OCTA

ソフトマテリアルのための統合化シミュレータ

動的平均場法シミュレータ

version 10.5

ユーザーズマニュアル

OCTA ユーザーズグループ

OCT. 22 2017

執筆者

第 1 章　森田裕史, 樹神弘也, 本田隆, 川勝年洋
第 2 章　森田裕史, 樹神弘也, 本田隆, 川勝年洋
第 3 章　本田隆
第 4 章　横溝勝行, 樹神弘也, 本田隆
第 5 章　本田隆
第 6 章　本田隆
第 7 章　川勝年洋
第 8 章　浦下真治
付録 A　本田隆
付録 B　本田隆
付録 C　本田隆
付録 D　樹神弘也

プログラム開発者

SUSHI	本田隆, 樹神弘也, Jiunn-Ren Roan, 森田裕史 川勝年洋
CPC: The Simple Python scripts for χ-parameter guess	川勝年洋
SPCF: The tool for correlation function analysis	浦下真治
InterfaceSimulator	樹神弘也

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」とプロジェクト終了後に OCTA ユーザーズグループにて継続して行われたものである。

SUSHI9 の GPU および MPI 並列化は、東京工業大学の先端研究施設共用促進事業 「『みんなのスパコン』TSUBAME によるペタスケールへの飛翔 」の採択課題「メソ構造を持つ高分子材料のマルチスケール・シミュレーション」として、東京工業大学 学術国際情報センターのスーパーコンピュータ TSUBAME2.0 を用いて行った。ご支援頂きました東京工業大学の青木先生、渡邊先生、東京工業大学学術国際情報センターの佐々木様、松本様には紙面を借りまして厚く御礼申し上げます。

また、SUSHI9 の MPI 並列化には北海道大学情報基盤センターのスーパーコンピュータ SR16000/M1 も利用した。ご支援頂きました北海道大学情報基盤センターの大宮先生、防衛大学応用物理学科の萩田先生、日立製作所の合田様、広瀬様には紙面を借りまして厚く御礼申し上げます。

SUSHI9 のテストに関しては公益社団法人　新化学技術推進協会の高分子技術セミナーの参加メンバーから多くのコメントを頂いた。新化学技術推進協会および高分子技術セミナーの参加メンバーには紙面を借りまして厚く御礼申し上げます。

SUSHI9 の大規模並列化計算については「京」産業利用 (トライアルユースおよび実証利用 ID hp130056) の制度を利用し検証した。ご支援頂きました RIST の皆様には紙面を借りまして厚く御礼申し上げます。

目 次

図 目 次

表 目 次

第1章　SUSHIとは

OCTA プロジェクトでは、空間スケールを大きくミクロスケール、メソスケール、マクロスケールという 3 つの領域にわけ、それぞれの領域での高分子の挙動をシミュレートできるシミュレータ群を開発した。このマニュアルで説明するシミュレータ（以下 SUSHI; Simulation Utilities for Surfaces and Interfaces）は、3 つのシミュレータの中で時間、空間スケールにおいて中間に位置する。（通常、適用範囲としては、10 − 100 nm のオーダーの現象、時間は、$10^{-8} \sim 1$ sec の時間スケール程度と言われている。）このシミュレータがカバーし得る領域は、主に界面を含む系である。よって、SUSHI は界面構造のシミュレーターとして考えられる。界面を含む系を対象とすることから、不均一系が取り扱えることがこのシミュレーターのセールスポイントである。取り扱える系の例としてマクロ、ミクロ相分離構造やさらに小さな相分離構造であるミセルなどが挙げられる。また、これらに関連する物性値、例えば界面張力、臨界ミセル濃度などを、SUSHI の結果を解析することで求めることができる。関連の書籍も出版されているのでぜひ参考にして頂きたい [1, 2]。

SUSHI で行うことができるのは、Self Consistent Field (SCF) 理論に基づいた静的および動的計算である [3, 4, 5, 6, 7, 8, 9, 10, 11]。ここでは、動的計算を特に動的平均場法 (Dynamic Density Functional Method) と呼ぶことにする。中間的な時間・空間スケールの現象を扱うための他の手法として、系の自由エネルギーを近似的に定式化した Approximate Density Functional 法があり、現行バージョンでは、動的平均場法に比べれば、機能制限はあるが、実装されている。

これらの手法の特徴を以下に述べる。SCF 理論に基づいた静的および動的計算は、（平均場近似の範囲内で）鎖のコンフォメーションが正しく考慮されるため、それが近似的にしか考慮されない Approximate Density Functional 法に比べて定量性に勝るが、一方で計算コストは高くなる。SCF 理論に基づく計算で平衡状態を求める際には、静的計算を行う方が動的平均場法を用いるよりも計算コストが幾分抑えられる。鎖のコンフォメーションを考慮した定量性のあるダイナミクスを扱う際には、動的平均場法を用いることになる。

SUSHI は、表 1.1、表 1.2 に示すように、極めて汎用性の高い設計になっている。高分子鎖の複雑な分岐構造やシーケンスに対応し、異なる種類のセグメントそれぞれに対して自由に有効結合長・比体積といったパラメータの値を設定できる。境界条件は周期境界、固体壁、反射壁に対応したそれぞれの条件を設定することが可能であり、空間メッシュに関しては正方メッシュ、直方メッシュ、球状メッシュ、円柱状メッシュを問題に応じて使用する。アンサンブルとしては、カノニカルおよびグランド・カノニカル集団を扱うことができる。さらに、鎖を壁に固定したり、ある部分領域に閉じこめるといった設定も可能である。これにより、非常に広範な問題に適用することが可能となり、材料設計における様々な場面で役立つことが期待される。

SUSHI では、このような高い汎用性をもたせるために、理論の様々な拡張を行っており、用いる基礎方程式もやや複雑になる。第 2 章では、簡単なホモポリマーブレンド系での SCF 理論の定式化から初めて、これを拡張していくやり方で、汎用的なシミュレーションを記述するのに必要な基礎理論を展開していく。次に、この理論方程式を解くための具体的な計算法について、第 2.9 節で述べる。

SUSHI の簡易な操作方法を第 3 章で説明し、SUSHI を用いたいくつかの具体的な例題を第 4 章に挙げる。例には、多分散性を考慮した高分子ブレンド系の界面強度、高分子が吸着した表面間に働く力や選択溶媒中のブロック共重合体（界面活性剤）の臨界ミセル濃度などの界面物性の計算などについて示した。

SUSHI の詳細な入力方法を第 5 章で説明する。また、SUSHI の計算結果を GOURMET 上でグラフや画像にして解析する簡易 Python ツールの利用方法を第 6 章で説明する。

SUSHI では、系を特徴づける重要なパラメータとして Flory の χ パラメータを用いるが、その値として現実の高分子に対応したものを簡易的に計算するための支援ツールが Python スクリプトとして用意されている。

これについては、第 7 章で説明する。

解析シミュレータとして、シミュレーションの結果得られた相分離構造などの実相関関数を計算するための実空間相関関数解析シミュレータ（spcf）が用意されている。その実行方法ならびに実行例について、第 8 章で述べる。

付録にはコンパイル方法とシステム拡張方法を簡単に述べる。また、SUSHI の基本機能を利用し、特別な問題に特化して開発された InterfaceSimulator の概要を述べる。

尚、正式なマニュアルは英文マニュアルである。もし、日本語マニュアル内に不明な点がある場合は英文マニュアルを参照して頂きたい。

SUSHI10.0 で修正されたバグ

- カノニカルアンサンブルでかつ壁のある系での高分子の自由エネルギーの計算では高分子のコンフォメーション・エントロピーの効果が考慮されていない。
- 散乱関数の一次元散乱関数の出力では同一の q でのデータ平均がされていない。

SUSHI10.3 で修正されたバグ

- 高分子電解質の計算で SCF が収束しない。
- シリンダー状メッシュを利用した計算の出力が異常。
- シリンダー状メッシュ＋壁（シリンダー末端）の計算ができない。
- action plot_1D で壁のある系の出力が異常。
- SUSHIInput.zoom.sigma_per_b はミスタイプ。SUSHIInput.zoom.b_per_sigma を追加。前バージョンの SUSHIInput.zoom.sigma_per_b は、SUSHIInput.zoom.b_per_sigma として扱われる。
- -Z を伴うズーミングオプションにバグ。ビーズの座標が出力されない。
- ソフト粒子の周期境界条件にバグ。
- 溶媒＋壁の計算にバグ。

SUSHI10.4 で修正されたバグ

- constV と constW を 0 にして実行すると異常な計算結果となる。
- MPI+GPU+流体の計算が異常。

表 1.1: SUSHI の機能一覧表（1/3）

	項目	内容	
1	概要	動的平均場の理論を使用して高分子のメソスケールのシミュレーションを行うプログラムである。	
2	バージョン	2014 年 1 月リリース版	
3	使用プログラミング言語	C++	
4	利用可能コンパイラー	Microsoft VC++ Ver. 2010 以上 g++ library for gcc Ver. 4. 以上	
5	平均場法	経路積分法を用いた自己無撞着場法 識別名 SCF （Self Consistent Field）	Ginzburg-Landau 法と RPA 法を用いた動的平均場法 識別名 GRPA (Ginzburg-Landau theory using Random Phase Approximation)
6	計算可能な高分子構造 分岐構造（トポロジー） シーケンス モノマーの特徴	SCF 任意 ホモ、ブロック、傾斜 有効結合長 (Effective bond length) と特徴体積 (Specific volume) を入力可能	GRPA 任意 ホモ、ブロック 有効結合長 (Effective bond length)
7	計算可能な系 状態 境界条件 空間メッシュ	 ポリマーの溶融状態 ポリマー＋溶剤の液体（気相も含む） 周期境界、反射壁、固体壁 規則格子メッシュ　１～３次元 不規則間隔格子メッシュ (規則格子で格子間隔を調整したもの)　１～３次元 円筒座標メッシュ　２次元 球座標メッシュ　１次元	
8	平均場計算方法	静的平衡計算 (Eq)、 動力学計算（動的な時間発展の計算）(Dy)	
9	アンサンブル	カノニカル（系内の組成一定） グランドカノニカル（バルクの組成一定）(SCF)	
10	外場	固体壁との相互作用が入力可能。(SCF)	
11	グラフト	固体壁面へグラフトした高分子が計算可能。(SCF)	

表中で、Eq は静的平衡計算で、Dy は動力学計算で利用できることを意味する。

表 1.2: SUSHI の機能一覧表（2/3）

	項目	内容
12	経路積分計算の安定化機能	Explicit 差分スキームに加え、計算がより安定な Implicit 差分スキームを追加。(SCF)
13	動力学計算の安定化機能 (Dy)	Explicit 差分スキームに加え、計算がより安定な Implicit 差分スキームを追加。
14	高分子自由末端拘束機能 (ミセル計算機能) (Eq)	高分子の自由末端が存在可能な領域に拘束をかけることによりミセルなどの計算が可能。(SCF)
15	相似経路積分の指定機能 (Eq)	相似なトポロジーをもつ高分子構造間で、共用できる経路積分の指定を行い計算時間の短縮が可能。(SCF)
16	ドメイン領域指定機能 (Eq)	自己無撞着場の初期値設定により、目的とするミクロ相分離構造のシミュレーションが可能。(SCF)
17	慣性半径計算機能	高分子の全体および部分鎖の慣性半径が計算可能。(SCF)
18	易動度入力機能 (Dy)	動的計算において高分子を形成するモノマーや溶媒の易動度の入力が可能。
19	吸着動力学計算機能 (Dy)	高分子溶融体や高分子溶液から固体壁面に高分子が吸着する動的計算が可能。(SCF)
20	化学反応動力学計算機能 (Dy)	1) 反応速度が速く体積分率変化で考えられる反応 2) モノマーや高分子上の活性点の反応 (SCF) 3) 高分子末端が壁面にグラフトする反応 (SCF) 4) 高分子が重合する反応 (SCF)
21	高分子電解質計算機能	強高分子電解質が計算可能。
22	SCF モンテカルロ法	自由端や結合点を一点に固定し、それを動かすモンテカルロ法で孤立した分子の描像が確認できる。
23	ずり流動 (Dy)	ずり流動を印加した動力学が可能。(SCF)
24	圧縮性のある動力学計算 (Dy)	圧縮率を導入することによる動力学が可能。(SCF)
25	熱ゆらぎ (Dy)	熱ゆらぎを印加した動力学が可能。(SCF)
26	システムサイズの最適化	自由エネルギー密度を最小とするようにシステムサイズの最適化が可能。
27	散乱関数の出力	セグメント間の散乱関数の出力が可能。
28	GRPA 法の導入 (Dy)	GRPA 法による動的平均場計算が可能。
29	HYBRID 法の導入 (Dy)	HYBRID 法による動的平均場計算が可能。
30	流体力学の導入 (Dy)	流体力学を取り入れた動的平均場計算が可能。
31	外部電場の導入	外部電場を取り入れた平均場計算が可能。
32	オブスタクルの導入	1) 系内に粒子を存在させ、その内外にポリマーを存在させられる。 2) 系内にファイバーを存在させ、その内外にポリマーを存在させられる。 3) オブスタクル表面に χ_s を設定できる。 4) オブスタクル表面に高分子末端をグラフトできる。
33	ズーミング	COGNAC 用の入力ファイルの作成を支援できる。

表 1.3: SUSHI の機能一覧表（3/3）

	項目	内容
34	並列計算の改良	壁を導入できる（MPI,GPU：但し静的計算のみ)。MPI では各軸の分割数を 1 でも実行可能とした。
35	任意形状オブスタクルの導入	三角形ポリゴン形式の任意形状の外部ファイルを入力できる。
36	ソフト粒子の導入	粒子を形状を固定した溶媒として導入できる。
37	流体力学的効果の MPI+GPU 計算への導入	MPI+GPU 計算に流体力学的効果を導入できる。
37	バグフィックス	MPI+GPU 計算で実行不能な場合の解消

第2章　理論背景

2.1　Self Consistent Field 理論

この節では、直鎖ホモポリマー・ブレンドを例に用いて、Self Consistent Field (SCF) 理論を概説する。より複雑な鎖（ブロック共重合体や枝分かれ鎖、あるいは内部状態を持つ鎖）への SCF 理論の拡張は、次節以降で解説する。

体積が $\mathcal{V}$ の系を考える。境界条件は、周期境界条件や Neumann 境界、Dirichlet 境界条件などそれぞれの系の状態に応じて決まるものとする。系を構成する最小単位はセグメントであり、系には、インデックス $K = 1, 2, \cdots$ で指定される複数の種類のセグメントが存在し、これらのセグメントから構成される複数種の高分子鎖が存在するものとする。高分子鎖の種類をインデックス $p = 1, 2, \cdots$ で指定するものとし、p 種の高分子鎖の重合度（鎖を構成するセグメントの総数）を $N^{(p)}$、系内の p 種の高分子鎖の本数を M_p 本とする。さらに、個々のセグメントを指定するために、この $N^{(p)}$ 個のセグメントを、鎖の一端から順に $i = 0, 1, 2, \cdots, N^{(p)}$ とインデックスをつけることにする。この節で考えるホモポリマー・ブレンドの場合には、セグメント種を表すインデックス K と高分子鎖を表すインデックス p は実質的に同一の意味を持つことになることに注意されたい。以下この節では、インデックス p の代わりに、全てインデックス K を用いることにする。

SCF 理論を用いると、経路積分により高分子鎖のコンフォメーションを考慮に入れた形で平衡状態における高分子鎖（セグメント種）$K(= p)$ の濃度分布 $\phi_K(\mathbf{r})$ を求めることができる。高分子鎖の経路積分表示についてわかりやすく説明するために、鎖を格子定数 b の立方格子上のランダムウォークとしてモデル化する。SCF 法では、鎖のコンフォメーションは確率分布を用いて統計的に取り扱われる。互いに相互作用しあっている多数の鎖からなる系の中での 1 本の鎖の平衡統計を、その鎖を構成するセグメントの種類に依存する平均的な場の下での 1 本の理想鎖の平衡統計として表すという近似（平均場近似）を行なう。後述する理由により、この平均場を自己無撞着場と呼ぶ。自己無撞着場は、セグメント間の相互作用や非圧縮条件などの外部条件によって決まるポテンシャル場であり、次のような形を持つ。

$$V_K(\mathbf{r}) = W_K(\mathbf{r}) + \Big[\text{拘束条件から決まる拘束力}\Big] \tag{2.1}$$

ここで、$W_K(\mathbf{r})$ はセグメント間相互作用に起因する平均場であり、

$$W_K(\mathbf{r}) = \sum_{K'} \chi_{KK'} \phi_{K'}(\mathbf{r}) \tag{2.2}$$

で与えられる。(2.1) 式の第 2 項の拘束力は非圧縮条件のような系に課せられた拘束条件を満足させるための場（Lagrange の未定定数に相当する）である。いくつかの拘束条件に対する拘束力の具体的な形については 2.5 節で述べる。

この自己無撞着場を用いた平均場近似のもとでは、1 本の高分子鎖を互いに重なり合わない幾つかの部分鎖に分割するとき、異なる部分鎖同士は統計的に独立になることが示せる。従って、高分子鎖全体のコンフォメーションの統計は、その鎖を構成する個々の部分鎖の統計を計算することで求めることができる。以下では、この部分鎖のコンフォメーションの統計を経路積分法を用いて計算する手法について述べる。

K 種セグメントで構成された鎖の i 番目のセグメントと j 番目のセグメントを結ぶ部分鎖を考え、平衡状態においてこの部分鎖の両端の i 番目のセグメントと j 番目のセグメントがそれぞれ位置 $\mathbf{r}_i$ と $\mathbf{r}_j$ にある統計重率を $Q_K(i, \mathbf{r}_i; j, \mathbf{r}_j)$(経路積分) と表すことにする。空間の位置 $\mathbf{r}$ における K 種セグメントの感じる自己無撞着

場 (平均場) ポテンシャルを $V_K(\mathbf{r})$ とするとき、鎖の i 番目のセグメントから j 番目のセグメントまでのセグメント位置が $\mathbf{r}_i, \mathbf{r}_{i+1}, \cdots, \mathbf{r}_{j-1}, \mathbf{r}_j$ となるような部分鎖の統計重率は定数因子を除いて

$$\exp\Big[-\beta\Big\{\frac{1}{2}V_K(\mathbf{r}_i) + V_K(\mathbf{r}_{i+1}) + \cdots + V_K(\mathbf{r}_{j-1}) + \frac{1}{2}V_K(\mathbf{r}_j)\Big\}\Big] \tag{2.3}$$

と表される。但し、$\beta = 1/k_{\mathrm{B}}T$ であり、両端のセグメントの重みは、内部のセグメントの重みの半分だけであるものとして計算した。(2.3) 式を用いれば、i 番目のセグメント位置が $\mathbf{r}_i$ でかつ j 番目のセグメントが位置 $\mathbf{r}_j$ にある統計重率 $Q_K(i, \mathbf{r}_i; j, \mathbf{r}_j)$ は、両端が固定された条件のもとでの可能な配置全てについて和をとることで求められ、

$$Q_K(i, \mathbf{r}_i; j, \mathbf{r}_j) = \frac{1}{z^{|i-j|}} \sum_{\text{全ての配置}} \exp\Big[-\beta\Big\{\frac{1}{2}V_K(\mathbf{r}_i) + \sum_{k=i+1}^{j-1} V_K(\mathbf{r}_k) + \frac{1}{2}V_K(\mathbf{r}_j)\Big\}\Big] \tag{2.4}$$

と表される。ただし、z は最近接格子点の数である。関係式 (2.4) を用いれば、以下の漸化式を容易に導出することができる。

$$Q_K(i, \mathbf{r}_i; j+1, \mathbf{r}) = \sum_{\mathbf{r}'} Q_K(i, \mathbf{r}_i; j, \mathbf{r}') \times \frac{1}{z} \exp\Big[-\frac{1}{2}\beta\big\{V_K(\mathbf{r}') + V_K(\mathbf{r})\big\}\Big] \tag{2.5}$$

ただし、$\sum_{\mathbf{r}'}$ は $\mathbf{r}$ の最近接格子点 $\mathbf{r}'$ についての和を表す。ここで「Q_K は、格子サイズ程度の距離のスケールで緩やかに変化する」という仮定を置くと、$Q_K(i, \mathbf{r}_i; j+1, \mathbf{r})$ を $\mathbf{r} = \mathbf{r}'$ の周りで展開することで、

$$\begin{aligned} Q_K(i, \mathbf{r}_i; j+1, \mathbf{r}) \quad = \quad & \frac{1}{z} e^{-\beta V_K(\mathbf{r})/2} \sum_{\mathbf{r}'} \Big[e^{-\beta V_K(\mathbf{r})/2} Q_K(i, \mathbf{r}_i; j, \mathbf{r}) \\ & + \nabla\Big\{e^{-\beta V_K(\mathbf{r})/2} Q_K(i, \mathbf{r}_i; j, \mathbf{r})\Big\} \cdot (\mathbf{r} - \mathbf{r}') \\ & + \frac{1}{2}\nabla\nabla\Big\{e^{-\beta V_K(\mathbf{r})/2} Q_K(i, \mathbf{r}_i; j, \mathbf{r})\Big\} : (\mathbf{r} - \mathbf{r}')(\mathbf{r} - \mathbf{r}') + \cdots \Big] \end{aligned} \tag{2.6}$$

となる。さらにここで、「$\beta V_K(\mathbf{r})$ は微小である」と仮定すると、(2.6) 式は、以下のシュレーディンガー型の発展方程式に従うことが示せる。

$$\frac{\partial}{\partial i} Q_K(i', \mathbf{r}'; i, \mathbf{r}) = \Big[\frac{b^2}{6}\nabla^2 - \beta V_K(\mathbf{r})\Big] Q_K(i', \mathbf{r}'; i, \mathbf{r}) \tag{2.7}$$

ここで、格子定数 b は鎖の 1 個 1 個の結合の長さに相当するため、有効結合長 (effective bond length) と呼ばれる（有効結合長 b は一般にはセグメントの種類 K に依存する）。この式は i を時間と見た時に拡散方程式と同形のものであることから、自己無撞着場の作る外場中の粒子の拡散運動と対応させることができる。偏微分方程式 (2.7) を解くためには初期条件が必要になるが、これは経路積分の定義により、

$$Q_K(i', \mathbf{r}; i', \mathbf{r}') = \delta(\mathbf{r} - \mathbf{r}') \tag{2.8}$$

となることがわかる。

(2.7)、(2.8) 式によって求められた経路積分を用いると、位置 $\mathbf{r}$ における K 種セグメントの濃度分布は、

$$\phi_K(\mathbf{r}) = C_K \sum_i \int d\mathbf{r}_0 \int d\mathbf{r}_{N_K} Q_K(0, \mathbf{r}_0; i, \mathbf{r}) Q_K(i, \mathbf{r}; N_K, \mathbf{r}_{N_K}) \tag{2.9}$$

と表される。ただし、C_K は規格化定数、N_K はこの高分子鎖に含まれるセグメントの総数、$\mathbf{r}_0$ と $\mathbf{r}_{N_K}$ は部分鎖の両端にある 0 番目及び N_K 番目のセグメントの位置であり、和 $\sum_i$ は、この鎖に含まれる N_K 個の全てのセグメントについて足すことを表す。

(2.9) 式の規格化定数 C_K は、この鎖の統計的扱いがカノニカル集団（系内の鎖の総数が一定の場合）であるか、あるいはグランド・カノニカル集団（系が鎖の濃度一定の粒子源と接触平衡にある場合）であるかによって、以下のように異なる。

1) カノニカル集団の場合

$$C_K = \frac{M_K}{\int d\mathbf{r}_0 \int d\mathbf{r}_{N_K} Q_K(0,\mathbf{r}_0;N_K,\mathbf{r}_{N_K})} \tag{2.10}$$

ただし、M_K は系全体に含まれるこの鎖の総数である。

2) グランド・カノニカル集団の場合
着目する系が外部の均一系（バルク）と化学平衡にあると考える。バルクにおける高分子鎖 K の濃度を $\phi_K^{(\mathrm{bulk})}$ とかくことにする。$\phi_K(\mathbf{r}) \equiv \phi_K^{(\mathrm{bulk})}$ のときに式 (2.2) から計算されるポテンシャル $W_K(\mathbf{r}) \equiv W_K^{(\mathrm{bulk})}$ を用いて C_K は

$$C_K = \frac{\phi_K^{(\mathrm{bulk})}}{N_K} \exp\Big[N_K(W_K^{(\mathrm{bulk})} + \mathrm{constant})\Big] \tag{2.11}$$

と書くことができる。右辺の constant は、(2.1) 式の自己無撞着場の定数の不定性に起因するもので、今後この constant を 0 におく。これは、グランド・カノニカル集団を扱う場合にバルクにおける拘束力の値を 0 におくことに対応する。

実際に計算を行なうにあたっては、簡単化のために、(2.7) 式中に表れた $Q_K(0,\mathbf{r}_0;i,\mathbf{r})$ 及び $Q_K(i,\mathbf{r};N_K,\mathbf{r}_{N_K})$(経路積分) を初期座標に関して積分された以下の形式の $Q_K(i,\mathbf{r}_i)$ 及び $\widetilde{Q}_K(i,\mathbf{r}_i)$ を用いると便利である。

$$Q_K(i,\mathbf{r}_i) = \int d\mathbf{r}_0 Q_K(0,\mathbf{r}_0;i,\mathbf{r}_i) \tag{2.12}$$

$$\widetilde{Q}_K(N_K-i,\mathbf{r}_i) = \int d\mathbf{r}_{N_K} Q_K(N_K,\mathbf{r}_{N_K};i,\mathbf{r}_i) \tag{2.13}$$

経路積分 $Q_K(i,\mathbf{r}_i)$ 及び $\widetilde{Q}_K(i,\mathbf{r}_i)$ はそれぞれ、次の発展方程式を初期条件

$$Q_K(0,\mathbf{r}) = \widetilde{Q}_K(0,\mathbf{r}) = 1 \tag{2.14}$$

の下で解くことによって求められる。

$$\frac{\partial}{\partial i} Q_K(i,\mathbf{r}) = \Big[\frac{b^2}{6}\nabla^2 - \beta V_K(\mathbf{r})\Big] Q_K(i,\mathbf{r}) \tag{2.15}$$

$$\frac{\partial}{\partial i} \widetilde{Q}_K(i,\mathbf{r}) = \Big[\frac{b^2}{6}\nabla^2 - \beta V_K(\mathbf{r})\Big] \widetilde{Q}_K(i,\mathbf{r}) \tag{2.16}$$

(2.15) (2.16) 式によって求められた経路積分を用いて、位置 $\mathbf{r}$ におけるセグメント濃度は、(2.9) (2.12) (2.13) 式より

$$\phi_K(\mathbf{r}) = C_K \sum_i Q_K(i,\mathbf{r}) \widetilde{Q}_K(N_K-i,\mathbf{r}) \tag{2.17}$$

と表される。

関係式 (2.1) からわかるように自己無撞着場 $V_K(\mathbf{r})$ はセグメント濃度場 $\phi_K(\mathbf{r})$ に依存し、この $\phi_K(\mathbf{r})$ は (2.17) により経路積分 $Q_K(i,\mathbf{r})$ から求められる。ところが経路積分 $Q_K(i,\mathbf{r})$ は自己無撞着場 $V_K(\mathbf{r})$ を含む方程式 (2.15) (2.16) を解くことによって求められる。従って、$V_K(\mathbf{r})$、$\phi_K(\mathbf{r})$ 、$Q_K(i,\mathbf{r})$ は、図 2.1 に示すように、拘束条件の下で互いに相手を再帰的に定義する関係にある。従って、この図式は互いに矛盾することがないように解かれる必要があり、これが自己無撞着場の名前の由来である。

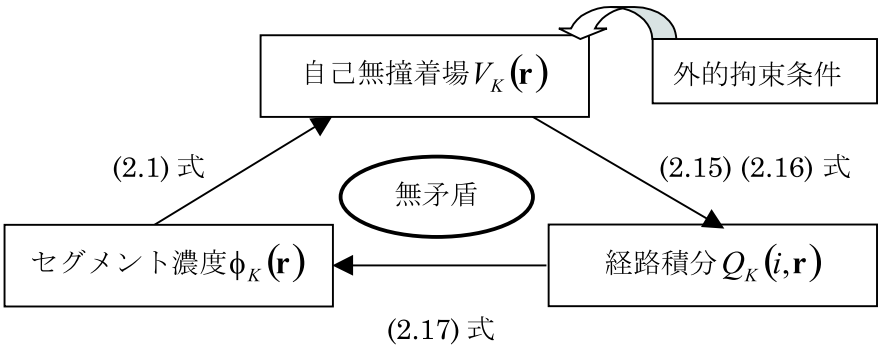


図 2.1: 自己無撞着場 (SCF) 理論の構造

2.2 枝分かれ鎖、ブロック共重合体への拡張

分岐のある鎖やブロック共重合体などを SCF で扱う場合には、前節で述べたホモポリマーブレンドの扱いを拡張する必要がある。図 2.2 に示すように、各高分子鎖は、部分鎖 (subchain) と呼ばれる単一の構造を持った直鎖状の鎖が連結された構造を持っていると考え、それぞれの部分鎖をインデックス r を用いて指定するものとする。また、部分鎖同士をつなぐ点および鎖の端点を結合点 (junction) と呼ぶことにする。さて、p 種の鎖の r 部分鎖は、$N_r^{(p)}$ 個のセグメントから構成されているとする。この $N_r^{(p)}$ 個のセグメントを、部分鎖の一端から順にインデックス i を用いて指定することにする。

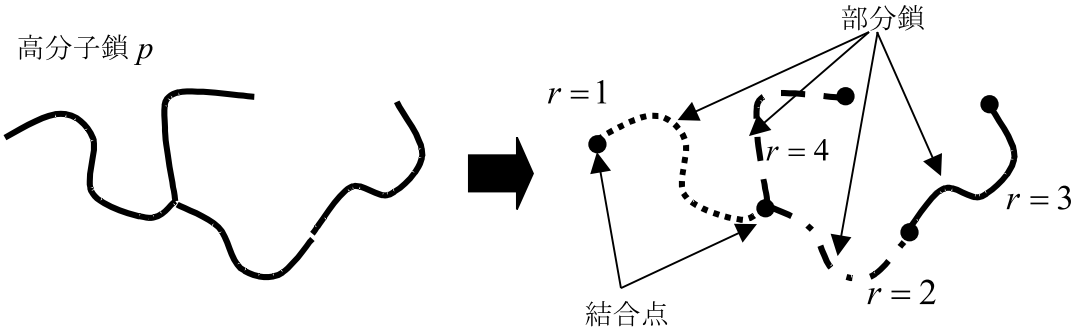


図 2.2: 1 本の高分子鎖を複数の部分鎖に分割

前節で導入した平均場近似のもとでは、1 本の高分子鎖を幾つかの部分鎖に分割するとき（図 2.2)、異なる部分鎖同士は統計的に独立になることが示せる。従って、高分子鎖全体のコンフォメーションの統計は、その鎖を構成する個々の部分鎖の統計を計算することで求めることができる。部分鎖の統計は、前節のホモポリマーの経路積分の方法をわずかに変更することで計算可能である。以下では、この部分鎖のコンフォメーションの経路積分の計算手法について述べる。

2.1 節のホモポリマーの場合には、鎖の両端は自由端であるのに対して、枝分かれ鎖やブロック共重合体の部分鎖の場合には、末端セグメントには、一般に部分鎖の集合が連結されている。従って、部分鎖のコンフォメーションの統計重率を計算する場合には、部分鎖の末端に連結された鎖の統計重率を考慮に入れる必要がある。この場合には、位置 r におけるセグメント濃度への p 種の高分子鎖の r 番目の部分鎖からの寄与は、

$$\phi_r^{(p)}(\mathbf{r}) = C_r^{(p)} \sum_i \int d\mathbf{r}_0 \int d\mathbf{r}_N q_0(\mathbf{r}_0) Q_K(0,\mathbf{r}_0;i,\mathbf{r}) Q_K(i,\mathbf{r};N,\mathbf{r}_N) q_N(\mathbf{r}_N) \tag{2.18}$$

と表されることになる。ただし、$Q_K(i,\mathbf{r}_i;j,\mathbf{r}_j)$ は (2.7) (2.8) で計算されるものと同一の経路積分であり、K はこの部分鎖を構成するセグメント種（すなわち $K_r^{(p)}$）である。(2.18) 式で、$C_r^{(p)}$ は規格化定数、N はこの部分鎖に含まれるセグメントの総数（すなわち $N_r^{(p)}$）、$\mathbf{r}_0$ と $\mathbf{r}_N$ は部分鎖の両端にある 0 番目及び N 番目のセグメントの位置、そして $q_0(\mathbf{r}_0)$ と $q_N(\mathbf{r}_N)$ は、両端のセグメントを介して連結された鎖の残りの部分の統計重率を表す。また、和 $\sum_{i=0}^{N}$ は、部分鎖に含まれる $N+1$ 個の全てのセグメントについて足すことを表す。(2.18) 式の物理的意味については、図 2.3 を参照。

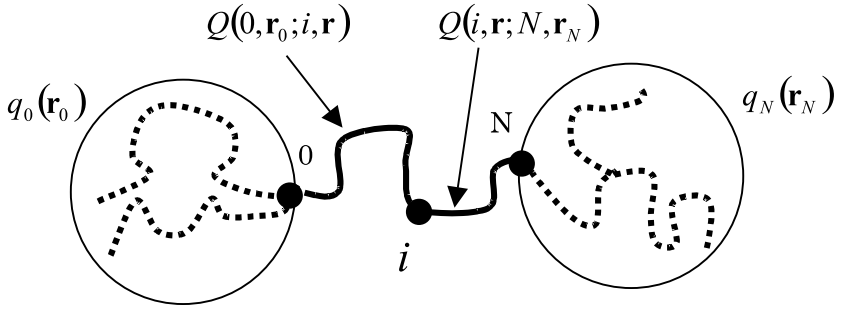


図 2.3: (2.18) 式の物理的意味を表す概念図

(2.18) 式の規格化定数は、(2.10) (2.11) と同様に、この部分鎖の属する高分子鎖の統計的扱いがカノニカル集団であるか、あるいはグランド・カノニカル集団であるかによって、以下のように異なる。

1) カノニカル集団の場合

$$C_r^{(p)} = \frac{M_r^{(p)}}{\int d\mathbf{r}_0 \int d\mathbf{r}_N q_0(\mathbf{r}_0) Q_K(0,\mathbf{r}_0;N,\mathbf{r}_N) q_N(\mathbf{r}_N)} \tag{2.19}$$

ただし、$M_r^{(p)}$ は系全体に含まれるこの部分鎖の総数であり、明らかに p 種の鎖の総数（$M^{(p)}$ とする）に等しい。また、$C_r^{(p)}$ も部分鎖に依らず、鎖の種類だけで決まる量（$C^{(p)}$ とする）であることが容易に示せる。

2) グランド・カノニカル集団の場合

$$C_r^{(p)} = \frac{\phi_p^{(\mathrm{bulk})}}{\sum_{r''} N_{r''}^{(p)}} \exp\Big[\sum_{r'} N_{r'}^{(p)} W_{K_{r'}^{(p)}}^{(\mathrm{bulk})}\Big] \tag{2.20}$$

ただし、$\phi_p^{(\mathrm{bulk})}$ は系が平衡状態にあるバルク（粒子源）における高分子鎖の濃度である。この場合も明らかに $C_r^{(p)}$ は部分鎖に依らない鎖の種類だけで決まる量（$C^{(p)}$）である。

実際の計算にあたっては、(2.12) (2.13) 同様に、以下の初期点に関して積分された関数を用いると便利である。

$$Q_K(i,\mathbf{r}_i) = \int d\mathbf{r}_0 q_0(\mathbf{r}_0) Q_K(0,\mathbf{r}_0;i,\mathbf{r}_i) \tag{2.21}$$

$$\widetilde{Q}_K(N-i,\mathbf{r}_i) = \int d\mathbf{r}_N q_N(\mathbf{r}_N) Q_K(N,\mathbf{r}_N;i,\mathbf{r}_i) \tag{2.22}$$

これらは、(2.15) (2.16) と全く同一の発展方程式に従うが、初期値は

$$Q_K(0,\mathbf{r}) = q_0(\mathbf{r}) \tag{2.23}$$

$$\widetilde{Q}_K(0,\mathbf{r}) = q_N(\mathbf{r}) \tag{2.24}$$

で与えられる。この初期条件の下で発展方程式 (2.15) (2.16) を解くことにより求められた経路積分を用いると、位置 $\mathbf{r}$ におけるセグメント濃度へのこの部分鎖からの寄与は、

$$\phi_r^{(p)}(\mathbf{r}) = C_r^{(p)} \sum_i Q_K(i,\mathbf{r}) \widetilde{Q}_K(N_r^{(p)} - i, \mathbf{r}) \tag{2.25}$$

となる。

鎖全体の経路積分は、上記の計算を各部分鎖について行うことにより実行できる。このようにして、位置 $\mathbf{r}$ における K 種のセグメント濃度 $\phi_K(\mathbf{r})$ は、(2.25) を K 種のセグメントで構成される全ての部分鎖について足し合わせることで

$$\phi_K(\mathbf{r}) = \sum_p \sum_{r \in K \text{ 種部分鎖}} \phi_r^{(p)}(\mathbf{r}) \tag{2.26}$$

と求められる。

枝分かれ鎖に対する自己無撞着場の定義は、(2.1) と同一であり、2.1 節と同様の自己無撞着な方法で決定される。

2.3 内部状態をもつ系への拡張

単一のモノマーから構成される部分鎖に対する 2.1 および 2.2 節の定式化は、部分鎖がある一定の統計分布に従う複数のモノマーから構成されている場合にも容易に拡張可能である。そのような部分鎖の例として、

1) テーパー（傾斜）共重合鎖

2) 基底状態にあるモノマーと励起状態にあるモノマーが一定の比率で混在する鎖

などがあげられる。これらの鎖のコンフォメーションの統計性の計算は、2.1 節の経路積分の定義にマルチ・ステイト（複数の内部状態）という概念を導入することで計算可能になる。

例えば、A 種のセグメントと B 種のセグメントの 2 種のセグメントからなるテーパード共重合部分鎖のコンフォメーションは、これら 2 種のセグメントに対応する 2 種の経路積分 $Q_A(i,\mathbf{r})$ と $Q_B(i,\mathbf{r})$ を用いて計算される。部分鎖内でのセグメントの配列の統計性を表す物理量として、次式で定義される状態間遷移確率行列 (transition state probability matrix) を導入する。

$$T_{KK'}(i) \equiv \left[\begin{array}{l} i \text{ セグメントが } K \text{ 種であるときに、} \\ (i+1) \text{ セグメントが } K' \text{種である確率} \end{array} \right] \tag{2.27}$$

i 番目のモノマーにおける A セグメントの体積分率を $f_A(i) = f(i)$ とすれば、$f_B(i) = 1 - f(i)$ である。i 番目と i+1 番目のモノマーにおけるセグメントの存在確率に何の相関もなければ、

$$T_{AA}(i) = T_{BA}(i) = f(i+1), \;\; T_{AB}(i) = T_{BB}(i) = 1 - f(i+1) \; . \tag{2.28}$$

となる。

この状態間遷移確率行列を用いると、(2.15) の経路積分の発展方程式は、以下のように拡張される。

$$\frac{\partial}{\partial i} Q_K(i,\mathbf{r}) = \sum_{K'} T_{KK'}(i) \Big[\frac{b_{K'}^2}{6} \nabla^2 - \beta V_{K'}(\mathbf{r}) \Big] Q_{K'}(i,\mathbf{r}) \tag{2.29}$$

ここで、有効結合長 b のセグメント種 K への依存性をあらわに記した。

2.4 自由エネルギー

自由エネルギーは、エントロピーによる項（主にコンフォメーションの自由度）と、セグメント間相互作用及び外場によるエンタルピー項に分けられる。上記の経路積分を用いると、鎖のコンフォメーションのエントロピーを考慮した上で自由エネルギーを求めることができる。この SCF 法では、Flory − Huggins モデルでは無視されている鎖の分岐構造やコンフォメーションの変形によるエントロピー変化の効果を含めることができることから、任意の高分子系に対して、自由エネルギーを精度よく求めることができるという利点がある。

導出の詳細を省略して結果のみ記せば、経路積分を用いた全 Helmholtz 自由エネルギーの表式は以下のようになることが知られている。

$$\begin{aligned}\mathcal{F}\big[\{\phi_K\},\{V_K\}\big] &= -k_{\rm B}T\sum_p M_p \ln \mathcal{Z}_p + \mathcal{W}\big[\{\phi_K\}\big] \\ &\quad -\sum_K \int d\mathbf{r} V_K(\mathbf{r})\phi_K(\mathbf{r}) \\ &\quad +k_{\rm B}T\sum_p M_p \ln M_p \end{aligned} \tag{2.30}$$

(2.30) 式の右辺第 1 項において、$\mathcal{Z}_p$ は自己無撞着場中の 1 本の p 種高分子鎖の状態和であり、

$$\mathcal{Z}_p = \sum_{\text{鎖の可能な全配置}} \exp\Big[-\beta \sum_{\text{全セグメント } i} V_{K(i)}(\mathbf{r}_i)\Big] \tag{2.31}$$

で定義される。ここで、$K(i)$ は i セグメントのセグメント種である。(2.31) 式は、さらに経路積分を用いて以下のように書くことができる。

$$\begin{aligned}\mathcal{Z}_p &= \sum_{j\in \text{ 全ての部分鎖の結合点}} \int d\mathbf{r}_j \prod_{r\in \text{ 全ての部分鎖}} Q(0,\mathbf{r}_0;N_r^{(p)},\mathbf{r}_{N_r^{(p)}}) \\ &= \int d\mathbf{r} Q(i,\mathbf{r})\widetilde{Q}(N_r^{(p)}-i,\mathbf{r}) \end{aligned} \tag{2.32}$$

ただし、第 2 式の積分は任意に選んだ部分鎖 r のどれか 1 つのセグメント i について行う。

(2.31) 式の定義より明らかなように、(2.30) 式の右辺第 1 項は、鎖のコンフォメーションのエントロピーを表す。一方、(2.30) 式の第 3 項は拘束条件からくる拘束力の寄与の項、第 4 項は混合のエントロピーに相当する項で、後者は系の成分固有の定数項である。(2.30) 式の第 2 項は、セグメント間相互作用項で、

$$\mathcal{W}\big[\{\phi_K\}\big] = \frac{1}{2}\sum_K\sum_{K'}\int d\mathbf{r}\epsilon_{KK'}\phi_K(\mathbf{r})\phi_{K'}(\mathbf{r}) \tag{2.33}$$

と表される。ここに示す $\epsilon_{KK'}$ は、K 種-K' 種セグメント間相互作用エネルギーパラメーターで、Flory の χ パラメーターとは、以下の関係がある。

$$\chi_{KK'} = z\beta\big[\epsilon_{KK'} - \frac{1}{2}(\epsilon_{KK}+\epsilon_{K'K'})\big] \tag{2.34}$$

また、Approximate Density Functional 法では、自由エネルギーは濃度場 ϕ_K の近似的な汎関数で表される。AB ホモポリマーブレンドを例にとると、Flory-Huggins 自由エネルギー

$$\frac{\mathcal{F}}{k_{\rm B}T} = \frac{\phi_A}{N_A}\ln\phi_A + \frac{1-\phi_A}{N_B}\ln(1-\phi_A) + \chi\phi_A(1-\phi_A) \tag{2.35}$$

や、それを ϕ_A の平均値の周りで展開した式、あるいは秩序変数の空間相関を取り入れた

$$\frac{\mathcal{F}\big[\{\phi_A(\mathbf{r})\}\big]}{k_{\rm B}T} = \int d\mathbf{r}\Big[\frac{\phi_A}{N_A}\ln\phi_A + \frac{1-\phi_A}{N_B}\ln(1-\phi_A) + \chi\phi_A(1-\phi_A) + \frac{b^2}{36\phi_A(1-\phi_A)}|\nabla\phi_A|^2 \tag{2.36}$$

等が用いられる。但し、非圧縮条件により $\phi_B = 1-\phi_A$ となることを用いた。

2.5 化学ポテンシャルと拘束力

セグメント濃度場をある与えられたプロファイル $\{\phi_K(\mathbf{r})\}$ に拘束したとする。$\{\phi_K(\mathbf{r})\}$ は平衡状態のプロファイルとは限らないが、これを拘束した下での鎖のコンフォメーションの平衡状態を考えることができる。このとき、位置 $\mathbf{r}$ で K 種セグメント ($K = A, B, C, \cdots$) が感じる化学ポテンシャル $\mu_K(\mathbf{r}) \equiv \delta\mathcal{F}/\delta\phi_K(\mathbf{r})$ は経路積分の定義及び自由エネルギーの表式 (2.30) より、以下のように表されることが示せる。

$$\mu_K(\mathbf{r}) = -V_K(\mathbf{r}) + \sum_{K'} \epsilon_{KK'}\phi_{K'}(\mathbf{r}) = -V_K(\mathbf{r}) + W_K(\mathbf{r}) \tag{2.37}$$

ただし、(2.2) 式の $W_K(\mathbf{r})$ の定義を用いた。別の言い方をすれば、セグメント濃度場を $\{\phi_K(\mathbf{r})\}$ に拘束したときの (2.1) 式の拘束力は、化学ポテンシャルを用いて $-\mu_K(\mathbf{r})$ で与えられることになる。

セグメント濃度分布の平衡状態を考える場合は、拘束条件を局所的な非圧縮条件

$$\sum_K \phi_K(\mathbf{r}) = 1 \tag{2.38}$$

のみで与えることになる。このときは、(2.37) 式の $\mu_K(\mathbf{r})$ がセグメント種に依らずに全て等しくなればよい。

$$\mu_K(\mathbf{r}) = \mu(\mathbf{r}) \quad \text{for all } K \tag{2.39}$$

すなわち、局所的非圧縮の拘束条件に対する拘束力はセグメント種に依らない関数、$-\mu(\mathbf{r})$ で与えられる。実際には自己無撞着場の不定性によりセグメントごとに定数分のずれが許される。従って、(2.39) 式は平衡セグメント濃度分布を与える μ_K に対する十分条件となる。

このように、拘束条件が与えられ、対応する拘束力が決まると、それらと併せて図 2.1 のように (2.1) (2.15) (2.16) (2.17) を同時に満たす自己無撞着解を求めることにより、その拘束条件の下での平衡状態が得られる。

2.6 GRPA 法

SCF 法は正確な方法であるが、計算速度が遅い。そこで、精度は SCF 法より劣るが、計算速度が速い動的な密度汎関数法をここでは説明する。この方法は基本的には Ginzburg-Landau 自由エネルギーモデルに RPA(Random Phase Approxiation) による理想鎖の統計を導入したものであるので、GRPA 法と呼ぶことにする。この方法は以下の Bohbot-Raviv と Wang らの自由エネルギーモデルを用いる [13]。

$$\begin{aligned}\mathcal{F}[\{\phi_i\}] &= \frac{1}{\beta}\int d\mathbf{r}\sum_i \frac{\phi_i(\mathbf{r})}{N_i}\ln\phi_i(\mathbf{r}) - \frac{1}{2\beta}\int d\mathbf{r}\sum_i \frac{1}{N_i\bar{\phi}_i}\delta\phi_i(\mathbf{r})\delta\phi_i(\mathbf{r}) \\ &\quad + \frac{1}{2\beta}\sum_{ij}\int d\mathbf{q} S_{ij}^{-1}(\mathbf{q})\delta\phi_i(\mathbf{q})\delta\phi_j(-\mathbf{q}),\end{aligned} \tag{2.40}$$

ここで、i, j は部分鎖のインデックスであり、右辺の第 1 項は、部分鎖による Flory-Huggins のエントロピー項である。第 2 項は、第 1 項のセグメントの濃度ゆらぎによる 2 次の展開項に負の符号をつけたものである。セグメントの濃度ゆらぎはセグメントの仕込み濃度とセグメントの局所的な濃度の差であり以下の式で表される。

$$\delta\phi_i(\mathbf{r}) \equiv \phi_i(\mathbf{r}) - \bar{\phi}_i. \tag{2.41}$$

(2.40) 式の第 3 項は、セグメントの濃度ゆらぎによるの 2 次の展開項をフーリエ変換したものであり、係数は 2 点間のセグメントの濃度ゆらぎの散乱関数となっている。この項は、自由エネルギーに対するセグメントの濃度ゆらぎの 2 次の展開項全てを含んでいる。よって、第 1 項のセグメントの濃度ゆらぎによる 2 次の展開項も含んでいるので、第 2 項によりその過剰分を打ち消している。第 3 項の係数の散乱関数は RPA 法により任

意の構造をもつポリマーに対して求めることができる [14]。波数空間で第 3 項を求めた後、再び実空間に戻すと、(2.40) 式は次のような式となる。

$$\begin{aligned}\mathcal{F}[\{\phi_i\}] &= \frac{1}{\beta}\sum_i^n \int d\mathbf{r}\frac{\phi_i(\mathbf{r})}{N_i}\ln\phi_i(\mathbf{r}) - \frac{1}{2\beta}\sum_i^n \int d\mathbf{r}\frac{1}{N_i\bar{\phi}_i}\delta\phi_i(\mathbf{r})\delta\phi_i(\mathbf{r}) \\ &\quad + \frac{1}{2}\int d\mathbf{r}\ \mathbf{x}^T(\mathbf{r})\mathbf{u}(\mathbf{r}) \end{aligned} \tag{2.42}$$

ここで $\mathbf{x}(\mathbf{r})$ は、セグメントの濃度ゆらぎを要素としたベクトルであり、$\mathbf{x}^T(\mathbf{r})$ はその転置ベクトルである。$\mathbf{u}(\mathbf{r})$ はセグメントの濃度揺らぎに働くポテンシャルである。

非圧縮条件下で式を ϕ で汎関数微分すれば部分鎖の化学ポテンシャルが以下のように得られる。

$$\mu_i(\mathbf{r}) = \frac{\delta\mathcal{F}[\{\phi_i\}]}{\delta\phi_i(\mathbf{r})} \tag{2.43}$$

$$\begin{aligned} &= \frac{1}{\beta}\Big\{\frac{\ln\phi_i(\mathbf{r})}{N_i} + \frac{1}{N_i} - \frac{\ln\phi_n(\mathbf{r})}{N_n} - \frac{1}{N_n} - \frac{1}{\bar{\phi}_i N_i}\delta\phi_i(\mathbf{r}) + \frac{1}{\bar{\phi}_n N_n}\delta\phi_n(\mathbf{r})\Big\} \\ &\quad + u_i(\mathbf{r}). \end{aligned} \tag{2.44}$$

この式を利用して動力学が可能となる。この方法は RPA を利用しているので、厳密には weak segregation の領域でしか利用できない。しかし、高分子溶融体混合物の相分離過程を追うには定性的であるが高速に結果を出せるので便利な方法である。ただし、GRPA 法はフーリエ変換を用いるため境界条件に制約を生じる。

2.7 オブスタクル

高分子溶融体中に物体が存在し、高分子の存在の妨げになるような場合に、高分子の存在状態を調べることは、高分子材料を研究する上で重要となる。そのような物体を総称してオブスタクルと呼ぶことにする。

SUSHI では、まず、オブスタクルとしてナノ粒子を存在させることに着手した。SUSHI では、ナノ粒子やファイバーの内外に高分子を存在させた系を SCF 計算することができる。ただし、動的な計算には対応していない。

2.7.1 ソフト粒子

ハードな境界面をもつオブスタクルの SCF 計算はコストがかかるので、他のアイディアとしてソフトな界面をもつ粒子の計算を考えることができる [15]。粒子は、系に存在させた溶媒（液体）を粒子形状に固定した疑似粒子とする。粒子の存在領域は次式で表し界面に濃度勾配を導入する。

$$\phi_p(\mathbf{r}) = \frac{\tanh\{\alpha(r - |\mathbf{r} - \mathbf{r}_0|)\} + 1}{2}. \tag{2.45}$$

ここで、$\mathbf{r}$ は空間の任意の位置、α は界面幅を決める任意のパラメータ、r は粒子の半径、$\mathbf{r}_0$ は粒子中心の座標である。次に静的な SCF 計算が収束した後に次式に従い、粒子中心に働く力を計算し、この力を元に粒子を微小時間動かし、SCF の静的計算に戻る。この繰り返しで相分離と粒子の移動が行え、安定な相分離構造と粒子位置が求められる。

$$\mathbf{F}_p = D_e \int_S \nabla\mu_p(\mathbf{r}_s)\frac{\mathbf{r} - \mathbf{r}_0}{|\mathbf{r} - \mathbf{r}_0|}ds \tag{2.46}$$

ここで、D_e は、任意の運動係数、$\mu_p(\mathbf{r}_s)$ は粒子表面の位置 $\mathbf{r}_s$ における SCF 法で求められる粒子として模擬した溶媒に働く化学ポテンシャルである。ds は粒子表面での面積分を表すものとする。

2.8 動的平均場法

動的平均場法は、セグメント化学ポテンシャルの空間勾配を駆動力とした拡散ダイナミックスを、鎖のコンフォメーションまで考慮して、取り扱う方法である。

時刻 t におけるセグメント濃度場およびセグメント化学ポテンシャルをそれぞれ $\phi_K(\mathbf{r},t)$, $\mu_K(\mathbf{r},t)$ とかくと、一般的には時間発展方程式は、以下のような形のものが仮定される。

$$\frac{\partial}{\partial t}\phi_K(\mathbf{r},t) = \sum_{K'}\int d\mathbf{r}'\Lambda_{KK'}(\mathbf{r},\mathbf{r}')\mu_{K'}(\mathbf{r}',t) + \xi_K(\mathbf{r},t) \tag{2.47}$$

