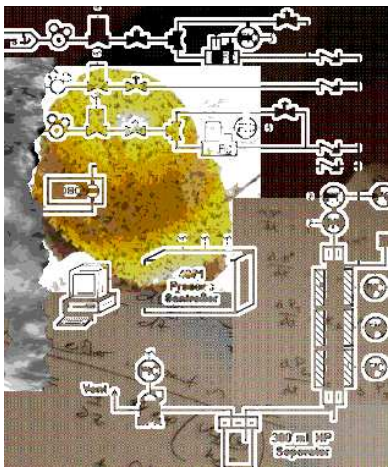
Bob-2.3



September 30, 2008

Contents

Chapter 1

Overview

BOB calculates the rheological response of polymer melts of arbitrary architecture using extended *tube model*. Rheological response of polymers extend over large number of decades making a direct simulation from atomistic model unfeasible. At the same time the molecular complexity allows for simple mean field description, often at different levels of complexity to capture the physics involved at a certain time-scale. The approach here is to link molecular structure to rheological properties keeping the bare minimum about the molecules.

The main barrier against relaxation in *entangled* polymers is the topological non-crossing condition. Once the time scale is few orders of magnitude larger than atomic time-scale, the chemical details of the monomers can be effectively described by two single parameters - one describing the entanglement length (tube diameter or entanglement molar mass M_e) and a time scale (or molecular friction). Different polymer melts of the same chemistry (having same monomers) behave differently because of the connectivity (branching pattern).

The first stage of the calculation generates a representation of polymers (possibly a large numbers of them) keeping only the connectivity and molar mass between the junctions. This can come from analysis of chemical reaction scheme, from analysis of GPCLS data or just from a cartoon to compare with some analytical formula. The numerical scheme makes it straightforward to include polydispersity in both the molar mass and the connectivity. The strength of bob is in the realm of branched polymers. It handles linear polymer also, but there are better predictive tools if you are interested about pure linear melt. Mostly there are few unavoidable (or carefully in-

serted) branches in the polymer of interest. And in such cases, much of the rheological signature is dominated by the branching topology.

In the rest of this chapter, you will be lead through the installation procedure and a quick tutorial about using the software. You can find out about the rest by experimenting or by looking up the later chapters of this document.

1.1 Installation

We assume that you have downloaded the tar-gzipped source file bob.2.3.tar.gz. For unix-like systems (UNIX, Linux, FreeBSD $\cdots$), use *gunzip bob.2.3.tar.gz* to unzip and *tar -xvf bob.2.3.tar* to retrieve the directory structure.

Go in the directory *bob2.3/code/src/obj* and type in *make*. This will create the stand alone executable *bob*. The top line of the *makefile* contains *cpp=g++ -Wall -O3*. You may need to change this line to access your local installation of c++ compiler.

For MS-Windows systems, you can install MinGW/MSYS (http://www.mingw.org) or Cygwin/X (http://x.cygwin.com). This gives you an Unix like environment on top of the windows. You can compile the code in exactly the same fashion as mentioned for unix-like systems. This will create a file *bob.exe* in the compilation directory.

If you prefer, you can download the pre-compiled executable from the distribution site for MS-Windows OS. Be aware that downloading an executable is prone to security risk.

Move the executable file to some location which is in your path and you are are almost ready to to use the software.

The only other things you need are a text editor - your favorite one (not word or notepad - but wordpad should be fine), a plotting software, and a cup of strong coffee.

1.2 Quick tutorial

The following is most directly useful for Unix / Linux / Msys / cygwin $\cdots$ systems. For windows OS without Msys or cygwin, you can still use the same approach by using a command line window (start $\rightarrow$ run $\rightarrow$ cmd). If you do not like command line, create a batch file which includes the path for the

executable and any command line arguments to run the executable.

For this section, I am assuming that you placed the binary in */home/joe/bin*, the distribution directory is */home/joe/bob2.3* and you are working within directory */home/joe/bobtest*.

copy the files *bob.rc* from */home/joe/bob2.3/examples* to the current directory. Also copy *star_comb_inp.dat* from the same directory. Type in */home/joe/bin/bob -b -i star_comb_inp.dat*.

After about 10 seconds the program ends. So far the screen probably looked like

```
$ pwd
/home/joe
$ mkdir bobtest
$ cd bobtest
$ cp /home/joe/bob2.3/examples/bob.rc .
$ cp /home/joe/bob2.3/examples/star_comb_inp.dat .
$ /home/joe/bin/bob -b -i star_comb_inp.dat

 bob-2.3 : rheology of Branch-On-Branch polymers
Entering batch mode
Using input file as star_comb_inp.dat
Using default configuration file polyconf.dat

$
```

A number of files are created in the run directory. View the file *info.txt*. The first 40 or so lines are about what options bob selected for the run (bob is not intelligent, so it was told by default choice, overridden by the contents of *bob.rc*, which in turn were overridden by *star_comb_inp.dat*). Jump to the line which says "End of input parameters". Next you will find that it has created 5 star polymers with 3 segments each and 5 comb polymers with 5 side arms.

Open the file *polyconf.dat* with WordPad or some other ASCII editor - there are entries like

```
   : : :
3
-1 -1 1 2 2.644074e+01 4.156591e-02
0 2 -1 -1 1.055119e+01 1.658690e-02
0 1 -1 -1 2.840818e+01 4.465880e-02
   : : :
7
-1 -1 1 2 1.639851e+01 2.684047e-02
-1 -1 0 2 1.167038e+01 1.910165e-02
0 1 3 4 1.945459e+00 3.184256e-03
-1 -1 2 4 1.301297e+01 2.129914e-02
2 3 5 6 5.192398e-01 8.498724e-04
-1 -1 4 6 1.104441e+01 1.807707e-02
4 5 -1 -1 5.123094e+00 8.385291e-03
   : : :
```

These are bob's way of representing polymers. Each polymer starts with number of segments forming the polymer and then description of each of the segments. Each segment has four integers telling about how it is connected to rest of the segments forming this particular polymer. The fourth column is mass in units of entangled molar mass and the last column tells about the weight fraction of the segment in the melt.

Note that the first five polymers each have 3 arms. The last five have variable number of segments.

Repeat the exercise and you will find quite different set of polymers in the file. The random number seed is chosen from the system random number pool. So, each time you run the code, you are expected to get different results.

But this is not an expensive random number generator! The file *gtp.dat* contains 3 columns: frequency(ω), elastic modulus $G^{'}(\omega)$ and the viscous modulus $G^{''}(\omega)$. Plot the frequency dependence on log-scale. The behavior should not depend too much on the particular run. In this case, however you will find that the low frequency modulus varies appreciably from run to run. So, we need to increase the number of polymer that we generate. Edit the file *star_comb_inp.dat*. The content is

```
 1          50000 500000
 2          1.0
 3          1
 4          28 40 0.70
 5          3e-7 463.15
 6          2
 7          0.4
 8          5  1
 9          2 22260 1.40
10          3
11          0.6
12          5  4
13          2 22260 1.010
14          2 12260 1.010
15          5.0
```

Here, we added the line numbers, which are not in the file. Line 8 tells the code that there should be 5 polymers of type 1 (star). Change the number of polymer to 500. Do the same on line 12, which right now asks for 5 polymers of type 4 (comb). Save and rerun bob. Now there should be much less variability even for the lowest frequency response among repeated runs.

You also can run the code interactively by running the executable without any argument. It will ask you a number of questions to find out about the polymers and the relaxation mechanism to be considered. You can control the format for the program output. You can continue reading this manual to find out about what all the lines in the input file means. Or you can go through the source code to figure out how the code is working. The later also gives you the opportunity to do things bob is not designed for!

