博士論文

液液界面の相間移動触媒反応と熱輸送の分子論

小泉 愛

令和３年

目次

第 1 章　序論 3

第 2 章　計算手法 5

2.1　力場 . . . 5

2.2　計算方法 . . . 6

2.2.1　イオン輸送自由エネルギー計算 . . . 6

2.3　自由エネルギー摂動計算 . . . 9

2.4　溶媒和自由エネルギー . . . 10

2.4.1　水和数分布 . . . 10

第 3 章　イオン輸送促進機構 11

3.1　序 . . . 11

3.1.1　2 次元自由エネルギー $\Delta G_{(z,r)}$ . . . 12

3.2　カチオンの場合の FIT の検証 . . . 16

3.2.1　Li^+-$TPBCl^-$ の場合 . . . 16

3.3　補足 . . . 17

3.3.1　四級アミンの分子モデリング . . . 17

3.3.2　Li^+ のモデル . . . 20

3.3.3　テトラクロロフェニルホウ酸 (TPBCl) の分子モデリング . . . 21

第 4 章　相間移動触媒反応 23

4.1　反応解析 . . . 24

4.1.1　分子モデル . . . 24

4.1.2　イオン輸送自由エネルギー計算 . . . 25

4.1.3　水和数分布 . . . 33

4.2　反応 . . . 37

4.2.1　量子化学計算の精度 . . . 37

4.2.2　遷移状態探索 . . . 38

4.3　反応機構解析 . . . 45

第 5 章　油水共存系のためのジクロロメタンの分子モデリング 46

5.1　バルクの物性 . . . 46

5.2　油水共存系のための分子モデル改良 . . . 49

5.3 水和数分布 55

第 6 章 液液界面での熱輸送 **57**

6.1 界面をまたぐ温度プロファイルの計算の LAMMPS への実装 57

6.1.1 コーディング 60

6.1.2 インプットファイル 97

参考文献 **98**

第 1 章

序論

液液界面では、２種類の異なる溶媒の差からイオン輸送や熱輸送の特性がバルクとは異なることが予想されるが、界面を選択的に観測することは実験的に難しく未解明な現象が多く残されている。これまでの先行研究では混ざり合わない２種類の溶媒からなる液液界面、主に水と疎水性溶媒間の界面について、実験と分子動力学シミュレーション（MD）の両方からイオン輸送機構が調べられてきた。MD では、界面を通過する時に親水性イオンの水和クラスターと水相側が水素結合することによって water finger 構造が形成されることが発見され [1, 2, 3, 4]、その構造形成が界面をまたぐイオン輸送自由エネルギープロファイルを変えることが明らかにされてきた。[5, 6]. 本研究では液体界面に関する以下の４つの研究を MD による解析のもと進めてきた。

1. 油水界面におけるイオン輸送促進機構の微視的解明 (3 章)
2. 相間移動触媒（PTC）反応の微視的反応機構解析 (4 章)
3. 油水共存系のためのジクロロメタン (DCM) 分子モデルの改良 (5 章)
4. 気泡崩壊過程における水/酸素界面の熱輸送の解析 (6 章)

■研究 (1): PTC のイオン輸送過程を Facilitated ion transfer（FIT）と呼ぶ。Laforge らは電場存在下で F-などの輸送が難しい親水性イオンに対して極微量の疎水性対イオンが界面輸送を劇的に促進することを報告し [7]、界面での局所的なイオン対形成が界面移動を触媒する”シャトリング機構”（図 1）を提唱した。本研究では、この“シャトリング”と呼ばれる現象が界面数ナノメートルで起こることを MD による２次元の自由エネルギー解析から解明した。

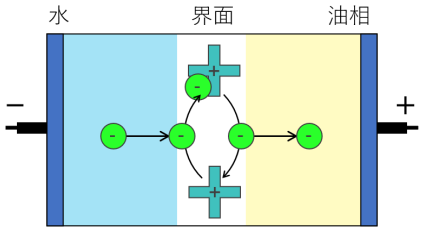
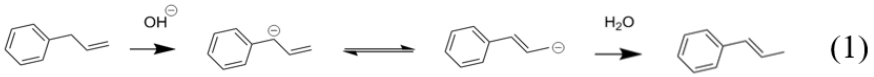


図 1.1　シャトリング機構

■研究 (2): PTC は、混ざり合わない油水二相下で、親水性と親油性の試薬の反応を触媒する。これまでに合成反応への応用は多く、反応速度論から反応機構も議論されてきた。しかし、油水界面をまたぐ物質輸送と界面近傍でおこる化学反応を分子レベルで捉えることは難しい。本研究では、研究 (1) を実際の PTC 反応へ適用し OH- の相間移動で

引き起こされる反応の代表例として、アリルベンゼン異性化反応 1 の解析を行い、反応機構の全貌の解明を目指している。



(1)

図 1.2　アリルベンゼン異性化反応

■研究 (3):　これまで油水界面や油中でのイオンの水和などが MD によって調べられてきた。しかし、水和数に関する先行研究によると、ジクロロエタン中の Cl-の水和数は実験では 2, 3 個であるのに対し MD では 0 個という結果であった。[8, 9] 親水性イオンは油中で不安定で水和による安定化が大きいと予想され、水和しないという結果は不自然であることから MD の結果を精査し分子モデルの改良を行った。本研究で界面系や混合系など油水共存系をよく記述できる分子モデル作成の方針を示したと共に、MD を用いた水和数予測が十分精度よくできることを確認し、研究 (2) での OH-の水和数まで考慮した反応解析に至った。

■研究 (4):　界面では二種類の溶媒の熱伝導率の差が生じ、界面をこえた熱輸送には温度ジャンプが生じるはずである。これは水中の酸素ナノバブル界面系に適用できる。数値シミュレーションの先行研究によると、水中の酸素ナノバブルの自己崩壊では気泡内が過熱され、気泡内の温度が約 3000 K まで上昇すると報告されている。[10] しかしこのモデルでは気泡崩壊の瞬間の熱の流入出とその熱抵抗が記述できていない可能性があった。本研究ではより信頼できる気泡崩壊の描像を得るために、数値シミュレーションへ組み込む熱抵抗値を非平衡分子動力学シミュレーションで見積もった。

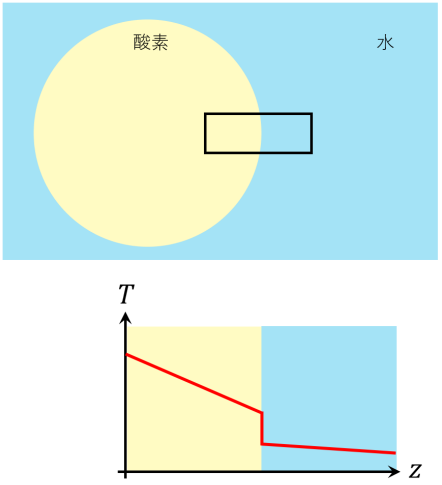


図 1.3　水/酸素気泡界面の温度プロファイル

まず 2 章に研究 (1)〜(4) で共通して用いる計算手法について記述し、3〜6 章に研究 (1)〜(4) の内容をそれぞれ記述する。

第 2 章

計算手法

この章では複数の章で登場する計算手法と MD の計算条件について記述する。

1. 力場
2. イオン輸送自由エネルギー計算 (Liq-Liq, PTC, DCM)
3. 自由エネルギー摂動計算 (PTC, DCM)
4. 溶媒和自由エネルギー (PTC, DCM)
5. 水和数分布計算 (PTC, DCM)

2.1 力場

分子動力学シミュレーション (MD) で採用する力場について記述する。分子間力は Lennard-Jones 相互作用、クーロン相互作用、誘起点双極子のサイト-サイト間相互作用で記述される。分子の構造は固定され（後述の TBA, THA, TPBCl 分子はフレキシブルモデルで、構造は分子内ポテンシャルによって記述される）凝集系の全相互作用エネルギーは次式で記述される。

$$U_{\mathrm{total}} = U_{\mathrm{LJ}} + U_{\mathrm{cl}} + U_{\mathrm{pol}} \tag{2.1}$$

U_{LJ} と U_{cl} は Lennard-Jones とクーロン相互作用で、U_{pol} は誘起双極子 μ_j に関する静電相互作用である。

$$U_{\mathrm{LJ}} = \sum_i \sum_{j(\neq i)} 4\varepsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] \tag{2.2}$$

$$U_{\mathrm{cl}} = \frac{1}{2} \sum_i \sum_{j(\neq i)} q_i T_{ij}^{(0)} q_j \tag{2.3}$$

$$U_{\mathrm{pol}} = -\sum_i \vec{\mu}_i \vec{T}_{ij}^{(1)} q_j - \frac{1}{2} \sum_i \sum_{j(\neq i)} \vec{\mu}_i \vec{T}_{ij}^{(2)} \vec{\mu}_j + \frac{1}{2} \sum_i \frac{|\vec{\mu}_i|^2}{\alpha_i}. \tag{2.4}$$

ここで $r_{ij} = |\vec{r}_i - \vec{r}_j|$ はサイト i,j 間の距離、σ_{ij} と ε_{ij} は Lennard-Jones パラメータ、q_i と α_i はサイト i の点電荷と分極率、$\vec{\mu}_i$ はサイト i の誘起双極子モーメントである。

$T_{ij}^{(n)} = T^{(n)}(\vec{r}_i, \vec{r}_j)$ $(n = 0, 1, 2)$ は相互作用テンソルであり真空の誘電率 ε_0 を用いて次のように表される。

$$T^{(0)}(\vec{r}, \vec{r}') = \frac{1}{4\pi\varepsilon_0}\frac{1}{|\vec{r} - \vec{r}'|}, \tag{2.5}$$

$$\vec{T}^{(1)}(\vec{r}, \vec{r}') = -\frac{1}{4\pi\varepsilon_0}\frac{|\vec{r} - \vec{r}'}{|\vec{r} - \vec{r}'|^3}, \tag{2.6}$$

$$\vec{T}^{(2)}(\vec{r}, \vec{r}') = -\frac{1}{4\pi\varepsilon_0}\left[\frac{\vec{1}}{|\vec{r} - \vec{r}'|} - 3\frac{(\vec{r} - \vec{r}') \otimes (\vec{r} - \vec{r}')}{|\vec{r} - \vec{r}'|^5}\right] \tag{2.7}$$

2.2　計算方法

全ての MD は NVT アンサンブルで Nosè-Hoover サーモスタット [11, 12] によって温度制御された。時間発展は時間刻み $\Delta t = 1fs$ で速度ベルレ法のアルゴリズムで行われ、分子内の配置は RATTLE アルゴリズムによって固定された。[13, 14, 15]. またすべてのＭＤは 3 次元周期境界条件で実施された。長距離相互作用は Particle Mesh Ewald(PME) 法 [16] で扱われ、Ewald のパラメータは $\kappa = 0.323\mathring{A}^{-1}$ で実施された。また PME での双極子相互作用は Toukmaji の PME (SPME) 法で扱った。[17]. MD シミュレーションは研究室で共同開発している FreeFlex を用いて実施された。

2.2.1　イオン輸送自由エネルギー計算

油水界面を通過するイオンの輸送自由エネルギーはただ MD を走らせても輸送へ移動することは稀である。本研究ではレプリカ交換アンブレラサンプリング法 (REUS)[18] でイオンに対してバイアスをかけた計算を実施した。アンブレラサンプリングでは次のようなアンブレラポテンシャルを輸送するイオンにかけてサンプリングを行う。

$$U^{\mathrm{bias}}(z;\ z_0, \sigma_z) = \frac{k_B T}{2}\left\{\frac{(z - z_0)^2}{\sigma_z^2}\right\}. \tag{2.8}$$

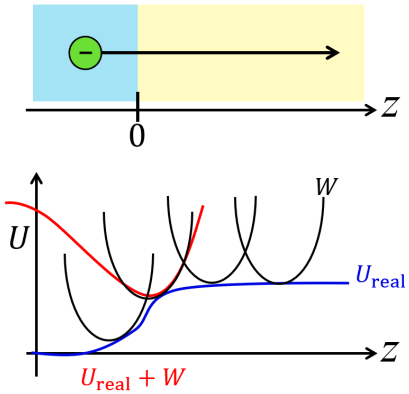


図 2.1　界面をまたぐ系でのアンブレラサンプリング。z は界面と垂直方向の軸である。バイアス W をかけた系では $U_{\mathrm{real}} + W$ の和のポテンシャルとなる。

