OCTA

ソフトマテリアルのための統合化シミュレータ

レオロジーシミュレータ

PASTA

ユーザーズマニュアル

OCTA ユーザー会

DEC. 25 2002

執筆者

第 1 章　滝本淳一、田崎弘恭
第 2 章　滝本淳一、田崎弘恭
第 3 章　滝本淳一、田崎弘恭
第 4 章　滝本淳一、田崎弘恭
第 5 章　滝本淳一、田崎弘恭
第 6 章　滝本淳一、庄司達也
第 7 章　滝本淳一、田崎弘恭
付録 A　滝本淳一、田崎弘恭

プログラム開発者

PASTA　滝本淳一
FORK　庄司達也

謝辞

本プログラム開発は、経済産業省の出資 · 補助を受け、新エネルギー · 産業技術総合開発機構 (NEDO) が (財) 化学技術戦略推進機構に委託した、大学連携型産業科学技術研究開発プロジェクト「高機能材料設計プラットフォーム」通称「土井プロジェクト」の下で行われたものである。

目 次

図 目 次

第1章　PASTAとは

高分子溶融体のレオロジーは、その成形加工性を支配する最大の要因である。レオロジー特性には分子量分布や分岐構造が非常に重要であることは従来から知られていたが、分子量分布や分岐構造からレオロジー特性を定量的に予測することは不可能であった。我々の目的はこの予測を可能にすることである。その実現のためにレオロジー予測プログラムの開発を進めてきた。近年、分子量分布や分岐構造を工業レベルで制御することが可能になりつつあるので、我々のプログラムと組み合わせれば、レオロジー特性、さらには成形加工性の優れる高分子材料の設計・製造が可能になるものと期待される。

我々はスリップリンク模型 (slip-link model) に基づき、高分子溶融体をスリップリンクに拘束された多数の鎖の集まりとしてモデル化し、その運動を確率論的にシミュレーションする手法を開発した。この手法を実装したプログラムが **PASTA** (*Polymer rheology Analyzer with Slip-link model of enTAnglement*) である[1] 。単分散、多分散の直鎖高分子、星型高分子、及びそれらの混合系についてシミュレーションが可能である。

PASTA の入力ファイル作成支援ツールが **FORK** である。任意の分子量分布に対応した入力ファイルを作成するとともに、計算条件を予め入力しておくことで、 **FORK** の出力ファイルをそのまま **PASTA** の入力ファイルとして使用することができる。

[1] ソースコード等の中でコードネームと異なる旧仮称を使用している場合があるので混乱の無いよう。

第2章　理論背景

2.1　計算モデル

高分子溶融体中の絡み合った分子鎖のダイナミクスを扱えるモデルとしては、土井-Edwards の管模型 [1] が標準的である。PASTA ではこの管模型を更に発展させた slip-link モデルを用いる。また、古典的な管模型ではレプテーション運動のみが考慮されているが、PASTA ではそれに加え、contour length fluctuation (分子鎖全長のゆらぎ) と constraint release (絡み合っている相手の分子の運動により絡み合いがはずれること) の 2 つの効果を取り入れた [2]。

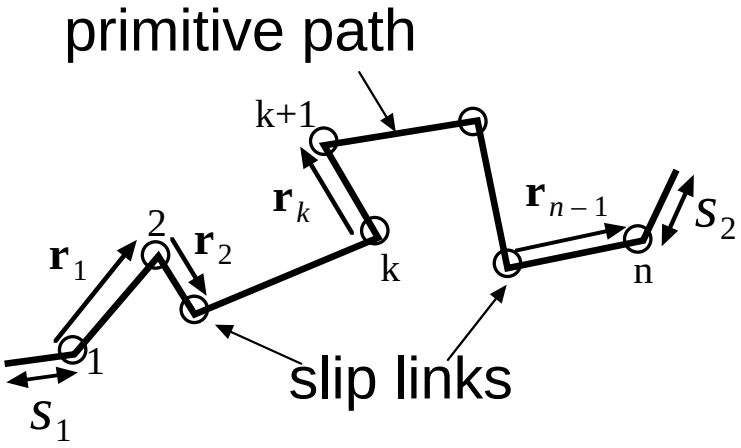

図 2.1: primitive path と slip-links の模式図

slip-link モデルの基本的な仮定は、

- slip-link は必ず 2 本の分子鎖を拘束する。どちらか一方の分子鎖が slip-link をすり抜けてしまうと、その slip-link は消滅する。
- 各々の分子鎖は、slip-link に拘束されているという点を除き、独立に運動する。

というものである。このモデルでは、まず一本の分子鎖を primitive path と、その上にある slip-link で表す (図 2.1)。slip-link は分子鎖同志の絡み合い点に対応する。或る分子鎖上の slip-link の個数 n は時間的に変動するが、平均としては $Z = M/M_e$ 個の slip-link があることになる (M はこの分子鎖の分子量、M_e はこの高分子の絡み合い点間分子量)。$r_j (j = 1, 2 \ldots, n-1)$ を隣り合う slip-link を結ぶベクトルとする。また、両端の slip-link からはみ出している tail の長さを s_1, s_2 とすると、分子鎖の全長 L は $L = \sum_{j=1}^{n-1} r_j + s_1 + s_2$ で表される。

高分子溶融体を、このような分子鎖の多数本の集合と考える (図 2.2)。分子量分布がある場合は、その分布に従った長さをもつ分子鎖の集合を考える。そして、各々の分子鎖上の絡み合い点を、自分以外の分子鎖と互いにランダムに対応させる。以下では対応する絡み合い点をパートナーと呼ぶ。

分子鎖間の相互作用はこの絡み合い点の対応関係としてだけ取り入れられ、各分子鎖はそれぞれ独立に運動する。具体的には、各タイムステップにおいて、各分子鎖に対し以下の操作を適用する：

1. 流動による primitive path のアフィン変形
   試料にマクロな流動が加えられている場合、それに従って slip-link の位置を移動させる (つまり、r_j を

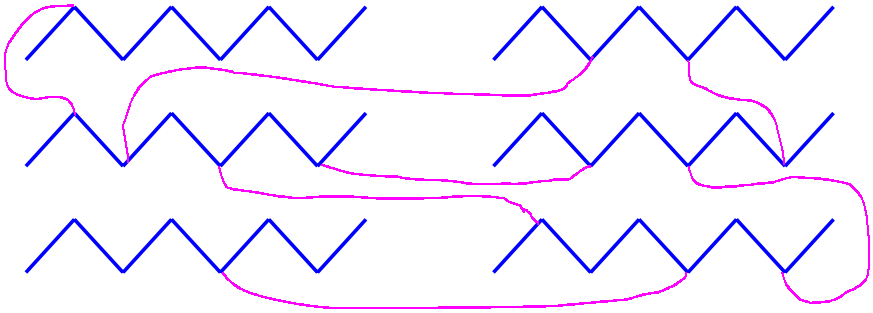

図 2.2: slip-link 間の対応の模式図。一部の対応関係だけ示した。

変形させる)。

2. Contour length fluctuation
   分子鎖の全長 L をランダムに増減する。具体的には Langevin 方程式

$$\frac{dL}{dt} = -\frac{1}{\tau_R}\left(L(t) - L_{eq}\right) + \dot{L}_{affine} + g(t) \tag{2.1}$$

   に従って変化させる。ここで $\tau_R = \tau_e Z^2$ はこの分子鎖のラウス緩和時間、τ_e は分子量が M_e と等しい分子鎖のラウス緩和時間 (これを時間の単位にとる)、$L_{eq} = Za$ は分子鎖の平衡長、a は slip-link 間の平均距離 (これを長さの単位にとる) である。また、$\dot{L}_{affine}$ はアフィン変形による L の変化である。$g(t)$ はランダム力で、

$$< g(t_1)g(t_2) >= \frac{2a^2}{3\tau_e Z}\delta(t_1 - t_2) \tag{2.2}$$

   を満たす。

3. レプテーション
   分子鎖 (の重心) を primitive path に沿ってランダムに動かす。拡散係数 D_c は

$$D_c = \frac{a^2}{3\pi^2\tau_e} \tag{2.3}$$

   である。

4. Constraint renewal
   2,3. の結果新しい s_1, s_2 が求まる。$s_i < 0$ なら末端の絡み合い点とそのパートナーとを取り去る。逆に $s_i > a$ なら末端に新しい絡み合い点を作り、分子鎖の長さに応じた確率で選んだ自分以外の分子鎖上に、パートナーとなる絡み合い点を追加する。

分子鎖当たりの応力の値は、

$$\sigma_{\alpha\beta} = \sum_j F_{j\alpha} r_{j\beta} \tag{2.4}$$

で計算される。張力 F は primitive path に沿って一様な値

$$F = \frac{3k_BT}{a}\frac{L}{L_{eq}} \tag{2.5}$$

をとると仮定すると、分子鎖一本の応力への寄与は

$$\sigma_{\alpha\beta} = \frac{3k_BT}{Za^2}\sum_j L\frac{r_{j\alpha}r_{j\beta}}{r_j} \tag{2.6}$$

で計算される。k_B はボルツマン定数、T は温度である。

以上は直鎖高分子の場合のシミュレーション法であるが、星形高分子の場合にも容易に拡張できる。但し、分岐点は試料のマクロな流動に乗って移動する以外の運動はしないと仮定する。この仮定の下では、星形高分子の各々の腕は独立に取り扱える[1]。従って、各腕はレプテーションを行わない点を除き、直鎖高分子と同様にシミュレーションできる。但し、分子量が Z_a である腕のラウス緩和時間としては $\tau_R = 4\tau_e Z_a^2$ 即ち、分子量 $2Z_a$ の直鎖のラウス緩和時間を用いる必要がある[2]。
なお、各々の腕を独立に取り扱っていることからわかるように、このシミュレーション手法は、星形高分子のレオロジーについて腕の本数に依存しない結果を与える。

2.2 モデルパラメータ

実際の高分子のレオロジーを予測するためには、対象とする高分子の分子量 (分子鎖の長さ) と時間スケールに応じた換算を施すために、少なくとも 2 つのパラメータが必要になる。前者が絡み合い点間分子量 M_e であり $Z = M/M_e$ が入力に用いられる。もう一つが時間の単位となる τ_e、即ち M_e と等しい分子量を持つ鎖のラウス緩和時間である。slip-link モデルでは、M_e とプラトーモデュラス G_N との関係は

$$G_N = \frac{4}{5}\frac{\rho RT}{M_e} \tag{2.7}$$

となる。ここで、ρ は密度、R は気体定数である。様々な高分子の G_N の値が報告されている。この関係から G_N の実験値を用いて M_e を決めることができる。但し、多くの文献では M_e の値は $G_N = \rho RT/M_e$ で計算されているため、例えばポリスチレンの場合は、一般的に知られている 18,000 ではなく、4/5 の係数のかかった 14,400 を用いる。

以下、シミュレーションと実験との比較によって、τ_e を求める方法について述べる。ここでは、ずり粘度のにおける実験と計算との比較から τ_e を求める例を示した。
シミュレーションの出力する応力 (無次元) を s とすると、モデルでは実際の応力 σ は

$$\sigma = \frac{15}{4}G_N s \tag{2.8}$$

のように表される。一方、時間の単位を τ_e としていることから、シミュレーションのずり速度 (無次元) を g とすると、実際のずり速度 $\dot{\gamma}$ と粘度 η は

$$\dot{\gamma} = \frac{g}{\tau_e} \tag{2.9}$$

$$\eta = \frac{\sigma}{\dot{\gamma}} = \frac{15}{4}G_N\tau_e\frac{s}{g} \tag{2.10}$$

と表せるので、シミュレーションにおける粘度 (無次元)s/g と実験値との比較から τ_e を決めることができる。

[1] もちろんそれらの間の絡み合いは考慮される。
[2] Z_a は正確には分子量を M_e で割ったもの。

第3章　操作入門

3.1　シミュレーションの概要

PASTA によるシミュレーションの流れは以下のようになる。

1. 系の熱平衡化（thermalization）
   PASTA では目的とする流動・変形を加える前に、初期構造の緩和をおこなう必要がある。熱平衡化した結果はリスタートファイルとして、別のシミュレーションに用いることができる。

2. シミュレーション（流動・変形）
   流動や変形を与えてシミュレーションを行う。単分散・多分散直鎖状高分子、単分散星型高分子、及びそれらの混合系の応力緩和、ずり、伸長（一軸、二軸、平面）変形等のシミュレーションが可能である。

3. 解析
   シミュレーションにより系の応力値（ずり応力、第一法線応力差、第二法線応力差）の経時変化が得られる。粘度、緩和弾性率等のレオロジー量を求めるためには、目的に応じた解析を行う必要がある。

PASTA の入力 UDF を作成するために FORK が用意されている。FORK を用いることで、分子量分布や計算条件に応じて簡単に入力 UDF を作成することができる。
ここでは、FORK を用い PASTA 用の入力 UDF を作成し、それを PASTA で計算する操作方法の手順を説明する[1] 。

3.2　PASTA チュートリアル

ファイル一覧

このチュートリアルで使用、あるいは説明したファイルを以下に示す。

- FORK による PASTA 用入力 UDF の作成

```
(.exe)         /PF_ENGINE/bin/$(PF_ENGINEARCH)/fork(.exe)
(Input UDF)    /PF_ENGINE/PASTA/FORK-1.4.1/sample/in_fork.udf
(Out def UDF) /PF_ENGINE/udf/forkdef.udf
(Output UDF)   /PF_ENGINE/PASTA/FORK-1.4.1/sample/out/out_fork.udf
```

- PASTA による計算

```
(.exe)         /PF_ENGINE/bin/$(PF_ENGINEARCH)/pasta(.exe)
(Input UDF)    /PF_ENGINE/PASTA/PASTA-2.8/sample/out_fork.udf
(Out def UDF) /PF_ENGINE/udf/PASTAout.udf
(Output UDF)   /PF_ENGINE/PASTA/PASTA-2.8/sample/out/out_pasta.udf
(Summary UDF) /PF_ENGINE/PASTA/PASTA-2.8/sample/out/sum_pasta.udf

(python)       pasta_python.py  pasta_stress.py  pasta_for_gt2gw.py
(Action)       pastaplot.act
```

[1]GOURMET 上で一連の操作方法を習得することに主眼を置いてサンプルを用意している。そのため計算回数等には短時間で計算が終了するような条件を設定している点に注意すること。

- 解析ツール

```
(.exe)    /PF_ENGINE/PASTA/tools/bin/$(PF_ENGINEARCH)/gt2et(.exe)
                                                      gt2gw(.exe)
                                                      smooth(.exe)
(script)  /PF_ENGINE/PASTA/tools/bin/$(PF_ENGINEARCH)/gwsmooth
(sample)   z10.gt(for gt2et and gt2gw)   z10.fft(for gwsmooth)
```

$(PF_ENGINEARCH) は使用する OS タイプによって異なる。

FORK による PASTA 用入力 UDF の作成

ここでは分子量分布 $M_w/M_n = 2.5$ を有する平均分子量 $M_w = 200,000$ のポリスチレンを試料として、定常ずり粘度を計算する場合を想定している。

1. 入力 UDF

   “in_fork.udf”

   試　料: ポリスチレン $M_e = 14480$

   τ_e は未定のため -1.0 を入力する。これにより時間の単位が τ_e 単位となる。

   分子量分布: $M_w/M_n = 2.5$ の log-normal distributoin。

   1000 本の鎖を分子量の異なる 10 成分に分割する。

   流動 (変形) 条件: ずり流動 (Shear flow)

   StrainRate を $0.1/\tau_e$ ($0.1s^{-1}$ ではない) とする。

```
Name            M0                Me     GN0   tau_e     MaxStretchRatio
PS             104             14480     0.2    -1.0                 4.4

ChainType       Mw         Mw_over_Mn     DistributionFunction
linear     200,000                2.5     LogNormal

Type_of_fraction         Num_of_Chain     NumDiv             DivType
Num_of_Chain                    1,000          9             Log

FlowType  DeformationType  StrainRate     dt    MaxTimeStep  IntervalStep
noflow    --                      0.0    1.0            100             1
flow      shear                   0.1    1.0            100             1
```

図 3.1: Editor window (tree view)

2. 出力 UDF
   以下のチュートリアルで使用するため、説明の便宜上、出力 UDF 名を “out_fork.udf” とする。計算を実行すると”out_fork.udf”という PASTA 用の入力 UDF が出力される。

3. FORK の起動
   FORK は GOURMET、またはコマンドラインから起動することができる。

   ◇ GOURMET からの起動

   (a) 入力 UDF の読込み
   File メニューから Open を選択し、“in_fork.udf” を GOURMET 上に読み込む。

   (b) Engine Run ウィンドウでの FORK の選択

   i. Tool メニューから Engine Run を選び、Engine Run ウィンドウを立ち上げる。エンジンリストの中から ‘FORK’ を選択する。

   ii. Input UDF:には “in_fork.udf” を入力する。

   iii. Output UDF:には “out_fork.udf” を入力する。

   iv. Working Dir:にはエンジンを実行するディレクトリを入力する。エンジン実行時にここで指定したディレクトリが作られ、入力 UDF“in_fork.udf” がコピーされる。上書きされてしまうため、元の入力 UDF があるディレクトリを指定してはならない。

