LP2R

Generated by Doxygen 1.9.2

Chapter 1

Linear rheology of linear polydisperse polymers

28/09/2022

1.1 Introduction

The code in this depository accompanies our paper *"A tube model for predicting the stress and dielectric relaxations of polydisperse linear polymers"* submitted to *Journal of Rheology* (2022). This implements modern ideas about how constraint release and tube escape modes in linear polymer melts affect each other in a numerical code to predict linear respose in arbitrarily polydisperse linear polymers. The information about the polymers can be supplied as moments of a distribution, or as data files containing gel permeation chromatography measurements (GPC), or a set of discrete molar masses and associated weight fractions. Arbitrarily complex blends can be designed by adding several such components. Besides the mechanical relaxation moduli, the code also can calculate dielectric relaxation for type-A polymers (polymers with monomer dipole moments aligned along the chain backbone). With appropriate instructions, the code outputs evolutions of model constructs like different dynamic tube diameters or how a specific molar mass chain will undergo the terminal relaxation.
The code is available for download from `github`. On UNIX/Linux systems, you can use *make* command from *LP2R/src/obj* subdirectory to create the executable. A precompiled windows executable is available from `google drive`. A snapshot of the code at the time of submission of this paper including the submitted preprint is available at `zenodo.`
The rest of this page documents the command line options, input file syntax, and output file formats.

Go to **Table of Contents**
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.2 Usage

```
LP2R [-L] [-d] [-i inputfile] [-r resourcefile] [-h] [--version]
```

Optional command line arguments:

- **-L** : Output debugging and extra information in a file named ***LP2Rlog.txt***

- **-i** inputfile : material and output frequency information supplied via a named file. Default inputfile name is ***inp.dat***

- **-r** resourcefile : Presumably chemistry independent parameters, and output options can be set via a resource file. Default resourcefile name is ***LP2R.rc***

- **-h** : help (this usage information)

- **- -version** : version information

Go to Table of Contents
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.3 Input files

The code expects input to be supplied from plain text files. The percentage (%) character is used as a comment character in the files. If the first non-space character in a line is %, the entire line is ignored. Similarly a % sign can be used beyond the intended input in the same line to add comments. The files are read one line at a time - so, if multiple entries are supposed to be in the same line, a line break between entries will lead to input error. The input file itself may contain names of some other file containing detailed molecular weight information. In such cases, names containing space or percentage sign will not be processed correctly. Avoid names like **"Molecular Weight.txt"** or **"PI100k(2%)_PI50k(98%).txt"**: The first name will be truncated to ***"Molecular"*** and the second will be truncated to ***"PI100k(2"*** and the code will fail to find the intended data file.
Two different files are used for input: The first should supply the material parameters and the frequency range over which relaxation spectra is desired. The default file name for this file is ***inp.dat*** . A different file can be supplied with the command line option ***-i filename*** . Details of the information required in this file are described in Material input file .
The second optional file supplies model parameters that are thought to be insensitive to the chemistry, and can be used to choose the results that the code should generate. The default file name for this resource file is ***LP2R.rc*** . A different file can be supplied with the command line option ***-r filename*** . Details of the information required in this file are described in Resource file .

Go to Table of Contents
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.3.1 Material input file

The information about the polymers is supplied via a plain-text file called **inp.dat** (a different file can be used by using the command line option ***-i filename***). The input can be imagined as having three distinct parts: the first part gives the frequency range and discretization for the relaxation spectra output, the second part inputs the material parameters (chemistry dependent but architucture independent parameters), and finally the third part inputs the information about the molar mass distribution(s) of the molecules of interest.
The first valid line (the code will ignore comment lines or empty lines) needs the minimum angular frequency (ω_{min}), the maximum angular frequency (ω_{max}) and the ratio of successive angular frequencies (ω_{ratio}) for the desired relaxation spectra output.
The second valid line requires the Kuhn molar mass (M_{Kuhn}) in g/mole, entanglement molar mass (M_e) in g/mole, plateau modulus (G_N^0) in Pa, and the entanglement time (τ_e) in seconds. The third line gives the glassy modulus (G_∞) in Pa, glassy relaxation time (τ_g) in seconds and the exponent for the stretched exponential glassy relaxation (β_g).
The fourth line contains a single integer specifying the number of components (n_{comp}) forming the blend. (In the special case of single component polymer, this number is one). A *component* is understood to be a set of *molecules* having a easily described distribution. For each of these components, additional two input-lines are given to characterize the components. The first of thse lines contain an integer parameter (p_{type}) and the weight fraction of this

component (w_{comp}) in the blend. If **p_{type}=0**, the component is assumed to be represented by a log-normal distribution. The following line in that case needs number of discrete molar masses to be used to represent the distribution (n_{poly}), the weight-averaged molar mass (M_W) in g/mole and the polydispersity index (PDI). Instead if, **p_{type}=1**, the molar mass distribution is assumed to exist as a GPC measurement and the second line of component specification in that case gives the file name of the GPC data. The GPC file should have usual {M, dW/dlog$_{10}$M } values with the molar mass in g/mole. Finally, **p_{type}=2**, assumes that the molar mass is specified by a set of weights associated with discrete sets of molar mass. Again, the second line of component specification is a file name and the file should contain two columns: molar mass in g/mole and the associated weights {M_i, w_i }.
The following is an input file for 1,4-PI at 25℃ for a 1131000 g/mole polymer assumed to be described by a log-normal distribution: 1.0e-4 1.0e6 1.2 % ω_{min}, ω_{max}, ω_{ratio}
113.0 4350.0 476000.0 1.30e-5 % M_{Kuhn}, M_e, G_N^0, τ_e
1.0e9 7.0e-11 0.370 % G_∞, τ_g, β_g
1 % Single component: n_{comp}=1
0 1.0 % p_{type}=0 (log-normal distribution), w_{comp}=1
50 1131000 1.05 % n_{poly}=50, M_W, PDI

Blending 30-wt% 226kg/mole polymer would require an input file like the following 1.0e-4 1.0e6 1.2 % ω_{min}, ω_{max}, ω_{ratio}
113.0 4350.0 476000.0 1.30e-5 % M_{Kuhn}, M_e, G_N^0, τ_e
1.0e9 7.0e-11 0.370 % G_∞, τ_g, β_g
2 % **Two components :** n_{comp}=2
0 0.70 % ***First component:*** p_{type}=0 (log-normal distribution), w_{comp}=0.70
50 1131000 1.05 % n_{poly}=50, M_W, PDI
0 0.30 % ***Second component:*** p_{type}=0 (log-normal distribution), w_{comp}=0.30
50 225900 1.03 % n_{poly}=50, M_W, PDI

Some more examples of input files can be found in the examples subdirectory in the code distribution.
Some pointers about the input:

• A strictly monodisperse polymer can be specified by choosing a log-normal distribution (p_{type}=0) and PDI=1, or n_{poly}=1 in the specification of the distribution.

• If the glassy parameters are not be available for the chemistry of interest, a good starting point may be G_∞=1.0e9, τ_g=1.0e5 × τ_e, and β_g=0.35.