特に、この界面をまたいでおかれた複数のバイアスを MD の最中で時々交換する方法がレプリカ交換アンブレラサンプリング法 (図 2.2.1) である。レプリカ i にかかるバイアス W_i とレプリカ j にかかるバイアス W_j の交換は次の形

式でメトロポリス判定によって採択される。

$$\frac{\exp\left[-\beta\{W_i(z_j)+W_j(z_i)\}\right]}{\exp\left[-\beta\{W_i(z_i)+W_j(z_j)\}\right]} \geq 1 \tag{2.9}$$

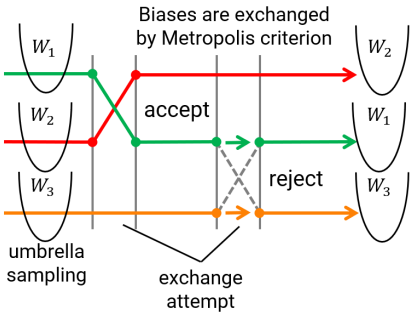


図 2.2　レプリカ交換アンブレラサンプリング法でのバイアスの交換

レプリカ交換アンブレラサンプリングを実施した後に、Weighted Histogram Analysis Method (WHAM) [19] によってバイアスを取り除き U_{real} を得る。バイアス W_i がかけられたレプリカ i でのポテンシャルエネルギーは次式で表される。

$$U_i(z) = U_0(z) + W_i(z) \tag{2.10}$$

この時レプリカ i でのイオンの存在確率は次のようになる。

$$P_i(z) = \frac{\exp\left[U_0(z)+W_i(z)\right]}{Z_i} \tag{2.11}$$

$$Z_i = \int \exp\left[U_0(z)+W_i(z)\right]dz \tag{2.12}$$

Z_i は分配関数である。またバイアスがかかっていない場合の存在確率は次のようになる。

$$P_{0,i}(z) = \frac{\exp\left[U_0(z)\right]}{Z_0} \tag{2.13}$$

$$= \frac{\exp\left[U_0(z)+W_i(z)\right]\int \exp\left[U_0(z)+W_i(z)\right]dz}{\int \exp\left[U_0(z)+W_i(z)\right]dz \int \exp\left[U_0(z)\right]dz} \exp\left[\beta W_i(z)\right] \tag{2.14}$$

$$= P_i(z)\frac{Z_i}{Z_0}\exp\beta W_i \tag{2.15}$$

また重み因子 $w_i(z)$ で線形結合をとると $P_0(z)$ は次のようになる。

$$P_0(z) = \sum_{i=1}^{n} w_i(z) P_i(z) \frac{Z_i}{Z_0} \exp\left(\beta W_i\right) \tag{2.16}$$

$$\sum_{i=1}^{n} w_i(z) = 1 \tag{2.17}$$

ここで分散 σ^2 を最小化するように w_i を決定する。

$$\sigma^2 = \left\langle P_0(z)^2 \right\rangle - \left\langle P_0(z) \right\rangle^2 \tag{2.18}$$

$$= \sum_{i=1}^{n} w_i(z)^2 \exp\left(2\beta W_i\right) \left(\frac{Z_i}{Z_0}\right) \left(< P_i(z)^2 > - < P_i(z) >^2\right) \tag{2.19}$$

$z + \delta z$ に含まれる存在確率 $P_i(z)\delta z$ は次式で書ける。

$$P_i(z)\delta z = \frac{H_i(z)}{M_i} \tag{2.20}$$

よって、式 2.19 は次のように変形できる。

$$\sigma^2 = \sum_{i=1}^{n} w_i(z)^2 \exp 2\beta W_i \left(\frac{Z_i}{Z_0}\right)^2 \frac{< H_i(z)^2 > - < H_i(z) >^2}{(M_i\delta z)^2} \tag{2.21}$$

$M_i \to \infty$ の時、次が成り立つ。

$$< H_i(z)^2 > - < H_i(z) >^2 = M_i P_i(z)\delta z(1 - P_i(z)\delta z) \simeq M_i P_i(z)\delta z \tag{2.22}$$

したがって、分散 σ^2 は

$$\sigma^2 = \sum_{i=1}^{n} w_i(z)^2 \exp\left(2\beta W_i\right) \left(\frac{Z_i}{Z_0}\right)^2 \frac{P_i(z)}{M_i\delta z} \tag{2.23}$$

ラグランジュ未定乗数法を用いて式 2.17 の拘束条件のもと最小化を行う。

$$\frac{\partial}{\partial w_i(z)} \left[\sigma^2 - \lambda \left(\sum_{i=1}^{n} w_i(z) - 1\right)\right] \tag{2.24}$$

式 2.24 を解いて、w_i と λ が得られる。

$$w_i(z) = \frac{\lambda}{2P_{0,i}(z)} \exp\left(\beta W_i(z)\right) M_i\delta z \frac{Z_0}{Z_i} \tag{2.25}$$

$$\lambda = \sum_{i=1}^{n} \frac{2P_{0,i}(z)}{\exp\left(-\beta W_i(z)\right) M_i\delta z \frac{Z_0}{Z_i}} \tag{2.26}$$

これを式 2.16 に代入して、

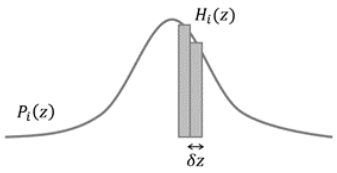
$$P_0(z) = \frac{\sum_{i=1}^{n} H_i(z)}{\sum_{i=1}^{n} \exp\left(-\beta W_i\right) M_i\delta z \frac{Z_0}{Z_i}} \tag{2.27}$$

$$\frac{Z_i}{Z_0} = \frac{\int \exp\left(U + W_i\right) dz}{\int \exp\left(U\right) dz} \tag{2.28}$$

$$= \int P_0(z) \exp\left(-\beta W_i(z)\right) dz \tag{2.29}$$

$P_0(z)$ と $\frac{Z_i}{Z_0}$ は式 2.27 と式 2.29 を self-consistent に解くことで得られる。自由エネルギーは次のように書ける。

$$F_0(z) = -k_{\mathrm{B}} T \ln\left(P_0(z)\right) \tag{2.30}$$


$H_i(z)$
$P_i(z)$
δz

2.3　自由エネルギー摂動計算

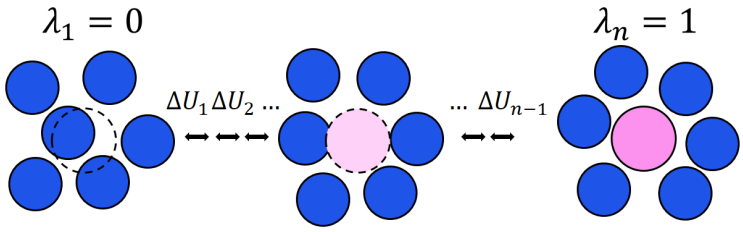


図 2.3　自由エネルギー摂動計算

真空中の溶質分子が溶媒に溶ける時の溶媒和自由エネルギーは自由エネルギー摂動計算 (Free energy Perturbation, FEP) 図 (2.3) で計算できる。FEP では何も相互作用しない真空中から溶媒中に溶けている状態へ λ でスケールした中間状態を作り出し準静的に溶質分子を溶媒中に出現させる。溶質分子が気相から溶媒中へ溶ける時の自由エネルギー ΔU_{solv} は各 λ で修飾された系の状態間でのエネルギー差 ΔU の和になる。

$$\Delta U_{\mathrm{solv}} = \Delta U_1 + ... + \Delta U_m + \Delta U_{m+1} + ... + \Delta U_{n-1} \tag{2.31}$$

隣り合う λ が作り出す系のエネルギーの分布図 (2.3)$P(\Delta U_{\lambda_m})$ と $P(\lambda_{m+1})$ が重なりあう部分が ΔU_m として得られる。

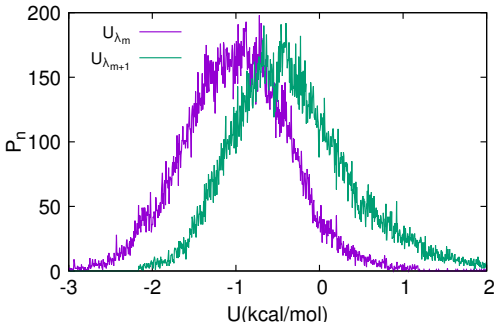


図 2.4　隣り合う λ の作る二つの系のエネルギー分布

$$\Delta U_m = -k_B T \ln \left(\frac{P(U_{\lambda_{m+1}})}{P(U_{\lambda_m})} \right) w(\Delta U_m) \tag{2.32}$$

この時二状態の作るエネルギー帯の中で最も重なっているエネルギーが ΔU_m に寄与するように重み $w(\Delta U_m)$ を付ける。

$$w(\Delta U_m) = \frac{1}{W}\frac{P(\Delta U_m)P(\Delta U_{m+1})}{P(\Delta U_m)+P(\Delta U_{m+1})} \tag{2.33}$$

$$W = \int \Delta U \frac{P(\Delta U_m)P(\Delta U_{m+1})}{P(\Delta U_m)+P(\Delta U_{m+1})} \tag{2.34}$$

各 λ_m で隣接した状態 λ_{m+1} とのエネルギーの重なったところを ΔU_m とし、その総和である ΔU_{solv} を計算する。

$$\Delta U_{\mathrm{solv}} = \sum_{i=1}^{n-1} \Delta U_i \tag{2.35}$$

2.4　溶媒和自由エネルギー

化学ポテンシャルは式 (2.36) と表すことができ、気相、水相、油相で平衡に達した時式 (2.40) が成り立つ。

$$\mu = \mu^* + k_B T ln \rho_{\mathrm{w}} \Lambda \tag{2.36}$$

$$\mu_{\mathrm{in\,gas}} = \mu^*_{\mathrm{in\,gas}} + k_B T \ln(\rho_{\mathrm{w}}\Lambda^3) \tag{2.37}$$
$$\mu_{\mathrm{in\,water}} = \mu^*_{\mathrm{in\,water}} + k_B T \ln(\rho_{\mathrm{w}}\Lambda^3) \tag{2.38}$$
$$\mu_{\mathrm{in\,oil}} = \mu^*_{\mathrm{in\,oil}} + k_B T \ln(\rho_{\mathrm{w}}\Lambda^3) \tag{2.39}$$
$$\mu_{\mathrm{in\,gas}} = \mu_{\mathrm{in\,water}} = \mu_{\mathrm{in\,oil}} \tag{2.40}$$

$$\Lambda = \frac{h}{\sqrt{2\pi m k_B T}} \tag{2.41}$$

Λ は並進のエントロピーを表す項である。気相では溶媒和自由エネルギー μ^* が 0 なので第二項目のみで決まり、気体の状態法的式より飽和蒸気圧 P から密度 ρ が得られるので水の化学ポテンシャルが得られる。したがってこれらの関係から、各相での μ^*_{w} の実験的な値が得られる。

2.4.1　水和数分布

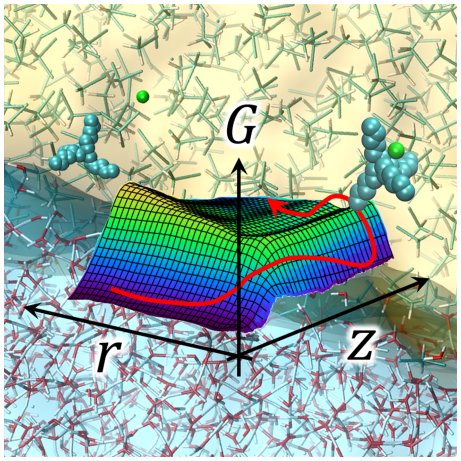
油相に存在する親油性イオンは水和水を保持する傾向が高い。油相でのイオンの水和数分布は水和数 $n-1$ 個から n 個への自由エネルギー変化 $\Delta G(n-1 \to n)$ で計算される。水和数 n 個のイオンの水和クラスターの存在確率は次式で表される。

$$P(n) = \frac{\exp\left[-\frac{G(n)-n\mu}{k_{\mathrm{B}}T}\right]}{\sum_{n'=0}^{\infty}\exp\left[-\frac{G(n')-n'\mu}{k_{\mathrm{B}}T}\right]} \tag{2.42}$$

k_{B} はボルツマン定数 T は温度である。μ は油相での水の化学ポテンシャルである。$G(n)$ と μ^* は MD で計算された。

第 3 章

イオン輸送促進機構



油水界面系では硬い親水性イオンは電場をかけた条件下でも水相から油相へのイオンの移動は難しいが、極微量のカウンターイオンによってイオン輸送が劇的に促進される。”シャトリング”と呼ばれるこの現象は MD シミュレーションによる微視的研究によって解明された。またアニオンとカチオンの両方の FIT を検証し、一般性が確認された。