1.3 Credits

The program was developed with a funding from EPSRC.

Internally it uses a Mersenne Twister random number generator:
`http://www-personal.engin.umich.edu/ wagnerrr/MersenneTwister.html`

Special thanks to

- Jorge Ramirez
- Jaap den Doelder

- Zuowei Wang
- Ron Larson
- Micheal Kapnistos

for pointing out bugs and helpful discussions.

1.4 Known issues

- Bob can not handle cyclic molecules or true gels. If any polymer remains unrelaxed after $10^{20}\tau_e$, the program is instructed to give up trying to relax.
- Prefactor for compound arm retraction probably needs more consideration. Experimental input from well characterized branch-on-branch polymers or theoretical / atomistic simulation is needed to resolve if the approach taken in this program is justified.

1.5 Bug reports

If you discover any bugs, please report to `chinmaydas@yahoo.com`. Also you can help extend the capabilities of bob by donating code segments which can generate new types of polymers or some new kind of analysis by sending a mail to the same address.

1.6 Copy/reuse policy

You are allowed to freely copy, redistribute, modify, delete, store in any media of your choice, install in any computer(s) of your choice. You are not required to, but highly encouraged to acknowledge the software and associated publications if you use this for scientific publication or use part or full of it in your own commercial or not-for-profit software.

1.7 Disclaimer

The software is provided "As Is" without any warranties. The authors are not responsible for any direct or indirect awful consequences of using this software. But they are more than willing to bask in the glory of successful usage. Do drop in a line if you find it useful!

1.8 Publications related to bob

- Computational linear rheology of general branch-on-branch polymers, *Chinmay Das, Nathaniel Inkson, Daniel J. Read, Mark A. Kelmanson, and Tom C. B. McLeish*, Journal of Rheology, **50**, 207–235 (2006).

  This describes in details the theory behind the relaxation scheme.

- Dynamic scaling in entangled mean-field gelation polymers, *Chinmay Das, Daniel J. Read, Mark A. Kelmanson, and Tom C. B. McLeish*, Phys. Rev. E., **74**, 011404 (2006).

  Mean-field gelation polymer generation.

- Synthesis, temperature gradient interaction chromatography, and rheology of entangled styrene comb polymers, *Pierre Chambon, Christine Fernyhough, Kyuhyun Im, Taihyun Chang, Chinmay Das, John Embery, Tom C. B. McLeish, Daniel J. Read*, Macromolecules, **41**, 5869–5875 (2008).

  Consideration of coupled combs.

1.9 Changes

This section summarizes the changes in version 2.3 from earlier versions. A few changes are critical, in the sense that unless you change the input style for these parts, the behavior will be unpredictable.

1.9.1 Critical changes

- Renumbering of polymer types. Check the polytype table.
- Comb molecules now require the number of side arms in a different line from the arm length information. Always arm information from input files now will contain *arm_type, arm_mw, arm_pdi*. When we require extra information for a polymer type, we use a separate line.

1.9.2 New features

- A number of parameters now can be set with a file called *bob.rc*.
- Output format can be changed to conform with *reptate*™ software.
- Output can be saved as figures for xmgrace.
- New polymer types : mean field gelation ensemble, coupled combs, Cayley tree with inner segment being linear or 3 arm star or 4 arm star.
- You can now read in polymer configuration from any number of files and create blends from that.
- GPCLS prediction for molar mass distribution, number of branch points and radius of gyration.

1.9.3 Features planned for next release

- Predictions for nonlinear rheology.
- Support for MPI-parallel run.
- LDPE and other polymer types

Expected date for next release : Jan. 2009

Chapter 2

Input files

There are four different ways to control the working of bob - (i) command line argument, (ii) bob.rc file, (iii) input parameter file and (iv) interactively in case of interactive mode. Some of the choice can be set more than one way. In that case, the bob.rc file has the least priority.

2.0.4 Command line arguments

Without any argument, bob starts interactively, expecting the user to input parameters by asking few questions.

- **- -help**: prints help file and quits.
- **- -version**: outputs the version number before quitting.
- **-b** : batch mode with default file names.
- **-i filename**: Use the named file as input file. Default input file name 'inp.dat'.
- **-c filename**: Use the named file for polymer configuration. Default configuration file name 'polyconf.dat'.
- **-p** : Generate the polymers and quit before trying to relax the polymers.

2.0.5 bob.rc

The program looks for an optional file 'bob.rc' in the current directory. If the file does not exist, the program sets default values for some parameters and continue. 'bob.rc' treats any line without an = sign as an comment. If a line has an '=' sign, it treats the left-hand side of '=' as the option/parameter and the right-hand side as the choice/value. If it fails to understand some line, it continues with the default value. A sample 'bob.rc' with all parameters set to default values is in the directory bob2.3/examples. A description of the valid options follows with the default option shown in square brackets. The option names are case sensitive.

bob.rc format		
option/parameter	choice/value	description
GenPolyOnly	[no]/yes	'yes' stops after generating polymers
OutMode	[0]	output format : headerless ascii file
	1	xmgrace plot
	2	ascii file with reptate header
PSquare	[0.0250]	p^2 for branch-point friction
Alpha	[1.0]	tube dilation exponent
TStart	[1.0e-4]	time at which to start integration
DtMult	[1.0050]	ratio of successive integration times
CalcGPCLS	[no]/yes	Calculate molar mass distribution, branching and ideal g-factor (R_g^2[branched] / R_g^2 [Linear])
GPCNumBin	[50]	Number of bins for GPC-LS histogram
FreqMax	[1.0e8]	maximum frequency to which to calculate G'/ G"
FreqMin	[1.0e-3]	minimum frequency for G'/ G" calculation
FreqInterval	[1.1]	ratio of successive frequencies of G'/G" data
untested options		**The effect of the following are not tested. Set to non-default values at your own risk!**
PrefMode		Kramer first passage time prefactor for compound arm
	0	same as outermost arm
	[1]	use the effective armlength
	2	include full effective friction
RetLim	[0.0]	side arms are considered free to hop once it reaches within 'RetLim' from branch point
ReptScheme	[1]	Reptation in thin tube
	2	Reptation in current tube
	3	use tube diameter from time at which a 'ReptAmount' long linear polymer can reptate
	4	Tube diameter from time when a linear polymer with 'ReptAmount' fraction of current polymer can reptate
ReptAmount	[1.0]	Used when 'ReptScheme' is set to 3 or 4

2.0.6 Input parameter file

The default input parameter file name is 'inp.dat'. You can use command line argument *-i filename* to use a different file. For interactive mode, input file is not used. For repeated use, using the input file probably will be the more efficient option.

