density functional simulator for block copolymers
version 0.2.6
27 October 2018

Takashi Uneyama

Table of Contents

1 Introduction 1

2 Installation of drops 3
- 2.1 To Download the Latest Version of drops 3
- 2.2 Install to Microsoft Windows by the Installer 3
- 2.3 Build and Install from the source 3
- 2.4 Build and Install as the RPM package (for Linux) 4
- 2.5 Compilation with Intel C++ Compiler (icc) 4

3 Invoking drops 5

4 Tutorial 7
- 4.1 Simple Example 7
- 4.2 Plot or Visualize Output Data 7
- 4.3 Changing Input File 7
- 4.4 Notes on Input File 10
  - 4.4.1 Boolean Variables 10
  - 4.4.2 Symmetric Matrices 11

5 Reporting Bugs 13

6 Input File Format 15
- 6.1 Simulation Condition 15
- 6.2 Input / Output Files 17
- 6.3 Multigrid Solver 18
- 6.4 Geometry of Simulation Box 18
- 6.5 geometry 18
- 6.6 Polymer Blend 19
- 6.7 Monomer Species 19
- 6.8 Polymer Species 20
- 6.9 How to Determine the Adjacency Matricies 20

7 Output File Format 23
- 7.1 Psi-Field 23
- 7.2 Density Field 23
- 7.3 Chemical Potential Field 24
- 7.4 Free Energy 24
- 7.5 Geometry 24
- 7.6 DX Output File 24

8 Utility Programs 25
8.1 Converter for DX Output Files 25
8.2 Converter for Gzipped Text Files 25

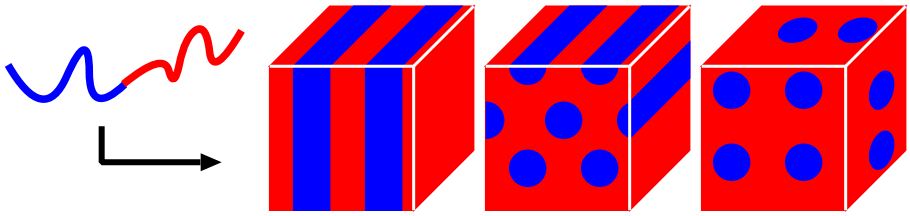
9 References 27

GNU GENERAL PUBLIC LICENSE 29
Preamble 29
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION 30
How to Apply These Terms to Your New Programs 34

Concept Index 35

1 Introduction

It is widely known that most of the mixture of polymers causes phase separation at low temperature [Introduction to Polymer Physics, Scaling Concepts in Polymer Physics]. The block copolymers (polymers which consists on chemically connected subchains of different monomer species) causes phase separation at low temperature, too, but the its phase separation behavior is different from the case of homopolymers (polymers which consists on one monomer species) [Bates-Fredrickson-1999]. While the blends of homopolymers cause the phase separation macroscopically, the block copolymers cause the microscopic, chain-length scale phase separation. The former is called as the 'macro phase separation' and the latter is called as the 'micro phase separation'.

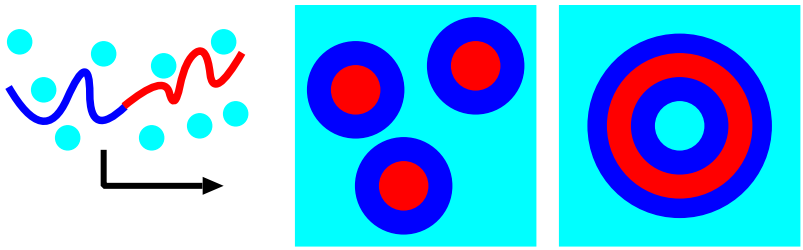


The micro phase separation takes various morphologies which depends on the structure of block copolymers or interaction of monomers. The scale of the micro phase separation structures is about the scale of polymer chains and is typically from $10nm$ - $1\mu m$.

To study the micro phase separation of block copolymers, various simulation method is proposed. The particle methods – the coarse-grained molecular dynamics (MD), the dissipative particle dynamics (DPD) [Groot-Warren-1997,Groot-Madden-1998,Groot-Madden-Tildesley-1999] – handle each polymer chains explicitly. These are good to study the dynamics of each chains, but takes too much memory and calculation time for studying the morphologies. The self consistent field (SCF) theory is widely used to study the morphologies of block copolymers [Helfand-Wasserman-1976,Helfand-Wasserman-1978,Helfand-Wasserman-1980, Matsen-Schick-1994,Matsen-Bates-1996,Fraaije-1993, Drolet-Fredrickson-1999,Fredrickson-Ganesan-Drolet-2002,Statistical Physics of Polymers: An Introduction]. It calculates the statistical weight of one polymer chain, based on the mean field theory. The SCF simulations are quite accurate and give very good results which is consistent with the experimental results quantitatively. While the SCF gives good results, it requires large memory and calculation time for large systems (for example, 3 dimensional systems or dynamics). The density functional (DF) theory is the theory which gives the free energy of the system as the functional of the subchain density fields [Joanny-Leibler-1978,de Gennes-1980,Leibler-1980,Ohta-Kawasaki-1986,Ohta-Kawasaki-1990, Kawasaki-Ohta-Kohrogui-1988,Nakazawa-Ohta-1993,Kawakatsu-1994,Ohta-Ito-1995, Bohbot-Raviv-Wang-2000,Uneyama-Doi-2005]. It requires small memory and calculation time compared with the SCF simulation while its accuracy is less than one of the SCF. The DF simulation is therefore suitable to survey the morphology of block copolymers qualitatively or to study the large systems which cannot be handled with the SCF simulation.