3.1　序

混ざり合わない二種類の電解質溶液からなる界面でのイオン輸送 (ITIES) は分離・抽出、化学センサー、相間移動触媒 (PTC)、生体膜輸送、薬理学などの多くの系の基礎となる過程である。[20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. イオン輸送はリガンドやイオノフォアによって制御され、この FIT と呼ばれる機構は PTC やイオンセンサーなどに利用されている。これまでに FIT は電気化学測定によって調べられ、界面での微視的な描像が議論されてきた。リガンドやイオノフォアの触媒効果に関する興味深い未解決問題がある。

電場をかけた系での水相から油相へのイオン輸送は、小さく硬いイオン (F^-,Li^+ 等) の場合にそれ単体での輸送は困難だが、極微量のリガンドがイオン輸送を劇的に促進することが報告されている。この現象はシャトリング機構と呼ばれます。[7] これまでに主に電気化学測定に基づいて解析され議論されてきたが、[21, 23, 31, 32, 33, 34, 35, 36, 37, 38, 39] 界面で起こる瞬間のダイナミクスを実験でとらえることは困難だった。したがって、本研究では MD を用いて触媒的なイオン輸送機構を解明した。MD は分子分解能で ITIES を詳細に調べることができる。例えば、MD は water finger と呼ばれるイオンが水相から油相へと移動する時に起こる界面揺らぎ構造を発見した。これは実験的には検出不可能だ

が、イオン輸送において重要な役割を担っていることがわかった。私たちの研究室の先行研究では、water finger を記述する適切な座標を定義して、イオン輸送を遅延させる water finger 形成・切断に関するバリアがあることを明らかにしています。この成功を更に異なる系について検討し、本研究では触媒的なイオン輸送機構の微視的解明につなげた。イオン輸送は単純には界面と垂直な z 座標上での界面からのイオンの位置で記述され、図 3.1 に示すように、$z < 0$ ($z > 0$) ではイオンは水相 (油相) に存在する。今回はさらにリガンドとのイオンペア形成に関するイオンとリガンド間の距離 r を加え z と r に関する自由エネルギー解析を実施した。

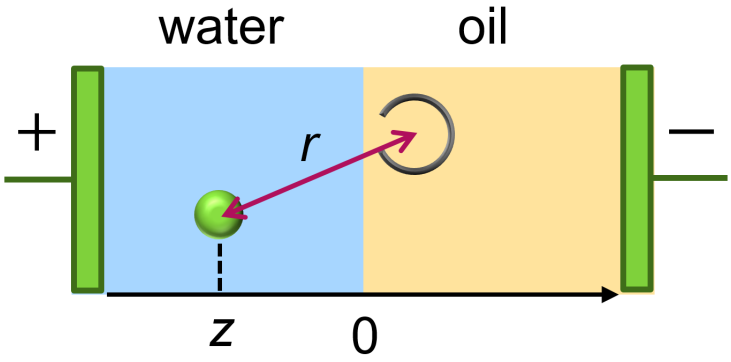


図 3.1　油水界面と二つの座標 (z, r)。緑色は F^- で灰色の一部が開いた輪は (THA^+/TBA^+) を表す。

3.1.1　2 次元自由エネルギー $\Delta G_{(z,r)}$

私たちは水/DCM 界面での F^- の輸送を典型例として調べた。リガンドとして用いたテトラヘキシルアンモニウム (THA, $N(C_6H_{13})_4{}^+$) は F^- の輸送に対して大きな触媒効果を示すことが報告されている。[7] またほとんど同じ構造のテトラブチルアンモニウム (TBAN$(C_4H_9)_4{}^+$) では触媒効果が観測されていない。今回はこの THA と TBA の 2 つのリガンドに注目した。温度は 300K で、z 軸方向に 0〜0.2V/nm の外部電場 E_z を印加して、シミュレーションセルは $L_x \times L_y \times L_z = 50$ Å $\times$ 50 Å $\times$ 115 Å で、2091 個の水と 2116 個の DCM から成るスラブ系で MD を実施して F^- とカウンターイオン (THA, TBA, モデルは付録 3.3.1 に記載。) のイオン輸送に対する二次元自由エネルギー面 $G^{(2)}(z, r)$ を計算した。$G^{(2)}(z, r)$ は、二次元表面上を REUS(2 章 2.2.1 に記載) を用いた並列計算で行われた。水、DCM、イオンの分子モデルは分極性分子モデルを使用した。またイオンペア形成の影響を評価するために、F^- イオン単独の移動の場合の一次元表面 $G^{(1)}(z)$ も計算した。外部電場 E_z はボルタンメトリーの実験条件を模倣したもので、界面における実際の電場は、印加された電位や支持電解質の分布など、多くの要因に支配されるため現実的な条件を見積もることは難しい。しかし、0〜1V の電位とナノメートルオーダーの二重層の厚さを考慮すると、0.1V/nm の電界を仮定することは妥当である。ここでは、イオン移動メカニズムに対する外場の影響を調べるには、この電場の仮定で十分である。

図 3.1.1 は F^--THA^+ の 2 次元自由エネルギー面 $G^{(2)}(z, r)$ である。

電場無しのパネル (a)) では、F^- が油相に存在する時リガンドとイオンペア形成している ($z > 0$ かつ $r \sim 4$ Å) を除いた ($z > 0$ and $r >> 0$) の範囲では水相 ($z < 0$) と比較して大きく不安定である。一方で電場を印加したパネル (b) では、リガンドとイオンペア形成した経路 (A)($r \sim 4$ Å) に加えてイオンペア形成していない (B)($r >> 0$) の新しい経路が見えている。しかし、($z \to \infty$) での二つの経路の振る舞いは明らかに異なる。経路 (A) の ($z >> 0$) では、F–THA^+ がイオンペア形成しており電荷が中性で平らな自由エネルギー形状であるが、経路 (B) では F^- が z が正の方向 ($z \to \infty$) に向かって、THA^+ が逆方向に向かって自由エネルギーが下がっていく。したがって電荷を印加した場

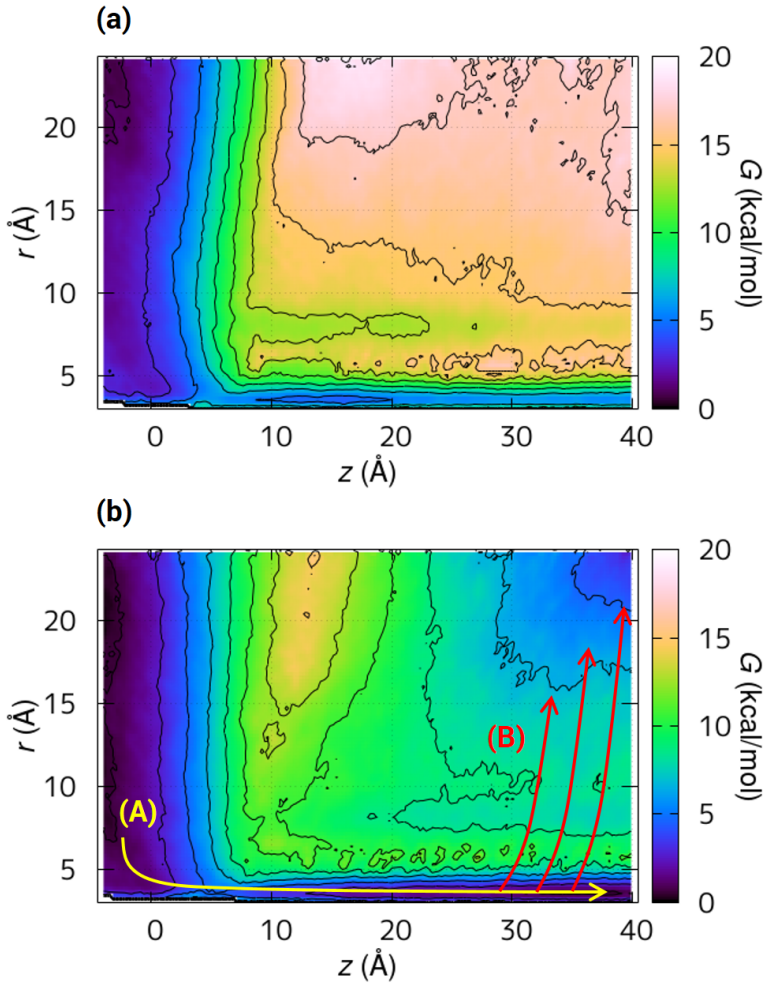


図 3.2　F^--THA^+ のイオン輸送二次元自由エネルギー面 $G^{(2)}(z,r)$。(a) は E_z= 0.0 V/nm、 (b) は 0.2 V/nm の電場を印加した時の結果。等高線は 2 kcal/mol 間隔でプロットしている。パネル (b) では経路 (A) と (B) が黄色と赤の矢印で描かれている。

合には F^- の輸送は ($z \to \infty$) で経路 (B) を通る。図 3.1.1(b) の経路 (B) を通過した場合の活性化障壁は $r \sim 6$ Å で約 11.5 kcal/mol で z (>10 Å) の広い範囲でどこも同じ障壁だった。F^- が経路 (B) の活性化障壁を抜けた後は、z と r が大きくなる方向に下り坂を転がっていく。つまり、F^--THA^+ のイオンペアが解離して油相へ移動していく。

また F^--TBA^+ の場合も調べた結果を図 3.1.1 に示す。

電場無しの図 3.1.1(a) では、F^--TBA^+ の場合も F^--THA^+ と同じ意味を持つ結果が得られた。そして、電場を印加した場合も図 3.1.1(b) に示すように、やはり F–TBA^+ の輸送に対して $G^{(2)}(z,r)$ 上に 2 つの経路が見える。F–TBA^+ と F–THA^+ の両方で、イオンペア形成する経路 (A) が同様に存在するが、F–TBA^+ (図 3.1.1(b)) の経路 (B) と F–THA^+ (図 3.1.1(b)) の経路 (B) は区別できる。図 3.1.1(b) で、経路 (B) の遷移状態は (z = 13 Å,r = 13 Å) 辺りに 14 kcal/mol の障壁で存在する。$r = 13\mathring{A}$ という大きな r は経路 (B) の遷移状態で F–TBA^+ のイオンペア形成は形成していないことを意味する。そして 14 kcal/mol という障壁の高さは F–THA^+ の場合の障壁の 11.5 kcal/mol よりも高く、F^- だけの場合の輸送の障壁と同じであった。THA^+ と TBA^+ を比較した結果、THA^+ は F^- の輸送を界面近くで一瞬イオンペア形成をすることによって触媒的なイオン輸送をするが、TBA^+ の場合はイオンペア形成しないという異なる結果が得られた。ではなぜ二種類のリガンドは F^- の輸送に対して異なる役割をしているのだろう

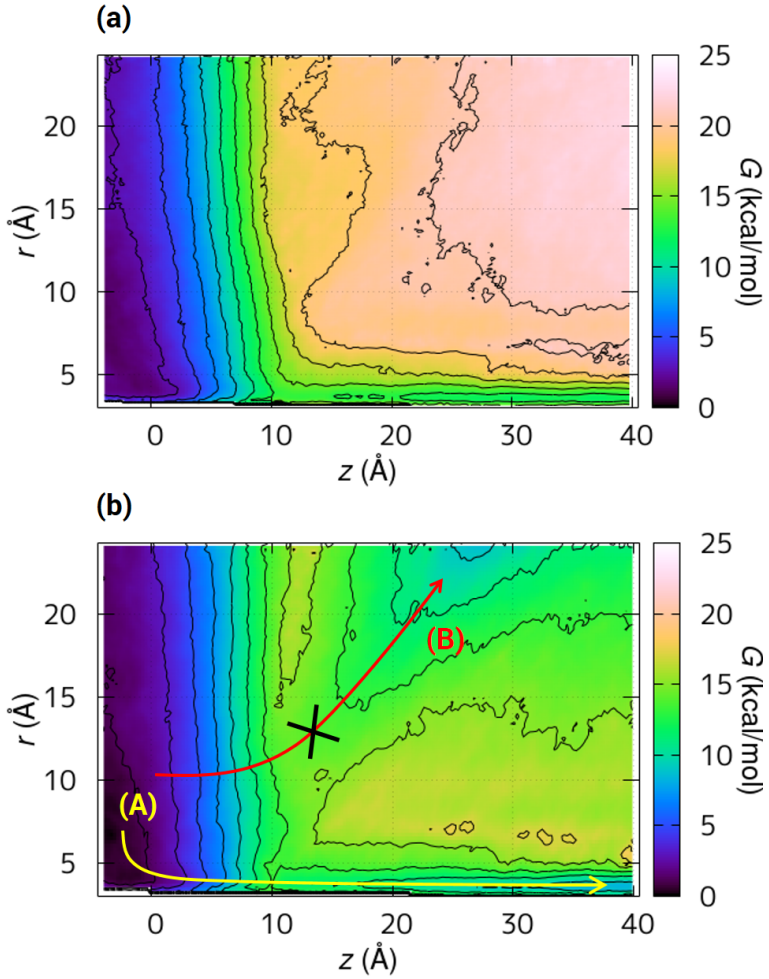


図 3.3　F^--TBA^+ のイオン輸送二次元自由エネルギー面 $G^{(2)}(z,r)$。(a) は E_z= 0.0 V/nm、 (b) は 0.2 V/nm の電場を印加した時の結果。等高線は 2 kcal/mol 間隔でプロットしている。パネル (b) では経路 (A) と (B) が黄色と赤の矢印で描かれている。また X は経路 (B) での遷移状態を意味する。