セグメント化学ポテンシャル $\mu_K(\mathbf{r})$ を (2.37) で与え、これまでの節で説明された一連の自己無撞着方程式を同時に解くことにより、鎖のコンフォーメーションを考慮したセグメントの拡散ダイナミクスを記述することができる。$\Lambda_{KK'}(\mathbf{r},\mathbf{r}')$ は位置 $\mathbf{r}'$ にある K' 種セグメントに作用する外力が、別の位置 $\mathbf{r}$ における K 種のセグメント濃度の変化を生じさせる割合であり、非局所運動係数と呼ぶ。また、$\xi_K(\mathbf{r},t)$ は熱揺らぎに対応するランダムノイズで、揺動散逸定理に従って

$$\langle\xi_K(\mathbf{r},t)\xi_{K'}(\mathbf{r}',t')\rangle = 2k_BT\Lambda_{KK'}(\mathbf{r},\mathbf{r}')\delta(t-t') \tag{2.48}$$

を満たす。

SUSHI における動的平均場法では、セグメントの濃度変化が化学ポテンシャルの勾配に比例したセグメントの拡散流によって生じる場合（Fick の法則）を扱う。このとき (2.47) 式は、

$$\frac{\partial}{\partial t}\phi_K(\mathbf{r},t) = \nabla\cdot[L_K(\mathbf{r},t)\nabla\{\mu_K(\mathbf{r},t)+\lambda(\mathbf{r},t)\}] + \xi_K(\mathbf{r},t) \tag{2.49}$$

のように簡単化される。ここで、$L_K(\mathbf{r},t)$ は K 種セグメントの運動係数を表す。$\lambda(\mathbf{r},t)$ は局所的非圧縮条件から決まる Lagrange 未定定数である。K 種セグメント鎖の濃度が非常に希薄な場合、$L_K(\mathbf{r},t)$ は $\phi_K(\mathbf{r},t)$ に依存し、次のように近似できる。

$$L_K(\mathbf{r},t) = \frac{N}{k_BT}D_G\phi_K(\mathbf{r},t) \qquad (\phi_K << 1) \tag{2.50}$$

ここで D_G は孤立 K 種セグメント鎖が非常に希薄な場合の重心拡散係数である。この値は K 種セグメント鎖が置かれている条件により異なる。代表的な条件としては次のようなものがある。

1) K 種セグメント鎖が希薄溶液中にある場合 (Rouse Dynamics 条件)

$$D_G = \frac{k_BT}{N_K\zeta} \tag{2.51}$$

$$L_K(\mathbf{r},t) = \frac{1}{\zeta}\phi_K(\mathbf{r},t) \qquad (\phi_K << 1) \tag{2.52}$$

2) K 種セグメント鎖が溶融高分子中にある場合 (Reptation Dynamics 条件)

$$D_G = (\frac{a^2}{b_K^2})\frac{k_BT}{3N_K^2\zeta} \tag{2.53}$$

$$L_K(\mathbf{r},t) = (\frac{a^2}{b_K^2})\frac{1}{3N_K\zeta}\phi_K(\mathbf{r},t) \qquad (\phi_K << 1) \tag{2.54}$$

ここで、ζ は K 種セグメント鎖以外の成分を溶媒と見立てた場合の溶媒の粘度、a は絡み合った高分子の網目を特徴づける量（網目の平均間隔）である [22],[23]。

K 種セグメント鎖の濃度が高い場合にはこれらの近似の正当性は保証されない。あくまでも現象論モデルであるということを十分認識した上で、上記の希薄な濃度での条件を高濃度側に拡張して使用することはできる。

$\lambda(\mathbf{r},t)$ は、局所的非圧縮条件から決まり、

$$\nabla\lambda(\mathbf{r},t) = -\frac{\sum_{K'}L_{K'}(\mathbf{r},t)\nabla\mu_{K'}(\mathbf{r},t)}{\sum_{K''}L_{K''}(\mathbf{r},t)} \tag{2.55}$$

で与えられる。この場合、$\xi_K(\mathbf{r},t)$ は

$$\langle \xi_K(\mathbf{r},t)\xi_{K'}(\mathbf{r}',t')\rangle = 2k_BT\mathcal{L}_{KK'}(\mathbf{r},t)\nabla^2\delta(\mathbf{r}-\mathbf{r}')\delta(t-t') \tag{2.56}$$

を満たすガウシアンノイズとなる。ただし、$\mathcal{L}_{KK'}(\mathbf{r},t)$ は運動係数 $L_K(\mathbf{r},t)$ を用いて

$$\mathcal{L}_{KK'}(\mathbf{r},t) = L_K(\mathbf{r},t)\delta_{KK'} + \frac{L_K(\mathbf{r},t)L_{K'}(\mathbf{r},t)}{\sum_{K''} L_{K''}(\mathbf{r},t)} \tag{2.57}$$

で与えられる。

動的平均場法では時々刻々と変化する濃度プロファイル $\{\phi_K(\mathbf{r})\}$ に従って、これを拘束した下での自己無撞着方程式を解いていくことになる。これを数値的に行うための具体的な手続きについては、次章以下で説明していく。

動的平均場法 (2.49) 式により最終的に得られる構造は、(2.39) 式を満たすという意味において、静的平均場の計算と本質的には変わらない。ただしその構造は、必ずしも自由エネルギーを真に最小にするものではなく、多くの場合局所的な極小を与えるものとなる。例えば、ブロック共重合体のミクロ相分離ではクエンチの深さにもよるが、たいていは多数の欠陥を含んだ準安定構造が最終構造として得られる。しかし、一様状態からクエンチして実験的に得られるミクロ相分離構造も同様に多数の欠陥を含んでおり、その意味ではそこに至るまでのプロセスを反映した構造が正しく得られているということができる。なお、動的なプロセスを正しく考慮する意味では、(2.49) 式におけるノイズ項は重要であるが、長鎖のメルトでは実質的に無視できるとして考慮されないことが多い。

動的計算は Approximate Density Functional 法を用いても可能である。こちらは、(2.35) あるいは (2.36) で与えられる自由エネルギーを汎関数微分することにより、化学ポテンシャルが解析的に表されるので、各時刻で自己無撞着方程式を解く動的平均場法よりも計算コストは低い。しかし、鎖のコンフォメーションが正確に考慮されないので、得られる結果は定量性に欠けたものとなる。

2.8.1 ずり流動

セグメント濃度の時間発展方程式には外的な流動を導入することが可能である。K-種セグメントに対する外的な流動速度を $v_K(\mathbf{r})$ とすると、(2.49) 式は次のように拡張される。

$$\frac{\partial}{\partial t}\phi_K(\mathbf{r},t) = \nabla\cdot[L_K(\mathbf{r},t)\nabla\{\mu_K(\mathbf{r},t)+\lambda(\mathbf{r},t)\}] + \xi_K(\mathbf{r},t) - \nabla\{v_K(\mathbf{r})\phi_K(\mathbf{r})\} \tag{2.58}$$

規則メッシュにおいて XZ 平面に平行で X 方向に向かうずり流動の場合、$v_K(\mathbf{r})$ は

$$v_K(\mathbf{r}) = \big(\dot{\gamma}(r_y - r_{y_0}), 0, 0\big) \tag{2.59}$$

となる。ここで、$\dot{\gamma}$ はずり速度で

$$\dot{\gamma} = \frac{\partial v_x}{\partial y} \tag{2.60}$$

であり、r_{y_0} は Y 座標でずり速度が 0 となる位置である。周期境界条件を課した実際の計算ではずりにより移動する XZ 平面の境界に Lees-Edwards の境界条件が導入される。

2.8.2 圧縮性のある動力学

局所的非圧縮条件から決まる Lagrange の未定定数である $\lambda(\mathbf{r},t)$ 項が (2.49) 式になくても、圧縮率 κ を導入することによってセグメント濃度の時間発展を計算することが可能である [8]。この場合、自由エネルギーには

弾性エネルギー項が追加され、(2.30) 式は次のように拡張される。

$$\begin{aligned}\mathcal{F}\big[\{\phi_K\},\{V_K\}\big] &= -k_{\mathrm{B}}T\sum_p M_p \ln \mathcal{Z}_p + \mathcal{W}\big[\{\phi_K\}\big] \\ &\quad -\sum_K \int d\mathbf{r} V_K(\mathbf{r})\phi_K(\mathbf{r}) \\ &\quad +k_{\mathrm{B}}T\sum_p M_p \ln M_p \\ &\quad +\frac{1}{2\kappa}\big(\sum_K \phi_K(\mathbf{r})-1\big)^2 \end{aligned} \tag{2.61}$$

そして、自己無撞着場も拡張され、(2.1) 式は次のようになる。

$$V_K(\mathbf{r}) = W_K(\mathbf{r}) + V_c(\mathbf{r}) + \Big[\text{拘束条件から決まる拘束力}\Big] \tag{2.62}$$

ここで、$V_c(\mathbf{r})$ は非圧縮な条件を満たすためのポテンシャルであり、

$$V_c(\mathbf{r}) = \frac{1}{\kappa}\big(\sum_K \phi_K(\mathbf{r})-1\big) \tag{2.63}$$

となる。$\lambda(\mathbf{r},t)$ の代わりにこれらの式を用いた動力学では完全な非圧縮条件は満たされなくなる。つまり、局所的にではあるが、$\sum_K \phi_k(\mathbf{r}) \neq 1$ となる。

2.8.3 化学反応の取り扱い

動的平均場法 (2.49) 式の右辺に、化学反応により時間的に変化する ϕ の反応項を追加すれば反応をシュミレートすることが可能である。例えば $\phi^{(A)}$ が反応定数 k により $\phi^{(B)}$ になるような場合、反応の式は以下のようになる。

$$\frac{\partial\phi^{(A)}(\mathbf{r},t)}{\partial t} = \frac{\partial\phi^{(A)}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} - k\phi^{(A)}(\mathbf{r},t) \tag{2.64}$$

$$\frac{\partial\phi^{(B)}(\mathbf{r},t)}{\partial t} = \frac{\partial\phi^{(B)}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} + k\phi^{(A)}(\mathbf{r},t) \tag{2.65}$$

ここで、右辺第 1 項は、(2.49) 式の右辺で与えられる拡散による寄与である。右辺第 2 項が反応項である。具体的な方法は以降の章で説明する。

2.8.4 ハイブリッド法

GRPA 法と SCF 法をハイブリッドすることにより、SCF 法の精度を損なわずに、高速に高分子溶融体の動的平均場計算を行える [14]。

ハイブリッド法は、GRPA 法の化学ポテンシャルを SCF 法の化学ポテンシャルにて逐次補正することにより次のように行う。ある時刻 t におけるハイブリッド法と GRPA 法による部分鎖 i の場所 $\mathbf{r}$ における化学ポテンシャルをそれぞれ $\mu_i^{HYB}(\mathbf{r},t)$、$\mu_i^{RAP}(\mathbf{r},t)$ とすれば、

$$\mu_i^{HYB}(\mathbf{r},t) = \mu_i^{RPA}(\mathbf{r},t) + \Delta\bar{\mu}_i(\mathbf{r}) \tag{2.66}$$

と表される。ここで、$\Delta\bar{\mu}_i(\mathbf{r})$ は補正のための化学ポテンシャルであり、SCF 法による化学ポテンシャルを $\mu_i^{SCF}(\mathbf{r},t)$ として

$$\Delta\bar{\mu}_i(\mathbf{r}) = \mu_i^{SCF}(\mathbf{r},t_0) - \mu_i^{RAP}(\mathbf{r},t_0) \tag{2.67}$$

とする。但し、各時間ステップ毎に $\Delta\bar{\mu}_i(\mathbf{r})$ の計算を行わず、ある期間のインターバルを空けて時刻 t_0 に計算する。このインターバルの期間が短ければハイブリッド法は SCF 法で得られる動力学のプロファイルに収束する。この手法は SCF 法による大規模計算に道を開くものである。

2.8.5 流体力学的効果

動的平均場法へ流体力学的効果を導入することが可能である [21] 。詳しくは Muffin のマニュアルを参照されたい。

解くべき Navier-Stokes の方程式は、次のようになる。

$$\rho\frac{\partial \mathbf{v}(\mathbf{r},t)}{\partial t} = -\rho\{\mathbf{v}(\mathbf{r},t)\cdot\nabla\}\mathbf{v}(\mathbf{r},t) - \nabla p(\mathbf{r},t) + \nabla\{\eta(\mathbf{r},t)\nabla\cdot\mathbf{v}(\mathbf{r},t)\} - \sum_i \phi_i(\mathbf{r},t)\nabla\mu_i(\mathbf{r},t) \tag{2.68}$$

ここで、$p(\mathbf{r},t)$ は圧力、$\eta(\mathbf{r},t)$ は局所的な粘度、最後の項が動的平均場法と流体力学をカップルするために導入された体積力である。粘度 $\eta(\mathbf{r},t)$ は、部分鎖のみの溶融体が異なる粘度をもつとして、次式のような加成性を用いて表されると仮定する。

$$\eta(\mathbf{r},t) = \sum \eta_i\phi_i(\mathbf{r},t) \tag{2.69}$$

ここで、η_i は、部分鎖 i のみの場合の粘度である。

非圧縮条件

$$\nabla\cdot\mathbf{v}(\mathbf{r},t) = 0 \tag{2.70}$$

を導入し、断熱条件

$$\frac{\partial \mathbf{v}(\mathbf{r},t)}{\partial t} = 0 \tag{2.71}$$

を仮定すると、次の $p(\mathbf{r},t)$ に関するのポアソン方程式が導かれる。

$$\nabla^2 p(\mathbf{r},t) = \nabla\cdot[-\rho\{\mathbf{v}(\mathbf{r},t)\cdot\nabla\}\mathbf{v}(\mathbf{r},t) + \nabla\{\eta(\mathbf{r},t)\nabla\cdot\mathbf{v}(\mathbf{r},t)\} - \sum_i \phi_i(\mathbf{r},t)\nabla\mu_i(\mathbf{r},t)] \tag{2.72}$$

この (4) 式を解いて $p(\mathbf{r},t)$ を求め、(2.68) 式に代入して $\mathbf{v}(\mathbf{r},t)$ を更新する。$\mathbf{v}(\mathbf{r},t)$ は、(2.58) 式で利用され、流体力学効果で得られた流速がセグメントの拡散方程式に導入される。

2.8.6 外部電場

効果が弱い外部電場の取り扱い

外部電場が印加され、かつ、その効果が弱いときは、動力学方程式 eq. (2.47) に簡単な以下のような変更を施すことができる [16, 17, 18, 19, 20]。

$$\frac{\partial}{\partial t}\phi_K(\mathbf{r},t) = L\nabla^2\mu_K(\mathbf{r}',t) + \alpha\frac{\partial^2}{\partial z^2}\phi_K(\mathbf{r},t), \tag{2.73}$$

ここで、L はすべてのセグメントの運動係数で、簡単に定数と仮定した。また、ランダム・ノイズの項は無視した。この式の最後の項が外部電場の影響を導入し、次のように表すことができる。

$$\alpha = \frac{\epsilon_0\epsilon_1^2}{\bar{\epsilon}}E_0^2 v L, \tag{2.74}$$

ここで、 ϵ_0 は真空の誘電率, E_0 は外部電場の強さであり、v は 1 本鎖の体積である（単位となる体積を導入するものであり、この記述は原論文の記述を踏襲した)。詳細は以下のようになる [16]。

我々は Z 方向の外部電場下にある系を考える。外部電場は次のようになる。$\mathbf{E}_0 \equiv (0,0,E_0)$. この電場 $\mathbf{E}_0$ の下, 系は誘電分極をし、静電ポテンシャル $\varphi(\mathbf{r})$ が生じる。よって、有効電場は次式で与えられる。

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}_0 - \nabla\varphi(\mathbf{r}). \tag{2.75}$$

我々は、また、外部電場が弱く局所的誘電率が次のように表されるとする。

$$\epsilon(\mathbf{r}) \approx \bar{\epsilon} + \epsilon_1(\phi_A(\mathbf{r}) - f) \tag{2.76}$$

$$\bar{\epsilon} = \epsilon|_{\phi_A=f} = \epsilon_A f + \epsilon_B(1-f) \tag{2.77}$$

$$\epsilon_1 = (\partial\epsilon/\partial\phi_A)\,|_{\phi_A=f} = \epsilon_A - \epsilon_B. \tag{2.78}$$

ポアソン方程式 $\nabla \cdot \epsilon(\mathbf{r})\mathbf{E}(\mathbf{r}) = 0$ が満たされなくてはならないので、

$$\nabla \cdot \{\bar{\epsilon} + \epsilon_1(\phi_A(\mathbf{r}) - f)\}\{\mathbf{E}_0 - \nabla\varphi(\mathbf{r})\} \tag{2.79}$$

$$\approx \nabla \cdot \{-\bar{\epsilon}\nabla\varphi(\mathbf{r}) + \epsilon_1\phi_A(\mathbf{r})\mathbf{E}_0\} = 0, \tag{2.80}$$

ここでは、高次項を無視した。したがって、次の式が成り立つ。

$$\bar{\epsilon}\nabla^2\varphi(\mathbf{r}) = \epsilon_1 E_0 \nabla_Z \phi_A(\mathbf{r}). \tag{2.81}$$

静電エネルギーによる $\phi_A(\mathbf{r})$ に対する化学ポテンシャルは次の式により導出される。

$$\mu_{el}(\mathbf{r}) = -v\frac{\delta}{\delta\phi_A(\mathbf{r})}\Big[\frac{\epsilon_0}{2}\int d\mathbf{r}\frac{\mathbf{E}(\mathbf{r})^2\epsilon(\mathbf{r})}{4\pi}\Big], \tag{2.82}$$

ここでは、レジェンドル変換が利用され非独立変数が $\mathbf{E}(\mathbf{r})$ から $\phi(\mathbf{r})$ に変換され、また、右変の符号は負となる。

(2.75) 式, (2.76) 式 (2.81) 式を用いると、 $\nabla^2\mu_{el}(\mathbf{r})$ は次式で与えられる。

$$\nabla^2\mu_{el}(\mathbf{r}) = -\frac{v\epsilon_0}{8\pi}\nabla^2\epsilon_1(\mathbf{E}_0^2 - 2E_0\nabla_Z\varphi(\mathbf{r}) + (\nabla\varphi(\mathbf{r}))^2) \tag{2.83}$$

$$\approx \frac{v\epsilon_0\epsilon_1 E_0}{4\pi}\nabla_Z\{\nabla^2\varphi(\mathbf{r})\} \tag{2.84}$$

$$= \frac{v\epsilon_0\epsilon_1^2 E_0^2}{4\pi\bar{\epsilon}}\nabla_Z^2\phi_A(\mathbf{r}). \tag{2.85}$$

運動係数 L を掛けてまとめると (2.74) 式が導出される。

外部電場の効果の一般的な取り扱い

ここでは、外部電場の効果を近似なしに導出する。外部電場 $E_0(\mathbf{r})$ の下では, 次のような一般化されたポアソン方程式が満たされなくてはならない。

$$\nabla \cdot \epsilon(\mathbf{r})\{\nabla U'(\mathbf{r}) - E_0(\mathbf{r})\} = -\rho(\mathbf{r}), \tag{2.86}$$

ここで、 $\epsilon(\mathbf{r})$ は局所誘電率であり、$\rho(\mathbf{r})$ は局所電荷である。$U'(\mathbf{r})$ は外部電場と局所電荷により誘起され静電ポテンシャルである。

我々は、次のように、加成性により局所誘電率 $\epsilon(\mathbf{r})$ と局所電荷 $\rho(\mathbf{r})$ を仮定する。

$$\epsilon(\mathbf{r}) = \epsilon_0\sum_K \epsilon_K\phi_K(\mathbf{r}), \tag{2.87}$$

$$\rho(\mathbf{r}) = \sum_K \rho_K\phi_K(\mathbf{r}), \tag{2.88}$$

ここで、ϵ_K と ρ_K は、それぞれ、K タイプのセグメントの比誘電率と単位体積あたりの電荷である。

(2.86) 式は次のように変形できる。

$$\nabla\epsilon(\mathbf{r})\nabla U'(\mathbf{r}) = -\rho(\mathbf{r}) + \nabla \cdot \epsilon(\mathbf{r})E_0(\mathbf{r}). \tag{2.89}$$

この式を解けば誘起された静電ポテンシャル $U'(\mathbf{r})$ を求めることができ、系に対する有効静電ポテンシャル $U(\mathbf{r})$ と有効電場 $E(\mathbf{r})$ を次のように求めることができる。

$$U(\mathbf{r}) = U(\mathbf{r})' - \int E_0(\mathbf{r})d\mathbf{r} \tag{2.90}$$

$$E(\mathbf{r}) = -\nabla U(\mathbf{r})' + E_0(\mathbf{r}) \tag{2.91}$$

これらの効果を SCF 計算に、次に示す自由エネルギーを通して導入することができる。

$$F' = F + \frac{1}{2}\int U(\mathbf{r})\rho(\mathbf{r})dv - \frac{1}{2}\int \epsilon(\mathbf{r})E^2(\mathbf{r})dv, \tag{2.92}$$

ここでは、右辺において、F は静電的な効果のない場合の自由エネルギー、第 2 項は局所電荷によるエネルギー、第 3 項は外部電場によるエネルギーである。自己無撞着場 $V(\mathbf{r})$ はこの自由エネルギーの式を用いて次のように修飾される。

$$V'_K(\mathbf{r}) = W'_K(\mathbf{r}) - \mu_K(\mathbf{r}), \tag{2.93}$$

ここで、 $W'_K(\mathbf{r})$ 修飾されたセグメントの相互作用に関する場であり、次のようになる。

$$W'_K(\mathbf{r}) = W_K(\mathbf{r}) + \rho_K U(\mathbf{r}) - \epsilon_K E^2_K(\mathbf{r}). \tag{2.94}$$

2.9 計算の具体的方法

2.9.1 鎖の分岐構造

高分子鎖の粗視化されたモデルが持つ重要な特徴の一つに分岐構造がある。SCF シミュレーションを行う際に、この分岐構造をどのように表現し、格納するかは重要なポイントとなる。SUSHI を用いた計算においては、2.1 および 2.2 節で説明したように、高分子鎖は特定のモノマー・シークエンスで構成された部分鎖（subchain）が結合点（junction point）で連結された枝分かれ構造としてモデル化される（図 2.4 参照）。ここで、部分鎖とは、同一のモノマーで構成された直鎖構造だけでなく、ランダム、交互、テーパーなどの一定のモノマー・シークエンスから構成される直鎖構造として定義される。このように定義された部分鎖同士を連結する点が結合点であり、その大きさは 0 であると仮定する。

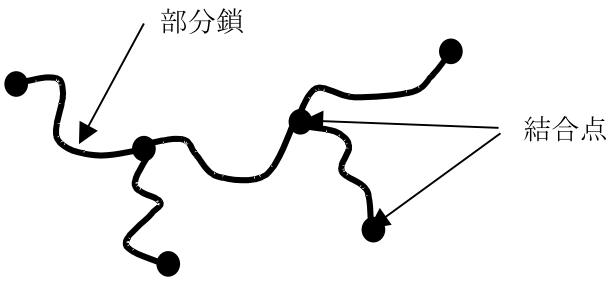


図 2.4: 鎖の分岐構造のモデル

SUSHI においては、図 2.5 に示すように、下記の 3 種類の比較的単純な分岐構造を持った鎖には特別の入力方法を用意している。

1) 直鎖形ブロック共重合体

2) 櫛形ブロック共重合体

3) 星形ブロック共重合体

これら以外の一般の分岐構造に関しては、結合点と部分鎖のつながり方を指定する必要がある。具体的な入力方法の詳細に関しては第 5 章を参照。

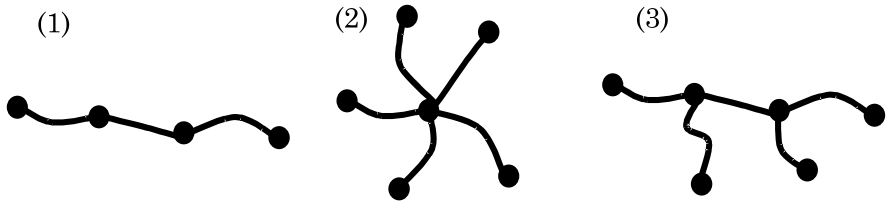


図 2.5: 単純な分岐構造の例: (1) 直鎖型、(2) 星形および (3) 櫛型の各種ブロック共重合体

2.9.2 経路積分の計算法

第 2.2 節で示したように、高分子鎖のコンフォメーションの統計確率分布は、(2.21) 式で定義される部分鎖ごとの経路積分 $Q_K(s,\mathbf{r})$ の従う発展方程式を数値的に解くことで求められる。(ここでは、第 2.1 節で用いた離散的なセグメント・インデックス i の代わりに、連続変数 s を用いた連続体表現により系を記述している。) 結合点 $s=0$ に連結されている他の部分鎖からの寄与は、初期条件 (2.23) 式

$$Q_K(0,\mathbf{r}) = q_0(\mathbf{r}) \tag{2.95}$$

を通じて反映される。

2.3 節で導入した内部状態（マルチ・ステイト）を取り入れた (2.29) 式に対応する発展方程式を、以下のように書く。

$$Q_K(s+ds,\mathbf{r}) = \exp\bigl[-\beta r_K V_K(\mathbf{r})ds/2\bigr] \sum_{K'} T_{KK'}(s)\Bigl(1+\frac{b_{K'}^2}{6}\mathcal{L}ds\Bigr)\Bigl(\exp\bigl[-\beta r_{K'} V_{K'}(\mathbf{r})ds/2\bigr] Q_{K'}(s,\mathbf{r})\Bigr) \tag{2.96}$$

ここに、$Q_K(s,\mathbf{r})$ は $s=0$ から s の間で s セグメントが位置 $\mathbf{r}$ において内部状態 K をとる統計的な重み、$V_K(\mathbf{r})$ は位置 $\mathbf{r}$ において内部状態 K のセグメントが感じる自己無撞着場である。$T_{KK'}(s)$ は、s で指定されるセグメントと $s+ds$ で指定されるセグメントの内部状態がそれぞれ K と K' となる状態間遷移確率である。また、$\mathcal{L}$ はラプラス演算子 ∇^2 であり、その係数に現れた b_K は、セグメントの大きさに相当する「有効結合長 (effective bond length)」である。さらに、セグメントのモル体積を適当な基準セグメント体積で割って無次元化した「比体積 (specific volume)」r_K を導入した。なお、数値計算の便宜上、発展方程式 (2.96) は、わざと、積分形式の漸化式 (2.5) に対応した形にしてある。

数値計算のための離散化を鎖に沿った変数 s と、空間変数 $\mathbf{r}$ に対して行う。変数 s の離散化は発展方程式 (2.96) の ds に有限の定数値を入れることにより実現できる。空間変数 $\mathbf{r}$ の離散化は、各種空間メッシュを導入し、それぞれのメッシュに対応した離散化ラプラス演算子 $\mathcal{L}$ を定義することによって行う。これらの方法について次の節以降で述べる。

2.9.3 各種空間メッシュ構造

SUSHI で行われる具体的な計算は、経路積分とセグメント濃度場の従う偏微分方程式の数値計算であるので、これらの物理量を表す場を離散化して計算する必要がある。SUSHI では、場を離散化する際の基礎となる空間メッシュ（空間格子）には、計算の対象となる現象に応じて種々のものが用意されている。

SCF 計算の対象となる系には、

1) 単一の界面の構造や球状および棒状ミセル構造のような比較的対称性の高い構造

2) 相転移に伴って生成される不規則なドメイン構造

の 2 種類の場合がある。第一の対称性の高い構造に対しては、系の対称性に応じて極座標、円筒座標のような座標系を用いることが計算効率の点で有利である。一方で、第二の不規則構造の場合には、通常の立方格子などが用いられる。

現在 SUSHI で使用可能な空間メッシュは以下の通りである。

1) 1，2，3 次元規則メッシュ(RegularMesh、図 2.6)
x, y, z の各方向に一定の間隔で刻まれた直交座標系メッシュ。不規則構造など、最も一般的な構造の計算に用いる。

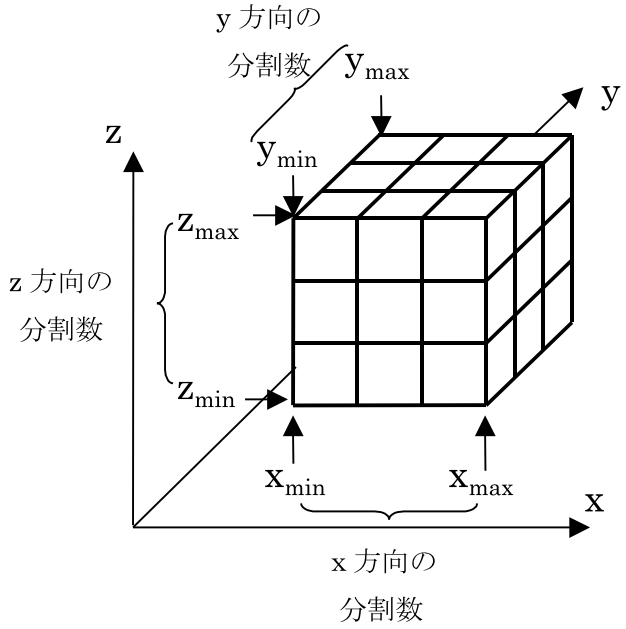


図 2.6: 3 次元規則メッシュ: 各軸方向のメッシュ・サイズは固定。

2) 1，2，3 次元直方メッシュ(RectangularMesh、図 2.7)
x, y, z の各方向に軸を持つ直交座標系であるが、格子間隔が可変にできる。界面で仕切られたドメイン構造のようように、（ある軸方向に垂直な）特定の面の近傍だけで物理量の変化が激しい系の計算に適している。また、図 2.7 に示すメッシュサイズの値は、場所ごとに変化しても構わない。

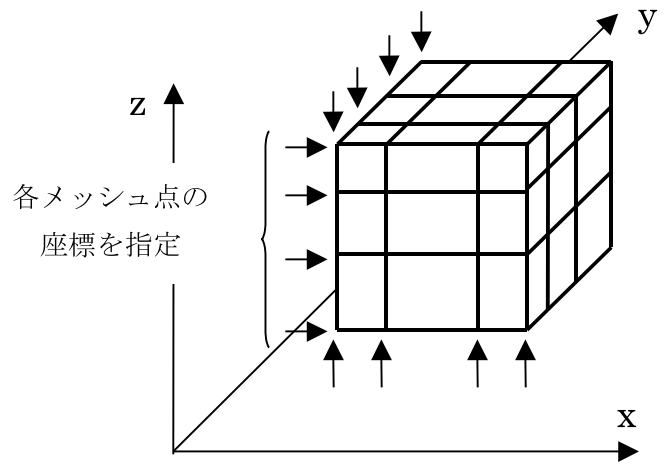


図 2.7: 3 次元直方メッシュ: 各軸方向のメッシュ・サイズは場所ごとに変化して構わない。

3) 極座標メッシュ(SphericalMesh、図 2.8)
原点を中心とする球対称な 3 次元極座標系であり、動径座標 r のみを変数とする座標系。球状ミセルやベシクルのような球対称な系の計算に用いる。メッシュ幅は、一定間隔に限定される。

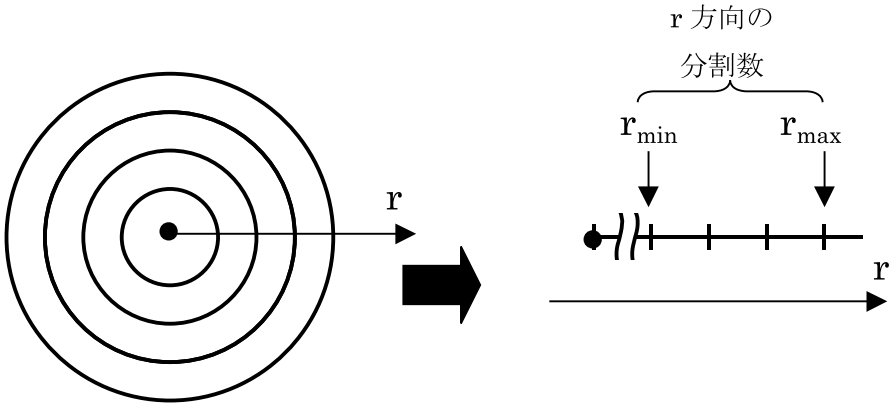


図 2.8: 3 次元極座標メッシュ

4) 円筒座標メッシュ(CylindricalMesh、図 2.9)
1 軸対称性をもつ 3 次元円筒座標であり、動径座標 r 及び軸方向座標 h で記述される。棒状ミセルのような構造の計算に適している。メッシュ幅は固定である。

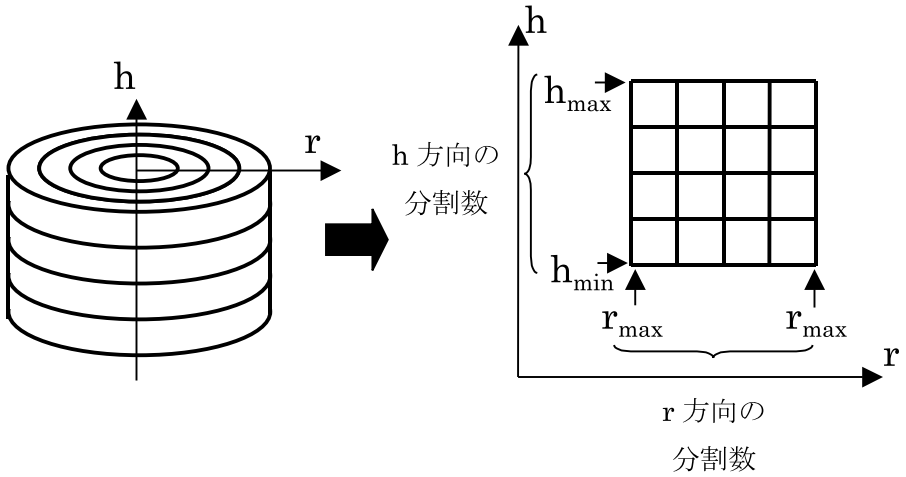


図 2.9: 3 次元円筒座標メッシュ: メッシュ幅は固定。

なお、直方メッシュ以外は、メッシュ幅の指定は、系の各軸方向の長さとその方向にメッシュ点で区切られたセルの総数を指定することで間接的に指定する。直方メッシュでは、メッシュ点の座標の列を与えることで、メッシュ幅を指定する。

2.9.4 境界条件

シミュレーションに用いられる系に課せられる境界条件としては、以下の 2 種類の本質的に異なる境界条件がある。

1) 幾何学的境界条件 (Geometrical boundary conditions)

2) 物理的境界条件 (Physical boundary condition)

これらのそれぞれについて以下で解説する。

1) 幾何学的境界条件（Geometrical boundary conditions)
この境界条件は、系の幾何学的な（あるいはトポロジカルな）性質を表現するためのものである。例えば、図 2.10 のように、2 次元の規則格子系は、系の境界において外界と接しているような系（非周期系）であると見なす場合だけでなく、周期境界条件によって対応する 2 つの境界線がつながれたトーラス状の系を 2 次元的に切り出したもの（周期系）であると考えることも可能である。これらの境界条件は各軸方向ごとに指定できるので、ある軸方向には周期的でかつ別の軸方向には非周期的であるような幾何学的境界条件を設定することも可能である。

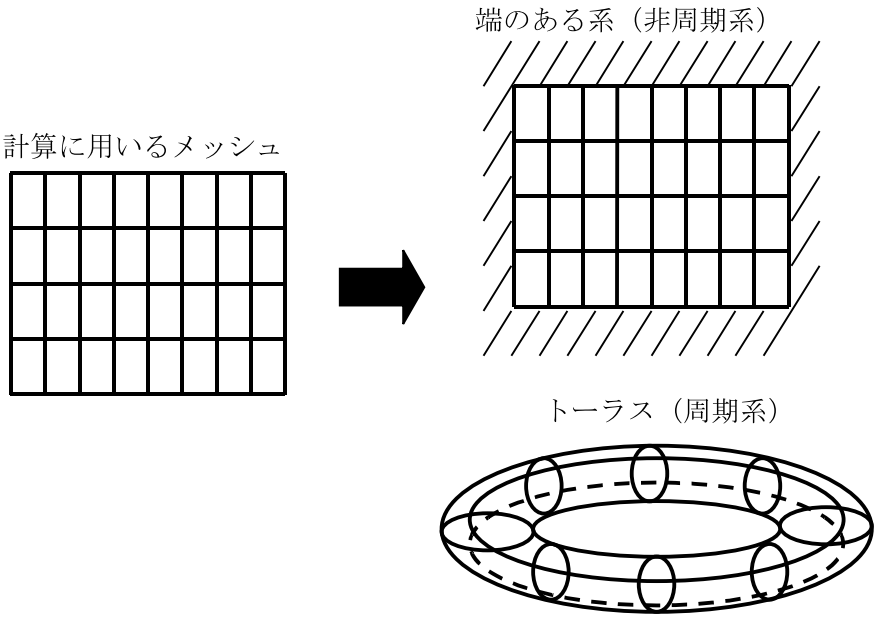


図 2.10: 幾何学的境界条件の説明

周期系と非周期系では、境界上の格子点の取り扱いが異なる。周期的な軸の両端の境界上の点は同一の点であると見なされるが、非周期系では、これらは別々の点であるとして扱われる。

2) 物理的境界条件（Physical boundary conditions）
物理的境界条件は、系の境界上における物理量の場の振る舞いを指定するための境界条件であり、

1) 物理量の 1 階導関数が指定される Neumann 境界条件
2) 物理量の値自身が指定される Dirichlet 境界条件

の 2 種が指定可能である。例えば、拡散場の固体反射壁における振る舞いは、壁に垂直方向の流れが消える条件から、1 階導関数が 0 となる Neumann 境界条件となる。また、熱伝導方程式における温度場は、熱の良導体で構成された一定温度の壁面においてはその温度が指定された Dirichlet 境界条件になる。

幾何学的境界条件と物理的境界条件の可能な組み合わせが、表 2.1 に示されている。

表 2.1: 可能な境界条件の組み合わせ

幾何学的境界条件	物理的境界条件
非周期	Neumann Dirichlet
周期	周期

経路積分の計算においては、セグメントが貫入できない固体壁は、境界値が 0 の Dirichlet 条件が適用される。一方、セグメント濃度場の時間発展方程式においては、この固体壁は、濃度場の傾きが 0 の Neumann 境界として扱われる。従って、同じ幾何学的境界条件で指定される系であっても、経路積分とセグメント濃度場は異なる物理的境界条件に従うことがあり得る。

規則メッシュと、直方メッシュは、表 2.1 の全ての組み合わせが可能であるが、極座標メッシュおよび円筒座標メッシュでは、動径方向の幾何学的境界条件としては非周期のみが許され、さらに動径方向の原点における物理的境界条件は Neumann 境界条件のみが許される。

2.9.5 離散化ラプラス演算子

2.9.2 節で導入した離散化ラプラス演算子 $\mathcal{L}$ の具体的な形は、メッシュの種類に依存するばかりでなく、ある特定のメッシュに対しても、幾つかの異なる形式が存在する。3 次元空間中のスカラー場 は、3 個の整数の組 (i,j,k) で指定されるメッシュ点上での値 $A(i,j,k)$ によって表現される。一般に、メッシュ点 (i,j,k) において、スカラー場 $A(\mathbf{r})$ に離散化ラプラス演算子を作用させた値 $\mathcal{L}A(\mathbf{r})$ は、メッシュ点 (i,j,k) の近傍の最大 27 個のメッシュ点における $A(\mathbf{r})$ の値を用いて、

$$\mathcal{L}A(i,j,k) = \sum_{l,m,n=0 \text{ or } \pm 1} C(l,m,n)A(i+l,j+m,k+n) \tag{2.97}$$

のように書かれる。ここで、$C(l,m,n)$ は、メッシュの種類およびラプラス演算子の離散化の方法に依存する係数であり、各軸方向のメッシュ幅がすべて等しい規則メッシュの場合以外は、一般に場所 (i,j,k) にも依存する量である。以下、係数 $C(l,m,n)$ の具体的数値を各種の空間メッシュについて列挙する。

1) 規則メッシュの場合
このメッシュでは、離散化ラプラシアンとして次の 4 つの型を用意している。(メッシュ幅を、仮に、すべて同一の値 Δx とする)

   i) 1NN-P 型： x, y, z の各軸方向のそれぞれについて、連続 3 個のメッシュ点を用いた中心差分で 2 階微分を近似する方法で、3 次元では最近接の 7 個のメッシュ点で定義する。1 次元、2 次元の離散化ラプラス演算子は、3 次元からの正射影になっている。

   ii) 2NN-NP 型： 2 次元と 3 次元のラプラス演算子を第 2 近接まで用いて定義することにより、等方性がよりよくなるようにする方法。1 次元の場合は上の 1NN-P 型と同じで、高次元からの正射影になっているが、2 次元の場合は 3 次元からの正射影とは一致しない。

   iii) 2NN-P 型：これも最近接と第 2 近接を用いる方法だが、2 次元の離散化ラプラス演算子を 3 次元からの正射影で定義している。1 次元、3 次元は 2NN-NP 型と同じものである。

   iv) 3NN-P 型： 第 3 近接までの近接 27 点を用いて 3 次元でより等方性がよくなるようにしている。2 次元、1 次元は 3 次元からの正射影になっている。

   表 2.2 に、上記の各場合について係数の具体的数値を記す。ただし表の中で Center は中心のメッシュ点、NN, NNN, NNNN はそれぞれ最近接 (nearest neighbor)、第 2 近接 (next nearest neighbor) および第 3 近接 (next-next nearest neighbor) の格子点を意味している。

2) 直方メッシュの場合
一般にメッシュ幅が場所ごとに異なるので、係数 $C(l,m,n)$ も i, j, k に依存する。直方メッシュでは i, j, k は、それぞれ x, y, z 方向の座標（それぞれ x_i, y_j, z_k とする）に対応している。現在のところ、このメッシュでは、x, y, z の各軸方向について連続 3 個のメッシュ点を用いた中心差分で 2 階微分を近似する方法が用意されている。この場合、メッシュ点 (i,j,k) における $\mathcal{L}A(\mathbf{r})$ の値は次のように書かれる。

$$\mathcal{L}A(i,j,k) = \sum_{l^2+m^2+n^2=0 \text{ or } 1} C(i,j,k;l,m,n)A(i+l,j+m,k+n) \tag{2.98}$$

   与えられたメッシュ点 (i,j,k) のまわりのメッシュ幅を

$$\Delta x^{(+)} \equiv x_{i+1} - x_i, \quad \Delta x^{(-)} \equiv x_i - x_{i-1}, \quad \Delta x \equiv (x_{i+1} - x_{i-1})/2,$$

表 2.2: 等方的な規則メッシュ上で定義された各種の離散化ラプラス演算子の具体的数値

型		Center ($\times\Delta x^{-2}$)	NN ($\times\Delta x^{-2}$)	NNN ($\times\Delta x^{-2}$)	NNNN ($\times\Delta x^{-2}$)
1NN-P	1D	-2	1	0	0
	2D	-4	1	0	0
	3D	-6	1	0	0
2NN-NP	1D	-2	1	0	0
	2D	-3	$1/2$	$1/4$	0
	3D	$-9/2$	$1/2$	$1/8$	0
2NN-P	1D	-2	1	0	0
	2D	$-7/2$	$3/4$	$1/8$	0
	3D	$-9/2$	$1/2$	$1/8$	0
3NN-P	1D	-2	1	0	0
	2D	$-34/11$	$6/11$	$5/22$	0
	3D	$-40/11$	$3/11$	$3/22$	$1/22$

$$\Delta y^{(+)} \equiv y_{j+1} - y_j, \quad \Delta y^{(-)} \equiv y_j - y_{j-1}, \quad \Delta y \equiv (y_{j+1} - y_{j-1})/2,$$
$$\Delta z^{(+)} \equiv z_{k+1} - z_k, \quad \Delta z^{(-)} \equiv z_k - z_{k-1}, \quad \Delta z \equiv (z_{k+1} - z_{k-1})/2,$$

と表すと、係数 $C(i,j,k;l,m,n)$ の値は表 2.3 のようになる。1 次元、2 次元の場合は 3 次元からの正射影になっている。この方法は、メッシュ幅を場所と軸方向に依らないように選ぶと、規則メッシュの 1NN-P に帰着する。

表 2.3: 直方メッシュ上で定義された離散化ラプラス演算子の具体的数値

(l,m,n)	$C(i,j,k;l,m,n)$
$(1,0,0)$	$1/(\Delta x^{(+)}\Delta x)$
$(-1,0,0)$	$1/(\Delta x^{(-)}\Delta x)$
$(0,1,0)$	$1/(\Delta y^{(+)}\Delta y)$
$(0,-1,0)$	$1/(\Delta y^{(-)}\Delta y)$
$(0,0,1)$	$1/(\Delta z^{(+)}\Delta z)$
$(0,0,-1)$	$1/(\Delta z^{(-)}\Delta z)$
$(0,0,0)$	$-2/(\Delta x^{(+)}\Delta x^{(-)}) - 2/(\Delta y^{(+)}\Delta y^{(-)}) - 2/(\Delta z^{(+)}\Delta z^{(-)})$

3) 極座標および円筒座標メッシュの場合

i) 極座標メッシュ（球対称）および 1 次元円筒座標メッシュ（2 次元回転対称）の場合
この場合には、本質的に動径座標 r だけの 1 次元系であり、差分化の方法として 1NN-P 型が適用される。

ii) 2 次元円筒座標メッシュの場合
動径座標 r と回転軸方向の座標 h の 2 次元系であり、現バージョンでは 2NN-NP 型の差分化法が使用されている。

2.9.6 グラフト鎖の扱い

メッシュ上の任意の点あるいは領域に高分子鎖中の任意の結合点（junction）をグラフトすることができる。セグメント $s = 0$ が点 $\mathbf{r}$ にグラフトされている部分鎖の経路積分は、(2.23) 式で $q_0(\mathbf{r}_0) = \delta(\mathbf{r}_0 - \mathbf{r})$ とすることにより計算される。領域でグラフトする場合は、$q_0(\mathbf{r}_0)$ をその領域内で値が 1 となるシータ関数とする。このやり方で領域にグラフトした場合、指定された結合点は、その領域内のみに存在できることになる。一般にこの結合点はグラフトされた領域内で非一様に再分布することができる。つまり、指定された領域内でグラフト点が移動し平衡分布した状態になる。SUSHI では入力時にグラフトしたい結合点のインデックスとメッシュの座標を指定する。詳細は第 5 章を参照のこと。

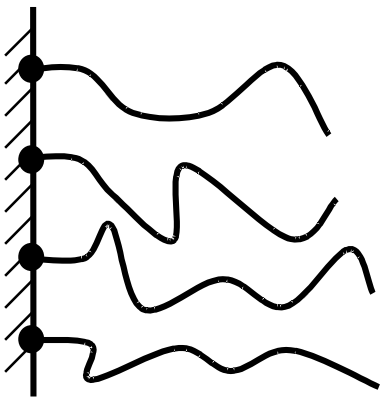


図 2.11: グラフト鎖のモデル

2.9.7 外場

SUSHI では、固体壁面と高分子セグメントの間の近距離の相互作用の効果を、K 種セグメントと壁 (S) の間の相互作用パラメタ χ_{KS} を用いて指定することができる。この相互作用パラメタは、壁面の最近接メッシュ点における自己無撞着場 $V_K(\mathbf{r})$ に値を足し合わせることで導入される。つまり、

$$V_K(\mathbf{r}) = W'_K(\mathbf{r}) - \mu(\mathbf{r}) \tag{2.99}$$

$$W'_K(\mathbf{r}) = W_K(\mathbf{r}) + \chi_{KS}\phi(\mathbf{r}')/\mathcal{Y}(\mathbf{r}'), \tag{2.100}$$

ここで、$\mathbf{r}'$ 壁面の最近接メッシュ点位置であり、$\mathcal{Y}(\mathbf{r})$ は壁面の最近接メッシュのヤコビアンである。

2.9.8 アンサンブルとセグメント濃度場の計算手法

SUSHI では、経路積分を計算する際の拘束条件として、

1) 温度が一定で、系内の分子（鎖および溶媒）の総数が一定のカノニカル集団

2) 温度が一定で、系が一定の濃度分布を持った粒子源と接した状態にあるグランド・カノニカル集団

の 2 種類の拘束条件のもとでの計算を提供している。これら 2 種の統計集団の計算上の相違点は、経路積分からセグメント濃度分布を計算する際の規格化条件に現れる。これらの規格化因子の計算方法は (2.10) (2.11) 式あるいは (2.19) (2.20) 式に書かれている。セグメント比体積 r_K を導入した場合のセグメント濃度分布（(2.18) 式と規格化因子の式を再掲しておく（セグメント比体積はグランドカノニカル集団の場合の規格化因子にのみあらわに含まれる）。セグメント濃度分布を計算する式は

$$\phi_r^{(p)}(\mathbf{r}) = C^{(p)} \sum_i \int d\mathbf{r}_0 \int d\mathbf{r}_N q_0(\mathbf{r}_0) Q_K(0, \mathbf{r}_0; i, \mathbf{r}) Q_K(i, \mathbf{r}; N, \mathbf{r}_N) q_N(\mathbf{r}_N) \tag{2.101}$$

であり、カノニカル集団における規格化因子は、

$$C^{(p)} = \frac{M^{(p)}}{\int d\mathbf{r}_0 \int d\mathbf{r}_N q_0(\mathbf{r}_0) Q_K(0,\mathbf{r}_0;N,\mathbf{r}_N) q_N(\mathbf{r}_N)} \tag{2.102}$$

グランドカノニカル集団における規格化因子は、

$$C^{(p)} = \frac{\phi_p^{(\mathrm{bulk})}}{\sum_{r''} r_{K_{r''}^{(p)}} N_{r''}^{(p)}} \exp\Big[\sum_{r'} r_{K_{r'}^{(p)}} N_{r'}^{(p)} W_{K_{r'}^{(p)}}^{(\mathrm{bulk})}\Big] \tag{2.103}$$

となる。

SUSHI では、各鎖の種類ごとに、カノニカル集団あるいはグランド・カノニカル集団のいずれに従うかを指定できる。例えば、固体壁面にグラフトされた高分子ブラシを粒子源と接した溶媒に浸したような場合には、グラフト高分子はカノニカル集団の方法を用い、一方で溶媒にはグランド・カノニカル集団の方法を用いると言うように、系の状態に応じて 2 種の統計集団を使い分ける必要がある。

2.9.9 静的計算

静的計算は、拘束条件が局所的非圧縮条件 (2.38) で与えられる場合に、この条件式ならびに拘束力を (2.39) で与えたときの一連の方程式 (2.1) (2.15) (2.16) (2.17) を同時に満たす自己無撞着解 $\{V_K(\mathbf{r})\}$ ならびに $\{\phi_K(\mathbf{r})\}$ をみつけるための手続きである。SUSHI では、次に述べる繰り返し法を用いて、数値的に解を求める。

まず、$V_K(\mathbf{r})$ の適当な初期分布を用いて、(2.96) 式より経路積分を計算する。さらに (2.101) を使って、セグメント濃度分布を計算する。C_K はカノニカルおよびグランドカノニカル集団に対してそれぞれ (2.102) および (2.103) で与えられる。次にこの $\phi_K(\mathbf{r})$ を用いて (2.37) 式の $W_K(\mathbf{r})$ と $\mu_K(\mathbf{r})$ を以下のように更新する。($W_K(\mathbf{r})$ の初期値には 0 が与えられる。)

$$W_K(\mathbf{r}) \quad \longrightarrow \quad W_K(\mathbf{r}) + \mathrm{constW} \times \Big(\sum_{K'} \chi_{KK'}\phi_{K'}(\mathbf{r}) - W_K(\mathbf{r})\Big) \tag{2.104}$$

$$\mu_K(\mathbf{r}) \quad \longrightarrow \quad \begin{cases} \mu_A(\mathbf{r}) - \mathrm{constV} \times \Big(1 - \sum_{K'} \phi_{K'}(\mathbf{r})\Big) & \text{for} \quad K = A \\ \mu_K(\mathbf{r}) - \mathrm{constV} \times \Big(\mu_K(\mathbf{r}) - \mu_A(\mathbf{r})\Big) & \text{for} \quad K \neq A \end{cases} \tag{2.105}$$

(2.105) 式において、セグメント種 $K = A, B, C, \cdots$ のうち、最初のセグメント種 A に対しては第 1 式を用い、2 番目以降のセグメント種に対しては第 2 式を用いる。これらの式に現れるパラメタ、constW、constV は、0 より大きく 1 より小さい適当な定数であり、SUSHI の入力パラメータとして与えられる（計算が不安定な場合、これらの定数の値を小さくすると安定になる場合がある）。更新された $\mu_K(\mathbf{r})$ と $W_K(\mathbf{r})$ から得られる自己無撞着場 $V_K(\mathbf{r}) = W_K(\mathbf{r}) - \mu_K(\mathbf{r})$ を用いて、経路積分以下の手順を繰り返す。更新前後の $V_K(\mathbf{r})$ と $W_K(\mathbf{r})$ の差の最大が許容値以下になり、かつ非圧縮である条件 $\sum_K \phi_K(\mathbf{r}) = 1$ が許容値以下で満たされたとき、誤差の範囲内で平衡条件を満たしている。この許容値も SUSHI の入力パラメータとして用意されている。

