:

```
Folder name:                    planar_SM_HFD_Nm100_table
Geometry:                       Planar, SM
```

```
EoS:                            HFD
Chain length of m chains:       100
Polymer-solid potential:        table
```

Input file:

```
# system setup
0                           ! domain geometry (Film)
0.300000000e+02             ! domain lx (Angstrom)
0.5                         ! domain dx
0.450000000D+03             ! system temperature (K)
0.000000000D+00             ! system pressure (atm)

# polymer parameters
0.826563810E+01             ! polymer C_inf
0.280000000D+02             ! polymer monomer_mass (g/mol)
0.766000000D+00             ! polymer mass_density (g/cm3)
1.540000000D+00             ! polymer bond_length (Angstrom)

# simulation parameters
10000000                    ! field iterations
True                        ! field read
0.100000000D-03             ! field max_error (J/k_BT)
0.00010e-00                 ! field mixing_fraction
0                           ! discret contour (uniform)
0                           ! discret spatial (uniform)
1                           ! integr contour (Simpson rule)
0                           ! integr spatial (uniform)

# computes
False                       ! export q
False                       ! export ads_free
False                       ! export shape

# wall parameters
10                          ! wall type (table)
0.000000000D+00             ! wall pos set
-1                          ! wall side (lo)

# matrix chains
True                        ! matrix set
100.0                       ! matrix chain_length
0.25                        ! matrix ds
1.280000000D+01             ! matrix r_adsorbed (Angstrom)


# boundary conditions
0                           ! boundary_condition lo matrix, (Dirichlet q=0)
-1                          ! boundary_condition hi matrix, (Dirichlet q=0)

# equation of state
0                           ! EOS type (Helfand)
1.430000000D-09             ! EOS coeffs ( kappa_T)
```

6.4. Solid-matrix-solid geometry (SMS)

Description:

A planar solid/matrix/solid geometry (SMS), with the SL-SGT EoS that reproduces the adsorbed chain states in Figure 10 in the main document. The parameters are retrieved from Table 1 in ref. [5].

Parameters:

```
Folder name:                    planar_SMS_SLSGT_Nm100_ra60
Geometry:                       Planar, SMS
EoS:                            SL-SGT
Chain length of m chains:       100
Polymer-solid potential:        hybrid: Hamaker + ramp
Critical adsorption distance:   60 Angstrom
```

Input file:

```
# system setup
0                           ! domain geometry (Film
1.600000000e+02             ! domain lx (Angstrom)
0.5                         ! domain dx
5.000000000e+02             ! system temperature (K)
0.000000000D+00             ! system pressure (atm)

# polymer parameters
0.985300000D+01             ! polymer C_inf
0.520000000D+02             ! polymer monomer_mass (g/mol)
```

```
0.953000000D+00               ! polymer mass_density (g/cm3)
1.540000000D+00               ! polymer bond_length (Angstrom)

# simulation parameters
1000000                       ! field iterations
True                          ! field read
0.100000000D-06               ! field max_error (J/k_BT)
0.002000000e-00               ! field mixing_fraction
0                             ! discret contour (uniform)
0                             ! discret spatial (uniform)
1                             ! integr contour (Simpson rule)
0                             ! integr spatial (Rectangle rule)

# computes

# wall parameters
-1                                                                ! wall type (hybrid)
1 3.700000000E+00 3.000000000E+00 5.840000000D+00 6.430000000D+00 ! wall coeffs (Hamaker)
3 1.280000000E+01 -3.975042293                                    ! wall coeffs (ramp)
5.000000000D+00                                                   ! wall pos auto (k_B T)
0                                                                 ! wall side (both)

# matrix chains
True                          ! matrix set
100.0                         ! matrix chain_length
0.25                          ! matrix ds
6.000000000D+01               ! chain r_ads_hi (Angstrom)
6.000000000D+01               ! chain r_ads_lo (Angstrom)

# boundary conditions
0                             ! boundary_condition lo matrix (Dirichlet q=0)
0                             ! boundary_condition hi matrix (Dirichlet q=0)

# equation of state
1                             ! EOS type (Sanchez-Lacombe)
1105.0 735.0 3.57D+08         ! EOS coeffs (rho_star,T_star, P_star)
0.550000000D+00               ! EOS influence_parameter (reduced units)

False                         ! export q
```

6.5. Grafted-matrix-grafted geometry (GMG)

Description:

A planar grafted/matrix/grafted geometry (GMG), with the presence of the SL-SGT EoS and a hybrid wall potential. The parameters are retrieved from Table 1 in ref. [5].

The system corresponds to the one in Figure 1(GMG) in the main document.