か。THA^+ と TBA^+ は THA^+ が TBA^+ よりも親油性が高いことを除けば非常に似た物質である。

まず電場無しの場合の F–THA^+ と F–TBA^+ の $G^{(2)}(z,r)$ を比較すると似たような形状をしている。with no field. F^- 単独の輸送は ($z >> 0$) で非常に不安定で、THA^+ と TBA^+ のどちらでも経路 (A) に沿って F^- と油相でイオンペア形成をする。しかし、これだけでは極微量のリガンドで輸送を促進する F^- の触媒的な輸送を解明することはできない。次に電場を印加した場合では、経路 (B) の領域 ($z \sim r \to \infty$) が安定化されて、油相へイオンが輸送される道ができている。

まとめると、図 3.1.1 の二次元自由エネルギー面 $G^{(2)}(z,r)$ 上の (B-1) と (B-2) のように右上の (B) に向かって、つまり水相 ($z << 0, r >> 0$) から油相 ($z >> 0, r >> 0$) へと抜ける 2 つの安定な経路が見える。

(B-1) の経路は (B) へ向かう間にイオンペア形成をする ($r \sim 4$ Å) 一方で、(B-2) の経路ではイオンペア形成をせず直接 (B) につながっている。つまり経路 (B-1) は一瞬起こるイオンペア形成が輸送を助けることを意味するが経路 (B-2) はイオンは単独で輸送する経路である。したがって安定な経路が (B-1) と (B-2) に分岐をけっていするメカニズムがある。輸送されるイオンは非常に親水的で、一方リガンドは疎水的である。非常に親水的なイオンが油相 ($z > 0$) に入る経路 (B-2) のバリアは高く、イオンペア形成をする領域 ($r \sim 4$ Å) は比較的安定だから経路 (B-1) が優先的に

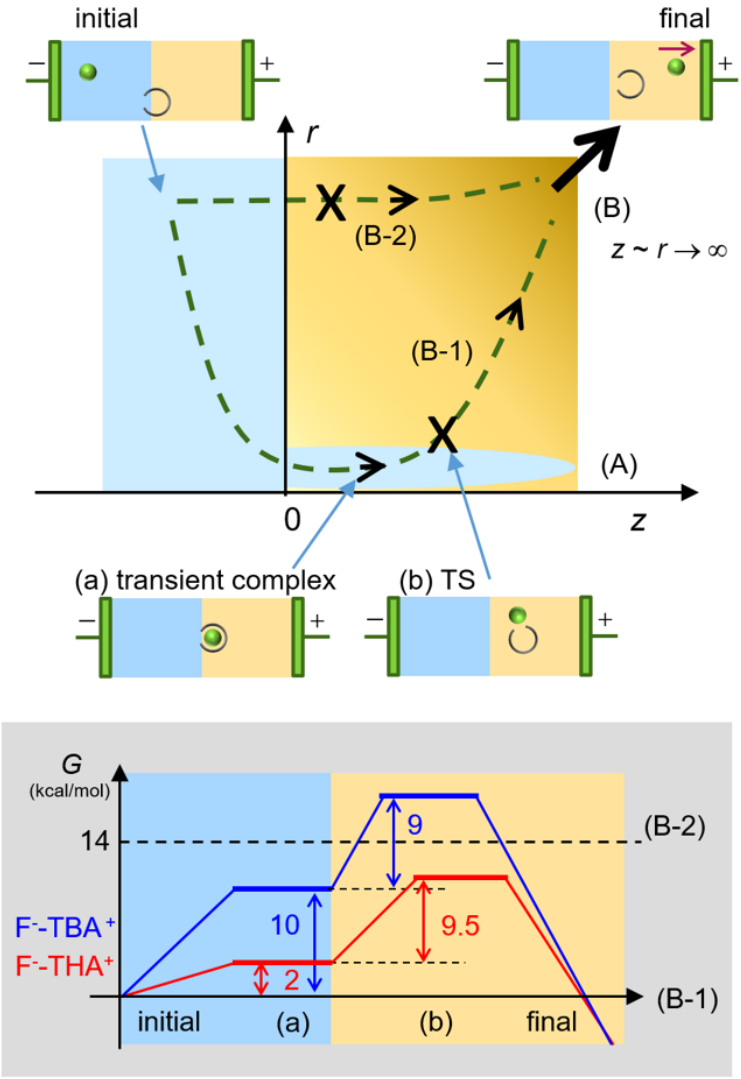


図 3.4　(上段) (z, r) 空間における (B-1: shuttle) と (B-2: no shuttle) の反応経路。X は遷移状態 (TS) を意味する。(下段)　F–THA$^+$ (赤) と F–TBA$^+$ (青) の経路 (B-1) に沿ったエネルギーダイアグラム。(a) イオンペア形成した状態、(b) イオンペア解離の障壁を表す。水平な点線は経路 (B-2) の障壁の高さを表す。

起こるはずである。経路 (B-1) が (B-2) よりも優先される場合、イオン輸送に対して一瞬のイオンペア形成が障壁を下げることを意味する。図 3.1.1 の下段は経路 (B-1) に沿ったエネルギーダイアグラムを見せる。(a) が油相でイオンペア形成をしたところを、(b) がイオンペア解離した時の障壁の高さを積算した形になっている。F–THA$^+$ と F–TBA$^+$ の系の両方で、経路 (B-1) を通った場合の (a) と (b) の内訳をエネルギーダイアグラムにまとめた。経路 (B-1) に沿った場合の全体の障壁は、F–THA$^+$ では 2+9.5 ∼ 11.5 kcal/mol で、F–TBA$^+$ では 10+9∼19 kcal/mol となる。一方で、経路 (B-2) に沿った障壁はリガンドの種類に関係なく 14 kcal/mol の障壁となっている。二つの経路に沿った障壁を比較すると、F–THA$^+$ では (B-1: 触媒作用がある) の経路が優先され、F–TBA$^+$ では (B-2:触媒作用が無い) の経路が優先される。

本研究で自由エネルギーに基づいた解析によって THA$^+$ と TBA$^+$ の差は、油相でのイオンペア形成をした時の安定性によって出ていることがわかった。また油水界面を通過するイオン輸送における一瞬のイオンペア形成の触媒効果

について示された。計算された二次元自由エネルギー面 $G^{(2)}(z,r)$ は THA^+ は F^- のイオン輸送を促進するが TBA^+ は促進しないことを明らかにし、この結果は実験事実と一致する。イオン輸送の経路の分岐のメカニズムが二種類のリガンドの親油性の差によって示され、十分な親油性をもつリガンドでは経路 (B-1) が開かれることが明らかになった。この微視的な解析は触媒機構が働く場合とそうでない場合の区別に役立つ。このピクチャーは他の系にも当てはまり、ITIES のイオン輸送を助けるための効果的なリガンド設計にも役立つと考えられる。

3.2　カチオンの場合の FIT の検証

3.2.1　Li^+-$TPBCl^-$ の場合

更に実験的に輸送が起こることが確認されている Li^+-$TPBCl^-$ の FIT についても検証を行った。SPC/E 水 2091 個、Dang の DCM 2116 個、Li^+1 個、TPBCl1 個からなる図 3.5 の $50 \times 50 \times 115 Å^3$ の系を用意してイオンのみ電荷のスケーリング [40] をしたものを使用した。TPBCl の分子モデルについては、既存のモデルが見つからなかったため作成した。TPBCl のモデルについては、付録 3.3.3 に記載した。

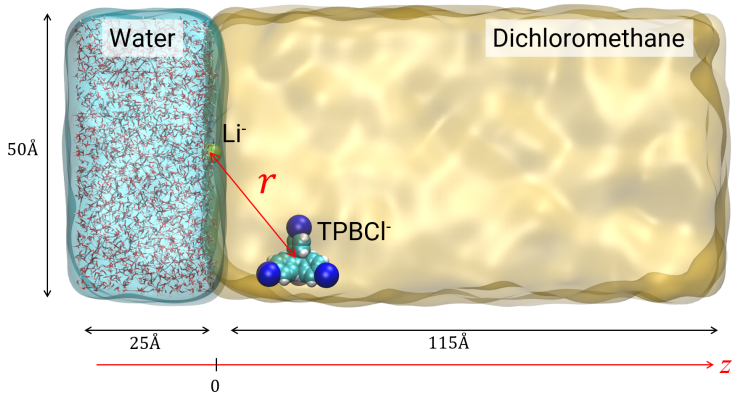


図 3.5　Li^+ と $TPBCl^-$ を含む水/DCM 界面系

結果を図 3.6 に示す。Li^+-$TPBCl^-$ の輸送に関しても実験結果と対応する F^--THA^+ と同様にシャトリングが起こ

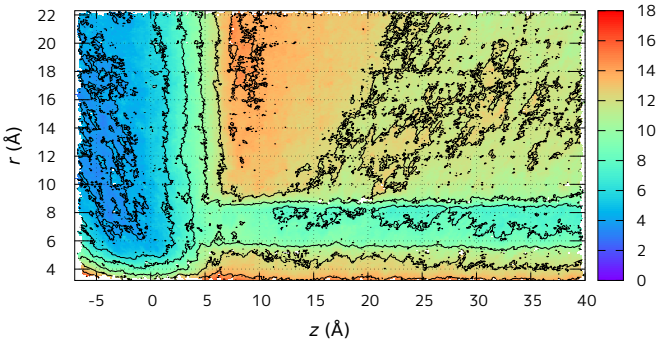


図 3.6　0.1V/nm での二次元自由エネルギー面 $G_{(z,r)}$

るパスが確認できた。

3.3　補足

3.3.1　四級アミンの分子モデリング

ここでは 3, 4 章で用いた四級アミン (TBA, THA, Sym, Asym) の分子モデルの作成について記述する。四級アミンのモデルは、AMBER ff03ua[41] をもとにして、アルキル鎖の二面角に関するパラメータについて変更して使用した。この変更は、ゴーシュ (g)/トランス (t) の存在確率自体は変えずに g/t 間の障壁を下げて MD シミュレーションで追跡可能なタイムスケールの中で g/t 間の遷移が十分に起こるモデルにすることを目的として行った。

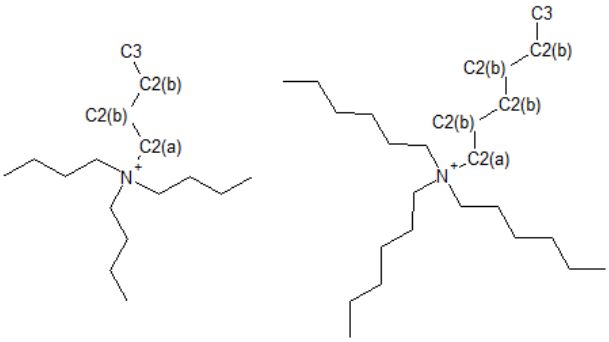


図 3.7　TBA（左）と THA（右）分子のサイト名

分子モデルの力場は AMBER ff03ua を基にしており、分子内のコンフォメーションやアルキル鎖のねじれを記述するフレキシブルモデルを採用している。分子内のポテンシャル関数は結合長の伸縮、折れ曲がり、ねじれを記述する次式で表される。

$$V_{\mathrm{bonds}} = \frac{1}{2}\sum_{i}^{\mathrm{bonds}} k_r^{(i)}\left(r^{(i)} - r_0^{(i)}\right)^2, \tag{3.1}$$

$$V_{\mathrm{angles}} = \frac{1}{2}\sum_{i}^{\mathrm{angles}} k_\theta^{(i)}\left(\theta^{(i)} - \theta_0^{(i)}\right)^2, \tag{3.2}$$

$$\begin{aligned} V_{\mathrm{dihederals}} = \sum_{i}^{\mathrm{dihedrals}} \Big[& V_1^{(i)}\{1+\cos(\phi^{(i)} - \gamma_1^{(i)})\} + V_2^{(i)}\{1+\cos(-2\phi^{(i)} - \gamma_2^{(i)})\} \\ & + V_3\{1+\cos(-3\phi^{(i)} - \gamma_3^{(i)})\} + \frac{1}{2}V_{\mathrm{LJ},1-4}\Big]. \end{aligned} \tag{3.3}$$