2.9.10 動的計算

動的計算では、2.8 節で述べたように、セグメント濃度場の時間発展方程式 (2.49)

$$\frac{\partial}{\partial t}\phi_K(\mathbf{r},t) = \nabla \cdot [L_K(\mathbf{r},t)\nabla\{\mu_K(\mathbf{r},t) + \lambda(\mathbf{r},t)\}] + \xi_K(\mathbf{r},t) \tag{2.106}$$

と、$\{\phi_K(\mathbf{r})\}$ を拘束した下での自己無撞着方程式を同時に解くことにより、鎖のコンフォメーションを正しく考慮したセグメントの拡散ダイナミクスを計算する。$\{\phi_K(\mathbf{r})\}$ の初期分布はランダムノイズ（標準偏差を入力パラメータとして与える）とし、運動係数 $L_K(\mathbf{r},t)$ は以下のような選択をする。

1) 値を 1 に固定する。成分が高濃度で同程度の鎖長をもつ高分子の溶融状態に向いている。

2) $L_K(\mathbf{r},t)$ を $\phi(\mathbf{r},t)$ の依存性のあるように設定する。
セグメント種毎に定数 L_0 を適当に入力し、以下の選択を行う。
Rouse Dynamics 条件

$$L_K(\mathbf{r},t) = L_0\phi_K(\mathbf{r},t) \tag{2.107}$$

Reptation Dynamics 条件

$$L_K(\mathbf{r},t) = \frac{L_0}{N^{(total)}}\phi_K(\mathbf{r},t) \tag{2.108}$$

ここで $N^{(total)}$ は、セグメントが属する鎖の全長である。Reptation Dynamics 条件は直鎖ホモポリマーについて正当性が保証されるものである。熱揺らぎに関するノイズ項 $\xi_K(\mathbf{r},t)$ は、無視している。

時間発展の各ステップにおいて、そのときの $\{\phi_K(\mathbf{r})\}$ にセグメント濃度分布を拘束した下での自己無撞着方程式を解いて、得られた $\{\mu_K(\mathbf{r})\}$ を使って次の時間発展を順次行う。

任意の拘束された $\phi_K(\mathbf{r})$ （$\phi_K^{\rm target}(\mathbf{r})$ とする）に対応する $\mu_K(\mathbf{r})$ を求める。この手続きは「セグメント濃度場を拘束した平衡化」であり、繰り返し法によって行われる。まず、$\phi_K^{\rm target}(\mathbf{r})$ を (2.33) 式に代入して得られる $W_K(\mathbf{r})$ を $W_K^{\rm target}$ とする。0 で初期化した $W_K(\mathbf{r})$ と $\mu_K(\mathbf{r})$ を

$$W_K(\mathbf{r}) \longrightarrow W_K(\mathbf{r}) + {\rm constW} \times \big(W_K^{\rm target}(\mathbf{r}) - W_K(\mathbf{r})\big) \tag{2.109}$$

および

$$\mu_K(\mathbf{r}) \longrightarrow \mu_K(\mathbf{r}) + {\rm constV} \times \big(\phi_K^{\rm target}(\mathbf{r}) - \phi_K(\mathbf{r})\big) \tag{2.110}$$

で更新していく。(2.110) の $\phi_K(\mathbf{r})$ は、更新前の $\mu_K(\mathbf{r})$ と $W_K(\mathbf{r})$ から得られる自己無撞着場 $V_K(\mathbf{r}) = W_K(\mathbf{r}) - \mu_K(\mathbf{r})$ の下で (2.96) より経路積分 $Q_K(s,\mathbf{r})$ を計算し、さらに (2.101) を用いて計算する。これを適当な収束条件を満たすまで繰り返し、セグメント濃度場を $\phi_K^{\rm target}(\mathbf{r})$ に拘束したときのセグメント化学ポテンシャル $\mu_K(\mathbf{r})$ を求める。動的計算中で行っている「セグメント濃度場を拘束した平衡化」のための繰り返し法における収束判定の方法は、静的計算における繰り返し法のときとやや異なり、更新前後の $W_K(\mathbf{r})$ と $\phi_K(\mathbf{r})$ の差の最大で収束を判定している。これらの値が許容値以下になったとき、誤差の範囲内で $\mu_K(\mathbf{r})$ が求められている。この許容値も SUSHI の入力パラメータとして用意されている。$\phi_K(\mathbf{r})$ の値を相対誤差で判定するやり方も用意されている。

2.9.11 自由エネルギー

静的計算終了時、および動的計算の各時間ステップにおいて、(2.30) 式

$$\begin{aligned}\mathcal{F}\big[\{\phi_K\},\{V_K\}\big] &= -k_{\rm B}T\sum_p M_p \ln \mathcal{Z}_p + \mathcal{W}\big[\{\phi_K\}\big] \\ &\quad -\sum_K \int d\mathbf{r} V_K(\mathbf{r})\phi_K(\mathbf{r}) \\ &\quad +k_{\rm B}T\sum_p M_p \ln M_p\end{aligned} \tag{2.111}$$

の自由エネルギーの値が出力される。また、グランドカノニカル系の計算で、バルクのセグメント濃度が既知の場合、過剰自由エネルギー

$$\mathcal{F}_{\rm exess} = \mathcal{F} - \mathcal{F}^{\rm (bulk)} - \sum_p \mu_p(M_p - M_p^{\rm (bulk)}) \tag{2.112}$$

が計算される。ここで、$\mathcal{F}^{\rm (bulk)}$ は系全体がバルクのセグメント濃度に等しい一様な濃度になっている場合の自由エネルギー、$M_p^{\rm (bulk)}$ は同じくそのときの p 種の鎖の総数、μ_p は p 種の鎖の化学ポテンシャルである。

2.9.12 多分散性高分子を効率的に計算する方法

鎖長が僅かずつ異なる高分子鎖を鎖長分布に基づく体積分率とともに別々の成分として用意すれば、多分散性高分子からなる系を SUSHI で扱うことは原理的に可能である。しかし、このままだと成分の数だけ独立に経路積分の計算を行うために、計算時間および必要なメモリ量が成分数の増加とともに増大する。従って、あまりに多くの成分を含めることは実際上は不可能である。

ただし、多分散性高分子がホモポリマーからなり、かつ、静的計算を行う場合に限っては、それぞれの鎖の経路積分を共有化することが可能であり、多くの成分を扱うときに生じる上記の問題を回避することができる。SUSHI では、一連の多分散性高分子を構成する成分の経路積分として、最大鎖長成分のそれを使うようにするオプションを入力 UDF で定義することができる。これによって、各成分の鎖長と体積分率を独立に入力する手間を無視すれば（将来はこの作業を分布関数や M_w, M_n などの入力だけから自動的に行えるようにする必要がある）、事実上無制限に多くの成分を扱うことができる。また、A-B ブロックコポリマーでブロック比の異なるポリマーの混合系の静的計算を行う場合、自由端からの経路積分は共有することが可能である。このような共有を行うことでも計算量を減らすことができる。SUSHI におけるこのようなオプション設定の詳細は第 5 章を参照のこと。

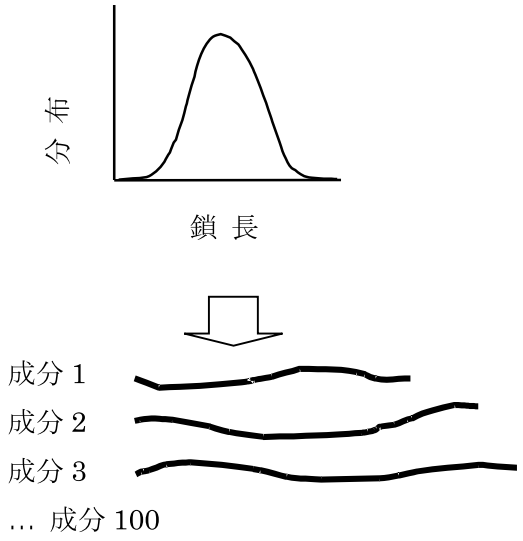


図 2.12: 多分散性高分子の成分構成

2.9.13 ドメイン領域の指定

静的な SCF 計算は、微妙な自由エネルギーの違いにより最安定構造に収束することはまず無く、種々の局所安定構造で計算が終了する場合が殆どである。よって、意図するモルフォロジーを結果として出すには、特別な操作が必要である。例えば、ジブロックコポリマーのように、既に相図が知られているようなもののモルフォロジーをシミュレートするような場合である。静的な SCF 計算で意図するモルフォロジーを結果として生成する 1 つの方法は、自己無撞着場の初期値を、狙ったモルフォロジーとなるように設定することである。SUSHI ではそのような設定も可能としている。詳細は第 5 章を参照のこと。

2.9.14 結合点の存在領域をマスクする方法

非常に濃度が低い分子がつくる構造をシミュレートする場合、SCF の計算では系内に溶解してしまうことを防ぐ手段が必要となる。例えばミセルを計算する場合等である。SUSHI では、このような手段の 1 つとして結合点（自由端も含む）の存在する領域を拘束することを可能としている。詳細は第 5 章を参照のこと。

2.9.15 化学反応を動的平均場法で計算する方法

動的平均場法 (2.49) 式の右辺に、化学反応を模擬した簡単な反応項を追加することにより種々の化学反応のシミュレーションが可能になる。以降で実際に取り扱える反応項を説明する。入力方法は第 5 章を参照のこと。

反応速度が速い場合

ラジカル重合反応のように高分子の拡散過程より十分に反応速度が速い場合、瞬時にモノマーが、ある重合度の高分子になると仮定することが可能である。これを式で表すと次式のようになる。

$$\frac{\partial \phi^{(S)}(\mathbf{r},t)}{\partial t} = \frac{\partial \phi^{(S)}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} - k\phi^{(S)}(\mathbf{r},t) \tag{2.113}$$

$$\frac{\partial \phi^{(P)}(\mathbf{r},t)}{\partial t} = \frac{\partial \phi^{(P)}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} + k\phi^{(S)}(\mathbf{r},t) \tag{2.114}$$

右辺第 1 項は、(2.49) 式の右辺で与えられる拡散による寄与である。右辺第 2 項が反応項である。k はモノマーから高分子への重合反応に対する仮想的な反応定数、$\phi^{(S)}(\mathbf{r},t)$ はモノマーの、$\phi^{(P)}(\mathbf{r},t)$ はポリマーの体積分率である。

モノマーや高分子上の活性点の反応

反応速度が高分子の拡散速度と同程度である場合を想定すると、高分子であれば高分子に、モノマーであればそれ自体の上に活性点を仮定し、活性点同志の 2 次反応により新たな分子が生成することを考えることが可能である。例えば次の図に示すような高分子末端の反応によるジブロックコポリマーの生成反応である。

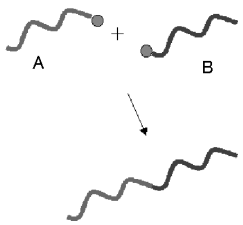


図 2.13: 高分子末端の反応によるジブロックコポリマーの生成反応の概念図

ここではカノニカルアンサンブルのみを考える。SUSHI では、第 2.2 節で説明したように、ポリマーの自由末端や部分鎖の接合部に図 2.4 で示した結合点を仮定している。ポリマー A、B 上のこのような点のどれか 1 点を活性点としよう。これらの分布を $\varphi^{(A)}_{reactive}(\mathbf{r},t)$、$\varphi^{(B)}_{reactive}(\mathbf{r},t)$ とする。これらの点に次式のような 2 次反応による時間変化を考える。

$$\frac{\partial \varphi^{(A)}_{reactive}(\mathbf{r},t)}{\partial t} = -k\varphi^{(A)}_{reactive}(\mathbf{r},t)\varphi^{(B)}_{reactive}(\mathbf{r},t) \tag{2.115}$$

$$\frac{\partial \varphi^{(B)}_{reactive}(\mathbf{r},t)}{\partial t} = -k\varphi^{(A)}_{reactive}(\mathbf{r},t)\varphi^{(B)}_{reactive}(\mathbf{r},t)\ , \tag{2.116}$$

ここで k は反応定数である。単位時間当たりに反応で消費されるポリマー A,B の体積分率を $\phi^{(A)}_{reactive}|_k$、$\phi^{(B)}_{reactive}|_k$、また、反応物の体積分率を $\phi^{(A)}_{reactant}(\mathbf{r},t)$、$\phi^{(B)}_{reactant}(\mathbf{r},t)$、生成物内で反応物 A、B に対応する部分鎖の体積分率をそれぞれ、$\phi^{(A)}_{product}(\mathbf{r},t)$、$\phi^{(B)}_{product}(\mathbf{r},t)$ とすれば反応は次式のようになる。

$$\frac{\partial \phi^{(A)}_{reactant}(\mathbf{r},t)}{\partial t} = \frac{\partial \phi^{(A)}_{reactant}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} - \phi^{(A)}_{reactive}(\mathbf{r},t)|_k \tag{2.117}$$

$$\frac{\partial \phi^{(B)}_{reactant}(\mathbf{r},t)}{\partial t} = \frac{\partial \phi^{(B)}_{reactant}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} - \phi^{(B)}_{reactive}(\mathbf{r},t)|_k \tag{2.118}$$

$$\frac{\partial \phi^{(A)}_{product}(\mathbf{r},t)}{\partial t} = \frac{\partial \phi^{(A)}_{product}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} + \phi^{(A)}_{reactive}(\mathbf{r},t)|_k \tag{2.119}$$

$$\frac{\partial \phi^{(B)}_{product}(\mathbf{r},t)}{\partial t} = \frac{\partial \phi^{(B)}_{product}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} + \phi^{(B)}_{reactive}(\mathbf{r},t)|_k \,, \tag{2.120}$$

右辺第 1 項は、前節と同様に拡散による寄与である。上式は単純な A、B ポリマーの結合反応を表すが、反応物と生成物の構造が複数の部分鎖からなる場合は、それぞれの構造内の部分鎖の対応関係を全て考慮した部分鎖の体積分率の時間発展方程式を考えることになる。

実際の計算での、$\phi^{(A)}_{reactive}|_k$ と $\phi^{(B)}_{reactive}|_k$ の求め方を説明する。SCF 計算においては、図 2.3 で説明したように、全ての部分鎖の両端は結合点として取り扱われる。これらの結合点を反応点として考えることができる。SCF 計算が終了時にこれらの点の空間分布は以下のように求められる。

$$\varphi^{(A)}_{reactive}(\mathbf{r},t) = C^{(A)}_r \prod_r \widetilde{Q}_K(N^{(A)}_r, \mathbf{r}) \tag{2.121}$$

$$\varphi^{(B)}_{reactive}(\mathbf{r},t) = C^{(B)}_r \prod_r \widetilde{Q}_K(N^{(B)}_r, \mathbf{r}) \,, \tag{2.122}$$

ここで、右辺の $\prod_r$ 以降は結合点へ連結されている部分鎖の経路積分により掛かってくる統計的重みの積を表す。$C^{(A)}_r$ と $C^{(B)}_r$ は規格化定数であり、カノニカルアンサンブルのみを考えているので、部分鎖の番号 r には依存しない (2.19) 式で求められる。次にこの値を用いて反応で消費される A- および B-ホモポリマーの量を求める。これらを $M^{(A)}_{reactive}$ と $M^{(B)}_{reactive}$ とすれば、これらは、反応前の A- および B-ホモポリマーの全量に、反応により失われる活性点の分率を掛ければ求められるから次のようになる。

$$M^{(A)}_{reactive} = M^{(A)} \frac{\int d\mathbf{r} k \varphi^{(A)}_{reactive}(\mathbf{r},t)\varphi^{(B)}_{reactive}(\mathbf{r},t)}{\int d\mathbf{r} \varphi^{(A)}_{reactive}(\mathbf{r},t)} \tag{2.123}$$

$$M^{(B)}_{reactive} = M^{(B)} \frac{\int d\mathbf{r} k \varphi^{(A)}_{reactive}(\mathbf{r},t)\varphi^{(B)}_{reactive}(\mathbf{r},t)}{\int d\mathbf{r} \varphi^{(B)}_{reactive}(\mathbf{r},t)} \,, \tag{2.124}$$

ここで、$M^{(A)}$,$M^{(B)}$ は反応前の A- および B-ホモポリマーの全量である。最終的に (2.18) 式と (2.19) 式と SCF が収束時の自己無撞着場を用いることにより、$\phi^{(A)}_{reactive}(\mathbf{r},t)|_k$ と $\phi^{(B)}_{reactive}(\mathbf{r},t)|_k$ は次の式で求められる。

$$\phi^{(A)}_{reactive}(\mathbf{r},t)|_k = C^{(A)}_{reactive} \sum_i \int d\mathbf{r}_0 \int d\mathbf{r}_N k \varphi^{(B)}_{reactive}(\mathbf{r}_0,t) q_0(\mathbf{r}_0) Q_A(0,\mathbf{r}_0;i,\mathbf{r}) Q_A(i,\mathbf{r};N,\mathbf{r}_N) q_N(\mathbf{r}_N) \tag{2.125}$$

$$C^{(A)}_{reactive} = \frac{{M^{(A)}}_{reactive}}{\int d\mathbf{r}_0 \int d\mathbf{r}_N q_0(\mathbf{r}_0) Q_A(0,\mathbf{r}_0;N,\mathbf{r}_N) q_N(\mathbf{r}_N)} \tag{2.126}$$

$$\phi^{(B)}_{reactive}(\mathbf{r},t)|_k = C^{(B)}_{reactive} \sum_i \int d\mathbf{r}_0 \int d\mathbf{r}_N k \varphi^{(A)}_{reactive}(\mathbf{r}_0,t) q_0(\mathbf{r}_0) Q_B(0,\mathbf{r}_0;i,\mathbf{r}) Q_B(i,\mathbf{r};N,\mathbf{r}_N) q_N(\mathbf{r}_N) \tag{2.127}$$

$$C^{(B)}_{reactive} = \frac{{M^{(B)}}_{reactive}}{\int d\mathbf{r}_0 \int d\mathbf{r}_N q_0(\mathbf{r}_0) Q_B(0,\mathbf{r}_0;N,\mathbf{r}_N) q_N(\mathbf{r}_N)} \,, \tag{2.128}$$

ここで $q_0(\mathbf{r}_0)$ と $q_N(\mathbf{r}_N)$ は 活性な末端 $(i=0)$ と活性でない他端 $(i=N)$ の統計的な重みである。ホモポリマー A の活性末端はホモポリマー B の活性末端の影響 $k\varphi^{(B)}_{reactive}(\mathbf{r}_0,t)$ を受け、逆にホモポリマー B の活性末端はホモポリマー A の活性末端の影響 $k\varphi^{(A)}_{reactive}(\mathbf{r}_0,t)$ を受ける。自分自身の活性末端の影響はは 2 重に計算に取り込まれることがないようあらわには計算に用いられない。この簡単な例では、反応物はホモポリマーであるので両端は自由端であり、$q_0(\mathbf{r}_0)$ と $q_N(\mathbf{r}_N)$ の量は 1 である。もし、複雑な構造を持つポリマーが反応物で両端が自由端でない場合は接続している部分鎖の影響を受けるので、その値は 1 以下となる。

ここで計算している反応で消費される高分子の体積分率の分布は、活性末端により拘束を受けたホモポリマーの体積分率である。反応後はブロックコポリマーになるので、その体積分率の分布は安定な状態ではなくなる。よって、動力学の進行によりその分布は変化をする。

グラフト反応

上節で述べた手法の応用として、自由鎖が固体壁へグラフトする反応をシミュレートすることが可能である。高分子 A が高分子溶液などから固体壁面へグラフトし、壁面へグラフトした高分子 G になるとする。高分子 A の壁上の活性末端点の密度を $\varphi^{(A)}_{reactive}(\mathbf{r}_w,t)$、また、高分子 G の末端の壁での密度を $\varphi^{(G)}_{reactive}(\mathbf{r}_w,t)$ とする。ここで、$\mathbf{r}_w$ は壁の位置ベクトルである。反応定数を k とすれば、グラフト反応の式は次のようになる。

$$\frac{\partial\varphi^{(A)}_{reactive}(\mathbf{r}_w,t)}{\partial t} = -k\varphi^{(A)}_{reactive}(\mathbf{r}_w,t) \tag{2.129}$$

$$\frac{\partial\varphi^{(G)}_{reactive}(\mathbf{r}_w,t)}{\partial t} = k\varphi^{(A)}_{reactive}(\mathbf{r}_w,t) \tag{2.130}$$

反応によるポリマーの体積変化のモデル式は次のようになる。

$$\frac{\partial\phi^{(A)}(\mathbf{r},t)}{\partial t} = \frac{\partial\phi^{(A)}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} - \phi^{(A)}_{active}(\mathbf{r},t)|_k \tag{2.131}$$

$$\frac{\partial\phi^{(G)}(\mathbf{r},t)}{\partial t} = \frac{\partial\phi^{(G)}(\mathbf{r},t)}{\partial t}\Big|_{diffusion} + \phi^{(A)}_{active}(\mathbf{r},t)|_k \tag{2.132}$$

ここで、$\phi^{(A)}(\mathbf{r},t)$、$\phi^{(B)}(\mathbf{r},t)$ はそれぞれ高分子 A,G の体積分率であり、$\phi^{(A)}_{active}(\mathbf{r},t)|_k$ は単位時間あたり反応により消費され高分子 G になる高分子 A の体積分率である。この反応により消費される高分子の体積分率の分布を計算しよう。それには前節で述べたのと同様にして活性点の密度を以下の式により計算する。

$$\varphi^{(A)}_{reactive}(\mathbf{r}_w,t) = C^{(A)}_r \prod_r \widetilde{Q}_K(N^{(A)}_r,\mathbf{r}_w)\,, \tag{2.133}$$

ここで、 $C^{(A)}_r$ は (2.19) 式で定義された規格化定数である。反応により単位時間に消費される高分子 A の量は、次の式となる。

$$M^{(A)}_{reactive} = M^{(A)}\frac{\int d\mathbf{r}k\delta(\mathbf{r}-\mathbf{r}_w)\varphi^{(A)}_{reactive}(\mathbf{r}_w,t)}{\int d\mathbf{r}\varphi^{(A)}_{reactive}(\mathbf{r}_w,t)} \tag{2.134}$$

反応で消費される高分子 A の体積分率の分布は次の式で求めることができる。

$$\phi^{(A)}_{reactive}(\mathbf{r},t)|_k = C^{(A)}_{reactive}\sum_i\int d\mathbf{r}_0\int d\mathbf{r}_N k\delta(\mathbf{r}-\mathbf{r}_w)q_0(\mathbf{r}_0)Q_A(0,\mathbf{r}_0;i,\mathbf{r})Q_A(i,\mathbf{r};N,\mathbf{r}_N)q_N(\mathbf{r}_N)\,, \tag{2.135}$$

ここで、 $q_0(\mathbf{r}_0)$ と $q_N(\mathbf{r}_N)$ は鎖両端の統計的な重みであり、この場合はホモポリマーであるのでどちらもその値は 1 である。規格化定数 $C^{(A)}_{reactive}$ は $M^{(A)}_{reactive}$ を用いて (2.126) 式にて計算される。

2.9.16 強高分子電解質

強高分子電解質とは、電離が可能な官能基をもつ高分子の溶液で官能基が殆ど電離している状態のものを言う。よって、高分子上の電荷は変化しないものとして取り扱うことができるので、セグメント種毎に変化しない電荷密度を仮定して系の静電ポテンシャルを計算し、その影響を自己無撞着場に加味する方法によりシミュレーションが可能である。系の誘電率を ϵ_0、K-種セグメントの比誘電率を ϵ_K とする。セグメントの誘電率に加成性が成り立つと仮定すれば、局所的な誘電率は次式となる。

$$\epsilon(\mathbf{r}) = \epsilon_0\sum_K \epsilon_K\phi_K(\mathbf{r}) \tag{2.136}$$

次に K 種のセグメントの単位体積あたりの電荷密度を ρ_K とする。全電荷の系内の分布は次式にて計算できる。

$$\rho(\mathbf{r}) = \sum_K \rho_K\phi_K(\mathbf{r}) \tag{2.137}$$

そして、静電ポテンシャル $U(\mathbf{r})$ は次のポアソン方程式で計算できる。

$$\nabla\epsilon(\mathbf{r})\nabla U(\mathbf{r}) = -\rho(\mathbf{r}) \tag{2.138}$$

静電ポテンシャルエネルギーは $\mathcal{W}_e$ は次のように求められる。

$$\mathcal{W}_e = \frac{1}{2}\int d\mathbf{r}\int d\mathbf{r}'\frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{4\pi\epsilon|\mathbf{r}-\mathbf{r}'|} = \frac{1}{2}\int d\mathbf{r}U(\mathbf{r})\rho(\mathbf{r}) \tag{2.139}$$

単位体積あたりの K 種セグメントが静電ポテンシャルから受ける外場としての影響は $U(\mathbf{r})\rho_K$ である。よって、$U(\mathbf{r})\rho_K$ を SCF における自己無撞着場 (2.1) 式に加えるだけで、静電場の影響を SCF 計算に取り込むことができる。

2.9.17 SCF モンテカルロ法

SCF 法の一つの限界として認識されているのは希薄溶液の計算ができないことである。何故なら希薄溶液中に存在する孤立高分子は平均場近似では記述できない。つまり、高分子希薄溶液を SCF 法で計算しても、高分子の体積分率が均一に系内に分布した結果しか戻ってこないが、希薄溶液中では孤立した高分子の描像が得られるべきである。そこで、希薄溶液を SCF 法で取り扱うには何らかの手法をを導入する必要がある。１つの方法は 2.9.14 節で述べたマスク法を導入することである。我々はマスク法を発展させてモンテカルロ法の導入を行った。この方法は適当に選択された高分子の自由端や結合点をある位置に固定し、系の自由エネルギーを評価しながらモンテカルロ法で選択された自由端や結合点をモンテカルロ法の方法で動かす方法である。この SCF モンテカルロ法のアルゴリズムは次のようになる。

1) まず任意の高分子上の自由端点や結合点を選択し、系を記述するメッシュ上のある一点に配置する。この点を固定点と呼ぶこととする。モンテカルロ法のステップを０とする。

2) 上記の拘束条件の下に SCF 計算を行い静的な平衡状態を求める。この状態での系全体の自由エネルギーを A^i とする。

3) 固定点を乱数を用いて移動する。つまり、固定点、もしくは固定点の最近接メッシュ点を乱数を用いて選択し、固定点を動かさないか、もしくは選ばれた最近接メッシュ点へ移動する。もし、選択された点への移動が高分子の構造上不可能な場合、例えば、高分子の両端を固定点として選択したが、移動後のそれらの間の距離が高分子鎖の長さを超えた場合は、固定点の位置を最初の状態にもどし、受け入れられる状態が出現するまで固定点の移動を試みる。

4) 上記の拘束条件の下に SCF 計算を行い静的な平衡状態を求める。この状態での系全体の自由エネルギーを A^{i+1} とする。

5) $dA = A^{i+1} - A^i$ を計算する。

6) $dA < 0.$ の場合、i+1 の状態を保存する。
それ以外の場合、$p = exp(-dA/k_BT)$ を計算し、０から１までの乱数 $rand$ を発生させる。
$p > rand$ の場合 i+1 の状態を保存する。それ以外の場合 i+1 の状態を棄却し、状態を i の状態に戻す。
ここで k_B はボルツマン定数、T は絶対温度である。

7) 3) から計算を繰り返す。

2.9.18 静的および動的な構造最適化

SCF 計算により静的な平衡状態を計算する上で重要となるのは自己無撞着場の初期値である。任意の自己無撞着場の初期値を与えれば準安定な構造を速やかに計算することができる。$\mathcal{F}$ を系の自由エネルギー、V を系の体積であるとすれば、得られた準安定な構造を初期構造として, 次の式により系の自由エネルギー密度を最低にするようにシステムサイズを最適化できる。

$$\frac{\partial(\mathcal{F}/V)}{\partial X_i} = 0 \tag{2.140}$$

ここで、X_i $(i = x, y, z)$ はシステムの各辺の長さである。左辺の微分項は数値解析的に計算することが可能であり、関数の極小値を求める問題としてこの式は解くことができる。

動的な平均場法では、K 種セグメント密度 $\phi_K(\mathbf{r})$ の時間発展方程式である (2.49) 式を解くと同時に、自由エネルギー密度を低くするように、次のシステムサイズの時間発展方程式を解くと、系は自由エネルギー密度を最低とする真の安定構造に近づいてゆく。

$$\frac{\partial X_i}{\partial t} = -Q_i \frac{\partial(\mathcal{F}/V)}{\partial X_i} \tag{2.141}$$

ここで、Q_i $(i = x, y, z)$ は任意の正の定数である。この計算方法は、擬似的に結晶と考えられる高分子のミクロ相分離構造の最安定構造を得るのに効果的な方法である。ただし、この方法では体積は保存されない。体積の大きな変化を防ぐには、自由エネルギー密度に圧縮率 κ を用いた弾性エネルギーを加えればよい。つまり自由エネルギー密度 $\mathcal{F}/V$ を次の式に変更する。

$$\frac{\mathcal{F}_c}{V} = \frac{\mathcal{F}}{V} + \frac{1}{2\kappa}\frac{(V - V_0)^2}{V_0^2} \tag{2.142}$$

ここで、V_0 は系の初期体積である。

2.9.19 まとめ

本章では、計算原理を実現するための方法について示した。最後にこの節では、これらの相互の関係と位置づけを明らかにするため、まとめを行なう。

まず Self Consistent Field 法について示す。図 2.14 に計算スキームを示す。Self Consistent Field 法は、静的な平衡状態についての最も自由エネルギーが安定な自己無撞着なセットである濃度分布 (density)、場のポテンシャル ($V_K(\mathbf{r})$)、及び統計重率 ($Q_K(s,\mathbf{r})$) を求めることである。これを実現化するために、式 (2.7) より自己無撞着場 $V_K(\mathbf{r})$ を用いて、それぞれの位置 $\mathbf{r}$ によって決まる経路積分 $Q_K(s,\mathbf{r})$ が求められ、式 (2.9) より $Q_K(s,\mathbf{r})$ を用いて $\phi_K(\mathbf{r})$ が求められ、式 (2.104) と (2.105) により、$\phi_K(\mathbf{r})$ より $V_K(\mathbf{r})$ を求めることができる。計算機上ではトライアルな $V_K(\mathbf{r})$ を初期値として、これらの式を順に解き、$\phi_K(\mathbf{r})$ が先に求めたものとある誤差内で一致するまで繰り返し解き、一致が見られたときに自己無撞着なセットが求められたと判定する。

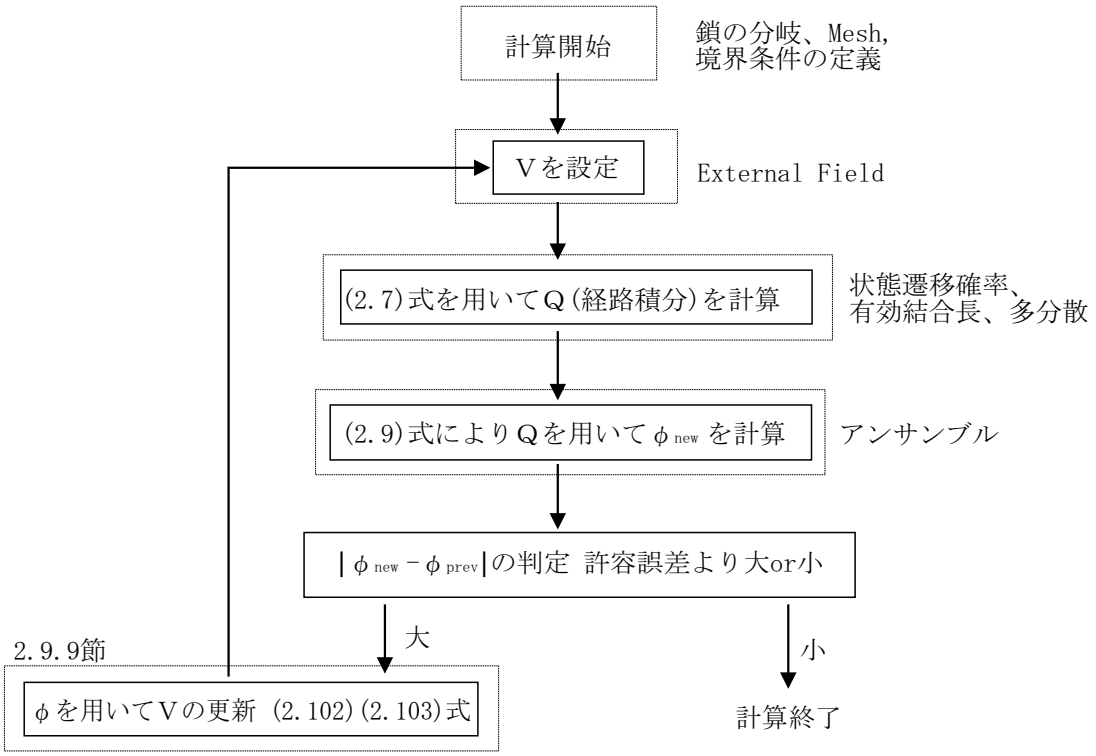


図 2.14: Self Consistent Field 法の計算の流れ図

これらの方程式を解くことで、様々な系に対して適用が行なえる。高分子鎖は、重合過程や触媒を変えることで様々な分岐構造や更にはループ構造などを作り出すことができる。これについては、鎖の構造に適した経路積分の計算法の適用が必要となる。それを実現化するために、2.9.1 節では、計算機上での鎖についての定義を示した。また、計算機内で表している空間における環境設定について、主なものとして、空間メッシュや境界条件について、2.9.3 節及び 2.9.4 節で定義した。また、複雑な構造を持った高分子鎖、例えばランダム重合された高分子鎖やテーパードポリマーなどにも適用できる自由度を持った $Q_K(s,\mathbf{r})$ の計算方法として、鎖の部分鎖に内部状態を設定し、更に内部状態間の遷移確率による計算方法を提案し (2.9.5 節)、計算に用いている。経路積分の計算のもう 1 つのパラメーターとして有効結合長があり、これと上記の状態間遷移確率が決まれば経路積分が計算できるようになる。経路積分及び濃度場の計算の際に、カノニカル、グランドカノニカルの 2 つのアンサンブルが用意されており、2.2 節および 2.9.8 節で定義している。鎖の特徴として、グラフト鎖、吸着鎖、多分散性などについてもそれぞれ 2.9.6、2.9.7、2.9.12 の各節において、我々の検討方法を示している。

また、動的平均場法では、Self Consistent Field 法を応用した方法であるが、ダイナミックに濃度場を変化させるように修正を行なっている。よってまず濃度場が定義され、それによる $V_K(\mathbf{r})$、$Q_K(s,\mathbf{r})$ と求められて

行く。更に、上記の静的 SCF の場合と比べて、ダイナミックスを取り扱うための時間発展のループが $V_K(\mathbf{r})$ の更新のループの外に表れる。

図 2.15 に動的平均場シミュレーション法における計算の流れ図を示す。

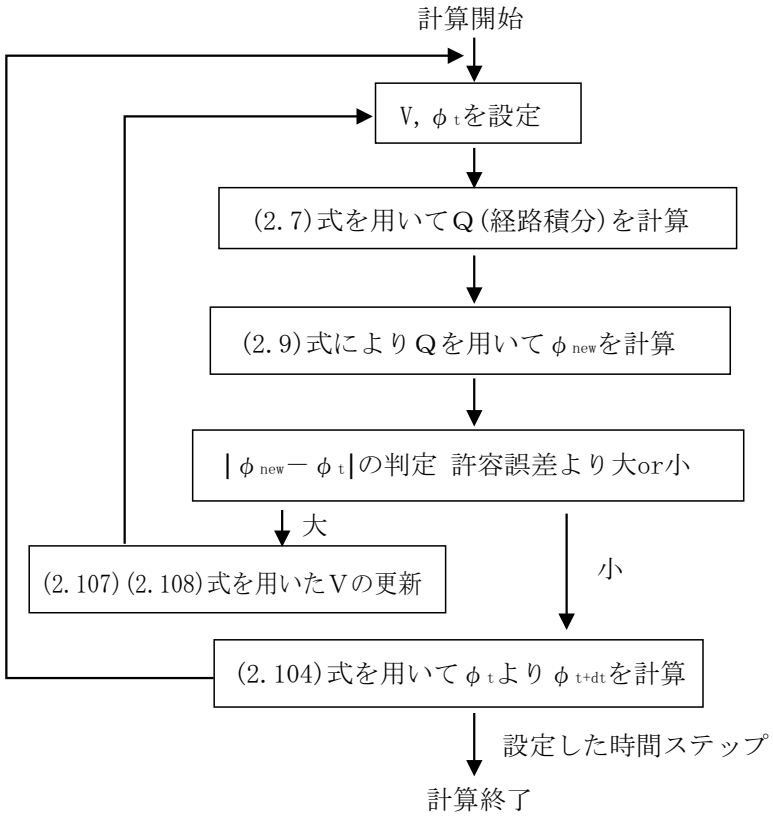


図 2.15: 動的平均場法による手続き。

動的計算に特有の変更点は、$V_K(\mathbf{r})$ などの更新の式である (2.109) (2.110) 式等に代表される。

第3章　操作入門

この章では、GOURMET を利用して SUSHI を初めて扱うユーザの導入時の障壁を低減することを目的として、他の章と異なり、基本的な操作をできるだけ簡易な表現方法を用いて説明することに努めた。

SUSHI Version 5 よりは、UDF の IO に汎用ライブラリーを用いたため、以下で説明する*_uin.udf ファイルの内容が、人間が目で見ても瞬時に理解可能な状況ではなくなったが、その点は後々改良するということでご了承頂きたい。

3.1　SUSHI の起動

SUSHI はコンソールからコマンドを打ち込んで起動させるプログラムです。Windows であれば DOS 窓から、UNIX 系であれば種々の Shell を用いてコマンドによりテキストファイルを入力して利用します。SUSHI は GUI を実装しておりませんが、UDF 入力ファイルを利用し、GOURMET に登録して使用することにより GUI が備わったプログラムのように利用することが可能です。

SUSHI は設計上、種々の入力インターフェースに対応できるようになっており、現在 UDF フォーマットと SEED フォーマットによる入力が可能です。UDF フォーマットは GOURMET から起動するのに用います。冗長な入力ファイルとなりますが、データ構造が明白で入力ミスが少なくなります。UDF フォーマットの詳細については UDF マニュアルを参照してください。SEED フォーマットは入力が簡単ですが、キーワードを自分で打ち込む必要があります。SEED ファイルは UNIX 形式（行末が LF で終わるもの）を使用してください。

SUSHI 起動の方法は第 5 章をご覧ください。

入力ファイルはサンプルファイルをコピーして一部を変更して作成するのが一番簡単です。何もないところからファイルを作成するには、UDF 入力の場合は、def_udf/SUSHIInput.udf を GOURMET で読み込んでデータを入力する方法が簡単です。もちろんエディターを利用して作成することも可能です。

3.2　１次元での計算例

SUSHI/sample 以下のサンプルファイルを使用して入力方法を説明します。まず、１次元での静的な平衡状態の例から始めます。

3.2.1 界面

簡単な例として、１次元における A/B ポリマーブレンドの界面を計算してみましょう。想定するのは、次の図のように両端が反射壁で左右に A/B ポリマーが相分離し、中央に界面ができる系です。

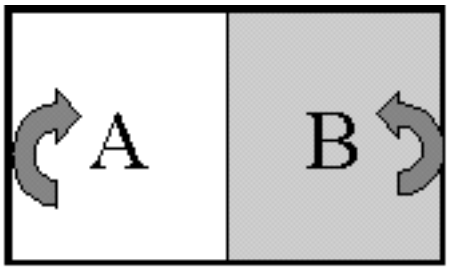


図 3.1: AB 界面計算用入力データの概念図

この計算用の入力ファイル interface_uin.udf を GOURMET を使って開けてみましょう。そして、SUSHIInput フォルダーを開けてみてください。次のようなフォルダーの中身が見えるはずです。

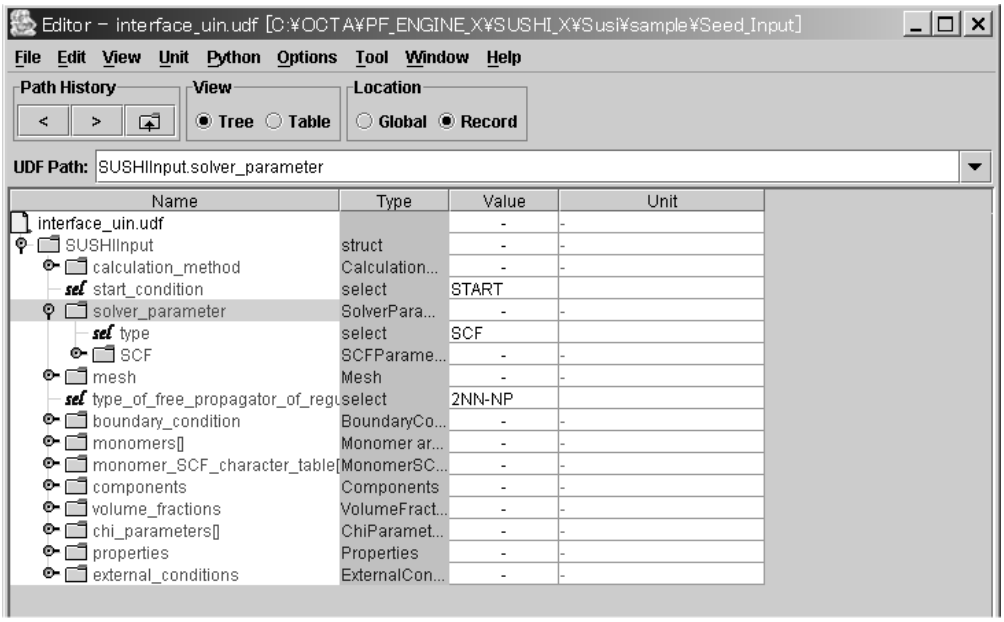


図 3.2: GOURMET 初期画面

最初に calculation_method ホルダー内の type が STATICS、start_condition の３つの変数が start solver_parameter が SCF になっています。どうやら、静的な計算、標準的なスタート、計算方法は SCF と選択されているようです。その他に、SUSHIInput はいろいろなフォルダーを持っています。これらの中身を全て入力しないと SUSHI は動かないのかと思うとやる気がうせてしまいますね。ご安心ください。SUSHI は非常に少ないパラメータで動くように設計されております。

想定した系の計算に最低限必要な入力項目は、

１）SCF 計算制御用パラメータ
２）計算する系を定義するメッシュ
３）系内の分子とその体積分率
４）セグメント間の相互作用を表す χ パラメータ

です。その他にオプションとして、

５）外的条件

で A/B の場所を指定してやれば計算が速く終了します。必要なデータはこの程度です。あまり心配する必要はありません。

ここからは、GOURMET ユーザーのために UDF 入力について詳しく説明してゆきましょう。説明を理解するにあたり、少なくともプログラム言語 C の基礎的な知識があり、”構造体 ”なるものを何となく知っていれば理解が早いと思います。以降にはデータの構造と具体的なデータが対になって登場します。初めての方でも、眺めていれば何となくわかるとは思います。それでは参考のために、interface_uin.udf ファイルをエディターやワープロで開いてください。残念ながら Windowns 上では DOS の改行文字にしか対応していない簡易エディターの類は利用できませが、少し高級なエディターであれば利用が可能です。ユーザーが入力に困らぬようにいろいろなことが書いてあります。最初に記述されているのはデータの構造の定義部分です。次にデータ部があります。

```
\begin{data}
```

以降にデータが記述してあります。ワープロやエディターで begin{data}まで検索してみてください。この後ろに具体的なデータ部分が入力されています。

初めの一歩

まず最初に冒頭で説明したデータの入力が必要です。この部分は以下のように入力されています。SCF の静的計算が選択されています。標準的なリスタートをします。

```
// Data
\begin{data}
SUSHIInput:{ // SUSHIInput/
  calculation_method { //CalculationMethod
    type "STATICS"
    .....
  }
  start_condition "START"
  .....
```

このデータ構造は次のとおりです。

```
// Data definition
class CalculationMethod:{ // The data structure of calculation method.
    // The data type "select" means that user can select the string data from {...}.
    type:select { "STATICS", "DYNAMICS", "MONTECARLO" }
    DYNAMICS:DynamicsParameter
    MONTECARLO:MonteCarloParameter
```

```
        // The DynamicsParameter and the MonteCarloParameter are data structures
        // defined in other classes.
}
SUSHIInput:{
  // Calculation control data ##############################
    calculation_method:CalculationMethod
    start_conditionrestart:select
        // The flag for starting condition
        // START            : normal start
        // CONTINUE         : continue with reading the mesh at the final recoerd
        // RESTART          : restart without reading the mesh at the final recoerd
        // RESTART_READMESH : restart with reading the mesh at the final recoerd
    .....
}
```

SCF 計算制御用パラメータ

次に、SCF 計算制御用パラメータの入力を説明します。データ部分は次のような入力になってます。

```
//データ
solver_parameter {
  type "SCF"
  SCF_parameter { //SCFParameter
    delta_s 1
    constV 0.05
    constW 0.1
    error 0.0001
    random_seed 0
    standard_deviation 0.00015
    method_of_convergence_test "ABSOLUTE"
    convergence_test_interval_step 0
    max_SCF_step 20000
    scf_output_interval_step 0
    SCF_method "INCORE"
    pathintegral_scheme ""
  }
  ....
}
```

たくさんのパラメータが並んでいますが、この例の場合、実際に使用しているのは次のものだけです。

```
        delta_s:double                  //  経路積分用の鎖長の刻み幅。値は正。方向はない。
```

```
        constV:double                   //  化学ポテンシャル更新用定数。
        constW:double                   //  セグメント間相互作用更新用定数。
        error:double                    //  収束判定用しきい値。
        standard_deviation:double       //  場の初期値に与えるノイズの標準偏差。
        max_SCF_step:int                //  SCF 計算における最大繰り返し回数。
```

このデータの構造は次のようになっています。

```
//データ定義
class SolverParameter:{ // The ‘‘solver" means a specialized simulator.
    type:select { "ADF", "FH", "SCF" }
    SCF:SCFParameter // The SCF (Self Consistent Field) method.
    ADF:ADFParameter // The ADF (Approximate Density Functional) method.
                     // (Under construction)
    FH:FHParameter   // The Cahn-Hilliard type dynamics method.
                     // (Under construction)
}
class SCFParameter:{
        delta_s:double                  //  経路積分用の鎖長の刻み幅。値は正。方向はない。
        constV:double                   //  化学ポテンシャル更新用定数。
        constW:double                   //  セグメント間相互作用更新用定数。
        error:double                    //  収束判定用しきい値。
        random_seed:int                 //  乱数発生用シード。
        standard_deviation:double       //  場の初期値に与えるノイズの標準偏差。
        method_of_convergence_test:string    //  収束判定方法。ダイナミクス計算でサポート。
        convergence_test_interval_step:int  //  収束判定をするステップの間隔。
        max_SCF_step:int                //  SCF 計算における最大繰り返し回数。
        output_interval_step:int        //  SCF 計算中の出力ステップ間隔。
        SCF_method:string               //  経路積分の取り扱い方法。:
        pathintegral_scheme:string      //  経路積分の差分スキーム。
}
```

---

delta_s は格子間隔が１の規則格子メッシュを利用し、b=1 と取ったときには１でかまいません。

次の２つの変数 constV、constW は魔法の定数です。この値如何で、SCF の収束の傾向が決まります。これらは、入力条件により変更する必要があります。SUSHI が一番使いにくいのは、この値を適当（何と難しい）に選ばなくてならないというところにあります。何度か使っているとあなたも経験的にその値を適当（何と素晴らしい）に入力できるようになると思います。残念ながら今のところあなたに職人技を期待するプログラムとなってはいますが、指針として constV=0.05、constW=0.1 は静的平衡計算における最大値であると思ってください。χ パラメータを大きくしてセグメント間相互作用を大きくしてゆくと、とたんに収束が難しくなります。いくら待っても SCF が収束しない場合も頻繁に出てきます。そうなったらこれらの値を小さくしてみてください。どちらかを一方的に小さくしても効果はないようです。程々（これもまた何と難しい）に両方とも小さくしてみてください。相対的な比率を 5:10 に保たなくてもよいですから、適当（何と良い言葉でしょう）に変えてみましょう。いくら待っても収束しない計算が収束するかもしれません。注意して頂きたいのは、無

闇にこれらの値を小さくすると、収束にまでに多数の SCF 計算が必要となり確実に計算は遅くなります。収束はするが SCF の回数が多くかかり過ぎるような場合は、これらの値を逆に大きめにしてみましょう。

残念ながら、いくらパラメータの値を変えても数値計算上 SCF が収束しない場合があります。そのような場合、どのくらいの SCF 回数で計算を諦めればよいのでしょう。経験的にですが 20,000 回 SCF を行っても収束する傾向がみられない場合、そのジョブは諦めた方がよいです。サンプルファイルは種々の条件で収束が終了している例を集めてあります。これらを参考にすれば、どの程度の値を使用しているかわかります。何回かの計算をこなすうちに、あなたも SUSHI を使いこなすことがきっと可能になると思います。今後、自動収束方法を実装したいと考えております。

error は主に体積分率の収束判定に利用する閾値です。0.0001 程度で十分であると思います。standard_deviation は静的平衡計算の場合、自己無撞着場の初期値に与えるノイズです。error の 1.5 倍くらいがよいと思います。

SCF の繰り返し計算が収束しなかった場合のことを考えて、計算を停止したい SCF の最大の回数を max_SCF_step に入力しておきましょう。

メッシュ

次に Mesh の入力について説明しましょう。UDF データ SUSHIInput.mesh は次のように入力されています。

```
// Data
  mesh { // Mesh
    name "test"
    type "REGULAR"
    axes [
      id0 { // MeshAxis
        values [ 0 32 32 ]
      }
    ]
    index_rule [ 0 1 2 ]
  }
  type_of_free_propagator_of_regular_mesh "2NN-NP"
```

データ構造の定義は次のとおりです。

```
// Data definition
class MeshAxis:{
    values[]:double
        // In case of the regular, spherical, or cylindrical mesh,
        // this array contains the minimum and the maximum values
        // of the range along the axis, and the total number of
        // divided cells along the axis, respectively.
        // In case of the rectangular mesh, an array of the coordinates
        // of the mesh points are stored.
}
class Mesh:{
    name:KEY // The data type "KEY" is a character string, which is used to search the
```

```
            // target data structure.
    type:select { "REGULAR", "RECTANGULAR", "CYLINDRICAL", "SPHERICAL" }
    axes[]:MeshAxis
        // Array of the mesh axes.
        // Its dimension is the total number of coordinate axes of the system.
    index_rule[]:int
        // This specifies how the array elements (i, j, k) are arranged on the memory.
        // The first argument i runs first for example (0,0,0),(1,0,0)...(X-1,Y-1,Z-1).
        // For SUSHI, this is fixed as [0, 1, 2] except cylindrical mesh and
        // is fixed as [2, 1, 0] for cylindrical mesh.
        // Although this parameter does not affect the functions of SUSHI,
        // it is used when the mesh data is passed to another simulator and
        // is passed to the viewer on GOURMET.
}
SUSHIInput:{
    ....
    mesh:Mesh
    type_of_free_propagator_of_regular_mesh:select { "1NN-P", "2NN-NP", "2NN-P", "3NN-P" }
        // The type of the discretized Laplacian operator.
        // This is for the regular mesh only.
    ....
}
```

入力値は、メッシュの名前が test で、タイプが規則メッシュです。MeshAxis は id0 と１つしか入力されておりません。この数が空間次元に対応しますのでこの系は１次元です。その値は values として３つの値が入力されています。これらの意味は定義のとおり、軸の最小値，最大値，分割数を意味します。0 から始まり 32 で終わる X 軸が 32 分割されています。多次元にしたい場合は MeshAxis の配列の数を増やして values を追加してください。データの追加方法については GOURMET のマニュアルを参照してください。直接エディターで追加してももちろん有効になります。index_rule はメッシュ点の出力方法を示します。SUSHI はメッシュ上のデータを空間的な次元に関わらずシーケンシャルな一次元情報として出力します。その場合のメッシュ点の出力方法をこのデータで示します。SUSHI の場合一般的に i,j,k の順にインデックスが増加します。つまり、(0,0,0)、(1,0,0)、(2,0,0)、... にようになります。ただし、円筒座標の場合のみ異なっていて j,k の順にインデックスが増加します。つまり、(0,0)、(0,1)、(0,2)、... のようになります。GOURMET の画面では次のように見えるはずです。