Parameters:

```
Folder name:                    planar_GMG_SLSGT_glo0.008_Nlo50_ghi0.004_Nhi200_Nm100
Geometry:                       Planar, GMG
EoS:                            SL-SGT
Grafting density of g- chains:  0.008 (Angstrom^-2)
Grafting density of g+ chains:  0.004 (Angstrom^-2)
Chain length of g- chains:      50
Chain length of g+ chains:      200
Chain length of m chains:       100
Polymer-solid potential:        Hybrid: Hamaker + ramp
```

Input file:

```
# system setup
0                             ! domain geometry (Film)
2.000000000e+02               ! domain lx (Angstrom)
0.5                           ! domain dx
5.000000000e+02               ! system temperature (K)
0.000000000D+00               ! system pressure (atm)

# polymer parameters
0.985300000D+01               ! polymer C_inf
0.520000000D+02               ! polymer monomer_mass (g/mol)
0.953000000D+00               ! polymer mass_density (g/cm3)
1.540000000D+00               ! polymer bond_length (Angstrom)

# simulation parameters
1000000                       ! field iterations
True                          ! field read
```

```
0.100000000D-06               ! field max_error (J/k_BT)
0.000850000e-00               ! field mixing_fraction
0                             ! edwards solver
0                             ! discret contour (uniform)
0                             ! discret spatial (uniform)
1                             ! integr contour (Simpson rule)
0                             ! integr spatial (Rectangle rule)

# computes
False                         ! export q

# wall parameters
-1                                                                  ! wall type (Hybrid)
1 3.700000000E+00 3.000000000E+00 5.840000000D+00 6.430000000D+00 ! wall coeffs (Hamaker)
3 1.280000000E+01 -3.975042293                                      ! wall coeffs (ramp)
5.000000000D+00               ! wall pos auto
0                             ! wall side (both)

# matrix chains
True                          ! matrix set
100.0                         ! matrix chain_length
0.25                          ! matrix ds
1.280000000D+01               ! matrix r_adsorbed (Angstrom)

# grafted lo chains
True                          ! grafted lo set
50.0                          ! grafted lo chain_length
0.25                          ! grafted lo ds
8.000000000e-03               ! grafted lo grafting_density (chains/Angstrom^2)

# grafted hi chains
True                          ! grafted hi set
200.0                         ! grafted hi chain_length
0.25                          ! grafted hi ds
4.000000000e-03               ! grafted hi grafting_density (chains/Angstrom^2)

# boundary conditions
0                             ! boundary_condition lo matrix  (Dirichlet q=0)
0                             ! boundary_condition hi matrix  (Dirichlet q=0)
0                             ! boundary_condition lo grafted (Dirichlet q=0)
0                             ! boundary_condition hi grafted (Dirichlet q=0)

# equation of state
1                             ! EOS type   (SL)
1105.0 735.0 3.57D+08         ! EOS coeffs (rho_star, T_star, P_star)
0.550000000D+00               ! EOS influence_parameter (reduced units)
```

6.6. Grafted-matrix-vacuum geometry (GMV)

Description:

A planar grafted/matrix/vacuum geometry (SMV), with the presence of the SL-SGT EoS and a hybrid wall potential. The parameters are retrieved from Table 1 in ref. [5].

The system corresponds to the one in Figures 6, 7, 8 and 11, in the main document.

Parameters:

```
Folder name:                    planar_GMV_SLSGT_glo0.004_Nlo50_Nm100
Geometry:                       Planar, GMV
EoS:                            SL-SGT
Grafting density of g- chains:  0.004 (Angstrom^-2)
Chain length of g- chains:      50
Chain length of m chains:       100
Polymer-solid potential:        Hybrid: Hamaker + ramp
```

Input file:

```
# system setup
0                             ! domain geometry (Film)
1.000000000e+02               ! domain lx (Angstrom)
0.5                           ! domain dx
5.000000000e+02               ! system temperature (K)
0.000000000D+00               ! system pressure (atm)

# polymer parameters
0.985300000D+01               ! polymer C_inf
0.520000000D+02               ! polymer monomer_mass (g/mol)
0.953000000D+00               ! polymer mass_density (g/cm3)
1.540000000D+00               ! polymer bond_length (Angstrom)

# simulation parameters
```

```
1000000                       ! field iterations
True                          ! field read
0.100000000D-06               ! field max_error (J/k_BT)
0.002000000e-00               ! field mixing_fraction
0                             ! discret contour (uniform)
0                             ! discret spatial (uniform)
1                             ! integr contour (Simpson rule)
0                             ! integr spatial (Rectangle rule)

# wall parameters
-1                                                                    ! wall type (hybrid)
1 3.700000000E+00 3.000000000E+00 5.840000000D+00 6.430000000D+00 ! wall coeffs (Hamaker)
3 1.280000000E+01 -3.975042293                                        ! wall coeffs (ramp)
5.000000000D+00                                                       ! wall pos auto (k_B T)
-1                                                                    ! wall side (lo)

# matrix chains
True                          ! matrix set
100.0                         ! matrix chain_length
0.25                          ! matrix ds
1.280000000D+01               ! matrix r_adsorbed (Angstrom)

# grafted lo chains
True                          ! grafted lo set
50.0                          ! grafted lo chain_length
0.25                          ! grafted lo ds
4.000000000e-03               ! grafted lo grafting_density (chains/Angstrom^2)

0.000000000D+00               ! grafted distance_from_solid (Angstrom)

# boundary conditions
0                             ! boundary_condition lo matrix  (Dirichlet q=0)
0                             ! boundary_condition hi matrix  (Dirichlet q=0)
0                             ! boundary_condition lo grafted (Dirichlet q=0)
0                             ! boundary_condition hi grafted (Dirichlet q=0)

# equation of state
1                             ! EOS type (Sanchez-Lacombe)
1105.0 735.0 3.57D+08         ! EOS coeffs (rho_star, T_star, P_star)
0.550000000D+00               ! EOS influence_parameter (reduced units)

True                          ! export phi_seg
-1                            ! export set phi_seg (export phi_seg for all segments)
False                         ! export q
```