力場パラメータは表 3.3.1 にまとめた。

遷移状態理論から予測すると、g→t, t→g の速度定数 $k_{\mathrm{g}\to\mathrm{t}}$ と $k_{\mathrm{t}\to\mathrm{g}}$ は、g, t での振動数 ν_{g}, ν_{t} を用いて、

$$k_{\mathrm{g}\to\mathrm{t}} = \nu_{\mathrm{g}} \exp\left(\frac{-E_{\mathrm{g}\to\mathrm{TS}}}{\mathrm{k_B}T}\right), \tag{3.4}$$

$$k_{\mathrm{t}\to\mathrm{g}} = \nu_{\mathrm{t}} \exp\left(\frac{-E_{\mathrm{t}\to\mathrm{TS}}}{\mathrm{k_B}T}\right). \tag{3.5}$$

振動数は Gaussian09 で計算し、$\nu_{\mathrm{g}} \simeq \nu_{\mathrm{t}} = 4.0 \times 10^{-9}\mathrm{s}^{-1}$ であった。これを用いて式 3.5 より 298.15 K で見積もった遷移頻度を表 3.1 にまとめた。

表 3.1　ゴーシュ・トランス間の遷移頻度の見積もり

$E_{g\to t}$[kcal/mol]	遷移頻度 [回/100 ps]
2.4(QM)	6.8
1.9	16
1.4	37.3
0.9	87
$E_{t\to g}$[kcal/mol]	遷移頻度 [回/100 ps]
3.24(QM)	1.65
2.74	3.85
2.24	9
1.74	20.1

したがって障壁を 1 kcal/mol 下げることで 5.5 倍程遷移頻度を高めることができ、MD で追跡可能なタイムスケール 100 ps 〜 1 ns の中で構造緩和とサンプリングが十分に起こりうることが予想される。g/t 遷移の障壁を 1 kcal/mol 程度下げたパラメータを 3.3.1(d') にまとめた。

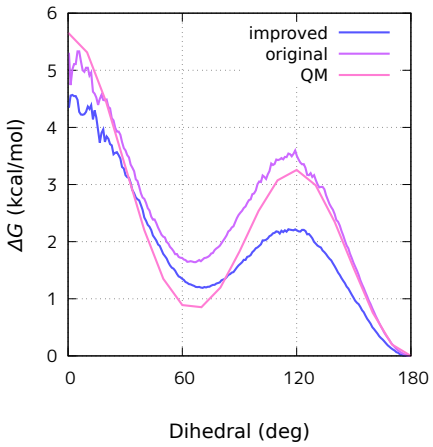


図 3.8　TBA のアルキル鎖の二面角のポテンシャル面

298.15 K で真空中に TBA のみが存在する系で C-C-C-C のねじれ角について 1n s 程度サンプリングし、その分布を自由エネルギーに変換したものを図 3.8 に示した。そして、MD 中では g → t と t → g の遷移頻度は同程度でオリジナルで 2.5 回/100ps、改良したモデルで、14.8 回/100ps で 6 倍程度遷移頻度が高くなった。オリジナルパラメータでは MD のタイムスケールで分子のコンフォメーションまで含めた十分なサンプリングが難しかったが、g/t 遷移の障壁を 1 kcal/mol 程度下げ、MD で追跡可能なモデルを実現した。

表 3.2　四級アミンの力場

(a) サイトパラメータ

site	q (e)	α (Å^3)	σ (Å)	ε (kcal/mol)
N	0.00	0.0	3.250	0.170
C2 (a)	0.25	0.0	3.905	0.118
C2 (b)	0.0	0.0	3.905	0.118
C3	0.0	0.0	3.905	0.175

(b) 結合パラメータ

bond	k_r (kcal/mol ·Å^2)	r_0 (Å)
N-C2	367.0	1.471
C2-C2	310.0	1.526
C2-C3	310.0	1.526

(c) 角度パラメータ

bond	k_θ (kcal/mol ·degree^2)	θ_0 (degree)
C2-N-C2	50.0	109.50
N-C2-C2	80.0	111.20
C2-C2-C2	40.0	109.50
C2-C2-C3	40.0	109.50

(d) 二面角パラメータ（オリジナル）

bond	V_n (kcal/mol ·degree^2)	γ_n (degree)
N-C2-C2-C2	1.00	180.0
	0.6	180.0
	1.6	0.0
C2-C2-C2-C2	1.00	180.0
	0.6	180.0
	1.6	0.0
C2-C2-C2-C3	1.00	180.0
	0.6	180.0
	1.6	0.0

(d') 二面角パラメータ（改良後）

bond	V_n (kcal/mol ·degree^2)	γ_n (degree)
N-C2-C2-C2	1.00	180.0
	0.24	180.0
	1.12	0.0
C2-C2-C2-C2	1.00	180.0
	0.24	180.0
	1.12	0.0
C2-C2-C2-C3	1.00	180.0
	0.24	180.0
	1.12	0.0

3.3.2　Li^+ のモデル

表 3.3　Li^+ の力場

site	q [e]	σ [Å]	ε [kcal/mol]
Li	0.75	1.4307	0.165

MD は表 3.3 の電荷を 0.75 倍でスケーリングした後に実施された。これは水中を想定して電荷を先にスケーリングしておくことで分極計算することなく電子分極の効果を入れることができるというアイデアで、そのスケーリングパラメータは溶媒の屈折率から $\sqrt{\frac{1}{n^2}}$ で見積もられる。[40]。溶媒が水の場合に 0.75 となるが、今回計算を行うのは水/DCM 界面系である。したがって DCM の場合にも同じパラメータでスケールしてもよいか確認した。水と DCM の屈折率はそれぞれ 1.31 と 1.4125 であり、屈折率から見積もられるスケーリングパラメータは 0.75 と 0.71 になった。水と DCM のスケーリングパラメータにほとんど差がないため、系に含まれるイオンの電荷は全て一律に 0.75 でスケールされた。また、電荷をスケールすることによって近距離での相互作用で安定化少し損なわれる。これを補うために LJ パラメータの σ を 0.95 でスケールした。図 3.9 にスケーリングを適用した Li^+ の水中での RDF を示す。AIMD の結果 [42] と比較して、スケーリング後の RDF がより AIMD の結果と合っていることを確認した。

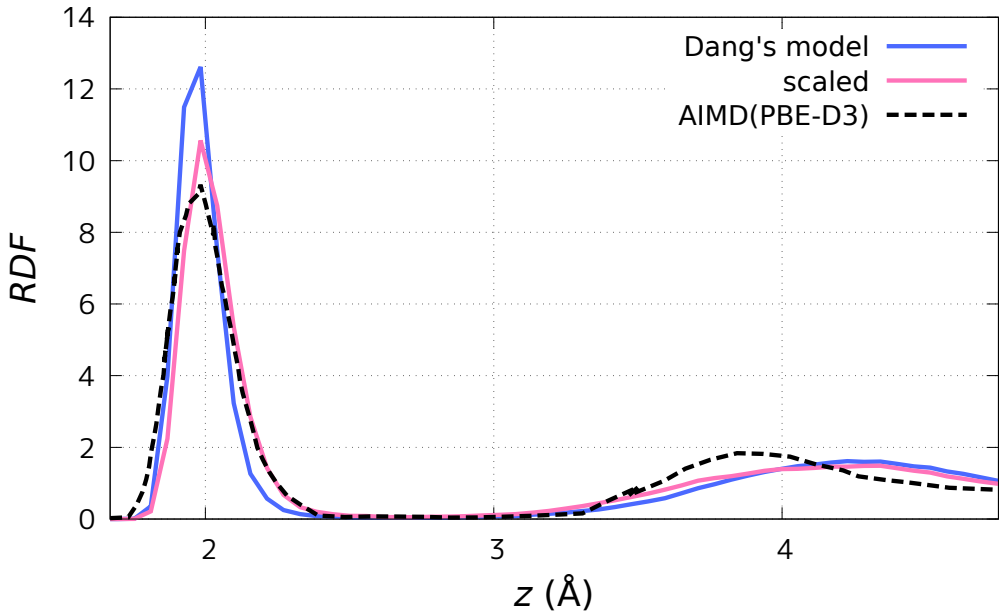


図 3.9　水中の Li^+ の RDF の比較。

3.3.3　テトラクロロフェニルホウ酸 (TPBCl) の分子モデリング

Li^+, Ca^{2+} カチオンの輸送についても同様に実験結果をサポートするような結果が得られるのか検証するためにそのカウンターイオンとして代表的なテトラクロロフェニルホウ酸 $TPBCl^-$ の分子モデリングを行った。TPBCl 分子は Gaussian09[43] を用いて B3LYP/6-31+G(d,p) レベルで最適化され、ChelpG (charges from electrostatic potentials using a grid-based method) オプションで部分電荷を決定した。サイト B 以外のサイトに関する LJ パラメータは CHARMM 力場のパラメータセットを採用した。そしてサイト B に関しては Otkidach らによって開発されたパラメータを採用した。[44]. また分子内二面角のバイアスポテンシャルも Otkidach らのパラメータを採用した。表 3.4 にパラメータをまとめた。

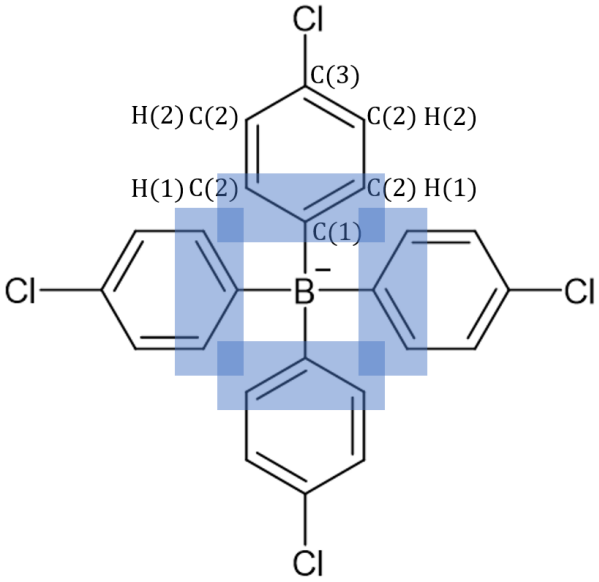


図 3.10　TPBCl と各サイトの対応。C(1), C(2), H(1) を含む青い部分同士には LJ とクーロン相互作用が働く。

表 3.4　$TPBCl^-$ の力場パラメータ

site	q [e]	σ [Å]	ε [kcal/mol]
B	0.00000000	3.52795892	0.03400000
C(1)	0.07367500	3.55005321	0.07000000
C(2)	-0.11710625	3.55005321	0.07000000
C(3)	0.03900000	3.55005321	0.07000000
H(1)	0.07706250	1.95997718	0.04600000
H(2)	0.07706250	2.42003728	0.03000000
Cl	-0.20250000	3.31414323	0.23000000

二面角は次式で記述される。

$$U_{\text{bias}} = \frac{V^{(0)}}{2} + \frac{V^{(0)}}{2}\cos\left(V^{(1)}\phi\right) + U_{\text{LJ}} + U_{\text{Coulomb}}. \tag{3.6}$$

パラメータ $V^{(n)}$ は表 3.5 にまとめた。

表 3.5　TPBCl の二面角パラメータ

n	V_n [kcal/mol]
0	0.04
1	6

このモデルを用いて TPBCl の二面角に関する分子内回転のポテンシャル面を計算したものが図 3.11 である。TPBCl のフェニル基回転のポテンシャル面はほとんど障壁がなく自由に回転できるモデルとなっていた。構造緩和が