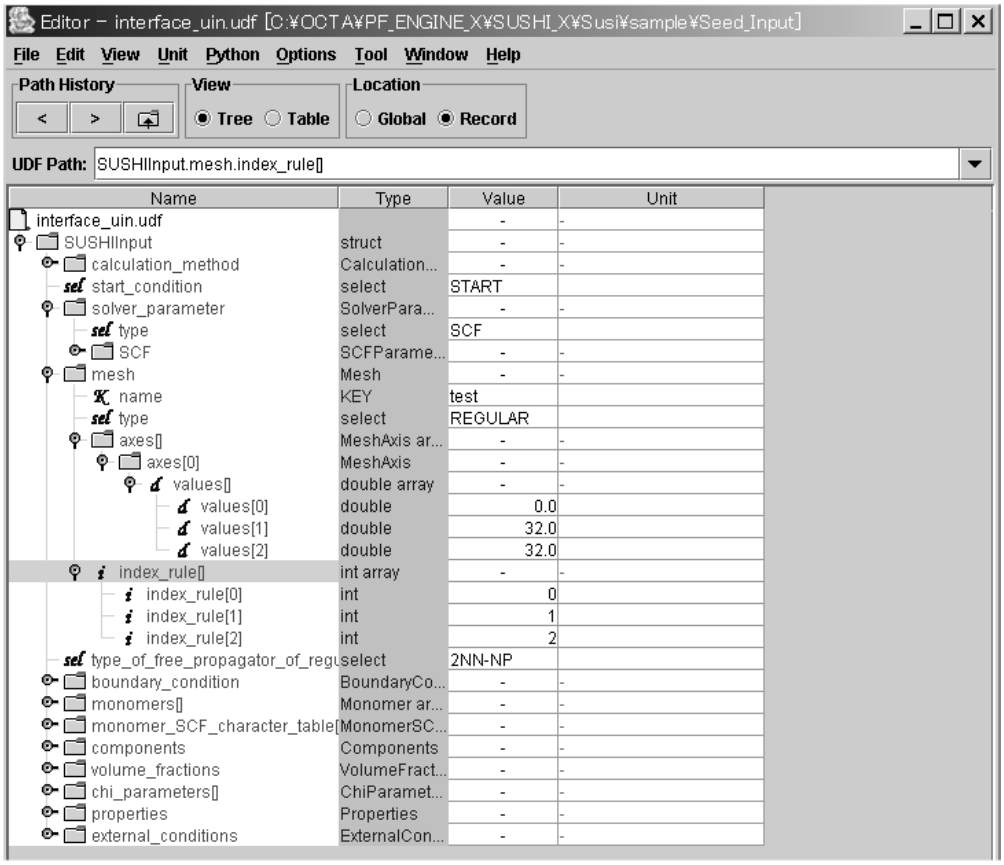


図 3.3: メッシュデータ画面

メッシュには境界条件がつきものです。境界条件の UDF データ SUSHIInput.boundary_condition の入力データとデータ構造は次のとおりです。

```
// Data
  boundary_condition { // BoundaryCondition
    conditions [
      id0 { // AxisBoundaryCondition
        axis_conditions [ "NEUMANN"  "NEUMANN"  ]
      }
    ]
    .....
  }
// Data definition
class AxisBoundaryCondition:{ // The boundary conditions at the both ends of the axis
                              // are specified.
    axis_conditions[]:string
        // PERIODIC :Periodic boundary conditions
        //          In this case, specifying only axis_condition[0] is enough.
        // DIRICHLET or WALL :Absorbing wall,
        //    i.e. the Dirichlet boundary condition with the boundary value 0.
        // NEUMANN :Reflective wall,
```

```
        //    i.e. the Neumann boundary conditions with vanishing gradient of the field.
}
class BoundaryCondition:{
    conditions[]:AxisBoundaryCondition // An array of the boundary conditions for each
                                       // axis.
    ....
}
SUSHIInput:{
    .....
    boundary_condition:BoundaryCondition
        // Boundary conditions.
        // Refer to Section 2.7.4.
    .....
}
```

メッシュの軸の両端が反射壁（定数が０の NEUMANN 境界条件) になっています。あとは空ですね。これでメッシュの入力は終わりです。

分子

さて計算をする系の入力は終わりました。それでは次に系内に存在させる分子の入力をしましょう。ポリマーはモノマーからつくられます。まず、モノマーの入力 UDF データ SUSHIInput.monomers を見て頂きましょう。

```
//データ
  monomers [
    id0 { //Monomer
      species_name "A"
      specific_volume 1
      effective_bond_length 1
    }
    id1 { //Monomer
      species_name "B"
      specific_volume 1
      effective_bond_length 1
    }
  ]
```

データ構造は簡単でモノマーの持つデータは名前、有効体積、有効結合長です。モノマーの名前が A、B と入力されていて、後は１です。一般的な入力はこれで十分です。名前を入力して後は１にしてください。モノマーと溶媒のデータ構造の定義は次のとおりです。

```
// Data definition
class Monomer:{
```

```
    species_name:string           // Name of the monomer.
    specific_volume:double        // Specific volume of the monomer.
    effective_bond_length:double  // Effective bond length corresponding to a single
                                  // monomer.
}
class Solvent:{
    name:string                   // Name of the solvent.
    specific_volume:double        // Specific volume of the solvent.
}
```

溶媒のデータ構造はモノマー以上に簡単で、名前、有効体積しかデータを持ちません。では、実際のポリマーはモノマーを利用してどのように定義されるのでしょう。次がポリマーのデータ定義部分です。

```
// Data definition
class Block:{         // Sub-chain.
    monomer_name:string
        // The name of the monomer which constitutes this sub-chain.
        // This monomer name must be defined as a Monomer in advance
    number_of_monomer:double
        // The total number of monomers contained in this sub-chain.
}
class JunctionPair:{  // A pair of the ID's of the junction points at the both ends of
                      // this sub-chain.
    first:int
    second:int
}
class Polymer:{
    type:select {"HOMO","BLOCK","COMB","STAR","GENERAL" }
        // The keyword specifying the branching structure of the chain,
        // i.e. how to connect the subchains.
        // HOMO :  A homo polymer.
        // BLOCK : A linear multiblock copolymer made of a sequence of the sub-chains
        //         with the same order as they are stored in this array.
        // STAR :  A star block copolymer made of sub-chains which are connected
        //         at a single junction point.
        // COOMB : A comb-type block copolymer.
             // The order of the sub-chains stored in the array is as follows.
             // main chain - side chain - main chain - side chain -.....
             // For example, an array of the sub-chains A1, B1, A2, B2, and A3
             // indicates the following structure.
             //     A1--+--A2--+--A3
             //         B1     B2
        //  GENERAL : The way how the sub-chains are connected to each other
             //       is specified by the ID's of the two junction points at the
             //       both ends of each sub-chain.
```

```
    blocks[]:Block                // Array of sub-chains.
    junction_pairs[]:JunctionPair // Array of the pair of the ID's of the junction points
            //                       at the both ends of the sub-chain.
            // This parameter is available only when the type of the polymer is GENERAL.
            // The values of the ID must start from 0.
            // For example, when the number of elements of Block is 1 and a pair of
            // ID's is [0, 0], it means that this polymer is a ring polymer.
}
```

ポリマーはモノマーからできた部分鎖と結合点の定義のみで記述されています。データ構造は簡単ですが、これだけでどのようなトポロジーを持ったポリマーでも記述することが可能です。ポリマーの入力イメージは次の図のようなものです。

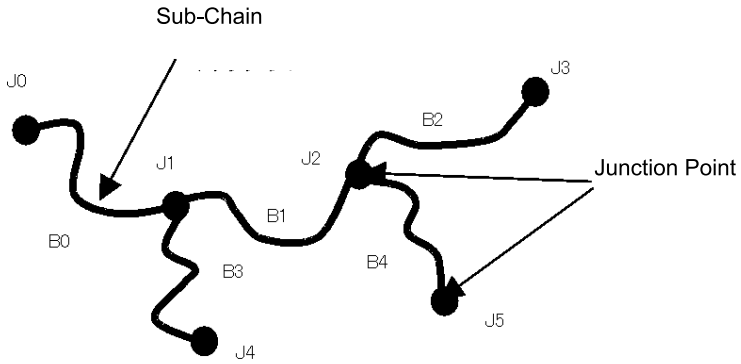


図 3.4: 鎖の分岐構造のモデル

部分鎖は同じ種類のモノマーでできているとします。部分鎖の両端には結合点があります。全ての部分鎖と結合点を定義します。次に部分鎖と結合点に番号を付けます。B が部分鎖、J が結合点の番号です。必ず 0 から番号を振ってください。これでポリマーのトポロジカルな構造の定義は終わりです。部分鎖のデータ構造 Block はモノマーの名前と長さをデータとして持ちます。部分鎖の数と同じ数だけ部分鎖両端の結合点の番号を持ったデータが必要です。それがデータ構造の JunctionPair です。番号の付け方は任意です。どの部分鎖や結合点から番号を付けてもかまいません。結合情報が正確に入力されれば SUSHI は経路積分に必要な経路情報を自動的に探索します。番号の重複や連番途中の番号の欠落があった場合、SUSHI はエラーを出して実行を停止します。最終的には次のようにポリマーと溶媒の配列をもつ UDF データ構造 SUSHIInput.components が定義されています。

```
// Data definition
class Components:{
    polymers[]:Polymer // Array of polymers.
    solvents[]:Solvent // Array of solvents.
}
SUSHIInput:{
    .....
  // Polymer and solvent ##############################
    monomers[]:Monomer // Array of a monomer.
        // Array of the characteristic parameters of monomers.
```

```
    monomer_SCF_character_table[]:MonomerSCFChar
        // A set of components (polymers and solvents) that can be used in the simulations.
    components:Components
    .....
}
```

実際のデータは次のように入力されています。

```
//データ
  components { //Components
    polymers [
      id0 { //Polymer
        type "HOMO"
        blocks [
          id0 { //Block
            monomer_name "A"
            number_of_monomer 20
          }
        ]
        junction_pairs [
        ]
      }
    .....
    solvents [
    ]
  }
```

おかしなことに junction_pairs に何のデータも入っていません。これは、type で既にポリマーの構造が決めてあるからです。代表的なポリマーの構造は type を決め、ポリマーを構成する部分鎖 Block を入力するだけで入力できるようにしてあります。詳しくはデータ定義部のコメントをご覧ください。この例では type はホモポリマーですので、部分鎖の配列 blocks にはモノマー A からなる長さ 20 の部分鎖 1 本だけが入力されています。ホモポリマー以外ではこの配列にモノマーの種類と長さが異なる複数の部分鎖が入力されます。type が GENERAL と入力された場合は、junction_pairs が有効になりますのでここにデータを入力しなくてはなりません。この例ではモノマーは入力されておりません。

体積分率

分子を入力したらその体積分率も入力します。データ構造は次のとおりです。

```
// Data definition
class VolumeFraction:{  // Volume fraction.
    id:int // The element index of the component. The number starts from 0.
    volume_fraction:double
        // If the ensemble is CANONICAL, this parameter specifies the total volume
```

```
        // fraction in the system.
        // If the ensemble is GRANDCANONICAL, this parameter specifies the equilibrium
        //  volume fraction in the bulk phase.
    ensemble:string  // Statistical ensemble of the system.
        // CANONICAL : Canonical ensemble.
        // GRANDCANONICAL : Grand canonical ensemble.
        //     Dynamic calculation is not available for this case.
    bulk_volume_fraction:double  // The volume fraction in the bulk phase.
        //     This parameter is used when the volume fractions in the simulation box and
        //     those in the reservoir should be distinguished.
        //     If you do not use this feature, set -1 to this parameter.
}
class VolumeFractions:{
    polymer_volume_fractions[]:VolumeFraction  // Array of the volume fractions of the
                                               // polymers.
    solvent_volume_fractions[]:VolumeFraction  // Array of the volume fractions of the
                                               // solvents.
}
SUSHIInput:{
    .....
    volume_fractions:VolumeFractions
    .....
}
```

SUSHI の入力設計の考え方としては実験をイメージしてます。つまり、沢山のポリマーや溶媒の試薬を用意して、それらの中から適当なものを選んで組成を決め、「系内に投入して種々のシミュレーションを行えたら便利だなぁ〜」と、いう考えに基付き入力ができるようにしてあります。ですから、components にポリマーと溶媒を入力したからといって、それが必ずしも計算に使われる訳ではありません。components はポリマーと溶媒が分けて置いてあるただの試薬棚であると思ってください。シミュレーションに投入される分子は、ここで分子の種類（ポリマーか溶媒）と試薬番号、そして体積分率が入力されることで決定されます。id：コンポーネントの配列番号、とあるのが試薬の番号です。この番号が components で入力した分子の配列番号に対応します。例えば、components で入力した配列番号１のポリマーを入力したければ１と入力してください。このような入力となりますので、入力される順番に意味はありません。不連続でも逆順でもかまいません。但し重複した id はいけません。SUSHI の入力において、全ての秤量は体積分率が単位となります。体積分率の入力以外でも様々な入力に components の配列番号が利用されております。それを体積分率で入力した配列番号と間違わないでください。あくまでも components における配列番号です。

さて、実際の UDF データ SUSHIInput.volume_fractions は次のように入力されております。

```
// Data
  volume_fractions { // VolumeFractions
    polymer_volume_fractions [
      id0 { // VolumeFraction
        Id 0
        volume_fraction 0.5
        ensemble ""
```

```
        bulk_volume_fraction -1
      }
      id1 { // VolumeFraction
        Id 1
        volume_fraction 0.5
        ensemble ""
        bulk_volume_fraction -1
      }
    ]
    solvent_volume_fractions [
    ]
  }
```

A、B それぞれのポリマーが体積分率で 0.5、0.5 づつ投入されています。ensemble に何の入力もありませんが、この場合、カノニカルアンサンブルがデフォルトとして設定されます。bulk_volume_fraction は特に使用しておりません。この値はグランドカノニカルアンサンブルの計算で有効になります。グランドカノニカルアンサンブルの計算が選択されたときにここに入力値がないと、バルクの体積分率は、volume_fraction の値になります。しかし、例えば系内にグラフト鎖が存在し系内の初期組成がバルクと異なり、系内の体積分率 volume_fraction とバルクでの体積分率 bulk_volume_fraction の値を別々に入力した方が SCF の収束が早くなるような場合はここに値を入力してください。使用していない場合、不使用がわかるように -1 を入れておきましょう。グランドカノニカルアンサンブルが選択されたポリマーのダイナミクス計算は出来ません。ご注意ください。

χ パラメータ

次に必要なのは χ パラメータです。この入力は簡単です。次のようになります。

```
// Data definition
class ChiParameter:{
    name_i:string     // The name of the species of i-th monomer.
    name_j:string     // The name of the species of j-th monomer.
    parameter:double  // The value of chi parameter.
                      // Chi_ij is automatically set using the symmetric relation
                      // Chi_ij = Chi_ji.
                      // If the value is not defined, it is assumed to be 0.
}
SUSHIInput:{
    .....
    chi_parameters[]:ChiParameter  // Array of Chi parameter values.
    .....
}
// Data
  chi_parameters [
    id0 { // ChiParameter
      name_i "A"
      name_j "B"
```

```
      parameter 0.2
    }
  ]
```

χ_{AB} が 0.2 と入力されています。鎖長が 20 でしたので $\chi N = 4$ となります。

外的条件

殆どの入力は終了しましたが、最後に系に魔法をかけてセグメント A を左に寄せて収束を早くさせてしまいましょう。そのようにできるのが、自己無撞着場に初期値を与えて任意の場所にドメインが生成することを早める UDF データ SUSHIInput.external_conditions.static_conditions.domain_specification_conditions[] の入力です。そのデータ構造は次のようになります。

```
// Data definition
class AxisRegion:{ // Specify an interval on an axis.
    axis_name:string // Name of the axis: X, Y, Z, R or H.
    r_min:double     // Minimum value of the interval.
    r_max:double     // Maximum value of the interval.
                     // When Maximum = Minimum, it corresponds to a mask on a point.
}
class DomainSpecificationCondition:{
    name:string                 // The name of the segment species.
    domain_regions[]:AxisRegion // The array of the intervals on the axes.
}
class StaticConditions:{
    ....
    domain_specification_conditions[]:DomainSpecificationCondition
        // An array of the conditions for setting the initial values of the
        // self-consistent fields.
    ....
}
class ExternalConditions:{
    ....
    static_conditions:StaticConditions
    ....
}
SUSHIInput:{
    .....
     external_conditions:ExternalConditions    .....
    .....
}
```

入力データは次のようになっています。

```
// Data
      domain_specification_conditions [
        id0 { // DomainSpecificationCondition
          name "A"
          domain_regions [
            id0 { // AxisRegion
              axis_name "X"
              r_min 0
              r_max 16
            }
          ]
        }
      ]
```

X 軸上 0〜16 の区間に A セグメントを存在させようとしています。

これで入力の説明は終わりです。

計算の実行

では、GOURMET 上や、コンソール上で SUSHI を起動してみてください。出力ファイルは、interface_uot.udf とすることにしましょう。

```
> sushi -Iinterface_uin.udf
```

計算終了後に interface_uot.udf を GOURMET から開いてください。初期画面は、入力データ用の画面です。まず、計算結果へ移りましょう。次の図のように record Step1 を選んでください。

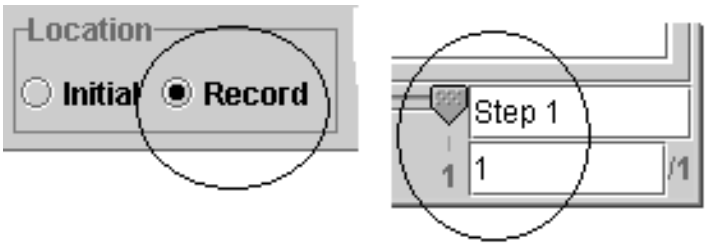


図 3.5: 計算結果への移動

次に計算結果を表示してみましょう。マウスを SUSHIOutput アイコンに移動してから右ボタンを押してください。次の図のような plot_1D_field を選択できる窓が表示されるはずです。OK ボタンを押すとグラフが表示されます。

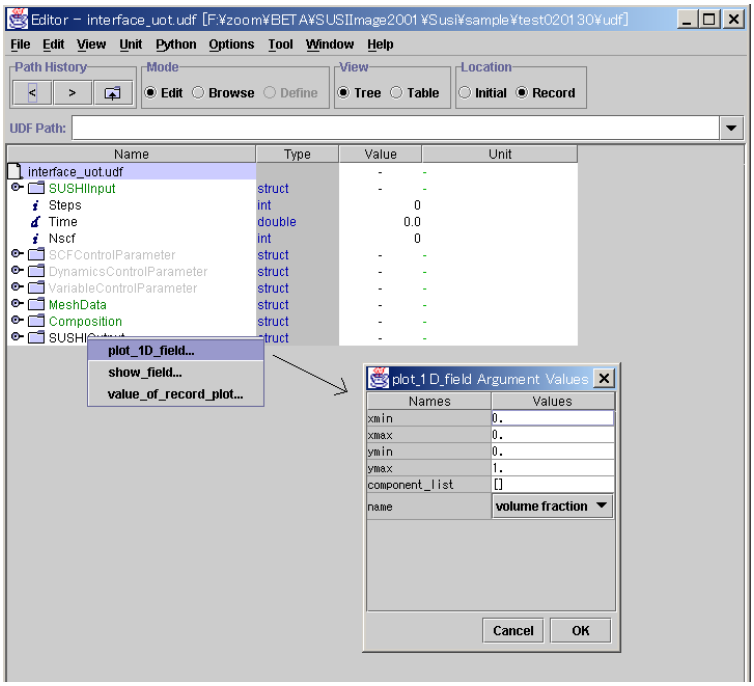


図 3.6: シミュレーション結果の出力

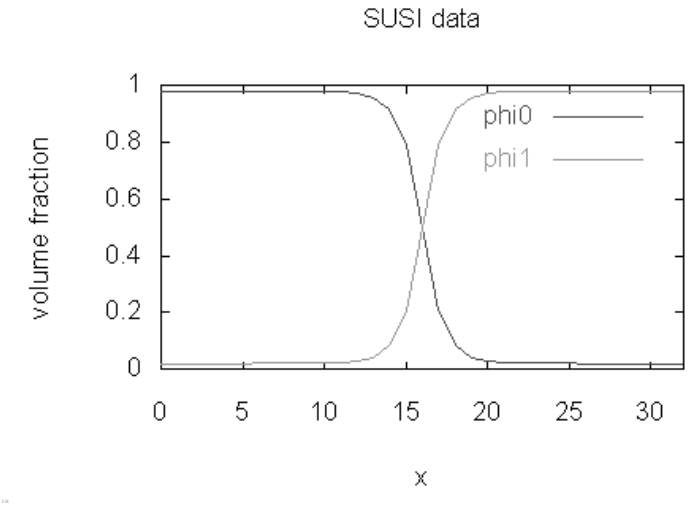


図 3.7: A/B ポリマーブレンド界面 χ N=4

A/B ポリマーブレンド系の界面が計算できました。SUSHI はこのような界面の計算を得意としています。ここでの入力を少し変えるだけで種々の計算が可能となります。以降でそれらを説明しましょう。

3.2.2 相分離

interface_uin.udf のデータでメッシュの境界条件 SUSHIInput.boundary_condition を変えてみます。

//データ

```
boundary_condition { //BoundaryCondition
  conditions [
    id0 { //AxisBoundaryCondition
      axis_conditions [ "PERIODIC"  ]
    }
  ]
}
```

このようにして周期境界条件にしたデータが blend_uin.udf です。計算結果は次の図のように、周期境界条件の影響で界面が 2 つ現れた相分離状態になります。

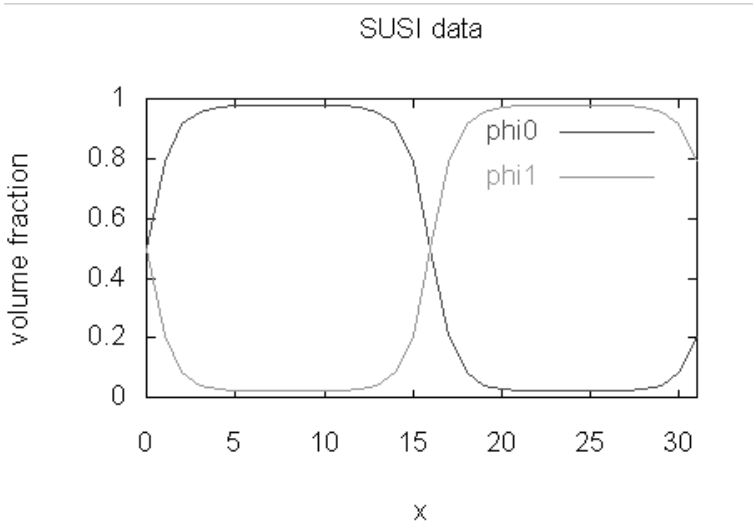


図 3.8: 周期境界条件をもつ系での A/B ポリマーブレンド界面 χ N=4

3.2.3 ラメラ構造

次にジブロック共重合体によるラメラ構造を計算してみましょう。blend_uin.udf で SUSHIInput.components.polymers の内容を変更し、A/B ポリマーをつないで次のようにブロックコポリマーにします。分子を 1 つにするので SUSHIInput.volume_fractions.polymer_volume_fractions[] の体積分率も ID 0 を 1 に変える必要があります。χN を大きくしないと相分離しませんので、χ を 0.4 にしましょう。$\chi N = 16$ となります。そして、ドメインの設定である UDF データ SUSHIInput.external_conditions.static_conditions.domain_specification_conditions[] の入力を削除します。

```
//データ
    .....
    polymers [
      id0 { //Polymer
        type "BLOCK"
        blocks [
          id0 { //Block
            monomer_name "A"
```

```
      number_of_monomer 20
    }
    id1 { //Block
      monomer_name "B"
      number_of_monomer 20
    }
  ]
  junction_pairs [
  ]
}
```

変更した入力データは block_uin.udf です。計算結果は次のようになります。ラメラ構造が現れています。残念ながら、システムサイズを最適化しておりませんので自由エネルギーが最安定の構造ではありません。

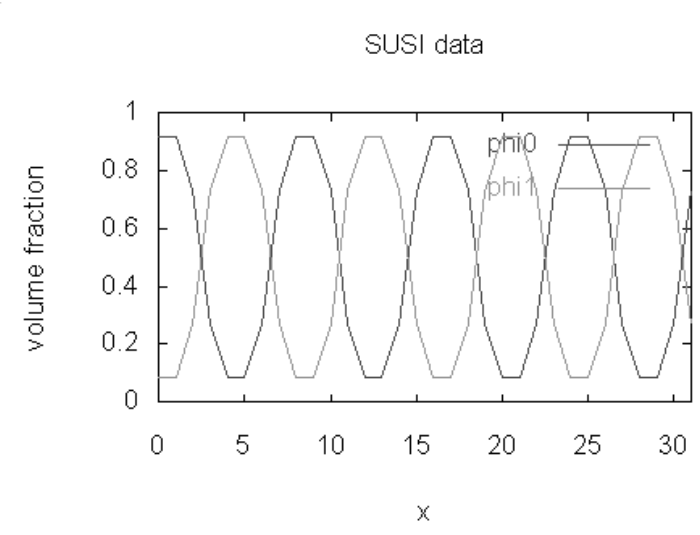


図 3.9: ラメラ構造 χ N=16

3.2.4 固体壁の効果

次に、固体壁近傍での溶液中の高分子の濃度をシミュレーションしてみましょう。interface_uin.udf の内容をちょっと変えるだけで計算は可能です。まず、系の境界条件を変えます。NEUMANN、NEUMANN の入力であったものを WALL（または DIRICHLET)、NEUMANN と片方を壁にします。溶液中の高分子をシミュレーションするために A の長さを 100、B の長さを 1 として、溶媒 B 中にある高分子 A をシミュレーションすることにします。長さ 1 のポリマー B の代わりに溶媒 B を入力してもかまいません。溶媒は良溶媒とするのに χ_{AB} は 0.5 と入力しましょう。体積分率はポリマーの準希薄溶液にするために A、B それぞれの体積分率を 0.001、0.999 にします。そして、ここが重要なのですが、系をグランドカノニカルアンサンブルにするために、体積分率の入力を次のように変更します。

```
//データ
  volume_fractions { //VolumeFractions
```

```
  polymer_volume_fractions [
    id0 { //VolumeFraction
      Id 0
      volume_fraction 0.001
      ensemble "GRANDCANONICAL"
      bulk_volume_fraction -1
    }
    id1 { //VolumeFraction
      Id 1
      volume_fraction 0.999
      ensemble "GRANDCANONICAL"
      bulk_volume_fraction -1
    }
  .....
}
```

アンサンブルが GRANDCANONICAL と入力されています。この入力により系は、自動的に反射壁での濃度がバルクの濃度となるグランドカノニカルアンサンブルの系として計算されます。入力した濃度がバルクの濃度となります。計算結果は、バルクとの平衡状態となります。変更した入力ファイルは depletion_uin.udf です。その計算結果は次のようになります。

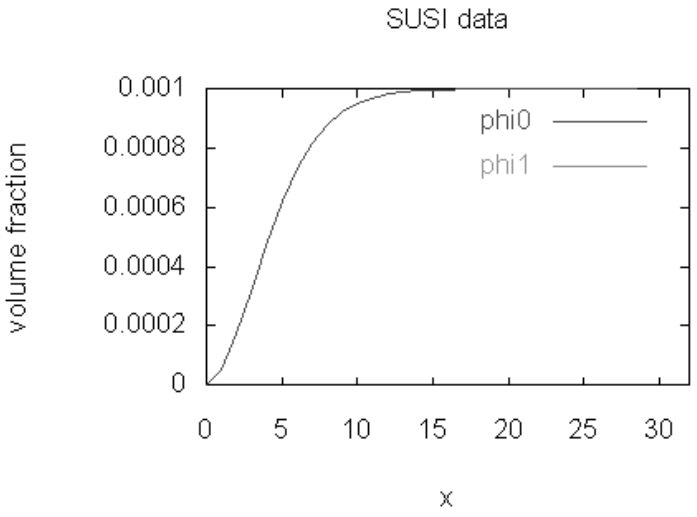


図 3.10: 固体壁近傍での枯渇構造

ポリマーは壁にへばり付くようなコンフォメーションを統計的にとりにくく、エントロピーの効果でセグメントの分布は固体壁近傍で枯渇しています。

3.2.5 吸着

上記の結果で、もし、壁と高分子間に引力相互作用が働いたらどうなるでしょう。今度は吸着をシミュレーションしてみます。入力の変更は簡単で、壁とポリマーの相互作用パラメータである UDF データ SUSHIInput.external_conditions.surface_chi_parameters[] を次のように入力します。

```
// Data definition
class SurfaceChiParameter:{
    boundary_name:string // The name of the boundary wall that produces the interaction.
    target_name:string   // The name of the target segment species.
    parameter:double     // The value of the chi parameter.
}
class ExternalConditions:{
    surface_chi_parameters[]:SurfaceChiParameter
    ....
}
SUSHIInput:{
    .....
    external_conditions:ExternalConditions    .....
    .....
}
// Data
  external_conditions { // ExternalConditions
    surface_chi_parameters [
      id0 { // SurfaceChiParameter
        boundary_name "XMin"
        target_name "A"
        parameter -2
      }
    ]
    ....
  }
```

X=0 での YZ 面とセグメント A との相互作用パラメータが −2 と入力されています。変更した入力ファイルは adsorption_uin.udf です。さて、計算結果は次の図のようになります。枯渇していたポリマーが壁に吸着しています。

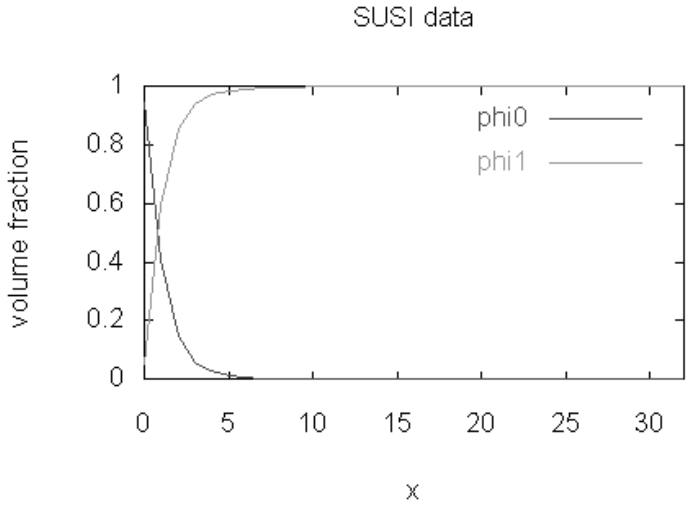


図 3.11: 吸着

3.2.6 グラフト

吸着ではなく、ポリマー末端を壁にグラフトさせたらどうなるでしょう。depletion_uin.udf の内容を変えて A ポリマーの末端を壁にグラフトしてみます。それには外的条件にあるグラフト条件の UDF データ SUSHIInput.external_conditions.graft_conditions[] を入力します。

```
// Data definition
class GraftCondition:{
    polymer_ID:int       // ID of the grafted polymer.
    junction_ID:int      // ID of the grafted free end.
    boundary_name:string // The name of the boundary onto which the polymer is grafted.
    obstacle_ID:int
}
class ExternalConditions:{
    ....
    graft_conditions[]:GraftCondition
    ....
}
SUSHIInput:{
    .....
     external_conditions:ExternalConditions    .....
    .....
}
// Data
    graft_conditions [
      id0 { // GraftCondition
        polymer_ID 0
        junction_ID 0
```

```
      boundary_name "XMin"
    }
  ]
```

IDが0のポリマーですから、ポリマーAのIDが0である自由端をX=0でYZ面にグラフトさせています。さて、次が重要なのですが、体積分率であるUDFデータSUSHIInput.volume_fractions.polymer_volume_fractions[]を変更します。

```
//データ
  volume_fractions { //VolumeFractions
    polymer_volume_fractions [
      id0 { //VolumeFraction
        Id 0
        volume_fraction 0.001
        ensemble "CANONICAL"
        bulk_volume_fraction -1
      }
      id1 { //VolumeFraction
        Id 1
        volume_fraction 1.
        ensemble "GRANDCANONICAL"
        bulk_volume_fraction -1
      }
    .....
  }
```

ポリマーAのアンサンブルをCANONICALにしました。これは、グラフトして系内に必ず存在する分子になったための変更です。そしてポリマーBの体積分率が0.999であったものが1.0になってます。これは、バルクにポリマーBしか存在しないことを意味しています。つまり、グラフトしているポリマーとしてだけポリマーAが存在するような設定にしました。変更したファイルはgraft_uin.udfです。このファイルを入力させてSUSHIを実行すると、体積分率の和が1.0にならないので、最初にSUSHIはWarningを出しますが、気にしないでください。結果は次の図のようになります。

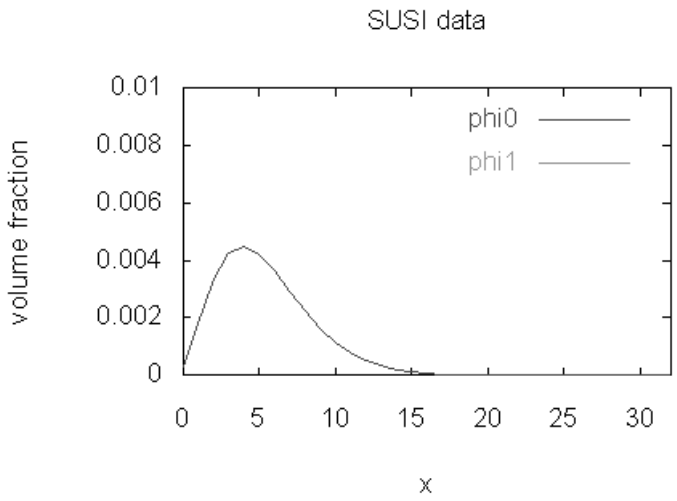


図 3.12: グラフト

グラフトによりセグメントは壁近辺に分布しますが、壁際で枯渇しています。

3.3 ２次元での計算例

さて、2 次元の計算方法についても説明しましょう。

3.3.1 A/B ポリマーブレンド系の相分離ダイナミクス

１次元の例の冒頭で説明した χN が 4 の A/B ポリマーブレンドの周期境界条件での相分離を計算してみましょう。2 次元へ拡張は非常に簡単です。メッシュの軸 SUSHIInput.mesh.axes[] を増やすだけです。32 × 34 （テスト用にわざと正方からずらしてあります）の系に拡張した例が blend2D_4_dy_uin.udf の入力ファイルです。計算時間を稼ぐためにポリマーの長さを 10 と短くし、χ パラメータを 0.4 にしてありますが、内容は blend_uin.udf と殆ど同じです。異なるのは動的計算用に冒頭の入力パラメータ SUSHIInput.calculation_method が DYNAMICS になります。SCF のパラメータについては、constW を 1. にしても殆ど問題ありません。何故なら、SCF のときに ϕ は固定されているからです。動力学計算が選択されると、動力学制御用のパラメータ SUSHIInput.dynamics_parameter の入力が有効になります。具体的なデータは次のようになります。

```
// Data definition
class DynamicsParameter:{
    delta_t:double              // The time mesh width.
    variable_delta_t:VariableDelt // Parameters for variable time mesh.
    max_dynamics_step:int       // The total numbers of the time steps.
    output_interval_step:int    // Output step interval.
    archives_interval_step:int  // Output step interval to the archives file.
    log_interval_step:int       // Output step interval to the standard output file.
    dynamics_scheme:select { "EXPLICIT", "IMPLICIT" } // EXPLICIT is the default value.
        // The scheme for the integration of the equation of motion.
        // EXPLICIT  The explicit scheme. Usually, this scheme is enough.
```

```
        // IMPLICIT  The implicit scheme.
    compressibility:double      // Compressibility for SCF
}
class CalculationMethod:{
    type:select { "STATICS", "DYNAMICS", "MONTECARLO" }
    DYNAMICS:DynamicsParameter  // Refer to Section 2.7.10.
    MONTECARLO:MonteCarloParameter
}
SUSHIInput:{
    calculation_method:CalculationMethod
    .....
}
// Data
SUSHIInput:{ // SUSHIInput
  ....
  calculation_method { //CalculationMethod
    type "DYNAMICS"
    DYNAMICS { //DynamicsParameter
      delta_t 0.01
      variable_delta_t { //VariableDelt
        max_delta_t 0
        variable_coef 0
      }
      max_dynamics_step 500000
      output_interval_step 5000
      archives_interval_step 5000
      log_interval_step 1
      dynamics_scheme ""
      compressibility 0
    }
    .....
  }
  ....
}
```

動力学をするために追加しなくてはならないパラメータはダイナミクス計算用時間刻み幅、最大ステップ数、出力ステップ間隔です。ダイナミクスの差分スキームに何も入力されていませんが、デフォルトの陽的解法が選択されています。スキームはデフォルトで十分です。圧縮率に 0 以上の値を入力すると圧縮性のある動力学を、圧縮率を 0 にしておけば完全に非圧縮な動力学を開始します。この圧縮率は SCF 計算でしか有効になりません。一番重要なのは時間刻み幅です。これが大きすぎるとすぐに計算が不能となります。小さすぎればなかなか先に進みません。空間メッシュの大きさに依存しますが、メッシュ幅が 1 のとき、0.001 前後が適当であると思います。動力学ではさらに分子の易動度である UDF データ SUSHIInput.external_conditions.dynamic_conditions にある segment_mobilities と polymer_mobilities、solvent_mobilities が入力可能です。その定義は次のとおりです。非常に低濃度の成分を含む場合、必ずこれらの入力をする必要があります。詳しくは、第 2.6、2.7.10 節をご覧ください。ここでは入力しません。variable_delta_t は、時間刻み幅を自動的に変化させる場合に利用し

ますが、ここでは利用しません。

```
// Data definition
class SegmentMobility:{  // Mobility of a segment.
    segment_name:string  // The name of the segment/solvent whose mobility is specified.
    mobility:double      // The value of the mobility.
}
class LocalMobility:{  // The position dependent mobility.
    component_ID:int   // ID of the component.
    type:select { "ROUSE", "REPTATION" }
}
class DynamicConditions:{
    segment_mobilities[]:SegmentMobility
    types_of_polymer_mobility[]:LocalMobility
    types_of_solvent_mobility[]:LocalMobility
    ....
}
class ExternalConditions:{
    ....
    dynamic_conditions:DynamicConditions
}
SUSHIInput:{
    .....
    external_conditions:ExternalConditions    .....
    .....
}
```

さて、計算結果 blend_dy_4_uot.udf を GOURMET で開いてください。SUSHIOutput サブホルダーにカーソルを移動し、右クリックし、show_filed.. を選択しましょう。そして OK ボタンを押します。アニメーションボタンが左下にありますから、スタートさせましょう。次の図のように A/B ポリマーブレンドが相分離を開始します。

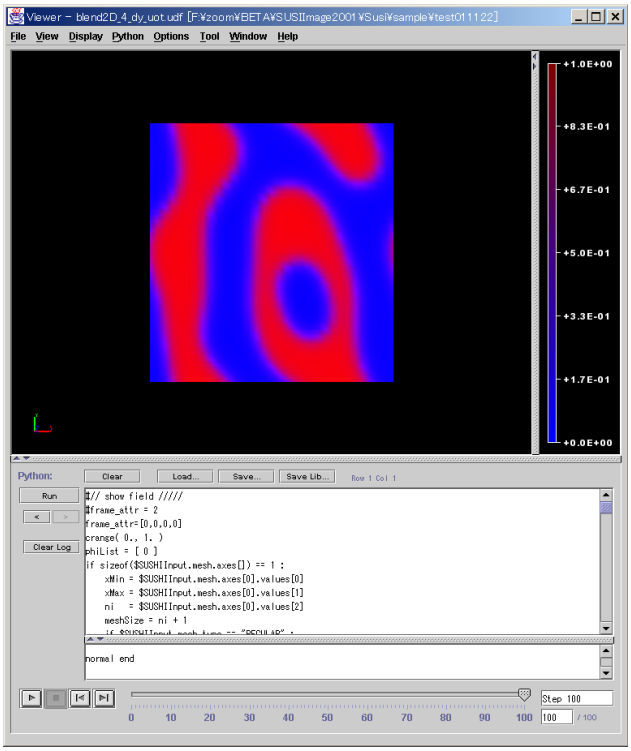


図 3.13: A/B ポリマーブレンド相分離ダイナミクス

3.4 ３次元での計算例

最後は３次元の例を示します。もうお気付きかと思いますが、３次元の計算に拡張するのはメッシュ軸をさらに１本増やすだけです。

3.4.1 シリンダー構造

せっかく３次元の計算をするのであれば、具体的なモルフォロジーを計算してみましょう。ブロック比が 0.25、$\chi N = 20$ のブッロク共重合体は、相図上ではシリンダー構造になります。SCF を使用した平均場の計算は、もちろんこのようなモルフォロジーをシミュレート可能です。しかし、微妙な自由エネルギーの谷間に捕まるので、無闇なシミュレーションをするだけではきれいなモルフォロジーはまず得られません。簡単に狙ったモルフォロジーを計算するのは、静的計算の１番最初の例で使用した UDF データ SUSHIInput.external_conditions.static_conditions.domain_specification_conditions[] を入力するのが最も効果的です。この機能を利用してシリンダー構造を計算した結果が cylinder_3D_uot.udf です。その入力ファイル cylinder_3D_uin.udf をご覧になれば、どのような入力をしているか、すぐに分かると思います。結果は、GOURMET の Viewer 上で show.py を Run するだけで次の図のように見ることが可能です。シリンダー状のモルフォロジーになっています。格子定数の最適化がなされていないので、最安定な hexyagonal 構造にはなっていませんが、簡単に狙ったモルフォロジーをシミュレートできることがお分かりになると思います。リスタートの機能を利用すれば構造の最適化も難しくはありません。

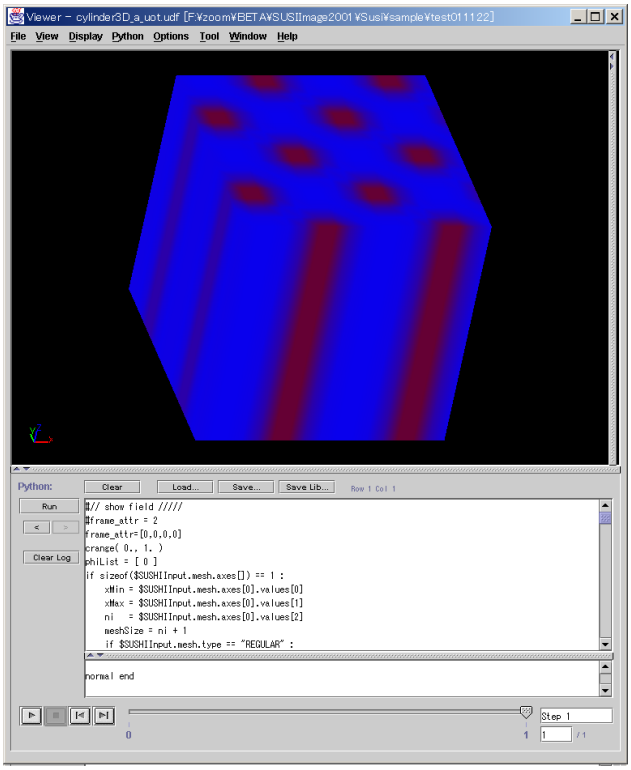


図 3.14: シリンダー構造

3.5　操作入門の最後に

例として用いた系は規則構造のメッシュですが、SUSHI は 1 次元の球座標や、2 次元の円筒座標等の計算も行えます。入力方法は規則構造のメッシュと大差ありません。また、説明しきれなかった種々の機能が備わっています。いくつかの機能を組み合わせるだけで、シュミレーションのバリエーションは無数といってよいくらいあると思います。不明な点は http://octa.jp の BBS へお問い合わせください。あなたの研究に SUSHI が少しでもお役に立つことを願っております。

第4章　適用事例

4.1　A/B ホモポリマーブレンド系界面張力の計算

4.1.1　概要

A/B ホモポリマーブレンド系の界面張力に与える多分散性の影響に関しての解析事例を示す。界面張力は界面のところに過剰に蓄えられた自由エネルギーに相当し、１つの界面が存在する系の自由エネルギーから一様平衡状態での自由エネルギーの値を差し引いたもの（過剰自由エネルギー）として定義される。前述のように、過剰自由エネルギーは各成分のバルク相でのセグメント濃度が既知の場合はグランドカノニカル計算で出力される。但し、この計算においてはバルクでのセグメント濃度をあらかじめフローリー・ハギンス等粗視化理論にて算出しておく必要があるが、多成分系を対象とする場合、自ずとその汎用性に限界がある。ここでは手法の汎用性を考慮しカノニカル計算から求めた濃度プロファイルから界面張力を計算する方法を試みた。

4.1.2　系とパラメータ入力

A/B ホモポリマーブレンド系において、A, B, それぞれのポリマーが、長さの異なる多成分の高分子鎖で構成される場合を考える。１つの界面が存在する平衡構造は、NEUMANN 境界条件の下で、K 種 ($K = A, B$) ポリマーの i 高分子鎖成分の系内の体積分率 ϕ^0_{Ki}、鎖長 N_{Ki}、セグメント間相互作用パラメータ χ_{AB} を与え、SUSHI の一次元の静的カノニカル計算から形成する。なお、外的条件として自己無撞着場の初期値を設定することにより静的平衡構造は簡単にシミュレートできる。SUSHI マニュアル 2.9.14 節参照。計算終了時には

各成分の濃度プロファイル、平衡状態での自由エネルギー値が出力されるので以下の手順で過剰自由エネルギーの評価に供する。

4.1.3　計算の手順

入力用 UDF サンプルファイルとして inputUDF を用意してあるので、プラットフォームにインポートする。これを実行して得られた outputUDF から、A,B 各成分のバルク相 (例えば A-rich α 相) 濃度 ϕ^α_{Ki}、および平衡状態の自由エネルギー $\mathcal{F}$ を読み取り、バルク相の自由エネルギー f^{bulk} 並びに各成分の化学ポテンシャル μ_{Ki} の計算から以下の式に従い過剰自由エネルギー $\mathcal{F}_{exces}$ を計算する。

$$f^{bulk} = \sum_i \frac{\phi^\alpha_i}{N_i} \ln \frac{\phi^\alpha_i}{N_i} + \frac{1}{2} \sum_{ij} \chi_{ij} \phi^\alpha_i \phi^\alpha_j \tag{4.1}$$

$$\mu_i = 1 + \ln \frac{\phi^\alpha_i}{N_i} + N_i \sum_j \left(\chi_{ij} \phi^\alpha_j - \frac{\phi^\alpha_j}{N_j} \right) - \frac{1}{2} N_i \sum_{jk} \chi_{jk} \phi^\alpha_j \phi^\alpha_k \tag{4.2}$$

$$\mathcal{F}_{exces} = L\mathcal{F} - L f^{bulk} - L \sum_i \frac{\mu_i \left(\phi^0_i - \phi^\alpha_i \right)}{N_i} \tag{4.3}$$

ここで、式中のインデックス i, j, k は、ブレンド系すべての高分子鎖 (K 種 i 成分, eg. A1, A2, $\cdots$, B1, B2, $\cdots$) を、L は自己無撞着場計算におけるシステムサイズを表す。以上の計算手順はプラットフォーム上の Python にて適当なスクリプトを与えることで自動化可能であるが、一連の計算手順は Interface Simulator 中にパッ

ケージしているので、Interface Simulator の sample ディレクトリにある入力用 udf ファイル fluid.udf を利用し Fluid Simulator(Interface Simulator の中の計算エンジン) での計算が便利である。

4.1.4 解析例

Broseta[24] 、Anastasiadis[25] らは Cahn-Hilliard 方程式と RPA に基づく自由エネルギー計算、および界面張力の実験的検討から、強相分離系ホモポリマーブレンドの界面張力に及ぼす多分散性の影響を構成ポリマーの数平均分子量で近似的に整理している。実際に、長鎖/短鎖のバイモーダルな同一の分子量分布を持つ A/B ２元系（長鎖/短鎖を区別し４元系）をモデルとして F_{exces} を自己無撞着場計算から求めた。鎖長、系内の体積分率を適当に与えた数種のブレンド系にて数平均分子量の依存性を調べたところ、それぞれの系は、ほぼ同一の曲線にのり、単分散ブレンド系の分子量依存性と高い相関があることが示唆された。また分子鎖長を長くするに連れ Helfand-Tagami[26] の分子量無限大極限における界面張力の解析値に漸近する様子が見られた。これらの計算結果は先述の理論、実験から導かれる予測とは矛盾しないものとみられ、我々の提案した計算スキームは汎用性の高い手法と考えられる。一方、バルクの濃度を基準に短鎖/長鎖それぞれの成分の濃度プロファイルをみると、界面近傍に短鎖成分が過剰に存在し長鎖成分は枯渇する構造が形成されることがわかった。多分散性をもつブレンド系界面における構造形成機構の解析は、界面物性（界面張力、厚み、力学強度 等）を発現する材料設計の上でも重要と考えられ、今後の検討課題であるが、動的平均場計算によるその構造形成過程の追跡は有効といえる。

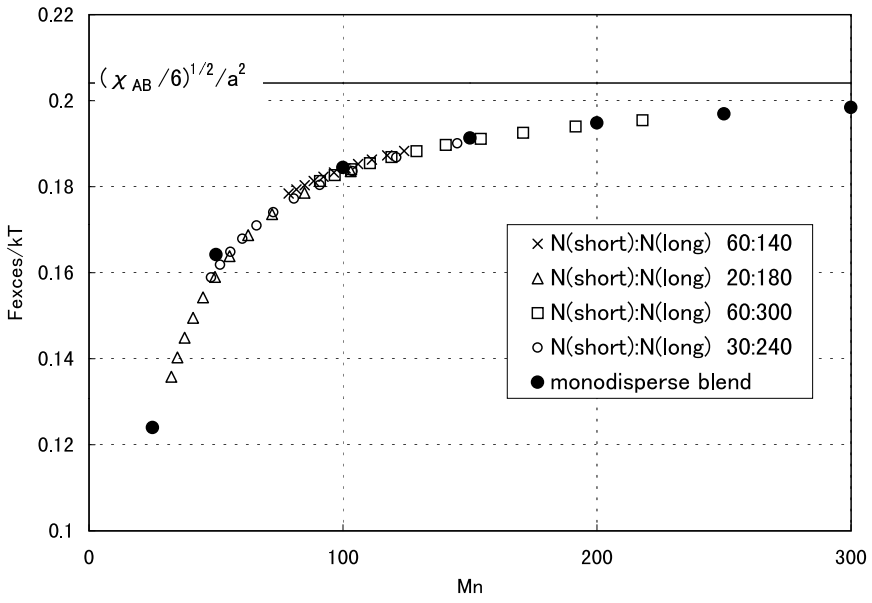


図 4.1: 過剰自由エネルギーの分子量依存性 ($\chi_{AB} = 0.25$)

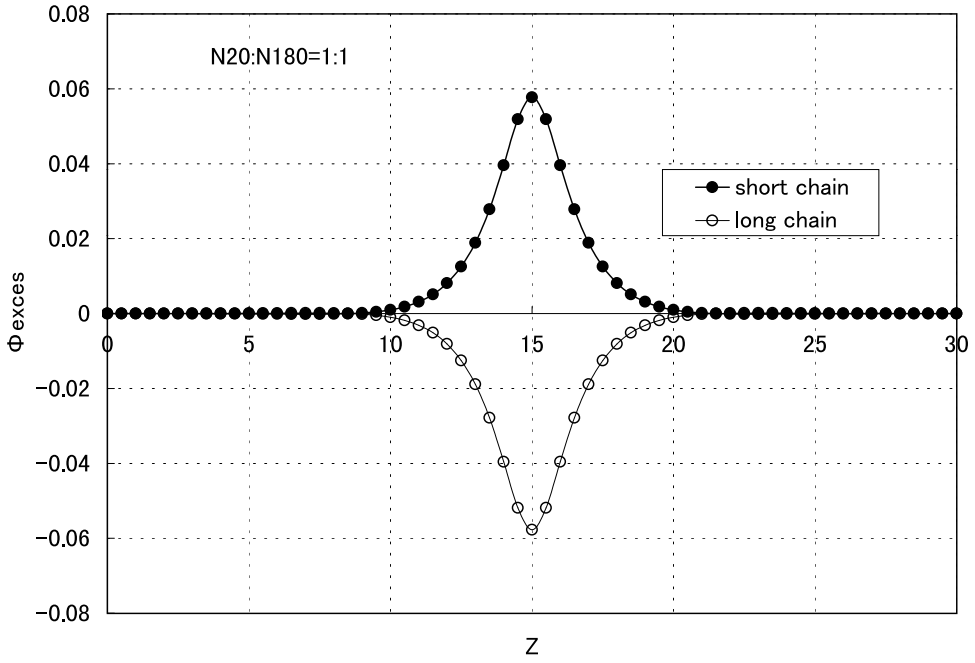


図 4.2: バルク相を基準にした長鎖/短鎖の濃度プロファイル ($\chi_{AB} = 0.25$)

4.2 A-B ジブロックコポリマーのミセル分布の計算

4.2.1 概要

A-B ジブロックコポリマーを選択溶媒に溶かしたときに形成される会合体を、ミセルと呼ぶ。通常、ポリマー濃度が臨界ミセル濃度（cmc）を超えると、均一なミセルが形成され始める。

ここでは、与えられた濃度のジブロックコポリマー溶液におけるミセルのサイズ分布を計算する方法について述べる。簡単のために、AB ジブロックコポリマーと溶媒 C からなる系を考え、ミセルは球状であると仮定する。

一般に AB ジブロックコポリマーをある濃度で溶媒 C に溶かした系を扱うとき、それぞれの体積分率を与えてカノニカル計算を行うことが考えられる。濃度がそこそこ高い場合に、ライオトロピック液晶と呼ばれる規則構造を作る場合には、このような計算は適切である。しかし、ミセル溶液を扱う場合には、不適切となる。これは、ミセルの並進自由度に関するエントロピーの寄与が考慮されないためである。したがって、ミセル溶液を計算するためには次の 2 つの手順が必要となる。

1. 可能なあらゆるサイズ（会合数）の孤立ミセルの構造を SUSHI で計算し、その過剰自由エネルギーの値を求める。
2. ミセル溶液の自由エネルギーをミセル分布の関数として定義し、ポリマー全体の濃度を拘束した条件下でこれをミセル分布に関して最小化する。