Go to Table of Contents
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.3.2 Resource file

The code searches for a file named ***LP2R.rc*** in the current directory to input additional parameters besides the material parameters. A different file can be specified with command line option ***-r filename*** . In the absence of this file, the code will continue with default parameters and options detailed below. Similarly any entry in the file that is not understood by the code is silenty ignored. If you use the command line option ***-L*** to generate a log file, such unresolved entries will be written in the log file as warning messages. Each entry in the resource file is a property-value pair separated by a equal (=) sign. Each line can contain only one such property-value pair. Unlike the material input file, the ordering of the different entries in the file does not have any consequence.
A sample file with all options set to default values is available in the examples/rcdefault subdirectory.
% Model parameters
Alpha=1.0 % Dilution exponent α
t_CR_START=1.0 % Constraint release starts after this time (in units of τ_e)
deltaCR=0.30 % Fractional drop in φ_{ST} for τ_{CR} » τ_e (δ_{CR}^∞)
B_zeta=2.0 % Proportionality constant relating friction coefficient to supertube fraction (B_ζ)
A_eq=2.0 % Proportionality constant connecting "effective equilibrium time" and time to locally equiilbiate in a certain supertube (A_{eq})
B_eq=10.0 % Constant delaying equilibrium for fast CR events (B_{eq})
ret_pref=0.189 % Constant in arm retraction formula ($C_{a,\infty}$)

ret_pref_0=0.020 % Short-time prefactor for CLF ($C_{a,0}$)
ret_switch_exponent=0.42 % Exponent ε_a determining how steeply CLF switches to long-time strength
Rept_Switch_Factor=1.664 % Constant deciding transition from CLF to reptation (K_R)
Rouse_Switch_Factor=1.5 % Minimum number of bare entanglements to be considered entangled (Z_U)
Disentanglement_Switch=1.0 % Number of surviving entanglements in the supertube below which chains relax by "disentanglement"

% Time discretization
Start_time=1.0e-3 % Start of integration (in units of τ_e)
Time_ratio=1.02 % Ratio of successive discrete times

% Options for results
CalcDielectric=no % "yes" asks for dielectric relaxation spectra.
OutTermRelaxPathways=no % "yes" outputs individual chain relaxation modes.
OutPhiPhiST=no % "yes" outputs evolution of different tube diameters
Output_G_of_t=no % "yes" asks for time relaxation of modulus

% Control of output
OutputFormat=Default % Other options are "Text", "CSV" and "RepTate"
CSVdelimiter=, % For *OutputFormat=CSV* you can specify a different delimiter than usual comma (,)
Add_header=yes % "no" does not add header line in the output files
label= % You can specify a string as a label (default is an empty string). The label will be used in output file names.
chem= % You can specify a string as chemistry (default is an empty string). If specifed and *Output↩ Format=RepTate* , this will be added in the output headers.
origin= % You can specify a string as origin (default is an empty string). If specifed and *OutputFormat=RepTate* , this will be added in the output headers.
Temp=0.0 % You can specify the temperature (in degree centigrades). If *OutputFormat=RepTate* , this will be added in the output headers.

Go to **Table of Contents**
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.4 Output files

By default the code only outputs the mechanical relaxation spectra G'(ω) and G" (ω). In addition, you can ask it to output the dielectric relaxation spectra ε'(ω) and ε"(ω) (assuming that the dipoles are aligned along the chain backbone), time domain mechanical relaxation G(t), evolution of different relevant tube constraints after an instantaneous small shear deformation, and assignment of the terminal relaxation pathway for each chain in the ensemble. These additional outputs are initiated by setting the appropriate flags in the resource file to *yes* . The different outputs are directed to different files with the contents somewhat dependent on the *OutputFormat* variable set from the resource file. If the *label* option is set in the resource file, the files include the label as part of their names.
If *OutputFormat=Default*, the output file names have ***.dat*** extensions. Unless the option *Add_header* is not set to *no* in the resource file (the default behaviour is *Add_header=yes*), the first line will be a header line starting with a hash (#) character (default comment option for many UNIX plotting softwares). The choice *OutputFormat=Text* behaves similarly as the *OutputFormat=Default* option, except that the filenames end with ***.txt*** extensions and the header line, if not switched off with *Add_header=no* in the resource file, start without a # character. *OutputFormat=CSV* behaves like *OutputFormat=Text* except the entries are separated by a comma (,) (or, any other character chosen via *CSVdelimiter* in the resource file) and the file names end with ***.csv*** extensions. *OutputFormat=RepTate* uses RepTate specific extensions and data formats for the relaxation spectra and the time relaxation function. For other outputs, it uses the same outputs as the *OutputFormat=Text* choice. The headers for the relaxation spectra and the time relaxation function in this case contain temperature that can be set via the variable *Temp* in the resource file. In absence of a supplied value, a default temperature of 0℃ will be added to the headers. Additional options *chem*, *label*, and *origin* can be set via the resource file and they will be added in the RepTate format output headers. The subsections below document the different outputs separately.

Go to **Table of Contents**

⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.4.1 Relaxation spectra outputs

By default, the code always outputs G'(ω) and G"(ω) in an range of angular frequencies ω_{min} and ω_{max} with the ratio of subsequent frequencies being ω_{ratio}. These frequency information is set in the first line of the material input file. In addition, dielectric relaxation output is given if either ***-d*** option is used at the command line or there is an entry *CalcDielectric=yes* in the resource file.
Except for *OutputFormat=RepTate*, if the varialbe *label* is not set, the releaxation spectra output is written in a file **RelSpec.extn** with *extn* is either *dat* , *txt*, or *csv* depending on the setting for the *OutputFormat*. If the *label* variable is set, for example *label=PS112k* , the file name instead will be set to **RS_PS112k.extn** with appropriate choices of *extn*. The entries in the file are **1.** frequency ω, **2.** dynamic storage modulus G'(ω), **3.** dynamic loss modulus G"(ω), **4.** dynamic viscosity η" (ω), **5.** dielectric storage modulus ε'(ω), **6.** dielectric loss modulus ε"(ω). The dielectric information (columns 5 and 6) are only present if appropriate instruction is given either via the command line flag or via the resource file.
If *OutputFormat=RepTate* is chosen, the mechanical output is written in **MechSpec.tts** in the absence of *label* variable in the resource file, and, if asked for, dielectric response in file **DiSpec.dls** . If *label* is set, for example as *PS112k* , the file names will be **PS112k.tts** and **PS112k.dls** respectively. The contents of the mechanical relaxation file are the **1.** frequency ω, **2.** dynamic storage modulus G'(ω), **3.** dynamic loss modulus G"(ω) and those of the dielectric file are the **1.** frequency ω, **2.** dielectric storage modulus ε'(ω), **3.** dielectric loss modulus ε"(ω).

Go to **Table of Contents**
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.4.2 Time relaxation

Time domain decay of modulus after a step strain is calculated if *Output_G_of_t=yes* option is set via the resource file (the default is *Output_G_of_t=no*). As with relaxation spectra, file name extensions for *OutputFormat=Default*, *Text* and *CSV* are respectively .dat, .txt, and .csv. In the absence of *label* in the resource file , the data is written in a file named **MechRel** with appropriate file extension. The data are set of time t, modulus G(t), tube survival probability μ(t), and the constraint release contribution R(t). With *OutputFormat=RepTate*, time t and modulus G(t) is written in file **MechRel.gt**. With the *label* variable set, for example as *label=PS100k*, the file name for *OutputFormat=RepTate* choice will be **PS100k.gt**. With the same label, *OutputFormat=Default* will use a file **MR_PS100k.dat** . Other choices will change the file extension appropriately.
The range of time for the time relaxation output is chosen to be between 10^{-4} τ_e and 10^4 τ_d, with τ_d being the longest relaxation time. These times can only be changed by editing *src/Rheology/Calc_goft.cpp* in the source code.

Go to **Table of Contents**
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.4.3 Tube diameters

If *OutPhiPhiST=yes* option is set via the resource file (the default is *OutPhiPhiST=no*), different dynamic tube diameters ($\mathbf{a_X}$) are output in terms of associated fractions of unrelaxed tube constraints (φ_X): $\mathbf{a_X = a_0}\ \varphi_X^{-\alpha/2}$ with $\mathbf{a_0}$ being the *equilibrium* tube diameter or the bare tube diameter. The output is written in a file named **STube** with appropriate extension in the absence of the *label* string. If the *label* variable is set, for example as *label=PI645k*, the file name will be **STube_PI645k** with appropriate extension. The output contains **1.** time after step deformation t, **2.** fraction of unrelaxed tube contraints φ(t), **3.** supertube fraction φ_{ST}(t), **4.** *equilibriated* constraint fraction φ_{eq}(t), **5.** constraint fraction allowing reptation φ_{rept}(t), and **6.** enhancement of reptation due to contour length fluctuation along fat tube Ψ_{min}(t).