   (c) 起動
   RUN ボタンを押すと、Engine Control ウインドウが立ち上がり、計算が開始される。

   (d) 出力 UDF の確認
   計算が終了したら、File メニューの Open コマンドで出力 UDF ができているかどうか確認する。

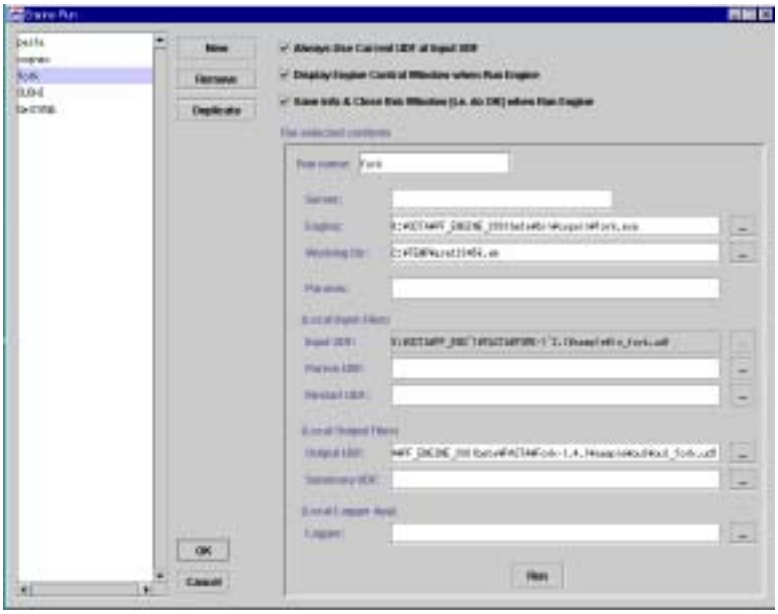

図 3.2: Engine Run window

◇ コマンドラインからの FORK の起動
実行モジュール名 (fork) の後に以下のようにタイプすると、shell プロンプト上で FORK を起動することができる。

```
% fork -I in_fork.udf -O out_fork.udf
```

4. 分子量分布のプロット
GOURMET 上のプロット機能を用いて、分子量分布のプロファイルを表示する事が出来る。詳しくは 6.4.1 を参照。

PASTA による計算

1. 入力 UDF
FORK の出力 “out_fork.udf” を PASTA の入力 UDF として使用する。もし “out_fork.udf” が無い場合は、あらかじめ FORK-1.4.1/sample/out に “out_fork.udf” を用意している。
説明の便宜上、出力 UDF 名を “out_pasta.udf” とする。

2. PASTA の起動

◇ GOURMET からの起動

(a) 入力 UDF の読込み
File メニューから Open を選択し、“out_pasta.udf” を GOURMET 上に読み込む。

(b) Engine Run ウィンドウでの PASTA の選択

i. Tool メニューから Engine Run を選び、Engine Run ウィンドウを立ち上げる。エンジンリストの中から ‘PASTA’ を選択する。

ii. Input UDF:には “out_fork.udf” を入力する。

iii. Output UDF:には “out_pasta.udf” を入力する。

iv. Summary UDF:には “pasta_summary.udf” を入力する。

v. Working Dir:にはエンジンを実行するディレクトリを入力する。エンジン実行時にここで指定したディレクトリが作られ、入力 UDF“in_fork.udf” がコピーされる。上書きされてしまうため、元の入力 UDF があるディレクトリを指定してはならない。

(c) 起動

RUN ボタンを押すと、Engine Control ウインドウが立ち上がり、計算が開始される。Engine Control ウインドウでは、以下のボタンが使える。

Pause : 計算の一時停止
Resume : 停止中の計算の再実行
Stop : 再計算可能な状態で計算終了
Kill : 計算の強制終了 (再計算不能)

(d) 出力 UDF の確認

計算が終了したら、File メニューの Open コマンドで出力 UDF ができているかどうか確認する。

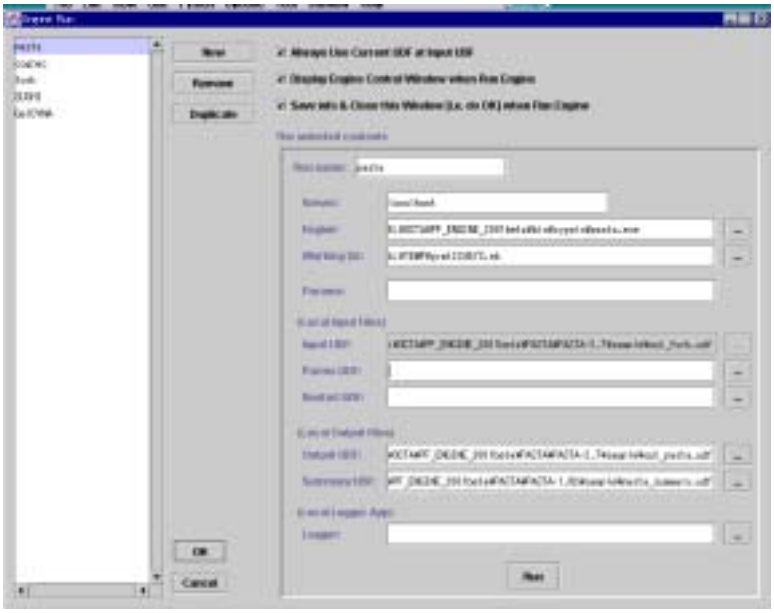

図 3.3: Engine Run window

◇ コマンドラインからの PASTA の起動

実行モジュール名 (pasta) の後に以下のようにタイプすると、shell プロンプト上で PASTA を起動することができる。

```
% pasta -I out_fork.udf -O out_pasta.udf
```

3. 出力データのプロット

PASTA で計算した結果をプロットするには、以下の 2 つの方法が用意されている。

- Action 機能を用いる方法

(a) 出力 UDF の読込み

出力 UDF ファイルを読み込む。

(b) Action の実行

Tree panel(画面左上部) の出力 UDF ファイル名を右クリックすると、選択可能な Action コマンド が表示される (Fig.3.5)。PASTA では以下の Action コマンドが用意されている。

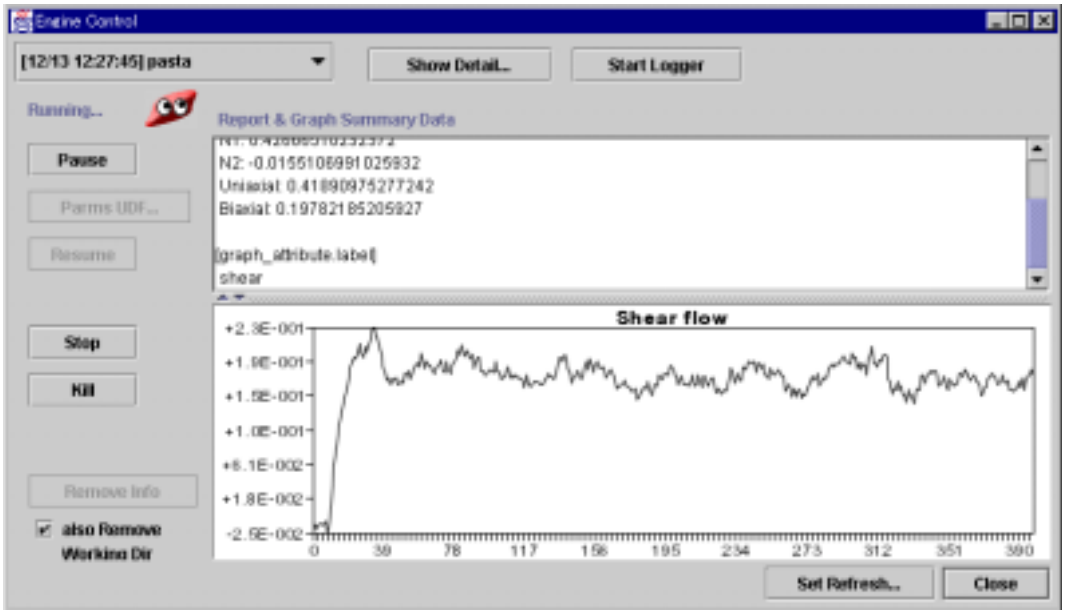

図 3.4: Engine Control window

– plot_Stress
  変形に応じた応力の時間発展がプロットされる (Fig.3.6)。
– plot_Viscosity_or_RelaxationModulus
  変形に応じた粘度、あるいは緩和弾性率の経時変化がプロットされる。

図 3.5: Executing Action command

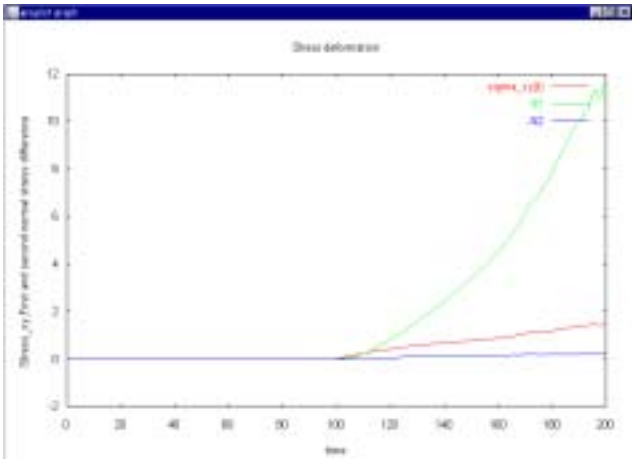

図 3.6: Example plot created by plot_stress

- Python スクリプトを用いる方法
  (a) 出力 UDF の読込み
  計算後の出力 UDF ファイルを読み込む。
  (b) Python スクリプトのロード
  Load ボタンを押して、Python スクリプト ‘‘pasta_python.py’’ をロードし、Run ボタンを押してスクリプトを実行する。プロット用のデータが GraphSheet に生成される (Fig.3.8)。

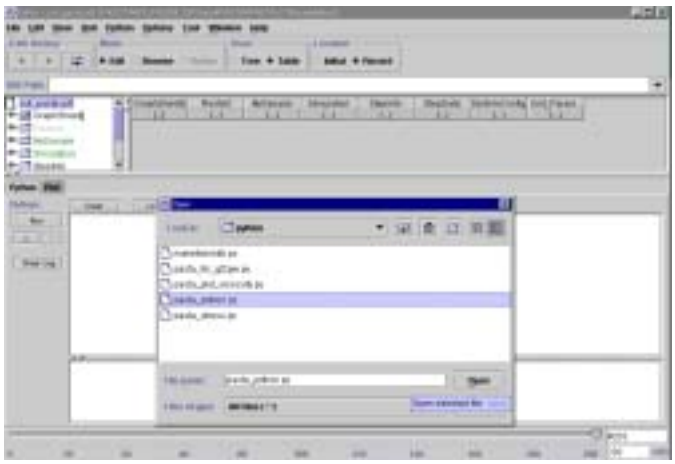

図 3.7: Load Python script “pasta_python.py”

  (c) Make の実行
  GraphSheet のデータを表示させて、Plot パネルの Make ボタンを押すと、以下のような gnuplot 形式のスクリプトが Plot パネルに自動的に作成される。同時に、GraphSheet と同じデータを持つテキストファイル “plot.dat” が GOURMET の起動ディレクトリに生成される。

  例)

```
# plot command template for platform
# datafile name is fixed as "plot.dat" in the current version
set title "GraphSheet[]"
plot "plot.dat" using 1:2 title 'Time' with lines ,
```

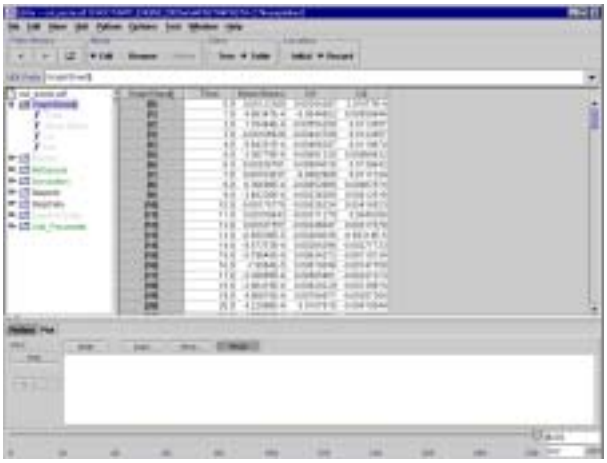

図 3.8: GraphSheet

```
"plot.dat" using 1:3 title 'Shear Stress' with lines ,
"plot.dat" using 1:4 title 'N1' with lines ,
"plot.dat" using 1:5 title 'N2' with lines
```

(d) グラフのプロット
ここで、例えば Shear Stress と Time との関係をプロットするならばスクリプトを以下のように変更し、Plot ボタンを押すとプロットが表示される。

```
# plot command template for platform
# datafile name is fixed as "plot.dat" in the current version
set title "GraphSheet[]"
plot "plot.dat" using 2:3 title 'Shear Stress vs. Time' with lines
```

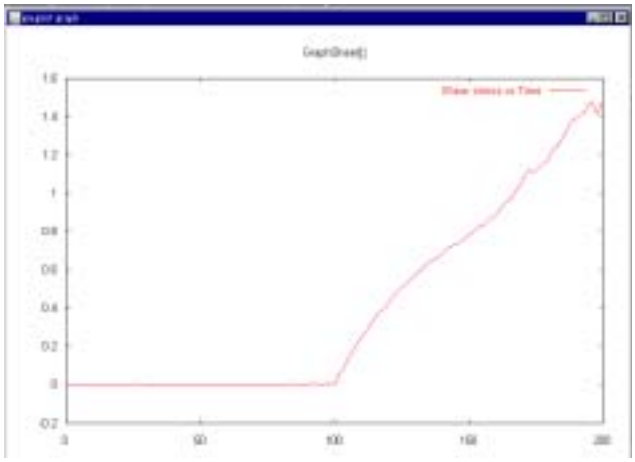

図 3.9: Plot of shear stress vs. time created by Python script ‘‘pasta_python.py’’

もちろんスクリプトには他の gnuplot コマンドを自由に用いることができる。また “plot.dat” は一般的な表計算ソフトで読み込むことも可能である。同様にして、Python スクリプト “pasta_plot_viscosity.py” を用いて粘度をプロットすることができる。.