以下これらの手順の詳細を述べる。

4.2.2 孤立ミセルの計算

孤立ミセルを扱う場合、ミセルの会合数を固定するために、ブロックコポリマーはカノニカルとする一方、溶媒はグランドカノニカルでバルク濃度を 1 に設定する（溶媒が一成分の場合）。

さらに、安定なミセルを得るために、溶媒を嫌うブロックの末端をある領域内に拘束する。このために、SUSHI に用意されている MASK 機能を用いる。末端を拘束する領域があまり小さすぎると、鎖のコンフォメーションが制限されるため、自由エネルギーを過大に見積もることになる。適切なサイズを見積もるためには、計算する系によってある程度の試行錯誤が必要となる。

A40B10 のジブロックコポリマーと溶媒 C からなる系で、会合数 $p = 30$ の孤立ミセルの静的 SCF を計算した例を示す。χ パラメータはそれぞれのモノマー間に対して $\chi_{AB} = 1.0$、$\chi_{BC} = 0$、$\chi_{AC} = 1.2$ にとってある。計算されたミセルの構造を図 4.3 に示す。また、このミセルの過剰エネルギーを会合数 p で割った量を $f(p)$ とする。

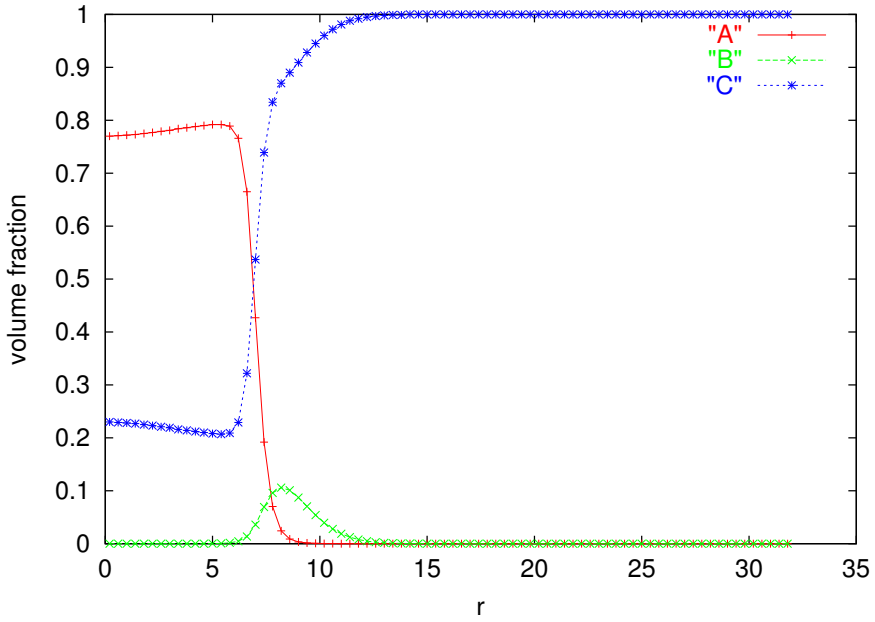


図 4.3: 会合数 30 のミセルにおける各モノマーの体積分率プロファイル

各会合数に対して上記の手続きを行い、計算された $f(p)$ をプロットすると図 4.4 のようになる。会合数 $p = 29$ のとき $f(p)$ は最小となる。この会合数は最適なミセルのサイズの目安となるが、正しくはミセルの並進エントロピーを含んだミセル溶液全体の自由エネルギーを最小化してやらなければならない。次節でその手続きについて述べる。

4.2.3 ミセル分布の計算

ミセル溶液の自由エネルギーは

$$F\left(\{\phi_i\}\right)/k_BT = \sum_p \phi_p \left(\frac{1}{Np} \ln \phi_p + f(p) \right) \tag{4.4}$$

とかける。ここで、ϕ_p は会合数 p のミセルの体積分率、すなわちミセル溶液中のミセルサイズの分布を表す。$f(p)$ として図 4.4 の結果を用い、与えられたコポリマー体積分率が $\phi = \sum_p \phi_p = 0.005, 0.04$ の場合に対して、式 (4.4) を最小化するミセルの分布を求めると、図 4.5 のようになる。コポリマーの体積分率が 10 倍以上増え

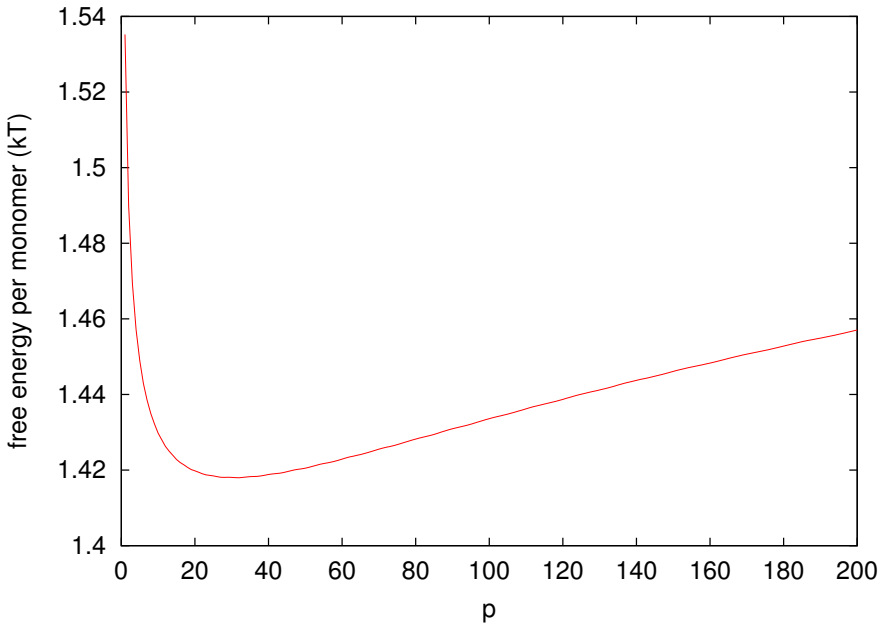


図 4.4: p-会合体を形成しているコポリマーの自由エネルギー

ても単量体の体積分率はほとんど変化しないことがわかる。この単量体濃度が臨界ミセル濃度に相当する。このように、ミセル形成に特徴的な挙動がよく再現されている。

具体的な手続きは python スクリプトを用いて行われる。このスクリプトは、ラグランジュの未定乗数 μ を導入して、$F(\{\phi_i\}) - \mu(\sum_p \phi_p - \phi)$ を最小化するものである。μ と ϕ は一対一に対応するので、一方を与えればもう一方が決まるが、現在の仕様では μ の方を与えるようになっている。したがって、ある ϕ の分布を求めるためには、μ の値を変えながら何度もスクリプトを繰り返す必要がある。将来的には ϕ を与えることにより、それに対応する分布が自動的に得られるようにしたいと考えているが、数回の試行錯誤で目的の ϕ の値に対応する μ を見つけることができる。

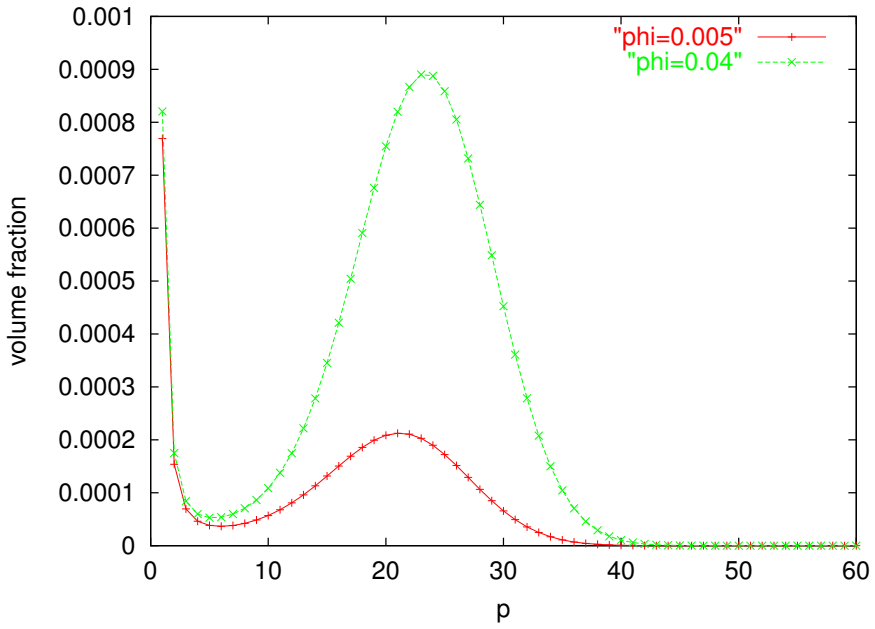


図 4.5: ミセル溶液における会合数の分布

4.3　サンプルデーター覧

sample ディレクトリのデータの内容を説明する。

ab_ring_uin.udf / uot.udf

1D static calculation of a ring A/B block copolymer.

adsorption_uin.udf / uot.udf

Adsorption of a polymer to a solid wall explained in Chapter 3.

blend_uin.udf / uot.udf

Interface of an A/B polymer blend explained in Chapter 3.

blend2D_4_ADF_dy_uin.udf / uot.udf

Simulation result of a 2D dynamic mean-field calculation of the Approximate Density Functional type model on the phase separation of an A/B polymer blend. This method is under development.

blend2D_4_FH_dy_uin.udf / uot.udf

Simulation result of a 2D dynamic mean-field calculation of the Flory-Huggins type model on the phase separation of an A/B polymer blend explained in Appendix C.

blend2D_4_HYBRID_uin.udf / uot.udf

Simulation result of a 2D calculation of the phase separation of an A/B polymer blend by using the hybrid theory.

blend2D_4_RPA_uin.udf / uot.udf

Simulation result of a 2D calculation of the phase separation of an A/B polymer blend by using the GRPA.

blend2D_4_RPA_HYDRO_uin.udf / uot.udf

Simulation result of a 2D calculation of the phase separation of an A/B polymer blend by using the GRPA and hydrodynamics.

blend2D_4_dy1_uin.udf / uot.udf

Simulation result of a 2D calculation of the phase separation of an A/B polymer blend explained in Chapter 3.

blend2D_4_dy1_comp_uin.udf / uot.udf

Simulation result of a 2D calculation of the phase separation of an A/B polymer blend with compressibility.

blend2D_4_dyr1_uin.udf / uot.udf

Simulation result of a 2D calculation of the phase separation of an A/B polymer blend by using the ϕ-dependent mobilities.

blend2D_4_dyr1_const_uin.udf / uot.udf

Restart of blend2D_4_dyr1_uot.udf

blend2D_4_dyr1_rest_uin.udf / uot.udf

Continue of blend2D_4_dyr1_uot.udf

blend2D_4_dy1r_shear_uin.udf / uot.udf

Simulation result of a 2D calculation of the phase separation of an A/B polymer blend with shear.

block_uin.udf / uot.udf

1D static calculation of the lamella structure of a block copolymer melt explained in Chapter 3.

block2D_7_dy1_uin.udf / uot.udf

2D dynamic calculation of the phase separation of a block-copolymer melt.

block2D_7_lamella_uin.udf / uot.udf

2D static calculation of the lamella structure of a block copolymer melt.

block2D_8_dy1_lo_uin.udf / uot.udf

2D dynamic calculation of the phase separation of a block-copolymer melt with system size optimization.

block2D_8_dy1_lo_comp_uin.udf / uot.udf

2D dynamic calculation of the phase separation of a block-copolymer melt with system size optimization and compressibility of system.

block2D_8_dy1_lo_comp2_uin.udf / uot.udf

2D dynamic calculation of the phase separation of a block-copolymer melt with system size optimization and compressibility in SCF.

block2D_8_dy1_lo_comp_comp_uin.udf / uot.udf

2D dynamic calculation of the phase separation of a block-copolymer melt with system size optimization, compressibility of system, and compressibility in SCF.

block2D_8_lo_uin.udf / uot.udf

2D static calculation of the phase separation of a block-copolymer melt with system size optimization.

block_el_uin.udf / uot.udf

1D static calculation of the lamella structure of a block copolymer melt, ends of which are strong polyelectrolyte.

block_el3_uin.udf / uot.udf

1D static calculation of the lamella structure of a block copolymer melt, ends of which are strong polyelectrolyte with dielectric constants of segment.

comb2D_16_uin.udf / uot.udf

2D static calculation of the equilibrium structure of a comb copolymer melt.

comb3D_16_uin.udf / uot.udf

3D static calculation of the equilibrium structure of a comb copolymer melt.

cylinder3D_a_uin.udf / uot.udf

3D static calculation of the cylinder structure of a block copolymer melt by the domain specification method. Explained in Chapter 3.

depletion_uin.udf / uot.udf

Depletion of a polymer near a solid wall explained in Chapter 3.

graft_uin.udf / uot.udf

A polymer grafting to a wall explained in Chapter 3.

graft_dy_step0_uin.udf / uot.udf, graft_dy_uin.udf / uot.udf

A sample input UDF for the dynamics of a graft reaction. Restart calculation is performed according to the input UDF file graft_dy_step0_uot.udf.

interface_uin.udf / uot.udf

Interface of an A/B polymer blend explained in Chapter 3.

micelle_uin.udf / uot.udf

1D static calculation of a micelle.

montecarlo_uin.udf / uot.udf

2D Monte Carlo calculation of a comb copolymer in a solution.

nonpolyd_uin.udf / uot.udf, polyd_uin.udf / uot.udf

1D static calculation for polydisperse polymers near a solid wall. In the "nonpolyd" case, the symmetry in the path integral is not taken into account in the calculation. One can use these two UDF files to compare the efficiency of the calculation using the symmetry in the path integrals.

quench_uin.udf / upm.udf / uot.udf

Simulation result of a 2D calculation of the phase separation of an A/B polymer blend with time dependent χ parameter. The χ parameters are recorded in the parameter file.

solventSol_eq_uin.udf / uot.udf

2D static calculation of the equilibrium state of a mixture of a solvent and a polymer.

star2D_8_uin.udf / uot.udf

2D static calculation of the equilibrium state of a star-type copolymer melt. The Python script show3color.py can display the density distributions of the three components with different colors.

star3D_8_uin.udf / uot.udf

3D static calculation of the equilibrium state of a star-type copolymer melt. The Python script show3color.py can display density distributions of the three components with different colors.

taper2D_12_uin.udf / uot.udf

2D static calculation of the equilibrium state of a tapered polymer melt.

taper3D_12_uin.udf / uot.udf

3D static calculation of the equilibrium state of a tapered polymer melt.

testRg_uin.udf / uot.udf

An example of the calculation of R_g. 1D static calculation of R_g of a chain whose one end is fixed at a point in a solvent.

testRg2_uin.udf / uot.udf

An example of the calculation of R_g. 1D static calculation of R_g of a full chain whose one end is fixed at a point in a solvent.

triblock2D_7_uin.udf / uot.udf

2D static calculation of the equilibrium structure of a triblock copolymer melt. The Python script show3color.py can display the density distributions of the three components with different colors.

EZ1a_uin.udf / uot.udf

2D dynamic calculation under a weak external electric field. Treatment of the weak effect of external electric field.

EZ2a_uin.udf / uot.udf

2D dynamic calculation under an external electric field. General treatment of the effect of external electric field.

EZ3a_uin.udf / uot.udf

2D static calculation under an external electric field. General treatment of the effect of external electric field.

particle1_uin.udf / uot.udf

2D static calculation with a particle.

particle2_uin.udf / uot.udf

2D static calculation with particles.

particle2B_uin.udf / uot.udf

3D static calculation:diblock copolymer in a sphere

cylinder_block_uin.udf / uot.udf

Diblock copolymer in cylindrical mesh.

第5章　リファレンス

5.1　SUSHIとは

SUSHI はコンソールからコマンドを打ち込んで起動させるプログラムである。Windows であれば DOS 窓から、UNIX 系であれば種々の Shell を用いてコマンド入力により入力テキストファイルを読み込ませて利用する。SUSHI は GUI を実装してないが、UDF 入力ファイルを利用し、GOURMET に登録して使用することにより GUI が備わったプログラムのように利用することが可能である。

5.2　SUSHIのファイル構成

SUSHI ディレクトリーの構成は次のとおりである。

```
OCTA/PF_ENGINE/SUSHI10.54/
       Susi--+--def_udf--+--SUSHIInput.udf     : definition of input UDF format
             |           +--SUSHIOutput.udf    : definition of output UDF format
             |           +--SUSHIParameter.udf : definition of parameter UDF format
             |           +--relax_in.udf, restart_in.udf : COGNAC7 input UDF file
             |
             +--include : include files
             |
             +--sample---+--Input      : sample inputs           ( sec. 5.5-7 )
             |           +--Input_V2   : sample inputs version 2 ( sec. 5.12 )
             |           +--Output     : sample outputs          ( sec. 5.8-10 )
             |           +--Seed_Input : sample inputs by SEED   ( sec. 5.14-15 )
             |
             +--src     : source files
```

SUSHIInput.udf は入力ファイルを作成するときに使用する。SUSHIOutput.udf は特に必要になることはないが、出力データ構造を調べるときに利用できる。

実行可能なプログラムは次のとおりである。

```
OCTA/PF_ENGINE/SUSHI10.54/
Susi/bin----+--cygwin--------+--sushi.exe : load module for Cygwin
            |
            +--k-------------+--sushi : load module for K-computer(compile option)
            |
            +--linux---------+--sushi : load module for 32 bit Linux
            |
            +--linux_64------+--sushi : load module for 64 bit Linux
            |
            +--aix-----------+--sushi : load module for AIX(compile option)
            |
            +--win32---------+--sushi.exe : load module for 32 bit windows
            |
            +--x64-----------+--sushi.exe : load module for 64 but windows
```

並列計算用実行モジュールは次の名前となる。

sushiPTL: Pthread Library 利用共有メモリー型並列版

sushiGPU: NVIDIA GPU 用

sushiMPI: MPI 並列用

5.3 起動方法

SUSHI は設計上、種々の入力インターフェースに対応できるようになっており、現在 UDF フォーマットと SEED フォーマットによる入力が可能である。UDF フォーマットは GOURMET から起動するのに用いる。冗長な入力ファイルとなるが、その分、データ構造が明白で入力ミスが少なくなる。UDF フォーマットの詳細については UDF マニュアルを参照のこと。SEED フォーマットは入力が簡単だが、入力コマンドを自分で打ち込む必要がある。SEED フォーマット用のファイルは UNIX 形式（行末が LF で終わるもの）を使用しなくてはならない。

コンソールからの起動方法

エンジンをコンソールから起動する場合は以下のようになる。

```
> sushi -I full_input_file_name
        -O full_output_file_name
        -R full_restart_file_name -rrecord_number_for_restart [-w]
        -L scf_log_file_name_of_dynamics
        -M message_file_name
        -C calculation_profile_file_name
        -A archives_of_latest_record_file_name
        -S summary_of_calculation_file_name
        -P control_parameter_file_name
        -Z special_COGNAC_input_file_name
       -n number_of_core_for_pthread_calculation
        -b number_of_threads_per_block_for_CUDA
        -d parameter_for_GPU_device
```

ここで , 大文字のフラグ-I,-O,-R,-L,-M,-C,-A,-S,-P,-Z の後ろはファイル名でなければならない。以下に sushi

の引数の説明をする。

入力　　-I 入力ファイル名

UDF フォーマット入力の場合、拡張子は、.udf でなくてはならない。かつ、内容は SUSHIInput.udf ファイルの定義に従ったものでなくてはならない。入力ファイルであることを区別するために "_uin.udf" が末尾に付いたファイルを入力 UDF ファイルとすることにする。であるから、ファイル名の形式は、
input_file_name_uin.udf　となる。
SEED フォーマット入力の場合拡張子は　.sin　でなくてはならない。ファイル名の形式は、
input_file_name.sin　となる。

出力　　-O 出力ファイル名

SUSHIOutput.udf の定義によるファイルを出力する。
省略すると　input_file_name_uot.udf　ファイルが出力される。

リスタート　-R リスタートファイル名

SUSHIOutput.udf の定義によるファイルを指定する。
省略すると　input_file_name_uot.udf　となる。

-r リスタートレコード数

レコード数とは出力ファイル中のレコード出力の冒頭にある
\begin{record}{"Step n"}
と記述された行における n の値である。動力学計算では最終レコードが記録用ファイルとして出力される。これを利用してリスタートするとリスタートが速い。

-w

UDF のパーシング無しに直接ファイルから必要なデータを読み取るオプションである。SUSHI が直接出力した UDF ファイルに対して有効である。
このオプションは非常に大きなファイルを読み込む場合、効果的である。
このオプションを利用してリスタートすることを推奨する。

SCF ログ　　-L ダイナミクスステップ用 SCF ログファイル名

ダイナミクスの各ステップにおける SCF の収束状態のログが出力される。
レコード毎に上書きされ更新される。
出力がディスクシステムへの負担になる場合、UNIX 系の OS の場合/dev/null、Windows の場合 nul を設定すれば出力を抑制できる。
省略すると　input_file_name.ual　ファイルが出力される。

ログ　　-C 計算進行状況出力ファイル名

計算進行状況の記録が出力される。
出力間隔は入力ファイルで変更可能。
省略すると標準出力へ出力される。

アーカイブス　-A 最終レコード記録用ファイル名

最終レコードが毎レコードこのファイルに上書きされ更新される。
出力間隔は入力ファイルからの入力で変更可能。
省略すると　input_file_name_uar.udf　ファイルが出力される。

メッセージ　-M message_file_name

コントロールメッセージ（ RESUME, RUN, STOP, etc. ）をこのファイルから読み取る。-P オプションと併用する場合が多い。
詳しくは GOURMET のマニュアルを参照のこと。

サマリー -S summary_of_calculation_file_name
SUSHI の実行中の途中結果をファイルに出力する。
このファイルは GOURMET により読み込まれ、グラフ出力される。
詳しくは GOURMET のマニュアルを参照のこと。

サマリー -m maximum_number_of_summary_data
最大数 上記データの最大数。グラフの横軸のサイズを指定することになる。
グラフは図 5.2 に示すものである。デフォルト値は 100 である。

CPU 数 -n number_of_CPU
PTL=ON のオプションを利用してコ make するとこのオプションが利用可能となる
PTL 版の実行モジュールが作成される。
number_of_CPU の数のスレッドを利用し、計算を開始するので、複数コアの環境
で実行すると計算が高速化される。
（だだし、並列化効率は未だ低いので、大きな系でしか効果がない。）

スレッド数 -b number_of_threads_per_block_for_CUDA
GPU 計算における 1 ブロック当たりのスレッド数（デフォルトは 512）

GPU パラメータ -d parameter_for_GPU_device
[デバイス ID の並び].[デバイスメモリへの係数]
012.9 のように入力すると、デバイス 0,1,2 の順に
[各デバイスメモリー量] × 0.9 となるまで当該のデバイスをメインに利用する。
全デバイスのメモリーは選択されているデバイスが全て利用する。
最後のデバイスは、メモリー利用量制限を超しても利用し続けられる。
物理的なデバイスメモリー量を超せば、ジョブは当然停止する。
8 MPI 計算の場合、全ての GPU を利用する（デフォルト：id0 のみ）
17 すべてのデバイス名を表示し停止する。

パラメータ -P control_parameter_file_name
SUSHI は計算途中にこのファイルから計算用のパラメータを読見込むことが
できる。操作は GOURMET 上で行う。
第 5.13 節を参照のこと。また、詳しくは GOURMET のマニュアルを参照のこと。

停止 -K kill_file_name
このオプションを利用した場合、指定したファイルが SUSHI を実行している
ディレクトリー上に存在した場合 SUSHI は計算途中の結果をファイルに出力
し停止する。

入力定義 UDF -i
入力定義 UDF ファイル SUSHIInput.udf の内容を標準出力へ出力し停止する。

出力定義 UDF -o
出力定義 UDF ファイル SUSHIOutput.udf の内容を標準出力へ出力し停止する。

SEED 入力 -s
input_file_name.sin の内容をアルファベット順に並び替えて標準出力へ出力し
停止する。

ステージング -t mode
ステージングとは MPI 大規模計算用に入出力ファイルを分割する操作を言う。
このオプションは主に MPI バージョンのみに有効である。非常に大規模計算
になると通常代表となる rank0 のノードのメモリー資源でグローバル領域
（全領域）のデータを保持できない。よって、グローバル領域に分割すること
でメモリー資源の抑制を図るのがステージングの主目的である。
mode により次の機能を起動できる。

---

mode
  0 ログファイルのみ出力する。大規模計算実行確認用。
  1 通常のステージング。グローバル領域を利用しない。
  2 グローバル領域からローカル領域に分割する。つまり scatter。
  3 ローカル領域をグローバル領域に集約する。つまり gather。

ここで、ローカル領域は、rank 番号:000000,000001,000002.....
をファイル名に伴う UDF ファイルとして入出力される。
その他のオプションは -h オプションにて出力されるので参照してください。
注意：大規模計算はテキストファイルでの入力ファイル作成が便利であること
から、ステージング機能の利用は主に SEED 入力の利用を前提としている。

SEED 入力変換 -u
SEED フォーマットの入力ファイル input_file_name.sin が読み込まれた場合 UDF フォーマットに変換し input_file_name_uin.udf ファイルへ出力し停止する。

version -v
SUSHI のバージョンを出力し停止する。

help -h
SUSHI の利用方法を出力し停止する。

ズーミング -Z special_COGNAC_input_file_name
自動ズーミングのための COGNAC の雛形ファイル
relax_in.udf もしくは restart_in.udf（def_udf に添付する特殊なもの)。
relax_in.udf を利用すると、自動的にビーズ用セグメント濃度分布を SUSHI が計算するが、ビーズの配座発生は COGNAC 側での計算となる。
restart_in.udf を利用すると、自動的にビーズ用セグメント濃度分布の計算とビーズの配座発生を SUSHI が行い、COGNAC 用の入力ファイルを作成する。
新たに COGANC 用に
relax_input_file_in.udf もしくは relax_restart_file_in.udf
ファイルが作成される。

SEED フォーマット入力がなされると自動的に UDF 入力用ファイル
input_file_name_uin.udf、input_file_name_uinv2.udf と初期値が入力されたパラメータファイルが作成される。

input_file_name_uinv2.udf ファイルは、SUSHIInputV2.udf フォーマットで出力された入力フォーマットである。詳しくは付録を参照して頂きたい。

MPI を利用した起動方法

```
> mpiexec/mpirun -n number_of_processes sushiMPI -I.....
```

のように実行する。 mpiexec か mpirun かは、コンパイル時の MPI ライブラリーに依存する。詳細は MPI の利用方法を調べて欲しい。

number_of_processes と MPI 用の軸の分割数の積が一致しないと実行は停止する。各軸は MPI 用に必ず分割されなければならない。軸の分割数の入力方法は Mesh UDF の説明を参照してほしい。

GOURMET からの起動方法

GOURMET では ”Engine”や ”エンジン ”という言葉をシミュレータの意味として使う。UDF 内でも同様な意味で使用する。以降で文中にあるエンジンとはシミュレータを意味することとする。

GOURMET を起動して Tool/Engine Run... を選択する。下図に示す窓が出力される。

Engine:

に 64 ビット Windows の場合は x64 版 sushi を、他の OS の場合は OS に合った sushi を入力する。

Input UDF:、Output UDF:

にそれぞれ入力 UDF ファイル名と出力 UDF ファイル名を入力する。また、他のファイル名に対しても適当な名前を入力する。

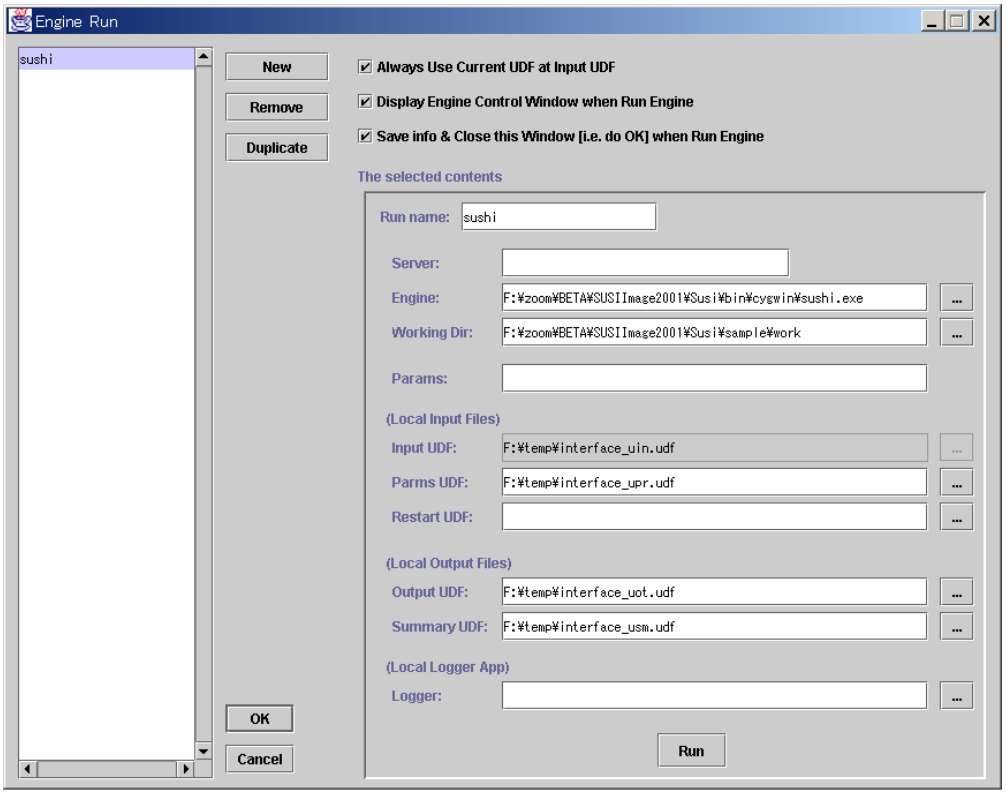


図 5.1: Engine run window

Run ボタンでエンジンが起動する。起動後、次の図のような Engine Control 窓が出力されるので計算の経緯をモニターすることができる。グラフの横軸は-m オプションで変更できる。Engine Control の詳細については GOURMET のマニュアルを参照のこと。

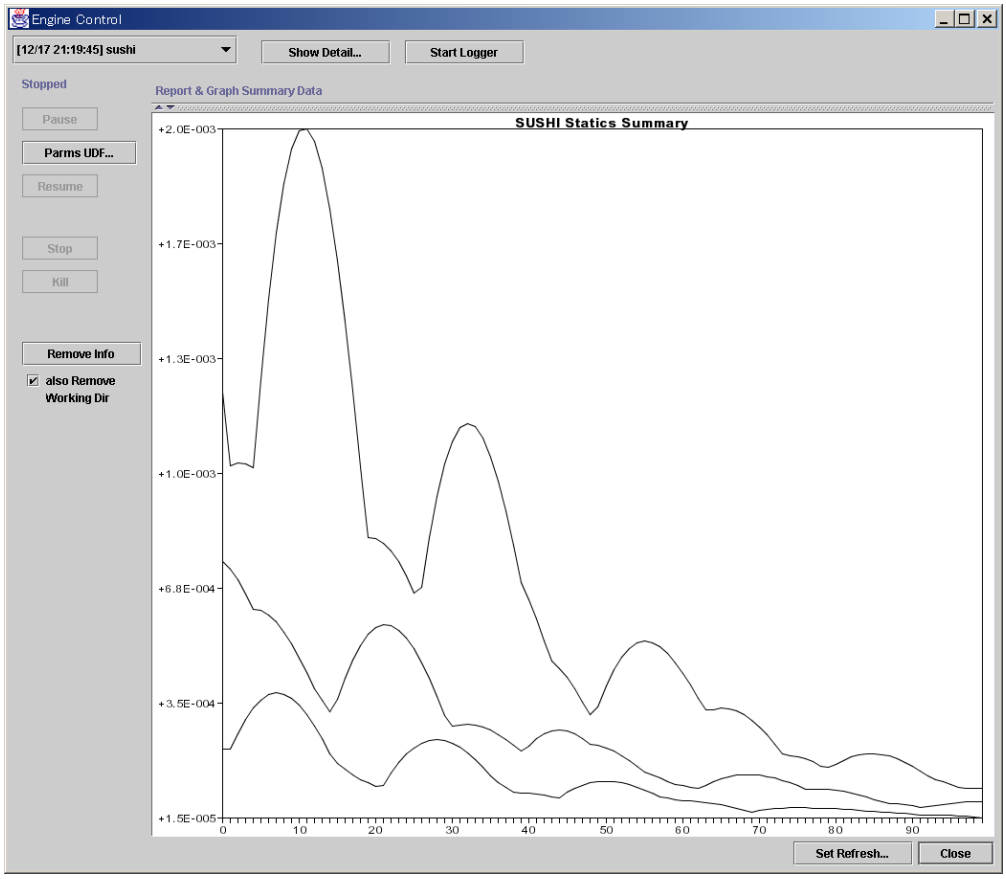


図 5.2: エンジン制御用窓

5.4 停止方法

-K オプションで指定したファイルを、内容は何でもよいから、SUSHI を実行しているディレクトリ上に存在させると、静的平衡計算の場合 SCF の合間で、動力学計算の場合はダイナミクスのステップの合間で SUSHI はその時点での結果を出力して停止する。または、Engine Control 窓からメッセージファイルに停止メッセージを送ると同様に停止する。詳しくは GOURMET のマニュアルを参照のこと。

5.5 入力 UDF ヘッダー

UDF ヘッダー部は解析エンジンの種別を表し、下記の形式をしている。

```
\begin{header}
\begin{def}
        EngineType:string;
        EngineVersion:string;
        IOType:string;
        ProjectName:string;
\end{def}
\begin{data}
        EngineType:"SCFEngine";  // Engine type (reserved keyword)
        EngineVersion:"141205";  // Engine version (reserved keyword)
        IOType:"IN";             // Input-and-output type (always "IN" for input UDF files)
        ProjectName:"WG2";       // Project name (reserved keyword)
```

```
\end{data}
\end{header}
```

SUSHI の本バージョンでは、常に次の文字列を用いる。

```
EngineType:     "SUSHI";
EngineVersion:  "141205";
IOType:         "IN";
```

5.6 入力 UDF 定義

SUSHI の入力項目は以下のような構造をとる。以降のデータの説明で//から始まる文字列はコメントである。デフォルト値が設定されているデータもあるが、入力を明白にすることが重要であるので、必要なデータは全て入力することを推奨する。

```
SUSHIInput:{
  // Parameters for controlling the calculations ################################
    calculation_method:CalculationMethod // The parameter for calculation method.
    start_condition:select { "START", "CONTINUE", "RESTART", "RESTART_READMESH" }
        //  START    標準的なスタート。
        //  CONTINUE 計算を継続する。
        //  RESTART  リスタートする。メッシュ形状は読み込まない。
        //  RESTART_READMESH リスタートする。メッシュ形状を読み込む。
    solver_parameter:SolverParameter // The parameter for solver.
  // Mesh and boundary condition ################################################
    mesh:Mesh
    type_of_free_propagator_of_regular_mesh:select
                                          { "1NN-P","2NN-NP","2NN-P","3NN-P" }
        //  離散化ラプラス演算子のタイプ。
        //  3.5 節参照。
    boundary_condition:BoundaryCondition
        //  2.7.4 節参照。
  // Polymer and solvent ########################################################
    monomers[]:Monomer
        // モノマーの配列
    monomer_SCF_character_table[]:MonomerSCFChar
        // モノマーの特性の配列
        // This parameter is required only when the monomer has the internal
        // モノマーが内部状態を持つ場合のみ必要。内部状態とは傾斜構造等をいう。
        // 2.7.2 節参照。
    components:Components
        // 選択できるコンポーネント（ポリマーおよび溶媒）の集合。
        // 注意
        //   入力に利用されるポリマーと溶媒それぞれの ID は
        //   components.polymers[] と components.solvents[] の配列の番号に対応する。
    volume_fractions: VolumeFractions
        // 系内に存在させるコンポーネントの体積分率。
        // 注意
```

```
        //    ここで体積分率が定義されたコンポーネントのみが計算に利用される。
    chi_parameters[]:ChiParameter
        // χパラメータの配列。
  // 出力を希望する物性。 ##########################################################
    properties:Properties
        // Specify the physical quantities that should be output.
  // 外的条件 ######################################################################
    external_conditions:ExternalConditions
        // External conditions to the system.
}
```

5.6.1 SUSHIInput 内の重要なデータ構造（class)

SUSHIInput.calculation_method

```
class CalculationMethod:{
    type:select { "STATICS", "DYNAMICS", "MONTECARLO" }
    DYNAMICS:DynamicsParameter
    MONTECARLO:MonteCarloParameter
}
```

SUSHIInput.calculation_method.DYNAMICS

```
class DynamicsParameter:{ // These parameters are valid when the type is DYNAMICS.
    delta_t:double                //  ダイナミクス計算用時間刻み幅
    variable_delta_t:{//  VariableDelt 時間刻み幅を自動的に変化させる場合のパラメータ
        max_delta_t:double        //  時間刻み幅の最大値
        variable_coef:double      //  時間刻み幅更新のための係数：
                                  //  delta_t を初期入力としてダイナミクスを開始する。
                                  //  (1) セグメント密度の更新をする。
                                  //  更新されたセグメント密度が全ての場所で正であるなら
                                  //    delta_t = variable_coef * delta_t
                                  //    とする。もし、
                                  //    delta_t > max_delta_t なら
                                  //    delta_t = max_delta_t とする。
                                  //    (2) に進む。
                                  //  更新されたセグメント密度に負の場所があるなら
                                  //    delta_t = delta_t / variable_coef
                                  //    として (1) へ戻る。
                                  //  (2)  次のダイナミクス・ステップへ進む。
    }
    max_dynamics_step:int         //  最大ステップ数。
    output_interval_step:int      //  出力ステップ間隔。
```

```
    archives_interval_step:int    // The output step interval to archives file.
                                  // The default value is the same as output_interval_step.
    log_interval_step:int         // The output step interval for log.
                                  // The default value is 1.
    dynamics_scheme:select { "EXPLICIT", "EXPLICIT2", "IMPLICIT" } // The default value is EXPLICIT.
            // 計算で利用されるスキーム。
            //  EXPLICIT  陽的解法。通常はこちらでよい。Euler 法
            //  EXPLICIT2 陽的解法の 2 次。2 段階 Runge-Kutta 法
            //  IMPLICIT  陰的解法。
    compressibility:double        // 圧縮率。0 の場合完全な非圧縮系となる。
}
```

SUSHIInput.calculation_method.MONTECARLO

モンテカルロ計算用のパラメータである。モンテカルロ計算を行うにはこのパラメータに加え SUSHIInput.external_conditoins.monte_carlo_conditions のデータも入力する必要がある。詳しくは 2.7.17 節と 5.7.11 節を参照のこと。

```
class MonteCarloParameter:{ // These parameters are valid when the type is MONTECARLO.
    max_monte_carlo_step:int      // The total numbers of the Monte Carlo steps.
    output_interval_step:int      // Output step interval.
    archives_interval_step:int    // The output step interval to archives file.
                                  // The default value is the same as output_interval_step.
    log_interval_step:int         // The output step interval for log.
                                  // The default value is 1.
}
```

SUSHIInput.solver_parameter

```
class SolverParameter:{ // The ‘‘solver" means a special-purpose simulator/simulation method.
    type:select { "ADF", "FH", "RPA", "SCF" }
    SCF:SCFParameter // SCF (Self Consistent Field) method.
    RPA:RPAParameter // RPA (Ginzburg-Landau using Random Phase Approximation)
    ADF:ADFParameter // ADF (Approximate Density Functional) method.
                     // (Under construction)
    FH:FHParameter   // Cahn-Hilliard type dynamics method.
                     // sample program, Refer to Appendix.
}
```

SUSHIInput.solver_parameter.SCF

```
class SCFParameter:{
        delta_s:double                  //  経路積分用の長さの刻み幅。値は正。方向はない。
        constV:double                   //  化学ポテンシャル更新用定数。(2.55) 式参照。
        constW:double                   //  セグメント間相互作用更新用定数。(2.54) 式参照。
        error:double                    //  収束判定用しきい値。
        random_seed:int                 //  乱数発生用シード。
        standard_deviation:double       //  場の初期値に与えるノイズの標準偏差。
        method_of_convergence_test:string  //  収束判定方法。ダイナミクス計算でサポート。
             //  ABSOLUTE φの絶対誤差で収束判定する。
             //  RELATIVE φの相対誤差で収束判定する。ABSOLUTE より判定は厳しい。
        convergence_test_interval_step :int  //  収束判定のステップ間隔。
             //  デフォルトは 0 であるが、0 の場合は 1（毎ステップ）として認識される。
        max_SCF_step:int                //  SCF 計算における最大繰り返し回数。
        scf_output_interval_step:int    //  SCF 計算中の出力ステップ間隔。
        SCF_method:string               //  経路積分の取り扱い方法。:
            //  INCORE  メモリーに保存。
            //  DIRECT  必要な場合に再計算をする。
        pathintegral_scheme:string      //  経路積分の差分スキーム。
            //  explicit  陽的解法。通常はこちらでよい。
            //  implicit  陰的解法。
}
```

SUSHIInput.solver_parameter.RPA

```
class RPAParameter:{ //  GRPA 用パラメータ  2.6 節を参照のこと。
        random_seed:int                 //  乱数発生用シード。
        standard_deviation:double       //  場の初期値に与えるノイズの標準偏差。
        // 注意  GRPA 法はダイナミクス計算でしか利用できない。
        //       計算の安定化のために、
        //       移動度は ROUSE もしくは REPTATION タイプ選ぶとよい。5.7.9 節
        //       variable_delta_t を選択するとよい。
}
```

SUSHIInput.mesh

```
class Mesh:{
    name:KEY
    type:select { "REGULAR", "RECTANGULAR", "CYLINDRICAL", "SPHERICAL" }
            //  REGULAR       規則メッシュ。
            //  RECTANGULAR   直方メッシュ。
            //  SPHERICAL     球座標メッシュ。
```

```
            //  CYLINDRICAL   円筒座標メッシュ。
    axes[]:MeshAxis                // Array of mesh axes.
        // Array of mesh axes.
        // Its dimension is the total number of coordinate axes of the system.
    index_rule[]:int
        // This specifies how the array elements (i, j, k) are arranged on the memory.
        // The first argument i runs first for example (0,0,0),(1,0,0)..(X-1,Y-1,Z-1).
        // For SUSHI, this is fixed as [0, 1, 2] except cylindrical mesh and
        // is fixed as [2, 1, 0] for cylindrical mesh.
        // Although this parameter does not affect the functions of SUSHI,
        // it is used when the mesh data is passed to another simulator and
        // is passed to the viewer on GOURMET.
}
```

SUSHIInput.boundary_condition

```
class VolumeFractionsOnBoundaries:{
    polymer_volume_fractions[]:VolumeFractionOnBoundary
    solvent_volume_fractions[]:VolumeFractionOnBoundary
}
class BoundaryCondition:{
    conditions[]:AxisBoundaryCondition  // An array of the boundary conditions for each
                                        // axis.
    volume_fractions_on_boundaries:VolumeFractionsOnBoundaries
}
```

SUSHIInput.components

```
class Components:{
    polymers[]:Polymer // Array of polymers.
    solvents[]:Solvent // Array of solvents.
}
```

SUSHIInput.properties

```
class Properties:{
    segment_volume_fraction_conditions[]:SegmentVolumeFractionCondition
        // A セグメントの体積分率条件の配列。
    radius_of_gyration_conditions[]:SubchainUnit
        // 慣性半径 Rg  条件の配列。
    scattering_function:ScatteringFunctionInput
```

```
        // 散乱関数用フラグ
}
```

SUSHIInput.external_conditions

```
class ExternalConditions:{
    surface_chi_parameters[]:SurfaceChiParameter
        // 壁との相互作用パラメータχsの配列。
    graft_conditions[]:GraftConditi          // グラフト条件の配列。
    mask_conditions[]:MaskCondition
        // マスク条件の配列。自由端の存在可能領域の設定。
    static_conditions:StaticConditions
        // 静的平衡計算で有効な条件。
    dynamic_conditions:DynamicConditions
        // 動的な計算で有効な条件。
    monte_carlo_conditions:MonteCarloConditions
        // モンテカルロ計算で必要な条件。
    electrostatic_conditions:ElectrostaticConditions
        // 強高分子電解質の入力条件。
    obstacles:Obstacles
        // オブスタクルの入力条件。
}
class StaticConditions:{
    polydispersity_conditions[]:PolydispersityCondition
        // 多分散静的計算用に最大鎖長を指定する配列。
    symmetry_conditions[]:SymmetryCondition
        // 対称性のある経路を指定する配列。
    domain_specification_conditions[]:DomainSpecificationCondition
        // 自己無道着場の初期値設定によるドメイン領域設定の配列。
    constraint_conditions[]:ConstraintCondition
        // SCF 拘束条件の配列。
    system_size_optimization:SystemSizeOptimaization
        // システムサイズを最適化するための入力条件。
}
class DynamicConditions:{
    segment_mobilities[]:SegmentMobility
        // 易動度の配列。
    polymer_mobilities[]:LocalMobility
        // Array of the mobilities that depend on the volume fraction of the polymer.
        // Refer sec. 2.6 and 2.7.10.
    solvent_mobilities[]:LocalMobility
        // Array of the mobilities that depend on the volume fraction of the solvent.
        // Refer sec. 2.6 and 2.7.10.
    reaction_conditions_of_rapid_reactions[]:ReactionConditionOfRapidReaction
```

```
    // 反応速度が速い反応条件の配列。
  reaction_conditions_of_active_sites[]:ReactionConditionOfActiveSites
    // 分子構造が変化する反応条件の配列。
  reaction_conditions_of_grafts[]:ReactionConditionOfGraft
    // グラフト反応条件の配列。
  shear:Shear
    // ずり流動の入力条件。
  noise:Noise
    // 熱ゆらぎの入力条件。
  system_size_dynamics:SystemSizeDynamics
    // システムサイズを最適化する入力条件。
  hydrodynamics_parameters:HydrodynamicsParameters
    // 流体力学的効果の入力条件。
  hybrid:Hybrid
    // ハイブリッド法の入力条件。
```

5.7 入力 UDF の詳細

5.7.1 メッシュ

第 2.7.3 節参照。

メッシュ軸

SUSHIInput.mesh.axes[]

```
MeshAxis:{
  values[]:double
    //  regular メッシュ の場合
    //  軸の最小値，最大値，分割数，MPI での領域の分割数（MPI 版でのみ有効）
    //  spherical，cylindrical メッシュ の場合
    //  軸の最小値，最大値，分割数。
    //  rectangular メッシュ の場合
    //  メッシュ点の座標の配列。
}
```

メッシュ軸の境界条件

SUSHIInput.boundary_condition.conditions[]
第 2.7.4 節参照。

SUSHIInput.boundary_condition

境界上での体積分率の拘束条件

SUSHIInput.boundary_condition.polymer_volume_fractions[]
SUSHIInput.boundary_condition.solvent_volume_fractions[]
この入力により擬似的なグランドカノニカルアンサンブルのダイナミクス計算が可能となる。そのためには、カノニカルアンサンブルのダイナミクス用に入力を用意し、外系に解放したい壁を反射壁として設定し、その上での体積分率をバルクの体積分率に拘束する。SUSHI はダイナミクスの各ステップ毎に境界上の体積分率が拘束されるように系内の体積分率を補正する。この操作により系内は外系に解放された状態となる。系内とバルクの体積分率の入力方法は以降で述べる第 5.7.4 節を参照のこと。

```
class VolumeFractionOnBoundary:{
    ID:int                  //  コンポーネントの ID
    boundary_name:string    //  境界の名前
      //  XMin X 軸の最小値における YZ 平面。
      //  XMax X 軸の最大値における YZ 平面。
      //  YMin Y 軸の最小値における ZX 平面。
      //  YMax Y 軸の最大値における ZX 平面。
      //  ZMin Z 軸の最小値における XY 平面。
      //  ZMax Z 軸の最大値における XY 平面。
      //  RMin R 軸の最小値における境界面。
      //  RMax R 軸の最大値における境界面。
    volume_fraction:double  //  境界上での体積分率
}
```

5.7.2 モノマーの定義

第 2.7.1 節参照。

モノマー

SUSHIInput.monomers[]

```
Monomer:{
    species_name:string             // モノマーの名前。
        //  内部状態をもつ場合、例えば、A、B 2 成分からなる傾斜濃度のポリマーの場合は
        //  "taperAB" のように名前をつけておく。
    specific_volume:double          // 有効体積。
        //  経路積分とモノマーの体積分率の計算に利用される。
        //  内部状態をもつ場合は各状態の 有効体積 から再計算され上書きされる。
    effective_bond_length:double  // 有効結合長。
        //  経路積分の拡散係数の計算に利用される。(2.55) 式参照。
}
```

部分鎖の内部状態

SUSHIInput.monomer_SCF_character_table[]
部分鎖が内部状態をもつ場合はその特性を入力する。

```
MonomerSCFChar:{
    name:string    //  名前。
        //  この名前と同名の名前の Monomer が定義されなくてはならない。
        //  例えば、A、B 2成分からなる傾斜濃度のポリマーは "taperAB" のように。
    states[]:State //  内部状態の配列。
                   //  内部状態間遷移確率の配列。
    array_of_transition_state_probabilities[]:TransitionStateProbabilities
}
```

ステイト（部分鎖の１内部状態）

SUSHIInput.monomer_SCF_character_table[].**states**[]

```
State:{
    name:string          //  名前。
        //  例えば、A、B 2成分からなる傾斜濃度のポリマーの場合は、
        //  "A" や "B" が名前となり、これらは Monomer で定義されなくてはならない。
    probability:double  //  部分鎖内での存在確率。
}
```

内部状態間遷移確率

SUSHIInput.monomer_SCF_character_table[].**array_of_transition_state_probabilities**[]

```
TransitionStateProbabilities:{
    probabilities[]:double //  正方向（ junction ID の小から大）への遷移確率の配列。
        //  遷移確率とは部分鎖上の各内部状態の存在確率である。
        //  部分鎖上の位置は配列要素の番号に対応し、
        //  配列要素の数 = 部分鎖の長さ / delta_s + 1 でなければならない。
        //  最初の配列要素は経路積分開始点における各内部状態の初期値である。
}
```

5.7.3 ポリマーと溶媒の定義

定義された分子が計算に利用されるのは以降で説明する体積分率が定義された場合である。第 2.7.1 節参照。

ポリマー

SUSHIInput.components.polymers[]

ポリマーは部分鎖の結合により定義される。部分鎖はモノマーの種類と長さで定義される。部分鎖の結合情報は部分鎖両端の結合点の ID を指定するか、簡単なキーワードで指定することができる。

```
Polymer:{
    type:string //  部分鎖の結合方法を選択するキーワード。
        //  HOMO    部分鎖１本からなるホモポリマーとなる。
        //  BLOCK   部分鎖の配列の順に結合したブロックポリマーとなる。
        //  STAR    １つの結合点に全ての部分鎖が結合した星型ポリマーとなる。
        //  COMB    櫛形ポリマーとなる。部分鎖の配列は 主鎖-側鎖-主鎖-側鎖-....
             //     と認識される。例えば A1, B1, A2, B2, A3 の順に部分鎖の配列が
             //     入力されると  A1--+--A2--+--A3
             //                         B1     B2
             //     のような櫛型になる。
        //  GENERAL 部分鎖の結合は部分鎖両端の結合点の ID の指定により定義される。
    blocks[]:Block                  //  部分鎖の配列。
    junction_pairs[]:JunctionPair //  部分鎖両端の結合点 ID のペアの配列。
        //  type が GENERAL のときのみ有効。
        //  ID は 0 から始まらなくてはならない。
        //  例えば Block の要素が 1 のときこのペアに [ 0, 0 ] の ID を入力すると
        //  両端が結合されたリング状のポリマーとなる。
}
Block:{         //  部分鎖 (subchain)。
    monomer_name:string       //  部分鎖を構成するモノマー名。
        // この名前のモノマーは Monomer で定義されなくてはならない。
    number_of_monomer:double //  部分鎖の長さ。
}
JunctionPair:{  //  部分鎖両端の結合点の ID のペア。
    first:int
    second:int
}
```

溶媒

SUSHIInput.components.solvents[]

```
Solvent:{
    name:string             // 名前。
    specific_volume:double // 有効体積。
}
```

5.7.4 体積分率

体積分率を定義された分子が計算に利用される。アンサンブルが異なると体積分率の意味も異なる。第 2.7.8 節参照。

コンポーネントの体積分率

SUSHIInput.volume_fractions wline

```
VolumeFractions:{
    polymer_volume_fractions[]:VolumeFraction  // ポリマーの体積分率の配列。
    solvent_volume_fractions[]:VolumeFraction  // 溶媒の体積分率の配列。
}
VolumeFraction:{  //  体積分率。
    id:int // コンポーネントの配列番号。番号は 0 から始まる。
    volume_fraction:double
        // ensemble が CANONICAL の場合は系内の体積分率。
        // ensemble が GRANDCANONICAL の場合はバルクの体積分率。
    ensemble:string
        // CANONICAL       カノニカル アンサンブル。
        // GRANDCANONICAL グランドカノニカル アンサンブル。
        //                 この選択がなされた場合ダイナミクス計算はできない。
    bulk_volume_fraction:double  //  バルクでの体積分率。
        // 系内の組成とバルクの組成を明白に区別して入力する場合に入力する。
        // 不必要な場合-1 を入れておく。
}
```

境界上での体積分率の拘束条件の節で説明したように擬似的なグランドカノニカルアンサンブルをする場合、系内の体積分率とバルクでの体積分率を明白に分けて入力する。この場合アンサンブルは CANONICAL でよい。