`drops` is a simulator for block copolymer melts and blends based on the density functional theory [Uneyama-Doi-2005]. It can handle arbitrary block copolymer systems – block

copolymers with arbitrary structure and arbitrary blend of them. The required parameter sets which describes the block copolymers are the same parameter sets as ones required by the SCF simulation. Thus one can do the simulation by `drops` just like by using the simulator based on the SCF. `drops` enables fast and efficient simulation for the micro phase separation of the block copolymers. For example, 3D structure formed by block copolymers such as the onion structure [Koizumi-Hasegawa-Hashimoto-1994,Uneyama-Doi-2005] or micellar structures (spherical micelles, cylindrical micelles and vesicles) [Disher-Eisenberg-2002,Choucair-Eisenberg-2003,Uneyama-Doi-2005a] can be simulated by `drops`.



From version 0.2.0, the dynamic simulation scheme is implemented. Thus `drops` enables the dynamics simulations as well as statics simulations for polymer blends, block copolymers or micellar systems [Uneyama-2007]. From version 0.2.3, some command line utility tools are bundled. They convert data files into other formats.

2 Installation of drops

2.1 To Download the Latest Version of drops

The latest version of drops is available at the following URL. Access the web page and download the latest version via HTTP (FTP is not supported).

http://www.ton.scphys.kyoto-u.ac.jp/~uneyama/drops.html

2.2 Install to Microsoft Windows by the Installer

The installer of drops for Microsoft Windows is now available. You can install drops just like other Windows applications, by executing the installer drops-0.2.6-win32-setup.exe. The installer is built by using Inno Setup (http://www.jrsoftware.org/isinfo.php) and works for most versions of Windows.

2.3 Build and Install from the source

You can build and install drops if the binary package of your system is not available, or if you want to customize the drops. The source package of drops is using GNU Automake and GNU Autoconf, therefore you can build and install drops just like usual free software. Note that drops requires zlib (http://www.zlib.net/), Lua (http://www.lua.org/) and FFTW3 (http://www.fftw.org/). You have to install them before build drops.

The source package is distributed as the gzipped tar archive file, thus first extract it. To extract the archive, do

```
$ zcat drops-0.2.6.tar.gz | tar xvf -
```

or if you are using GNU tar, do

```
$ tar zxvf drops-0.2.6.tar.gz
```

Then the source directory will extracted. Move to the directory drops-0.2.6.

```
$ cd drops-0.2.6
```

To build drops, do configure-make-make install just like other free software.

```
$ ./configure
$ make
$ su -
# make install
```

Now drops will be installed under /usr/local of your system. If you have an error messages and the compilation is aborted, some commands or libraries may be missing. Install the required packages and retry.

If you want to customize or tune drops,

```
$ ./configure --help
```

will help you.

2.4 Build and Install as the RPM package (for Linux)

The RPM package for your Linux system can be built from the source RPM (SRPM) package.Make sure that the headers and libraries and headers of `zlib`, `lua` and `fftw3` is already installed to your system. If they are not installed, first you have to install them (`zlib`, `zlib-devel`,`lua`,`lua-devel`,`fftw3`, and `fftw3-devel`). Of course you need standard development tools such as C compiler (`gcc`) or Make (`make`).

If you are using old system (`rpm` compatible with RedHat 7.3 or older), use the `rpm` command to build it.

```
# rpm --rebuild drops-0.2.6-1.src.rpm
```

If you are using new system (`rpm` compatible with RedHat 8.0 or newer), use `rpmbuild` instead of `rpm`.

```
# rpmbuild --rebuild drops-0.2.6-1.src.rpm
```

Now the binary RPM package for your system is stored in the directory which is shown in the output message of `rpm` or `rpmbuild`. Install it by `rpm`, for example, if you are using RedHat Linux or Fedora Core on a PC (or i386 compatible computer), like the following.

```
# rpm -Uvh /usr/src/redhat/RPMS/i386/drops-0.2.6-1.i386.rpm
```

2.5 Compilation with Intel C++ Compiler (`icc`)

You may want to compile `drops` Intel C++ Compiler (`icc`). `icc` is mostly compatible GNU C Compiler (`gcc`) and thus you can compiler `drops` with `icc` easily. But the optimization flag `-ipo` will cause troubles when compiling `drops`. Also note that the flag `-ipo` is automatically enabled if you set the optimization flag `-fast` or if you using `icc` version 9.0 or later.

There are two way to avoid troubles with the flag `-ipo`. One way is to add the flag `-ipo_obj`. This means, to run `configure` like

```
$ ./configure CC=icc CFLAGS='-O3 -ipo -ipo_obj'
```

(Here note that, this method can be used only for `icc` version 8. If you are using `icc` version 9, you should use the following method.) Another way is to use `xiar`,`xild` instead of `ar`,`ld`. In this case, the additional flag `-ipo_obj` is not needed.

```
$ ./configure CC=icc CFLAGS='-O3 -ipo'
$ make AR=xiar LD=xild
```

See the manual of Intel C++ Compiler for more information.