Go to **Table of Contents**
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation

spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.4.4 Terminal relaxation pathways

If the variable ***OutTermRelaxPathways*** is set to ***yes*** via the resource file, information about terminal relaxation pathways of each chain is written in an output file named *TermRelax.dat* (or *.txt*, or *.csv*) in the absence of the *label* variable. With *label* set to some string, that string is appended to *TermR_* along with an appropriate extension.
The output contains **1.** Index ***i*** (index of the polymer in the ensemble), **2.** Weight fraction w_i, **3.** Molar mass M_i, **4.** Relaxation time τ_{relax} , **5.** Integer code for relaxation pathway, **6.** Relevant constraint fraction, **7.** Speed up factor from fat tube CLF.
The integer code in the fifth column is set to 0 for unentangled chains. For those chains, τ_{relax} is set to the Rouse time of the chains τ_R . The entries in both the sixth and seventh columns in this case are 1 signifying irrelevance of tube dilation for the relaxation of these chains.
When the chain of concern relaxes by reptation, the fifth column is set to 1 and the τ_{relax} is set to the reptation time τ_d . The sixth column in this case gives $$\varphi_T$$ associated with the optimal tube diameter for reptation at the time the chain switches from relaxing by contour length fluctuation to reptation. The seventh column in this case is the speed up due to fat tube CLF Ψ_{min}, evaluated at the time the chain commits to relax remaining stress via reptation.
When a chain effectively becomes unentangled in the *supertube*, the remaining stress associated with the chain relaxes with this time scale via *disentanglement*. In such cases, the fifth column is set to 2 and the τ_{relax} is set to the time at which the chain first becomes disentangled. The sixth column in this case report φ_{ST} and the seventh column reports the speed up factor Ψ_{min} evaluated at the time of disentanglement.

Go to **Table of Contents**
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

1.4.5 Log file

Command line argument **-L** directs the code to write information in a file named **LP2Rlog.txt** . If something goes wrong during processing the input files, the log file can be helpful in resolving the place where the code failed. The log file also reports molar mass moments (M_N, M_W, and M_Z) and the zero-shear viscosity.

Go to **Table of Contents**
⇒ Introduction ⇒ Usage ⇒ Input files • Material input file • Resource file ⇒ Output files • Relaxation spectra • Time relaxation • Tube diameters • Relaxation pathways • Relaxation pathways • Log file

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 LP2R_NS Namespace Reference

Variables

- double M_Kuhn
- double M_e
- double N_e
- double G_0
- double tau_e
- double G_glass
- double tau_glass
- double beta_glass
- double Alpha =1.0
- double t_CR_START =1.0
- double deltaCR =0.30
- double B_zeta =2.0
- double A_eq =2.0
- double B_eq =10.0
- double ret_pref =0.189
- double Rept_Switch_Factor =1.664
- double Rouse_Switch_Factor =1.5
- double Disentanglement_Switch =1.0
- double ret_pref_0 =0.020
- double ret_switch_exponent =0.42
- double cur_time =1.0e-3
- double DtMult =1.02
- double Log_DtMult
- double FreqMin =1.0e-3
- double FreqMax =1.0e3
- double FreqRatio =1.1
- bool CalcDielectric =false
- bool OutTermRelaxPathways =false
- bool OutPhiPhiST =false
- bool Output_G_of_t =false
- std::string OutputFormat ="Default"
- std::string CSVdelimiter =","
- bool Add_header =true
- bool has_temp =false
- bool has_origin =false
- bool has_label =false
- bool has_chem =false

- double reptate_temp =0.0
- std::string reptate_origin
- std::string reptate_label
- std::string reptate_chem
- bool GenLogFL =false
- std::string inpFNM ="inp.dat"
- std::string rcFNM ="LP2R.rc"
- std::fstream f_Log
- std::string RelSpecFNM ="RelSpec"
- std::string MechRelSpecFNM
- std::string DiRelSpecFNM
- std::string MechRelFNM ="MechRel"
- std::string OutTermRelaxFNM ="TermRelax"
- std::string OutPhiPhiSTFNM ="STube"
- std::fstream f_trelax
- std::fstream f_phi
- int npoly =0
- std::vector< C_LPoly * > LPoly
- double Rouse_wt =0.0
- double Sys_MN =0.0
- double Sys_MW =0.0
- double Sys_PDI =1.0
- bool Entangled_Dynamics =true
- double phi_true =1.0
- double phi_ST =1.0
- double phi_rept =1.0
- double phi_eq =1.0
- double Psi_rept =1.0
- double LastReptationTime =1.0
- double LastReptZ =1.0
- bool supertube_activated =false
- bool AboveTauEFirst =false
- double phi_ST_0 =1.0
- double ST_activ_time =1.0
- double STmaxDrop =1.0
- std::vector< double > t_ar
- std::vector< double > phi_ar
- std::vector< double > phi_ST_ar
- std::vector< double > t_eq_ar
- int phi_eq_indx =0

5.1.1 Detailed Description

Global variables

5.1.2 Variable Documentation

5.1.2.1 A_eq

`double LP2R_NS::A_eq =2.0`

Proportionality constant connecting "effective equilibrium time" and time to locally equiilbiate in a certain supertube

5.1.2.2 AboveTauEFirst

`bool LP2R_NS::AboveTauEFirst =false`
Start with false and set to true once t > t_CR_START

5.1.2.3 Add_header

`bool LP2R_NS::Add_header =true`
(true) Add appropriate headers in the output files

5.1.2.4 Alpha

`double LP2R_NS::Alpha =1.0`
Dilution exponent

5.1.2.5 B_eq

`double LP2R_NS::B_eq =10.0`
Constant delaying equilibrium for fast CR events

5.1.2.6 B_zeta

`double LP2R_NS::B_zeta =2.0`
Proportionality constant relating friction coefficient to supertube fraction

5.1.2.7 beta_glass

`double LP2R_NS::beta_glass`
stretching exponent for glassy relaxation

5.1.2.8 CalcDielectric

`bool LP2R_NS::CalcDielectric =false`
(false) If `true`, output dielectric loss (assuming type-A dipoles)
Commad line flag -d takes precedence over resource file instruction.

5.1.2.9 CSVdelimiter

`std::string LP2R_NS::CSVdelimiter =","`
Allow for non-standard delimiter (ex. some european locale uses semicolon)

5.1.2.10 cur_time

`double LP2R_NS::cur_time =1.0e-3`
Read as Start_time from resource file, start of integration time

5.1.2.11 deltaCR

`double LP2R_NS::deltaCR =0.30`
Fractional drop in tube constraints at CR events (in long polymers)

5.1.2.12 DiRelSpecFNM

`std::string LP2R_NS::DiRelSpecFNM`
Output file name for Dielectric relaxation (reptate mode)

5.1.2.13 Disentanglement_Switch

`double LP2R_NS::Disentanglement_Switch =1.0`
Number of entanglement in the supertube below which chains relax by "disentanglement".

5.1.2.14 DtMult

`double LP2R_NS::DtMult =1.02`
Read as Time_ratio from resource file, ratio of consecutive discrete times

5.1.2.15 Entangled_Dynamics

`bool LP2R_NS::Entangled_Dynamics =true`
true if polymers are entangled to begin with

5.1.2.16 f_Log

`std::fstream LP2R_NS::f_Log`
File stream for log

5.1.2.17 f_phi

`std::fstream LP2R_NS::f_phi`
File stream for tube diameters as a function of time

5.1.2.18 f_trelax

`std::fstream LP2R_NS::f_trelax`
File stream for detailed relaxation information

5.1.2.19 FreqMax

`double LP2R_NS::FreqMax =1.0e3`
Maximum frequency for dynamic rheology output

5.1.2.20 FreqMin

`double LP2R_NS::FreqMin =1.0e-3`
Minimum frequency for dynamic rheology output

5.1.2.21 FreqRatio

`double LP2R_NS::FreqRatio =1.1`
Multiplier (>1) between subsequent freqencies for output

5.1.2.22 G_0

`double LP2R_NS::G_0`
Plateau modulus

5.1.2.23 G_glass

`double LP2R_NS::G_glass`
Glassy modulus

5.1.2.24 GenLogFL

`bool LP2R_NS::GenLogFL =false`
(false) If `true`, create a log file to output each step

5.1.2.25 has_chem

`bool LP2R_NS::has_chem =false`
flag to note if reptate header variables have been set

5.1.2.26 has_label

`bool LP2R_NS::has_label =false`
flag to note if reptate header variables have been set

5.1.2.27 has_origin

`bool LP2R_NS::has_origin =false`
flag to note if reptate header variables have been set

5.1.2.28 has_temp

`bool LP2R_NS::has_temp =false`
flag to note if reptate header variables have been set

5.1.2.29 inpFNM

`std::string LP2R_NS::inpFNM ="inp.dat"`
Input file name

5.1.2.30 LastReptationTime

`double LP2R_NS::LastReptationTime =1.0`
Keep track of time at which some chain switched to reptation.
Any subsequent molecules may not have reptation time smaller than this.