5.7.5 セグメント間相互作用パラメータ（χ パラメータ）

SUSHIInput.chi_parameters[]
6 章を参照のこと。

```
ChiParameter:{
    name_i:string     // セグメント i の名前。
    name_j:string     // セグメント j の名前。
    parameter:double  // χ ij 、自動的に χ ij = χ ji となる。
                      // 定義されていないセグメント間の χ ij は 0 となる。
}
```

5.7.6 物性

物性の入力を定義するのに利用されるポリマー、溶媒、部分鎖、結合点の ID はコンポーネントの定義での配列番号に対応する。配列番号は 0 から始まる。

セグメントの体積分率

SUSHIInput.properties.segment_volume_fraction_conditions[]
部分鎖の特定区間を定義すると、その区間にあるセグメントの体積分率を出力する。ループの無いポリマー構造に対して有効である。

```
class SegmentVolumeFractionCondition:{
    polymer_ID:int      //  ポリマーの ID。
    subchain_ID:int     //  部分鎖の ID。
    begin_length:double //  部分鎖上の開始点の座標。
                        //  結合点 ID が若い番号の部分鎖端からの距離で指定する。
                        //  部分鎖両端の結合点 ID を明白に定義するには、ポリマーを
                        //  type  GENERAL  で定義する。有効となる長さの最小単位は、
                        //  SCF_parameter の delta_s である。
                        //  もちろん部分鎖長を超えてはいけない。
    end_length:double   //  部分鎖上の終了点の座標。
                        //  区間長が delta_s で割り切れない場合、計算される区間長は
                        //  入力区間長/delta_s を四捨五入した値に delta_s を掛けたものとなる。
}
```

慣性半径 R_g

SUSHIInput.properties.radius_of_gyration_conditions[]
慣性半径を出力する部分鎖の定義。

```
class SubchainUnit:{
    polymer_ID:int      //  ポリマーの ID。
    subchain_ID:int     //  部分鎖の ID。
                        //  この ID を-1 とするとポリマー全体の慣性半径を出力する。
}
```

散乱関数

SUSHIInput.properties.scattering_function
散乱関数の出力のためのフラグ。

```
class ScatteringFunctionInput:{
    calculate:select { "ON", "OFF", "RESTART" }
                        //  ON  散乱関数を計算する。
                        //  OFF  散乱関数を計算しない。
                        //  RESTART  既に出力された構造の散乱関数のみを計算しレコードとし
                        //  て出力する。構造は出力しない。
    intensity:select { "ON", "OFF" }
                        //  ON   散乱強度を出力する。１Ｄの散乱関数（powder pattern）も
                        //       出力する。
                        //  OFF  散乱強度を出力しない。１Ｄの散乱関数のみを出力する。
    coef_for_mesh:double // １Ｄの散乱関数用の出力メッシュ幅調整用の係数
                         // メッシュ幅=この値 × 最小メッシュ幅（２π/最長軸長）
}
```

5.7.7 外的条件

外的条件を定義するのに利用されるポリマー、部分鎖、結合点の ID はコンポーネントで定義した構造の配列番号に対応する。配列番号は 0 から始まる。

壁面とセグメント間の相互作用パラメータ（χ_s）

SUSHIInput.external_conditions.surface_chi_parameters[]
第 2.7.7 節参照。

```
SurfaceChiParameter:{
    boundary_name:string 相互作用を働かせる面の指定。
      //  XMin X 軸の最小値における YZ 平面。
      //  XMax X 軸の最大値における YZ 平面。
      //  YMin Y 軸の最小値における ZX 平面。
      //  YMax Y 軸の最大値における ZX 平面。
      //  ZMin Z 軸の最小値における XY 平面。
      //  ZMax Z 軸の最大値における XY 平面。
      //  RMin R 軸の最小値における境界面。
      //  RMax R 軸の最大値における境界面。
      //  Partcle 粒子オブスタクルの表面。
      //  Fiber ファイバーオブスタクルの表面。
    target_name:string // 対象となるセグメントの名前。
    parameter:double // パラメータの値。
    id:int // 各オブスタクル種の ID（オブスタクルを選択した場合有効）。
}
```

グラフト条件

SUSHIInput.external_conditions.graft_conditions[]
自由端の初期状態を設定することによりグラフト鎖を定義する。第 2.7.6 節参照。

```
GraftCondition:{
    boundary_name:string // グラフトさせる面の指定。
       // SurfaceChiParameter.boundary_name と同じ。
    polymer_ID:int       // グラフトするポリマーの ID。
    junction_ID:int      // グラフトする自由端点の ID。
    id:int              // 各オブスタクル種の ID（オブスタクルを選択した場合有効)。
}
```

マスク条件

SUSHIInput.external_conditions.mask_conditions[]
自由端の存在可能領域を設定する。ミセル等の計算に利用する。第 2.7.14 節参照。

```
MaskCondition:{
    polymer_ID:int            // グラフトするポリマーの ID。
    junction_ID:int           // マスクする自由端点の ID。
    mask_regions[]:AxisRegion // マスクする軸の領域の配列。
}
AxisRegion:{ // 軸上の領域の設定。
    axis_name:string // 軸の名前: X,Y,Z,R,H
    r_min:double // 最小値。
    r_max:double // 最大値。
        // 最大値 = 最小値 の場合、その点上でのマスクとなる。
}
```

5.7.8 静的平衡計算で有効な外的条件

多分散静的平衡計算用の最大鎖長の指定

SUSHIInput.external_conditions.static_conditions.polydispersity_conditions[]
静的平衡計算かつ単一内部状態の HOMO ポリマー多分散系にのみ有効である。第 2.7.12 節参照。

```
PolydispersityCondition:{
    longestHomoPolymerId:int    // 最長 HOMO ポリマーの ID。
    targetHomoPolymerId:int     // 経路積分を共有させるポリマーの ID。
}
```

静的平衡計算用の対称経路積分の指定

SUSHIInput.external_conditions.static_conditions.symmetry_conditions[]

第 2.7.12 節参照。静的平衡計算時に対称性のある経路積分の指定を行う。この指定により経路積分の計算時間を短縮できる。例えば、$A_{10}B_{10}, A_{20}B_{20}$ のような 2 本の鎖のある計算の場合、鎖長の長い $A_{20}B_{20}$ の鎖の自由端から接合点への経路積分は経路積分の初期値 (2.14) 式は鎖長の短い $A_{10}B_{10}$ の同じセグメント種の経路積分の初期値と同じで共用できる。一方、接合点から自由端への経路積分は経路積分の初期値が異なるので共用できない。

```
SymmetryCondition:{
    parent_path:Path // 経路積分が共用される部分鎖の経路。
    child_path:Path  // 経路積分を共用する部分鎖の経路。
}
Path:{ // 経路積分の経路。
    polymer_ID:int         // ポリマーの ID。
    subchain_ID:int        // 部分鎖の ID。
    begin_junction_ID:int // 経路積分を開始する接合点の ID。
    end_junction_ID:int    // 経路積分を終了する接合点の ID。
}
```

ドメイン領域の指定

SUSHIInput.external_conditions.static_conditions.domain_specification_conditions[]

静的平衡計算において自己無撞着場の初期値設定により目的とするドメインを簡単にシミュレーションするのに利用する。第 2.7.13 節参照。

```
DomainSpecificationCondition:{
    name:string                  // 設定をするセグメントの名前。
    domain_regions[]:AxisRegion // 軸上の領域の設定。マスク条件に同じ。
}
```

SCF 拘束条件

SUSHIInput.external_conditions.static_conditions.constraint_conditions[]

SCF 法で計算する場合に、拘束をかけ変化させない条件を指定する。

```
class ConstraintCondition:{
    target_ID:SCFUnitIDSet     //  拘束をかける部分の ID のセット。
    constraint_coondition:int  //  拘束部分：
        //  1  phi  体積分率。現実装はこれのみ。
}
class SCFUnitIDSet:{  //  拘束部分の ID のセット。
```

```
    type:int       //  反応物のタイプ：
        //  0   ポリマー
        //  1   溶媒
    IDSet[]:int    //  反応部分の ID のセット：
        //  コンポーネントの ID、部分鎖の ID、ステイトの ID
        //        溶媒の場合部分鎖の ID 以降は必要ない。
        //        1 内部状態のモノマーではステイトの ID は必要ない。
}
```

システムサイズ最適化

SUSHIInput.external_conditions.static_conditions.system_size_optimization
自由エネルギー密度を最低とするようにシステムサイズを最適化する。SCF 計算でのみ有効。

```
class OptimizingAxes:{
    x:int // 1:optimize 0:not optimize
    y:int // 1:optimize 0:not optimize
    z:int // 1:optimize 0:not optimize
    // もし、x,y 軸を比を固定したまま最適化したいなら以下のように入力する。
    //   1 1 0.
    // もし、x,y 軸をそれぞれ独立して最適化したいなら以下のように入力する。
    //   1 0 0
    //    0 1 0.
}
class SystemSizeOptimaization:{
    max_iteration_of_optimization:int // 最適化のための最大の SCF ステップ数。
    cubic_system:int                  // この値が 1 の場合、系を立法体として最適化する。
    optimizing_axes[]:OptimizingAxes  //  最適化用フラグの配列
    delta_dL:double                      //  自由エネルギー微分のための dL（相対値）
    allowed_error:double                 //  L の収束判定用誤差。
}
```

max_iteration_of_optimization のステップ内で SCF が収束もしくはこのステップ数を超えた段階でシステムサイズを更新し、次の SCF へ進む。各辺の更新量を辺の長さで割った値の絶対値が $1e^{-4}$ 以下になれば計算を終了する。

5.7.9 動力学で有効な外的条件

セグメント種毎の易動度

SUSHIInput.external_conditions.dynamic_conditions.segment_mobilities[]
モノマーや溶媒の種類に応じて異なる定数の易動度を定義できる。定義されていないものの易動度は 1 である。L_0 に当たる。第 2.7.10 節参照。

```
class SegmentMobility:{
    segment_name:string  //  易動度を定義するセグメントの名前。これと同名のもの
                         //  が Monomer か Solvent で定義されなくてはならない。
    mobility:double      //  易動度。
}
```

分子の存在条件による易動度

SUSHIInput.external_conditions.dynamic_conditions.polymer_mobilities[]
SUSHIInput.external_conditions.dynamic_conditions.solvent_mobilities[]

ポリマーの濃度と存在するマトリックス種による移動度。体積分率 ϕ が小さいポリマーに対して意味がある定義である。この定義をすると易動度は $L_0\phi$ として扱われる。L_0 は上記の値に対して鎖の寄与が考慮されたものとなる。A を対象とするポリマー、B をマトリクスの成分を意味するとする。第 2.7.10 節参照。

ROUSE 条件：マトリクスの成分よりも十分鎖長が長く、低濃度の場合。$L = L_0\phi$ として計算される。

ROUSE 条件：マトリクスの成分と同等の鎖長で絡み合うほどの鎖長があり、低濃度の場合。$L = (L_0/N)\phi$ として計算される。N は鎖の全長である。溶媒に対しては有効にならない。

```
class LocalMobility:{
    component_ID:int     //  易動度を定義する分子の ID
    type:string          //  易動度のタイプ
            // ROUSE
            // REPTATION
    mobility:double      //  易動度。
}
```

反応速度が速い反応条件

SUSHIInput.external_conditions.dynamic_conditions.reaction_conditions_of_rapid_reactions[]

反応速度が速いという意味は、反応速度が動力学計算で用いている時間ステップより速く、ある体積分率の成分が瞬時に同じ体積分率の別の成分になるということである。例えば、モノマーがラジカル反応である重合度のポリマーに変化するような条件がこれに該当する。入力できる化学反応は A+B+C+.... が D になるような反応で、反応物の数が反応次数となる。反応が複数（生成物が複数）の場合はこの条件を反応の数だけ入力する。第 2.7.15 節参照。

```
class ReactionConditionOfRapidReaction:{
    reactant_IDs[]:ReactantIDSet     //  反応物の ID。
                                     //  以降で説明する、反応の定義に共通するデータの定義参照。
    product_ID:int                   //  生成物の ID。生成物は 1 種類に限る。
    volume_fractions_of_reactants_in_product[]:double
                                     //  生成物内の反応物の体積分率の配列。
    reaction_constant:double         //  反応定数。
}
```

分子構造が変化する反応条件

SUSHIInput.external_conditions.dynamic_conditions.reaction_conditions_of_active_sites[]
ポリマーの結合点、ポリマーの自由端またはモノマーが反応して結合して異なる構造のポリマーになる反応を定義できる。生成物はポリマーでなくてはならない。反応物は 2 つに限られる。よって反応次数は 2 次となる。反応が複数（生成物が複数）の場合はこの条件を反応の数だけ入力する。第 2.7.15 節参照。

```
class ReactionConditionOfActiveSites:{
    reactant_IDs[]:ReactantIDSet    //  反応物の ID セットの配列。反応物は 2 種類に限る。
                                    //  以降で説明する、反応の定義に共通するデータの定義参照。
    polymer_ID_of_product:int       //  生成ポリマーの ID。反応物は 1 種類に限る。
    map_of_subchains[]:SubchainMap  //  部分鎖の対応関係の配列。
                                    //  以降で説明する、反応の定義に共通するデータの定義参照。
    reaction_constant:double        //  反応定数。
}
```

グラフト反応条件

SUSHIInput.external_conditions.dynamic_conditions.reaction_conditions_of_grafts[]
反応が複数（生成物が複数）の場合はこの条件を反応の数だけ入力する。第 2.7.15 節参照。

```
class ReactionConditionOfGraft:{
    graft_condition:GraftCondition  //  反応ポリマーのグラフト条件。
                                    //  前に示したグラフト条件の定義に同じ。
    polymer_ID_of_product:int       //  生成ポリマーの ID。反応物は 1 種類に限る。
    map_of_subchains[]:SubchainMap  //  部分鎖の対応関係の配列。
                                    //  以降で説明する、反応の定義に共通するデータの定義参照。
    reaction_constant:double        //  反応定数。
}
```

反応の定義に共通するデータの定義

SUSHIInput.external_conditions.dynamic_conditions 内の
reaction_conditions_of_rapid_reactions[].reactant_IDs[]
reaction_conditions_of_active_sites[].reactant_IDs[]
reaction_conditions_of_active_sites[].map_of_subchains[]
reaction_conditions_of_grafts[].map_of_subchains[]

```
class ReactantIDSet:{  //  反応成分の ID のセット。
    type:int    // 反応物のタイプ：
        //  0  ポリマー。
        //  1  溶媒。
```

```
    IDSet[]:int // ID のセット
        //  反応速度が速い反応条件の場合        反応物の ID。
        //  分子構造が変化する反応条件の場合   反応物の ID、Junction の ID。
        //                                      溶媒の場合   Junction の ID は不要。
}
class SubchainMap:{  //  生成物と反応物の間の部分鎖の対応関係。
    reactant_ID:int                 //  反応物の ID。
        //  注意   ここでの ID は reactant_IDs[] の配列番号である。component の配列番号ではない。
        //         分子構造が変化する反応条件の場合は 2 次反応なので 0 もしくは 1 である。
        //         以下の部分鎖の ID は、反応物と生成物の部分鎖の ID の対応関係を入力する。
    subchain_ID_of_reactant:int  //  反応物の部分鎖の ID。
    subchain_ID_of_product:int   //  生成物の部分鎖の ID。
}
```

ずり流動

SUSHIInput.external_conditions.dynamic_conditions.shear
ずり速度と周期を入力する。第 2.6.1 節参照のこと。周期が 0 の場合は単純なずりとなる。

```
class Shear:{
    shear_rate:double
    period:double
}
```

熱ゆらぎ

SUSHIInput.external_conditions.dynamic_conditions.noise
セグメント濃度に対する熱ゆらぎの標準偏差を入力する。熱ゆらぎが入力される方程式はセグメント濃度の動力学方程式である。熱揺らぎは揺動散逸定理を満たすように発生する。

```
class Noise:{
    random_seed:int // ノイズを発生するための乱数の種。
                    // 0 を入力すると時刻を参照してランダムに変化する。
                      // 0 以上の値を入力するとその値を乱数の種として固定してしまう。
                      // よって、0 以上の数値はデバッグのためだけに利用する。
    standard_deviation_of_thermal_noise:double
        // 発生するセグメント濃度に対する熱ゆらぎの標準偏差
}
```

システムサイズ最適化

SUSHIInput.external_conditions.dynamic_conditions.system_size_dynamics
動力学と同時にシステムサイズを最適化する。第 2.7.18 節参照のこと。

```
class SystemSizeDynamics:{
    parameter_of_system_size_dynamics:double  //  (2.103) 式の Q である。
    compressibility:double                    //  (2.103) 式の圧縮率である。
    interval_of_system_size_dynamics:int      //  (2.103) 式を実行するステップ間隔
    optimizing_axes[]:OptimizingAxes  //  最適化用フラグの配列
                                  //  class SystemSizeOptimaization  を参照のこと。
    delta_dL:double                       //  自由エネルギー微分のための dL（相対値）
    allowed_error:double                  //  L の収束判定用誤差。
}
```

流体力学的効果

SUSHIInput.external_conditions.dynamic_conditions.hydrodynamics_parameters
流体力学的効果を導入する。第 2.7.5 節参照のこと。

```
class Viscosity:{ // 粘度のデータ
    name:string                                          // セグメント名
    viscosity:double                                     // 粘度
}
class HydrodynamicsParameters:{ // 流体力学的パラメータ
    density:double                                       // 密度
    viscosities[]:Viscosity                              // 粘度の配列
    neglect_convection_term:int                          // 移流項  (2.66) 式の第 1 項
                                                         // を無視するなら 1 を入れる。
    dynamics_scheme:select { "EXPLICIT", "EXPLICIT2", "IMPLICIT" }
            // 計算で利用されるスキーム。
            //  EXPLICIT  陽的解法。通常はこちらでよい。Euler 法
            //  EXPLICIT2 陽的解法の 2 次。2 段階 Runge-Kutta 法
            //  IMPLICIT  陰的解法。
    poisson_solver_parameter:PoissonSolverParameter
            // ポアソン方程式を解くためのパラメータ  以降の高分子電解質の節を参照のこと。
    noise:Noise
            // 熱ゆらぎを Navier-Stokes の方程式に追加するためのパラメータである。
            // データの構造は 熱ゆらぎ の節で述べたものと同じである。
}
```

ハイブリッド法

SUSHIInput.external_conditions.dynamic_conditions.hybrid
ハイブリッド法を起動するためのパラメータ。第 2.7.4 節参照のこと。

```
class Hybrid:{
    SCF_step:int //  SCF 法の回数  通常は 1
    RPA_step:int //  GRPA 法の回数  任意
    free_energy_by:select { "RPA", "SCF" }
        //  自由エネルギーの計算方法
        //  RPA   RPA 法
        //  SCF   SCF 法
        // 注意
        // SUSHIInput.solver\_parameter.RPA
        // の節を参照のこと。 l
}
```

5.7.10 SCF モンテカルロ法

SUSHIInput.external_conditions.monte_carlo_conditions

```
class MonteCarloConditionOfJunction: {
    // モンテカルロ法で更新する Junciton の情報
    junction_ID:int    // ID of the junction.
    position:Vector3D  // Position of the junction.
}
class MonteCarloConditionOfPolymer:{
    polymer_ID:int    // ID of the polymer.
    monte_carlo_conditions_of_junction[]:MonteCarloConditionOfJunction
}
class MonteCarloConditions:{
    monte_carlo_conditions_of_polymer[]:MonteCarloConditionOfPolymer
}
```

5.7.11 静電場

SUSHIInput.external_conditions.electrostatic_conditions

もし、
SUSHIInput.external_conditions.electrostatic_conditions.electrostatic_condition.dielectric_constant_of_system
へ正の値が入力されると静電場の計算が有効となる。

```
class ElectrostaticConditions: {
    // 高分子電解質のための入力パラメータ。
    electrostatic_condition:ElectrostaticCondition
    poisson_solver_parameter:PoissonSolverParameter
}
```

```
class ElectrostaticCondition: {
    dielectric_constant_of_system:double  // 系の誘電率
    electrostatic_parameters_of_segment[]:ElectrostaticParametersOfSegment
    electrolyte[]:Electrolyte // 弱電解質のパラメータ
    external_electric_field:Vector3D // 2.7.6 節参照
      // 外部電場ベクトル
    parameter_of_external_electric_field_for_dynamics
   :ParameterOfExternalElectricFieldForDynamics
}
class ElectrostaticParametersOfSegment: {
    name:string    // セグメント名
    charge:double  // 単位体積あたりの電荷
    dielectric_constant:double  //  セグメントの比誘電率
}
class Electrolyte: {
    name:string
    name_of_dissociated_ion:string
    name_of_free_ion:string
    kp:double
}
class ParameterOfExternalElectricFieldForDynamics:{
    direction:select { "X", "Y", "Z" }
    alpha:double // 2.7.6 節参照、
      //  注意    同時に external_electric_field:Vector3D を利用できない。
}
class PoissonSolverParameter: {
    // ポアソン方程式を解くためのパラメータ   ICCG 法を利用
    omega:double          // 利用していない。
    allowed_error:double // 収束判定誤差
    max_iteration:int     // ICCG 法を解く場合の最大繰り返し回数
}
```

ポアソン方程式は次のように定義される。

$$\nabla\epsilon(\mathbf{r})\nabla U(\mathbf{r}) = -\rho(\mathbf{r}) \tag{5.1}$$

ここで $U(\mathbf{r})$ は静電ポテンシャル、$\rho(\mathbf{r})$ は系内の全電荷分布、$\epsilon(\mathbf{r})$ は誘電率分布である。この方程式を解くのに ICCG 法が用いられる。

5.7.12 オブスタクル

SUSHIInput.external_conditions.obstacles

外的条件としてオブスタクル（ポリマーを排除する障害物）を投入できる。オブスタクルとして投入できるのは球とファイバーである。オブスタクルの領域はオブスタクルの内側もしくは外側とすることができる。

```
class Particle: {
```

```
    // 球を入力するための入力パラメータ。
    position_of_center:Vector3D // 球の中心座標
    radius_of_particle:double   // 球の半径
    region_of_obstacle:select { "IN", "OUT" } // オブスタクルの存在領域：球殻の内側、外側
    // ソフト粒子として追加入力項目
    coef_for_surface_depth:double // 粒子界面厚さを設定するためのパラメータ
    name_as_solvent:string // 対応する溶媒名
    effective_diffusion_constant:double // 移動させるための任意の拡散係数
}
class Fiber: {
    // ファイバーを入力するための入力パラメータ。
position_of_end0:Vector3D // ファイバーの片末端の座標
position_of_end1:Vector3D // ファイバーの片末端の座標
radius_of_fiber:double    // ファイバーとファイバー末端の球の半径
end_cap:select { "YES", "NO" } // 末端を球でキャップする場合 YES 、デフォルトは YES
region_of_obstacle:select { "IN", "OUT" } // オブスタクルの存在領域：ファイバーの内側、外側
}
class Hexahedron: {
    // 6 面体は対向する頂点 2 つとその頂点からの 3 つのベクトルで定義する。
    position_of_origin0:Vector3D // 頂点 1
    direction_a0:Vector3D // 頂点 1 からのベクトル 1
    direction_b0:Vector3D // 頂点 1 からのベクトル 2
    direction_c0:Vector3D // 頂点 1 からのベクトル 3
    position_of_origin1:Vector3D 頂点 2
    direction_a1:Vector3D // 頂点 1 からのベクトル 1
    direction_b1:Vector3D // 頂点 1 からのベクトル 2
    direction_c1:Vector3D // 頂点 1 からのベクトル 3
    coefs_of_surface_chi[]:double // \chi_s  に掛ける係数
    region_of_obstacle:select { "IN", "OUT" } // オブスタクルの存在領域：6 面体の内側、外側
}
class ShapeByGrids: { // VER. 140808
    input_filepath:string // ポリゴンデータファイル名
    region_of_obstacle:select { "IN", "OUT" } // オブスタクルの存在領域：形状の内側、外側
}
class Obstacles:{
    // オブスタクルのデータセット
    ndiv_for_volume_calculation:int // オブスタクルの境界を設定するための 1 メッシュ当りの分割数

                                    // 100 ぐらいがよい。
    particles[]:Particle // 球の配列
    fibers[]:Fiber       // ファイバーの配列
    hexahedrons[]:Hexahedron // 六面体の配列
    shape_by_grids[]:ShapeByGrids // 三角形ポリゴンデータの配列
}
```

5.7.13 ズーミング

SUSHIInput.zoom

SUSHI の計算結果を他のエンジンにズーミングする。現在のところ COGNAC における Kremer-Grest モデルへのズーミングのみサポートする。

```
class COGNACZoomingParameter: {
    b_per_sigma:double // 1 セグメントが Kremer-Grest モデルにおける何ビーズ分に相当するかの値
                       // 理論値は  1.6878  ですが、d_s の値によって、
                      // b/sigma = b/(n*d_s) を満たすように変更しなくてはならない。
                      // ここで、n は任意の整数。
    density:double     // Kremer-Grest モデルにおける密度、デフォルト値は 0.85
    by:select { "RUN", "RESTART" }
       // RUN:通常の実行でセグメント濃度場を自動的に計算する。
       // RESTART:リスタートでセグメント濃度場のみを自動的に計算して停止する。
}
class Zoom:{ // COMMON
    type:select { "COGNAC" } // COGNAC へのズーミングを行いたいなら、COGNAC を選択する。
    COGNAC:COGNACZoomingParameter
}
```

この計算の場合、relax_in.udf もしくは restart_in.udf ファイルを COGANC の雛形ファイルとして利用する。第 5.3 節を参照してください。また、文献 [34] を参考にしてください。

5.8 共通 UDF 定義

SUSHI では出力の冒頭に上節で述べた SUSHIInput と以下の共通データが出力される。共通データの主な内容は SUSHIInput の処理結果である。

5.8.1 メッシュの座標

```
MeshData: {  //  各メッシュ点の位置ベクトルである。
    position[]:Vector3D
}
class Vector3D: { // 3 次元位置ベクトル
    x:double
    y:double
    z:double
}
```

5.8.2 系内の組成分析結果

入力時の組成の分析結果である。

```
Composition: {  //  組成分析結果
    species[]:VolumeFractionData  // 各成分毎の体積分率
        //  ここでの成分とは taperedAB のような内部状態を持つものを１単位として扱う。
    states[]:VolumeFractionData   // ステイト毎の体積分率
        //  ステイトは taperedAB のようなものの内部状態をさらに分離し、A、B を１単位として扱う。
    SCF_units[]:SCFUnitData       // SCFUnit 毎の体積分率
    junctions[]:JunctionData      // 結合点の情報
}
class VolumeFractionData: {
    name:string             //  名前
    volume_fraction:double  //  体積分率
}
class SCFUnitData: {
    // SCF ユニットとは体積分率を分類する最小の単位である。
    // セグメントの体積分率φ i はこの単位に分割されて出力される。
    // φ i に共役している自己無撞着場 Vi も同様にこの単位の数分出力される。
    // ポリマーの場合、分類に利用されるインデックスはポリマー ID、部分鎖 ID、
    // 内部状態 ID（例えば taperedAB ポリマーの場合 A,B の２種類。）の３つである。
    // 溶媒の場合は溶媒の ID のみ１種類である。
    name:string             //  名前。最小分離単位はステイトである。
    volume_fraction:double  //  体積分率。
    ID_set[]:int            //  ID のセット。
        //  SCFUnit の ID のセットは、以下のように状態により異なる。
        //  内部状態を持つポリマーの場合
        //    ポリマーの ID、部分鎖の ID、内部状態の ID の３つの ID。
        //  内部状態を持たないポリマーの場合
        //    ポリマーの ID、部分鎖の ID の２つの ID。
        //  溶媒の場合
        //    溶媒の ID。１つの ID。
}
class JunctionData: {  //  結合点の情報
    ID_set[]:int  //ポリマーと結合点の ID のセット。
        //  ポリマーの ID、結合点の ID
}
```

5.9 出力 UDF 定義

出力は各レコード毎に区切り出力される。開始と終了は次のようになる。

```
\begin{record}{"Step number"}
\begin{data}
    ～データが出力される。
\end{data}
\end{recoed}
```

上記の number にレコード番号が出力される。GOURMET でのレコード番号とはこの番号である。以降で説明するダイナミクスのステップ数と混同してはいけない。レコードには、最初に以下の 3 つのデータがある。

```
Steps:int   // ダイナミクスのステップ。
Time:double // ダイナミクスの時間。
Nscf:int    // 収束するの要した SCF 回数。
```

全ての計算において、最初にレコード番号 0、ダイナミクスのステップ 0、ダイナミクスの時間 0.、収束するの要した SCF 回数 0 として初期入力状態が出力される。静的平衡計算の場合、この次にレコード番号 1、ダイナミクスのステップ 0、ダイナミクスの時間 0. となった計算終了時のレコードが出力される。ダイナミクス計算の場合、初期入力状態レコードの後に、各値がダイナミクス計算の結果とおりのレコードが追加されてゆく。これらの出力の後に以下のようなデータ本体が出力される。詳細は第 5.10 節の出力 UDF 詳細定義を参照のこと。

```
SUSHIOutput: {
  // 計算終了時体積分率
    volume_fractions:VolumeFractions // 共通データ出力 UDF 定義に同じ。
  // φ および V ##########################################
    flag_of_dynamics:int          // 動的平均場法フラグ
        // 0 静的平衡計算
        // 1 動的な時間発展の計算
    phi:ScalarField               // φ スカラー場の配列
    V  :ScalarField               //  V スカラー場の配列
  // セグメント 体積分率
    segment_volume_fraction:ScalarField
  // 自由エネルギー ####################################
    free_energy:double            //  自由エネルギー
                                  //  第 2.7.11 節参照。
    excess_free_energy:double     //  過剰自由エネルギー 壁があるグランドカノニカル
                                  //  アンサンブルの系で有効。
  // 追加された出力情報 ################################
    optional_output:OptionalOutput
  // 計算の終了状況 ####################################
    flagConvergence:int           //  SCF 計算の収束の状態
        // 0 : 初期状態。
        // 1 : 計算成功。
        // 2 : ＳＣＦの最大回数を超える。
        // 3 : ポアソン方程式を解けなかった。
```

```
        // 4 : kill ファイルにより停止する。
        // 5 : "STOP" メッセージにより停止する。
        // 6 : "END"  メッセージにより停止する。
        // 7 : 不明なエラーにより停止する。
}
```

5.10 出力 UDF 詳細定義

5.10.1 スカラー場データ

SUSHIOutput.phi
SUSHIOutput.V
SUSHIOutput.segment_volume_fraction
スカラー場として V, ϕ と segment density conditions の入力があった場合以下の構造のセグメントの体積分率が出力される。

```
class ScalarArray:{ //
    comp[]:float     // メッシュ上の成分のデータの配列。
}
class ScalarField:{ //
    name:KEY              // 名前、KEY 定義で Python 上の検索に利用できる。
                          // 詳細は UDF マニュアル参照。
    num_of_component:int  // 成分の数
    value[]:ScalarArray   // 値
};
```

5.10.2 追加出力データ

SUSHIOutput.optional_output
このデータ構造には出力が要請されたデータが追加される。

```
class OptionalOutput: {
    segment_volume_fraction:ScalarField
        // Volume fraction of segment species specified in
        // the SUSHIInput.properties.segment_volume_fraction_conditions[].
    radius_of_gyration[]:RadiusOfGyration
        // Radii of gyration.  Refer to the next section.
    monte_carlo_condition_of_polymer[]:MonteCarloConditionOfPolymer
        // The positions of junctions specified in
        // the SUSHIInput.external_conditions.monte_carlo_conditions.
    electric_potential:ScalarField
        // Electric potential is output when the
```

```
        // SUSHIInput.external_conditions.electric_conditions is specified.
}
```

慣性半径

SUSHIOutput.optional_output.radius_of_gyration[]
入力で慣性半径計算の指定がなされた場合に出力される。

```
RadiusOfGyration: {
    dim:int           // 計算された空間の次元。メッシュの次元に等しい。
    rg:double         // 慣性半径
    rg_xyz[]:double   // 慣性半径の x,y,z 要素の値。空間次元の要素数しか出力されない。
}
```

5.11 ログファイル

SUSHI では 2 つのログが出力される。1 つは標準出力に出力される計算状況であり、静的平衡計算とダイナミクス計算では内容が異なる。-C オプションで出力先をファイルに変更できる。もう 1 つは、ダイナミクスの各ステップ毎に上書きされる SCF 計算の収束状況である。デフォルトでは".usl"の拡張子がついたファイルとなる。以下に各ログの内容を説明する。詳しくは第 5.3 節参照のこと。

5.11.1 標準出力ログ

静的平衡計算とダイナミクス計算に共通して最初にエンジンのバージョンとファイル名が次の例のように出力される。

```
SCFEngine Version 4.0 Revision 031228L, Octa Project
file names
input ab_ring.sin
output ab_ring_uot.udf
restart ab_ring_uar.udf
archives ab_ring_uar.udf
log ab_ring.usl
cout ab_ring.ual
stop ab_ring.stp
```

ここでのファイル名の意味は次のようなものである。

---

input 入力ファイル
output 出力ファイル
restart リスタートファイル
archives アーカイブスファイル
log ダイナミクス計算 SCF ログファイル
cout 標準出力ファイル
stop 緊急停止用ファイル

これらは、SUSHI の認識状態を出力したものであって、全てのファイルが利用されることを意味していない。リスタートが不要ならリスタートファイルは利用されない。静的平衡計算であればダイナミクス計算 SCF ログファイルは出力されない。

静的平衡計算標準出力ログ

以下のような出力となる。

```
.....
nSCF 1005 nW 0 err  3.25043835e-05  nV 1 err  1.01382781e-04 nTotPhi 0 err  8.44784069e-05
nSCF 1006 nW 0 err  3.89468651e-05  nV 0 err  9.99588160e-05 nTotPhi 0 err  7.55554209e-05
nSCF    1006 FreeEnergy 1.49953121e-01   ExcessFE -2.50396581e-01
//!!!!!!!! SCF Convergence succeeded. : nSCF = 1006 ////////
cpu time 10 [sec]
```

各 SCF 計算で出力されるのは、

nSCF [SCF 回数]
nW [未収束な W の数] err [未収束な W の誤差の最大値]
nV [未収束な V のメッシュ点数] err [未収束な V の誤差の最大値]
nPhi [非圧縮条件を満たさないメッシュ点数] err [未収束な体積分率の和の誤差の最大値]

である。誤差の最大値が収束判定条件で入力した error の値より小さくなれば収束完了である。詳しくは第**??**節参照のこと。計算終了時に SCF 回数と自由エネルギーと過剰自由エネルギーを出力する。正常終了の場合

```
//!!!!!!!! SCF Convergence succeeded. : nSCF = 1006 ////////
```

のように、異常終了の場合

```
//XXXXXXXX Exceeded maximum step number of SCF. : nSCF = 20000 ////////
```

のように出力する。最後は計算に要した実際の経過時間である。

ダイナミクス計算標準出力ログ

以下のような出力となる。

```
.....
nStep 499999 time 499.999 nSCF 2 F -2.14330212e-01  ExcessFE -1.60555993e+01
nStep 500000 time 500     nSCF 3 F -2.14330226e-01  ExcessFE -1.60556145e+01
//!!!!!!!! SCF Convergence succeeded. : nSCF = 3 ////////
cpu time 20614 [sec]
```

各ダイナミクスのステップで出力されるのは、

```
nStep [ダイナミクスのステップ数] time  [時間]
nSCF  [SCF 回数]
F     [自由エネルギー]  ExcessFE [過剰自由エネルギー]
```

である。途中で SCF 計算が収束しなくなった場合

```
//XXXXXXXX Exceeded maximum step number of SCF. : nSCF = 20000 ////////
```

のような出力で停止する。最後には実際の経過時間が出力される。

ダイナミクス計算 SCF ログファイル

内容は

```
nSCF 1 not equilibrated Polymers 2
    polymerId   0  nW       0  err 0.00000000e+00    nPhi       4  err 1.42054109e-04
    polymerId   1  nW       0  err 0.00000000e+00    nPhi       7  err 1.38242704e-04
.....
```

のようになり、各 SCF ステップ毎の未収束なポリマーの数の後に

```
polymerId [ポリマーの ID]
nW   [未収束な W の数] err [未収束な W の誤差の最大値]
nPhi [未収束なφのメッシュ点数]  err [未収束なφの誤差の最大値]
```

が繰り返し出力される。誤差の最大値が収束判定条件で入力した error の値より小さくなれば収束完了である。詳しくは第**??**節参照のこと。

5.12 パラメータ UDF

SUSHI の起動時に “-Pparameter_udf_file” オプションが用いられたとき、ファイル “parameter_udf_files” にレコードされたパラメータが読み込まれる。このファイルの定義を次に示す。

```
\begin{header}
\begin{def}
EngineType:string;
EngineVersion:string;
IOType:string;
ProjectName:string;
Comment:string;
\end{def}
\begin{data}
EngineType:"SUSHI"
EngineVersion:"011221"
IOType:"IN"
ProjectName:"WG2"
Comment:"Control Parameter"
\end{data}
\end{header}
\begin{def}
Time:double
class ChiParameter:{
    name_i:string
    name_j:string
    parameter:double
}
///// START SUSHIParameter.udf ////////////////////////////////////////////////
SCFControlParameter:{
    constV                          :double
    constW                          :double
    error                           :double
    judge_method                    :select { "ABSOLUTE", "RELATIVE" }
    scf_output_interval_step        :int
    max_SCF_step                    :int
}
DynamicsControlParameter:{
    delta_t                         :double
    max_dynamics_step               :int
    output_interval_step            :int
    archives_interval_step          :int
    log_interval_step               :int
}
VariableControlParameter:{
    chi_parameters[]:ChiParameter
}
///// END SUSHIParameter.udf //////////////////////////////////////////////////
\end{def}
```

パラメータの内容は SUSHIInput.udf ファイル内のものと同じである。SUSHI をコントロールするには、“parameter_udf_file”ファイルを用意する。このファイル内のレコードの時間がシミュレーションの時間と一致すると、利用しているパラメータが読み込まれたパラメータで上書きされる。時間の意味は計算方法で異なる。静的な平衡計算の場合、有効な時間は 0. のみである。動力学計算の場合、動力学のシミュレーション時間が利用される。例えば動力学計算で、時間に依存するような χ パラメータをパラメータファイルに設定しておけばクエンチを模擬したシミュレーションが可能である。

Alphabetical UDF classes	
UDF parameter	Description
ADFParameter	class ADFParameter
alpha	double
allowedError	double
maxIteration	int
alphaForPoissonSolver	double
allowedErrorForPoissonSolver	double
maxIterationForPoissonSolver	int
bondFactor	double
bulkModulus	double
phiMin	double
phiMax	double
isIncompressible	int (no effect)
AxisBoundaryCondition	class AxisBoundaryCondition
axis_conditions[]	array of string PERIODIC/DIRICHLET/WALL/NEUMANN
AxisRegion	class AxisRegion
axis_name	[select] X/Y/Z/R/H
Block	class Block
monomer_name	string
number_of_monomer	double
BoundaryCondition	class BoundaryCondition
conditions[]	array of AxisBoundaryCondition class
volume_fractions_on_boundaries	VolumeFractionsOnBoundaries class
CalculationMethod	class CalculationMethod
type	[select] STATICS/DYNAMICS/MONTECARLO
ChiParameter	class ChiParameter
name_i	string
name_j	string
parameter	double
Components	class Components
polymers[]	array of Polymer class
solvents[]	array of Solvent class
ConstraintCondition	class ConstraintCondition
target_ID	SCFUnitIDSet class
constraint_coondition	int 1: phi/ 2: V
DomainSpecificationCondition	class DomainSpecificationCondition
name	string
domain_regions[]	array of AxisRegion class
initial_chemical_potential	double

UDF parameter	Description
DynamicConditions	class DynamicConditions
segment_mobilities[]	array of SegmentMobility class
types_of_polymer_mobility[]	array of LocalMobility class
types_of_solvent_mobility[]	array of LocalMobility class
reaction_conditions_of_rapid_reactions[]	array of ReactionConditionOfRapidReaction class
reaction_conditions_of_active_sites[]	array of ReactionConditionOfActiveSites class
reaction_conditions_of_grafts[]	array of ReactionConditionOfGraft class
shear	class Shear
noise	class Noise
system_size_dynamics	class SystemSizeDynamics
DynamicsParameter	class DynamicsParameter
delta_t	double
max_dynamics_step	int
output_interval_step	int
archives_interval_step	int
log_interval_step	int
dynamics_scheme	[select] EXPLICIT/EXPLICIT2/IMPLICIT
compressibility	double
Electrolyte	class Electrolyte
name	string
name_of_dissociated_ion	string
name_of_free_ion	string
kp	double
ElectrostaticCondition	class ElectrostaticCondition
dielectric_constant_of_system	double
electrostatic_parameters_of_segment[]	array of ElectrostaticParametersOfSegment class
electrolyte[]	array of Electrolyte class
external_electric_field	Vector3D class
parameter_of_external_electric_field_for_dynamics	ParameterOfExternalElectricFieldForDynamics class
ElectrostaticConditions	class ElectrostaticConditions
electrostatic_condition	class ElectrostaticCondition
poisson_solver_parameter	PoissonSolverParameter class
ElectrostaticParametersOfSegment	class ElectrostaticParametersOfSegment
name	string
charge	double
dielectric_constant	double
ExternalConditions	class ExternalConditions
surface_chi_parameters[]	array of SurfaceChiParameter class
graft_conditions[]	array of GraftCondition class
mask_conditions[]	array of MaskCondition class
static_conditions	StaticConditions class
dynamic_conditions	DynamicConditions class
monte_carlo_conditions	MonteCarloConditions class
FHParameter	class FHParameter
kappas[]	array of Kappa class

UDF parameter	Description
Fiber	class Fiber
position_of_end0	Vector3D class
position_of_end1	Vector3D class
radius_of_fiber	double
end_cap	[select] YES/NO
region_of_obstacle	[select] IN/OUT
GraftCondition	class GraftCondition
polymer_ID	int
junction_ID	int
boundary_name	[select] XMin/XMax/YMin/YMax/ZMin/ZMax RMin/RMax/HMin/HMax/Particle/Fiber
obstacle_ID	int
Hexahedron	class Hexahedron
position_of_origin0	Vector3D
direction_a0	Vector3D
direction_b0	Vector3D
direction_c0	Vector3D
position_of_origin1	Vector3D
direction_a1	Vector3D
direction_b1	Vector3D
direction_c1	Vector3D
coefs_of_surface_chi[]	double
region_of_obstacle	[select] IN/OUT
HydrodynamicsParameters	HydrodynamicsParameters
density	double
viscosities[]	Viscosity
neglect_convection_term	int
dynamics_scheme	{select} EXPLICIT/EXPLICIT2/IMPLICIT
poisson_solver_parameter	PoissonSolverParameter class
Hybrid	Hybrid
SCF_step	int
RPA_step	int
free_energy_by	{select} RPA/SCF
JunctionData	class JunctionData
ID_set[]	array of int
JunctionPair	class JunctionPair
first	int
second	int
Kappa	class Kappa
polymer_ID	int
value	double
LocalMobility	class LocalMobility
component_ID	int
type	[select] ROUSE/REPTATION

UDF parameter	Description
MaskCondition	class MaskCondition
polymer_ID	int
junction_ID	int
mask_regions[]	array of AxisRegion class
Mesh	class Mesh
name	KEY
type	[select] REGULAR/RECTANGULAR CYLINDRICAL/SPHERICAL
MeshAxis	class MeshAxis
values[]	array of double
Monomer	class Monomer
species_name	string
specific_volume	double
effective_bond_length	double
MonomerSCFChar	class MonomerSCFChar
name	string
states[]	array of State class
array_of_transition_state_probabilities[]	array of TransitionStateProbabilities class
MonteCarloConditionOfJunction	class MonteCarloConditionOfJunction
junction_ID	int
position	Vector3D class
MonteCarloConditionOfPolymer	class MonteCarloConditionOfPolymer
polymer_ID	int
monte_carlo_conditions_of_junction[]	array of MonteCarloConditionOfJunction class
MonteCarloConditions	class MonteCarloConditions
monte_carlo_conditions_of_polymer[]	array of MonteCarloConditionOfPolymer class
MonteCarloParameter	class MonteCarloParameter
max_monte_carlo_step	int
output_interval_step	int
archives_interval_step	int
log_interval_step	int
Noise	class Noise
standard_deviation_of_thermal_noise	double
Obstacles	class Obstacles
ndiv_for_volume_calculation	int
particles[]	array of Particle class
fibers[]	array of Fiber class
hexahedrons[]	Hexahedron
shape_by_grids[]	ShapeByGrids
OptionalOutput	class OptionalOutput
segment_volume_fraction	ScalarField class
radius_of_gyration[]	array of RadiusOfGyration class
monte_carlo_condition_of_polymer[]	array of MonteCarloConditionOfPolymer class
electric_potential	ScalarField class

UDF parameter	Description
Particle	class Particle
position_of_center	Vector3D class
radius_of_particle	double
region_of_obstacle	[select] IN/OUT
Path	class Path
polymer_ID	int
subchain_ID	int
begin_junction_ID	int
end_junction_ID	int
coef_for_surface_depth	double
name_as_solvent	string
effective_diffusion_constant	double
ParameterOfExternalElectricFieldForDynamics	class ParameterOfExternalElectricFieldForDynamics
direction	[select] X/Y/Z
alpha	double
PoissonSolverParameter	class PoissonSolverParameter
omega	double
allowed_error	double
max_iteration	int
PolydispersityCondition	class PolydispersityCondition
longest_homo_polymer_ID	int
target_homo_polymer_ID	int
Polymer	class Polymer
type	[select] HOMO/BLOCK/COMB/STAR/GENERAL
Properties	class Properties
segment_volume_fraction_conditions[]	array of SegmentVolumeFractionCondition class
radius_of_gyration_conditions[]	array of SubchainUnit class

UDF parameter	Description
RadiusOfGyration	class RadiusOfGyration
dim	int
rg	double
rg_xyz[]	array of double
ReactantIDSet	class ReactantIDSet
type	int 0 : polymer/ 1 : solvent
IDSet[]	array of int: molecular ID, junction ID
ReactionConditionOfActiveSites	class ReactionConditionOfActiveSites
reactant_IDs[]	array of ReactantIDSet class
polymer_ID_of_product	int
map_of_subchains[]	array of SubchainMap class
reaction_constant	double
ReactionConditionOfGraft	class ReactionConditionOfGraft
graft_condition	GraftCondition class
polymer_ID_of_product	int
map_of_subchain_IDs[]	array of int
reaction_constant	double
ReactionConditionOfRapidReaction	class ReactionConditionOfRapidReaction
reactant_IDs[]	array of ReactantIDSet class
product_ID	ReactantIDSet class
volume_fractions_of_reactants_in_product[]	array of double
reaction_constant	double
RPAParameter	RPAParameter
random_seed	int
standard_deviation	double

UDF parameter	Description
ScatteringFunctionInput	ScatteringFunctionInput
calculate	[select] ON/OFF/RESTAR
intensity	[select] ON/OFF
coef_for_mesh	double
SCFParameter	class SCFParameter
delta_s	double
constV	double
constW	double
error	double
random_seed	int
standard_deviation	double
method_of_convergence_test	[select] ABSOLUTE/RELATIVE
convergence_test_interval_step	int
max_SCF_step	int
scf_output_interval_step	int
SCF_method	[select] INCORE/DIRECT
pathintegral_scheme	[select] EXPLICIT/IMPLICIT
SCFUnitData	class SCFUnitData
name	string
volume_fraction	double
ID_set[]	array of int
SCFUnitIDSet	class SCFUnitIDSet
type	int 0 : polymer/1 : solvent
IDSet[]	array of int: molecular ID, subchain ID, state ID
ShapeByGrids	class ShapeByGrids
input_filepath	string
region_of_obstacle:select	[select] IN/OUT
SUSHIInput	class SUSHIInput
calculation_method	CalculationMethod class
restart	[select] START/CONTINUE/RESTART/RESTART_READMESH
solver_parameter	SolverParameter class
mesh	Mesh class
type_of_free_propagator_of_regular_mesh	[select] 1NN-P/2NN-NP/2NN-P/3NN-P
SUSHIOutput	class SUSHIOutput
volume_fractions	VolumeFractions class
flag_of_dynamics	int
phi	ScalarField class
V	ScalarField class
free_energy	double
excess_free_energy	double
optional_output	OptionalOutput class
flag_of_convergence	int
ScalarArray	class ScalarArray
comp[]	array of double
ScalarField	class ScalarField
name	KEY
num_of_component	int
value[]	array of ScalarArray class

UDF parameter	Description
SegmentVolumeFractionCondition	class SegmentVolumeFractionCondition
polymer_ID	int
subchain_ID	int
begin_length	double
end_length	double
Shear	class Shear
shear_rate	double
shear_period	double
Solvent	class Solvent
name	string
specific_volume	double
SolverParameter	class SolverParameter
type	[select] ADF/FH/SCF
State	class State
name	string
probability	double
StaticConditions	class StaticConditions
polydispersity_conditions[]	array of PolydispersityCondition class
symmetry_conditions[]	array of SymmetryCondition class
domain_specification_conditions[]	array of DomainSpecificationCondition class
constraint_conditions[]	array of ConstraintCondition class
SubchainMap	class SubchainMap
reactant_ID	int
subchain_ID_of_reactant	int
subchain_ID_of_product	int
SubchainUnit	class SubchainUnit
polymer_ID	int
subchain_ID	int
SurfaceChiParameter	class SurfaceChiParameter
boundary_name	[select] XMin/XMax/YMin/YMax/ZMin/ZMax RMin/RMax/HMin/HMax
obstacle_ID	int
SymmetryCondition	class SymmetryCondition
parent_path	Path class
child_path	Path class
SystemSizeDynamics	class SystemSizeDynamics
parameter_of_system_size_dynamics	double
compressibility	double
interval_of_system_size_dynamics	int
SystemSizeOptimaization	class SystemSizeOptimaization
max_iteration_of_optimization	int
cubic_system	int 0:no / 1:yes
TransitionStateProbabilities	class TransitionStateProbabilities
probabilities[]	array of double

UDF parameter	Description
Vector3D	class Vector3D
x	double
y	double
z	double
VolumeFraction	class VolumeFraction
Id	int
volume_fraction	double
ensemble	[select] CANONICAL/GRANDCANONICAL
VolumeFractionData	class VolumeFractionData
name	string
volume_fraction	double
VolumeFractionOnBoundary	class VolumeFractionOnBoundary
Id	int
boundary_name	[select] XMin/XMax/YMin/YMax/ZMin/ZMax RMin/RMax/HMin/HMax
VolumeFractions	class VolumeFractions
polymer_volume_fractions[]	array of VolumeFraction class
solvent_volume_fractions[]	array of VolumeFraction class
VolumeFractionsOnBoundaries	class VolumeFractionsOnBoundaries
polymer_volume_fractions[]	array of VolumeFractionOnBoundary class
solvent_volume_fractions[]	array of VolumeFractionOnBoundary class

5.13 Seed フォーマット定義

Seed は、SUSHI の開発当初に、SUSHI のコアの開発を目的に開発された簡易データ入力用フォーマットである。今後の開発にも有用と考え、そのマニュアルをここに記載する。Seed の読み込みは、C++の STL の multimap や string を多用したプログラムにより実装されている。速度やメモリー利用の観点から大規模なデータの入力には適さない。しかし、小規模なデータ入力には必要十分である。入力用ファイルは UNIX 形式のテキストファイルとする。MS-DOS 形式の CR LF で行が終わるテキストファイルは受け付けない。基本的なデータ列は次のようになる。

```
Key data1 data2 data3 ........
```

最初の Key はこの行を検索するための文字列のキーワードである。以降に続く data1,data2,... は文字列で記述されたデータであり、セパレータは空白文字または tab である。データ列は改行で終わる。もし、データが複数行に渡る場合は Key に続くデータ列を"(","）"でくくる。例えば

```
Key ( data1 data2 data3 .......
      data11 data12 data13 ...........)
```

とする。"(","）"の前後にはセパレータを挿入し、Key やデータが続いてはならない。データ列は Key に対するマルチマップとして記憶され、Key が重複する複数のデータ列も許される。データ列中のデータは配列番号により検索される。Key とデータ列からなるマルチマップを mmapWords とする。Seed の基本構造はこの mmapWords をデータとして持つ構造体である。さらにこの構造体に Key と Seed からなるマルチマップ mmapSeeds を追加する。C++のクラスで非常に簡単にこの構造を実現すると以下のように書ける。

```
class Seed {
    Seed* pParent;
    multimap< string, map< int, string > > mmapWords;
    multimap< string, Seed* >              mmapSeeds;
};
```

ここで pParent は親となる Seed のポインタである。この構造によりどのような階層構造も読むことを可能とする。

データ構造の書式は"{","}"でくくり記述するものとする。例えば

```
Key1 data1 data2 data3 ........
Key2 data1 data2 data3 ........
.....
Key3 {
    Key1 data1 data2 data3 ........
    Key2 data1 data2 data3 ........
    .......
    Key3 {
        ..........
    }
    ........
}
Key4 {
    ........
.......
```

のような形で記述されるのが Seed である。データ列と Seed 列は、Key とマルチマップにより保管されので、データの出現の順序は同じ Key を持つ列にのみ意味を持つものとする。これにより異なる Key のデータ列やデータ構造を間に挿入することが許される。例えば

```
Key1 data1 data2 data3
Key1 data4 data5 data6
Key2 data7 data8 data9
```