4. データ解析用 Python スクリプト

Python スクリプト “pasta_for_gt2gw.py” を使用して、PASTA の出力 UDF を変換し、データ解析プログラム (gt2et, gt2gw) の入力データ用のテキストファイルを作成する事ができる。

リスタート

1. 入力 UDF
   リスタート用の入力 UDF“restart.udf” には、先行して計算した出力 UDF“out_pasta.udf” が Restart.UDFname として設定されている。この入力 UDF を読み込んで計算を開始することで “out_pasta.udf” のリスタートが実現する。もし “out_pasta.udf” が無い場合は、あらかじめ FORK-1.4.1/sample/out に用意している。

FlowType	DeformationType	StrainRate	dt	MaxTimeStep	IntervalStep
flow	biaxial	0.05	1.0	100	1

2. 出力 UDF
   説明の便宜上、出力 UDF 名を “restart_out.udf” とする。

3. PASTA の起動

   (a) 入力 UDF の読込み
   入力 UDF“restart.udf” を GOURMET 上に読み込む。

   (b) Restart.UDFname が正しく設定されていることを確認する[2]。

   (c) GOURMET あるいはコマンドラインから PASTA を起動する。詳細は 12 ページ参照。

[2] UDFname は絶対パスかワーキングディレクトリからの相対パスで指定する必要がある。

第4章　適用事例

ここでは PASTA のサンプルデータについて説明する。サンプルデータは以下のディレクトリ下に置いている：

```
PF_ENGINE/PASTA/PASTA-2.8/sample
```

ごく短時間で計算が終了するサンプルデータを 4.1 テストサンプルに用意した[1] 。4.2 ずり粘度、4.3 緩和弾性率、4.4 一軸伸長粘度、4.5 星型高分子のずり粘度に実用的な計算例とそのサンプルデータを示した。4.6 注意事項に PASTA で計算を行う上での一般的な注意事項を記した。

4.1　テストサンプル

単分散高分子のずり流動のシミュレーションを行う。

入力サンプル UDF ファイル名　“pasta_test_in.udf”
出力サンプル UDF ファイル名　“pasta_test_out.udf”

$Z = 10$ の 100 本の鎖を用意する。MaxStretchRatio は Polystyrene を想定した値である。

計算条件

```
Z =10, MaxStretchRatio =4.4, NumChains =100
```

系の平衡化 (FlowType = ‘noflow’) は 100 ステップ行う。現実的な計算ではもっと長時間の平衡化が必用である (次節参照)。その後で、ずり流動 (FlowType = ‘flow’ とし、 DeformationType = ‘shear’ とする) を 100 ステップ行う。IntervalStep = 10 とすると系の応力は 10 ステップ毎に出力される。パラメータの詳細は 5.2UDF 解説参照のこと。

FlowType	DeformationType	StrainRate	dt	MaxTimeStep	IntervalStep
noflow	--	0.0	1.0	100	10
flow	shear	0.01	1.0	100	10

4.2　ずり粘度

単分散直鎖高分子の定常ずり流動のシミュレーションにおいて、分子量 (Z) の異なる試料におけるずり粘度のずり速度依存性の結果を図 4.1 に示す。100 本の鎖を用意し、MaxStretchRatio は 4.4、時間刻みは $dt = 1.0$ として計算した。各分子量において以下に記したステップ数の平衡化を行い、その後、各ずり速度において定常ずり流動のシミュレーションを行いずり粘度を求めた。

Z	ポリマー数 (本)	平衡化 (steps)	ずり流動 (steps)
5	100	1000	10000
10	100	10000	100000
20	100	50000	500000
30	100	100000	1000000
60	100	1000000	10000000

[1] 高分子の数、計算回数等には現実的でない条件を設定している。

ここで用いたサンプルの中から、$Z = 10$ の入力 UDF と 出力 UDF を以下に一例として置いた：

入力サンプル UDF ファイル名　“pasta_shear_in.udf”
出力サンプル UDF ファイル名　“pasta_shear_out.udf”

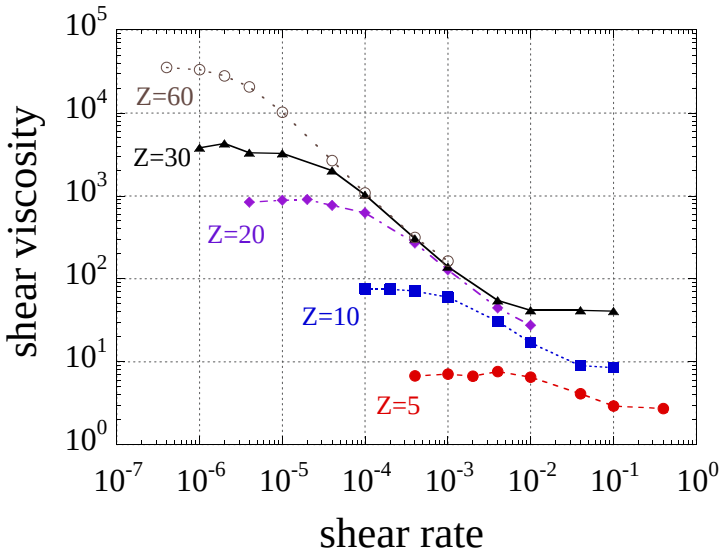

図 4.1: ずり粘度のずり速度依存性

4.3　緩和弾性率

直鎖単分散試料 ($Z = 20$) の緩和弾性率 $G(t, \gamma)$ の計算例を図 4.2 に示す。ここで、計算で得られたずり応力の値をずり歪みで割って $G(\gamma, t)$ としている。横軸は歪みを与えてからの経過時間である。$\gamma \leq 0.5$ の時、$G(t, \gamma)$ は γ に対し鈍くなるので、$G(t, 0.5)$ を線形緩和弾性率 $G(t)$ と見なした。ここで用いた、サンプルデータの一例を以下に置いた。

入力サンプル UDF ファイル名　“pasta_step_mono_in.udf”
出力サンプル UDF ファイル名　“pasta_step_mono_out.udf”

計算条件

```
Z =20, MaxStretchRatio =4.4, NumChains =10000

FlowType  DeformationType  Strain     dt   MaxTimeStep   IntervalStep
noflow    --                  0.0    1.0        100000            100
step      shear               0.5    0.0             0              0
noflow    --                  0.0    1.0        100000              1
```

また多分散試料へ適用した例として、5 成分からなる系を対象として計算した。計算条件は以下の通り：

入力サンプル UDF ファイル名　“pasta_step_poly_in.udf”
出力サンプル UDF ファイル名　“pasta_step_poly_out.udf”

計算条件

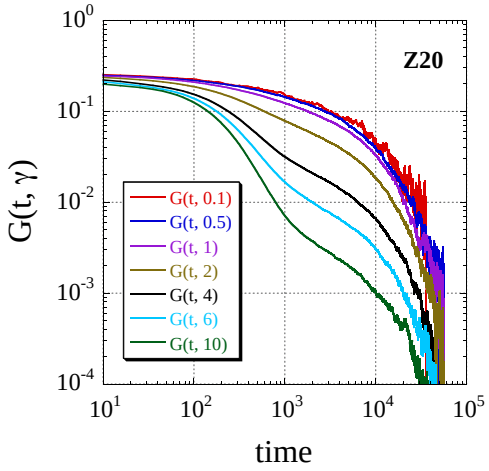

図 4.2: 直鎖単分散試料の緩和弾性率 ($Z = 20$)

Components	Z	NumChains	MaxStretchRatio
component-1	44.1	30	4.4
component-2	35.3	403	4.4
component-3	26.4	2743	4.4
component-4	17.6	5932	4.4
component-5	8.8	892	4.4

FlowType	DeformationType	Strain	dt	MaxTimeStep	IntervalStep
noflow	--	0.0	1.0	100000	100
step	shear	0.5	0.0	0	0
noflow	--	0.0	1.0	100000	1

図 4.3 に緩和弾性率を示す (γ=0.5)。$G(t)$ を解析ツール (レオロジー解析プログラムの説明 (70 ページ) 参照) を用いてフーリエ変換し、動的粘弾性測定から求まる貯蔵弾性率 G'、損失弾性率 G'' を求めた結果を図 4.4 に示す。

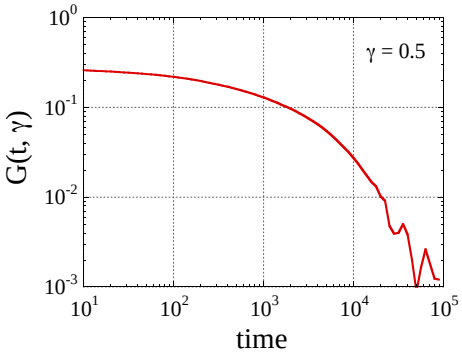

図 4.3: 直鎖多分散試料の緩和弾性率 ($\gamma = 0.5$)

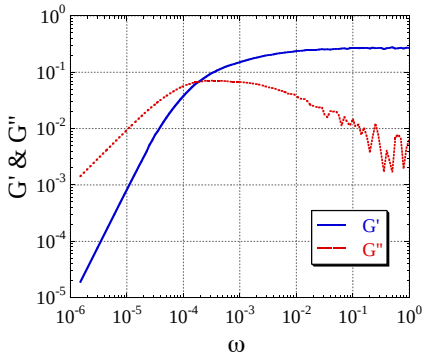

図 4.4: 直鎖多分散試料の貯蔵弾性率及び損失弾性率 (G', G'')

4.4　一軸伸長粘度

直鎖単分散試料の一軸伸長粘度 $\eta_E^+(t)$ の結果を図 4.5 に示す。一軸伸長応力 $(= N_1 + N_2/2)$ を歪速度で割って $\eta_E^+(t)$ を求めた。横軸は歪みを与えてからの経過時間である。サンプルデータの一例を以下に示す。

入力サンプル UDF ファイル名　“pasta_unielong_in.udf”
出力サンプル UDF ファイル名　“pasta_unielong_out.udf”

計算条件

```
Z =20, MaxStretchRatio =4.4, NumChains =10000

FlowType  DeformationType  StrainRate     dt    MaxTimeStep   IntervalStep
noflow    --                    0.0      1.0          20000             20
flow      uniaxial              0.001    1.0         100000              1
```

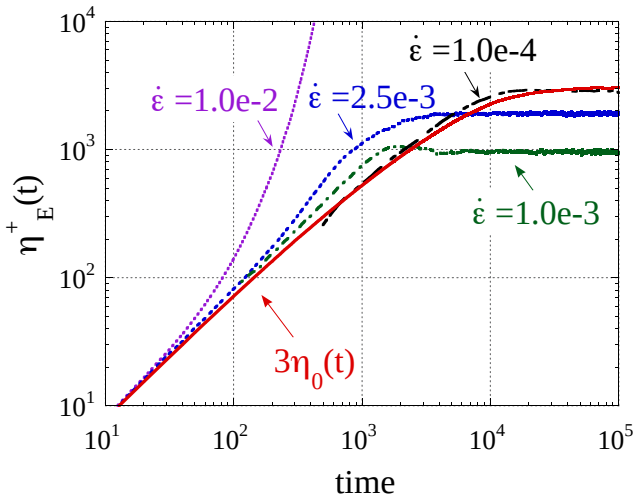

図 4.5: 一軸伸長粘度の時間発展

4.5　星型高分子のずり粘度

直鎖単分散試料 $(Z = 5, 10, 20, 30, 60)$、及び星形試料 $(Z_a = 5, 10, 15)$ の定常ずり粘度の結果を図 4.6 に示す。ここで、Z_a は腕の分子量 M_a を絡み合い点間分子量 M_e で割った量である。ここで用いた、星型高分子のサンプルデータの一例を下記に示す。

入力サンプル UDF ファイル名　“pasta_star_in.udf”
出力サンプル UDF ファイル名　“pasta_star_out.udf”

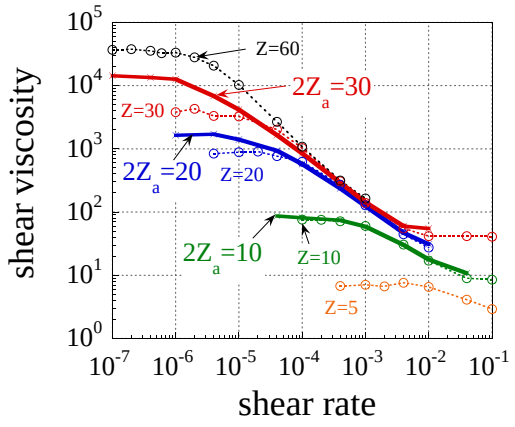

図 4.6: 直鎖及び星型高分子の定常ずり粘度

計算条件

```
Za =10, MaxStretchRatio =4.4, NumChains =100

FlowType  DeformationType  StrainRate     dt    MaxTimeStep   IntervalStep
noflow    --                      0.0    1.0          10000            100
flow      shear                   0.01   1.0         100000             10
```

4.6 注意事項

系の平衡化

PASTA では目的とする流動・変形を加える前に、初期構造の緩和 (平衡化) をおこなう必要がある。少なくとも、平衡化中の系の NumLinks(average number of slip links) や応力の値が一定値に達してから計算を開始するべきである。より厳密には、少なくとも最長緩和時間 τ_d の数倍のタイムステップの平衡化が必要である。

NumChains 及び MaxTimeStep の設定

定常流動 (例えば、定常ずり、など) の場合は、鎖の本数 (NumChains) は比較的小さい値、例えば 100 本程度で十分である。歪速度が $1/\tau_d$ よりも遅い場合、計算ステップ数 (MaxTimeStep) は τ_d よりも長く設定する必要がある。一方、歪速度が $1/\tau_d$ よりも速い領域では、MaxTimeStep は $1/(strainrate)$ より十分長く設定すればよい。

時間発展のシミュレーション (例えば、応力緩和、伸長粘度、など) の場合は、時間平均を取ることができないので、より多くの鎖が必要である。

単位系

PASTA で用いる単位系は、時間、分子量、及び応力について、各々、τ_e、M_e、及び $G_e = (4/15)G_N$ である。Unit_Parameter の中で、τ_e、M_e、及び G_N に具体的な値を設定すると、GOURMET によって簡単に単位変換ができる。詳細は 'GOURMET 操作マニュアル' 及び 5.2 節参照のこと。

第5章　リファレンス

5.1　起動方法

5.1.1　入力 UDF の編集

この節では GOURMET による 入力 UDF の編集方法について解説する。もちろん入力 UDF は別のテキストエディターを用いて直接編集することもできる。いくつかの 入力 UDF のサンプルを、以下のディレクトリに置いた。

```
PF_ENGINE/PASTA/PASTA-2.7/sample/
```

1. ファイルの読み込み
   サンプルとして用意した “PASTAin.udf” を 入力 UDF の雛形として使うことができる。以前に計算に用いた 入力 UDF ファイルにを修正して計算に用いることもできる。

2. 入力 UDF の編集

   (a) Sample
   Sample.Linear[] and/or Sample.Arm[] に必要な値を入力する。Linear[] は直鎖高分子の、Arm[] は星形高分子の腕の分子量成分である (PASTA では星形高分子のレオロジーについて、腕の本数に依存しない結果を与える。)。Sample に分子量分布を生成する際、Action コマンドの一つである simpleFORK を使用することができる。Action については 7.1 の節を参照のこと。

   (b) Simulation
   計算条件を入力する。

   (c) ファイルの保存
   入力が終了したら、File/Save As を用いてファイルを保存する。新規ファイル名を入力する必要がある。

5.1.2　GOURMET 上での PASTA の起動

1. 環境変数
   PASTA の起動には出力定義 UDF が必要である。デフォルト名は “PASTAout.udf” であり、GOURMET 起動時に自動的に環境変数 UDF_DEF_PATH が設定されて “PASTAout.udf” はそこへコピーされる。もし PASTA が起動しない場合、そのディレクトリに “PASTAout.udf” 存在するかどうか確認する必要がある。

2. 入力 UDF の読み込み
   テキストエディタで編集した場合 入力 UDF を GOURMET 上に読み込む。GOURMET 上で編集した場合は編集したファイルをそのまま用いるので、新たに読み込む必要はない。

3. PASTA 実行ファイル (エンジン) の登録 (未登録の場合)

   (a) Tool メニューから Engine Run を選び、Engine Run ウィンドウを立ち上げる。

   (b) New ボタンを押し、Run name:に適当な名称、例えば ‘PASTA’ と入力する。

図 5.1: GOURMET による 入力 UDF の編集

(c) Engine:に PASTA 実行ファイルをフルパス (full path name) で入力する。この時、... ボタンを使ってファイル (pasta 又は pasta.exe) を指定することができる。

(d) 必要に応じて Working Dir: を変更する。計算開始時に GOURMET は 入力 UDF をこのワーキングディレクトリーにコピーする。Working Dir: に 入力 UDF のあるディレクトリを指定すると、入力 UDF を上書きしてしまうため、元の 入力 UDF のあるディレクトリ以外を指定しなければならない。

(e) Input UDF: は 通常、GOURMET により自動的に入力される。

(f) Output UDF: に 出力 UDF 名を入力する。出力 UDF はワーキングディレクトリに出力される。

(g) Summary UDF:に適当な Summary UDF 名を入力する (例えば “pasta_sum.udf”)。Summary UDF: を指定しないと Report & Graph Summary モニター機能がサポートされない。

(h) 他のフィールドには入力する必要はない。

例)

```
Run name:       PASTA
Engine:         PF_ENGINE/bin/pasta(.exe)
Input UDF:      (入力 UDF のパス名。自動的に入力される)
Output UDF:     pasta_test_out.udf
Summary UDF:    pasta_sum.udf
```

4. PASTA の起動

Run ボタンを押すと、Engine Control window が立ち上がり、計算が開始される。Summary UDF: が設定されていれば、計算の進行を反映して応力値や応力の履歴のグラフが表示される。Summary UDF へ応力を書き出すタイミング n_s は、最初は 入力 UDF で定められた IntervalStep に等しいが、データの

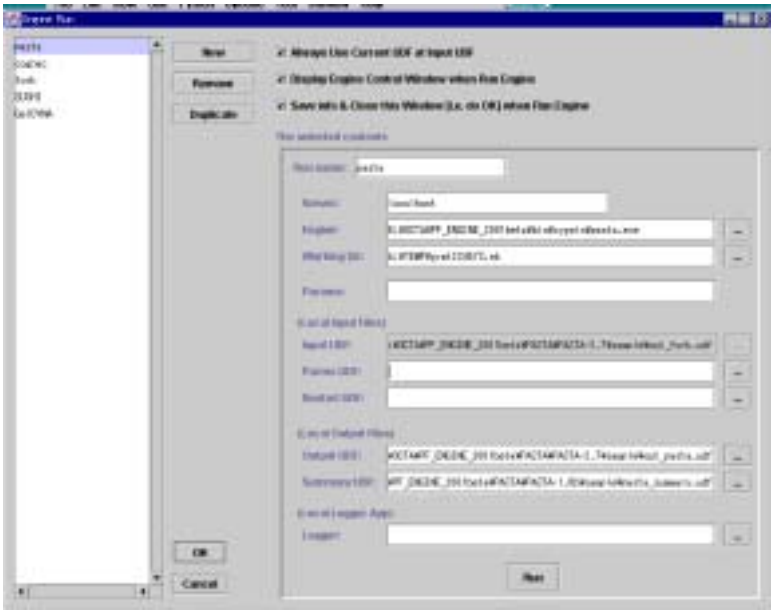

図 5.2: Engine Run window

数が 512 を越える毎に、n_s は 4 倍づつ増加し、履歴のデータ数は $512/4 = 128$ に減少する。
またこの画面で実行中の計算の Pause、 Resume、Stop、Kill 等の操作をおこなうことができる