The input file is in ascii with rather strict format. You can only add comments after all the valid entries. Let us look at the first five lines of the input file used in quick tutorial again (file star_comb_inp.dat in examples directory of the distribution):

```
1         50000 500000
2         1.0
3         1
4         28 40 0.7
5         3e-7 463.15
```

Remember that the input file does not have the line numbers, they are here just to help us refer easily to specific lines. The first 3 lines control the memory and relaxation assumptions. Lines 4-5 gives input about the chemistry of the polymer.

line 1

The first line has two integers: The first entry is for maximum number of polymers and the second entry is for maximum number of segments. This is used for memory allocation. You need to use large enough space to accommodate all the polymers in the current session. Having more than you actually use probably will slow down the computation a bit. But having less than you actually use will abort execution of the code.

line 2

This is a single real number telling the value of the dynamic dilation exponent α. In the example we asked bob to use $\alpha = 1.0$.

line 3

The integer entry on 3rd line gives you control of fine tuning several parameters. If you put in 1, bob uses the stored values for these parameters. For compatibility issues, this line still is needed and will work as in past versions with old input files. But if you don't know what else you are allowed to put there, just keep it 1. Use *bob.rc* to set other parameters (that provides a cleaner input interface).

line 4

Three real numbers giving the monomer mass M_0 in g/mol, the number of monomers in one entangled segment N_e and density ρ in g/cc. For the density you can also use kg/m^3. If the density is less than 2.0, the program assumes that the unit used is g/cc. Else it uses kg/m^3. This ambiguity is to allow old input files being correctly interpreted and at the same time always have gram as the unit of mass for input.

line 5

Two real numbers giving entanglement time τ_e and temperature T at which relaxation information is required.

From line 6 onwards we specify the architecture:

```
 6        2
 7        0.4
 8        5  1
 9        2 22260 1.40
10        3
11        0.6
12        5  4
13        2 22260 1.010
14        2 12260 1.010
15        5.0
```

line 6

Line six needs an integer telling the number of components you want to blend. For a single polymer type (there can be any number of polymers of this kind), the entry would have been 1.

0 (zero) also is a valid entry on this line - this says the code that the polymer configurations are already generated (possibly from another code) and the code should use the file to read in the polymers. In that case this is the last line the code reads from the input file.

Once we specified the number of components (a non-zero integer), next entries are for individual components, one at a time.

Always the fist entry in this part is the weight fraction of the current component (you explicitly need to mention 1.0, if you use one component polymer).

The next entry has two integers: first the number of polymers in the current component, the second is the polymer type. For each polymer type you need to provide different kind of information. See the section on known polymers in chapter 3 to find out the code and input parameters for a particular

polymer type.

For now, we note that 1 is the code for star polymer. **Line 7** tells that the first component occupies an weight fraction of 0.4. **Line 8** asks to generate 5 polymers of kind star (type 1). Star polymers require the distribution arms are sampled from, M_W and PDI of the arms. They are coded in **line 9**. **Line 10** says that the star polymer is a 3-arm star.

Similar information is coded for comb polymers as the second component in **lines 11-15**.

2.0.7 Interactive mode

When you execute bob without any argument, it requires you to type in the input parameters by asking questions. This is useful if you are just going to use the code occasionally. For repeated use, input file is a better idea - if you type something wrong during the interactive mode, you have to restart from the beginning.

The following is what a session looked like:

```
$ ../../bob2.3/code/src/obj/bob.exe

 bob-2.3 : rheology of Branch-On-Branch polymers
Starting up interactive mode

Maximum number of polymer you want to consider ? ... 100
Maximum total number of arms (each linear require two arms) ?5000


Please enter dilation exponent alpha : 1.0

We need assumptions/parameters re. branch point hops and reptation
Type 1 if you want me to choose for you else type 0 ?? ... 1
```

The first two questions are about memory allocation. The typed information asked to reserve memory for 100 polymers and 5000 segments (arms). The dilation exponent (α) is set to 1.0. For assumptions regarding branch point hops and reptation, we asked the code to use the default set up or look them up from *bob.rc*. The question/answer session continues.

```
Mass of a monomer (in atomic unit, ex.:PE->28.0) ?  28.0
Number of monomers in an entanglement length ?  40.0
mass-density of the polymer (in g/cc) ?  0.7
Entanglement time tau_e (s) ?  1.0e-7
Temperature (Kelvin) ? 430.0
```

Here we have set the chemical description of the monomers and the experimental temperature. The parameters used in this example are typical for polyethylene melt at 430K.

```
Do you want to generate the polymers or read from file ?
Type in 1 for generating or 0 for reading from file ... 1
How many components you want ? 2
```

We asked the code to generate polymers for us. And we inform it that we are interested about two component blend.

```
Weight fraction occupied by component 1 ? ...0.4

 Polymer component 1 :
Number of polymers in current type?  20
Select polymer type :
Some (but not all) known polymer types follows:
Look at manual for all possibilities
    type '0' for linear
         '1' for star
         '2' for asymmetric star
         '3' for H
         '4' for Comb (number of side arms from Poisson dist.)
         '10' for Cayley tree
         '20' for metallocene-catalyzed polyethylene
0
Type 0 for strictly monodisperse
     1 for Gaussian distribution in segment lengths
     2 for Lognormal
     3 for Semi-living (several monomer acts as unit to match PDI
     4 for Flory distribution
arm type ?  2
M_w of single segment (in g/mol) ?  10000.0
Polydispersity Index ?  1.4
```

The code asks for component 1. We have selected the component 1 to

occupy an weight fraction of 0.4 (i.e. 40% by weight). The code gave a short choice of polymers that are available. The actual number of available polymers is considerably larger. You need to look up the table of available polymers for that. We selected a linear with $M_W = 10000\,\mathrm{g/mol}$, PDI = 1.4 from a log-normal distribution and created 20 copies of it.

```
Weight fraction occupied by component 2 ? ...0.6

 Polymer component 2 :
Number of polymers in current type?  30
Select polymer type :
Some (but not all) known polymer types follows:
Look at manual for all possibilities
    type '0' for linear
         '1' for star
         '2' for asymmetric star
         '3' for H
         '4' for Comb (number of side arms from Poisson dist.)
         '10' for Cayley tree
         '20' for metallocene-catalyzed polyethylene
3
First we need information about the four side arms.
Type 0 for strictly monodisperse
     1 for Gaussian distribution in segment lengths
     2 for Lognormal
     3 for Semi-living (several monomer acts as unit to match PDI
     4 for Flory distribution
arm type ?  2
M_w of single segment (in g/mol) ?  12000
Polydispersity Index ?  1.2
Now information about the backbone.
Type 0 for strictly monodisperse
     1 for Gaussian distribution in segment lengths
     2 for Lognormal
     3 for Semi-living (several monomer acts as unit to match PDI
     4 for Flory distribution
arm type ?  0
M_w of single segment (in g/mol) ?  15000
```

We had to give the weight fraction for the second component also. This time we choose a H polymer with 12K side arms with PDI 1.2 and monodisperse crossbar of molar mass 15K.

After a minute or so, the program exits normally. And before that it creates a number of files. A snippet from the file info.txt:

```
Selected linear polymer from Lognormal distribution with
                  M_w = 1.000000e+04 and PDI = 1.400000e+00
created 20 Linear polymers.
component  0 : Total mass = 1.532971e+02
Selected H polymer
Side arms : from Lognormal distribution with
                  M_w = 1.200000e+04 and PDI = 1.200000e+00
Backbone : : monodisperse with M_w = 1.500000e+04
Created 30 H  polymers
component  1 : Total mass = 1.520772e+03
```

Now is the time to find out a bit more details about the different kind of polymers that can be used. And that's the topic for the next chapter.

Chapter 3

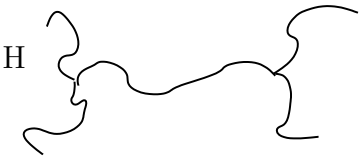
Polymer configuration

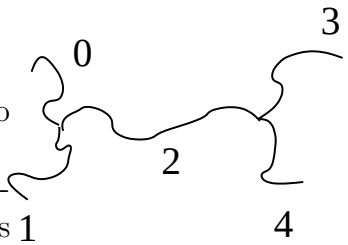
Bob uses polymer information at the segment level. Each segment has associated molar mass and are possibly connected to utmost two other segments on either ends. The number of polymers of a given kind in the representation need not have direct relationship with the actual mole fraction in the experimental blend. Instead each polymer comes with associated weight-fractions. The user is free to generate more polymers where polydispersity is higher and less where polymers relax very much the same way by adjusting the weight fraction of the components.