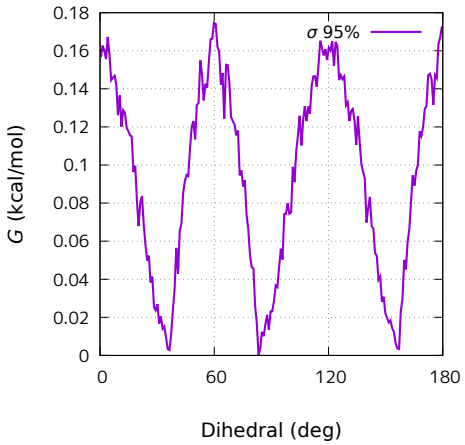


図 3.11　TPBCl のフェニル基回転のポテンシャル面

起こりやすいことにおいて MD のタイムスケールでの平衡化とサンプリングに都合がよい。

第 4 章

相間移動触媒反応

3 章では、油水界面における親水性イオンの輸送に関する親油性カウンターイオンの効果について実験と分子動力学シミュレーションの双方からアプローチし、界面近傍でのイオン輸送を触媒する機構を分子レベルで明らかにした結果を示した。この章では、3 章の研究をさらに発展させて実際の液液界面を反応場とする相間移動触媒反応の反応機構解析を行った結果を示す。混ざり合わない油水二相下で、水相に溶けている親水性イオンと油相に溶けている疎水性反応物の反応は一般に起こらないが、相関移動触媒（PTC) を系に加えると反応が起こるようになることが知られている。この反応はこれまでの有機合成での反応とは異なり界面を反応場とする反応として合成の幅を広げ、大規模合成向きであることやグリーンケミストリーの観点から、工業化学分野においても重要な反応である。しかし、この反応機構は未解明であり、これまでに実験家によって反応速度論を用いた反応機構解析が行われてきた。PTC 反応の代表的な反応を 2 つあげ、これまでに提案されている反応機構とその根拠となるデータを示す。

フェニルアセトニトリルのアルキル化は化学式 4 で表される反応で、油水界面で起こる反応と考えられている。その

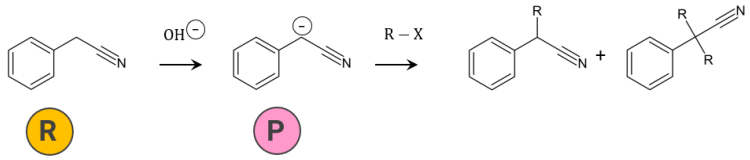


図 4.1　フェニルアセトニトリルのアルキル化

根拠として図 4 の系を攪拌させた時の反応速度に対する効果を調べた実験結果がある。界面系を攪拌させ、攪拌速度を変化させると共に界面積が増加する効果があると考えており、攪拌速度に対して反応速度が 2000 rpm まで上昇していることからフェニルアセトニトリルのアルキル化は界面で塩基触媒反応が起こる界面反応が提案されていた。図 4 に界面反応の反応機構を示す。界面反応では、油水界面で親水性 OH^- と反応物 R が出会い、界面で化学式 4 の第一ステップのプロトン引き抜きが起き、P が生成される。この P は電荷を持つため親水性と親油性を持ち、界面に留まる。フェニルアセトニトリルのアルキル化では界面に留まった P を油相バルクへと輸送させて後続の反応を起こすために PTC が P の輸送を促進する働きがあると考えられている。

続いて、アリルベンゼン異性化反応について説明する。アリルベンゼン異性化反応は化学式 4 で表される反応で、油相バルクで起こる反応と考えられている。その根拠として図 4 の攪拌速度を上げても反応速度が頭打ちになる実験結果が報告されている。図 4 と比較して、界面増加が反応に効かないことがアリルベンゼン異性化反応が界面ではなく油相バルクで起こっていることを示唆している。また、図 4 の PTC の親油性を上げることで反応効率が高くなる実験結果が報告されている。PTC の親油性の高さが反応を促進していることから油相へイオン輸送が起こってから油相バルク

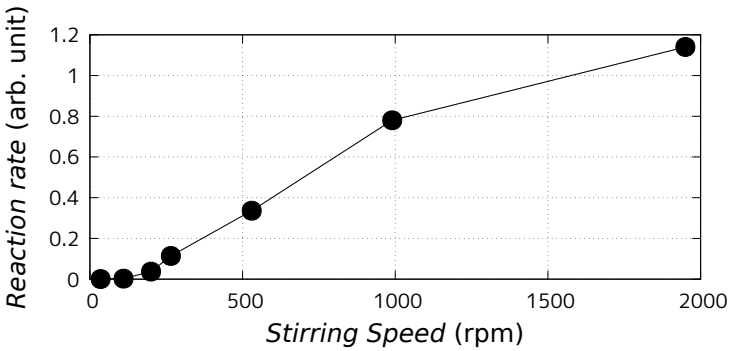


図 4.2　フェニルアセトニトリルのアルキル化反応に対する攪拌の効果

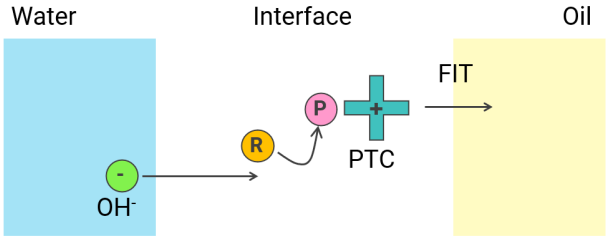


図 4.3　フェニルアセトニトリルの界面反応

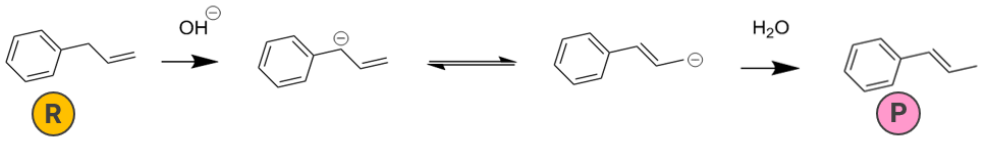


図 4.4　アリルベンゼン異性化反応

で塩基触媒反応が起こる反応機構が提案されている。しかし、OH^- は非常に親水性の高いイオンで油相への輸送が困難であることが考えられ、PTC 条件でも反応を起こすことが難しいと言われている。本研究ではアリルベンゼン異性化反応を取り上げて反応機構解析を行う。

4.1　反応解析

4.1.1　分子モデル

本研究では非分極と分極の両方の計算を実施し比較した。非分極モデルでは 6 サイトベンゼンと SPC/E[45, 46] 水を採用し、分極モデルでは 12 サイトベンゼン [47] と POL3[48] 水を採用した。PTC は 3 章の 3.3.1 のパラメータを用いて図 4.1.1 に示す対称アミン (Sym, $N^+(C_{10}H_{21})_4$) と非対称アミン (Asym, $N^+(CH_3)(C_{13}H_{27})_3$) を比較した。この 2 つは炭素数が等しいため親油性が同じもの同士での比較となっている。

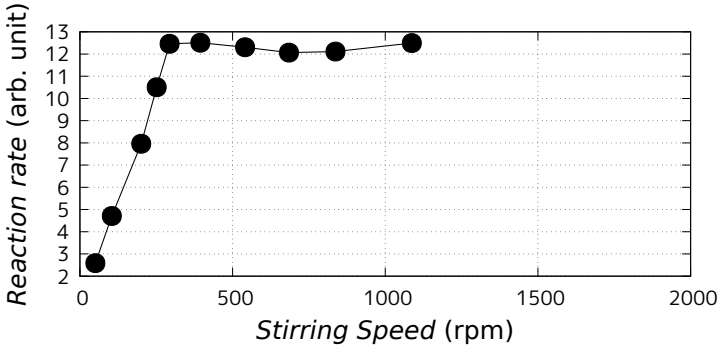


図 4.5　アリルベンゼン異性化反応に対する攪拌の効果

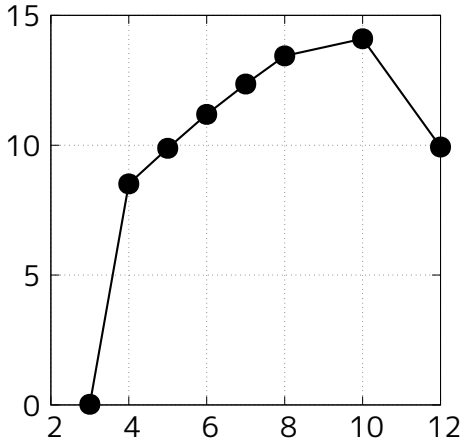


図 4.6　アリルベンゼン異性化反応に対する PTC の親油性の効果

4.1.2　イオン輸送自由エネルギー計算

セルサイズ $40 \times 40 \times 125$Å^3 の水分子 1333 個、ベンゼン 1078 個から成る系で、z 方向に水相が 25Å、ベンゼン相が 100Å の厚みのスラブ系を用意し、OH^- は界面に、カウンターイオンがある系ではカウンターイオンは界面近くのベンゼン相に配置した初期構造を用いて MD を実施した。計算コストを削減するため、時間刻み幅 Δt を大きくとることができるように水の水素を重水素化し、Δt =2 fs で計算を実施した。平衡化はバイアスポテンシャルをかけて 200ps、サンプリングは 2 章の 2.2.1 に示したレプリカ交換アンブレラサンプリング法 (REUS) で 400ps 実施した。

OH^- のみの輸送では界面からの垂直距離 z と、water finger 座標 [5, 6] の 2 次元自由エネルギー $G(z, w)$ を表 4.1.2 のバイアスをかけて計算し、2 次元自由エネルギー $G(z, w)$ を得た。得られた $G(z, w)$ について次式で積分し $G(z)$ を得た。

$$G(z) = -k_B T \ln \int_0^\infty dw \exp\left(-\frac{G(z, w)}{k_B T}\right) \tag{4.1}$$