と

```
Key1 data1 data2 data3
Key2 data7 data8 data9
Key1 data4 data5 data6
```

は同じデータを意味する。Seed に固定フォーマットはない。SUSHI は構造に矛盾がなければそのデータを読み、必要なデータを抽出する。

5.14 Seed での入力方法

以降の説明では、”.....” の記述は、”.....”の前にあるデータ、上部のデータ列、または上部のデータ構造が連続することが可能であることを意味する。Key に続く [] で囲まれものが入力データである。入力データの型は UDF を参照のこと。

基本制御の入力は次のとおりである。

```
SOLVER                  [計算ソルバー]
    //          ソルバーとは用途が特化されたシミュレータのことである。
    // ADF ：(Approximate Density Functional) 法。開発中。
    // SCF ：(Self Consisitent Field) 法。
    // RPA : (GL theory using RPA) 法
RPA_SCF   [SCF 回数] [RPA 回数]   //  ハイブリッド法
CALCMETHOD              [動的平均場計算か静的平均場平衡計算の選択]
    // DYNAMICS ：動的な時間発展の計算。
    // STATICS  ：静的平衡計算。
RESTART
    // リスタート用フラグ。UDF 入力に同じ。
    // 引数なし RESTART
    // RESTART
    // CONTINUE
    // RESTART_READMESH
```

5.14.1 SCF 計算用のパラメータ Key

```
DEL_S                   [経路積分用の長さの刻み幅]
CONST_V                 [化学ポテンシャル更新用定数]
CONST_W                 [セグメント間相互作用更新用定数]
ERROR                   [収束判定用しきい値]
RANDOM_SEED             [乱数発生用シード]
STANDARD_DEVIATION      [場の初期値に与えるノイズの標準偏差]
```

```
JUDGEMETHOD          [収束判定方法]
    //  ダイナミクス計算でのみサポートされる。
    // ABSOLUTE ：φの絶対誤差で収束判定する。
    // RELATIVE ：φの相対誤差で収束判定する。ABSOLUTE より判定は厳しい。
JUDGESTEP            [収束判定のステップ間隔]
MAX_COUNT            [SCF 計算における最大繰り返し回数]
SCF_OUTPUTSTEP       [SCF 計算中の出力ステップ間隔]
MATRIX_ELEMENT_TYPE  [離散化ラプラス演算子のタイプ]
    // 1NN-P, 2NN-NP, 2NN-P, 3NN-P。
    // 第 2.7.5 節参照。
SCFMETHOD            [経路積分の取り扱い方法]
    //  INCORE  ：メモリーに保存。
    //  DIRECT  ：必要な場合に再計算をする。
PATHINTEGRAL_SCHEME  [経路積分の差分スキーム]
    //  IMPLICIT  ：
    //  EXPLICIT  ：デフォルト
```

5.14.2 ダイナミクス計算用のパラメータ Key

動的な時間発展の計算が選択された場合に有効。 第 2.7.10 節参照。

```
DEL_T                [ダイナミクス計算用時間刻み幅]
VARIABLE_DEL_T       [最大時間刻み幅]    [時間刻み幅更新係数]
NTIME                [最大ステップ数]
OUTPUTSTEP           [出力ステップ間隔]
ARCHIVESSTEP         [アーカイブスファイルへの出力ステップ間隔]
LOGSTEP              [ログへの出力ステップ間隔]
DYNAMICS_SCHEME      [動力学計算の差分スキーム]
    //  IMPLICIT  ：
    //  EXPLICIT  陽的解法。通常はこちらでよい。Euler 法
    //  EXPLICIT2 陽的解法の 2 次。2 段階 Runge-Kutta 法
 COMPRESS_DY          [圧縮率：SCF 計算のみで利用可能　第 2.6.2 節参照]
```

5.14.3 メッシュ

第 2.7.3 節参照。

メッシュ

```
MESH {
    NAME [名前]
    TYPE [タイプ]
         // REGULAR      ： 規則メッシュ。
         // RECTANGULAR  ： 直方メッシュ。
         // SPHERICAL    ： 球座標メッシュ。
         // CYLINDRICAL  ： 円筒座標メッシュ。
```

```
    X [X 軸の最小値] [最大値] [分割数] [MPI での領域の分割数（MPI 版でのみ有効）]
    Y [Y 軸の最小値] [最大値] [分割数] [MPI での領域の分割数（MPI 版でのみ有効）]
    Z [Z 軸の最小値] [最大値] [分割数] [MPI での領域の分割数（MPI 版でのみ有効）]
    X [メッシュ点の座標] .....（rectangular メッシュ）
    Y [メッシュ点の座標] .....（rectangular メッシュ）
    Z [メッシュ点の座標] .....（rectangular メッシュ）
    R [R 軸の最小値] [最大値] [分割数]
    H [H 軸の最小値] [最大値] [分割数]
}
```

境界条件

第 2.7.4 節参照。

```
BOUNDARYCONDITION {
    X [X 軸の最小側での条件] [最大側での条件]
    Y [Y 軸の最小側での条件] [最大側での条件]
    Z [Z 軸の最小側での条件] [最大側での条件]
    R [R 軸の最小側での条件] [最大側での条件]
    H [H 軸の最小側での条件] [最大側での条件]
        // PERIODIC           ：周期境界条件。
        // DIRICHLET or WALL  ：吸収壁。Dirichlet 境界条件で指定された値が 0 の条件。
        // NEUMANN            ：反射壁。NEUMANN 境界条件で指定された値が 0 の条件。
        // PERIODIC 以外は両端の条件を入力する。
    VOLUME {  // 境界上での体積分率の拘束条件。UDF の詳細説明を参照のこと。
        POLYMER {
            FRACTION [ポリマーの ID] [軸端] [体積分率]
            .....
                // ID は COMPONENT で記述した順番。0 から始まる。
                // 軸端　指定した軸端に垂直な面が拘束面である。:
                // XMin, XMax, YMin, YMax, ZMin, ZMax, RMin, RMax, HMin, Hmax
        }
        SOLVENT {
            FRACTION [溶媒の ID] [軸端] [体積分率]
            .....
                // ID は COMPONENT で記述した順番。0 から始まる。
                // 軸端　指定した軸端に垂直な面が拘束面である。:
                // XMin, XMax, YMin, YMax, ZMin, ZMax, RMin, RMax, HMin, Hmax
        }
    }
};
```

5.14.4 モノマーの入力

第 2.2、2.3 節参照。

モノマー

```
MONOMER {
    CHARACTER [名前] [有効体積] [有効長]
    .....
        // 名前：内部状態をもつ場合、例えば、AB 2成分からなる傾斜濃度のポリマーの場合は、
        //       "taperAB" のように名前をつけておく。
        // 有効体積：経路積分とモノマーの体積分率の計算に利用される。
        //         内部状態をもつ場合は各状態の 有効体積 から再計算され上書きされる。
        // 有効長： 経路積分の各線係数の計算に利用される。
}
```

部分鎖の内部状態

部分鎖が内部状態を持つ場合はその特性を入力する。内部状態を持つ部分鎖が無ければ不要。

```
MONOMERSCFCHARTABLE {
    MONOMERSCFCHAR {
        NAME [名前]
            // この名前と同名の名前の MONOMER が入力されなくてはならない。
            // 例えば、AB 2成分からなる傾斜濃度のポリマーは "taperAB" のように。
        STATE { // 内部状態の配列。
            DATA [名前] [部分鎖内での存在確率]
        .....
        }
        TRANSITIONSTATEMATRIX {
            NMATRIX [内部状態間遷移確率の数]
            PROBABILITY [内部状態の遷移確率] .......
                // 正方向（ junction ID の小から大）への遷移確率。
                // 遷移確率とは部分鎖上の各内部状態の存在確率である。
                // 内部状態の遷移確率：STATE の DATA の数だけ記述されなくてはならない。
                // 部分鎖上の位置は配 PROBABILITY の番号に対応し、
                // NMATRIX=PROBABILITY の数=部分鎖の長さ/delta_s+1 でなければならない。
                // 最初の PROBABILITY は経路積分開始点における各内部状態の初期値である。
        }
    }
    .....
}
```

5.14.5 ポリマーと溶媒の入力

選択できる分子（ポリマーおよび溶媒）の構造を入力する。入力された分子が計算に利用されるのは以降で説明する体積分率が入力された場合である。第 2.1、2.2、2.7 節参照。

コンポーネント（分子の構造入力）

```
COMPONENT {
    //ポリマーは部分鎖の結合により入力される。部分鎖はモノマーの種類と長さで入力される。
    POLYMER {
        TYPE [部分鎖の結合方法を選択するキーワード]
            // HOMO    ：部分鎖１本からなるホモポリマーとなる。
            // BLOCK   ：部分鎖の配列の順に結合したブロックポリマーとなる。
            // STAR    ：１つの結合点に全ての部分鎖が結合した星型ポリマーとなる。
            // COMB    ：櫛形ポリマーとなる。部分鎖の配列は 主鎖-側鎖-主鎖-側鎖-...
            //  と認識される。例えば A1, B1, A2, B2, A3 の順に部分鎖の配列が
            //  入力されると  A1--+--A2--+--A3
            //                      B1     B2
            //  のような櫛型になる。
            // GENERAL：部分鎖の結合は部分鎖両端の結合点の ID の指定により入力される。
            // TYPE が省略されると HOMO がデフォルトとなる。
        BLOCK [部分鎖を構成するモノマー名] [部分鎖の長さ]
        .....
            // この名前のモノマーは MONOMER で入力されなくてはならない。
        JUNCTION [結合点の ID] [結合点の ID]
        .....
            // ブロック両端の結合点の ID を記述する。
            // TYPE GENERAL のときのみ有効。
            // ID は 0 から始まらなくてはならない。
            // 例えば、 Block の要素が 1 のときこのペアに [ 0, 0 ] の ID を入力すると
            // 両端が結合されたリング状のポリマーとなる。
    }
    .....
    Solvent [名前] [有効体積]
    .....
}
```

5.14.6 体積分率

体積分率を入力された分子が計算に利用される。アンサンブルが異なると体積分率の意味も異なる。第 2.7.8 節参照。

```
VOLUME {
    POLYMER {
        FRACTION [ポリマーの ID] [体積分率] [アンサンブル] [バルクでの体積分率]
        .....
            // ID は COMPONENT で記述した順番。 0 から始まる。
            //  アンサンブル：
            // CANONICAL       カノニカル アンサンブル。
            // GRANDCANONICAL グランドカノニカル アンサンブル。
            //                 この選択がなされた場合ダイナミクス計算はできない。
```

```
    }
    SOLVENT {
        FRACTION [溶媒の ID] [体積分率] [アンサンブル] [バルクでの体積分率]
        .....
            // ID は COMPONENT で記述した順番。 0 から始まる。
            //  アンサンブル：
            // CANONICAL       カノニカル アンサンブル。
            // GRANDCANONICAL グランドカノニカル アンサンブル。
            //                 この選択がなされた場合ダイナミクス計算はできない。
    }
}
```

バルクでの体積分率は通常は必要ない。UDF の詳細説明を参照のこと。

5.14.7 セグメント間相互作用パラメータ（χ パラメータ）

6 章を参照のこと。

```
CHI {
    ZBE [セグメント i の名前] [セグメント j の名前] [χ ij の値]
        // 自動的に χ ij = χ ji となる。
        // 入力されていないセグメント間の χ ij は 0 となる。
        // セグメントの名前とは MONOMER と SOLVENT の名前を意味する。
}
```

5.14.8 外的条件

外的条件を入力するのに利用されるポリマー、部分鎖、結合点の ID は COMPONENT で入力した分子の順番に対応する。番号は 0 から始まる。

壁面とセグメント間の相互作用パラメータ（χ_s）

第 2.7.7 節参照。

```
SURFACECHI {
    // 相互作用を働かせる面とセグメントとパラメータの値を設定する。
    ZBE [軸端] [対象となるセグメントの名前] [パラメータ値] [各オブスタクルの ID]
    .....
        // 軸端　指定した軸端に垂直な面がグラフト面である。:
        // XMin, XMax, YMin, YMax, ZMin, ZMax, RMin, RMax, HMin, Hmax, PARTICLE, FIBER
        // 各オブスタクルの ID  PARTICLE/FIBER を選択した場合のみ有効
}
```

グラフト条件

自由端の初期状態を設定することによりグラフト鎖を入力する。第 2.7.6 節参照。

```
GRAFT {
    // グラフトさせる面とポリマーの末端を設定する。
    SET [軸端] [グラフトするポリマーの ID] [グラフトする JUNCTION の ID] [各オブスタクルの ID]
    .....
        // 軸端　指定した軸端に垂直な面がグラフト面である。:
        // XMin, XMax, YMin, YMax, ZMin, ZMax, RMin, RMax, HMin, Hmax, PARTICLE, FIBER
        // 各オブスタクルの ID　PARTICLE/FIBER を選択した場合のみ有効

}
```

マスク条件

自由端の存在可能領域を設定する。ミセル等の計算に利用する。

```
MASK {
    JUNCTION {
        ID    [グラフトするポリマーの ID] [マスクする自由端点の ID]
        AXIS  [軸の名前] [領域の最小値] [領域の最大値]
        .....
            // 軸の名前: X,Y,Z,R,H
    }
    .....
}
```

5.14.9 静的平衡計算で有効な外的条件

多分散静的平衡計算用の最大鎖長の指定

静的平衡計算かつ単一内部状態の HOMO ポリマー多分散系にのみ有効である。第 2.7.12 節参照。

```
POLYDISPERSITY {
    SET [最長 HOMO ポリマーの ID] [経路積分を共有させるポリマーの ID]
    .....
}
```

静的平衡計算用の対称経路積分の指定

静的平衡計算時に対称性のある経路積分の指定を行う。この指定により経路積分の計算時間を短縮できる。例えば、$A_{10}B_{10}, A_{20}B_{20}$ のような 2 本の鎖のある計算の場合、鎖長の長い $A_{20}B_{20}$ の鎖の自由端から接合点への経路積分は経路積分の初期値 (2.14) 式は鎖長の短い $A_{10}B_{10}$ の同じセグメント種の経路積分の初期値と同じで共用できる。一方、接合点から自由端への経路積分は経路積分の初期値が異なるので共用できない。

```
SYMMETRY {
    PATH [ID1] [ID2] [J1] [J2] [ID3] [ID4] [J3] [J4]
    .....
        // 経路積分が共用される経路の指定。部分鎖と経路積分の方向を入力する。
        // ID1：ポリマーの ID。
        // ID2：部分鎖の ID。
        // J1 ：経路積分を開始する接合点の ID。
        // J2 ：経路積分を終了する接合点の ID。
        // 経路積分を共用する経路の指定。部分鎖と経路積分の方向を入力する。
        // ID3：ポリマーの ID。
        // ID4：部分鎖の ID。
        // J3 ：経路積分を開始する接合点の ID。
        // J4 ：経路積分を終了する接合点の ID。
}
```

セグメントの体積分率

体積分率を知りたいセグメントの範囲指定をする。現バージョンではループの無いポリマー構造に対しての計算が行える。

```
SEGMENT {
    SET [ポリマーの ID] [部分鎖の ID] [部分鎖上の開始点の長さ] [部分鎖上の終了点の長さ]
    .....
         // 部分鎖両端の JUNCTION ID の小さい方が長さ 0 の基点とする。
}
```

ドメイン領域の指定

静的平衡計算において自己無撞着場の初期値の自動設定により目的とするドメインを簡単に発生させるのに利用する。

```
DOMAIN {
    SPECIFICATION {
        NAME [セグメントの名前]
        REGION [軸の名前] [初期濃度を高くする領域の最小値] [領域の最大値]
        .....
            // 軸の名前: X,Y,Z,R,H
    }
    .....
}
```

部分鎖の慣性半径の指定

部分鎖の慣性半径を計算するときに入力する。

```
RG {
    ID [ポリマーの ID] [部分鎖の ID]  //  部分鎖の ID が-1 の場合鎖全体の慣性半径を出力。
```

```
    .....
}
```

散乱関数

散乱関数を計算するときに入力する。

```
SCATT [リスタートオプション]
     //  RESTART  既に出力された構造の散乱関数のみを計算しレコードとし
     //  て出力する。構造は出力しない。
COEFFORMESHWIDTH [1 D の散乱関数用の出力メッシュ幅調整用の係数]
     // メッシュ幅=この値 × 最小メッシュ幅（2 π/最長軸長）
}
```

SCF 拘束条件

SCF 法で計算する場合に、拘束をかけ変化させない条件を指定する。

```
CONSTRAINT {
    POLYMER {
        TARGET [拘束する対象] [ポリマーの ID] [部分鎖の ID] [ステイトの ID]
        .....
            //  拘束する対象
            //    PHI  体積分率。現実装はこれのみ。
            //  ステイトの ID  内部状態を持たないモノマーであれば記述の必要なし。
    }
    SOLVENT {
        TARGET [拘束する対象] [溶媒の ID]
        .....
    }
}
```

静的なシステムサイズ最適化

```
MAX_LO          [システムサイズ最適化を行う SCF ステップ間隔]
DL              [辺の長さの数値微分を行うための相対的微小量　デフォルト値：1e-6]
LATTICEERROR    [辺の長さの相対的収束判定誤差　デフォルト値：1e-4]
DUMPPARAM       [辺の相対的変化許容量　デフォルト値：0.4]
```

5.14.10 動力学で有効な外的条件

易動度

モノマーや溶媒の種類に応じて異なる定数の易動度 L_0 を入力できる。入力されていないものの易動度は 1 である。ポリマーの濃度と存在するマトリックス種による移動度を入力できる。体積分率 ϕ が小さいポリマーに対して意味がある入力である。この入力をすると易動度は $L\phi$ として扱われる。L はオプションにより次のようになる。第 2.7.10 節参照。

ROUSE 条件：マトリクスの成分よりも十分鎖長が長く、低濃度の場合。$L = L_0\phi$ として計算される。

REPTATION 条件:マトリクスの成分と同等の鎖長で絡み合うほどの鎖長があり、低濃度の場合。$L = (L_0/N)\phi$ として計算される。N は鎖の全長である。

```
MOBILITY {
    SEGMENT [名前] [易動度]              //  Lo である。
    POLYMER [ポリマーの ID] [タイプ]     //  ポリマーの易動度のタイプの入力。
      // ROUSE
      // REPTATION
    SOLVENT [溶媒の ID] [タイプ]         //  溶媒の易動度のタイプの入力。
      // ROUSE この条件のみ有効。
}
```

化学反応条件

ダイナミクスの計算中に化学反応を入力できる。計算可能な化学反応は

1) A+B+C+.... が D になるような反応。ここで、各成分は Component で入力されていなくてはならない。

2) Junction が反応して他の化合物になる反応である。例えば A,B ポリマーの末端が反応して AB ジブロックコポリマーとなる反応である。

3) 壁にグラフトする反応。

1) A+B+C+.... が D になるような反応

```
REACTION {
    GROUP {
        REACTANT [反応物の種類] [反応物の ID]
        .....
        PRODUCT  [生成物の種類] [生成物の ID] [生成物内での反応物の体積分率] .....
        CONSTANT [反応定数]
    }
    .....
        // 反応物の種類：POLYMER, SOLVENT
}
```

2) Junction が反応して他の化合物になる反応

2 次反応のみが入力可能である。

```
REACTION {
    JUNCTION {
        REACTANT [反応物の種類] [反応物の ID] [反応物の Junction の ID]
        REACTANT [反応物の種類] [反応物の ID] [反応物の Junction の ID]
        PRODUCT  [生成物の ID]
        //部分鎖の対応関係の指定。
        SUBCHAIN [反応物の ID] [反応物の部分鎖の ID] [生成物の部分鎖の ID]
            //  ここでの反応物の ID は上段で説明した REACTANT の ID である。
            //  0か1しかとらない。
        .....
        CONSTANT [反応定数]
    }
```

```
    .....
}
```

3）壁にグラフトする反応

```
REACTION {
    GRAFT {
        REACTANT [反応物の ID] [反応物の Junction の ID]
        PRODUCT  [生成物の ID]
        SUBCHAIN [生成物の部分鎖の ID] ......
              //  部分鎖の対応関係の指定。上記データ列の順番は反応物の部分鎖の ID の順
              //  であり、反応物の部分鎖に対応する生成物の部分鎖の ID を全て入力する。
        CONSTANT [反応定数]
    }
    .....
}
```

ずり流動

```
SHEAR_RATE  [ずり速度]
```

熱ゆらぎ

```
NOISE  [セグメント濃度に対する熱ゆらぎの標準偏差] [乱数の種]
                                                  // 0 を入れる。無視してもよい。
       // 乱数の種に 0 以上の値を入力するとその値を乱数の種として固定してしまう。
       // よって、0 以上の乱数の種はデバッグのために利用する。
```

動的なシステムサイズ最適化

動力学と同時にシステムサイズを最適化する。第 2.7.18 節参照のこと。

```
LATTICEOPT_COEF  [(2.103) 式の Q 値]
COMPRESS         [圧縮率　(2.104) 式で利用　デフォルト値：0.]
LO_INTERVALSTEP  [システムサイズを最適化するダイナミクスステップ間隔]
SSO {
    DIRECTION [X 軸用最適化フラグ] [Y 軸用最適化フラグ] [Z 軸用最適化フラグ]
    ....
    // もし、x,y 軸を比を固定したまま最適化したいなら以下のように入力する。
    //   DIRECTION 1 1 0
    // もし、x,y 軸をそれぞれ独立して最適化したいなら以下のように入力する。
    //   DIRECTION 1 0 0
    //   DIRECTION 0 1 0
}
```

流体力学的効果

動力学に流体力学的効果を　導入する。第 2.7.5 節参照のこと。

```
HYDRO {
    DENSITY [密度]
    VISCOSITY [セグメント名]　[粘度]
    ....
    IMPLICIT　//IMPLICT 法を利用するなら記述する。
    POISSON { // 圧力を求めるためのポアソン方程式用のパラータ
        ERROR　　[収束判定誤差　10^-10 程がよい]
        MAXSTEP　[ICCG 用最大繰り返し計算回数]
    }
    //　熱ゆらぎを Navier-Stokes 方程式へ入れるためのパラメータ
    NOISE [セグメント濃度に対する熱ゆらぎの標準偏差] [乱数の種]
                                                    // 0 を入れる。無視してもよい
}
```

5.14.11　SCF モンテカルロ法

2.7.17 節参照のこと。CALCMETHOD が “MONTECARLO” と設定された場合この入力が必要となる。

```
MONTECARLO {
    POLYMER { // Specify the polymer whose junctions are updated
              // in the Monte Carlo simulation.
        ID [ポリマーの ID]
        JUNCTION [Junction の ID] [X 座標] [Y 座標] [Z 座標]
        .....
             // Set the initial positions of junctions.
    }
    .....
}
```

5.14.12　静電場

```
ELECTROSTATIC {
    DIELECTRIC [系の誘電率]
        // この値に正の値が入力されると静電場の計算が有効となる。
    CHARGE [セグメントの名前] [単位体積当たりの電荷] [比誘電率]
        // 5.7.12 節を参照。
    .....
    ALPHA [direction] [value of alpha]
        // 動力学用外部電場パラメータ
        // direction:　X, Y, and Z, 2.7.6 節参照
    EFIELD [x value of E0]　[y value of E0]　[z value of E0]
        // 外部電場ベクトル, 2.7.6 節参照
        // 注意! ALPHA と EFIELD は同時に利用できない。
```

```
    POISSON { // Parameters for Poisson solver of ICCG method.
        OMEGA [SOR 法の加速パラメータ] // The parameter for SOR method.
                    // The value 0.8 is recommended as the default value.
                    // 現バージョンでは利用していない。
        ERROR  [ICCG 法の収束判定用誤差の値]
                    // The value 10^-10 (previous 0.0001 is a bug) is recommended as the default valu
        MAXSTEP [ICCG 法の最大繰り返し計算回数]
                    // The value 10000 is recommended as the default value.
    }
}
```

5.14.13 オブスタクル

```
OBSTACLE {
    NDIV [オブスタクルの境界を設定するための 1 メッシュ当りの分割数]
         // 1000 ぐらいがよい。
    PARTICLE [中心の X 座標] [中心の Y 座標] [中心の Z 座標] [半径] [IN or OUT]
                                                              // オブスタクルの領域
                                                    // 球殻の内側か外側
    FIBER [片末端の X 座標] [片末端の Y 座標] [片末端の Z 座標]
          [片末端の X 座標] [片末端の Y 座標] [片末端の Z 座標] [半径]
          [IN or OUT]                [IN or OUT]
          // オブスタクルの領域       // エンドキャップの有無
          // ファイバーの内側か外側
    .....
}
```

5.15 並列計算の制限

並列計算には現在のところ以下の制限がある。

1. 削除：周期境界条件のみ計算可能
2. 変更：GPU 計算では各軸の分割数は 8 の倍数が望ましい。
3. 削除：MPI 計算では各軸は 2 以上で分割し多領域としなければならない。
4. SSO は利用できない。
5. オブスタクルは設置できない。
6. 削除：MPI で GRPA 法は利用できない。

利用不可能な機能を利用しようとした場合、警告を発して SUSHI は停止する。改良によりこれらの制限は逐次解消されるはずである。

第6章　簡易Pythonツール

SUSHI の python スクリプトのほとんどは、アクションとしての実行に移行している。*.py ファイルについては参考程度の記述と認識して頂きたい。実際の利用は以降のアクションファイルの節を参照して頂きたい。

6.1　出力対象

SUSHI では SCF ユニットがセグメントの体積分率の最小分割単位となる。ユーザーは SCF ユニットの組み合わせで様々なセグメントの体積分率の分布を出力することができる。SCF ユニットについては **??**節を参照して頂きたい。また、“Composition”データも参考にして頂きたい。SUSHIOutput.phi は SCF ユニットに分割されたセグメントのスカラー場を要素とするデータである。

6.2　sushi_show.py

“sushi_show.py” パイソンスクリプトは SUSHIOutput.phi を出力するプログラムであり、その挙動はメッシュの空間次元により異なる。

6.2.1　１次元の場合

以下の説明は第 3 章の記述と同様なものです。この章を参考にしてください。

Python 窓で SUSHI 用の Python スクリプト sushi_show.py を Load して Run ボタンを押してください。そして、最下段のメッセージ窓に表示されたコメント一式をコピーしてください。次に Plot 窓へ移り、左上にある GraphSeet[] のアイコンをピックして有効にしてください。そして Make ボタンを押してください。すると Plot 窓に出力があります。次に Clean ボタンを押して Plot 窓に表示された出力をを消去してから、先ほどコピーした Python 窓の表示をペーストして Plot 窓に移してください。Plot ボタンを押しますと、グラフが出力されるはずです。ペーストした内容は Gnuplot のコマンドです。このコマンドを適当に変更するとお好みの表示が得られます。不明な点は GOURMET の Plot の利用方法を参照してください。

SCF ユニットの選択

スクリプト中の”component_list”に SCF ユニット ID をセットするとユーザが望む SCF ユニットまたはそれの和が出力できる。一例として ID 0 と ID 1 の SCF ユニット両方を出力したければ
component_list = [[0], [1]]
のように入力する。内部括弧 “[]” は必須である。もし、和を出力したければこの内部括弧内に ID を列記する。例えば ID 0,1 の和と ID 2 の 2 種類を出力したければ、
component_list = [[0,1], [2]]
と入力する。とにかく 1 種類の出力だけでも内部括弧を忘れないように。

6.2.2　２、３次元の場合

２、３次元の場合 “sushi_show.py” スクリプトをロードして “Run” ボタンを押すと “Window/Viewer” 窓が出力されモルフォロジーが図示される。デフォルトは ϕ[0] (最初の SCF ユニット) が図示される。他の SCF ユニットを出力したい場合は以下の行を変更する。
componentCoef = [0]
括弧内整数を出力したい SCF ユニットの ID にリストに変更する。ただしこの場合、ID の負の値も許される。ID の符号は和、差の違いを表す。例えば、
componentCoef = [0, -1]
と入力すると phi[0] - phi[1] のプロファイルが出力される。
着色して出力される範囲を変更したい場合は以下の行をコメントアウト（#を取る）してから変更する。
`#crange=( 0., 1. )`
括弧内は出力範囲の最小値と最大値である。

6.3　sushi_show3color.py

３つの SCF ユニットを含む出力の場合、各セル中で最大の存在確率の成分の色に塗られた結果が出力される。利用方法は “sushi_show.py”をロードし “RUN” ボタンを押す。

6.4　sushi_show_surf.py

セグメントの等値面が３次元表示される。利用方法は “sushi_show.py”をロードし “RUN” ボタンを押す。

6.5　アクションファイル

GOURMET 上で SUSHIOutput アイコンにマウスを移動し、右ボタンを押すと次の図のようにアクションファイルを呼び出せる。

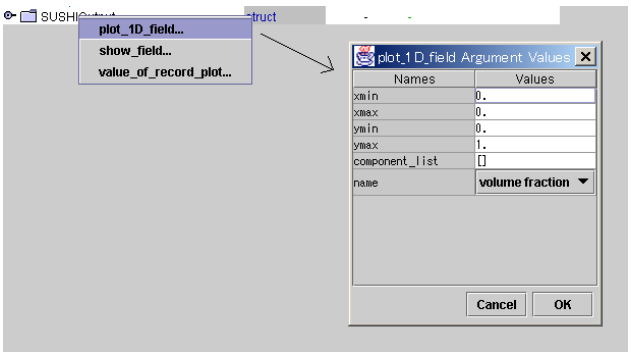


図 6.1: アクションファイルの選択窓

6.5.1 plot_1D_field

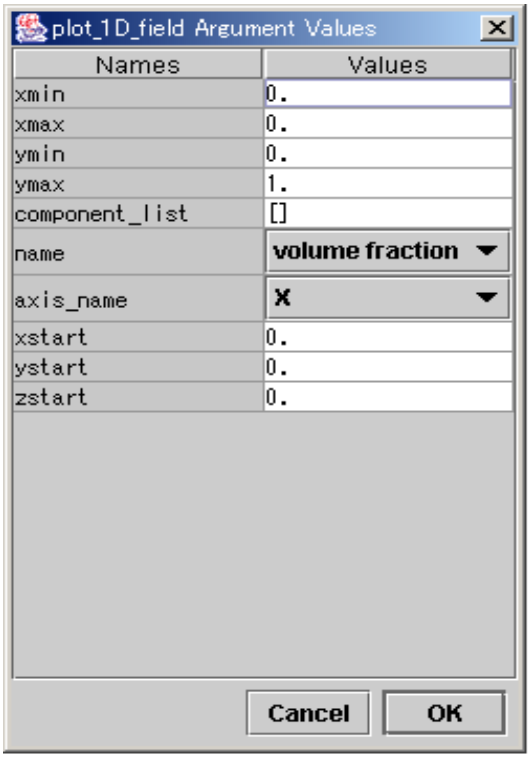


図 6.2: ”plot 1D field”のパラメータ入力窓

1) xmin: X 軸の最小値

2) xmax: X 軸の最大値

3) ymin: Y 軸の最小値

4) ymax: Y 軸の最大値

5) component_list: 出力するスカラー場の ID のリスト。
例として ID 1 と ID 2 を出力したい場合 [[1], [2]] と入力する。
他の例として ID 1 と ID 2 の和と ID 3 を出力したい場合は
$[[1, 2], [3]]$ と入力する。
内部括弧 “[]” は成分が 1 つでも必要である。忘れてはならない。

6) name: 出力したい場の名前を選択する。

7) axis_name: 出力する軸の名前

8) xstart: 出力を開始する X 軸または R 軸の位置（2 D 以上で有効）

9) ystart: 出力を開始する Y 軸または H 軸の位置（2 D 以上で有効）

10) zstart: 出力を開始する Z 軸の位置（2 D 以上で有効）

6.5.2　show_field

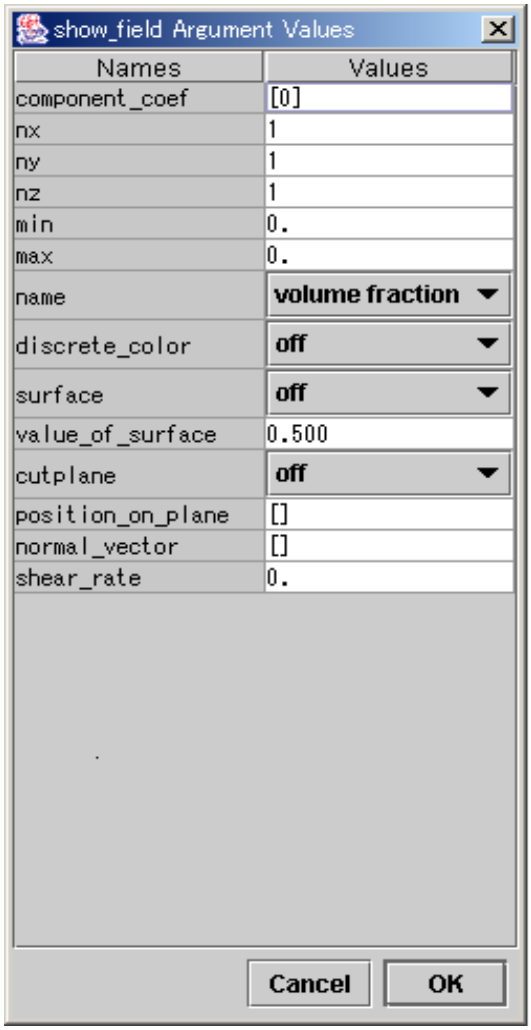


図 6.3: ”show field”用入力パラメータ窓

1) nx: 拡大する X 軸の周期

2) ny: 拡大する Y 軸の周期

3) nz: 拡大する Z 軸の周期

4) component_coef: 出力するスカラー場の ID のリスト（符号付き）。
デフォルト値は “0” である。これを希望の ID に変更する。
1 つの ID ではなく複数のリストにした場合、和や差が出力される。
負の符号の ID を入力すると負の符号の場が出力される。例えば
component_coef = [0, -1] の意味は以下の場の出力を要請したことになる。
field[0] - field[1].

5) min: 着色範囲の最小値

6) max: 着色範囲の最大値
デフォルトは　min = 0 と max = 0　となっている。
この意味は自動的に最小値と最大値が着色範囲になることを意味する。

7) name: 出力したい場の名前を選択する。

8) discrete_color: 多成分を多色に塗り分けたければこのスイッチを on にする。

9) surface: 等値面を 3 次元表示したい場合このスイッチを on にする。

10) value_of_surface: 等値面の値。

11) cutplane: 3 次元表示の場合断面を表示したければこのスイッチを on にする。

12) position_on_plane: 断面上の位置ベクトル。

13) nomal_vector: 断面の法線ベクトル。

14) shear_rate: ずり速度。ずりが印加された場合自動的に入力されるので特に入力する必要はない。

discrete_color、surface、cutplane の 3 つのスイッチのうち 1 つしか有効にならない。

6.5.3 value_of_record_plot

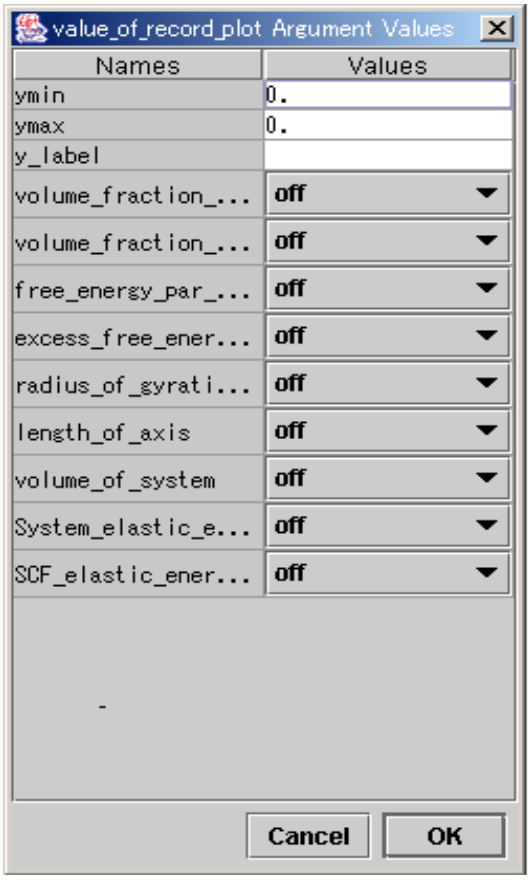


図 6.4: ”record plot”用の入力パラメータ窓

動力学計算や SCF モンテカルト計算のように複数のレコードが出力される計算の場合、各レコード上の出力値を 2 次元プロットするアクションファイルである。動力学計算の場合時間が横軸に、SCF モンテカルロ計算の場合ステップ番号が横軸となり選択された値がプロットされる。

1) ymin: Y 軸の最小値

2) ymax: Y 軸の最大値
デフォルト値は ymin = 0 と ymax = 0. である。
これは自動的に最小値と最大値が設定されることを意味する。

3) y_label: Y 軸のラベル
入力がなければ自動的に選択された項目となる。

4-12) values to plot: 出力したい項目を選択する。

第7章　χ-パラメタ推算用 簡易 Python scripts

7.1　はじめに

χ-パラメタは、高分子混合系における相溶性や界面張力などの物性を決める重要なパラメタであるにも関わらず、その具体的数値を理論的に求めることは困難である。比較的簡単に χ-パラメタを推算する方法として、溶解度パラメタを用いた方法がよく用いられている。[27, 28, 29]

OCTA システムでは、GOURMET より、高分子の物性データを参照できるように、各種の高分子のデータを記録した UDF ファイルを *POLYMERDATABASE* フォルダ内に、収録している。

以降では、これらの UDF ファイルを総称して *PolymerDatabase* と呼ぶ。

ChiParameterCalculator.py は、*PolymerDatabase* に登録されている溶解度パラメタとセグメントのモル体積のデータを用いて、χ-パラメタの推算値を計算する Python script である。

また、*PolymerDatabase* に溶解度パラメタのデータが登録されていない場合に、原子団寄与法 (group contribution method) を用いて、モノマーユニットの化学的構造から、溶解度パラメタを推算するための Python script として *SolubilityParameterCalculator.py* も同時に提供されている。

7.2　χ パラメタの推算法の原理

7.2.1　Flory-Huggins の格子理論と χ-パラメタ

この節では、Flory と Huggins によって提案された高分子溶液／アロイの格子理論に従って、Flory-Huggins の相互作用パラメタ、いわゆる χ-パラメタの定義を与える。[30]

高分子溶液や高分子アロイの相溶性あるいは溶解性を判定するためには、混合自由エネルギー ΔG^M を求めることが必要である。この混合自由エネルギーは、混合前後の系の自由エネルギーの変化として定義され、一般に、混合組成、温度、分子量などの関数となる。混合に際して体積変化がないものと仮定すると、混合自由エネルギーは、エントロピーの寄与 ΔS^M とエンタルピーの寄与 ΔH^M を用いて次のように書かれる。

$$\Delta G^M = \Delta H^M - T\Delta S^M \tag{7.1}$$

(7.1) 式の各項の具体的な表式を得るために、格子モデルを用いた評価を行う。簡単のために、A と B の 2 種類のホモポリマーからなる体積 $\mathcal{V}$ の混合系を考える。まず、これら 2 種の高分子鎖は、同一の体積をもつセグメントから構成されていると仮定する。A、B それぞれの高分子鎖に含まれるセグメントの総数を N_A および N_B とし、系内にはこれらの高分子鎖がそれぞれ M_A および M_B モル含まれているとする。正則混合溶液の理論を用いると、系の混合エントロピー ΔS^M は、理想気体の混合のエントロピーと同様にして、M_A 本の A 高分子鎖と M_B 本の B 高分子鎖を $M_A + M_B$ 個の格子に配置する場合の数として、次の様に計算される。

$$\Delta S^M = -R(M_A \ln \phi_A + M_B \ln \phi_B) \tag{7.2}$$

ここで、R はガス定数である。また、 ϕ_A と ϕ_B は、A 高分子と B 高分子の体積分率であり、

$$\begin{aligned} \phi_A &= \frac{M_A N_A}{M_A N_A + M_B N_B} \\ \phi_B &= \frac{M_B N_B}{M_A N_A + M_B N_B} \end{aligned} \tag{7.3}$$

で定義される。(7.3) 式を M_A および M_B について解けば、

$$M_A = \frac{\mathcal{V}}{V_r}\frac{\phi_A}{N_A}, \qquad M_B = \frac{\mathcal{V}}{V_r}\frac{\phi_B}{N_B} \tag{7.4}$$

となる。ここで、V_r はセグメントのモル体積であり、A、B 共通の値を持っていると仮定されている。Flory-Huggins 格子モデルにおいては、このセグメントモル体積 V_r は、格子点あたりに割り当てられた空間の体積に相当する量で、有効結合長 b の 3 乗程度の大きさを持つ。

(7.4) 式を (7.2) 式に代入すれば、混合エントロピーの表現として、次の関係式を得る。

$$\frac{-T\Delta S^M}{RT(\mathcal{V}/V_r)} = \frac{\phi_A}{N_A}\ln\phi_A + \frac{\phi_B}{N_B}\ln\phi_B \tag{7.5}$$

Flory-Huggins の格子理論においては、(7.5) 式で与えられる正則溶液の混合のエントロピー以外の全ての混合の効果を、単一のパラメタ χ_{AB} を用いて、以下のように書き表す。

$$\frac{\Delta G^M}{RT(\mathcal{V}/V_r)} = \frac{\phi_A}{N_A}\ln\phi_A + \frac{\phi_B}{N_B}\ln\phi_B + \chi_{AB}\phi_A\phi_B \tag{7.6}$$

この式は、Flory-Huggins の混合自由エネルギーと呼ばれ、パラメタ χ_{AB} は Flory-Huggins の相互作用パラメタ、あるいは単に χ-パラメタと呼ばれる。

上記の定義より明らかなように、χ-パラメタは、セグメント間の接触の相互作用に起因するエンタルピーのみならず、正則溶液の混合のエントロピー (いわゆる combinatorial エントロピー) 以外のエントロピーの寄与（例えば相互作用する 2 つのセグメントの相対的な配向のような内部自由度のエントロピーなど）も含んでいる。また、(7.6) 式の接触の相互作用では、セグメント濃度の積に比例する形が仮定されているが、実際の接触の相互作用はセグメント濃度の複雑な関数になることがふつうである。これらの理由により、(7.6) 式で定義される χ-パラメタは、一般に高分子鎖の重合度や濃度、温度などの複雑な関数になる。

また、2.4 節で導入された K-種セグメントのセグメント比体積 r_K は、K-種高分子のモル体積 V_K と、上記のセグメントモル体積 V_r を用いて、

$$r_K = V_K/V_r \tag{7.7}$$

によって計算される。

7.2.2 溶解度パラメタ

(7.6) 式の相互作用の項を計算するために、最近接格子点の間にだけ接触の相互作用が働くと仮定する。最近接格子点の数を z として、平均場近似の下に接触の相互作用を評価すれば、(7.7) 式で定義される χ-パラメタは以下のように表される。

$$\chi_{AB} = \frac{z}{k_{\mathrm{B}}T}\left[\frac{1}{2}(\epsilon_{AA}+\epsilon_{BB}) - \epsilon_{AB}\right] \tag{7.8}$$

ここで、$\epsilon_{KK'}$ は、最近接格子点対に配置された K-種セグメントと K'-種セグメントの間の相互作用エネルギーである。長距離の相互作用を生じない非極性セグメント対に関しては、

$$\epsilon_{AB} \sim \sqrt{\epsilon_{AA}\cdot\epsilon_{BB}} \tag{7.9}$$

と仮定することは一般に良い近似であることが知られており、この近似を (7.8) 式に用いれば、

$$\chi_{AB} = \frac{z}{2k_{\mathrm{B}}T}(\sqrt{\epsilon_{AA}} - \sqrt{\epsilon_{BB}})^2 \tag{7.10}$$

という表現が得られる。

(7.10) 式のセグメント対相互作用エネルギー ϵ_{KK} は、純粋系の凝集エネルギーを用いて計算することができる。ここで、凝集エネルギーとは、凝集状態にある物質の各構成分子を互いに無限に離れた状態にまで引き

離すために必要とされるエネルギーとして定義される。全体で $\mathcal{N}$ 個の格子点からなる体積 $\mathcal{V}$ の系を考える。全ての格子点が K-種のセグメントで占められた状態を考えると、この系の全相互作用エネルギーは、

$$E_K^{\rm coh} = -\frac{1}{2} z\mathcal{N}\epsilon_{KK} \equiv \mathcal{V}\delta_K^2 \tag{7.11}$$

と表される。ここに、δ_K $(K = A \text{ or} B)$ は溶解度パラメタとよばれるパラメタで、K-種純粋系の凝集エネルギー $E_K^{\rm coh}$ を用いて、

$$\delta_K = \Big(\frac{E_K^{\rm coh}}{\mathcal{V}}\Big)^{1/2} \tag{7.12}$$

で定義される。A と B を、体積分率 ϕ_A と ϕ_B の比率で含む一様な混合系の全相互作用エネルギーは、平均場近似の下では、

$$\begin{aligned} E_{A+B}^{\rm coh} &= -\frac{1}{2} z\mathcal{N}\Big[\epsilon_{AA}\phi_A^2 + \epsilon_{BB}\phi_B^2 + 2\epsilon_{AB}\phi_A\phi_B\Big] \\ &= -\mathcal{V}\Big[\delta_A^2\phi_A^2 + \delta_B^2\phi_B^2 + 2\delta_A\delta_B\phi_A\phi_B\Big] \end{aligned} \tag{7.13}$$

と表される。(7.11) 式と (7.13) 式を用いれば、混合のエンタルピーは次のように書ける。

$$\begin{aligned} \Delta H^M &= (E_{A+B}^{\rm coh} - E_A^{\rm coh}\phi_A - E_B^{\rm coh}\phi_B)/\mathcal{V} \\ &= (\delta_A - \delta_B)^2\phi_A\phi_B \end{aligned} \tag{7.14}$$

この関係式と (7.8) 式から、χ-パラメタの溶解度パラメタを用いた表現として、次の関係式を得る。

$$\chi_{AB} = \frac{V_r}{RT}(\delta_A - \delta_B)^2 \tag{7.15}$$

ここで、V_r は前節で定義したモル体積である。

通常、(7.15) 式のエンタルピー効果以外の寄与（主としてエントロピー効果）を χ_S と書き、これを追加することで、χ-パラメタの最終的な表現として、

$$\chi_{AB} = \chi_S + \frac{V_r}{RT}(\delta_A - \delta_B)^2 \tag{7.16}$$

と書き表すことが多い。ここで、χ_S は、conbinatorial エントロピーなど、(7.15) 式では表現できない効果を表す項で、経験的に

$$\chi_S \sim 0.34 \tag{7.17}$$

という定数 [27] に取られる。

χ-パラメタの最終的な表式である (7.16) 式において、溶解度パラメタ δ_A および δ_B は温度を定めれば物質固有の定数である。一方、セグメントのモル体積 V_r の方には、セグメントの定義の仕方に依存した任意性が残る。従って、(7.16) 式を利用するためには、セグメントのモル体積 V_r を指定すること、すなわちセグメントを何に取るかということを指定する必要が生じる。(セグメントを定義することで、１本の鎖を構成するセグメント数 N_K が定まることになる。)

7.2.3 原子団寄与法

7.2.2 節で示したように、純粋系の溶解度パラメタ（あるいは凝集エネルギー）が求まれば、(7.16) 式により χ-パラメタの推算値を求めることができる。ここで必要になる溶解度パラメタは、実験から求めることができるが、より簡単に計算により推算する方法として、原子団寄与法 (group contribution method) という手法が知られている。この手法は、モノマーユニットの溶解度パラメタを、そのモノマーユニットを構成する個々の原子団からの独立な寄与の和として表す方法である。[27]

例えば、図 7.1 に示されるモノマーユニットの場合には、このモノマーは次のような原子団に分割できる。[27] 各原子団からの凝集エネルギーへの寄与の具体的数値は、色々な研究者によって与えられている。[27, 28, 29]

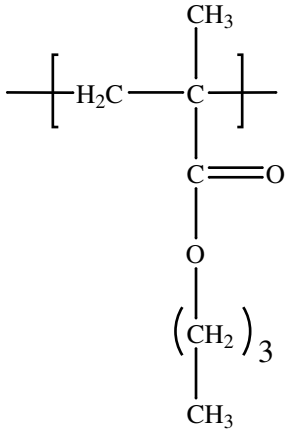


図 7.1: メタクリル酸 n-ブチル・モノマー

原子団	原子団の個数
$-CH_2-$	4
$-CH_3$	2
$>C<$	1
$-COO-$	1

例えば、Hoftyzer と van Krevelen (1976) の数値 [31] を用いれば、

原子団	原子団の個数	凝集エネルギー (J/mol)
$-CH_2-$	4	4190
$-CH_3$	2	9640
$>C<$	1	−5580
$-COO-$	1	13410

$$\text{合計} \quad E^{\text{coh}} = 4 \times 4190 + 2 \times 9640 + 1 \times (-5580) + 1 \times 13410 = 43870(\text{J/mol}) \tag{7.18}$$

一方、このモノマーユニットのモル体積は $V_r = 136$ (cm^3/mol) であることが知られており、この結果溶解度パラメタは

$$\delta = \sqrt{E^{\text{coh}}/V_r} = 18.0 \quad (\text{J}^{1/2}/\text{cm}^{3/2}) \tag{7.19}$$

と求まる。

もし、モル体積 V_r の数値が未知な場合には、Fedors の方法 [32] を用いて推算することができる。この方法では、各原子団についてのモル体積の数値があらかじめ与えられており、凝集エネルギー同様にモル体積も各原子団からの寄与の和として表される。上記の例では、

原子団	原子団の個数	モル体積 (cm^3/mol)
$-CH_2-$	4	16.1
$-CH_3$	2	33.5
$>C<$	1	−19.2
$-COO-$	1	18.0

$$\text{合計}\quad V_r = 4\times 16.1 + 2\times 33.5 + 1\times(-19.2) + 1\times 18.0 = 130.0(\mathrm{cm^3/mol}) \tag{7.20}$$

のように計算される。この計算値は、実測値 $V_r = 136$ ($\mathrm{cm^3/mol}$) に近い値になっていることがわかる。

7.3 各種スクリプトの操作方法

7.3.1 ChiParameterCalculator : PolymerDatabase を用いた χ-パラメタの推算法

PolymerDatabase には、各種モノマーごとに多数の物性値のデータが格納されており、その中には溶解度パラメタ（単位は $\mathrm{J^{1/2}/cm^{3/2}}$) とモル体積 (単位は $\mathrm{cm^3/mol}$) のデータも入っている。これらのデータを用いて χ-パラメタを計算するための Python script が *ChiParameterCalculator.py* である。

ChiParameterCalculator.py を使用する手順は以下の通りである。

(1) GOURMET を起動し、`UDF Editor` より *PolymerDatabase* の polymerdata.udf を開く。

(2) View を Table にセットして、PolymerDatabase → GeneralPolymers[] → Properties[] を開く。

(3) 種々の項目の中で、編集する項目は Flag:int だけである。
１つ上の階層の PolymerDatabase.GeneralPolymers[] を見ると、どの行がどのモノマーに対応しているかがわかるので、χ-パラメタを計算したいモノマー種に相当する行の Flag:int 項目に 0 以外の整数値を入力する。（３個以上のモノマー種を指定することもできる。この場合には、指定されたモノマー種の全ての対についての χ-パラメタが計算される。）

(4) Python パネルの ”Load” ボタンをクリックし、
”SUSHI3/ChiParameterCalculator/python/ChiParameterCalculator.py”
をロードする。

(5) Python Scripting Window をスクロールし、以下の行を表示させる。（スクリプトの 28 行 – 30 行目にある。）

```
###############################################################################
     Vr          = -1.0E99   # Please enter molar volume
                             #   of the segment in ( cm^3 / mol ).
     T           = -1.0E99   # Please enter absolute temperature in ( K ).
###############################################################################
```

これらの行の -1.0E99 の数値を、セグメントのモル体積 ($\mathrm{cm^3}$ / mol) および温度 (K) の数値で置き換える。（この置き換えを行わないと、script の実行時にエラーとなる。）

(6) Python パネルの Run ボタンをクリックして、script を実行する。

(7) 結果は、Python Log Window に示される（χ-パラメタは無次元量）。

注意１：Python script の本文中に、セグメントのモル体積 V_r （単位は $\mathrm{cm^3/mol}$ ） と絶対温度 T （単位は K ）の値を代入してから実行すること。7.2.2 節で述べたように、セグメントのモル体積はユーザーの定義の仕方に依存した任意性があり、このセグメントのモル体積の定義と高分子鎖の長さ N_K は互いに整合するように選ばれなくてはならない。

通常は、セグメントのモル体積の数値としては、低分子溶媒（トルエンなど）のモル体積の代表的な大きさである $\mathrm{V_r} = 100$ ($\mathrm{cm^3}$ / mol) 程度に選ぶとよい。このとき、鎖を構成するセグメント数 N_K は、

$$\begin{aligned} N_K &= \text{(鎖を構成するセグメントの総数)} \\ &= \text{(鎖を構成するモノマーの総数)} \times \text{(モノマーのモル体積)/(セグメントのモル体積 } V_r\text{)} \\ &= \{\text{(鎖の全分子量)} \times \text{(モノマーのモル体積)}\} \\ &\quad /\{\text{(モノマーユニットの分子量)} \times \text{(セグメントのモル体積)}\} \end{aligned} \tag{7.21}$$