3 Invoking drops

The format for running the `drops` program is:

```
$ drops option ... input
```

input is the input file for `drops`. If no input file is specified, `drops` will read the default input file `dropsin.lua`.

`drops` supports the following options:

`--input=`*input*
`-i` *input*
: Read the parameters for simulation from the input file *input*. If no input file is specified, drops will read the input file named `dropsin.lua`.

`--psi=`*psi*
`-p` *psi*
: Read the initial value of the psi-field (square root of the density) from the file *psi*. The psi-field input file *psi* must be the gzipped plain text. You can create one easily by using `gzip`. By default, drops set the density field to be homogeneous.

`--external=`*external*
`-e` *external*
: Read the external force field from the file *external*. The external force input file density must be the gzipped plain text. You can create one easily by using `gzip`. By default, drops does not apply any external force to the system.

`--help`
`-h`
: Show summary of options.

`--version`
`-v`
: Show version of program.

4 Tutorial

4.1 Simple Example

Here are a simple examples how to use `drops` (the input file and some scripts for plotting / visualization can be found in the directory `examples/`). But first of all, you have to install the `drops` to your system. If `drops` is not installed to your system, see the 'Install `drops`' section. We starts from the simplest block copolymer systems, the diblock copolymer melts. The input file is distributed with the source code or binary of `drops`. To run this example, move to the directory `examples/ab_melt_1d/` and just type `drops`

```
$ drops
```

`drops` will output some information about the simulation, and starts the simulation. The simulation will end in several seconds. You can find some output files.

4.2 Plot or Visualize Output Data

The output file of `drops` is gzipped plain text and the OpenDX (http://www.opendx.org/) data format. If you have OpenDX, you can visualize it directly. The gzipped plain text is more portable and can be handled by most of the plotting / visualizing applications. For example, here we plot the output data of the previous simulation by using Gnuplot (http://www.gnuplot.info/). For Gnuplot cannot handle the gzipped text directly, we have to decompress it. There are two way to do it. The first way is to use `gunzip` and then plot the decompressed data.

```
$ gunzip phi.dat.gz
$ gnuplot
gnuplot> plot "phi.dat" using 1:2 with lines
```

This will show the density profile of the 'A' subchain. The second way is to use `zcat`.

```
$ gnuplot
gnuplot> plot "< zcat phi.dat.gz" using 1:2 with lines
```

The result is just the same as the first way.

4.3 Changing Input File

The input file is the Lua script which sets the parameters needed for the DF simulation. The following is the input file used in the previous section.

```
condition =
{
    optimize_lattice = false,

    use_multigrid_solver = false,

    dynamics_simulation = false,

    save_psi_sequential = false,
    save_density_sequential = false,
    save_chemical_potential_sequential = false,
```

```
    save_free_energy_sequential = false,
    save_geometry_sequential = false,
    save_dx_sequential = false,

    error_tolerance = 1.0e-12,
    seed = 19876,
    noise = 0.0e-2,
    initial_noise = 1.0e-3,

    phi_min = -1.0,

    iteration_max = 5000,
    interval = 500,
    omega = 0.1,
}

file =
{
    wisdom = "fftw.wisdom",

    psi_output = "psi.dat.gz",
    density_output = "phi.dat.gz",
    chemical_potential_output = "mu.dat.gz",
    free_energy_output = "fe.dat",
    geometry_output = "geometry.dat",
    dx_output = "dropsout.dx",

    psi_template = "psi.%d.dat.gz",
    density_template = "phi.%d.dat.gz",
    chemical_potential_template = "mu.%d.dat.gz",
    free_energy_template = "fe.%d.dat",
    geometry_template = "geometry.%d.dat",
    dx_template = "ppoutput.%d.dx"
}

multigrid =
{
    n_cycle = 10,
    n_pre = 5,
    n_post = 5,
    error_tolerance = 1.0e-4
}

geometry =
{
    dimension = 1,
```

```
    nx = 256,
    ny = 1,
    nz = 1,

    lx = 32,
    ly = 1,
    lz = 1
}

blend =
{
    polymer = {"AB_diblock"},
    volume_fraction = {1}
}

monomer =
{
    name = {"A", "B"},
    b = {1, 1},
    chi = {{0, 3.0},
           {0, 0  }}
}

AB_diblock =
{
    N = 40,
    f = {0.5, 0.5},
    a = {{false, true },
         {false, false}},
    monomer = {"A", "B"},
    lambda = 20
}
```

There are many parameters required by `drops`. The detail of the input file will be expressed in the section 'Input File Format'.

Here we modify this input file simply. The first example is to change the size and dimension(s) of the simulations box. This can be done by changing the `geometry` in the input file. Change the `geometry` in the input file as follows.

```
geometry =
{
    dimension = 2,

    nx = 32,
    ny = 32,
    nz = 1,

    lx = 16,
```

```
    ly = 16,
    lz = 1
}
```

`dimension` means the dimension(s) of the system. `nx,ny,nz` and `lx,ly,lz` mean the number of division and edge length for x, y, z axis. Thus the parameters shown above mean the 2 dimensional system with each edge length is 16 and divided into 32.