OH^- の水相から油相へのイオン輸送自由エネルギーを図 4.1.2 に示す。

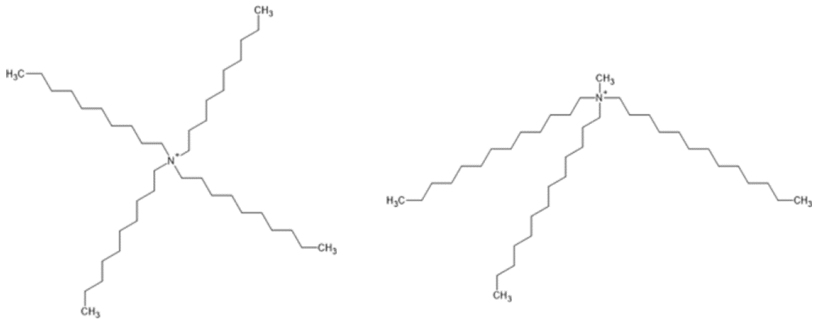


図 4.7　対称アミン (左, Sym) と非対称アミン (右, Asym) の構造。

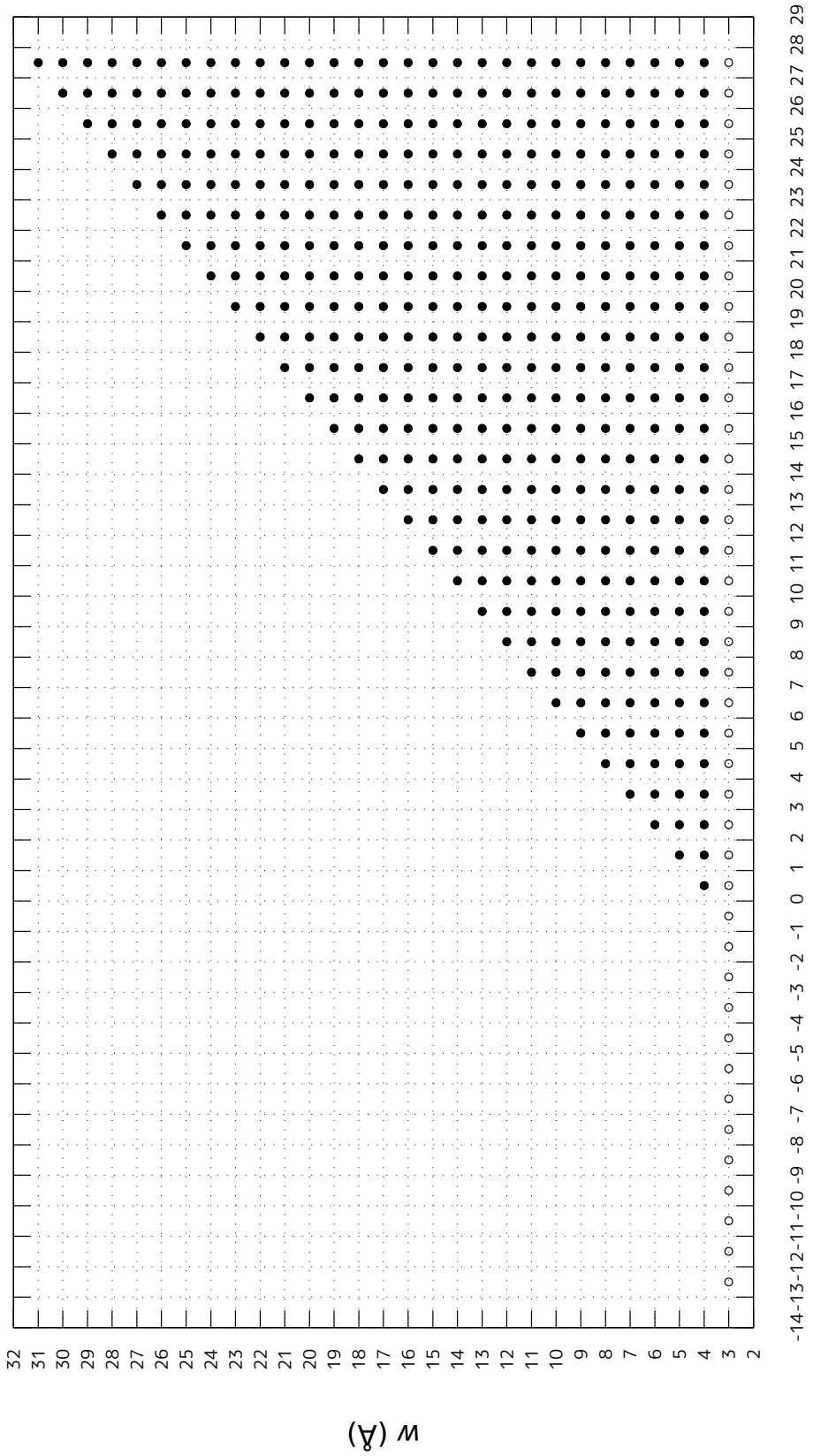


図 4.8　二次元自由エネルギー $G^{(2)}(z,w)$ 計算でかけたバイアスポテンシャルの位置。1Å 間隔で置かれ、$z=0$ は水/ベンゼン界面を表す。黒丸は z と w の両方にバイアスがかけられ、白丸は z のみかけられている。

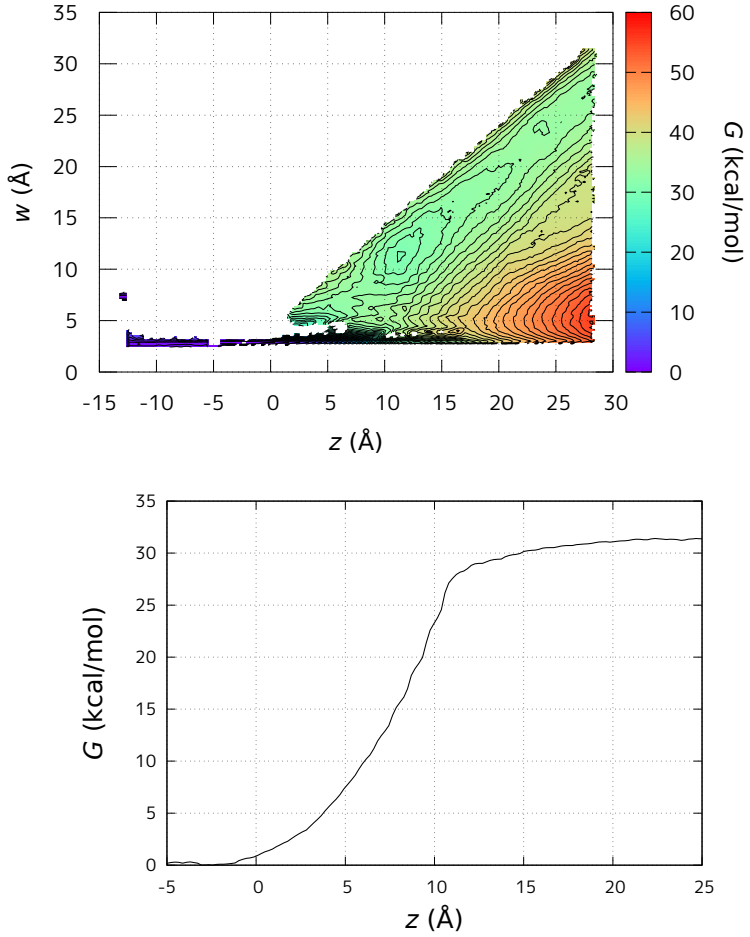


図 4.9　OH^- 単独での輸送自由エネルギー。上段は $G(z,w)$、下段は $G(z)$ をプロットしている。

また Sym-OH^- と Asym-OH^- の水相から油相へのイオン輸送自由エネルギーは OH^- の界面からの位置 z に加えて OH^- とカウンターイオン間の距離 r の座標に対して表 4.1.2 に示すようにバイアスをかけて REUS を実施し 2 次元自由エネルギー $G(z,r)$ を計算した。得られた $G(z,r)$ について次式で積分し $G(z)$ を得た。

$$G(z) = -k_B T \ln \int_0^\infty 4\pi r^2 dr \exp\left(-\frac{G(z,r)}{k_B T}\right) \tag{4.2}$$

Sym-OH^- と Asym-OH^- の 2 次元自由エネルギー面の結果を図 4.1.2 に、積分して求めた 1 次元自由エネルギーの結果を図 4.1.2 に示す。

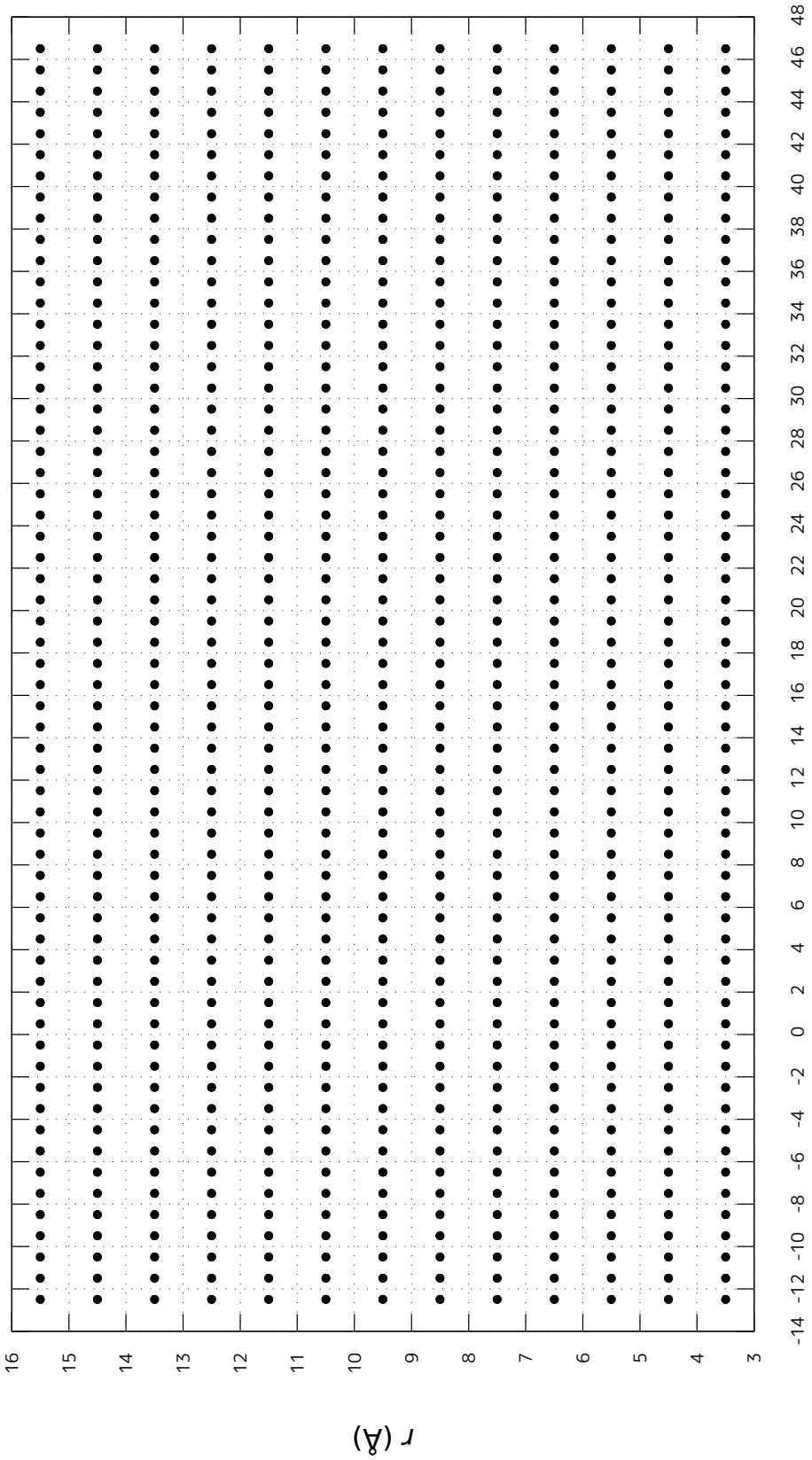


図 4.10　二次元自由エネルギー $G^{(2)}(z,r)$ 計算でかけたバイアスポテンシャルの位置。1Å 間隔で置かれ、$z=0$ は水/ベンゼン界面を表す。

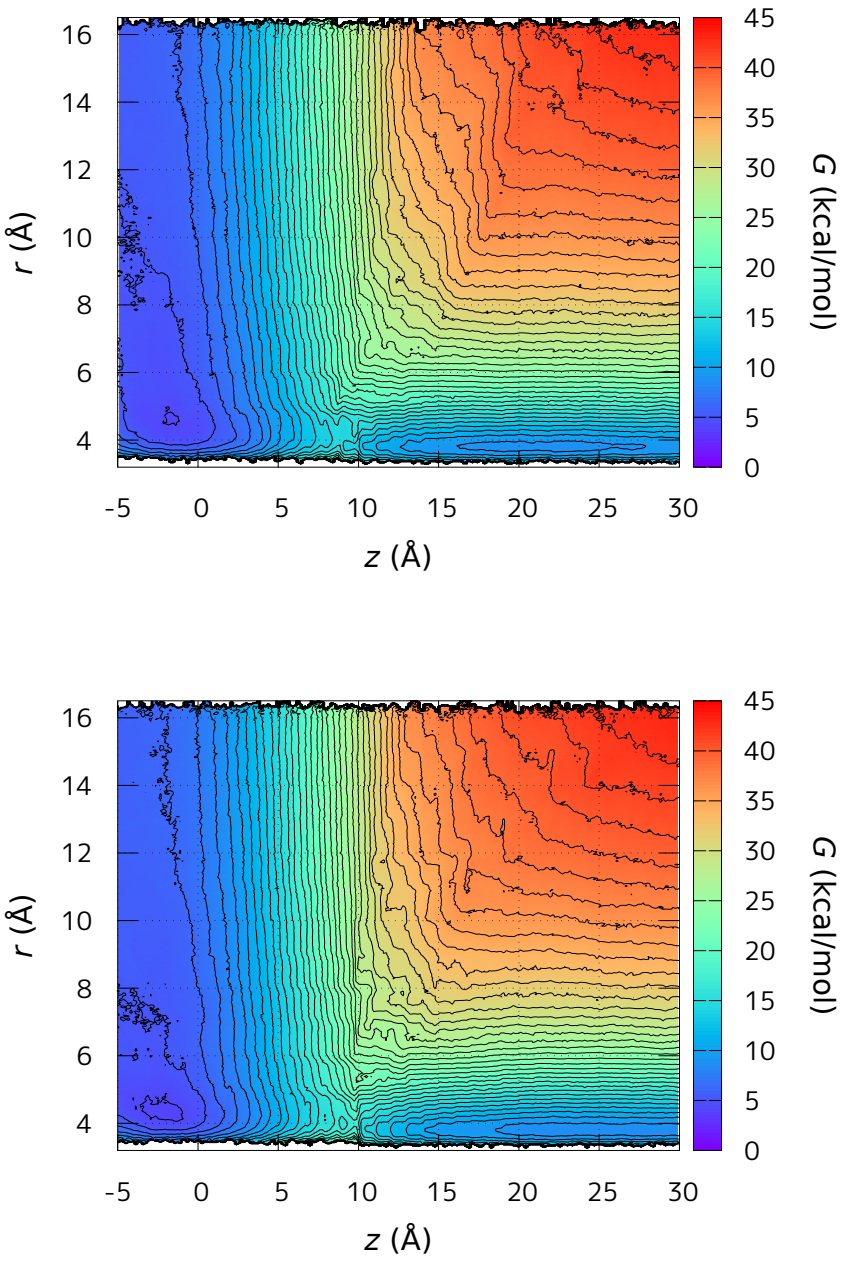


図 4.11　2 次元イオン輸送自由エネルギー $G(z, r)$ 上段が Sym-OH^-、下段が Asym-OH^- の系。

対称と非対称の差は見られなかったが、どちらもイオン輸送で界面を通過するあたり $(z = 0)$ にバリアがある。バリア付近のトラジェクトリーを確認すると、図 4.1.2 のイオンが水相側の界面を引っ張った構造ができていた。
これは、界面を通過する時にイオンによって水相が引っ張られてできる不安定な構造が原因であり、その時の界面面積の増加 ΔS による界面張力由来の不安定化 ΔG を見積もることで確かめられた。バリアは OH^- によって水相が引っ張られて形成され、PTC とのイオンペア形成による安定化が大きくなったところでバリアが解消されると考えている。つまり、