5.1.2.31 LastReptZ

`double LP2R_NS::LastReptZ =1.0`
Z_chain corresponding to the largest reptation time assigned so far

5.1.2.32 Log_DtMult

`double LP2R_NS::Log_DtMult`
Log(DtMult)

5.1.2.33 LPoly

`std::vector< C_LPoly * > LP2R_NS::LPoly`
Polymer objects

5.1.2.34 M_e

`double LP2R_NS::M_e`
Entanglement molar mass

5.1.2.35 M_Kuhn

`double LP2R_NS::M_Kuhn`
Kuhn Molar mass

5.1.2.36 MechRelFNM

`std::string LP2R_NS::MechRelFNM ="MechRel"`
Output file name for G(t)

5.1.2.37 MechRelSpecFNM

`std::string LP2R_NS::MechRelSpecFNM`
Output file name for Mechanical relaxation (Reptate mode)

5.1.2.38 N_e

`double LP2R_NS::N_e`
N_e == M_e/M_Kuhn; Number of Kuhn beads in one entanglement

5.1.2.39 npoly

`int LP2R_NS::npoly =0`
Number of polymers

5.1.2.40 OutPhiPhiST

`bool LP2R_NS::OutPhiPhiST =false`
(false) If `true`, time evolution of different phi (can be mapped to tube diameters) as a function of time is written in a file

5.1.2.41 OutPhiPhiSTFNM

`std::string LP2R_NS::OutPhiPhiSTFNM ="STube"`
File in which different phi are written to

5.1.2.42 Output_G_of_t

`bool LP2R_NS::Output_G_of_t =false`
(false) If `true`, output mechanical relaxation in the time domain.

5.1.2.43 OutputFormat

`std::string LP2R_NS::OutputFormat ="Default"`
Output format in result files:
Default (.dat extension, space as dilimiter, header as comment with hash)
Text (.txt extension, space as dilimiter, header as string)
CSV (.csv extension, comma as (default) dilimiter)
RepTate (various extensions, see https://reptate.readthedocs.io)

5.1.2.44 OutTermRelaxFNM

`std::string LP2R_NS::OutTermRelaxFNM ="TermRelax"`
File in which terminal relaxation information should be written to

5.1.2.45 OutTermRelaxPathways

`bool LP2R_NS::OutTermRelaxPathways =false`
(false) If `true`, terminal relaxation of each chain triggers detailed output

5.1.2.46 phi_ar

`std::vector< double > LP2R_NS::phi_ar`
unrelaxed fraction (as function of time)

5.1.2.47 phi_eq

`double LP2R_NS::phi_eq =1.0`
related to tube in which CLF is possible

5.1.2.48 phi_eq_indx

`int LP2R_NS::phi_eq_indx =0`
index in `t_eq_ar` closest to the current time

5.1.2.49 phi_rept

```
double LP2R_NS::phi_rept =1.0
```
related to the tube in which reptation is preferred currently

5.1.2.50 phi_ST

```
double LP2R_NS::phi_ST =1.0
```
current unrelaxed supertube fraction

5.1.2.51 phi_ST_0

```
double LP2R_NS::phi_ST_0 =1.0
```
phi_ST just before supertube relaxation is activated

5.1.2.52 phi_ST_ar

```
std::vector< double > LP2R_NS::phi_ST_ar
```
unrelaxed supertube fraction (as function of time)

5.1.2.53 phi_true

```
double LP2R_NS::phi_true =1.0
```
current unrelaxed fraction

5.1.2.54 Psi_rept

```
double LP2R_NS::Psi_rept =1.0
```
Speed up factor for reptation by accessing fatter tube

5.1.2.55 rcFNM

```
std::string LP2R_NS::rcFNM ="LP2R.rc"
```
resource file name

5.1.2.56 RelSpecFNM

```
std::string LP2R_NS::RelSpecFNM ="RelSpec"
```
Output filename for relaxation spectra in default mode

5.1.2.57 Rept_Switch_Factor

```
double LP2R_NS::Rept_Switch_Factor =1.664
```
Constant deciding transition from CLF to reptation

5.1.2.58 reptate_chem

```
std::string LP2R_NS::reptate_chem
```
reptate header string

5.1.2.59 reptate_label

```
std::string LP2R_NS::reptate_label
```
reptate header string

5.1.2.60 reptate_origin

```
std::string LP2R_NS::reptate_origin
```
reptate header string

5.1.2.61 reptate_temp

`double LP2R_NS::reptate_temp =0.0`
temperature for reptate header

5.1.2.62 ret_pref

`double LP2R_NS::ret_pref =0.189`
Constant in arm retraction formula ($\mathcal{C}_{a,\infty}$)

5.1.2.63 ret_pref_0

`double LP2R_NS::ret_pref_0 =0.020`
Short-time prefactor for CLF ($\mathcal{C}_{a,0}$)

5.1.2.64 ret_switch_exponent

`double LP2R_NS::ret_switch_exponent =0.42`
Exponent ϵ_a determining how steeply CLF switches to long-time strength

5.1.2.65 Rouse_Switch_Factor

`double LP2R_NS::Rouse_Switch_Factor =1.5`
Minimum number of bare entanglements to be considered entangled (Z_u)

5.1.2.66 Rouse_wt

`double LP2R_NS::Rouse_wt =0.0`
Weight fraction of chains that relax by free Rouse

5.1.2.67 ST_activ_time

`double LP2R_NS::ST_activ_time =1.0`
time at which current supertube relaxation is activated

5.1.2.68 STmaxDrop

`double LP2R_NS::STmaxDrop =1.0`
Long time maximum drop in phi_ST druing one time step.
STmaxDrop = exp(-log(DtMult)/(2.0∗Alpha))

5.1.2.69 supertube_activated

`bool LP2R_NS::supertube_activated =false`
(false) set to true during supertube relaxation

5.1.2.70 Sys_MN

`double LP2R_NS::Sys_MN =0.0`
Number averaged molar mass of the blend

5.1.2.71 Sys_MW

`double LP2R_NS::Sys_MW =0.0`
Weight averaged molar mass of the blend

5.1.2.72 Sys_PDI

`double LP2R_NS::Sys_PDI =1.0`
polydispersity index of the blend

5.1.2.73 t_ar

`std::vector< double > LP2R_NS::t_ar`
Discrete times at which relaxation is tracked

5.1.2.74 t_CR_START

`double LP2R_NS::t_CR_START =1.0`
Time (units of tau_e) below which no CR events are included in the tube model. (faster events are accounted in the bare tube picture itself.)