The second example is to change the polymers used in the simulation. This needs more complicated changes. The change will be as follows.

```
blend =
{
    polymer = {"A_homo", "B_homo"},
    volume_fraction = {0.5, 0.5}
}

A_homo =
{
    N = 10,
    f = {1},
    a = {{false}},
    monomer = {"A"},
    lambda = 0
}

B_homo =
{
    N = 10,
    f = {1},
    a = {{false}},
    monomer = {"B"},
    lambda = 0
}
```

The `blend` is changed to simulate the blend of A homopolymer / B homopolymer. The polymer species which the blend is consists on is listed in `polymer`. The volume fractions of each polymer species are specified by `volume_fraction`. The A homopolymer `A_homo` and the B homopolymer `B_homo` is defined as well (the `AB_diblock` is no longer needed and can be deleted because now it is not used). `N` is the degree of polymerization, `f` is the block ratio, `a` is the adjacency matrix for subchains and `monomer` is the monomer species.

4.4 Notes on Input File

4.4.1 Boolean Variables

There are many boolean variables (of which value is `true` or `false`) in the input file for `drops`. However it may seem verbous to write many boolean values (especially for large adjacency matrices). In such situations one can use `1` and `0` instead of `true` and `false`. `drops` automatically converts `1` and `0` into boolean values, `true` and `false`, for the boolean

variables. (Strictly speaking, number value `0` corresponds to `false` and non-zero numbers, including `1`, correspond to `true`. This is just the same as the standard C mannar.)

4.4.2 Symmetric Matrices

Adjacency matrices and Flory-Huggins chi parameter matrices are symmteric. Thus we don't need to set all the elements in these matrices. In the input file for `drops`, adjacency matrices are required to set their upper triangular part and chi parameter matrices are required to set their diagonal and uppser triangular part. `drops` automatically fill the lower critical part by copying the values of upper triangular elements. (See examples in previous sections.)

5 Reporting Bugs

Currently, the error handling routines in `drops` is not complete and therefore `drops` may suddenly stops if some input error or calculation error is caused.

If you find a bug in `drops`, please send electronic mail to uneyama@ton.scphys.kyoto-u.ac.jp. Include the version number, which you can find by running `drops --version`. Also include in your message the output that the program produced and the output you expected.

If you have other questions, comments or suggestions about `drops`, contact the author via electronic mail to uneyama@ton.scphys.kyoto-u.ac.jp. The author will try to help you out, although he may not have time to fix your problems.

6 Input File Format

In this section, the input file format for `drops` is expressed. The input file is the Lua script which sets the parameters. The parameters are set as the table variables.

6.1 Simulation Condition

The simulation condition will be set as the table `condition`. The following elements are required.

`condition.optimize_lattice`
(*boolean* or *number*)

Whether to perform the lattice optimization or not. If `condition.optimize_lattice` is set to `true`, `drops` automatically modify the size of the system `lx,ly,lz` to minimize the free energy.

`condition.use_multigrid_solver`
(*bollean* or *number*)

Whether to use the multigrid solver for Poisson equation or not. If `condition.use_multigrid_solver` is set to `true`, `drops` uses the multigrid Poisson equation. If it is set to `false`, the fast solver using FFT is used.

`condition.dynamics_simulation`
(*bollean* or *number*)

Whether to perform the dynamics simulation or not. If `condition.dynamics_simulation` is set to `true`, `drops` performs the dynamics simulation. If it is set to `false`, it performs the statics (equilibrium) simulation.

`condition.save_psi_sequential`
(*boolean* or *number*)

Whether to save the psi-field sequentially or not. If `condition.save_psi_sequential` is set to `true`, `drops` saves the psi-field every `condition.interval` steps. The output file name is generated from `file.psi_template`. If it is set to `false`, `drops` saves psi-field to the output file named `file.psi_output` every `condition.interval` steps (in other words, the output file is overwritten). This behavior is the same for other output files (the chemical potential field, the density field, the free energy, the geometry and the DX output).

`condition.save_density_sequential`
(*boolean* or *number*)

Whether to save the chemical potential field sequentially or not. See `condition.save_psi_sequential` for detail.

`condition.save_chemical_potential_sequential`
(*boolean* or *number*)

Whether to save the density field sequentially or not. See `condition.save_psi_sequential` for detail.

`condition.save_free_energy_sequential`
(*boolean* or *number*)

Whether to save the free energy sequentially or not. See `condition.save_psi_sequential` for detail.

`condition.save_geometry_sequential`
(*boolean* or *number*)

Whether to save the geometry field sequentially or not. See `condition.save_psi_sequential` for detail.

`condition.save_dx_sequential`
(*boolean* or *number*)

Whether to save the DX output sequentially or not. See `condition.save_psi_sequential` for detail.

`condition.error_tolerance`
(*number*)

Allowed error for the free energy. `drops` stops the simulation if the absolute value of the difference of the free energy is smaller than `condition.error_tolerance`.