3.1 Polymer representation

For each polymer, the specification starts with number of segments forming the polymers. The segments are numbered from 0 onwards in some arbitrary fashion. For each segment, the two connections at the left and at the right end gives connectivity. -1 implies lack of connection at one end (free arm). Except for linear polymer, end of a segment is either a branch-point (two connections) or a free end (no connection). The mass is scaled by entangled molar mass M_e. The final entry for the segments is the weight fraction of the segment.

As a concrete example, let us consider a single H polymer which is composed of five segments.

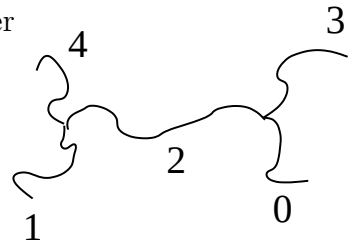




The first step is to number the segments from zero and form a connectivity matrix.
The lines in the matrix specify connection for segments 0 onwards. The 0'th arm (the first line) is not connected to anything on the left (entry -1, -1) and is connected to 1 and 2 on the right. And so on.

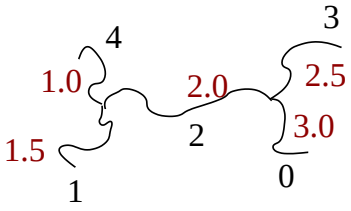
-1	-1	1	2
-1	-1	0	2
0	1	3	4
2	4	-1	-1
2	3	-1	-1

The numbering is not unique. Here is another possible arrangement representing the same polymer and the connectivity matrix:



2	3	-1	-1
-1	-1	4	2
1	4	0	3
2	0	-1	-1
-1	-1	1	2

Next, we decorate the segments with molar mass (in units of M_e). The table becomes:



2	3	-1	-1	3.0	
-1	-1	4	2	1.5	
1	4	0	3	2.0	
2	0	-1	-1	2.5	
-1	-1	1	2	1.0	

The final step is to put in the weight fraction of the segments. In this case we have only one polymer and conveniently the total mass of this polymer adds up to 10.0. So, the full specification of the polymer in this case become:

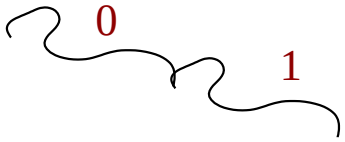
2	3	-1	-1	3.0	0.30
-1	-1	4	2	1.5	0.15
1	4	0	3	2.0	0.20
2	0	-1	-1	2.5	0.25
-1	-1	1	2	1.0	0.10

The specification is complete if this is preceded by an integer specifying

the number of segments in this particular polymer (5).

The linear polymer is a bit different. It is made of two segments of equal molar masses. Only one connector at one end of the first segment is connected to the second segment. For a single linear polymer with molar mass 5.0, the entry will be

-1	-1	1	-1	2.5	0.50
0	-1	-1	-1	5.5	0.50



Once again the full specification will need to specify number of segments (2).

Now we are ready to write the polymer configuration file. By default the file name is *polyconf.dat*. You can specify other file names with the flag -c in the command line argument.

The first line in the file can have a maximum of 10 character long descriptor. The second line has a single floating point number giving N_e. This is as a reminder of the way segment masses have been scaled. The N_e will be skipped by bob and only the N_e specified through the input file will be used for any calculation. The third line is the number of polymers.

Then for each polymer the first entry is number of segment and then for each segment the connectivity, molar mass and weight fraction.

If we want to create a file for the single H polymer, the file can be

```
H-polymer
40.0
1
5
 2   3   -1   -1   3.0   0.30
-1  -1    4    2   1.5   0.15
 1   4    0    3   2.0   0.20
 2   0   -1   -1   2.5   0.25
-1  -1    1    2   1.0   0.10
```

Similarly for a single linear polymer

```
LIN
40.0
1
2
-1  -1   1   -1   2.5   0.50
 0  -1  -1   -1   5.5   0.50
```

We can also blend the polymers - suppose we want to study 10% blend of H polymer (weight fraction 0.1) in a 90% linear matrix of the kind we generated. Now the total weight fraction of the segments forming the H polymer should add up to 0.1 and the weight fraction of the segments forming the linear polymer should add up to 0.9. Since the polymers are monodisperse - we can just use two polymers - one from each kind. The file will look like

```
H-LIN
40.0
2
5
 2   3   -1   -1   3.0   0.030
-1  -1    4    2   1.5   0.015
 1   4    0    3   2.0   0.020
 2   0   -1   -1   2.5   0.025
-1  -1    1    2   1.0   0.010
2
-1  -1    1   -1   2.5   0.450
 0  -1   -1   -1   5.5   0.450
```

Note that we specified two polymers on the third line and we changed the weight fractions as described above.

3.2 Known polymers

By default Bob is capable of generating a number of different polymers of fixed architectures and a few reaction schemes. There are three different ways to augment the capability:

1. Add a new routine to model whatever reaction scheme you require.

2. Use a prototype file to define the polymers and let Bob add in polydispersity in runtime.

3. Use some different routine to generate the polymers and let Bob handle the rheology calculation only. Using polymer type 60, you can read in multiple polymer types from different files and can blend with in-built polymers.

While all polymers considered are *model* of real polymers, we reserve the name *model* for polymers where one adds segments in a well defined way. For such polymers, you need to supply the M_W and PDI for each of the segments separately. One exception is the linear polymer. Input for linear polymer consider the M_W of the whole polymer and not of the two segments that are considered internally to make up the polymer.

Table 3.1 summarizes the model polymers and table 3.2 summarizes all other polymers.

polycode	name	input parameters	comment
0	Linear	arm_type, mass, PDI	mass : Mw of linear
1	Star	arm_type, mass, PDI n_arm	mass : Mw of segment n_arm=3,4,6 or 18
2	Asym. Star	long arms : arm_type, mass, PDI short arm : arm_type, mass, PDI	
3	H	side arms : arm_type, mass, PDI cross bar : arm_type, mass, PDI	
4	Combs with Poisson distributed number of arms	backbone : arm_type, mass, PDI side arms: arm_type, mass, PDI n_arm	 average n_arm
5	Combs with fixed number of side armrs	backbone : arm_type, mass, PDI side arms: arm_type, mass, PDI n_arm	 integer n_arm
6	Coupled combs (two comb backbones attached) (at some random position)	backbone : arm_type, mass, PDI side arms: arm_type, mass, PDI n_arm	 average n_arm
10	Cayley tree	num_gen { arm_type, mass, PDI }	0th generation 3 arm star repeated for gen 0, 1 $\cdots$.
11	CayLIN	num_gen { arm_type, mass, PDI }	Cayley tree 0th generation linear.
12	CayST4	num_gen { arm_type, mass, PDI }	Cayley tree 0th generation 4 arm star.

Table 3.1: Polymer types and input formats for model polymers.

polycode	name	input parameters	comment
20	MPE	M_W, b_m	b_m : branch / molecule
21	MPEwtav	M_W, b_m	sampled weight averaged
25	GELwtav	M_N,S, b_u	gelation: segment molar mass upstream branching probability
40	prototype	none	defined in prototype file
60	from-file	file name	read from configuration file(s).

Table 3.2: Polymer types and input formats for other polymers.