Pause :	計算の一時停止
Resume :	停止中の計算の再実行
Stop :	再計算可能な状態で計算終了
Kill :	計算の強制終了 (再計算不能)

5. 出力 UDF の確認
GOURMET window 上で 出力 UDF を開き、内容を確認する。

6. 出力データのプロット (オプション)
PASTA で出力したレオロジー量について、GOURMET のプロット機能でプロットする事ができる。以下に手順を示す：

   (a) 出力 UDF を読み込む。

   (b) Python スクリプト "`pasta_python.py`" を読み込み、実行する。gnuplot 用のデータが `GraphSheet` に生成される。この時同様のデータを含むテキストファイル "`plot.dat`" も生成される。

   (c) Make の実行
   `GraphSheet` を表示して、plot window の Make ボタンを押すと、gnuplot 用のスクリプトが自動的に生成される：

   例)

```
# plot command template for platform
# datafile name is fixed as "plot.dat" in the current version
set title "GraphSheet[]"
plot "plot.dat" using 1:2 title 'Time' with lines , \
     "plot.dat" using 1:3 title 'Shear Stress' with lines , \
```

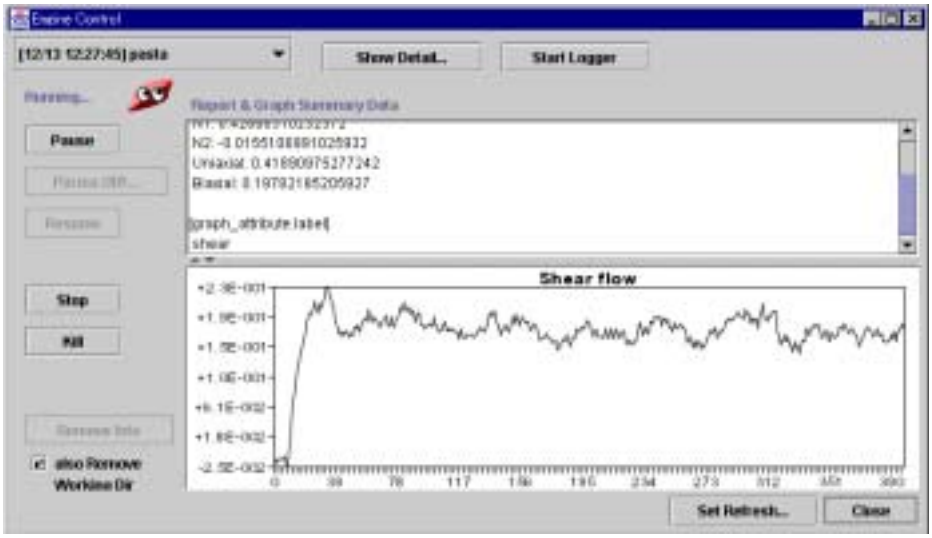

図 5.3: Engine Control window

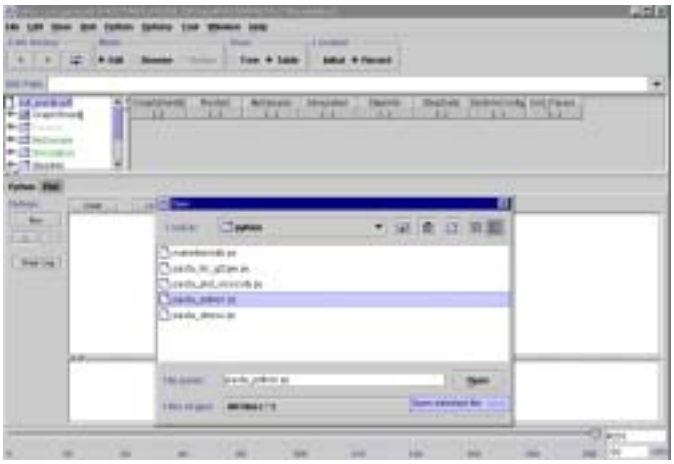

図 5.4: Python スクリプト “pasta_python.py” の読み込み

```
"plot.dat" using 1:4 title 'N1' with lines , \
"plot.dat" using 1:5 title 'N2' with lines
```

はコメント文の開始を意味している。ここで、例えば Shear Stress と Time との関係をプロットするならばスクリプトを以下のように変更する：

```
# plot command template for platform
# datafile name is fixed as "plot.dat" in the current version
set title "GraphSheet[]"
plot "plot.dat" using 2:3 title 'Shear Stress' with lines
```

GOURMET の起動ディレクトリに、テキストファイル “plot.dat” が生成される。もちろん “plot.dat” は一般的な表計算ソフトで読み込むことも可能である。

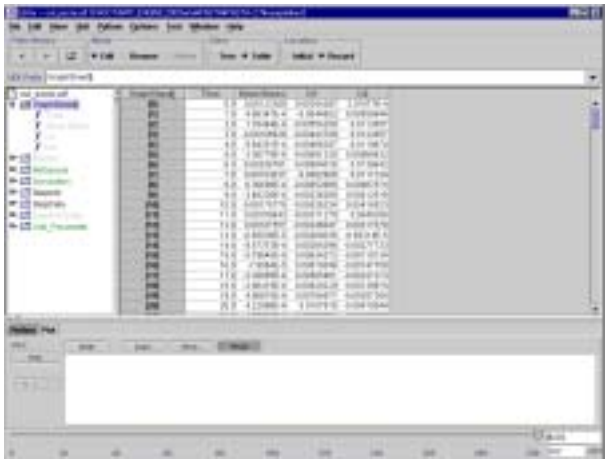

図 5.5: GraphSheet

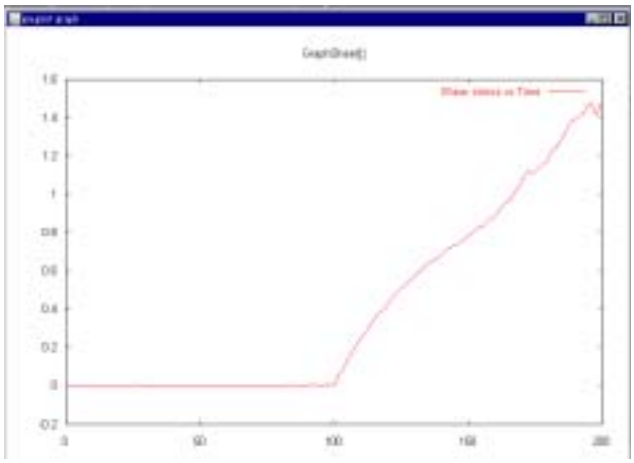

図 5.6: Python スクリプト “pasta_python.py” によるプロット例

5.1.3 コマンドラインからの PASTA の起動

PASTA はコマンドラインからの実行も可能である。そのためには、入力 UDF を含むディレクトリで、shell プロンプトに続いて以下のようにタイプすればよい：

```
% pasta -I inputUDF  -O outputUDF
```

もし起動できない場合は、環境変数 UDF_DEF_PATH が正しく設定されているかどうか、また出力定義 UDF “PASTAout.udf” がディレクトリ $UDF_DEF_PATH にあるかどうかを確認する。また、cygwin から実行する場合、環境変数 CYGWIN_UDF_PATH の値を unix に設定してみる。

PASTA-2.8 からは、計算の進行状況が logfile（デフォルトは標準エラー出力）に出力されるようになった。この出力をしないようにするには、起動時に -q というオプションをつける。

5.2 UDF 解説

PASTA を実行するには、

“入力 UDF” と “出力定義 UDF”

の 2 つの UDF ファイルが必要である。但し、PASTA 用の “出力定義 UDF” ファイルである PASTAout.udf は、PF_ENGINE/udf にインストールされており、このディレクトリが UDF_DEF_PATH に含まれれば自動的に検索されるので、ユーザが指定する必要は無い。

“出力 UDF”

は常に新規作成される (既存の 出力 UDF に追記は出来ない)。
出力 UDF の最後に計算終了時の状態が書かれるので、それをもとにリスタートが可能である。その場合、次のシミュレーションの入力 UDF の中の Restart.UDFname にリスタートに用いる既存の出力 UDF 名を指定する。

以下に、入力 UDF と 出力 UDF とについて詳細に述べる。

5.2.1 入力 UDF 解説

FORK の出力 UDF をそのまま PASTA の 入力 UDF として用いる事ができる。詳細は FORK のマニュアルを参照のこと。ここでは自分で入力 UDF を作成する場合について説明する。

入力 UDF に記述しなくてはならない情報は、

計算対象 と 計算手順

の 2 つである。

計算対象としては

Restart あるいは Sample

のどちらか一方のみを指定することができる。

```
//----- Restart -----

Restart: {
        UDFname: string    " name of restart UDF file"
}
```

ここで、UDFname には過去に実施した計算の出力 UDF 名を指定する。出力 UDF は フルパス名で入力するか、もしワーキングディレクトリにある場合は、ファイル名のみを入力する。

一方 Sample の定義は以下の通り：

```
//----- Sample -----

class Component: {              // Molecular weight component
    Z: double[Me]               " Z = MW/Me"
    MaxStretchRatio: double     "Max Chain Stretch or Max Stretch ratio"
    NumChains: int              "number of chain"
};

Sample: {
```

```
    Linear[]: Component
    Arm[]: Component
};
```

Linear[] と Arm[] とが、各々直鎖、星形高分子の分子量成分に対応する。Linear[] 又は Arm[]、或いはその両方を与えることができる。各々の分子量成分は、Component として表され、その要素は以下の通りである：

```
    Z:
            (分子量)/(絡み合い点間分子量)
            (正の実数)

    MaxStretchRatio:
            (伸び切り鎖の長さ)/(平衡時の contour length)
            (0 または正の実数)
             0 を入力すると、幾らでも伸びるガウス鎖で計算される。
    NumChains:
            この分子量成分の分子鎖の本数
            (正の整数)
             0 でもエラーとはならないが、その分子量成分は無視される。
```

Restart 又は Sample のどちらか一方を指定する。両方は指定できない。
次に、以下のような計算手順を与える：

```
//----- Simulation -----

class Deformation:{
    FlowType: select{
                    "noflow",
                    "flow",
                    "step"
    }                           "Type of Flow. PASTA Input condition"

    DeformationType: select{
                    "shear",
                    "uniaxial"
                    "biaxial"
                    "planar"
    }                           "Type of Deformation. PASTA Input condition."

    Strain: double              "strain (only for FlowType==’step’)"
    StrainRate: double[1/Tau_e] "strain rate. (only for FlowType==’flow’) "
    dt: double[Tau_e]           " TimeStep per 1 tau_e "
    MaxTimeStep: int            " number of iterations"
    IntervalStep: int           "output stress every IntervalStep steps"
}

Simulation: {
    Deformations[]: Deformation
};
```

Simulation は Deformation の配列である。Deformation は 1 つの変形を表し、それを与えられた順に 1 つづつ資料に適用していくことで Simulation が構成される。Deformation の各々の要素の意味は以下の通り：

```
FlowType:
                "noflow"    流動させずに時間発展させる
                "flow"      一定変形速度で流動させる
                "step"      瞬間ひずみを与える
DeformationType:
                "shear"     ずり変形
                "uniaxial"  一軸伸長変形
                "biaxial"   二軸伸長変形
                "planar"    平面伸長変形
                (FlowType=="noflow" の場合は無視される)
Strain:
                FlowType=="step" の場合の歪み量
                (それ以外の場合は無視される)
StrainRate:
                歪速度 (1/tau_e 単位).
                (FlowType が "flow" でない場合は無視される)
dt:
                time step (tau_e 単位).
                通常 1.0 程度で良い。
MaxTimeStep:
                計算ステップ数。
                (FlowType=="step" の場合無視される)
IntervalStep:
                応力を計算して出力するステップ間隔。
                (FlowType=="step" の場合無視される)
```

PASTA-2.8 では、特定の分子鎖を選び、その運動の軌跡をテキストファイル（トラジェクトリファイルと呼ぶ）に出力する機能が追加された。そのテキストファイルはアニメーションプログラム pastanim （7.4 節を参照）の入力に用いることが出来る。

トラジェクトリファイルを出力する場合、PASTA の入力 UDF に、以下で定義される Trajectory を指定する：

```
//----- Trajectory -----

Trajectory: {
ChainID: int "index of the chain whose trajectory will be output"
Interval: int "output interval of the trajectory"
FileName: string "output file name"
}
```

ChainID には、トラジェクトリを出力したい分子鎖の番号を指定する。例えば、

```
Sample: {
    Linear[]: [ // Z MaxStretchRatio NumChains
                { 20             0.0       200 } // Linear[0]
                { 30             0.0        50 } // Linear[1]
```

```
                ]
        Arm[]:  [
                  { 10          0.0       100 } // Arm[0]
                ]
}
```

という Sample の場合、ChainID = 0 なら Linear[0] の最初（0 番目）の分子鎖、ChainID = 200 なら Linear[1] の最初（0 番目）の分子鎖、ChainID = 349 なら Arm[0] の最後（99 番目）の分子鎖を指定したことになる。Interval はトラジェクトリを出力するタイムステップ間隔を指定する。FileName はトラジェクトリファイル名である。

PASTA で用いる単位系について、\begin{unit} と \end{unit} の間に記述されている。各単位の具体的な説明は Unit_Parameter に記述されている。

```
\begin{unit}
const               =   0.2666666666  // 4/15
[Tau_e]             =   {$Unit_Parameter.tau_e}[s]
[Me]                =   {$Unit_Parameter.Me}[g/mol]
[GN0]               =   {$Unit_Parameter.GN0}[MPa]
[Stress]            =   [const*GN0]
\end{unit}

// Unit_Parameter
Unit_Parameter:{
    Name:string         "polymer name"
    tau_e:double[s]     "Unit of time "
    Me:double[g/mol]    "molecular weight between entanglements(g/mol)"
    GN0:double[MPa]     "plateau modulus GN0 (MPa)"
}
```

[tau_e], [Me] 及び [Stress] は、各々、時間の単位、分子量の単位、応力の単位である。 Unit_Parameter の各項目の意味は以下の通り：

Name:	高分子の名前、及び備考。
tau_e:	分子量が M_e と等しい分子鎖のラウス緩和時間。時間の単位。
Me:	絡み合い点間分子量 M_e (g/mol)。
GN0:	プラトーモデュラス G_N^0 (MPa)。

注 :

- FlowType, DeformationType は大文字、小文字を区別しない。例えば ‘noflow’, ‘NoFlow’, ‘NOFLOW’, 等は全て等価である。
- MaxTimeStep が IntervalStep の倍数でない場合、強制的に倍数まで切り上げる。

5.2.2 出力 UDF 解説

コモンデータ部に出力されるもの

多分散系において、$Z <$ Zcutoff の成分は PASTA によって無視される (現在のバージョンでは Zcutoff = 3.0)。NumChains = 0 の成分も無視される。よって計算対象が入力 UDF で記述したものと異なっているかも知れないので注意すること。実際に計算に用いた試料は MySample に以下のように記述される：

```
//----- MySample -----

class MyComponent: {
Ctype: string "chain Type 'Linear' or 'Arm'"
Z: double[Me] "average number of slip link"
MaxStretchRatio: double "max stretch ratio of target polymer"
NumChains: int "number of chain"
WeightFraction: double "weight Fraction"
};

MySample: {
Zn: double[Me] "number-average Z(Average number of slip link)"
Zw: double[Me] "weight-average  Z(Average number of slip link)"
Zz: double[Me] "z-average Z(Average number of slip link)"
TotalChains: int "total number of chains in this sample"
NumComponents: int "number of MyComponents[]"
MyComponents[]: MyComponent
};
```

MyComponent は分子量成分の情報で、WeightFraction(重量分率) を含む。この値は $Z <$ Zcutoff の成分を無視した後で PASTA が計算し直したものである。