`condition.seed`
(*number*)

Seed for the Mersenne twister random number generator.

`condition.noise`
(*number*)

Magnification of the random noise added to the chemical potential field. Set to `0` if you do not want to apply any random force.

`condition.initial_noise`
(*number*)

Magnification of the random noise added to the density field at the beginning of the simulations. Set to `0` if you do not want to add any random noise to the field.

`condition.phi_min`
(*number*)

Allower minimum value of the density field for dynamics simulations. If the value of the density, `phi`, is lower than `condition.phi_min`, `drops` automatically correct the density field. If `condition.phi_min` is positive value, the density correction is not performed. This parameter is only for dynamics simulations, and not used in the statics simulations.

`condition.iteration_max`
(*number*)

Maximum number of iterations for the simulation. `drops` ends the simulation if the number of iterations reaches `condition.iteration_max`.

`condition.interval`
(*number*)

Interval for the lattice optimization, file output or calculating the difference of the free energy.

`condition.omega`
(*number*)

Acceleration factor for the density evolution. Too large `condition.omega` causes numerical instability associated with the employed numerical scheme.

6.2 Input / Output Files

The input / output file names are set as the `file` table.

`file.wisdom`
(*string*)

FFTW wisdom file used by the FFTW library. If the file named `file.wisdom` exists already, FFTW reads the wisdom from it. If the file does not exist FFTW saves its wisdom to the file which can be used for latter simulations.

`file.psi_output`
(*string*)

Output file name for the psi-field. This is used when `condition.save_psi_sequential` is set to `false`. If the `file.psi_output` is set to the null string (`""`), no output file will be created. It is the same for for other output files (the chemical potential field, the density field, the free energy, the geometry and the DX output).

`file.density_output`
(*string*)

Output file name for the density field.

`file.chemical_potential_output`
(*string*)

Output file name for the chemical potential field.

`file.free_energy_output`
(*string*)

Output file name for the free energy.

`file.geometry_output`
(*string*)

Output file name for the geometry.

`file.dx_output`
(*string*)

Output file name for the DX output file.

`file.psi_template`
(*string*)

Template for the output file of the psi-field. `file.psi_template` must contains `%d` once. `%d` will be replaced by the sequential number `1,2,3,...` If `file.psi_template` is set to the null string (`""`), no output file is created.

`file.density_template`
(*string*)
Template for the output file of the density field.

`file.chemical_potential_template`
(*string*)
Template for the output file of the chemical potential field.

`file.free_energy_template`
(*string*)
Template for the output file of the free energy.

`file.geometry_template`
(*string*)
Template for the output file of the geometry.

`file.dx_template`
(*string*)
Template for the output file of the DX output file.

6.3 Multigrid Solver

The input / output file names are set as the `multigrid` table. The multigrid solver is not used if `condition.use_multigird_solver` is set to `false`.

`multigrid.n_cycle`
(*number*)
Number of iteration for multigrid V-cycle.

`multigrid.n_pre`
(*number*)
Number of iteration for pre-smoothing Gauss-Seidel method.

`multigrid.n_post`
(*number*)
Number of iteration for post-smoothing Gauss-Seidel method.

`multigrid.error_tolerance`
(*number*)
Allowed error for the Poisson equation. The multigrid solver stops relaxation if the mean square residual is less than `multigrid.error_tolerance`.

6.4 Geometry of Simulation Box

6.5 geometry

The input / output file names are set as the `geometry` table.

`geometry.dimension`
(*number*)
Number of dimension(s). This must be set to `1`,`2` or `3`.

`geometry.nx`
(*number*)
Number of division in x-direction.

`geometry.ny`
(*number*)
Number of division in y-direction (not used for 1 dimensional systems).

`geometry.nz`
(*number*)
Number of division in z-direction (not used for 1, 2 dimensional systems).

`geometry.lx`
(*number*)
Length of the edge of the simulation box in x-direction.

`geometry.ly`
(*number*)
Length of the edge of the simulation box in x-direction (not used for 1 dimensional systems).

`geometry.lz`
(*number*)
Length of the edge of the simulation box in x-direction (not used for 1,2 dimensional systems).

6.6 Polymer Blend

The information about the polymer blend is set as the `blend` table. The polymers which is contained in the system is set as the individual tables.

`blend.polymer`
(*array of strings*)
Polymers which is contained in the blend. The polymers used here must be defined as individual tables.

`blend.volume_fraction`
(*array of numbers*)
Volume fraction of each polymers (`drops` automatically normalize `blend.volume_fraction`).

6.7 Monomer Species

The information about monomers is set as the `monomer` table.

`monomer.name`
(*array of strings*)
Names for each monomers.

`monomer.b`
(*array of numbers*)
Kuhn length (effective segment size) for each monomers.

`monomer.chi`
: (*array of array of numbers*)
Flory-Huggins chi parameters for monomers. Only the diagonal and upper triangular part are used.

6.8 Polymer Species

The polymers which is used in `blend.polymer` is defined as individual tables of which name is same as the element of `blend.polymer`. For example, if `blend.polymer` is set to `{AB_diblock, C_homo}` the tables `AB_diblock` and `C_homo` must be defined.

polymer`.N`
: (*number*)
Polymerization index of the polymer.

polymer`.f`
: (*array of numbers*)
Block ratio for each subchains (`drops` automatically normalize *polymer*`.f`).