In the polymer description, we add snippet of the specific entries in the input file. Of course you need to add the memory requirements and chemistry information in the file. For simplicity let us consider one component polymer melt. The first few lines of the input file may look like:

```
50000  500000
1.0
1
28.0 40.0 0.760
2.0e-7 463.0
1
1.0
```

The meaning of each of the lines:

1. The first line asks for memory large enough to hold 50000 polymers with a maximum of 500000 segments (total). You don't need to actually create this many polymers. It's safer to allocate more space unless you are straining the computer RAM seriously.

2. The second line specifies that the dilation exponent α is 1.

3. The solitary 1 on third line asks to use some default settings.

4. Fourth line gives $M_0 = 28$, $N_e = 40$, $\rho = 0.76\,\mathrm{g/cc}$.

5. Fifth line inputs $\tau_e = 2.0e-7$ and temperature $T = 463\,\mathrm{K}$.

6. Sixth line asks to generate a single component.

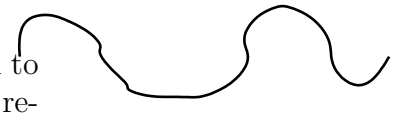
7. Seventh line specifies that the weight fraction of this single component is 1.

The thing missing from this snippet is actual specification of the polymers. That part is shown for each polymer types next. You can add this seven lines to any of them to generate a valid input file.

3.2.1 Type 0 : Linear polymer

To generate linear polymers, you need to choose a distribution for the molar mass, the weight averaged molar mass, and the PDI. The entry in the input file may be:

```
100 0
2 20000 1.4
```

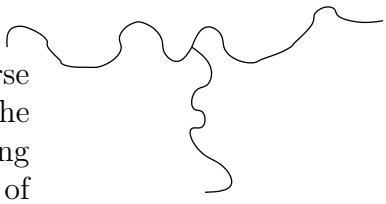


The first line (actually line 8, because you need to add the general description about the memory requirement and chemistry information) asks to generate 100 polymers of type 0. Type 0 signifies linear polymers. The last line says that the distribution is of type 2 (log-normal), $M_W = 20000\,\mathrm{g/mol}$ and PDI = 1.4. (The different distributions from which arms can be sampled are in section 3.3).

3.2.2 Type 1 : Star polymer

The entry is similar to that for linear. But the M_W and PDI refer to individual arms making up the star.

```
1 1
0 25000 1.0
3
```

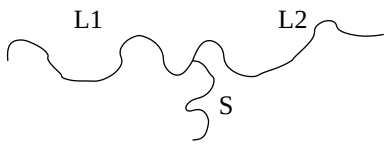


Here, we have asked for star with monodisperse (arm_type 0) arm of $M_W = 25000\,\mathrm{g/mol}$. The PDI entry is required but not used - so entering anything else instead of 1.0 will create the same of polymers. The last line says that the star polymer generated should have 3 arms. You can use 4, 6, or 18 to have that many arm star polymers. Since we have asked for monodisperse arms, nothing is gained by creating more copies of the polymer - so the first line asked for a single polymer.

3.2.3 Type 2 : Asymmetric star polymer

Asymmetric stars are thought to have two arms of equal length and the third arm having a different length. Only 3 arm stars are created. The input file entry can be

```
100 2
2 25000 1.50
0 5000 1.0
```

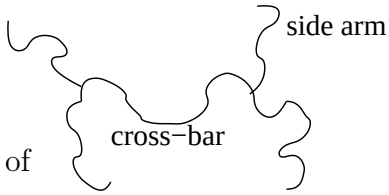


Here, we asked to create 100 star polymers, each with two of the symmetric arms created from log-normal distribution with $M_W = 25000\,\mathrm{g/mol}$ and PDI = 1.5. The asymmetric arm is monodisperse with $M_W = 5000\,\mathrm{g/mol}$.

3.2.4 Type 3 : H polymer

H polymers have one cross-bar and four segments attached to the crossbar. The input entry:

```
200 3
2 15000 1.50
2 25000 1.20
```



will generate 200 H polymers with side arms of $M_W = 15\,\mathrm{K}$ and PDI = 1.5 attached to a cross bar of $M_W = 25\,\mathrm{K}$ and PDI = 1.2. All arms are sampled from log-normal distributions.

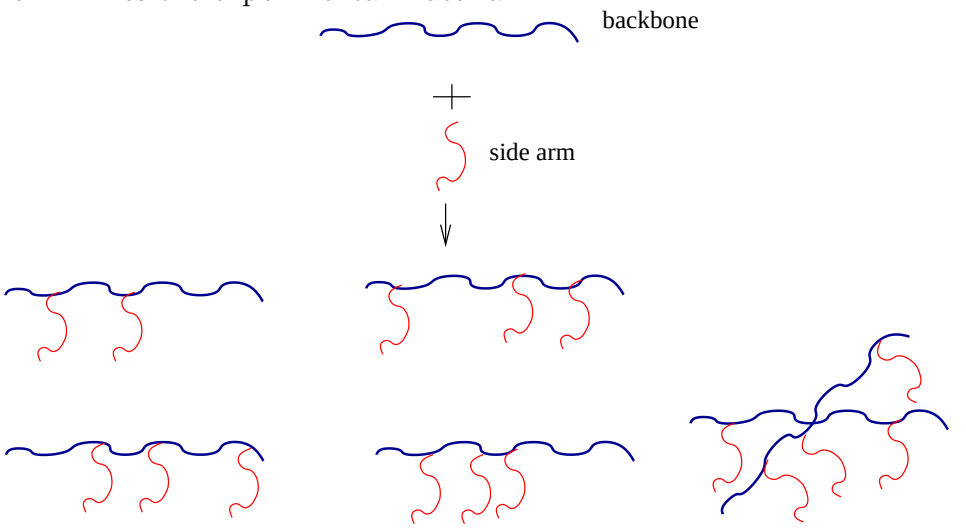
3.2.5 Type 4, 5, 6 : Comb polymers

The Comb polymers typically are synthesized by separate synthesis of the backbone and the side arms. The backbone is functionalized to attach the side arms. The active sites at which the side-arms attach are randomly placed on the backbone. In typical synthesis procedure, there is little control on number of side arms that attaches on a given backbone. A simple model would consider the final product as having Poisson distributed number of side arms connected at random places on the backbone. The distribution is chosen to give same mean number of side arms as observed in analytical

experiments. Thus, the average number of arms need not be an integer.

To compare with theories, or to observe the effect of different number of side arms, one might be interested in keeping the number of arms fixed to some integer value. Still, the different comb molecules generated will have different rheological signature because of the random placement of the side arms on the backbone.

During synthesis, there is a possibility of two of the backbone material reacting with each other, generating a *coupled comb*. Careful analytical techniques can resolve the amount of coupled material in the product and one can blend *pure* comb and couple comb polymers to generate an ensemble which mimics the experimental material.



Type 4

```
200 4
2 75000 1.50
2 15000 1.20
4.2
```

Generate combs with 75 K backbone, 15 K side arms and on average 4.2 side arms per molecule.

Type 5

```
200 5
```

```
2 75000 1.50
2 15000 1.20
4
```

Same as before, except have exactly 4 side arms per molecule. Number of arm for this type is an integer.

Type 6

```
200 6
2 75000 1.50
2 15000 1.20
4.2
```

Generate comb polymers as in the example for type 4 and attach two such combs at some random point along the two backbones.