に従って計算することができる。この方法で、セグメントのモル体積 V_r を与えれば、K-種の鎖を構成するセグメントの総数 N_K を計算することができるが、有効結合長 b_K を決定するためには、別の条件が必要である。たとえば、鎖の慣性半径 R_G の実測値が与えられると、

$$\frac{1}{6} N_K b_K^2 = R_G^2 \tag{7.22}$$

に従って、有効結合長 b_K を定めることができる。

また、K-種セグメントのセグメント比体積 r_K は、(7.7) 式に従って、

$$r_K = V_K / V_r \tag{7.23}$$

により求められる。

注意２ :*PolymerDatabase* の中で、溶解度パラメタの数値が -1.0E99 となっているときには、データが未定義であることを表す。スクリプトを実行すると、計算結果は表示されず、かわりに次のメッセージが返される。

```
================================================
  The value(s) of the solubility parameter is(are)
  not defined.
  Calculation aborted.
================================================
```

この場合には、正しい溶解度パラメタの数値データを SolubilityParam:double の項目の所に入力する必要がある。（単位は、$\mathrm{J^{1/2}/cm^{3/2}}$ ）もし、溶解度パラメタの数値データが手元にない場合には、次の SolubilityParameterCalculator を用いて、原子団寄与法による溶解度パラメタの推算を行うことができる。

7.3.2 SolubilityParameterCalculator : 原子団寄与法による溶解度パラメタの推算法

溶解度パラメタのデータが手元にない場合には、原子団寄与法を用いて溶解度パラメタの推算を行うことができる。この目的のために、*PolymerDatabase* には原子団寄与法のベースとなる数値データを蓄えた以下のデータベース (UDF ファイル) が用意されており、目的に応じてどれか１つを選択して用いる。

- SolubilityParameter_Hoftyzer&vanKrevelen.udf :
  参考文献 [31] のデータ使用。ただし、モル体積のデータは参考文献 [32] による。
  (参考) 他のデータベースに比べて精度は高いが、登録されている原子団の種類が少ない。

- SolubilityParameter_Dunkel.udf :
  参考文献 [33] のデータ使用。ただし、モル体積のデータは参考文献 [32] による。
  (参考) 登録されている原子団の数は、SolubilityParameter_Hoftyzer&vanKrevelen.udf と同程度。精度もほぼ同じくらい。

- SolubilityParameter_Fedors.udf :
  参考文献 [32] のデータ使用。
  (参考) 精度はやや落ちるが、多数の原子団に関するデータが登録されている。

これらのデータベースから溶解度パラメタを計算するために、*SolubilityParameterCalculator.py* という Python script が用意されている。

SolubilityParameterCalculator.py を使用する手順は以下の通りである。

(1) GOURMET を起動し、`UDF Editor` より *PolymerDatabase* の原子団寄与法の基礎データのデータベース（UDF file 形式) を開く。このデータベースの UDF ファイル名は、
"POLYMERDATABASE/SolubilityParameter_Hoftyzer&vanKrevelen.udf"、
"POLYMERDATABASE/SolubilityParameter_Dunkel.udf"
あるいは
"POLYMERDATABASE/SolubilityParameter_Fedors.udf"
のいずれか１つである。

(2) View を Table にセットして、SolubulityParameterDatabase → GeneralProperties[] を開く。

(3) 種々の項目の中で、編集する項目は NumberOfUnits:int だけである。
GroupName の項目に原子団の構造が記載されているので、計算対象となるモノマーユニットに含まれている原子団ごとに、相当する行の NumberOfUnits:int 項目のところに、そのモノマーユニットに含まれる原子団の個数を整数値で入力する。

(4) Python パネルで Load ボタンをクリックし、
"SUSHI3/ChiParameterCalculator/python/SolubilityParameterCalculator.py"
をロードする。

(5) Python パネルの Run ボタンをクリックして、script を実行する。

(6) 計算結果（凝集エネルギー、溶解度パラメタ、モル体積）は、Python Log Window に示される。

注意１ : この SolubilityParameterCalculator.py で計算した溶解度パラメタの数値は、自動的には保存されないので、*PolymerDatabase* あるいは他の UDF ファイルに、ユーザー自身で値を保存する必要がある。

注意２ : SolubilityParameterDatabase_***.udf の中で、CohesiveEnergy:double の項目、あるいは MolarVolume:double の項目の数値が -1.0E99 となっている場合、この原子団に関する凝集エネルギーあるいはモル体積のデータが未定義であることを表す。この場合にスクリプトを実行すると、計算結果は表示されず、かわりに次のメッセージが返される。**

```
================================================
  The value(s) of the solubility parameter is(are)
  not defined.
  Calculation aborted.
================================================

================================================
  The molar volume of the segment Vr
    or the temperature T is not defined.
  Calculation aborted.
================================================
```

第8章　相関関数解析ツール spcf

spcf の実行については、SUSHI のアクションにて対応した。SUSHIOutput サブフォルダー上にマウスを移動し、右クリック→ SPCF を選択し実行を試して頂きたい。

8.1　概要

SUSHI の実行結果として、高分子の分布構造に対応したセグメント濃度場と呼ばれるスカラー場のデータが得られる。空間相関関数 $\langle\phi(0)\phi(r)\rangle$ はスカラー場 $\phi(\mathbf{r})$ の構造を示す指標の一つであり、解析ツール spcf は、その解析に用いる補助プログラムである。spcf の実行により、空間相関関数に対応する X-Y プロット図用のデータが作成される。ここで、空間相関関数 $\langle\phi(0)\phi(r)\rangle$ は、(8.1) 式のように定義される。

$$\langle\phi(0)\phi(r)\rangle = \frac{\sum_{i,j\in|\mathbf{r}_{ij}|=r} \phi_i\phi_j}{\sum_{i,j\in|\mathbf{r}_{ij}|=r} 1} \tag{8.1}$$

但し、i, j はデータ点を区別するインデックス、ϕ_i, ϕ_j はスカラー場、$\mathbf{r}_{ij}$ は相対位置ベクトルである。数値計算では、スカラー場は離散化されているので、距離 r は連続的には取れない。そのため、spcf では適当な刻み幅 Δr を設定し、距離 r となるデータ点を、位置ベクトル $\mathbf{r}_i$, $\mathbf{r}_j$ を用いて、(8.2) 式のように定義して計算している。

$$|\mathbf{r}_{ij}| = r \qquad \Longrightarrow \qquad r - \frac{\Delta r}{2} \le |\mathbf{r}_i - \mathbf{r}_j| \le r + \frac{\Delta r}{2} \tag{8.2}$$

8.2　実行方法

spcf はコマンド・ベースのプログラムであり、実行の際に、spcf の計算動作を指定するコマンドファイルと、解析の対象である濃度場データファイルを必要とする。あらかじめ、これらのファイルが用意されているとして、実行方法は、コマンド・プロンプトで次の様にする。

```
% spcf.exe ctrl.cmd
```

例ではプログラム本体、コマンドファイルの名称は、それぞれ spcf.exe 、 *ctrl.cmd* としている。なお、OCTA システムのメディアでは、`SUSHI3/spcf/bin` 以下に、WindowsNT/2000 (cygwin) 用と Linux(x86) 用の実行プログラム本体が収録されている。

([注] spcf は、単独で動作するプログラムであるが、Cygwin 用のバイナリは、実行時にダイナミックリンク・ライブライリ (cygwin1.dll) を必要とするので、バイナリと同じ場所、または環境変数 `PATH` の設定している場所に cygwin1.dll をコピーしておく必要がある。)

コマンドファイルの内容は、各行について、コロンで終わるキーワードとそれに対応する値を、空白 (半角スペースかタブ) で区切って並べたものである。なお、キーワードには、以下のものがある。

'rmax:', 'rstep:', 'spherical:', 'dirvec:', 'clip:', 'shift:', 'pairlist:', 'infile:', 'outfile:', 'outfile-format:'

また、値としては、対応する数値または文字列をとる。

次の例で書式を示した。

```
# input data for 'spcf.exe'
# The lines starting with '#'-character are the comment lines,
# and are neglected.
#  Blank lines (such as the next line) are also neglected.

# The upper bound of the distance up to which the calculation is performed.
# [format] rmax: (numeric)
rmax: 16.0

# The mesh width for the distance
# [format] rstep: (numeric)
rstep: 1.0


# In case the keyword 'spherical:' is false,
# the distance is defined as the size of the projection of the relative
# position vector to the direction specified by the keyword
# dirvec:
# spherical: false
# dirvec:  1.0 1.0 1.0


# The upper bound and the lower bound of the field data.
# The field values outside of this range are replaced by the upper bound value
# or the lower bound value.
# [format] clip: (lower-limit) (upper-limit)
clip:  0.0  1.0

# The values by which the field data for each component are shifted.
# The number of the values should be equal to the
# number of the components. Each of the values is added to the field data
# of each component.
# [format] shift: (numeric) (numeric) ...
shift: -0.2  -0.8


# The type of the component pair for which the correlation function is
# calculated.  "self", "distinct" and "all" mean the self-correletion,
# the correleation function between distinct species, and the correletion
# function for all possible pairs of the components, respectively.
# [format] pairlist: self | distinct | all
pairlist: self

# The name of the file that stores the input data of the scalar field.
# [format] infile:  filename
infile: Susi_Phi.dat

# The name of the file to which the output data are written
# and the data format of the outfut file.
# The type of the data format is one of GNUPLOT-style(gnuplot),
# tab separated style(tabtext), or comma separated style (csvtext).
# In the case of GNUPLOT-style, the command file for GNUPLOT is also
# created. The command file is named as XXX.gnuplot.
# [format] outfile: filename
# [format] outfile-format:  gnuplot | tabtext | csvtext
outfile:  plot.dat
outfile-format: gnuplot
```

また、濃度場データファイルは、各行に (x,y,z) の座標値と、その点での各成分の値を列挙したものである。次の例で、書式を示した。

```
#  scalar field data
# The lines starting with '#'-character are comment lines, and are neglected.
#
#  X-value   Y-value   Z-value    phi-0      phi-1
   0.0       0.0       0.0        0.199685   0.800365
   0.0       0.0       0.5        0.200342   0.801061
   0.0       0.0       1.0        0.201941   0.798672
   0.0       0.0       1.5        0.20148    0.801061
   0.0       0.0       2.0        0.20088    0.801031
# In this example, the number of the fileds is 2.
# As long as the data are described in a single line,
# any number of the fields are allowed.
```

計算結果は、X-Y プロット図のデータが、コマンドファイルで指定した書式のファイルとして得られる。なお、SUSHI による計算結果から濃度場データファイルを簡便に作成するために、 `SUSHI3/spcf/python` 内の補助 Python スクリプト `SUSHI2spcf.py` を使用する事ができる。使用手順を次に示した。

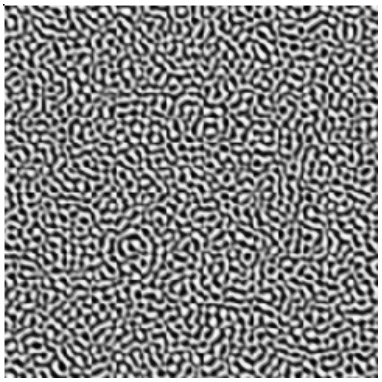
1. GOURMET を起動し、`Browser` または `Editor` で、計算の対象とする SUSHI の出力ファイルを表示する。
2. View を Table に、Location を Record にセットし、スライダーを操作して、目的とする Record に変更する。
3. Python パネルの **Load** を押して、`SUSHI2spcf.py` を取り込む。
4. Python パネルの **Run** を押すと、濃度場データファイルが作成される。作成される場所 (ディレクトリ名)、ファイル名を変更するには、`SUSHI2spcf.py` の 180 行目の destdir、177 行目の datafile の値を変更すればよい。

8.3 サンプルデータ

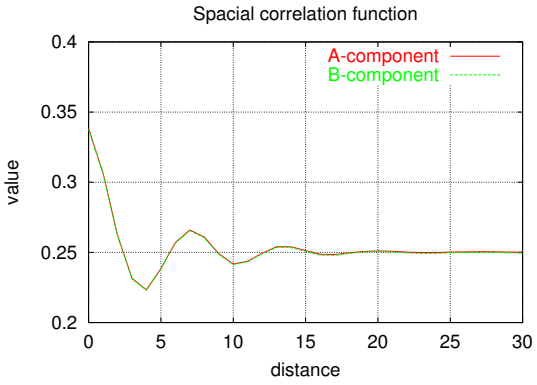
使用事例を示すため、サンプルデータを `SUSHI3/spcf/sample` 以下に収録している。サンプルの内容を、以降に示した。

8.3.1 A/B ポリマーブレンド

高分子 A と高分子 B の混合系 (体積分率比 0.5:0.5, 高分子 A の鎖長：10, 高分子 B の鎖長：10, 相互作用パラメータ χ_{AB}：0.5) について、SUSHI により 2 次元モデル・シミュレーション (メッシュサイズ 256×256) を行い、得られたセグメント濃度分布を図 8.1(a) に、対応する空間相関関数を図 8.1(b) に示した。図 8.1(a) では、高分子 A について分率 0.0 を黒、分率 1.0 を白とした濃淡で示している。なお、用いたサンプルファイルは、`A-B-blend.cmd`、`A-B-blend.dat`、`A-B-blend.plot` である。



(a) セグメント濃度分布



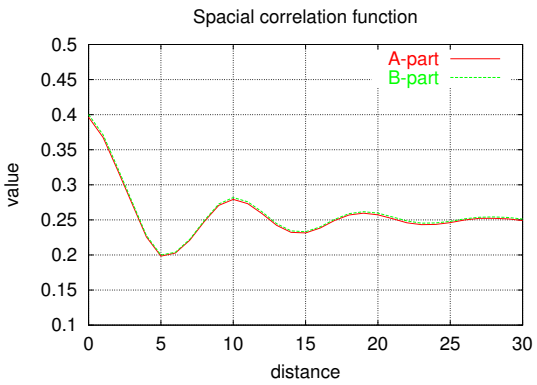
(b) 空間相関関数

図 8.1: A/B ポリマーブレンドの結果

8.3.2　A-B ブロックポリマー

A-B 対称ブロックポリマー (セグメント A の鎖長 : 20、セグメント B の鎖長 : 20、相互作用パラメータ χ_{AB} : 0.5) の系について、SUSHI により 2 次元モデル・シミュレーション (メッシュサイズ 256×256) を行い、得られたセグメント濃度分布を図 8.2(a) に、空間相関関数を、図 8.2(b) に示した。図 8.2(a) では、高分子 A について分率 0.0 を黒、分率 1.0 を白とした濃淡で示している。この例で使用したファイルは、`A-B-block.cmd`、`A-B-block.dat`、`A-B-block.plot` である。

(a) セグメント濃度分布



(b) 空間相関関数

図 8.2: A-B ブロックポリマーの結果

8.3.3　方向ベクトルの効果

一般には、相関関数の計算に用いる「距離」は、あらゆる方向 (球状) について計算する。但し、特定方向についての構造が存在する場合、「距離」を、その方向についての射影にとると、構造が明瞭になる場合がある。

例えば、図ẽffigure:8.3 に示したデータでは、(1,1,0) 方向の周期性が存在する。

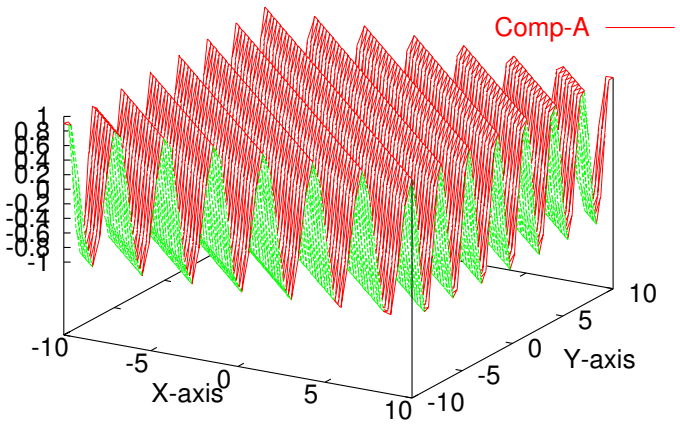
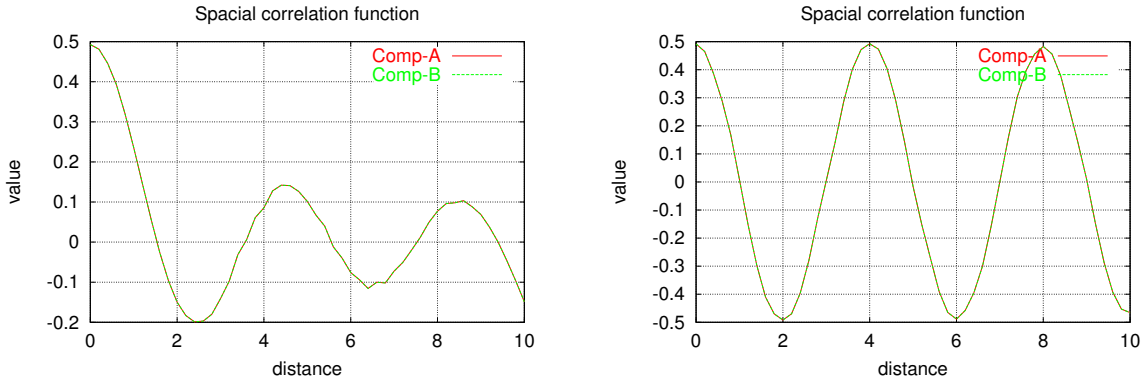


図 8.3: (1,1,0) 方向の周期性のあるデータ：$z_A = cos((x+y) \times \frac{\pi}{\sqrt{2}})$, $z_B = -z_A$

spcf では、ある特定方向についての射影を「距離」として定義して、このような異方的な系についての相関関数を計算することができる。このような場合、キーワード **spherical:** を `false` に設定し、その方向ベクトルをキーワード **dirvec:** を用いて指定する。事例として、図 8.3 に示したデータについて、通常の相関関数と特定方向の相関関数を計算した結果を、それぞれ、図 8.4(a) と図 8.4(b) に示した。

なお、使用したサンプルファイルは、`dirvec.cmd`、`dirvec.dat`、`dirvec.plot` である。



(a) 通常の空間相関関数

(b) (1,1,0) 方向の空間相関関数

図 8.4: 通常の相関関数 (a) と特定方向の相関関数 (b)

付 録A　ナビゲーションタイプのUDF入力フォーマット

SUSHIInput.udf は SUSHI のデータ構造のために作成されたフォーマットである。プログラム用の構造化されたデータであるために、一概にユーザーに理解しやすいフォーマットであるとは言い難い。そこで、ユーザーの負担を低減する目的で Select 機能を多用した新たなフォーマット SUSHIInputV2.udf を用意した。このフォーマットは、入力が必要なデータ項目が Select 機能によりナビゲーションされるように設定したものである。入力後 SUSHI 内で SUSHIInput.udf フォーマットに変更され利用される。主なデータ構造は次のとおりである。

```
SUSHIInputV2:{
    System:{
      name:KEY
      mesh:select { "REGULAR", "RECTANGULAR", "CYLINDRICAL", "SPHERICAL" }
    }
    Monomers[]:Monomer
    Components:{
        polymers[]:Polymer
        solvents[]:Solvent
    }
    Chi_parameters[]:ChiParameter
    Ensemble: {
        type:select { "CANONICAL", "GRANDCANONICAL" }
    }
    Properties:{
        segment_volume_fraction_conditions[]:SegmentVolumeFractionCondition
        radius_of_gyration_conditions[]:SubchainUnit
    }
    External_conditions:{
        surface_chi_parameters[]:SurfaceChiParameter
        graft_conditions[]:GraftCondition
        mask_conditions[]:MaskCondition
    }
    Solver: {
     type:select { "ADF", "FH", "SCF" }
    }
    Run: {
        type:select { "START", "CONTINUE", "RESTART", "RESTART_READMESH" }
    }
```

```
}
```

要素をなす小さなデータ構造の内容は SUSHIInput.udf 内のものと同じである。入力の手順として以下のデータ構造を順次入力してゆく。

Sysytem, Components, Chi_parameter, Ensemble, Properties, External_condition, Solver, and Run.

このようにした理由として、例えば Regular メッシュは 3 次元までの入力が可能であるが、Spherical メッシュは 1 次元の入力しか許されない。ユーザーはこのような規制において間違えを犯さないよう、ナビゲーションされるようにデータ構造が工夫されている。Select 機能の詳細は次のようになる。

```
System:
  mesh:select
    +-REGULAR:
        dimension:select
          +-D1 // one-dimensional
          +-D2 // two-dimensional
          +-D3 // three-dimensional
    +-RECTANGULAR:
        dimension:select
          +-D1
          +-D2
          +-D3
    +-CYLINDRICAL:
        dimension:select
          +-D2
    +-SPHERICAL:
        dimension:select
          +-D1
Ensemble:
  type:select
    +-CANONICAL:
        calculation_method:select
          +-STATICS
              external_conditions_of_statics:ExternalConditionsOfStatics
          +-DYNAMICS
              dynamics_parameter:DynamicsParameter
              external_conditions_of_dynamics:ExternalConditionsOfDynamics
          +-MONTECARLO
              monte_carlo_parameter:MonteCarloParameter
              external_conditions_of_monte_carlo:ExternalMonteCarlo
    +-GRANDCANONICAL:
        calculation_method:select
          +-STATICS
              external_conditions_of_statics:ExternalConditionsOfStatics
Solver:
  type:select
```

```
    +-ADF
    +-FH
    +-SCF
Run:
  type:select
    +-START    // start new calculation
    +-CONTINUE
    +-RESTART
    +-RESTART_READMESH
```

もし、参考にする SUSHIInput.udf が存在せず。何も無い状態から入力ファイルを作る場合、このフォーマットを利用する方が便利である。

付 録B　コンパイル方法

B.1　ソースファイルのディレクトリ構造

SUSHI10.54 以下の主要なディレクトリ構造は次のとおりである。

```
SUSHI10.54--+--SurfaceSimulator
            |
            +--Susi--+--def_udf--+--SUSHIInput.udf   入力 UDF 定義
                     |           +--SUSHIOutput.udf 出力 UDF 定義
                     |
                     +--bin      実行モジュール
                     |
                     +--include  インクルードファイル
                     |
                     +--src      ソースファイル
```

B.2　コンパイル方法

OS が UNIX 系（Linux、Windowns 上の cygwin も含む）の場合は GNU の make(gmake) を使う。GNU の make 以外の make では動作は保証しない。多分コンパイルできない。GNU の make であるかの確認方法は、make -v コマンドを使用する。

```
> make -v
> GNU Make version ....
> ......
```

のように出力されるなら GNU の make である。

コンパイルは SUSHI10.54 ディレクトリーで make をする。

```
> cd SUSHI10.54
> make all
```

とする。各モジュールの

`bin/[OS 名]`

のディレクトリー下に実行モジュールが作成される。ただし、1 コア版の実行モジュールのみが make される。

B.2.1　SUSHI のコンパイル方法

もし、SUSHI のみをコンパイルするなら、SUSHI のソースディレクトリー上で make する。

```
> cd SUSHI10.54/Susi/src
> make all
```

make all で最初に SUSHI に関する全てのライブラリーを Susi/lib 以下に作成する。次に作成したライブラリーを利用して SUSHI と InterfaceSimulator をコンパイルする。OS の選択は make が自動的に行う。もし SUSHI の Makefile がコンパイル環境（OS）に合致していない場合新たに Makefile をつくることができる。それには、以下のコマンドを実行する。

```
> cd Susi/src
> sh ./mkmk.sh
```

これらのコマンドで新しく Makefile が作成される。
RPA 計算を高速に行うためにはFFTW ライブラリーを利用したコンパイルをする。オプションは FFTW=ON を利用する。
バージョンを実行モジュール名の後ろに付加するには

```
> make all VERSION=version_number_etc
```

のように version_number_etc(文字列) のオプションを利用する。

B.2.2 並列化版 SUSHI のコンパイル方法

Pthread 版

共有メモリー型の並列化 SUSHI として Pthread を利用した実行モジュールを make するには

```
> cd SUSHI10.54/Susi/src
> make all PTL=ON
```

のように PTL=ON のオプションをつけてコンパイルする。

GPU 版

GPU 利用型の並列化 SUSHI として CUDA を利用した実行モジュールを make するには

```
> cd SUSHI10.54/Susi/src
> make all GPU=ON COMPUTECAPABILITY=60
```

のように GPU=ON のオプションをつけてコンパイルする。但し、予め CUDA がインストールされている必要がある。また、CUDA がリンクされるためには Makefile に記述されている CUDA ライブラリーに関する部分を書き換える必要がある。COMPUTECAPABILITY は搭載されているデバイス用に変更する。デフォルトは 60 である。

MPI 版

分散メモリー/CPU 利用型の並列化 SUSHI として MPI を利用した実行モジュールを make するには

```
> cd SUSHI10.54/Susi/src
> make all MPI=ON
```

のように MPI=ON のオプションをつけてコンパイルする。但し、予め MPI がインストールされている必要がある。また、MPI がリンクされるためには Makefile に記述されている MPI ライブラリーに関する部分を書き換える必要がある。

K-computer の場合は　MACHIN=K　のオプションを利用する。
MPI と GPU、MPI と PTL の併用ができる。バージョンとしては
MPU=MPI + GPU MPR=MPI + PTL
として記述する。

全バージョンを make する方法

1 コア版を含む全ての種類の SUSHI を make するには SUSHI10.54 ディレクトリーで make する。

```
> cd SUSHI10.54
> make allsushi [VERSION=version_number_etc]
```

B.3　ビルド方法

VC++の場合は Susi ディレクトリー上の*.sln を選択し、Microsoft Visual Studio を立ち上げてからプロジェクト susi をビルドする。各並列化版をビルドするには、
Susi/include/definitions.h
に記述してある。コメントアウトされているマクロ変数

```
//#define MPIUSE
//#define CUDAUSE
//#define PTHREADUSE
```

のどれかを有効にする。尚、UNIX 系の場合、これらが全てコメントアウトされていないと正常に make できない。

VC++ のための環境

各ライブラリーのパスは環境変数により以下のように設定されなくてはいけない。

- libplatrom:PF_FILES(/include or /lib)
- MPI:MPI_INC_PATH, MPI_LIB_PATH
- CUDA:CUDA_PATH(/include, /lib)
- Pthread:PTHREAD_INC_PATH, PTHREAD_LIB_PATH
- FFTW:FFTW_INC_PATH, FFTW_LIB_PATH

もし、ライブラリが用意されておらず、かつ不要なら、リンクされるライブラリを削除して build する。
もしくは、SUSHI10.54 ディレクトリー上で DOS 窓を開き

```
> .\buildall2017
```

と入力すると VC++2017 を利用した全ての実行モジュールの build ができる。

B.4 インストール方法

UNIX 系の OS の場合 SUSHI10.54 ディレクトリー上で

```
> make install
```

する。全ての実行モジュールと関連ファイルが PF_ENGINE/bin/[OS 名] のディレクトリーや関連ファイルのディレクトリーにコピーされる。

Windows の場合はマニュアルで Susi/bin や InterfaceSimulator/bin 以下にある実行モジュールをコピーする。

B.5 クリア方法

UNIX 系の OS の場合 SUSHI10.54 ディレクトリー上で

```
> make clean
```

によりオブジェクトファイルとライブラリーの削除が行われる。

付 録C　システム拡張方法

この章では、SUSHI を初めて扱うユーザの導入時の障壁を低減することを目的として、第 3 章と同様に、システム拡張方法をできるだけ簡易な表現方法を用いて説明することに努めた。

C.1　簡単に書ける静的平均場法プログラム

SUSHI は複雑怪奇なプログラムではありません。機能的な部品を効率よく組み上げて作成されています。基本をなすのは SCFEngine です。このシミュレータの使い方が理解できれば、そのコンパクト性を生かして、SUSHI で行っている SCF 計算を簡単に他のプログラムへ移植することが可能です。その参考になると思いますので、必要最低限のコーディングで 1 次元の A/B ポリマーブレンドの静的平衡計算を行い、結果を出力してみましょう。そのプログラムは次のようになります。

```
#include "SCFEngine.h"    //  シミュレータ本体です。
#include "FieldNoise.h"   //  場に与えるノイズです。

int main() {

    try {

        double sigma      = 0.0001; //  場のノイズに与える標準偏差です。
        int    randomSeed = 12345;  //  ノイズ発生用の乱数の種です。
        int    numPolymer = 2;      //  ポリマーの数です。

        Monomer monomerA( "A", 1., 1. );  //  モノマー A,B をつくります。
        Monomer monomerB( "B", 1., 1. );

        Polymer homoPolymerA( monomerA, 20 );  //  モノマーから長さ 20 のポリマー
        Polymer homoPolymerB( monomerB, 20 );  //  A,B をつくります。

        //  構造的な周期境界条件（周期性の有無）をつくり、規則メッシュをつくります。
        //  メッシュの名前は  test、1 次元で最小値が０、最大値が３２、分割数３２です。
        GeometricalBoundaryCondition geometricalBoundaryCondition;
        RegularMesh mesh( "test", geometricalBoundaryCondition, 0., 32., 32 );

        //  物理的な境界条件（反射か固体壁）をつくります。
        //  デフォルトでは何も設定されません。
        PhysicalBoundaryCondition physicalBoundaryCondition;

        ChiParameter chiParameter;          //  χパラメータを 0.2 にします。
```

```
        chiParameter.set("A", "B", 0.2 );  //  χ N は 4 になります。

        vector<Polymer*> polymers;             //  ポリマーの入れ物です。
        polymers.push_back( &homoPolymerA );  //  つくっておいたポリマーをいれます。
        polymers.push_back( &homoPolymerB );

        vector<double> polymerVolumeFraction;    //  体積分率の入れ物です。
        polymerVolumeFraction.push_back( 0.5 );  //  値をいれます。
        polymerVolumeFraction.push_back( 0.5 );

        vector< ScalarField* > phi, fV;    //  φと V の入れ物です。以下で実態を入れます。
        for( int i = 0; i < numPolymer; i++ ) {
            phi.push_back( new ScalarField( mesh, 0.5 ) );  //  初期値 0.5 です。
            fV .push_back( new ScalarField( mesh, 0. ) );   //  初期値 0.0 です。
        }

        *fV[ 0 ] += FieldNoise( mesh, sigma, randomSeed );  //  V にノイズを加えます。

        SCFEngine engine(  polymers                          //  シミュレータをつくります。
                         , polymerVolumeFraction
                         , chiParameter
                         , mesh
                         , physicalBoundaryCondition
                        );

        cout << "nSCF " << engine.getEquilibrium( phi, fV ) << endl;  //  計算の実行です。

        for( i = 0; i < numPolymer; i++ ) {
            cout << "phi " << i << " " << *phi[i] << endl;  //  φを出力します。1 行につき
                                                            //  5 個の値を出力します。
        }

        deleteFields( phi );  //  後始末です。
        deleteFields( fV );

    }

    catch( MFException x) {
        x.display(cerr);  //  エラーによる例外処理があったら捕まえてくれます。
    }

    return 0;  //  おしまい。
}
```

こんなプログラムでもきちんと動きます。計算の制御は getEquilibrium の引数に class SCFChar のインスタ

ンスを加えることで行えます。デフォルト状態でも簡単な問題なら収束します。結果は次のようになります。最後に ϕ の生データが出力されています。きちんと相分離しています。

ログが出力されますが省略します。

```
nSCF 388
phi 0
 0.022021319 0.023472855 0.02768036 0.039853661 0.076080462
 0.18612184 0.45549125 0.76249782 0.90618754 0.95434671
 0.97035272 0.97586031 0.9777627 0.97839868 0.97856398
 0.97846946 0.97799456 0.97654264 0.97232646 0.96013022
 0.92385038 0.81371286 0.54438197 0.23758755 0.093894333
 0.045705149 0.029682295 0.024167216 0.022258887 0.021616024
 0.021445561 0.021542225

phi 1
 0.97798162 0.97653571 0.97225781 0.96010011 0.92395215
 0.81397296 0.54452043 0.23743126 0.09379216 0.0456568
 0.02965645 0.024150693 0.022248863 0.021613048 0.021447831
 0.021542434 0.022017497 0.023469804 0.027687145 0.039888208
 0.076186746 0.18636283 0.45562929 0.76235723 0.90609141
 0.95424992 0.9702444 0.97582681 0.97774844 0.97837143
 0.97855848 0.97845004
```

複雑怪奇で使えないプログラムではないことがお分かり頂けたと思います。

C.2 簡単に書ける動的平均場法プログラム

SUSHI は動的平均場法のシミュレータとなっておりますので、何も SCF 法による計算しかできないような設計になっている訳ではありません。SCF 法のコアプログラムである SCFEngine は MeanFieldEngine から派生させております。MeanFieldEngine はバーチャルなクラスで、入力されたポリマーに関する情報を簡単にとることができるインターフェースを多数取り揃えております。他タイプの平均場法のシミュレータも MeanFieldEngine から派生することにより簡単に書くことができます。それでは試しにポリマーの A/B ブレンドの相分離をシミュレーションできる Cahn-Hilliard タイプの動的平均場シミュレータをつくってみましょう。基礎式は次のようなものです。

$$\frac{\partial}{\partial t}\phi_K(\mathbf{r},t) = \nabla[L_K(\mathbf{r},t)\nabla\{\mu_K(\mathbf{r},t) - \kappa\nabla^2\phi_K(\mathbf{r},t)\}] \tag{C.1}$$

ここで、κ は界面に蓄積されるエネルギーに関係する正の係数です。必要になる自由エネルギーは Flory-Huggins 自由エネルギー (2.35) 式を多成分系に拡張した次の式を使います。

$$\frac{\mathcal{F}}{k_\mathrm{B}T} = \sum_K \frac{\phi_K}{N_K}\ln\phi_K + \sum_K\sum_{K'>K}\chi_{KK'}\phi_K\phi_{K'} + \sum_K\frac{\kappa}{2}|\nabla\phi_K|^2, \tag{C.2}$$

以降に説明するプログラムは、既に SUSHI に組み込まれております。詳しくはソースプログラムを参考にしてください。さて、この新しいクラスを `FHEngine` と名付けましょう。その `include` 文は次のようなものです。

```
#ifndef _FHENGINE_H_
#define _FHENGINE_H_

#include "MeanFieldEngine.h"
#include "FHChar.h"

class FHEngine: public MeanFieldEngine  {
public:
  FHEngine( .....    //  コンストラクター
    .....
          );
  int getEquilibrium
        (  vector<ScalarField*>& phi, vector<ScalarField*>& fV
         , vector<int> isConstrained = vector<int>(0)
         , MeanFieldChar* pMeanFieldChar = NULL
        ) ;
  int getChemicalPotential
        (  const vector<ScalarField*>& phi, vector<ScalarField*>& fV
         , MeanFieldChar* pMeanFieldChar = NULL
        );
};

#endif //_FHENGINE_H_
```

最初にコンストラクターがなくてはなりませんが、MeanFieldEngine から派生させたプログラムなので、その中身は空でかまいません。バーチャル宣言されている２つの関数 getEquilibrium と getChemicalPotential を実装しなくてはなりませんが、動力学しか扱わないので getEquilibrium の中身も空にしておきます。最後に残った getChemicalPotential は化学ポテンシャルを Flory-Huggins 自由エネルギーから計算するものです。詳細は省きますが、その実装は次のようなものです。ϕ や自己無撞着場 V は class ScalarField になっています。ScalarField は４則演算の機能や、ln(),exp() のような関数を備えています。∇^2 が*p_freePropagator です。ポリマーの配列 d_polymer[] からは長さを d_polymer[i]->numMonomer() のように聞くことが可能です。組成については p_composition に聞くことが可能です。

```
int FHEngine::getChemicalPotential
    (  const vector<ScalarField*>& phi //  phi ( segment volume fraction )
     , vector<ScalarField*>& fV        //  chemical potential
     , MeanFieldChar* pMeanFieldChar   //  control parameters for the calculation
    )
{
    FHChar& rFHChar = *( (FHChar*) pMeanFieldChar ); // casting MeanFieldChar onto FHChar.
    int n = p_composition->numSCFUnit();
    int speciesI, speciesJ;
    *fV[ n - 1 ] = 0.;
    double polymerLength = d_polymer[ n - 1 ]->numMonomer();
```

```
    ScalarField engN = ( phi[ n - 1 ]->log() ) / polymerLength + 1. / polymerLength;
    for( int i = 0; i < n - 1; i++ ) {
        polymerLength = d_polymer[ i ]->numMonomer();
        // enthalopy part: ln( phi ) / N + 1 / N
        *fV[ i ] =  ( phi[ i ]->log() ) / polymerLength + 1. / polymerLength - engN;
        speciesI = p_composition->stateId( p_composition->SCFUnit( i ) );
        // enthalpy part: chi * phi
        for( int j = 0; j < n; j++ ) {
            speciesJ = p_composition->stateId( p_composition->SCFUnit( j ) );
            *fV[ i ] += *phi[ j ] * p_chi[ speciesI ][ speciesJ ];
        }
        *fV[ i ]     -= *phi[ i ] * p_chi[ speciesI ][ speciesJ ];
        *fV[ n - 1 ] += *fV[ i ];
        *fV[ i ] -= ( *p_freePropagator * *phi[ i ] ) * ( 2. * rFHChar.d_kappas[ i ] );
    }
    *fV[ n - 1 ] *= - 1.;
    *fV[ n - 1 ] -=   ( *p_freePropagator * *phi[ n - 1 ] )
                    * ( 2. * rFHChar.d_kappas[ n - 1 ] );
    return 1;
}
```

このようにそれ程長いプログラムにはなりません。強力な class で構築されているメンバーや引数に問い合わせることで、効率よくプログラムを簡単に書くことが可能です。class が持つメソッドについては

`ScalarField, FreePropagator, Polymer, Composition`

のヘッダーファイルをみてください。

制御用のパラメータをもつデータは class FHChar で、これも基本的なデータは既に持つ `class MeanFieldChar` から派生させてあります。その構造は次のように κ が追加されただけです。

```
class FHChar :public MeanFieldChar{
public:
    FHChar(): MeanFieldChar( MeanFieldChar::FH );
    ....
    vector< pair< int, double > > d_kappaData; // pair of the polymer ID and kappa.
};
```

このデータは SUSHI の IO インターフェースを受け持つ class SCFEngineDataIO の public メンバーであり、次のように呼ばれています。

```
class SCFEngineDataIO {
    .....
    FHChar*  p_FHChar;      //  Public member.
    .....
```

```
    MeanFieldChar* getEngineControlChar()  //  SUSHI calls this function.
    FHChar*        newFHChar( classOfInterface& interfaceObject );
                           // This function is called by the above function.
                           // The argument "classOfInterface" means the I/O interface
                           // objects such as the UDF object or the SEED object.
                           // Off course, the user can modify or replace the interface
                           // as he/she likes.
    .....
}

MeanFieldChar* SCFEngineDataIO::getEngineControlChar() {
    switch( d_solver ) { //  This function returns the appropriate controller class
                         //  depending on the solver type "d_solver".
    case MeanFieldChar::ADF :
        return getADFChar();
    case MeanFieldChar::FH :
        return getFHChar();
    case MeanFieldChar::SCF :
    default:
        ;
    }
    return getSCFChar();
}

FHChar* SCFEngineDataIO::newFHChar( classOfInterface& interfaceObject ) {
    //  Please refer the source code for the detail.
    ....
}
```

新たな “SUSHIInput.udf” フォーマットのための C++のソースコードは makeinterface　プログラムのより自動生成することが可能です。詳しくは libplatform マニュアルを参照してください。

最後に EngineGenerator クラスへの登録が必要となります。”EngineGenerator” が必要とするプライベートな関数とデータ構造は次のようなものです。

```
class EngineGenerator {
....
private:
    void setEngine
        (  const vector<Polymer*>& polymer
         , const vector<double>&   polymerVolumeFraction
         , const vector<Solvent*>& solvent
         , const vector<double>&   solventVolumeFraction
         , const ChiParameter& chiParameter
         , const MFBaseMesh& mesh
         , const PhysicalBoundaryCondition& physicalBc
```

```
         , const map<string, MonomerMeanFieldChar* >* pMonomerMeanFieldCharTable
         , const FreePropagator* pFreePropagator
         , const MeanFieldChar* pMeanFieldChar
        );
    ....
    FHEngine*  p_fhEngine;
    ....
};
```

実際の EngineGenerator.setEngine の実装は次のとおりです。

```
    switch( pMeanFieldChar->type() ) {
    ....
    case MeanFieldChar::FH :
        p_fhEngine = new FHEngine
             (  polymer, polymerVolumeFraction
              , solvent, solventVolumeFraction
              , chiParameter, mesh, physicalBc
              , pMonomerMeanFieldCharTable
              , pFreePropagator
              , pMeanFieldChar
             );
        p_engine = p_fhEngine;
        break;
    ....
    }
```

ユーザは化学ポテンシャルの計算部分を記述するだけで動力学計算が可能になります。さて、blend2D_4_FH_dy_uin.udf をサンプルファイルとし、$\kappa = 0.1$ を入力した計算結果は次の図のようになりました。

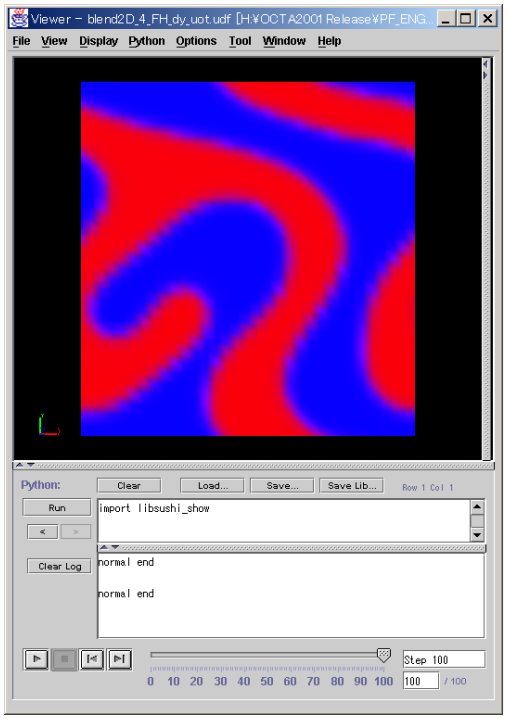


図 C.1: FHEngine 実行結果

付 録D　界面問題を解くためのシステム拡張例（InterfaceSimulator）

界面に特化した問題を効率的に扱うために、先のシステム拡張を発展的に行った例を三つ挙げる。ソースプログラムは InterfaceSimulator ディレクト以下に存在する。シミュレータ（バイナリ）の名前は、

- FluidSimulator（fluid）
- MicelleSimulator（micelle）
- SurfaceSimulator（surface）

である。入力は UDF ファイルによって行えるように実装している。出力は複数のテキストファイルからなる。計算の実行はコマンドラインにより

```
fluid -I fluid.udf
```

などとする。名前が固定された多数の出力ファイルが作成されるため、個々のシミュレーションごとに、別々のディレクトリを用意してやることが望ましい。MicelleSimulator には出力ファイルを解析するための Python スクリプトを添付してある。またサンプルの入力 UDF ファイルが udf ディレクトリ以下に置かれている。以下の節で、それぞれのシミュレータについて簡単に述べる。解説としては甚だ不十分であるが、その内容はサンプル UDF ファイルをみれば自然に理解できるよう設計されている。

D.1　FluidSimulator

FluidSimulator は、多相共存している多成分系における組成変化にともなう、界面構造ならびに界面張力の挙動をシミュレートする。系を構成する成分としては、任意のアーキテクチャをもつ高分子ならびに溶媒を選択することができる。系の組成変化は入力 UDF に各成分の体積分率（カノニカルアンサンブルにおける仕込組成）のリストとして与える。

界面張力は過剰自由エネルギーとして SUSHI で計算することができるが、現実問題への適用を考えた場合、ある一点の組成に関する絶対値はほとんど意味が無い。多くの場合に重要となるのは、組成を変化させたときに界面張力が増加するのか減少するのか、または極大極小が存在するのかどうかといったトレンドである。たとえば、相分離しているホモポリマーブレンドにブロックコポリマーを添加したとき、添加量に応じてどのくらい効率的に界面張力が低下するかといった問題に関心がもたれる。

このような問題への適用のために、FluidSimulator では二つの工夫がなされている。第一は、入力された組成リストにしたがって順に平衡化計算を行う際、前回の計算結果を次の計算の初期条件として用いている点である。これにより、リストにおいてなだらかに組成が変化しているならば、効率的な収束が得られる。第二はカノニカル計算で過剰自由エネルギーを計算するスキームが与えられている点である。見落とされがちな事実であるが、カノニカル計算でマクロ相分離を扱った場合に、相分離しているバルクの組成の体積平均は、一般に仕込みの組成と一致しない。これは、数値計算上の有限サイズ効果に由来する。したがって、仕込み組成は数値計算上のアーティファクトであり、計算後のバルク組成を読み取って、これに対応する過剰自由エネルギーを求めなければならない。複数のバルクが存在する場合の過剰自由エネルギーを求める際、ある一つのバルク

に注目して全系の過剰自由エネルギーを形式的に求めると、相共存している他のバルクからの寄与が 0 になること利用する。これは必ずしも自明でないが、ギブスーデュエムの関係式から一般に導かれる事実である。

FluidSimulator の出力は、リストにおける見かけの仕込み組成に対応した各成分の体積分率プロファイル（phis_xxxx_xxxx.dat）ならびに各バルク組成と界面張力のリスト（energyOfInterface.dat）からなる。

D.2 MicelleSimulator

コポリマーを選択溶媒に溶かすと会合体（ミセル）の溶液が形成される。SUSHI を用いてコポリマーと溶媒からなる系のカノニカル計算を行うと、一応はミセルの形成が観測されるものの、このような系で本質的なミセルの並進自由度の寄与が考慮されないため、臨界ミセル濃度（cmc）といったミセル溶液の物性を正しく扱うのには不十分である。

MicelleSimulator はこのようなミセル溶液を扱うためのシミュレータである。このシミュレータでは、与えられたすべての会合数のミセル構造とその自由エネルギーを計算する。得られたデータを python スクリプトで処理することにより、与えられたポリマー濃度におけるミセル分布を求めることができる。

個々のミセルの計算は、SUSHI のもついくつかの機能を用いて可能となる。まず、会合体を形成するポリマーは、会合数を陽に与えるためにカノニカルアンサンブルとして扱う。SUSHI では体積分率に換算して入力しなければならないが、MicelleSimulator は会合数そのものを入力できる。溶媒にはバルクの体積分率が与えて、グランドカノニカルアンサンブルとして扱う。ただし、孤立ミセルを計算するために溶媒の体積分率の総和は 1 でなければならない。会合体を形成するポリマーは、コアを形成するブロックの末端の存在をある領域に制限される。このために、SUSHI の MASK 機能を用いる。制限する領域があまりに狭いと、鎖のコンフォメーションが限られるために自由エネルギーが高めに見積もられる。しかし、ある程度制限を緩めていくと、自由エネルギーの値はそれに依らなくなる。このため、たった一つのミセルを計算するためにも MASK 領域の大きさを変えた複数のシミュレーションが必要になる。MicelleSimulator は MASK の変化のさせ方と収束誤差の値を入力 UDF で与えることにより、この作業を自動的に行う。さらに、数十、数百にわたる会合数の異なるミセルを一回のシミュレーションで計算することができる。

ミセルシミュレータの主な出力は

```
energyPerMonomer.dat       // 孤立ミセルの自由エネルギー（モノマーあたり）
getMicelle_xxxx_xxxx.dat    // マスクサイズと自由エネルギーの関係
phis_xxxx_xxxx.dat          // 体積分率のプロファイル
plot_phis_xxxx_xxxx.plt  // phis_xxxx_xxxx.dat をプロットするための gnuplot ファイル
```

である。xxxx はミセルの会合数（共ミセルの場合には各成分の会合数）である。

Python スクリプトを用いた解析は次のようにして行う。

energyPerMonomer.dat が存在するディレクトリ上で、

```
python distribution.py <N> <mu>
```

としてやる。これにより、球状ミセルの分布が得られる。<N>は会合体を形成しているポリマーの鎖長（ポリマーが一成分の場合）、<mu>はポリマーの全体積分率を保存させるためのラグランジュ未定乗数である。これにより、"distribution.dat"というファイルが作成される。同様に

```
python analysis.py <N> <mu0> <mu1>
```

とすると、単量体の体積分率、平均会合数、ミセルを形成しているポリマーの全体に対する割合といった量を<mu0>、<mu1>に対応した体積分率の範囲でその関数として出力する。出力ファイル名はそれぞれ、

```
unimerVsOverall.dat
paveVsOverall.dat
micelleFractionVsOverall.dat
```

である。

D.3 SurfaceSimulator

SurfaceSimulator はポリマー溶液が介在する固体表面間の相互作用を界面間距離の関数として計算するためのシミュレータである。固体表面は平板の場合のみを考える。高分子鎖をグラフトさせることはできる。入力 UDF に書かれた表面間距離のリストにしたがって、それぞれの場合の SCF 計算を実行し、各距離での表面過剰自由エネルギーを"energyOfSurfaces.dat"に出力する。他に

```
phis_xxxx.dat          // 体積分率のプロファイル
plot_phis_xxxx.plt  // phis_xxxx.dat をプロットするための gnuplot ファイル
```

が出力される。

References

1) （公社）新化学技術推進協会（編）：”高分子材料シミュレーション―OCTA 活用事例集―”, 化学工業日報社 (2014).

2) 高分子ナノテクノロジーハンドブック: ”～最新ポリマー ABC 技術を中心として～”, NTS Inc. 社 (2014).

3) Helfand, E. and Wasserman, Z. R.: *Macromolecules*, **9**, 879 (1976).

4) Helfand, E. and Wasserman, Z. R.: *Macromolecules*, **9**, 960 (1978).

5) Helfand, E. and Wasserman, Z. R.: *Macromolecules*, **9**, 994 (1980).

6) Hong, K. M. and Noolandi, J.: *Macromolecules*, **14**, 727 (1981).

7) Evers, A., Scheutjens, J. M. H. M., and Fleer, G. J.: *Macromolecules*, **23**, 5221 (1990).

8) Fraaije, J. G. E. M.: *J. Chem. Phys.*, **99**, 9202 (1993).

9) Fleer, G. J., Cohen Stuart, M. A., Scheutjens, J. M. H. M., Cosgrove, T., and Vincent B. : *Polymers at Interfaces* (Chapman & Hall, London, 1993).

10) Kawakatsu, T.: *Statistical Physics of Polymers: An Introduction* (Springer, Berlin, 2004).

11) Honda, T. and Kawakatsu, T.: *Macromolecules*, **39**, 2340 (2006).

12) Honda, T. and Kawakatsu, T.: ”Computer simulations on nano-scale phenomena based on the dynamic density functional theory. Applications of SUSHI in OCTA system” in *Nanostructured Soft Matter: Experiments, Theory and Perspectives*, A. V. Zvelindovsky, ed., Springer-Verlag (2007).

13) Bohbot-Raviv, Y. and Wang, Z.-G.: *Phys. Rev. Lett.*, Vol. 85, p. 3428 (2000).

14) Honda, T. and Kawakatsu, T.: *Macromolecules*, **40**, 1227 (2007).

15) 本田　隆：高分子計算機科学研究会予稿集, p-12, 東京 (2014).

16) Lyakhova, K. S., Zvelindovsky, A. V., and Sevink, G. J. A.: *Macromolecules*, **39**, 3024 (2006).

17) Ly, D. Q., Honda, T., Kawakatsu, T., and Zvelindovsky, A. V.: *Macromolecules*, **40**, 2928 (2007).

18) Ly, D. Q., Honda, T., Kawakatsu, T., and Zvelindovsky, A. V.: *Macromolecules*, **41**, 4501 (2008).

19) Ly, D. Q., Honda, T., Kawakatsu, T., and Zvelindovsky, A. V.: *Soft Matter*, **5**, 4814 (2009).

20) Ly, D. Q., Pinna, M., Honda, T., Kawakatsu, T., and Zvelindovsky, A. V.: *J. Chem. Phys.*, in press (2013).

21) Honda, T. and Kawakatsu, T.: *J. Chem. Phys.*, **129**, 114904 (2008).

22) 土井正男, 小貫明：高分子物理・相転移ダイナミクス, 第５章, 岩波書店 (1992).

23) de Gennes, P. G.: *Scaling Concepts in Polymer Physics*, chapter XIII, (Cornell University Press, Ithaca, 1979).

24) Broseta, D., Fredrickson, G. H., Helfand, E. and Leibler, L.: *Macromolecules*, **23**, 132 (1990).

25) Anastasiadis, S. H., Gancar, I. and Koberstein, J. T.: *Macromolecules*, **21**, 2980 (1988).

26) Helfand, E. and Tagami, Y.: *J. Chem. Phys.*, **62**, 1327 (1975).

27) van Krevelen, D. W.: *Properties of Polymers*, chapters 7 & 8, (Elsevier, Amsterdam, 1990).

28) Brandrup, J., Immergut, E. H. and Grulke, E. A. eds.: *Polymer Handbook (4th Ed.)*, chapter VII, 675, (John Wiley and Sons, New York, 1999).

29) Mark, J. E. ed.: *Physical Properties of Polymers Handbook*, chapter 19, (AIP Press, Woodbury, 1996).

30) 高分子学会（編）：高性能ポリマーアロイ, 第 3 章, 丸善 (1991).

31) Hoftyzer, P. J. and van Krevelen, D. W.: in van Krevelen, D. W. ed., *Properties of Polymers*, chapter 7, pp. 152 – 155, (Elsevier, Amsterdam,1990).

32) Fedors, R. F.: *Polym. Eng. Sci.*, **14**, 147 (1974).

33) Dunkel, M.: *Z. physik. Chem.*, **A138**, 42 (1928).

34) Aoyagi, T., Honda T., and Doi, M.: *J. Chem. Phys.*, **117**, 8153 (2002).