MySample は、各分子量成分の情報である MyComponents[] 以外に以下の変数を含む：

Zn:
: 数平均分子量を Me で割ったもの。

Zw:
: 重量平均分子量を Me で割ったもの。

Zz:
: z-平均分子量を Me で割ったもの。

TotalChains:
: 分子鎖の総本数 (1 本の腕は 1 本の鎖と数える)。

NumComponents:
: 分子量成分の数、つまり MyComponents[] の要素数。

Deformation

新しい Deformation を開始する毎に、その Deformation に関する情報が 1 つのレコードとして出力される。

Stress

応力等の計算結果は以下の StepInfo 及び StepData に従って、一つのレコードに出力される。データが出力されるタイミングは、(1) 計算の開始時、(2)IntervalStep ステップ毎、(3) 瞬間ひずみを与えた直後である。

StepInfo は以下の通り：

```
//----- StepInfo -----

StepInfo: {
Iteration: int "total iterations so far"
Time: double [Tau_e]  "current time"
```

```
Gamma: double "shear strain"
Eps: double "uniaxial (Hencky) strain"
};
```

Iteration:
この UDF の最初から数えたタイムステップ数。
FlowType=="step" (瞬間変形) によっては加算されない。
Time:
現在の時間 = Iteration * dt
Gamma, Eps:
これまでの積算のひずみ。

注：リスタートした場合、すべて 0 にリセットされる。

The StepData は以下の通り：

```
//----- StepData -----

StepData: {
TotalStress:    Stress          "total stress of the system"
SubStress[]:    Stress          "stress of each component"
SubData[]:      ComponentData   "length of each component, etc"
};
```

TotalStress:
全応力。
SubStress[]:
各分子量成分の応力。
SubData[]:
各分子量成分の分子鎖の長さ、など。

ここで SubStress[i] は各成分が仮に 100% を占めたとした時の応力である。つまり SubStress[i] に成分 i の重量分率を掛け、i について加えると TotalStress になる。

Stress は以下の通り：

```
//----- Stress -----
class Stress: {
Shear:  double [Stress]     "shear stress"
N1:     double [Stress]     "1st normal stress diff. or  uniaxial stress"
N2:     double [Stress]     "2nd normal stress diff."
};
```

Stress の単位は $3\rho RT/M_e = (15/4)G_N$ である。
Stress の各成分について、DeformationType が 'shear' の時のみ、Shear、N1 及び N2 が、各々、ずり応力、第一法線応力差、第二法線応力差の意味を持つ。各種伸長流動の時は、Stress.Shear に意味はなく (平均すると 0)、N1 及び N2 に次のような意味を持つ：

- 一軸伸長変形 (Uniaxial elongation)

$$\begin{cases} v_x = \dot{\varepsilon} x & \sigma_E = \sigma_{xx} - \frac{1}{2}(\sigma_{yy} + \sigma_{zz}) \\ v_y = -\frac{1}{2}\dot{\varepsilon} y & \quad = N_1 + \frac{1}{2}N_2 \\ v_z = -\frac{1}{2}\dot{\varepsilon} z & \end{cases}$$

- 二軸伸長変形 (Biaxial elongation)

$$\begin{cases} v_x = \dot{\varepsilon} x & \sigma_B = \frac{1}{2}(\sigma_{xx} + \sigma_{yy}) - \sigma_{zz} \\ v_y = \dot{\varepsilon} y & \quad = \frac{1}{2}N_1 + N_2 \\ v_z = -2\dot{\varepsilon} z & \end{cases}$$

- 平面伸長変形 (Planar elongation)

$$\begin{cases} v_x = \dot{\varepsilon} x & \sigma_{P_1} = \sigma_{xx} - \sigma_{yy} = N_1 \\ v_y = -\dot{\varepsilon} y & \sigma_{P_2} = \sigma_{zz} - \sigma_{yy} = -N_2 \\ v_z = 0 & \end{cases}$$

ComponentData は以下の通り：

```
//----- ComponentData -----
class ComponentData:{
NumLinks: double "average number of slip links"
Length: double "average contour length"
};
```

NumLinks:
この分子量成分に属する分子鎖上に
現時刻に存在する slip link 数の平均。

Length:
この分子量成分に属する分子鎖の
contour length の現時刻での平均 (単位 a)。

リスタート用の情報

最終レコードに、計算終了時の系の状態が SystemConfig として出力される：

```
//----- system configurations (for restart) -----

class ChainConfig: {
Ctype: string "'linear' or 'arm'"
Z: double [g/mol] "Z (constant)"
LmaxInv: double "1/Lmax (constant)"
L: double "contour length"
Lt: double "tube length"
xf: double "tail length at the front of the tube"
xb: double "tail length at the back of the tube"
Rfirst: Vector3D "position of the first slip link"
NumLinks: int "number of slip links"
Links[]: LinkConfig "config. of each slip link"
};

SystemConfig: {
TotalChains: int "total number of chains"
NumComponents: int "number of components"
NumChains[]: int "number of chains in each component"
Chains[]: ChainConfig "config. of each chain"
};

//----- Vector3d -----

class Vector3d: {
    x:              double
    y:              double
    z:              double
}
```

レコード並びの例

たとえば、

```
Simulation: {
[
//FlowType  DeformType  strain   SRate    dt   MaxStep   Interval
{ "noflow"   ""           0.0    0.0     1.0       100       10 }
{ "step"     "shear"      1.0    0.0     0.0         0        0 }
{ "flow"     "shear"      0.0    0.001   1.0       100       10 }
]
}
```

という変形を与えた場合、出力されるレコードは：

```
Record  0:   初期状態 (t=0) での応力
```

```
Record  1:   Deformation[0] = { "noflow"  ""  0.0  0.0  1.0  100  10 }
Record  2:   t=10 での応力
Record  3:   t=20 での応力 ...
...
Record 11:   t=100 での応力
Record 12:   Deformation[1] = { "step"  "shear"  1.0  0.0  0.0  0 0 }
Record 13:   瞬間変形直後の応力 (t=100)
Record 14:   Deformation[2] = { "flow"  "shear"  0.0  0.001  1.0  100  10 }
Record 15:   t=110 での応力 ...
...
Record 24:   t=200 での応力
Record 25:   SystemConfig (for restart)
```

第6章　- FORK - The support tool for PASTA

6.1　FORKについて

FORK は、PASTA の Input UDF を作成する支援ツールである。。FORK は、ユーザーが実験等から求めた平均分子量や分布を基に、例えば Poisson 分布や対数正規分布などの任意の分子量分布を作成する機能を持つ。また、GPC 測定データを取り込む事もできる。

FORK の入力ファイルにおいては、次の 2 つの入力項目がある。

- 作成したい分子量分布関数のタイプおよびそのパラメータ。
  例えば、‘Poisson 分布関数で $M_w = 25000$’。
- PASTA にてシステムに与える変形などの計算条件。例えば、‘Steady shear flow で shear rate $= 0.01/s$’。

この様に FORK は、分子量分布を作成するだけではなく、PASTA の計算条件を予め入力することができる。これによって、FORK の出力ファイルをそのまま PASTA の入力ファイルとして使用することが可能である。

6.2 FORK Overview

6.2.1 分子量分布の作成

分子量分布成分の計算には、以下の項目を必要とする。

- 重量平均分子量 M_w
- 分子量分布 M_w/M_n
- 分布関数の種類
- 最小分子量成分
- 最大分子量成分
- 分割した分子量成分の数
- 分割法

分布関数の種類

高分子の分子量分布や平均分子量は重合メカニズムによって決定される。例えば付加重合反応や重縮合反応では SchultzZimm 分布関数、リビング重合反応では Poisson 分布関数、チーグラー法で得られたものは対数正規分布関数に従うものが多い。FORK で使用可能な分布関数は、高分子の重合特性や実験結果等から以下の 4 種類である。ここで $n(M)$ はモル分布関数、$wf(M)$ は重量分布関数、Γ(X) はガンマ関数、M_0 はモノマー単位分子量である。

1. Schultz-Zimm distribution

$$n(M) = \frac{\beta^{\alpha+1}}{\Gamma(\alpha+1)} M^{\alpha} \exp(-\beta M) \tag{6.1}$$

$$\text{ここで} \quad M_n = \frac{\alpha+1}{\beta}\ , \quad M_w = \frac{\alpha+2}{\beta}$$

2. Poisson distribution

$$n(M) = \frac{1}{M_0} \frac{\nu^{\frac{M}{M_0}}}{\Gamma(\frac{M}{M_0}+1)} \exp(-\nu) \tag{6.2}$$

$$\text{ここで} \quad \nu = \frac{M_n}{M_0}$$

3. normal distribution

$$n(M) = \frac{1}{\sqrt{2\pi}\sigma} \exp[-\frac{(M-\mu)^2}{2\sigma^2}] \tag{6.3}$$

$$\text{ここで} \quad \mu = M_n\ , \quad \sigma^2 = M_n(M_w - M_n)$$

4. log-normal distribution

$$w(M) = \frac{1}{\sqrt{2\pi}\sigma M} \exp[-\frac{(\ln M - \mu)^2}{2\sigma^2}] \tag{6.4}$$

$$\text{ここで}\quad \mu = \frac{\ln M_w + \ln M_n}{2},\quad \sigma^2 = \ln M_w - \ln M_n$$

これらの分布関数は次の規格化条件を満たす。

$$\int_0^\infty n(M)dM = \int_0^\infty w(M)dM = 1 \tag{6.5}$$

また、任意の単分散試料についても、値を直接入力することが可能である。さらに GPC の測定結果については、指定のファイル形式に従って、分子量 (MW)、重量分率 (WeightFraction) を読み込む機能を備えている。

分割した分子量成分の数

FORK では、選択した分布関数を設定した分割数 n によって離散化する。分割数が n であれば、分子量成分は $n+1$ となる。

設定最小および最大分子量

分子量成分作成時の最小分子量成分 $M_{\min}$ と最大分子量成分 $M_{\max}$ を設定する。分割数 n の時、0 番目の成分が最小分子量成分 $M_{\min}$、n 番目の成分が最大分子量成分 $M_{\max}$ である。

デフォルト値 (-1) を入力すると自動的に最小分子量 $M_{\min}$ と最大分子量 $M_{\max}$ を設定する機能を持つ。SchultzZimm 分布および Poisson 分布では、モル分率が下限から 0.05%での分子量を最小分子量成分（Mmin）に、99.95%での分子量を最大分子量成分（Mmax）に設定する。図 6.1 に分割例を示す。

正規分布および対数正規分布においては、それぞれ分布関数のピーク位置から± 3 σ での分子量をそれぞれ設定する。

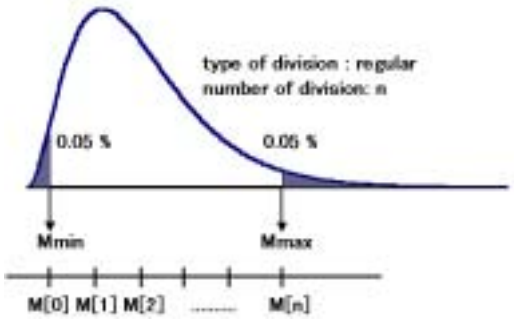

図 6.1: Determination of $M_{\min}$ and $M_{\max}$ for Shultz-Zimm and Poisson distributions.

分割方法と重量分率計算

等間隔に分子量 $M_i (i = 1, 2, \ldots, n)$ を離散化する必要はない。FORK では、分割方法の種類として、分子量を等分割する方法、Log 分割する方法、および重量分率ごとに分割する方法の 3 種類を用意している。等分割および Log 分割での重量分率は、分割によって決められた各分子量成分 M_i の中点 $(M_{i-1} + M_i)/2$ および $(M_i + M_{i+1})/2$ 間の分布関数を積分して離散化する (図 6.2)。重量分率分割では、分割数 n に応じて決まる重量分率毎に分子量 M を分割する (図 6.3)。つまり $w_1 = w_2 = \ldots = w_{n-1} = 1/n$ となる。この場合最大および最小分子量成分の重量分率は、設定した重量分率の半分 $w_0 = w_n = 1/(2n)$ となる。

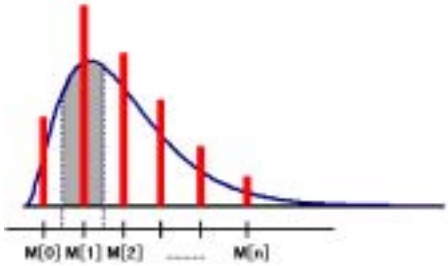

図 6.2: The weight distribution discretized by the linear division scheme.

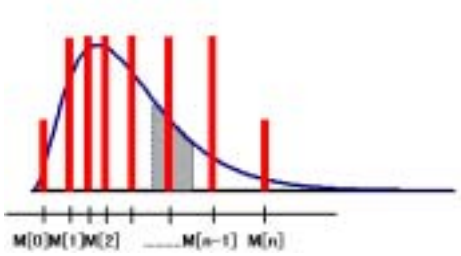

図 6.3: The weight distribution discretized by the weight fraction division scheme.

平均分子量の計算

離散化処理後、離散化された各分子量成分および重量分率から、平均分子量を計算する。これは分割数によって入力値である平均分子量と実際に離散化後の平均分子量が若干異なるためである。平均分子量は以下の式で計算する。M_n は数平均分子量、M_w は重量平均分子量、M_z は z 平均分子量である。

$$M_n = \sum_{i=0}^{n} M_i \cdot n_i = \frac{1}{\sum_{i=0}^{n} \frac{w_i}{M_i}} \tag{6.6}$$

$$M_w = \frac{\sum_{i=0}^{n} M_i^2 \cdot n_i}{\sum_{i=0}^{n} M_i \cdot n_i} = \sum_{i=0}^{n} M_i w_i \tag{6.7}$$

$$M_z = \frac{\sum_{i=0}^{n} M_i^3 \cdot n_i}{\sum_{i=0}^{n} M_i^2 \cdot n_i} = \frac{\sum_{i=0}^{n} M_i^2 w_i}{\sum_{i=0}^{n} M_i w_i} \tag{6.8}$$

ここで、n_i および w_i は、それぞれ i 成分によって規格化されたモル分率および重量分率である。

GPC 測定結果の入力

GPC 測定結果について、分子量および重量分率を以下の形式（分子量、重量分率の順番）に従ってテキストデータ化したファイルを作成した場合に限り、ファイルを読み込む機能を持つ。

Example: `gpc.dat`

```
20000           0.01
28000           0.03
45000           0.08
68000           0.19
90000           0.35
105000          0.15
130000          0.1
156000          0.03
180000          0.02
.....           .....
分子量          重量分率
```

複数サンプルのブレンド

任意の分布を持つ複数のサンプルを任意の重量分率またはモル分率（ここでは chain 数）でブレンドした系についても取扱うことができる。ブレンド方法は、モノモーダルで離散化した成分を比率に応じて足し合わせる。例えば、A サンプルを 9 分割（10 成分）したものと B サンプルを 4 分割（5 成分）したものをブレンドすると全部で 15 成分となる。また、GPC データから読み込んだサンプルに、任意のサンプルをブレンドする事も可能である。

6.2.2 PASTA 用入力ファイルの作成

高分子の物性データ

FORK の入力 UDF において、ユーザーは計算に必要な高分子固有の物性値をわかる範囲で入力ファイルに書き込む必要がある。

- `Name`：高分子名
- `M0`：モノマー単位分子量 M_0 (g/mol) (for Poisson distribution only)
- `Me`：絡み合い点間分子量 M_e (g/mol)
- `GN0`：プラトーモジュラス G_N^0 (MPa)
- `tau_e`：時間の単位 (分子量 M_e の鎖のラウス緩和時間) τ_e (s.)
- `MaxStretchRatio`：高分子鎖の最大伸長比 λ_{max}
- `Temp`：基準温度 T (K)

注意事項

- FORK の入力において唯一絶対に必要な値は M_e である。G_N^0、τ_e、λ_{max}、および T について、もしユーザーが正確な値が分からない時は -1 と入力するのがよい。この時それぞれ、 $G_N^0 = 1$、$\tau_e = 1$、$\lambda_{max} = 100$、および $T = 273$ が設定される。
- τ_e に秒単位で具体的な数値を入力した場合、PASTA の入力条件の一つである歪み速度の入力値の単位が (1/sec.) になるので注意する必要がある。
- 高分子材料の基本物性値を収めた UDF ファイルの説明は、section6.4.2 で行なうのでそちらを参照すること。

PASTA 用入力値への変換

離散化された各分子量成分は、PASTA 用入力値に変換される。各成分分子量 M_i から、絡み合い点の数を意味する $Z_i(= M_i/M_e)$ を求める。また、設定した分子の総本数や重量分率から分子量成分に対応する鎖の本数 n_i を計算する。

PASTA 入力条件の設定

PASTA の入力条件を FORK で設定することが出来る。以下に PASTA に必要な入出力項目を挙げる。

- `FlowType`：流動について設定する。”flow” ”noflow” ”step”の 3 種類から選択する。
- `DeformationType` :変形について設定する。”uniaxial” ”biaxial” ”planar” ”shear”。