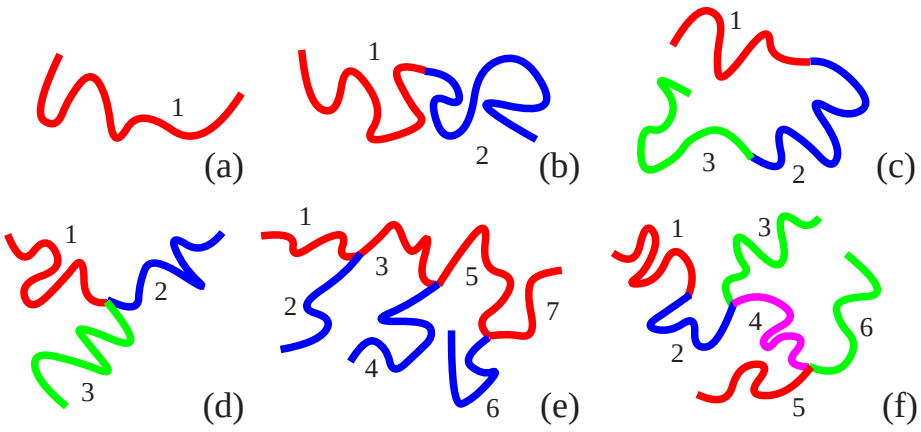
polymer`.a`
: (*array of array of booleans or numbers*)
Adjacency matrix. *polymer*`.a` specifies the topology of the polymer (connectivity of subchains). Only the upper triangular part is used.

polymer`.monomer`
: (*array of numbers*)
Monomers for each subchains.

polymer`.lambda`
: Tang-Freed type cutoff length for the long range interaction. If *polymer*.lambda is negative or 0, cutoff length is set to infinity (no cutoff length).

6.9 How to Determine the Adjacency Matricies

Most of the input parameters are easy to understand. The most confusing parameters are the adjacency matrix `polymer.a` which represents the connectivity of subchains. Here we show how to determine the adjacency matricies for block copolymers.



The above figure shows some polymer species; (a) homopolymer, (b) diblock copolymer, (c) triblock linear copolymer, (d) triblock star copolymer, (e) comb copolymer, (f) copolymer with complicated structure. The numbers (`1,2,...`) shown in figure corresponds to the index of the subchains.

The adjacency matricies for these polymer species are determined as follows.

(a) homopolymer

The homopolymer has only one subchain. Since there are no non-diagonal elements, the adjacency matrix has no meaning (but it must be specified, or `drops` won't work).

```
homopolymer.a = {0}
```

(b) diblock copolymer

This is the simplest block copolymer except for the homopolymer. Subchain `1` and subchain `2` are connected, so `a[1][2] = 1`.

```
diblock.a = {{0, 1},
             {1, 0}}
```

Note that we can use `true` and `false` instead of `1` and `0`. Also note that the diagonal term and lower triangular part of the matrix are actually not used.

(c) triblock linear copolymer

The triblock linear copolymer and the triblock star copolymer (d) are good example for determining adjecency matrices. For linear copolymer, Subhain `1` is connected to subchain `2`, but not connected to subchain `3`. Subchain `2` is connected to subchain `3`. Thus `a[1][2] = 1, a[1][3] = 0, a[2][3] = 1`.

```
triblock_linear.a = {{0, 1, 0},
                     {1, 0, 0},
                     {0, 1, 0}}
```

(d) triblock star copolymer

The triblock star copolymer contains three subchains, but the connectivity is different from the triblock linear copolymer (c). In this case, all subchains are connected each other. `a[1][2] = a[1][3] = a[2][3] = 1`.

```
triblock_star.a = {{0, 1, 1},
                   {1, 0, 1},
                   {1, 1, 0}}
```

(e) comb copolymer

Here we consider the comb copolymer which contains seven subchains. Although it may look complicated, but determining the adjecency matrix is not so complicated. First, the backbone subchains `1,3,5,7` is connected just like the linear copolymer. Thus `a[1][3] = a[3][5] = a[5][7] = 1`. Next, each side subchains `2,4,6` is connected to two backbone subchains. `a[1][2] = a[3][2] = 1, a[3][4] = a[5][4] = 1, a[5][6] = a[5][7] = 1`. Any other elements are equal to 0. Finally `a` is expressed as

```
comb.a = {{0, 1, 1, 0, 0, 0, 0},
          {1, 0, 1, 0, 0, 0, 0},
          {1, 1, 0, 1, 1, 0, 0},
```

```
{0, 0, 1, 0, 1, 0, 0},
{0, 0, 1, 1, 0, 1, 1},
{0, 0, 0, 0, 1, 0, 1},
{0, 0, 0, 0, 1, 1, 0}}
```

(f) general block copolymer (with complicated structure)

As a final example, here we show the general block copolymer with complicated structure. This block copolymer has 6 subchians. We start from the subchain `1`. It is connected to subchain `2`, so `a[1][2] = 1`. Subchain `2` is connected subchains `3` and `4` (of course it is connected to `1`, but we already know it). `a[2][3] = a[2][4] = 1`. Subchain `3` is connected to subchain `4`. `a[3][4] = 1`. Subchain `4` is connected to subchains `5` and `6`. `a[4][5] = a[5][6] = 1`. Subchain `5` is connected to subchain `6`. Thus we scanned all the connectivity. Now the adjacency matrix is

```
general.a = {{0, 1, 0, 0, 0, 0},
             {1, 0, 1, 1, 0, 0},
             {0, 1, 0, 1, 0, 0},
             {0, 1, 1, 0, 1, 1},
             {0, 0, 0, 1, 0, 1},
             {0, 0, 0, 1, 1, 0}}
```

7 Output File Format

In this section, the output file format for `drops` is described.