5.1.2.75 t_eq_ar

`std::vector< double > LP2R_NS::t_eq_ar`
equilibration time in current supertube

5.1.2.76 tau_e

`double LP2R_NS::tau_e`
Entanglement time

5.1.2.77 tau_glass

`double LP2R_NS::tau_glass`
Glassy relaxation time

Chapter 6

Class Documentation

6.1 C_LPoly Class Reference

`#include <LP2R.h>`

Public Member Functions

- **C_LPoly** (const double M, const double W, const double M_e)

Public Attributes

- double mass =0.0
- double wt =0.0
- double Z_chain =0.0
- double z =0.0
- bool alive =true
- bool relax_free_Rouse =false
- bool rept_set =false
- double tau_d_0 =1.0e22
- double Z_rept =0.0
- double rept_wt =0.0
- int p_max =0
- int p_next =0
- double t_FRouse

6.1.1 Detailed Description

Agreegate class to hold information about a linear polymer

6.1.2 Member Data Documentation

6.1.2.1 alive

`bool C_LPoly::alive =true`
Becomes false if the chain completely relaxes

6.1.2.2 mass

`double C_LPoly::mass =0.0`
molar mass of the polymer (g/mole)

6.1.2.3 p_max

`int C_LPoly::p_max =0`
Highest reptation mode

6.1.2.4 p_next

`int C_LPoly::p_next =0`
Next available reptation mode

6.1.2.5 relax_free_Rouse

`bool C_LPoly::relax_free_Rouse =false`
True for unentangled chains

6.1.2.6 rept_set

`bool C_LPoly::rept_set =false`
Becomes true once further relaxation is assigned tobe via reptation

6.1.2.7 rept_wt

`double C_LPoly::rept_wt =0.0`
Weight associated with reptation

6.1.2.8 t_FRouse

`double C_LPoly::t_FRouse`
Relaxation time by free Rouse relaxation (Z_chain^2) tau_e

6.1.2.9 tau_d_0

`double C_LPoly::tau_d_0 =1.0e22`
Reptation time for the zeroth mode; initialized to a large number

6.1.2.10 wt

`double C_LPoly::wt =0.0`
Weight fraction associated with the polymer in the ensemble

6.1.2.11 z

`double C_LPoly::z =0.0`
Escaped number of entanglements from either ends of the chain at current time

6.1.2.12 Z_chain

`double C_LPoly::Z_chain =0.0`
Number of entanglements in the chain. Z_chain=mass/M_e

6.1.2.13 Z_rept

`double C_LPoly::Z_rept =0.0`
Amount of chain that should relax by reptation
The documentation for this class was generated from the following file:

- include/LP2R.h

6.2 InvSqSum Class Reference

Class to hold and retrieve partial sums of the form 1/p^2
.
`#include <LP2R.h>`

Public Member Functions

- double **intg_psum** (int n1, int n2)
- double **operator()** (int Z)
- double **operator()** (int Z1, int Z2)

6.2.1 Detailed Description

Class to hold and retrieve partial sums of the form 1/p^2
.
Explicitly sum 1/p^2 till 1/499^2
Higher terms are added as integrals as required.
Function call operator is overloaded to return partial sums
The documentation for this class was generated from the following file:

- include/LP2R.h

Chapter 7

File Documentation

7.1 include/LP2R.h File Reference

Define classes and namespaces.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <cmath>
#include <algorithm>
#include "./LP2R_NS.h"
#include "./routines.h"
```

Classes

- class C_LPoly
- class InvSqSum

  Class to hold and retrieve partial sums of the form 1/p^2

  .

7.1.1 Detailed Description

Define classes and namespaces.

7.2 LP2R.h

Go to the documentation of this file.

```
1 #ifndef _LinLin_H_
2 #define _LinLin_H_
3 #include <iostream>
4 #include <fstream>
5 #include <sstream>
6 #include <vector>
7 #include <string>
8 #include <cmath>
9 #include <algorithm>
10
20 class C_LPoly{
21 public:
22 C_LPoly()=default;
23 C_LPoly(const double M, const double W, const double  M_e) :
24  mass(M), wt(W), Z_chain(M/M_e), t_FRouse(Z_chain*Z_chain) {}
25
26 double mass=0.0;
27 double wt=0.0;
28 double Z_chain=0.0;
29 double z=0.0;
31 bool alive=true;
```

```
32 bool relax_free_Rouse=false;
34 bool rept_set=false;
35 double tau_d_0=1.0e22;
36 double Z_rept=0.0;
37 double rept_wt=0.0;
38 int p_max=0;
39 int p_next=0;
41 double t_FRouse;
42              };
43
44
53 class InvSqSum{ // sum(1/p^2)
54 double psum[500];
55 public:
56   InvSqSum(){
57    psum[0]=0.0; psum[1]=1.0;
58    for(int i=2; i<500; i++){psum[i]=psum[i-1]+1.0/((double) (i*i));}
59             }
60   double intg_psum(int n1, int n2){
61    double n1d=((double) n1); double n2d=((double) n2);
62    double n1dsq=n1d*n1d; double n2dsq=n2d*n2d;
63    double val=1.0/n1d - 1.0/n2d + 0.50*(1.0/n1dsq + 1.0/n2dsq);
64      val+=(1.0/(n1d*n1dsq) - 1.0/(n2d*n2dsq))/6.0;
65      return val;
66                                    }
67    double operator()(int Z){
68      if(Z < 500){return psum[Z];}
69      else{ return psum[499]+intg_psum(500,Z);}
70                           }
71    double operator()(int Z1, int Z2){
72       double s1, s2;
73       if(Z1 <= 500){s1=psum[Z1-1];}
74       else{s1=psum[499]+intg_psum(500,Z1-1);}
75       if(Z2 < 500){s2=psum[Z2];}
76       else{s2=psum[499]+intg_psum(500,Z2);}
77       return s2-s1;
78                                    }
79
80              };
81
82 #include "./LP2R_NS.h"
83 #include "./routines.h"
84
85
86 #endif
```

7.3 LP2R_global.h

```
1 // Define global namespace objects
2 namespace LP2R_NS{
3 // Model parameters
4 double  M_Kuhn, M_e, N_e, G_0, tau_e;
5 double G_glass, tau_glass, beta_glass;
6
7 double Alpha=1.0, t_CR_START=1.0, deltaCR=0.30;
8 double B_zeta=2.0, A_eq=2.0, B_eq=10.0;
9 double ret_pref=0.189, Rept_Switch_Factor=1.664;
10 double Rouse_Switch_Factor=1.5, Disentanglement_Switch=1.0;
11 double ret_pref_0=0.020, ret_switch_exponent=0.42;
12
13 // discrete time evolution control
14 double cur_time=1.0e-3, DtMult=1.02, Log_DtMult;
15
16 // Output control
17 double FreqMin=1.0e-3, FreqMax=1.0e3, FreqRatio=1.1;
18 bool CalcDielectric=false, OutTermRelaxPathways=false, OutPhiPhiST=false;
19 bool Output_G_of_t=false;
20 std::string OutputFormat="Default", CSVdelimiter=","; bool Add_header=true;
21 bool has_temp=false, has_origin=false, has_label=false, has_chem=false;
22 double reptate_temp=0.0;
23 std::string reptate_origin, reptate_label, reptate_chem;
24 bool GenLogFL=false;
25
26 // input filenames and file handles
27 std::string inpFNM="inp.dat", rcFNM="LP2R.rc";
28 std::fstream f_Log;
29
30 // output filenames
31 std::string RelSpecFNM="RelSpec", MechRelSpecFNM, DiRelSpecFNM;
32 std::string MechRelFNM="MechRel";
33 std::string OutTermRelaxFNM="TermRelax", OutPhiPhiSTFNM="STube";
34 std::fstream f_trelax;
35 std::fstream f_phi;
36
37 // Polymer collection
```

```
38 int npoly=0;
39 std::vector<C_LPoly *> LPoly;
40 double Rouse_wt=0.0;
41 double Sys_MN=0.0, Sys_MW=0.0, Sys_PDI=1.0;
42 bool Entangled_Dynamics=true;
43
44 // time dependent relaxation variables
45 double phi_true=1.0, phi_ST=1.0, phi_rept=1.0, phi_eq=1.0, Psi_rept=1.0;
46 double LastReptationTime=1.0, LastReptZ=1.0;
47
48 bool supertube_activated=false, AboveTauEFirst=false;
49 double phi_ST_0=1.0, ST_activ_time=1.0, STmaxDrop=1.0;
50
51 // Storage of relaxation data
52 std::vector<double> t_ar, phi_ar, phi_ST_ar, t_eq_ar;
53 int phi_eq_indx=0;
54                 };
```

7.4 include/LP2R_NS.h File Reference

global namespaces

Namespaces

- namespace LP2R_NS

7.4.1 Detailed Description

global namespaces

7.5 LP2R_NS.h

Go to the documentation of this file.