- `Strain` :歪み量。瞬間変形時に設定する。(only for flow type = ”step”)。
- `StrainRate` : 変形時の歪み速度。 (only for flow type = ”flow”)。
- `dt` : 1step 当たりの時間。無次元数（τ_e 単位）
- `MaxTimeStep` :シミュレーションの Iteration 回数。
- `IntervalStep` : 設定ステップごと結果をに出力する。

ただし、物性データ入力時に τ_e に秒単位で具体的な数値を入力した場合、歪み速度の入力値の単位が (1/sec.) になるので注意を要する。

6.3 例題

この節では、FORK についての例題をいくつか示す。この節で使われるサンプルファイルは、次のディレクトリにある。

```
PF_ENGINE/PASTA/FORK-1.4.1/sample
```

6.3.1 Sample1

直鎖ポリスチレンについて $M_w = 200000$、$M_w/M_n = 2.5$ で 9 分割させる（10 成分）Chain 数は 1000 本。PASTA の変形条件として、100step 構造緩和させた後、100step の間歪み速度 $0.1/\tau_e$) でずり変形させる。τ_e には、-1.0 を入力する。

Input sample name	“in_fork.udf”
Output sample name	“out_fork.udf”

Input data

- PolymerData[]

Name	PS
Me:	14480
tau_e:	-1.0 (use default)

- MWDist.Sample[]

ChainType	‘linear’
GPC_flag	‘off’
Mw	200000
Mw_over_Mn	2.5
DistributionFunction	LogNormal
Type_of_fraction	Num_of_Chain
DivType	Log
NumDiv	9 (10 components)
Mmin	-1
Mmax	-1 (automatically set by FORK)
Num_of_Chain	1000

- Simulation.Deformation[0]

FlowType	‘noflow’ (thermalization)
Deformation	‘shear’
Strain	0.0
StrainRate	0.0
dt	1
MaxTimeStep	100
IntervalStep	1

- Simulation.Deformation[1]

FlowType	'flow'
Deformation	'shear'
Strain	0.0
StrainRate	0.1
dt	1
MaxTimeStep	100
IntervalStep	1

分子量分布のプロット

Fig.6.4 は、FORK で生成した分子量分布のプロット（鎖の本数 vs. log Z_i) である。プロットについては、section 6.4.1 を参照のこと。

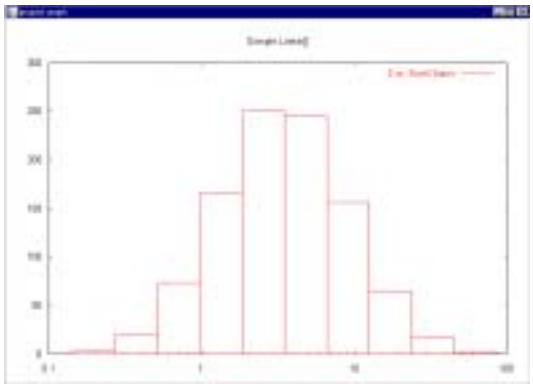

図 6.4: Molecular weight distribution of Sample1 (number of chains vs. log Z_i.)

6.3.2 Sample2

ポリスチレンの GPC ファイル "gpc.dat" からデータを読み込む。PASTA の変形条件として、10step 構造緩和させた後、10step の間歪み速度 $1.0/\tau_e$ でずり変形させる。

Input sample name	"gpcin_fork.udf"
Output sample name	"gpcout_fork.udf"

Input data

- PolymerData[]

Name	PS
Me	14480
tau_e	-1.0 (use default)

- MWDist.Sample[]

ChainType	'linear'
GPC_flag	'on'
GPC_file	"gpc.dat"

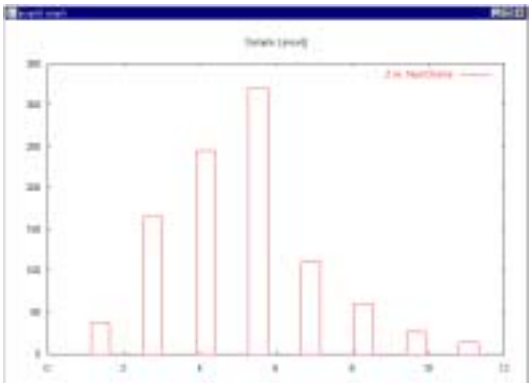

図 6.5: Molecular weight distribution of Sample2 (number of chains vs. Z_i).

分子量分布のプロット

Fig.6.5 は出来た分子量分布のプロットである。

6.3.3 Sample3

ポリスチレン 2Sample をブレンドする。`Sample[0]` は直鎖で $M_w = 200000$、$M_w/M_n = 2.5$、70w%で 10 分割（11 成分)、`Sample[1]` は直鎖で $M = 500000$ の単分散試料を 30w%でブレンドする。

Input sample name	“`blendin_fork.udf`”
Output sample name	“`blendout_fork.udf`”

Input data

- `PolymerData[]`

`Name`	PS
`Me`	14480
`tau_e`	-1.0 (use default)

- `MWDist.Sample[0]`

`ChainType`	‘linear’
`GPC_flag`	‘off’
`Mw`	200000
`Mw_over_Mn`	2.5
`DistributionFunction`	LogNormal
`Type_of_fraction`	WF
`WeightFraction`	0.7
`DivType`	Linear
`NumDiv`	10 (11 components)
`Mmin`	15000
`Mmax`	-1 (automatic)
`Num_of_Chain`	1000

- `MWDist.Sample[1]`

ChainType	‘linear’
GPC_flag	‘off’
Mw	500000
Mw_over_Mn	1.0
DistributionFunction	Discrete
Type_of_fraction	WF
WeightFraction	0.3
DivType	Linear
Mmin	15000
Mmax	-1 (automatic)
Num_of_Chain	1000

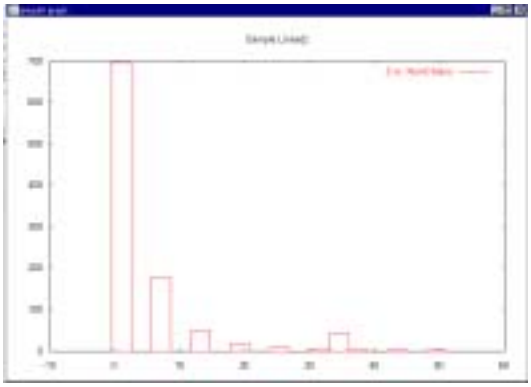

図 6.6: Molecular weight distribution of Sample3 (number of chains vs. Z_i).

分子量分布のプロット

Fig.6.6 は生成した分子量分布のプロットである。

6.4 FORK 操作ガイド

6.4.1 FORK の起動方法

FORK は、GOURMET およびコマンドラインから起動可能である。GOURMET から起動する場合、FORK の定義 UDF である “forkdef.udf” が PF_ENGINE/udf 無くてはならない。万が一 FORK が起動しない場合、これらのファイルについて調べるべきである。

Input UDF の Edit

FORK の InputUDF は、テキストエディタあるいは GOURMET 上で自由に Edit できる。
ここでは、GOURMET 上での操作方法について述べる。

1. Input UDF の読み込み
   メニュー File/Open を使って inputUDF を GOURMET 上に読み込む。サンプルファイルは PF_ENGINE/PASTA/FORK-1.4/sample/にある。

2. Input UDF の Edit

   (a) GOURMET の他ウインドウからの polymer database UDF の読み込み
   メニュー File/ New で GOURMET の別のウインドウを開く。このウインドウにて PF_ENGINE/POLYMERDATABASE/polymerdata.udf ファイルを読み込み (Fig.6.7)、計算したい高分子名を探す。高分子の Properties が複数ある中から、、計算したい Properties の index を決める。

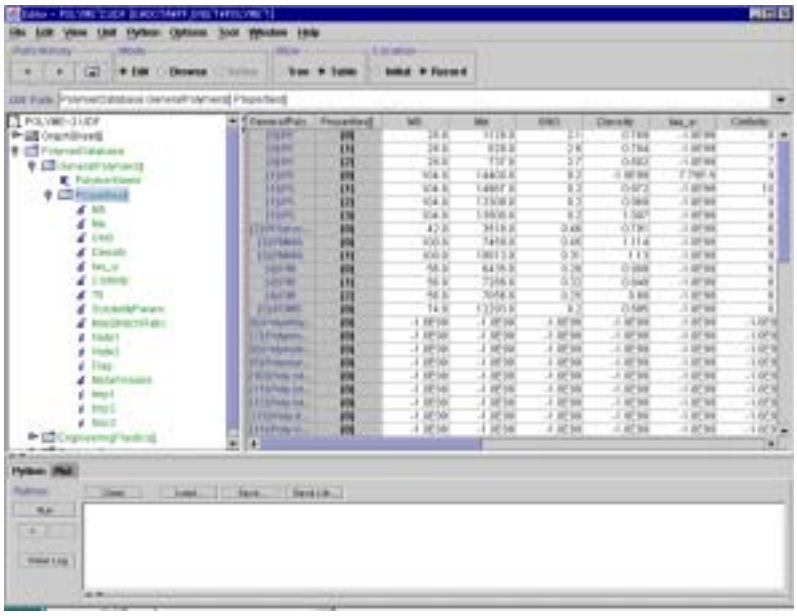

図 6.7: polymer data base UDF “polymerdata.udf”

   (b) Action command set_PolymerData の起動
   Action command set_PolymerData を用いて、input UDF の PolymerData にデータをセットする。set_PolymerData については、section7.1 を参照のこと。

   (c) MWDist の Edit
   メニュー Edit/Insert Row を用いて、MWDist.Sample[] に新しい列を追加する。2 つ以上のブレンドサンプルを入力する場合、 MWDist.Sample[] を必要分追加し、MWDist.TotalChain にシステム全体の鎖の数を入力する必要がある。

(d) Simulation の Edit
Simulation.Deformations[] に列を追加し、シミュレーション条件を入力する。

(e) Save
メニュー File/Save As で Input UDF を保存する。Save を用いると、ファイルは上書きされる。

GOURMET 上での FORK の実行

1. Input UDF の Open
GOURMET で Input UDF を開く。

2. FORK の Engine Run window への登録
Tool/EngineRun を選ぶと、Engine Run ウインドが立ち上がる。FORK 実行ファイルを登録するために New ボタンをクリックし、Run name:に 'FORK' と入力する。次に Engine:に FORK 実行ファイルを full path name で入力する。通常 FORK 実行ファイルは PF_ENGINE/bin/$(PF_ENGINEARCH)/にある fork(.exe) であり、... をクリックしてファイル指定できる。ここで 'OK' をクリックするとエンジンが登録される。エンジン登録は一度行えば以後保存される。

3. Engine Runwindow の入力
再び Engine Run ウインドを立ちあげる。Input UDF:には、編集した inputUDF を入力 Output UDF:には任意のファイル名を設定する。その他のには、何も入力する必要はない。

Example)

```
Run name:       FORK
Engine:         PF_ENGINE/bin/fork(.exe) (path name of the executable file)
Input UDF:      /somewhere/in_fork.udf (set by GOURMET)
Output UDF:     out_for.udf (will be created in the Working Dir)
```

4. FORK の開始 Run ボタンを押すと Engine Run ウインドが立ち上がり計算を開始する。計算終了時には、'Stopped' という表示がされる。

5. Output UDF の確認
計算終了後、Output UDF ファイルを GOURMET から読み込み、中身をチェックする。通常 Output UDF は、そのまま PASTA の Input UDF として扱うことが出来る。

6. 分子量分布のプロット（オプション）
FORK によって作られた分子量分布について、以下の手順で GOURMET 上にプロットすることが出来る。

(a) Output UDF の Open
Output UDF を GOURMET で読み込む。

(b) Sample.Linear[] あるいは Sample.Arm[] の選択
Sample.Linear[]orArm[] をマウスでクリックさせると青くハイライトされ、カラムにデータが表示される。

(c) Make

ウインド下部の python/plot 部の plot を選択すると、plot 関連ウインドが手前に表示される。ここで、Make ボタンを押すと、gnuplot 形式の script が自動的に作成される。

例)

Example)

```
# plot command template for platform
# datafile "plot.dat" is fixed current version
set title "Sample.Linear[]"
plot "plot.dat" using 1:2 title 'Z' with lines ,\
     "plot.dat" using 1:3 title 'MaxStretchRatio' with lines ,\
     "plot.dat" using 1:4 title 'NumChains' with lines
```

ここで、例えば Z と NumChains の関係を plot するならば

```
# plot command template for platform
# datafile "plot.dat" is fixed current version
set title "Sample.Linear[]"
set boxwidth log(0.2)
set logscale x
plot "plot.dat" using 2:4 title 'Z vs. NumChains' with boxes
```

Plot ボタンを押すと Gnuplot 形式で plot が行われる (図 6.8)。 また同時に "`plot.dat`" というテキストファイルが GOURMET 起動ディレクトリ上に出来るので、そのファイルは他の Editor などで読み込む事も可能である。

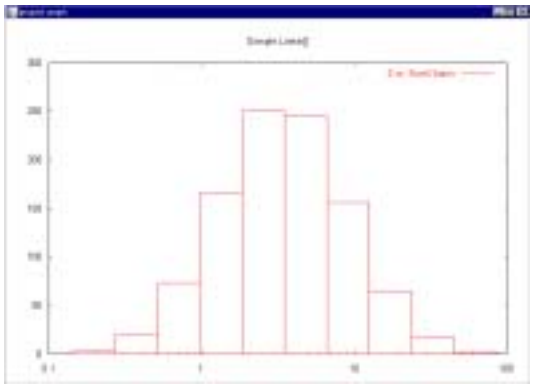

図 6.8: Plot of molecular weight distribution created by gnuplot

コマンドラインからの FORK の起動

FORK は、shell プロンプト上などでコマンドラインから実行可能である。以下に実行例を示す。

`% fork -I` *Input-UDF-name* `-O` *Output-UDF-name* `-D` *Def-UDF-name*

6.4.2 UDF 解説

この節では、FORK で用いる UDF の構造について説明する。

UDF リスト

FORK で用いる UDF について、以下に挙げる。

- 入出力定義 UDF
  入力および出力 UDF の定義部が書かれた UDF。デフォルト名は、“forkdef.udf”。
- Input UDF
  FORK の入力 UDF。
- Output UDF
  FORK の出力 UDF。PASTA の入力 UDF でもある。
- Polymer database UDF
  高分子材料の基本物性値を持つ UDF。

入出力定義 UDF

FORK の入出力定義 UDF 名は、“forkdef.udf” である。
入力 UDF および出力 UDF のヘッダー部には、入出力定義 UDF が include されている。

```
#include forkdef.udf
```

入力 UDF

入力 UDF は、以下の構造からなる。

- PolymerData
  分子量成分計算および PASTA 用入力ファイル作成に用いるポリマーの物性値を設定。入力値は単位を持つ。
- MWDist
  分子量分布の種類およびパラメータを設定。
- Simulation
  PASTA のシミュレーション条件を設定する。

各データ構造について、以下の説明する。

PolymerData

```
PolymerData---Name
            |-M0
            |-Me
            |-GN0
            |-tau_e
            |-Temp
            |-MaxStretchRatio
```

- Name . . . 高分子名
- M0 . . . モノマー単位分子量 M_0 (g/mol)

- Me ... 絡み合い点間分子量 M_e (g/mol)
- GN0 ... プラトーモジュラス G_N^0 (MPa)
- tau_e ... 分子量 $M = M_e$ の鎖のラウス緩和時間。
- T ... 基準温度 (K)
- MaxStretchRatio ... 高分子の最大伸長比 λ_{max}。

注意事項

1. GN0、MaxStretchRatio および T に -1 を入力すると、$G_N^0 = 1$ (MPa)、$\lambda_{max} = 100$ および $T = 273$ (K) が設定される。
2. τ_e に秒単位で具体的な数値を入力した場合、PASTA の入力条件の一つである歪み速度の入力値の単位が (1/sec.) になるので注意する必要がある。一方、τ_e がわからない時 -1 を入力すると、 StrainRate は、 $1/\tau_e$ の単位に変換される。

MWDist

MWDist は次の構造を持つ。

```
MWDist---Sample[]---ChainType
       |           |-GPC---GPC_flag
       |           |     |-GPC_file
       |           |-Mw
       |           |-Mw_over_Mn
       |           |-DistributionFunction
       |           |-Type_of_fraction
       |           |-WeightFraction
       |           |-Num_of_Chain
       |           |-NumDiv
       |           |-DivType
       |           |-Mmin
       |           |-Mmax
       |-TotalChain
```

- TotalChain ... 複数の Sample を含む系全体の chain 本数。Sample[] の Sample[].Type_of_fraction が WeightFraction であるときに有効。

Sample[]