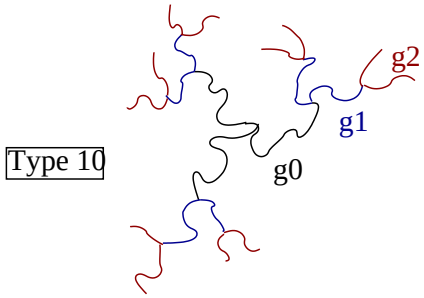
3.2.6 Type 10, 11, 12 : Cayley tree

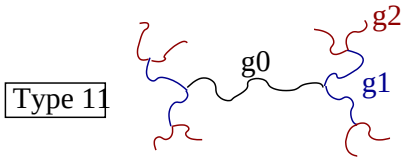
Cayley tree are polymers with some inner core (generation 0), which in subsequent generations branch out from every free ends. The implemented Cayley trees can have a 3 arm star (type 10), a linear (type 11), or a 4 arm star (type 12) as the generation 0. The input are number of generations and specifications for each generations.

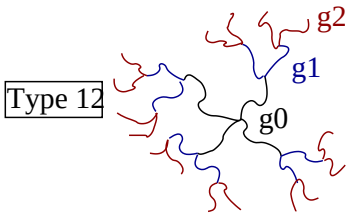
Type 10/11/12

```
200 X
2
2 75000 1.50
2 45000 1.20
2 15000 1.20
```

Here, `X` is one from {10, 11, 12 }. There are 3 generations, with generation 0 (the inner core) having 75 K molar mass. If `X` is 10, the inner core is a 3 arm star. If `X` is 11, the inner core is a linear segment. And, for `X` 12, the inner core is a 4 arm star.
The subsequent two generations have molar masses 45 K and 15 K.







3.2.7 Type 20, 21 : MPE

Metallocene catalyzed polyethylene (MPE) is one of the industrial polymers with very simple theoretical description. The final product is described by just two parameters: weight averaged molar mass M_W and average number of branches per molecule b_m.

In this reaction scheme, monomers get added along the *downstream* direction. A randomly picked segment will have different probabilities of encountering a branch point in the downstream direction b^D and the *upstream* direction b^U (the direction opposite to monomer addition direction along the polymer). The segments between branch point has a number-averaged molar mass $M_{N,S}$. Experiments like NMR can calculate the average number of branch points per 10^3 carbon atoms (λ).

But as mentioned earlier, only two parameters describe the full ensemble.

The relationship among the parameters are:

$$M_{N,S} = \frac{M_W}{2(b_m+1)(2b_m+1)}, \tag{3.1}$$

$$\lambda = \frac{14000}{M_{N,S}}\left(\frac{b_m}{2b_m+1}\right), \tag{3.2}$$

$$b^D = 2b^U, \tag{3.3}$$

$$b_m = \frac{b^U}{1-2b^U}. \tag{3.4}$$

Since the full distribution for this system is known, we can sample the distribution biasing in the region of interest. The weight fraction of the molecules so generated, takes care of compensating for the bias introduced in the sampling. Both these schemes are implemented in bob - type 20 generates MPE with number-based sampling, while type 21 generates MPE in weight-based sampling.

The input :

```
2000 X
86000 0.1160
```

with `X` either being 20 or 21 generates MPE molecules. The generated ensemble will contain a blend of molecules: linear, star, comb, branch-on-branch molecules $\cdots$.

3.2.8 Type 25 : Gelation ensemble

Consider linear polymers with functionalized ends (molecules **A**) and trifunctional groups (**B**). If they are reacted in such a way that always **A** reacts with **B** but neither of **A-A** or **B-B** reactions are allowed, as a function of the extent of the reaction, the mixture shows a sol-gel transition. Progressively larger molecules are created and both the molar mass distribution and dynamic properties show power-law behaviors. Since reaction progresses on both ends of the segments, the probabilities of encountering a branch point on either direction are the same.

In the implementation in bob, we consider that the linear molecules are Flory distributed and the trifunctional groups have negligible molar mass compared to the linear molecules. So, the only parameters for the distribution

are the number averaged molar mass of the linear segments $M_{N,S}$ and the branching probability p. The sampling is done in weight biased scheme.

The input file can be like:

```
2000 25
15000 0.2
```

Note that the gel point is at $p_c = 0.5$ - as you approach p_c from below, there will be progressively larger molecules. You will need large number of molecules and memory to get sensible results beyond $p \sim 0.3$. The molar mass for the linear segments in the input file is number-averaged. Since Flory distribution will generate a PDI=2, the $M_{N,S}$ is half the weight averaged molar mass of the segments.

3.2.9 Type 40 : Polymer prototype from file

When Bob encounters polymer type 40, it looks for a file called "poly.proto" in the current directory. You can use several instances of type 40. Each time a single polymer definition is read and replicated with the required distribution to add polydispersity.

Each polymer starts with a maximum 10 character tag to identify the polymer. Next line is a single integer telling the number of arms. For each arm, the next lines contain four integers describing left and right connectivity, type of random distribution to add polydispersity, **molar mass**, polydispersity index. Note that, here we need actual molar mass of the segment in g/mol and not the entanglement length.

As an example, let us create a set of star polymers with the three arms created from different reaction schemes and functionalized so as to only attach to one each from the other two ensemble. The content of "poly.proto" may be:

```
FunStar
3
-1 -1  1  2  0  10000  1.0
 0  2 -1 -1  2  12000  1.4
 0  1 -1 -1  4  18000  2.0
```

Here, the arm with index 0 has molar mass 10000 g/mol and is completely monodisperse. The arm with index 1 is with Mw=12000 g/mol and a lognor-

mal distribution is chosen to get a polydispersity index of 1.4. The arm with index 2 has Mw=18000 g/mol and is generated from a Flory distribution.

3.2.10 Type 60 : From configuration files

You can use this type to supply pre-generated polymers (possibly through some other program). The format must match bob's own format - in particular, the weight fractions should add up to one in each file.

Bob multiplies the weight fractions specified in the configuration file by the weight fraction in the input file to make blends. Consider the following input file (the full input file for a change)

```
50000 500000
1.0
1
28.0 40.0 0.760
2.0e-7 463.0
3
0.3
100 0
2 20000 1.4
0.3
100 60
poly1.dat
0.4
100 60
poly2.dat
```

This asks for a blend of three polymers. The first is a linear polymer occupying 30% of the weight. The next two are read from files 'poly1.dat' and 'poly2.dat' and assigned weight fractions 0.3 and 0.4 respectively. The number of polymers in the files need not match the number in the input file for type 60. The code will read the actual number of polymers from the file. But because the input is in fixed format, you need to supply an integer place holder for number of polymers in the input file. After reading the file name from the input file, the program strips any white space (space or tab). So, the file name can not have any white space in it.

3.3 Segment length, M_W, M_N and PDI

The number averaged molar mass of N polymers with individual masses M_i

$$M_N = \frac{\sum_i M_i}{N}. \tag{3.5}$$

The weight averaged molar mass

$$M_W = \frac{\sum_i M_i^2}{\sum_i M_i}. \tag{3.6}$$

The polydispersity index

$$\text{PDI} = \frac{M_W}{M_N}. \tag{3.7}$$

Input to bob requires M_W (in gm/mol) and where applicable PDI. For thinking about random distribution, it is helpful to think in terms of mean $\mu =< M >$ and variance

$$\sigma^2 =< M^2 > - < M >^2 \equiv \mu^2 \left(\text{PDI} - 1\right). \tag{3.8}$$

There is a subtlety about PDI - the program generates arms with a given polydispersity. The PDI of the whole polymer will not be same as that of an individual arm. In some cases, one can have reasonable estimate of arm polydispersity from the PDI of the whole polymer (some discussion follows in the subsections on individual distributions). In general, the inelegant but workable way will be to use *GenPolyOnly = yes* and *CalcGPCLS = yes* in *bob.rc* file, tune arm polydispersities, generate polymers and read out the PDI from *info.txt*. Repeat the whole process till the polymers have the required PDI.