7.1 Psi-Field

The output file for the psi-field is gzipped text. Each low corresponds to the one lattice point and each column corresponds to the subchain. For example, if `blend.polymer` is set to `{AB_diblock, C_homo}` and `AB_diblock.monomer` is set to `{"A","B"}`, The output data is like the following data (the output file itself is gzipped).

```
0.0241512 0.29688  0.954609
0.0240064 0.291801 0.956178
0.0238619 0.286763 0.957704
0.0237176 0.281766 0.95919
0.0235735 0.276813 0.960634
0.0234294 0.271903 0.962039
0.0232853 0.267038 0.963404
0.0231412 0.262219 0.964731
    :         :        :
```

The first and second column correspond to the psi-field of A subchain and B subchain of AB diblock copolymer, The third column corresponds to one of C homopolymer. You can deflate the output file by using `gunzip` or `zcat`.

7.2 Density Field

The output file for the density field is gzipped text. The format is like the psi-field output file, but contains position data. The first column (and the second, third column(s) if the dimensions of the system is greater than 1) is the position.

```
0       0.000583278 0.0881376 0.911279
0.03125 0.000576307 0.0851477 0.914276
0.0625  0.000569391 0.0822328 0.917197
0.09375 0.000562526 0.0793923 0.920045
0.125   0.000555708 0.0766254 0.922819
0.15625 0.000548935 0.0739314 0.925519
0.1875  0.000542204 0.0713094 0.928148
0.21875 0.000535514 0.0687586 0.930705
   :         :          :         :
```

It is convenient to use `gnuplot` to plot the 1D or 2D density data. For example, to plot 1D data, `gnuplot` command will be

```
gnuplot> plot "< zcat phi.dat.gz" using 1:2 title "A" with lines, \
                                  using 1:3 title "B" with lines, \
                                  using 1:4 title "C" with lines
```

and to plot 2D data, the command will be

```
gnuplot> splot "< zcat phi.dat.gz" using 1:2:3 title "A" with lines, \
                                   using 1:2:4 title "B" with lines, \
                                   using 1:2:5 title "C" with lines
```

7.3 Chemical Potential Field

The output file for the chemical potential field is gzipped text. The data format is same as the density output file.

7.4 Free Energy

The output file for the free energy is text file. The total free energy, the long range term, the local term, the gradient term are stored in this order. The sample output file is as follows.

```
-0.0925369 0.00859006 -0.13708 0.0359527
```

7.5 Geometry

The output file for the geometry is text file. The first row describes dimension(s) of the system, the second row describes number(s) of division and the third row describes length(es) of the edge of the simulation box. The sample output file is as follows.

```
1
1024
32
```

7.6 DX Output File

The DX output file is the data format for OpenDX (visualization software). It contains the density field (`phi0,phi1,...`) and the chemical potential field (`mu0,mu1,...`). You can visualize it by using OpenDX. The sample OpenDX program to visualize the OpenDX format `drops` output data will be found in the `example/` directory.

8 Utility Programs

8.1 Converter for DX Output Files

The utility programs `drops-dx2phi`,`drops-dx2psi`, and `drops-dx2mu` convert an OpenDX format output file generated by `drops` into gzipped text files. `drops-dx2phi`,`drops-dx2psi`, and `drops-dx2mu` convert density fields, chemical potential fields, and psi-fields in the DX file.

To convert density fields in a DX output file `dropsout.dx` into a gzipped text file `phi.dat.gz`, execute `drops-dx2phi` as follows.

```
$ drops-dx2phi dropsout.dx phi.dat.gz
```

`drops-dx2psi` and `drops-dx2mu` can be used in the same way.

8.2 Converter for Gzipped Text Files

The utility program `drops-phi2psi` converts a gzipped text file of density fields into another gzipped text file of psi-fields.

To convert density fields file `phi.dat.gz` into a psi-fields file `psi.dat.gz` in a 3 dimensional system, execute `drops-phi2psi` as

```
$ drops-dx2phi -d 3 phi.dat.gz psi.dat.gz
```

The dimension of the system must be specified (use `-d` option).

9 References

[Bates-Fredrickson-1999] F. S. Bates and G. H. Fredrickson, *Phys. Today* **52**, 32 (1999).

[Bohbot-Raviv-Wang-2000] Y. Bohbot-Raviv and Z.-G. Wang, *Phys. Rev. Lett.* **85**, 3428 (2000).

[Choucair-Eisenberg-2003] A. Choucair and A. Eisenberg, *Eur. Phys. J. E* **10**, 37 (2003).