```
1 #ifndef _LP2R_NS_
2 #define _LP2R_NS_
12 namespace LP2R_NS{
13
14 // Model parameters
15
16 extern double M_Kuhn;
17 extern double M_e;
18 extern double G_0;
19 extern double tau_e;
20 extern double N_e;
22 extern double G_glass;
23 extern double tau_glass;
24 extern double beta_glass;
26 extern double Alpha;
27 extern double t_CR_START;
30 extern double deltaCR;
31 extern double B_zeta;
32 extern double A_eq;
34 extern double B_eq;
35 extern double ret_pref;
36 extern double ret_pref_0;
37 extern double ret_switch_exponent;
39 extern double Rept_Switch_Factor;
40 extern double Rouse_Switch_Factor;
41 extern double Disentanglement_Switch;
43 extern double cur_time;
44 extern double DtMult;
45 extern double Log_DtMult;
47 // Files
48 extern std::string inpFNM;
49 extern std::string rcFNM;
51 // control output
52 extern bool CalcDielectric;
54 extern std::string OutputFormat;
59 extern std::string CSVdelimiter;
60 extern bool Add_header;
61 extern bool OutTermRelaxPathways;
62 extern bool Output_G_of_t;
63 extern bool GenLogFL;
64 extern std::fstream f_Log;
65 extern std::fstream f_trelax;
66 extern std::fstream f_phi;
68 extern bool OutPhiPhiST;
```

```
70 extern double FreqMin;
71 extern double FreqMax;
72 extern double FreqRatio;
74 // relaxation data
75 extern double phi_true;
76 extern double phi_ST;
77 extern double phi_rept;
78 extern double phi_eq;
79 extern double Psi_rept;
81 extern bool supertube_activated;
82 extern double phi_ST_0;
83 extern double ST_activ_time;
84 extern bool AboveTauEFirst;
85 extern double STmaxDrop;
88 // Reptate software specific input
89 extern bool has_temp;
90 extern bool has_origin;
91 extern bool has_label;
92 extern bool has_chem;
93 extern std::string reptate_origin;
94 extern std::string reptate_label;
95 extern std::string reptate_chem;
96 extern double reptate_temp;
98 // Output file names
99 extern std::string RelSpecFNM;
100 extern std::string MechRelSpecFNM;
101 extern std::string DiRelSpecFNM;
102 extern std::string MechRelFNM;
103 extern std::string OutTermRelaxFNM;
104 extern std::string OutPhiPhiSTFNM;
106 extern std::vector<double> t_ar;
107 extern std::vector<double> phi_ar;
108 extern std::vector<double> phi_ST_ar;
109 extern std::vector<double> t_eq_ar;
110 extern int phi_eq_indx;
111 extern double LastReptationTime;
113 extern double LastReptZ;
115 extern std::vector<C_LPoly *> LPoly;
116 extern int npoly;
117 extern double Rouse_wt;
118 extern double Sys_MN;
119 extern double Sys_MW;
120 extern double Sys_PDI;
121 extern bool Entangled_Dynamics;
122                       };
123
124 #endif
```

7.6 routines.h

```
1 int parse_arg(int argc, char* argv[]);
2 void ReadRCFL(void);
3 int ReadInput(void);
4
5 int genPolyLin(std::fstream&);
6 double aaerfcc(const double x);
7 void GenLinLogNormal(const int n, const double mw, const double pdi, const double wtcomp);
8 int GenLinGPC(std::string &fname, const double wtcomp);
9 int GenLinWt(std::string &fname, const double wtcomp);
10 void assign_FNMs(void);
11
12 std::istream& safeGetline(std::istream& is, std::string& t);
13 std::istream& readEquality(std::istream& is, std::string& sL, std::string& sR);
14
15 // relaxation
16 double GetPhiEq(void);
17 void frac_unrelaxed(void);
18 void try_reptate(const int np);
19 void arm_retraction(const int np, const int indx);
20 int time_step(const int indx);
21
22 // Calculation of responses
23 void add_spectra_headers(std::fstream &fRh, std::fstream &fDi);
24 void GStarGlass(const double w, double &gp, double &g2p);
25 void GStarRouse(double freq, double &gRs, double &g2Rs, double &eRs, double &e2Rs);
26 void GStarFastRouse(const double w, double &gpf, double &g2pf);
27 void GStarSlow(const double w, double &gp, double &g2p, double &ep, double &e2p);
28 void CalcGstar(std::fstream &fRh, std::fstream &fDi);
29
30 double CalcVisc(void);
31 void add_goft_headers(std::fstream &fRh);
32 double GoftRouse(const double t);
33 double GoftFast(const double t);
34 double GoftTube(const double t, double &muoft, double &Roft);
35 void Calc_goft(std::fstream &fRh);
```

```
36
37 void LinRheology(void);
```

7.7 main/LinPoly2Rheo.cpp File Reference

main for LP2R

```
#include "../include/LP2R.h"
#include "../include/LP2R_global.h"
#include "../include/tclap/CmdLine.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

7.7.1 Detailed Description

main for LP2R

Parameters

in	*argc*	Number of command line arguments
in	*argv*	argument list

Returns

nonzero integer is run fails

7.8 main/parse_arg.cpp File Reference

parse command line arguments

```
#include "../include/LP2R.h"
#include "../include/tclap/CmdLine.h"
```

Functions

- int parse_arg (int argc, char ∗argv[ ])

7.8.1 Detailed Description

parse command line arguments
options [-i input_file] [-r resource_file] [-d] [-L] [-v] [-h] [–version] [–help]
-i input_file : Read material and polymer information from this file (default: inp.dat)

-r resource_file : Model parameters (default: LP2R.rc).
Will use default values if the file does not exist.

-d : Switch on output of dielectric response.

-L : Output steps in LP2Rlog.txt, useful for debugging.

-v / –version : version information
-h / –help : help page.

7.8.2 Function Documentation

7.8.2.1 parse_arg()

```
int parse_arg (
            int argc,
            char * argv[] )
```

parse command line arguments

Parameters

in	*argc*	number of arguments passed to main.
in	*argv*	character array containing the arguments.

Returns

zero if parsed properly and non-zero in case of error.

7.9 mainpage.h

```
1
```

7.10 prep/assign_FNMs.cpp File Reference

Depending on the output format and presence of "reptate_label", select file names for output.

```
#include "../include/LP2R.h"
```

Functions

- void **assign_FNMs** (void)

7.10.1 Detailed Description

Depending on the output format and presence of "reptate_label", select file names for output.

7.11 prep/GenLinGPC.cpp File Reference

Generate polymers based on GPC data from a file.

```
#include "../include/LP2R.h"
#include <algorithm>
```

Functions

- void aa_sort2_minmax (std::vector< double > &m, std::vector< double > &p)
- int GenLinGPC (std::string &fname, const double wtcomp)

7.11.1 Detailed Description

Generate polymers based on GPC data from a file.

7.11.2 Function Documentation

7.11.2.1 aa_sort2_minmax()

```
void aa_sort2_minmax (
            std::vector< double > & m,
            std::vector< double > & p )
```

sort paired vectors in ascending order of the first

Parameters

in,out	*m*	molar mass
in,out	*p*	dwdm associated with m

7.11.2.2 GenLinGPC()

```
int GenLinGPC (
            std::string & fname,
            const double wtcomp )
```

GPC data in {M, dw/dlog_10(M)} format in the specified file As with all other files, comments are allowed with a "%" symbol

Parameters

in,out	*fname*	string containing the filename
in	*wtcomp*	weight fraction assigned to the current component

Returns

zero if successful

7.12 prep/GenLinLogNormal.cpp File Reference

Generate discrete representation of a logNormal distribution.

```
#include "../include/LP2R.h"
```

Functions

- double aaerfcc (const double x)
- double LogNormalWt (const double Mw, const double PDI, const double M1, const double M2, double &Mw↩ Bin)
- void GenLinLogNormal (const int n, const double mw, const double pdi, const double wtcomp)

7.12.1 Detailed Description

Generate discrete representation of a logNormal distribution.

7.12.2 Function Documentation

7.12.2.1 aaerfcc()

```
double aaerfcc (
            const double x )
```

Series for complementary error function

Parameters

in	*x*	

Returns

errc(x)

7.12.2.2 GenLinLogNormal()

```
void GenLinLogNormal (
            const int n,
            const double mw,
            const double pdi,
            const double wtcomp )
```

Generate polymers from a logNormal distribution characterized by molar mass mw and PDI pdi.
Special case: if either the number of discrete molar mass is one or PDI < 1, create a single polymer with the molar mass supplied.