6.7. Polymer-grafted nanoparticle in vacuum

Description:

A polymer-grafted nanoparticle embedded in a vacuum phase (spherical grafted-matrix interface, GV) with SL-SGT EoS.

The system corresponds to the one in Figure 2(GV), in the main document.

Parameters:

```
Folder name:                    spherical_GV_SLSGT_Nm100
Geometry:                       spherical, GV
EoS:                            SL-SGT
Grafting density of g- chains:  0.008 (Angstrom^-2)
Chain length of g- chains:      50
Polymer-solid potential:        Hybrid: Hamaker + ramp
```

Input file:

```
# system setup
1                             ! domain geometry, (0: Film, 1: sphere)
2.000000000e+02               ! domain lx, (Angstrom)
0.5                           ! domain dx
2.000000000e+01               ! domain sphere_radius, (Angstrom)
5.000000000e+02               ! system temperature, (K)
0.000000000D+00               ! system pressure, (atm)

# polymer parameters
0.985300000D+01               ! polymer C_inf
0.520000000D+02               ! polymer monomer_mass, (g/mol)
0.953000000D+00               ! polymer mass_density, (g/cm3)
1.540000000D+00               ! polymer bond_length, (Angstrom)
```

```
# simulation parameters
1000000                         ! field iterations
True                            ! field read, (0: no, 1: yes)
0.100000000D-06                 ! field max_error, (J/k_BT)
0.000850000e-00                 ! field mixing_fraction
0                               ! edwards solver (implicit)
0                               ! discret contour (uniform)
0                               ! discret spatial (uniform)
1                               ! integr contour (Simpson rule)
0                               ! integr spatial (Rectangle rule)

# computes
False                           ! export q
False                           ! export ads_free
False                           ! export shape

# wall parameters
-1                                                                  ! wall type (hybrid)
1 3.700000000E+00 3.000000000E+00 5.840000000D+00 6.430000000D+00 ! wall coeffs (Hamaker)
3 1.280000000E+01 -3.975042293                                      ! wall coeffs (ramp)
5.000000000D+00                                                     ! wall pos auto (k_B T)
-1                                                                  ! wall side (lo)

# grafted lo chains
True                            ! grafted lo set
50.0                            ! grafted lo chain_length
0.25                            ! grafted lo ds
8.000000000e-03                 ! grafted lo grafting_density (chains/Angstrom^2)

0.000000000D+00                 ! grafted distance_from_solid, (Angstrom)

# boundary conditions
0                               ! boundary_condition lo grafted (Dirichlet q=0)
0                               ! boundary_condition hi grafted (Dirichlet q=0)

# equation of state
1                               ! EOS type (Sanchez-Lacombe)
1105.0 735.0 3.57D+08           ! EOS coeffs (rho_star, T_star, P_star)
0.550000000D+00                 ! EOS influence_parameter (reduced units)
```

6.8. Polymer-grafted nanoparticle in melt

Description:

A polymer-grafted nanoparticle embedded in a matrix phase (spherical grafted-matrix interface, GM) with SL-SGT EoS. If the ramp potential is deactivated, then this input file reproduces the results presented in ref. [4].

The system corresponds to the one in Figure 2(GM), in the main document.