The inbuilt distributions that are available

type	description
0	monodisperse
1	Gaussian
2	Log-normal
3	Poisson
4	Flory distribution

When the polymer information is given through input file, you need to specify PDI for all distributions. For some cases, the PDI will be ignored.

3.3.1 Monodisperse segments

When accessed through input file, specification of monodisperse segments require a PDI, which is not used in the calculations.

3.3.2 Gaussian distribution

Highly monodisperse polymers can be approximated as having a Gaussian distribution. For large PDI, this is not appropriate, because some of the segments will be negative. The code puts a small positive value for such arms - the resulting distribution will have a peak at small mass.

For Gaussian distribution, estimating the distribution of the polymers from segment distribution is easy. The M_N for the whole polymer is sum of the segmental M_N. Same is true for the variance. In terms of PDI

$$\mathrm{PDI}_{poly} = 1 + \frac{\sum_i^{segment} M_{N,i}^2 \left(\mathrm{PDI}_i - 1\right)}{\sum_i^{segment} M_{N,i}^2}. \tag{3.9}$$

3.3.3 Log-Normal distribution

Without some insight about the synthesis process, this probably will be the distribution of choice. The distribution

$$p(x) = \frac{1}{xs\sqrt{2\pi}} \exp\left(\frac{-(\ln x - m)^2}{2s^2}\right) \tag{3.10}$$

has mean of $\ln x = m$ and variance of $\ln x = s^2$. The distribution is a normal distribution for the variable $\ln x$. The mean and variance of x itself are $\mu = \exp(m + s^2/2)$ and $\sigma^2 = \mu^2(e^{s^2} - 1)$. In terms of PDI and M_N

$$\begin{aligned} s^2 &= \ln\left(1 + \frac{\sigma^2}{\mu^2}\right) \equiv \ln \mathrm{PDI} \\ m &= \ln M_N - \frac{\ln \mathrm{PDI}}{2}. \end{aligned} \tag{3.11}$$

We use a normal distribution with this mean m and variance s^2. The exponential gives the required log-normal distribution for polymer segments.

When the polymer is made of k segments with all segments from log-normal distributions, the whole polymer can be approximated as being from another log-normal distribution. If mass and PDI of all the segments are the

same, the number averaged molar mass of the whole polymer is just the sum of the k segmental M_N. The PDI of the polymer is given as

$$\mathrm{PDI}_{\mathrm{poly}} = 1 + \frac{\mathrm{PDI}_{\mathrm{seg}} - 1}{k}. \tag{3.12}$$

To get some appreciation, if you add five segments of same mass and same PDI 1.4 to form a H-polymer, the resulting polymer will have a PDI of just 1.08.

3.3.4 Poisson distribution

Living polymerization is expected to generate polymers from Poisson distribution with PDI $\sim 1/\sqrt{N}$, where N is the number of monomers in the polymer. Often the actual PDI is much larger in the synthesis. One possible way to accommodate this large PDI is by assuming that a bunch of monomer is added in each step. The PDI and the M_W is used to calculate the effective size of the bunch and a Poisson distribution is returned.

3.3.5 Flory distribution

When the monomer concentration is kept constant during polymerization and there is no chain transfer or termination by disproportionation, we can think of a probability p for monomer addition and $(1-p)$ for termination. The probability for a n-mer is $(1-p)p^n$. The number averaged molar mass,

$$M_N = \frac{M_0}{1-p}, \tag{3.13}$$

with the monomer molar mass M_0. The weight averaged molar mass

$$M_W = M_0 \left(\frac{1+p}{1-p} \right). \tag{3.14}$$

Typically p is close to 1, giving PDI$= 1 + p \simeq 2$. When this distribution is used in Bob, the supplied PDI is used to convert M_W to M_N. M_N decides p and the resulting segments will have PDI close to 2.

For p close to 1, $\ln p \simeq -(1-p)$. Using this approximation, the Flory distribution is generated from an uniform random number $r \in [0, 1]$ by calculating $\ln r / \ln p$.

Chapter 4

Output

The output from bob are a set of files. The output format is selected through the file *bob.rc* in the current directory by setting *OutMode=X* where *X* can be 0 (default choice), 1 or 2.

When **OutMode=0**, the output are in headerless ascii files with .dat extension. If **OutMode=1**, the data will be presented with .agr extension in a format which when opened with xmgrace will show the plots. If you have a UNIX like system (including cygwin), probably you also have xmgrace installed on your system. Else, the website for xmgrace is `http://plasma-gate.weizmann.ac.il/Grace`. If **OutMode=2**, data are in a format which are meant to be read with a limited availability software *Reptate*. If you have the software, you should have a description of what the extensions in this case mean. So, this document will ignore trying to describe the output in this mode. If you are using MS-windows, use Wordpad or some other editor which is aware of unix end-of-line character to read the files. Notepad is not aware of this format.

4.1 Run information summary

The file *info.txt* keeps a summary of the input parameters chosen and the calculation. The start of the file looks like

```
Found bob.rc
OutMode = 0
PSquare = 2.500000e-02
Alpha = 1.000000e+00
 :  :  :
End of rc file

 parameters after reading inputs:
alpha  = 1.000000e+00
R_L  = 1.000000e-05
p^2  = 2.500000e-02
Mass of monomer  = 2.800000e+01
 :  :  :
compound arm prefactor includes effective armlen.
Reptation in thin tube.

 End of input parameters
```

This records the choices made for the relaxation mechanism and parameters for the calculation through a combination of inputs from *bob.rc*, from input file or from interactive mode.

The next few lines

```
Selected 3 arm Star from Lognormal distribution
   with M_w = 2.226000e+04 and PDI = 1.400000e+00
created 5000 Star
component  0 : Total mass = 2.124985e+05
GPC module for component 0  :
Mw = 5.381942e+04 ,   Mn =  4.759966e+04, PDI = 1.130668e+00
```

record about the polymers generated. In this case we had asked for 5000 star polymers with each segments having $M_W = 22260\,\mathrm{g/mol}$ and PDI $= 1.4$. Note that in general the final product will have very different PDI compared to the segments. The GPCLS module puts in the actual M_W and M_N of the polymers generated in each component of the blend.

The next few lines does a descriptive analysis of the full blend (single component in this case)

```
maximum priority = 1
 maximum seniority = 1
Branching topology of the polymers
Number of branches per molecule = 1.000000e+00
number of linear molecules = 0
mass fraction of linear molecules = 0.000000e+00
number of star molecules = 5000
mass fraction of star molecules = 1.000000e+00
number of comb molecules = 0
mass fraction of comb molecules = 0.000000e+00
number of branch-on-branch molecules = 0
mass fraction of branch-on-branch molecules = 0.000000e+00
```

In this particular case, we have only 3-arm star molecules. So, number of branches per molecule is just 1 and no branch-on-branch material are present.

The final line notes the zero shear viscosity from the calculation:

```
zero-shear viscosity = 5.124211e+05
```

4.2 Linear rheology

Linear rheology data are in three different files (i) *supertube.dat* notes amount of unrelaxed material as a function of time, (ii) *gtp.dat* or *gtp.agr* has frequency dependent relaxation modulus, and (iii) *gt.dat* or *gt.agr* has relaxation modulus as a function of time.