1Sample 当たりの離散化された分子量成分を得るための必要な入力データを設定する。

- ChainType ... Chain が直鎖であるか分岐鎖の腕（Arm) であるかを選択する。”linear” or ”arm”を入力。
- GPC
  - GPC_flag ... 分子量分布および重量分率分布ファイルを読み込ませるフラッグ。
    ‘ON’:読み込む,‘OFF’:読み込まない。
  - GPC_file ... GPC データファイル名を絶対パスで指定する。GPC_flag が ‘ON’ の時有効。
- Mw ... 重量平均分子量

- MW_over_Mn ... 重量平均分子量と数平均分子量の比 M_w/M_n。
- DistributionFunction ... 分布関数の設定。

分布関数の種類	入力文字列
正規分布	'Normal' 'N' 'normal' 'n'
正規対数分布	'LogNormal' 'LN' 'lognormal' 'ln'
Schultz-Zimm 分布	'SchultzZimm' 'SZ' 'schultzzimm' 'sz'
Poisson 分布	'Poisson' 'P' 'poisson' 'p'
離散化データ（単分散）	'Discrete' 'D' 'discrete' 'd'

'Discrete' は単分散試料や離散化されたデータを入力する時に使う。もし、二様単分散試料を作る時などは 2 つの単分散サンプルをブレンドすればよい。

- Type_of_fraction ... 複数サンプルをブレンドする時、1 サンプル当たりのブレンド量を重量分率あるいは分子数のどちらかで入力するときに選択する。重量分率を選択する場合 'WeightFraction' または 'WF' とすると、つぎの Sample[].WeightFraction が有効になる。分子数を選択する場合 'Num_of_Chain' または 'NC' と入力すると、つぎの Sample[].Num_of_Chain が有効となる。
- WeightFraction ... 重量分率。Type_of_fraction で重量分率を選択したときに有効。Sample が一つの場合（ブレンドしない場合：monomodal）は、1.0 を入力する。次の Sample[].Num_of_Chain の値は総本数 (MWDist.TotalChain) に応じて自動設定される。

  複数の Samples をブレンドする場合は重量分率を 0.0 - 1.0 の範囲で入力する。その際、ブレンドした Sample[] の WeightFraction の和が 1.0 にならない場合、和に応じて、再度重量分率を計算し直す。
- Num_of_Chain ... Type_of_fraction で分子数を選択したときに、有効。PASTA で入力するべき分子の本数が設定される。ブレンドの場合、その割合を Chain 数で設定することが出来る。
- NumDiv ... 分割数 n。分子量成分は、$n+1$ 成分生成する。
- DivType ... 分割のタイプ。

分割タイプ	DivType
等分割	'Linear'
Log 分割	'Log'
重量分率分割	'WFraction'

- Mmin ... 設定最小分子量。。−1 を入力すると自動的に決定する。
- Mmax ... 設定最大分子量。−1 を入力すると自動的に決定する。

Simulation

Simulation は次の構造からなる。

```
Simulation---Deformations[]---FlowType
                            |-DeformationType
                            |-dt
                            |-Strain
                            |-StrainRate
                            |-MaxTimeStep
                            |-IntervalStep
```

Deformations[]

PASTA の変形条件入力部を設定する。

- `FlowType` ... 'flow', 'noflow', or 'step'。
- `DeformationType` ... 'shear', 'uniaxial', 'biaxial' or 'planar'。
- `dt` ... τ_e 単位当たりのタイムステップ。
- `Strain` ... 歪み量 (only for `FlowType`=='step')。
- `StrainRate` ... 歪み速度 (only for `FlowType`=='flow')。

  `PolymerData.tau_e` に -1 を設定すると `StrainRate` は $1/\tau_e$ 単位でセットされる。

  `PolymerData.tau_e` に秒単位で具体的な数値を入力した場合、PASTA の入力条件の一つである歪み速度の入力値の単位が (1/sec.) になるので注意する必要がある。
- `MaxTimeStep` ... 最大 Iteration 回数。
- `IntervalStep` ... 設定 step 毎に結果を出力する。

出力 UDF

出力 UDF は次の構造からなる。

- `Results`
  分子量分布計算により離散化された分子量成分から計算した平均分子量のデータなどを持つ。ブレンドした系では、各 `Sample[]` ごとおよびブレンドした計算結果を持つ。
- `Sample`
  離散化された Sample の Z、最大伸長比、Chain 数を持つ。
- `Simulation`
  PASTA のシミュレーション条件を持つ。

各データ構造は以下に示される通りである。

Results

`Results` は次の構造からなる。

```
Results---MWDistResults[]---AvMn
        |                  |-AvMw
        |                  |-AvMz
        |                  |-Mw_over_Mn
        |                  |-Mz_over_Mw
        |                  |-InputWFraction
        |-BlendedResults---AvMn
                          |-AvMw
                          |-AvMz
                          |-Mw_over_Mn
                          |-Mz_over_Mw
                          |-InputWFraction
```

MWDistResults[]

各サンプルごとの平均分子量。`ChainType` が 'Arm' の場合、平均分子量は、各々の arm に対するものである。

- `AvMn` . . . 数平均分子量
- `AvMw` . . . 重量平均分子量
- `AvMz` . . . z-平均分子量
- `Mw_over_Mn` . . . 平均分子量分布。`AvMw`/`AvMn`。
- `Mz_over_Mw` . . . 平均分子量分布。`AvMz`/`AvMw`。
- `InputWFraction` . . . 1Sample 当たりの重量分率。

BlendedResults

複数 Sample をブレンドした系の平均分子量のデータなどを持つ。ただし、`ChainType` が 'Arm' の場合は、Arm1 本当たりを 1 分子数として、計算している。モノモーダル計算（1Sample のみで、複数ブレンドしない場合）では、すべての値に 0 が入る。

- `AvMn` . . . 数平均分子量

- AvMw ... 重量平均分子量
- AvMz ... z-平均分子量
- Mw_over_Mn ... 平均分子量分布。AvMw/AvMn。
- Mz_over_Mw ... 平均分子量分布。AvMz/AvMw。
- InputWFraction ... ブレンド系の総重量分率

Sample

Sample は次の構造からなる。

```
Sample---Linear[]---Z
       |           |-MaxStretchRatio
       |           |-NumChains
       |-Arm[]------Z
                   |-MaxStretchRatio
                   |-NumChains
```

Linear[],Arm[]

Linear[] および Arm[] 毎に、離散化された各 i 成分の Z、λ max、Chain 数を持つ。

- Z ... 平均 slip link 数 $Z = M/M_e$
- MaxStretchRatio ... 最大伸長比
- NumChains ... 成分ごとの鎖の数

Simulation

Simulation は次の構造からなる。

```
Simulation---Deformations[]---FlowType
                             |-DeformationType
                             |-dt
                             |-Strain
                             |-StrainRate
                             |-MaxTimeStep
                             |-IntervalStep
```

Deformations[]

Deformations[] は、入力 UDF と同じであるので省略する。

高分子物性データベース UDF

高分子物性データベース UDF は次の構造からなる。

```
PolymerDatabase---GeneralPolymers[]---PolymerName
                |                   |-Properties[]
                |
                |-EngineeringPlastics[]---PolymerName
                |                        |-Properties[]
                |
                |-Rubbers[]---PolymerName
                            |-Properties[]


-------Properties[]---M0
                    |-Me
                    |-GN0
                    |-Density
                    |-tau_e
                    |-Cinfinity
                    |-Temp
                    |-SolubilityParam
                    |-MaxStretchRatio
                    |-Note1
                    |-Note2
                   ------------- 以降は、高分子物性データベース UDF では、無関係。
                    |-Flag
                    |-MolarVolume
                    |-tmp1
                    |-tmp2
                    |-tmp3
```

ここでは、高分子材料を性質によって以下の 3 つに分類する。

- GeneralPolymers[]
  汎用ポリマー。
- EngineeringPlastics[]
  エンジニアリングプラスチック。100°C 以上あるいは 0°C 以下でも寸法安定性や機械的強度を保つ熱可塑性樹脂。ここでは、以下の高分子材料を対象にしている。アセタール、ポリアミド（ナイロン)、ポリイミド、ポリエーテルイミド、ポリエステル、ポリカーボネート、ポリエーテル、ポリスルホンなど。
- Rubbers[]
  エラストマー。非晶性でガラス転移点 Tg が常温より低い、あるいは結晶性で融点が常温より低く、ゴム状弾性を示す。ポリブタジエン、ポリイソプレンなど。

GeneralPolymers[],EngineeringPlastics[],Rubbers[]
各カテゴリーは同じ構造を持つ。

- PolymerName . . . 高分子名

Properties[]

各々の項分子物性データ（単位付き）。Note には、引用文献などが書かれる。データが不明の場合、$-1.0e^{-99}$ が入っているので注意。

- `M0` ... モノマー単位分子量 (g/mol)。
- `Me` ... 絡み合い点間分子量 (g/mol)。
- `GN0` ... プラトーモジュラス (MPa)。
- `Density` ... 密度 (g/cm^3)。
- `tau_e` ... 分子量 $M = M_e$ の分子のラウス緩和時間 (second)
- `Cinfinity` ... 特性比
- `Temp` ... 基準温度 (K)。
- `SolubilityParam` ... 溶解度パラメータ (J/cm^3)。
- `MaxStretchRatio` ... 最大伸長比
- `Note1,Note2` ... 引用文献などを記載。

以降は高分子物性データベースでは無関係である。

- `Flag` ... χ パラメタ計算用 Flag(平均場法シミュレータ SUSHI 用）
- `MolarVolume` ... モル体積 (cm^3/mol)。
- `tmp1, tmp2, tmp3` ... 追加項目用定義名

第7章　解析ツール

ここでは PASTA で使用する Python スクリプトの解説およびレオロジーデータ解析プログラムの使用方法について説明する。

7.1　Action コマンド

PASTA で使用することのできる Action コマンド[1]は以下の通りである。

Action command	File Type	Location
simpleFORK	PASTA 用 入力 UDF (FORK の 出力 UDF)	`Sample`
set_Restart_file	PASTA 用 入力 UDF	`Restart`
set_GPC_file	FORK 用 入力 UDF	`MWDist.Sample[].GPC_INPUT`
set_PolymerData	FORK 用 入力 UDF	`PolymerData`
plot_Stress	PASTA の 出力 UDF	出力 `UDF` 名
plot_Viscosity_or _RelaxationModulus	PASTA の 出力 UDF	出力 `UDF` 名

以下のコマンドが次のようなファイルに定義されている：

Action command	Action file
simpleFORK	`PF_ENGINE/action/simplefork.act`
set_Restart_file	`PF_ENGINE/action/pastainput.act`
set_GPC_file	`PF_ENGINE/action/pastainput.act`
set_PolymerData	`PF_ENGINE/action/pastainput.act`
plot_Stress	`PF_ENGINE/action/pastaplot.act`
plot_Viscosity_or _RelaxationModulus	`PF_ENGINE/action/pastaplot.act`

以下に各々の Action コマンドについて説明する。

1. simpleFORK

   simpleFORK を用いて簡単に PASTA 用入力 UDF の中の分子量分布を再生成することができる。

   (a) PASTA 用 入力 UDF の読み込み
   PASTA 用 入力 UDF を GOURMET から開く。

[1] Action に関しては GOURMET 操作マニュアル参照。

(b) simpleFORK の実行

Tree panel の `Sample` を右クリックし、ポップアップメニューの中から simpleFORK を選択すると、simpleFORK window が開く (図 7.1)。そこで必要なデータを Values 欄に入力する。各々のフィールドの意味は FORK 操作ガイド (6.4) 参照。分子量 M ではなくて、slip-links の個数 $Z = M/M_e$ を入力する点に注意すること。OK ボタンを押すと、`Sample` が再生成される。

(c) `Sample` のデータを確認し、このファイルをセーブする。

図 7.1: simpleFORK window

2. set_Restart_file

set_Restart_file を用いて、リスタートファイル名を 'マウス' 操作で簡単に設定することができる。

(a) PASTA 用 出力 UDF の読み込み

PASTA 用 出力 UDF を GOURMET から開く。

(b) set_Restart_file の実行

Tree panel の `Restart` を右クリックし、ポップアップメニューの中から set_Restart_file を選択すると、set_Restart_file window が開く (図 7.2)。そこで空欄になっている Values 欄を右クリックし、ポップアップメニューから実行したいリスタートファイル名を選択する。

3. set_GPC_file

set_GPC_file を用いて FORK 用入力 UDF の中の GPC データファイル名を 'マウス' 操作で簡単に設定することができる。使用方法は set_Restart_file と同様。

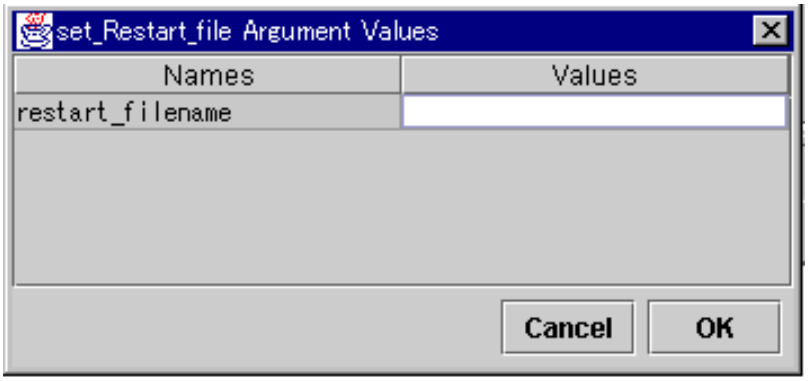

図 7.2: set_Restart_file window

4. set_PolymerData

set_PolymerData を用いて FORK 用入力 UDF の中の PolymerData を簡単に設定することができる。

(a) 入力 UDF と “polymerdata.udf” の読み込み
FORK 用 入力 UDF と “polymerdata.udf” を GOURMET から開く。“polymerdata.udf” は通常 PF_ENGINE/POLYMERDATABASE にある。

(b) set_PolymerData の実行
Tree panel の PolymerData を右クリックし、ポップアップメニューの中から set_PolymerData を選択すると、set_PolymerData window が開く (図 7.3)。“polymerdata.udf” の中から計算の対象とする POLYNAME と PROPERTIES_NUMBER を選び、set_PolymerData window に入力する。OK ボタンを押すと、入力 UDF の PolymerData にデータが生成される。

(c) PolymerData のデータを確認し、このファイルをセーブする。

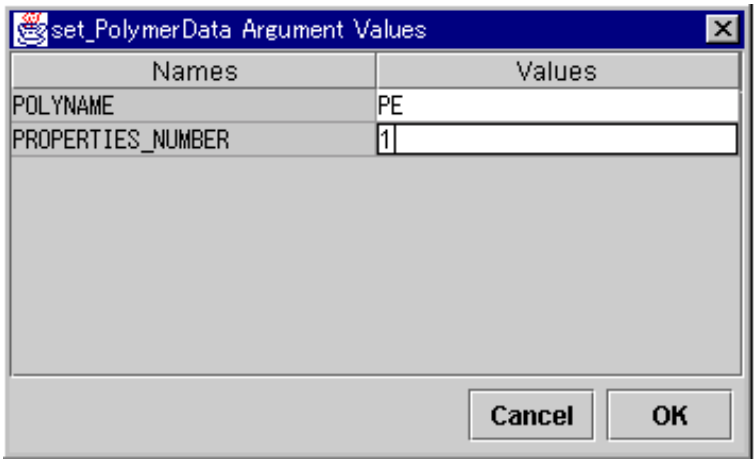

図 7.3: set_PolymerData window

5. plot_Stress 及び plot_Viscosity_or_RelaxationModulus

   plot_Stress を用いると、gnuplot により、応力の時間変化をプロットすることができる。また、plot_Viscosity_or_RelaxationModulus では、同様に粘度や緩和弾性率の時間変化をプロットすることができる。使用方法は以下の通り：

   (a) 出力 UDF の読み込み
   PASTA の 出力 UDF を GOURMET から開く。

   (b) plot_Stress (又は plot_Viscosity_or_RelaxationModulus) の実行
   Tree panel の 出力 UDF 名を右クリックし、ポップアップメニューの中から plot_Stress (又は plot_Viscosity_or_RelaxationModulus) を選択すると、gnuplot graph window が開き、適切なプロットが表示される (例：図 7.4)。