Parameters:

```
Folder name:                    spherical_GM_SLSGT_glo0.008_Nlo50_Nm100
Geometry:                       spherical, GM
EoS:                            SL-SGT
Grafting density of g- chains:  0.008 (Angstrom^-2)
Chain length of g- chains:      50
Chain length of m chains:       100
Polymer-solid potential:        Hybrid: Hamaker + ramp
```

Input file:

```
# system setup
1                               ! domain geometry (sphere)
2.000000000e+02                 ! domain lx (Angstrom)
0.5                             ! domain dx
2.000000000e+01                 ! domain sphere_radius (Angstrom)
5.000000000e+02                 ! system temperature (K)
0.000000000D+00                 ! system pressure (atm)

# polymer parameters
0.985300000D+01                 ! polymer C_inf
0.520000000D+02                 ! polymer monomer_mass (g/mol)
0.953000000D+00                 ! polymer mass_density (g/cm3)
1.540000000D+00                 ! polymer bond_length (Angstrom)
```

```
# simulation parameters
1000000                       ! field iterations
True                          ! field read
0.100000000D-06               ! field max_error (J/k_BT)
0.000850000e-00               ! field mixing_fraction
0                             ! edwards solver (implicit)
0                             ! discret contour (uniform)
0                             ! discret spatial (uniform)
1                             ! integr contour (Simpson rule)
0                             ! integr spatial (Rectangle rule)

# computes
False                         ! export q
False                         ! export ads_free
False                         ! export shape

# wall parameters
-1                                                                  ! wall type (hybrid)
1 3.700000000E+00 3.000000000E+00 5.840000000D+00 6.430000000D+00 ! wall coeffs (Hamaker)
3 1.280000000E+01 -3.975042293                                      ! wall coeffs (ramp)
5.000000000D+00                                                     ! wall pos auto (k_B T)
-1                                                                  ! wall side (lo)

# matrix chains
True                          ! matrix set
100.0                         ! matrix chain_length
0.25                          ! matrix ds
1.280000000D+01               ! matrix r_adsorbed (Angstrom)

# grafted lo chains
True                          ! grafted lo set
50.0                          ! grafted lo chain_length
0.25                          ! grafted lo ds
8.000000000e-03               ! grafted lo grafting_density (chains/Angstrom^2)

0.000000000D+00               ! grafted distance_from_solid, (Angstrom)

# boundary conditions
0                             ! boundary_condition lo matrix (Dirichlet q=0)
1                             ! boundary_condition hi matrix (Neumann dq/dr=0)
0                             ! boundary_condition lo grafted (Dirichlet q=0)
0                             ! boundary_condition hi grafted (Dirichlet q=0)

# equation of state
1                             ! EOS type (Sanchez-Lacombe)
1105.0 735.0 3.57D+08         ! EOS coeffs (rho_star, T_star, P_star)
0.550000000D+00               ! EOS influence_parameter (reduced units)
```

7. References

1. Lakkas, A.T.; Sgouros, A.P.; Theodorou, D.N. Self-Consistent Field Theory Coupled with Square Gradient Theory of Free Surfaces of Molten Polymers and Compared to Atomistic Simulations and Experiment. *Macromolecules* **2019**, *52*, 5337–5356, doi:10.1021/acs.macromol.9b00795.

2. Hamaker, H.C. The London—van der Waals attraction between spherical particles. *Physica* **1937**, *4*, 1058–1072, doi:10.1016/S0031-8914(37)80203-7.

3. Sun, W. Interaction forces between a spherical nanoparticle and a flat surface. *Phys. Chem. Chem. Phys.* **2014**, *16*, 5846–5854, doi:10.1039/c3cp55082f.

4. Lakkas, A.T.; Sgouros, A.P.; Revelas, C.J.; Theodorou, D.N. Structure and Thermodynamics of Grafted Silica/Polystyrene Nanocomposites Investigated Through Self-Consistent Field Theory. *Soft Matter* **2021**.

5. Sgouros, A.P.; Revelas, C.J.; Lakkas, A.T.; Theodorou, D.N. Potential of Mean Force between Bare or Grafted Silica/Polystyrene Surfaces from Self-Consistent Field Theory. *Polymers (Basel).* **2021**, *13*, doi:10.3390/polym13081197.

6. Drolet, F.; Fredrickson, G.H. Combinatorial screening of complex block copolymer assembly with self-consistent field theory. *Phys. Rev. Lett.* **1999**, *83*, 4317–4320, doi:10.1103/PhysRevLett.83.4317.

7. Theodorou, D.N.; Vogiatzis, G.G.; Kritikos, G. Self-consistent-field study of adsorption and desorption kinetics of polyethylene melts on graphite and comparison with atomistic simulations. *Macromolecules* **2014**, *47*, 6964–6981, doi:10.1021/ma501454t.

8. Daoulas, K.C.; Theodorou, D.N.; Harmandaris, V.A.; Karayiannis, N.C.; Mavrantzas, V.G. Self-consistent-field study of compressible semiflexible melts adsorbed on a solid substrate and comparison with atomistic simulations. *Macromolecules* **2005**, *38*, 7134–7149, doi:10.1021/ma050218b.