Parameters

in	*n*	number of discrete molar mass to represent the distribution
in	*mw*	weight averaged molar mass
in	*pdi*	Polydispersity index
in	*wtcomp*	weight fraction of this polymer componennt

7.12.2.3 LogNormalWt()

```
double LogNormalWt (
            const double Mw,
            const double PDI,
            const double M1,
            const double M2,
            double & MwBin )
```

Weight fraction in a specified molar mass range from a logNormal distribution

Parameters

in	*Mw*	weight-averaged molar mass
in	*PDI*	Polydispersity index
in	*M1*	Lower limit of the molar mass range
in	*M2*	upper limit of the molar mass range
out	*MwBin*	weight averaged molar mass in this interval

Returns

weight fraction in the molar mass range.

Negative M1 ==> (0, M2); Negative M2 ==> (M1, infinity)

7.13 prep/GenLinWt.cpp File Reference

Generate polymers with {molar mass, weight fraction} entries in a file.

```
#include "../include/LP2R.h"
```

Functions

- int **GenLinWt** (std::string &fname, const double wtcomp)

7.13.1 Detailed Description

Generate polymers with {molar mass, weight fraction} entries in a file.

Parameters

in,out	*fname*	file containing the weight fractions
in	*wtcomp*	weight fraction of the current component

Returns

zero if successful

7.14 prep/genPolyLin.cpp File Reference

Read input for polymer components and generate/read polymers.
`#include "../include/LP2R.h"`

Functions

- int genPolyLin (std::fstream &f1)

7.14.1 Detailed Description

Read input for polymer components and generate/read polymers.

7.14.2 Function Documentation

7.14.2.1 genPolyLin()

```
int genPolyLin (
            std::fstream & f1 )
```

Read input for polymer components and generate polymers

Parameters

in	*f1*	input file stream

Returns

status=0 if success, else non-zero

7.15 prep/ModelParams.cpp File Reference

Set default model parameters that are unlikely to be changed by most users.
`#include "../include/LinLin.h"`

Functions

- void **ModelParams** (void)

7.15.1 Detailed Description

Set default model parameters that are unlikely to be changed by most users.

7.16 prep/ReadInput.cpp File Reference

Read material parameters from the input file and call `genPolyLin`.
`#include "../include/LP2R.h"`

Functions

- int **ReadInput** (void)

7.16.1 Detailed Description

Read material parameters from the input file and call `genPolyLin`.

Returns

nonzero integer in case input fails

7.17 prep/ReadRCFL.cpp File Reference

Read options given as option=value pair in a file.
`#include "../include/LP2R.h"`

Functions

- bool assign_RHS_bool (std::string &sR)
- void assign_RC_dbl (double &param, const std::string &sR, const std::string sprm, const double val)
- void ReadRCFL (void)

7.17.1 Detailed Description

Read options given as option=value pair in a file.

7.17.2 Function Documentation

7.17.2.1 assign_RC_dbl()

```
void assign_RC_dbl (
            double & param,
            const std::string & sR,
            const std::string sprm,
            const double val )
```

Try to assign double equivalent value of sR to param (repeated as string sprm). If it fails, it will assign the default value val (repeated as string sval).

Parameters

out	*param*	parameter value represented in string sR

Parameters

in	*sR*	string object representing some double
in	*sprm*	Name of the parameter we are trying to assign
in	*val*	default value of the parameter

7.17.2.2 assign_RHS_bool()

```
bool assign_RHS_bool (
            std::string & sR )
```

return `true` if string sR=="yes"

7.17.2.3 ReadRCFL()

```
void ReadRCFL (
            void  )
```

(If present) read resource file Ignore anything that does not make sense

7.18 Relax/arm_retraction.cpp File Reference

relaxation from contour length fluctuation

```
#include "../include/LP2R.h"
```

Functions

- void **arm_retraction** (const int np, const int indx)

7.18.1 Detailed Description

relaxation from contour length fluctuation

Parameters

in	*np*	Index for polymer relaxing by CLF
in	*indx*	Set to zero at first call when z=0 (avoid division by zero)

7.19 Relax/frac_unrelaxed.cpp File Reference

Decide on supertube relaxation based on material relaxed in the current time interval.

```
#include "../include/LP2R.h"
```

Functions

- void **frac_unrelaxed** (void)

7.19.1 Detailed Description

Decide on supertube relaxation based on material relaxed in the current time interval.

7.20 Relax/GetPhiEq.cpp File Reference

Tube diameter for CLF (phi_eq)
`#include "../include/LP2R.h"`

Functions

- double **GetPhiEq** (void)

7.20.1 Detailed Description

Tube diameter for CLF (phi_eq)
When a certain tube will be accessible for CLF is stored in `t_eq_ar` as that information is aquired. Always, this time is in the future. This routine interpolates the stored values to return the tube diameter relevant for CLF at the current time.

7.21 Relax/time_step.cpp File Reference

update relaxation by one time step
`#include "../include/LP2R.h"`

Functions

- int **time_step** (const int indx)

7.21.1 Detailed Description

update relaxation by one time step

Parameters

in	*indx*	set to zero for first call

Returns

number of chains still trapped in old tubes

7.22 Relax/try_reptate.cpp File Reference

Attempt relaxation by reptation.
`#include "../include/LP2R.h"`

Functions

- void **try_reptate** (const int np)

7.22.1 Detailed Description

Attempt relaxation by reptation.

Parameters

in	*np*	index of chain for which reptation is attempted

7.23 Rheology/add_goft_headers.cpp File Reference

add appropriate header to time domain relaxation output file
`#include "../include/LP2R.h"`

Functions

- void **add_goft_headers** (std::fstream &fRh)

7.23.1 Detailed Description

add appropriate header to time domain relaxation output file

Parameters

in	*fRh*	File handle for G(t) output

7.24 Rheology/Calc_goft.cpp File Reference

Calculate G(t)
`#include "../include/LP2R.h"`

Functions

- void **Calc_goft** (std::fstream &fRh)

7.24.1 Detailed Description

Calculate G(t)
The span of time is hardcoded here between $10^{-4}\tau_e$ and $10^{4}\tau_d$ with τ_d being the longest relaxation time.

Parameters

in	*fRh*	File handle for the output

7.25 Rheology/CalcGstar.cpp File Reference

`#include "../include/LP2R.h"`

Functions

- void **CalcGstar** (std::fstream &fRh, std::fstream &fDi)

7.25.1 Detailed Description

\breif Calculate and output frequency responses

Parameters

in	*fRh*	File handle for mechanical response
in	*fDi*	File handle for dielectric response

7.26 Rheology/CalcVisc.cpp File Reference

Zero shear viscosity.
`#include "../include/LP2R.h"`

Functions

- double viscRouseModes (void)

  Viscosity contribution from internal Rouse modes, Longitudinal modes, and Free Rouse chains.
- double **viscGlass** (void)

  zero-shear contribution from glassy modes
- double **viscTubeRelax** (void)

  zero-shear viscosity from tube relaxation in units of [G_0 ∗ tau_e]
- double CalcVisc (void)

7.26.1 Detailed Description

Zero shear viscosity.

7.26.2 Function Documentation

7.26.2.1 CalcVisc()

```
double CalcVisc (
            void  )
```

Return zero-shear viscosity as Pa-s

7.26.2.2 viscRouseModes()

```
double viscRouseModes (
            void  )
```

Viscosity contribution from internal Rouse modes, Longitudinal modes, and Free Rouse chains.
Return viscosity in units of G_0∗tau_e

7.27 Rheology/GoftFast.cpp File Reference

G(t) contribution from in-tube Rouse modes.
`#include "../include/LP2R.h"`

Functions

- double **GoftFast** (const double t)

7.27.1 Detailed Description

G(t) contribution from in-tube Rouse modes.

Parameters

in	*t*	time in units of tau_e

Returns

G(t) in units of G_0

7.28 Rheology/GoftRouse.cpp File Reference

G(t) contribution from unentangled chains.
`#include "../include/LP2R.h"`

Functions

- double **GoftRouse** (const double t)

7.28.1 Detailed Description

G(t) contribution from unentangled chains.