4.2.1 supertube.dat

This is ascii file containing four columns which looks like:

```
0.000000e+00 1.000000e+00 1.000000e+00  1.000000e+00
1.000000e-04 1.000000e+00 9.995004e-01  9.965458e-01
1.001000e-04 1.000000e+00 9.990010e-01 9.965450e-01
1.002001e-04 1.000000e+00 9.985019e-01 9.965441e-01
1.003003e-04 1.000000e+00 9.980030e-01 9.965433e-01
  :    :    :
```

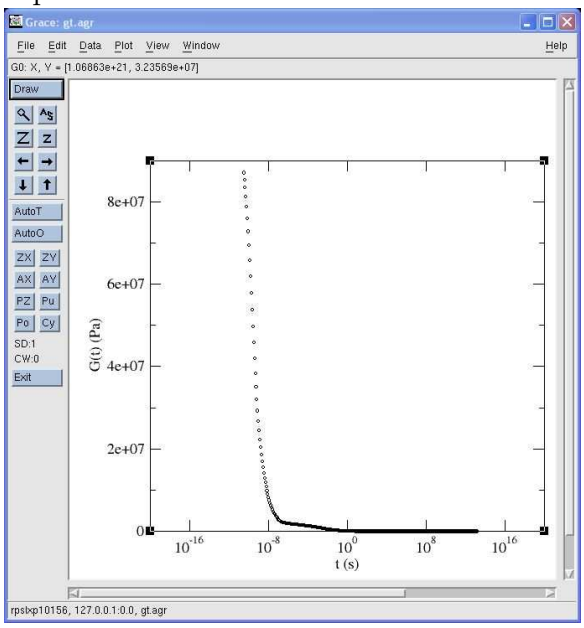
The first column is the time in units of τ_e. The second column ϕ is the effective fraction of unrelaxed material that relaxing segments feel. The

third column ϕ_{ST} is the effective fraction of unrelaxed material that drives the supertube relaxation. And finally the fourth column ϕ_t is the true amount of unrelaxed material. For more details about why we need all three related, but not necessary same number, look at the theoretical background part of the document.

4.2.2 gt.dat/gt.agr

The relaxation modulus $G(t)$ is stored in *gt.dat* or *gtp.agr*. *gt.dat* is an ascii file with two columns. The first column gives time t in seconds and the second column gives the relaxation modulus $G(t)$ is Pascal.

If *OutMode* were set to 1, the output file will be gt.agr. When opened with xmgrace, the plot in this case looks like as the following screenshot:



xmgrace is a powerful graphics package which will allow you to fit, import experimental data to compare with the prediction, add text, zoom $\cdots$. Most importantly it will generate extremely good quality output in a large number of graphics formats.

4.2.3 gtp.dat/gtp.agr

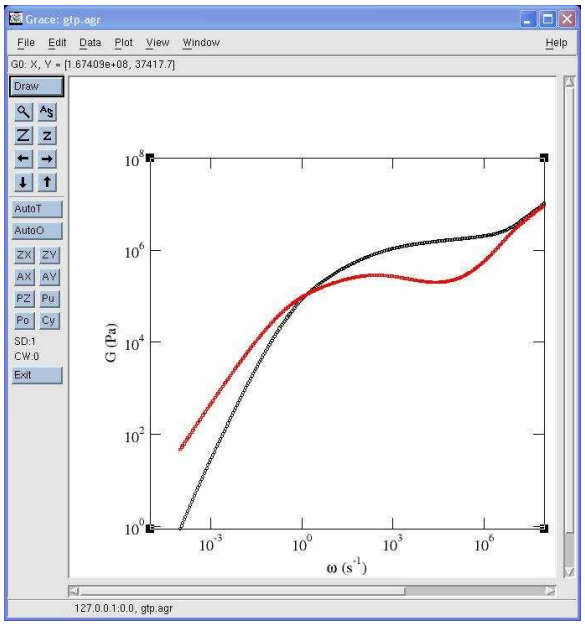
The dynamic moduli results are in *gtp.dat* or *gtp.agr*, as directed by the *OutMode* in *bob.rc* file.

gtp.dat contains three columns:

```
1.000000e+08 1.041311e+07 9.408720e+06
9.090909e+07 9.930355e+06 8.967337e+06
8.264463e+07 9.470043e+06 8.545483e+06
7.513148e+07 9.031150e+06 8.143009e+06
  :    :    :
```

The first column is frequency ω in rad/s. The second and third columns respectively are the elastic modulus $G^{'}$ (ω) and the viscous modulus $G^{''}$ (ω).

If you had set *OutMode=1*, the file is *gtp.agr*. `xmgrace gtp.agr` will pop up a window with the plot, which in this case looks like the following screenshot:



4.3 GPCLS

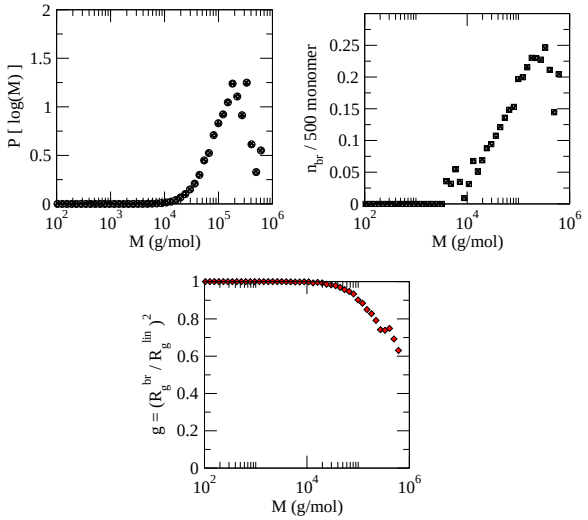
If you have set *CalcGPCLS = yes* in *bob.rc*, for each component the program will calculate the probability of having a particular value of mass (actually log of the mass). You get similar information from GPC (gel permeation chromatography). Once *CalcGPCLS* is set, it calculates the M_W, M_N and PDI of each component and stores this information in the file *info.txt*. If the polydispersity is appreciable, then it generates a series of files with names

gpcls1.dat, *gpcls2.dat* $\cdots$ for component 0, 1 $\cdots$. And for more than one component one final file *gpclssys.dat* for the entire blend. The files contain four columns, mass M (in g/mol), probability of having that mass $P[\log(M)]$, number of branches / 500 monomers for polymers having that mass n_{br} and the ratio of the square of radius of gyration of polymers of that mass to radius of gyration of linear polymer having the same molar mass $g \equiv R^2_{g,br}/R^2_{g,Lin}$. n_{br} is per 500 monomers, so that for polyethylene it matches with number of branches per 1000 carbon atoms, the preferred literature unit. g is assuming Gaussian statistics of the chains. Radius of gyration depends on the solvent quality. But when expressed as a ration of linear polymers of the same molar mass, the quantity g is rather insensitive and the ideal output from Bob can be compared directly with experimental results.

To have the output more interesting, we created a blend of two metallocene catalyzed polyethylene (type 20) with the input file:

```
15000 500000
1.0
1
28 40 0.785
1.0e-8 428.0
2
0.5
5000 20
68000 0.340
0.5
5000 20
108000 0.340
```

The plot of *gpclssys.dat* gives:



4.4 Error and warning messages

Occasionally bob will throw out error or warning messages - if the run is in the interactive mode, these would print messages on the run window. If the run is non-interactively, the output will be redirected to a file called *bob.err*.

Index