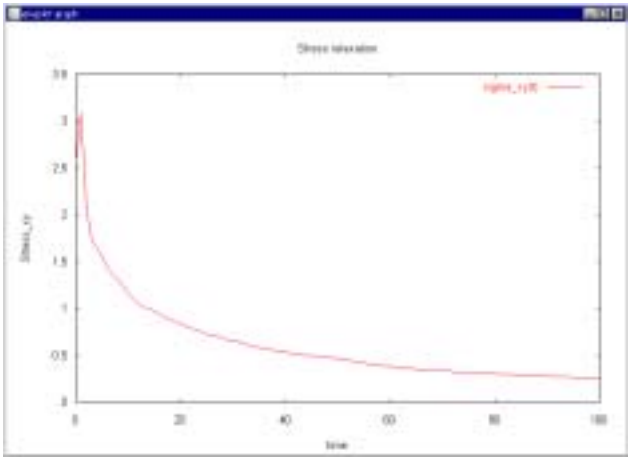

図 7.4: plot_Stress コマンドにより生成した Gnuplot graph window

7.2 Python scripts

以下の Python スクリプトが用意されている。

pasta_python.py	GraphSheat 用。テキストファイル”plot.dat”の出力
pasta_plot_viscosity.py	粘度時間発展のプロット用
pasta_stress.py	応力データのテキストファイル出力用
pasta_for_gt2gw.py	解析プログラム (gt2et、gt2gw) 用の入力データ作成

各 Python スクリプトは、以下のディレクトリに置かれている。

```
PF_ENGINE/PASTA/PASTA-2.8/python
```

以下に各々の Pyton スクリプトについて説明する。

1. “pasta_python.py”

このスクリプトは GOURMET の Plot panel から起動する。gnuplot を用いて、ずり応力、第一法線応力差、第二法線応力差などをプロットすることができる。

```
# Following locationList and tagList are examples for PASTA
# to plot Shear Stress, N1 and N2 as a function of time.
locationList = [
'StepInfo.Time',
'StepData.TotalStress.Shear',
'StepData.TotalStress.N1',
'StepData.TotalStress.N2'
]
tagList = [
'Time',
'Shear Stress',
'N1',
'N2'
]
```

解説：
locationList は GraphSheet に加えたい 出力 UDF の変数のリストである。tagList には各変数に対応した GraphSheet の列の名前を書く。

使用方法：

(a) 出力 UDF の読み込み
PASTA の 出力 UDF を GOURMET から開く。

(b) Python スクリプトのロードと実行
Plot panel に “pasta_python.py” をロードし、Run ボタンを押すと gnuplot 用のデータが GraphSheet に生成される。

(c) GraphSheet の確認
Tree panel の GraphSheet[] をクリックし、texttttGraphSheet[] を表示させ内容を確認する。

(d) Make の実行
Plot panel の Make ボタンを押すと、gnuplot 用のスクリプトが Plot panel に生成される。この時、GraphSheet と同じ内容のテキストファイル “plot.dat” が GOURMET 起動ディレクトリ上に生成される。

以下に、生成した gnuplot 用のスクリプトの一例を示す：

```
# plot command template for platform
# datafile name is fixed as "plot.dat" in the current version
```

```
set title "GraphSheet[]"
plot "plot.dat" using 1:2 title 'Time' with lines , \
     "plot.dat" using 1:3 title 'Shear Stress' with lines , \
     "plot.dat" using 1:4 title 'N1' with lines , \
     "plot.dat" using 1:5 title 'N2' with lines
```

(e) プロット
例えば ‘Time’ と ‘Shear stress’ の関係をプロットするならば、生成したスクリプトを以下のように修正し、Plot ボタンを押す。ここで、#はコメントアウトである。

```
# plot command template for platform
# datafile "plot.dat" is fixed current version
set title "GraphSheet[]"
plot "plot.dat" using 2:3 title 'Shear Stress vs. Time' with lines
#    "plot.dat" using 1:3 title 'Shear Stress' with lines , \
#    "plot.dat" using 1:4 title 'N1' with lines , \
#    "plot.dat" using 1:5 title 'N2' with lines
```

“plot.dat” を読み込むことで、gnuplot 以外のプロット用ソフトウェアを使用することができる。

“plot.dat” の例：
最初の列は自動的に付加される。

```
0   0.0     0.00237363   0.013161    -0.0120411
1   1.0     0.00331907   0.0125386   -0.0128968
2   2.0     0.00394328   0.0105911   -0.00743197
3   3.0     0.00478283   0.0098793   -0.00746133
4   4.0     0.00251843   0.0125421   -0.0110899
5   5.0     0.00236831   0.0152175   -0.0101838
...
```

プロットの例：

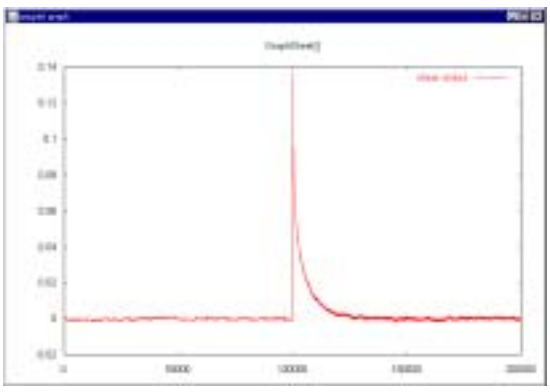

図 7.5: 応力緩和計算結果のプロット例

2. “pasta_plot_viscosity.py”

DeformationType に応じた様々な粘度をプロットするためのスクリプト。

```
# Plot Viscosity for shear, uniaxial, and biaxial deformation,
# and two Viscosities for planar deformation.
FlowType = $Simulation.Deformations[].FlowType
Strain = $Simulation.Deformations[].Strain
StrainRate = $Simulation.Deformations[].StrainRate
Deformation = $Simulation.Deformations[].DeformationType
nn = len(FlowType)
#
locationList = [
'StepInfo.Time',
'StepData.TotalStress.Shear',
'StepData.TotalStress.N1',
'StepData.TotalStress.N2'
]
shearList = [
'Time',
'Shear_Viscosity',
'N1/shear_rate^2',
'N2/shear_rate^2'
]
uniaxialList = [
'Time',
'Uniaxial_Viscosity',
'non-data1',
'non-data2',
]
biaxialList = [
'Time',
'Biaxial_Viscosity',
'non-data1',
'non-data2',
]
planarList = [
'Time',
'Planar_Viscosity_p1',
'Planar_Viscosity_p2',
'non-data',
]
```

使用方法：
“pasta_python.py” と同様。

プロットの例：

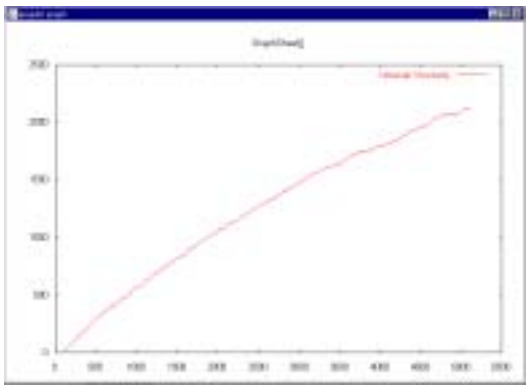

図 7.6: 一軸伸長粘度計算結果のプロット例

3. “pasta_stress.py”

このスクリプトは GOURMET の Python panel から起動する。PASTA で計算した応力成分を時間の関数としてテキストファイルに出力する。

```
#for cygwin
filename = "E:\\somewhere\\stress.txt"
#for unix or linux
#filename = "/somewhere/stress.txt"
t = $StepInfo.Time
if t == None:
print "no data in this record"
else:
fout = open(filename,'a')
totalshear = $StepData.TotalStress.Shear
totalN1 = $StepData.TotalStress.N1
totalN2 = $StepData.TotalStress.N2
outstr = "%f\t%f\t%f\t%f\n" %(t,totalshear,totalN1,totalN2)
print outstr
fout.write(outstr)
```

解説：
filename には各々の OS の記述に従って、出力ファイル名を入力する。スクリプトを修正すれば、応力以外の出力も可能である。

使用方法：

(a) 出力 UDF の読み込み
PASTA の 出力 UDF を GOURMET から開く。

(b) Python スクリプトのロード
Python panel に “pasta_stress.py” をロードし (Plot panel ではない点に注意)、‘filename’ を適切なファイル名に変更する。

(c) Python スクリプトの実行
Run ボタンを押す。

(d) 出力ファイルの確認
生成された出力ファイルの内容を、適当なエディタを使用して確認する。

出力ファイルの例：

```
0.000000    -0.007746   -0.026151   0.017489
10.000000   -0.006312   -0.012012   0.009465
20.000000   -0.003527   -0.017801   0.018997
30.000000   -0.001595   -0.005478   0.013647
40.000000   -0.002701   -0.017810   0.019853
50.000000    0.006846   -0.028618   0.024955
60.000000    0.000780   -0.011298   0.009039
```

4. “pasta_for_gt2gw.py”

PASTA の 出力 UDF からレオロジーデータ解析プログラム (gt2et 及び gt2gw) の入力ファイルを生成する。gt2et 及び gt2gw については、次節で説明する。このスクリプトは GOURMET の Python panel から起動する。

```
#for cygwin
filename = "k:\\somewhere\\gt.txt"
#for unix or linux
#filename = "/somewhere/gt.txt"
t = $StepInfo.Time
if t == None:
print "no data in this record"
else:
fout = open(filename,'a')
totalshear = $StepData.TotalStress.Shear
totalN1 = $StepData.TotalStress.N1
totalN2 = $StepData.TotalStress.N2
gamma = $StepInfo.Gamma
if gamma != 0:
outstr = "%f\n" %(totalshear/gamma)
fout.write(outstr)
else:
print "gamma == 0 "
```

使用方法：

(a) 出力 UDF の読み込み
PASTA の 出力 UDF を GOURMET から開く。

(b) Python スクリプトのロード
Python panel に “pasta_for_gt2gw.py” をロードし、‘filename’ を適切なファイル名に変更する。

(c) Python スクリプトの実行
Run ボタンを押す。

(d) 出力ファイルの確認
生成された出力ファイルの内容を、適当なエディタを使用して確認する。

出力ファイルの例：

```
0.287674
0.263989
0.253326
0.235426
0.226458
0.214798
0.203959
0.196313
0.190866
0.195164
0.183709
```

7.3　レオロジーデータ解析プログラム

データ解析用に 3 つのプログラムが用意されている。ただし、いずれのプログラムも入力ファイルにはテキストファイルを用いる必要がある (UDF ファイルではない点に注意)。Python スクリプト “pasta_for_gt2gw.py” を用いて、事前に 出力 UDF をデータ解析プログラム用に変換しておく必要がある。

各レオロジーデータ解析プログラムは、以下のディレクトリに置かれている。

```
PF_ENGINE/PASTA/tools
```

以下の 3 つのプログラムが用意されている。

♢ gt2et ： 線形緩和弾性率 $G(t)$ から linear stress growth function $\eta^+(t)$ を計算する。

♢ gt2gw ： 線形緩和弾性率 $G(t)$ から動粘度 $\eta^*(\omega)$ あるいは複素弾性率 $G^*(\omega)$ を計算する。

♢ smooth ： $G(t)$, $\eta^+(t)$, $G^*(\omega)$ 等のスムージング (対数プロット用) をおこなう。

gt2et

線形緩和弾性率 $G(t)$ から linear stress growth function $\eta^+(t)$ を計算する。

$$\eta^+(t) = \int_0^t G(t)dt \tag{7.1}$$

- 使用方法：

```
gt2et [-s dt] < infile > outfile
 -s dt   : 時間刻み幅 dt を指定。dt のデフォルトは 1.0。
```

- Input file format：以下のように、1 行に一つづつ $G(t)$ を書く：

```
G(t=0)
G(dt)
G(2dt)
...
```

- Output file format：各行に t と $\eta^+(t)$ がタブ区切りで出力される。

```
0     η+(0)
dt    η+(dt)
2dt   η+(2dt)
...
```

gt2gw

線形緩和弾性率 $G(t)$ から動粘度 $\eta^*(\omega)$ あるいは複素弾性率 $G^*(\omega)$ を計算する。

$$\eta^*(\omega) = \int_0^\infty G(t)e^{i\omega t}dt \tag{7.2}$$

$$G^*(\omega) = i\omega \int_0^\infty G(t)e^{i\omega t}dt \tag{7.3}$$

- 使用方法：

```
gt2gw [-n m] [-s dt] [-e] < infile > outfile
 -n m : FFT に用いるデータ数 N = 2^m。デフォルトは m=16。
 -s dt : 時間刻み幅 dt を指定。デフォルトは dt=1.0。
 -e : η*(ω) の出力を選択。デフォルトは G*(ω)。
```

- Input file format：gt2et と同様。

```
G(t=0)
G(dt)
G(2dt)
...
G(Mdt)
```

注：m は $N = 2^m > M$ が成り立つように選ばなければならない。$N > 4M$ が望ましい。

- Output file format：角速度 ω, $G^*(\omega)$ の実部・虚部を一行に出力する。ω の刻み幅は $d\omega = 2\pi/(Ndt)$.

```
0       G'(0)       G''(0)
dω      G'(dω)      G''(dω)
2dω     G'(2dω)     G''(2dω)
..      ..          ..
N/2dω   G'(N/2dω)   G''(N/2dω)
```

smooth

$G(t)$, $\eta^+(t)$, $G^*(\omega)$ 等のスムージング (対数プロット用) をおこなう。1 次または 2 次の Savitzky-Golay filter を用いている。

- 使用方法：

```
smooth [-n ndiv] [-r r] [-x nmax] [-s skip] [-2] [file]
 -n ndiv   : 1 桁に ndiv 点出力する。デフォルトは 20。
 -r r      : スムージングの際、1 つ前のデータの出力点の r 倍の範囲を用いる。
             デフォルトは 2.0。
 -x nmax   : スムージングの際、現在のデータから最大左右に nmax のデータを用いる。
             デフォルトは 1000。
 -s skip   : 最初の skip 個のデータはスムージングしない。デフォルトは 0。
 -2        : 2 次の多項式でフィットする。デフォルトは 1 次。
 file      : 入力ファイル。デフォルトは標準入力。
```

結果は標準出力に書かれる。

- Input file format：測定時刻 t と測定値 $v(t)$ を時間の順に並べたもの。

t_0 v(t_0)
t_1 v(t_1)
t_2 v(t_2)
.. ..

$t_0 < t_1 < t_2 <$ ... でなければならないが、等間隔である必要はない。

- Output file format：

T_0 $\bar{v}(T_0)$
T_1 $\bar{v}(T_1)$
T_2 $\bar{v}(T_2)$
.. ..

データ出力点 $T_0, T_1, T_2, ...$ は logarithmic scale で 1 桁を ndiv 等分した点。 $\bar{v}$(T) はスムージングされた結果。

例えば、 gt2gw の結果のように 3 列 (ω, G', G'') の出力ファイルにスムージングをかける場合には、スクリプト “gwsmooth” を使用すると便利である：

```
gwsmooth infile > outfile
```

7.4 アニメーションプログラム pastanim

5.2.1 節にあるように、PASTA の入力 UDF で Trajectory を指定すると、特定の分子鎖の運動の軌跡（トラジェクトリ）をファイルに出力することが出来る。出力されたトラジェクトリファイルを読んでアニメーション表示するプログラムが pastanim である。

- 起動方法以下のコマンドをシェルプロンプトに対して入力する：

```
pastaim [-w window_size] trajectory_file
```

ここで、window_size は描画ウインドウのサイズ（ピクセル単位）で、デフォルトは 600。trajectory_file はトラジェクトリファイル名である。

- 使用法

  以下のマウスおよびキーボードコマンドが使用可能：

  マウス操作：

drag	rotate
Shift+drag	zoom in/out
Ctrl+drag	translate

  キーボードコマンド：

Q or q	quit pastanim
Return	start/stop animation
Space	go to next frame
b	go back to the previous frame
h j k l	rotate
i o	zooom in/out
r	go to the first frame
R	go to the first frame, and reset zoom and rotation
e	go to the last frame
G	go to any frame (enter the frame number from the console)
c	reset translation
C	reset translation, zoom, and rotation
W	write current frame into a ppm file

付 録A　コンパイル方法

A.1　PASTA およびFORK のコンパイル方法

PASTA 及び FORK のソースコードのコンパイルは簡単である。
`PASTA/PASTA-2.8/src` あるいは `PASTA/FORK-1.4.1/src` ディレクトリにて、‘make’ とタイプすることでコンパイルを開始する。コンパイルを行なう際、OS およびマシンタイプの判別は `Makefile` の中で行われる。

もしも ‘make’ に失敗した場合、環境変数 `PF_FILES` が正しくセットされているか確認する必要がある。例えば “`pfinterface.h`” のような GOURMET とエンジンとのインターフェース関係のヘッダーファイルは、`$PF_FILES/include/`にあるからである。

‘make’ に成功したら、新しく出来た binary を`$PF_ENGINE/bin/OS-type/`にコピーするとよい。

参考文献

1) M.Doi, and S.F.Edwards, eds.: *The Theory of Polymer Dynamics*, Oxford University Press (1986).

2) J. Takimoto, H. T. and Doi, M.: Predictions of the rheological properties of polymer melts by stchastic simulation, in *Proceeding of XIIIth International Congress on Rheology*, p. 97 (2000).