Parameters

in	*t*	time in units of tau_e

Returns

G(t) in units of G_0

7.29 Rheology/GoftTube.cpp File Reference

G(t) contribution from in-tube Rouse modes.
`#include "../include/LP2R.h"`

Functions

- double **GoftTube** (const double t, double &muoft, double &Roft)

7.29.1 Detailed Description

G(t) contribution from in-tube Rouse modes.

Parameters

in	*t*	time in units of tau_e
out	*muoft*	Tube survival probability $\mu(t)$
out	*Roft*	Constraint release contribution $R(t)$

Returns

G(t) in units of G_0

7.30 Rheology/GStarFastRouse.cpp File Reference

Internal Rouse and longitudinal modes (in units of G0)
`#include "../include/LP2R.h"`

Functions

- void **GStarFastRouse** (const double w, double &gpf, double &g2pf)

7.30.1 Detailed Description

Internal Rouse and longitudinal modes (in units of G0)

Parameters

in	*w*	Frequency
out	*gpf*	Internal Rouse and longitudinal mode contribution to G'
out	*g2pf*	Internal Rouse and longitudinal mode contribution to G"

7.31 Rheology/GStarGlass.cpp File Reference

Return glassy contribution of G' and G" at given frequency.
`#include "../include/LP2R.h"`

Functions

- double **kwws** (const double, const double)
- double **kwwc** (const double, const double)
- void GStarGlass (const double w, double &gp, double &g2p)

7.31.1 Detailed Description

Return glassy contribution of G' and G" at given frequency.

7.31.2 Function Documentation

7.31.2.1 GStarGlass()

```
void GStarGlass (
            const double w,
            double & gp,
            double & g2p )
```

Glassy contribution to G' and G" Used kww.c from Wuttke, Algorithms 5, 604-628 (2012), doi:10.3390/a5040604 @param [in] w frequency of interest @param [out] gp Glassy contribution in storage modulus G'(w) @param [out] g2p Glassy contribution in loss modulus G"(w)

7.32 Rheology/GStarRouse.cpp File Reference

Rouse spectra for unentangled chains.
`#include "../include/LP2R.h"`

Functions

- void **GStarRouse** (double freq, double &gRs, double &g2Rs, double &eRs, double &e2Rs)

7.32.1 Detailed Description

Rouse spectra for unentangled chains.

Parameters

in	*freq*	Frequency
out	*gRs*	G'
out	*g2Rs*	G" @param [out] eRs epsilon' @param [out] e2Rs epsilon"

7.33 Rheology/GStarSlow.cpp File Reference

Tube relaxation part of relaxation.

```
#include "../include/LP2R.h"
```

Functions

- void symbint (const double tk, const double td, const double w, double &rint1, double &rint2)

  analytical result from extending R(t) to time infinity.
- void GStarSlow (const double w, double &gp, double &g2p, double &ep, double &e2p)

  Calculate tube escape contrubution to relaxation moduli.

7.33.1 Detailed Description

Tube relaxation part of relaxation.
Given some frequency w, returns G'(w), G"(w), epsilon'(w), epsilon"(w)
Assume G(t)/G_0 = mu(t) R(t), with mu and R determined by exponential weighted integral of changes in phi and phi_ST
R(t) is slowly relaxing. Extend R(t) to infinity analytically.
Dielectric relaxation is considered to be proportional to mu(t)

7.33.2 Function Documentation

7.33.2.1 GStarSlow()

```
void GStarSlow (
            const double w,
            double & gp,
            double & g2p,
            double & ep,
            double & e2p )
```

Calculate tube escape contrubution to relaxation moduli.

Parameters

in	*w*	: frequency at which result is required
out	*gp*	: elastic modulus G'
out	*g2p*	: viscous modulus G" @param [out] ep : dielectric storage modulus epsilon' @param [out] e2p : dielectric loss modulus epsilon"

7.33.2.2 symbint()

```
void symbint (
            const double tk,
            const double td,
            const double w,
            double & rint1,
            double & rint2 )
```

analytical result from extending R(t) to time infinity.

Parameters

in	*tk*	: discrete time interval at which contribution from the integral is sought.
in	*td*	: time at which mu(t) first goes to zero
in	*w*	: frequency
out	*rint1*	: int((a+x)/((a+x)^2 + b x^2) dx/sqrt(x), x=1..inf); I2(a,b) in Eqn B6 in the manuscript
out	*rint2*	: int(sqrt(x)/((a+x)^2 + b x^2) dx, x=1..inf); I1(a,b) in Eqn B7 in the manuscript

7.34 Rheology/LinRheology.cpp File Reference

Calculate relaxation spectra.

```
#include "../include/LP2R.h"
```

Functions

- void **LinRheology** (void)

7.34.1 Detailed Description

Calculate relaxation spectra.

7.35 Rheology/RepTateOut.cpp File Reference

Handle output in RepTate format.

```
#include "../include/LP2R.h"
```

Functions

- void RepTateOpen (std::fstream &fRh, std::fstream &fDi)

  Open files for Reptate format and add headers.
- void RepTateWrite (std::fstream &fRh, std::fstream &fDi, const double(&res)[5])

  write relaxation data in RepTate format
- void CSVOpen (std::fstream &fRh)

  Open file for CSV format and add headers.
- void CSVWrite (std::fstream &fRh, const double(&res)[5])

  write relaxation data in CSV file

7.35.1 Detailed Description

Handle output in RepTate format.

7.35.2 Function Documentation

7.35.2.1 CSVOpen()

```
void CSVOpen (
            std::fstream & fRh )
```

Open file for CSV format and add headers.

Parameters

out	*fRh*	: file for mechanical spectra output

7.35.2.2 CSVWrite()

```
void CSVWrite (
            std::fstream & fRh,
            const double(&) res[5] )
```

write relaxation data in CSV file

Parameters

in	*fRh*	: file for relaxation spectra output
in	*res*	: fixed length array containing w, G', G", and possibly e', e"

7.35.2.3 RepTateOpen()

```
void RepTateOpen (
            std::fstream & fRh,
            std::fstream & fDi )
```

Open files for Reptate format and add headers.

Parameters

out	*fRh*	: file for mechanical spectra output
out	*fDi*	: file for Dielectric spectra output

7.35.2.4 RepTateWrite()

```
void RepTateWrite (
            std::fstream & fRh,
            std::fstream & fDi,
            const double(&) res[5] )
```

write relaxation data in RepTate format

Parameters

in	*fRh*	: file for mechanical spectra output
in	*fDi*	: file for Dielectric spectra output
in	*res*	: fixed length array containing w, G', G", and possibly e', e"

7.36 util/safeGetLine.cpp File Reference

read a nonempty line with either unix or windows style line-ending, discarding anything after a % sign

```
#include <string>
#include <iostream>
#include <algorithm>
```

Functions

- std::istream & safeGetline_int (std::istream &is, std::string &t)
- std::istream & safeGetline (std::istream &is, std::string &t)
- std::istream & readEquality (std::istream &is, std::string &sL, std::string &sR)

7.36.1 Detailed Description

read a nonempty line with either unix or windows style line-ending, discarding anything after a % sign
handle line ending with \n, \r, \r\n, or no line ending

7.36.2 Function Documentation

7.36.2.1 readEquality()

```
std::istream & readEquality (
            std::istream & is,
            std::string & sL,
            std::string & sR )
```

Read lines with contents like a=b On either side of the equality, space or tabs are removed return the LHS and RHS separately in sL and sR

Parameters

in	*is*	input stream
out	*sL*	string containing LHS of some equality
out	*sR*	string containing RHS of some equality

Returns

input stream

7.36.2.2 safeGetline()

```
std::istream & safeGetline (
            std::istream & is,
            std::string & t )
```

Read a line from the input stream, discard anything after a '' sign (treat as comment indicator) If the resulting string has no non-space characters and the input stream hasn't reached the end of file, read the next line. Continue till either the end of file is reached, or a line with some non-empty, non-comment character is found.

Parameters

in	*is*	input stream
out	*t*	string

Returns

input stream

7.36.2.3 safeGetline_int()

```
std::istream & safeGetline_int (
            std::istream & is,
            std::string & t )
```

Implementation of getline() to handle different line endings.

Parameters

in	*is*	input stream
out	*t*	string

Returns

input stream

Index