[Disher-Eisenberg-2002] D. E. Disher and A. Eisenberg, *Science* **297**, 967 (2002).

[Drolet-Fredrickson-1999] F. Drolet and G. H. Fredrickson, *Phys. Rev. Lett.* **83**, 4381 (1999).

[Fraaije-1993] J. G. E. M. Fraaije, *J. Chem. Phys.* **99**, 9202 (1993).

[Fredrickson-Ganesan-Drolet-2002] G. H. Fredrickson, V. Ganesan and F. Drollet, *Macromolecules* **35**, 16 (2002).

[Groot-Madden-1998] R. D. Groot and T. J. Madden, *J. Chem. Phys.* **108**, 8713 (1998).

[Groot-Madden-Tildesley-1999] R. D. Groot, T. J. Madden and D. J. Tildesley, *J. Chem. Phys* **110**, 9739 (1999).

[Groot-Warren-1997] R. D. Groot and P. B. Warren, *J. Chem. Phys.* **107**, 4423 (1997).

[Helfand-Wasserman-1976] E. Helfand and Z. R. Wasserman, *Macromolecules* **9**, 879 (1976).

[Helfand-Wasserman-1978] E. Helfand and Z. R. Wasserman, *Macromolecules* **11**, 960 (1978).

[Helfand-Wasserman-1980] E. Helfand and Z. R. Wasserman, *Macromolecules* **13**, 879 (1980).

[Joanny-Leibler-1978] J. F. Joanny and L. Leibler, *J. Phys. (Paris).* **39**, 951 (1978).

[Kawakatsu-1994] T. Kawakatsu, *Phys. Rev. E* **50**, 2856 (1994).

[Kawasaki-Ohta-Kohrogui-1988] K. Kawasaki, T. Ohta and M. Kohrogui, *Macromolecules* **21**, 2972 (1988).

[Koizumi-Hasegawa-Hashimoto-1994] S. Koizumi, H. Hasegawa and T. Hashimoto, *Macromolecules* **27**, 6532 (1994).

[Leibler-1980] L. Leibler, *Macromolecules* **13**, 1602 (1980).

[Matsen-Bates-1996] M. W. Matsen and F. S. Bates, *Macromolecules* **29**, 1091 (1995).

[Matsen-Schick-1994] M. W. Matsen and M. Schick, *Phys. Rev. Lett.* **72**, 2660 (1994).

[Nakazawa-Ohta-1993] H. Nakazawa and T. Ohta, *Macromolecules* **26**, 5503 (1993).

[Ohta-Ito-1995] T. Ohta and A. Ito, *Phys. Rev. E* **52**, 5250 (1995).

[Ohta-Kawasaki-1986] T. Ohta and K. Kawasaki, *Macromolecules* **19**, 2621 (1986).

[Ohta-Kawasaki-1990] T. Ohta and K. Kawasaki, *Macromolecules* **23**, 2413 (1990).

[Uneyama-Doi-2005] T. Uneyama and M. Doi, *Macromolecules* **38**, 196 (2005).

[Uneyama-Doi-2005a] T. Uneyama and M. Doi, *Macromolecules* **38**, 5817 (2005).

[Uneyama-2007] T. Uneyama, *J. Chem. Phys.* **126**, 114902 (2007).

[de Gennes-1980] P. G. de Gennes, *J. Chem. Phys.* **72**, 4756 (1980).

[Introduction to Polymer Physics] M. Doi, *Introduction to Polymer Physics*, Clarendon Press (1996).

[Scaling Concepts in Polymer Physics] P. G. de Gennes, *Scaling Concepts in Polymer Physics*, Cornell University Press (1979).

[Statistical Physics of Polymers: An Introduction] T. Kawakatsu, *Statistical Physics of Polymers : An Introduction*, Springer Verlag (2004).

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.
5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software

which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) 19yy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type ‘show w’.  This is free software, and you are welcome
to redistribute it under certain conditions; type ‘show c’
for details.
```

The hypothetical commands ‘`show w`’ and ‘`show c`’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘`show w`’ and ‘`show c`’; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program ‘Gnomovision’
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Concept Index

A

adjacency 20

B

block copolymer 1, 20
bugs 13

C

chemical potential 24
chemical potential field 24
connectivity 20
converter 25

D

density field 23
download 3
drops 1, 5
drops-dx2mu 25
drops-dx2phi 25
drops-dx2psi 25
drops-phi2psi 25

E

example 7

F

FFTW 3
format 15, 23
free energy 24

G

geometry 24
getting help 5
Gnuplot 7
GPL 29

H

help 5

I

icc 4
input 15
input / output files 17
install 3
Installer 3
Intel C++ Compiler 4
introduction 1
invoking 5

L

license 29
Lua 3

M

microphase separation 1
monomer 19
multigrid solver 18

O

OpenDX 7, 24
options 5
output 23

P

plot 7
polymer 20
polymer blend 19
problems 13
psi-field 23

R

references 27
RPM 4

S

simulation condition 15
source 3
SRPM 4

T

tutorial 7

U

usage 5
utilities 25
utility programs 25

V

version 5
visualization 7

W

Windows 3

Z

zlib 3