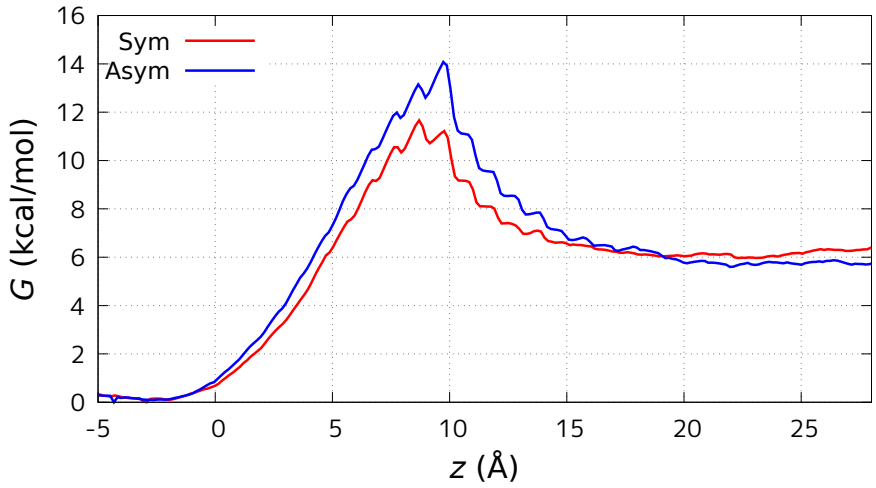


図 4.12　Sym-OH^-, Asym-OH^- の輸送自由エネルギーの比較。上段から Sym と Asym の順に記載。

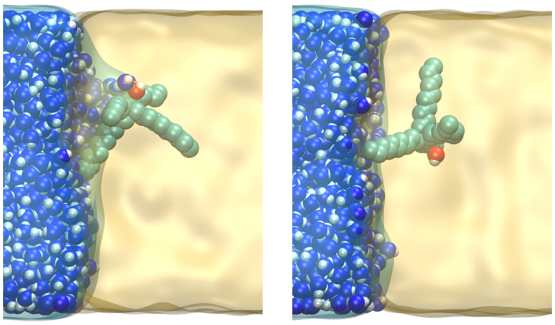


図 4.13　バリアの構造。左はバリアの頂点での系、右は頂点より少し z が大きい系の構造。

1. OH^- が油相へ移動する時に水和水を引き連れる (安定化)
2. 水和水と水相の水分子が水素結合によって連結する (安定化)
3. 水和水と水相の水分子が水素結合によって連結したことによって界面面積が増加する (不安定化)
4. OH^- と PTC がイオンペア形成する (安定化)
5. PTC が水相に近づく (不安定化)

これらが競合して、バリアの位置と高さが決まると考える。PTC が水相に近づくことによる不安定化は、主に PTC が水相に溶ける時に水-水間の結合を切断することに起因する。この不安定化は界面張力由来であると考えられ、油相の種類を変えた時に界面張力が変化することでバリアの高さは変化することが想定される。

$$\Delta S = \pi r(r + L) - \pi r^2 \tag{4.3}$$

$$\Delta G = \gamma \Delta S \text{ (destabilization)} \tag{4.4}$$

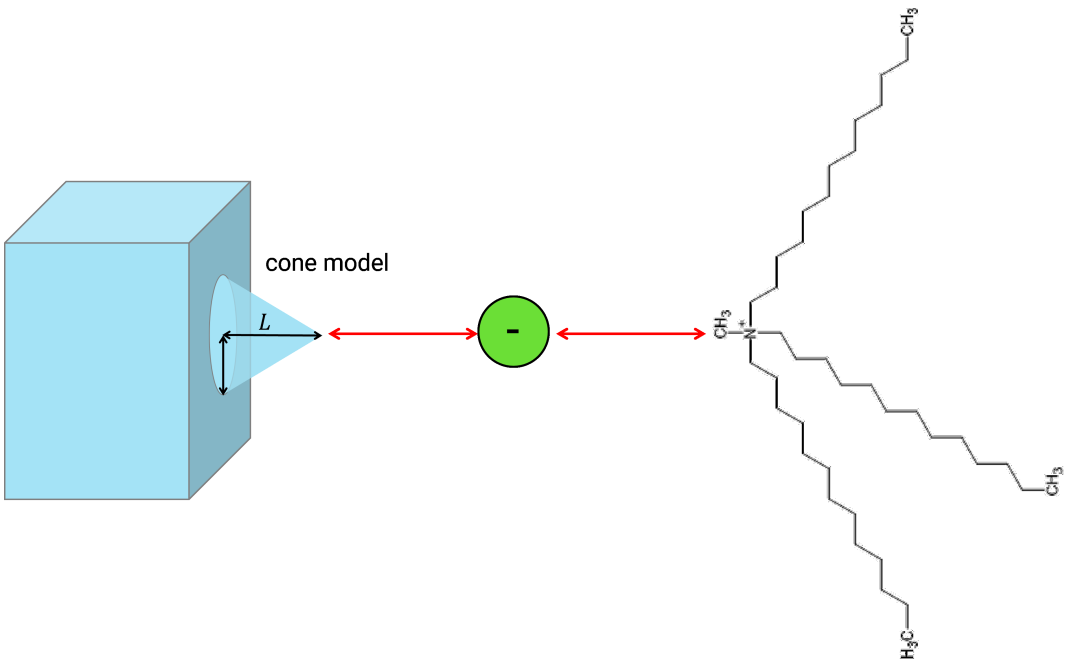


図 4.14　バリア構造の円錐モデル。L は円錐の高さ。

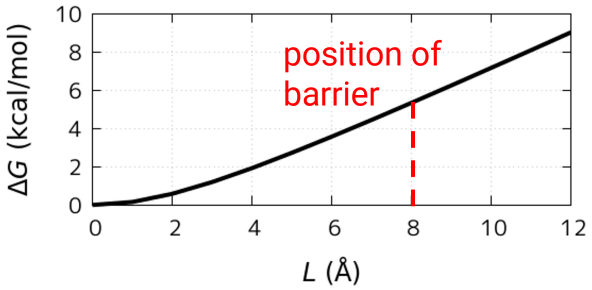


図 4.15　界面増加によって生じる界面張力由来のバリアの見積もり。L は界面増加面積を円錐モデルでの円錐の高さ。

4.1.3　水和数分布

水和数の分布に対する PTC の効果を検証した。また温度の効果、非分極と分極モデルの差も検証した。セルサイズ $40 \times 40 \times 40\text{Å}^3$ の水分子 n_w 個、ベンゼン (431-n_w) 個から成る系を用意した。計算コストを削減するため、時間刻み幅 Δt を大きくとることができるように水の水素を重水素化し、Δt =2 fs で計算を実施した。200ps 平衡化 200 し、200ps サンプリングした。また分極モデルは非分極モデルで平衡化 200ps の後、分極モデルに変換し 50ps 平衡化し、200ps サンプリングした。温度は 298.15K と 348.15K で実施した。

まず PTC が無い場合の OH^- の水和数分布を図 4.1.3 に示す。温度変化による水和数の変化は小さく、$\Delta G = \Delta H - T\Delta S$ においてほとんど ΔH で決まっていると考えられる。

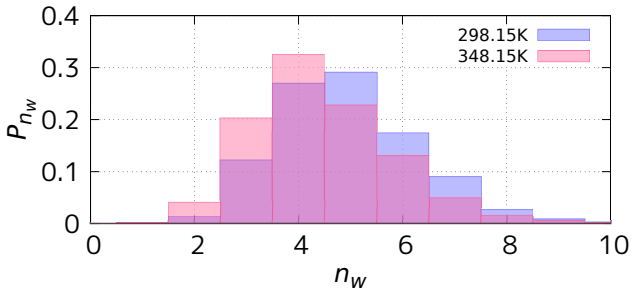


図 4.16　298.15K と 348.15K での OH^- の水和数分布の比較

図 4.17 に示すように、対称アミン (Sym) があるときと無い時 (without) で比較すると分布が n_w が減少する方向へ移動している。まず PTC に水和数を減少させる効果があることがわかった。

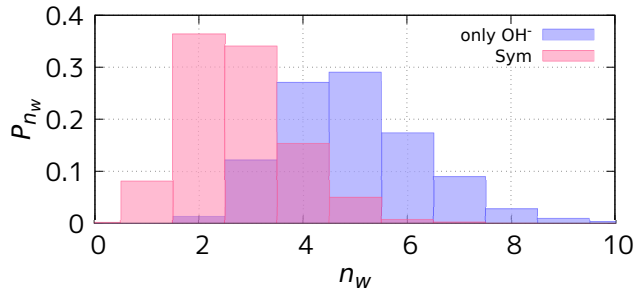


図 4.17　水和数分布に対する対称アミン (Sym) の影響

また、PTC の親油性を変化させて親油性の効果を調べた結果を図 4.18 に示す。横軸は $N^+(C_nH_{2n+1})_4$ の n で、縦

軸は $n_{\mathrm{w}} = \sum_i n_{\mathrm{w}} P_i$ から算出した平均水和数である。$n = 0$ は PTC が無い時を表す。この結果からは、PTC の親油

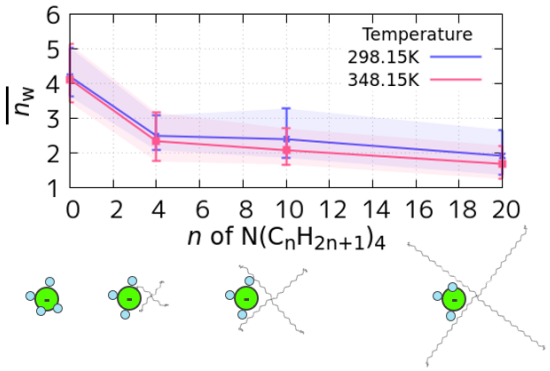


図 4.18　平均水和数に対する PTC の親油性の影響

性の高さによって水和数はほとんど変わらず、PTC 無しの場合と比較して、水和数 2 個程度減っていることがわかる。また温度変化の影響はほとんど無いことが確認できる。

以下の表 4.1〜4.6 に { 非分極, 分極 }{298.15K, 348.15K}{without, Sym, Asym} の結果をまとめた。またそれぞれの条件での平均水和数を表 4.7 にまとめた。

n_{w}	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =298.15K)	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =348.15K)
1	-3.79599563192096	-3.61011448809328
2	-6.23481147969803	-6.03050289210941
3	-7.63061725022179	-7.16347220122250
4	-8.15158878363383	-7.48384468928456
5	-8.22761717121593	-7.17228687942448
6	-8.02929963914627	-6.95679433695040
7	-7.67728122322448	-6.21563726660943
8	-6.99301071687081	-5.38867098364688
9	-6.21777970194424	-4.71212580003916
10	-5.69660871413036	-3.97024683548705

表 4.1　非分極モデル, PTC 無しの時のベンゼン中での OH^- の水和クラスターの自由エネルギー $\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$

表 4.7 の平均水和数で分極と非分極を比較すると、そこまで大差ないように見えるが、$G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ の値で比較すると、分極モデルの方が安定化が大きい。

n_{w}	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =298.15K)	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =348.15K)
1	-2.53569371934902	-2.38524946413089
2	-3.57110117288638	-3.17099795889077
3	-3.54377967768659	-2.82592219620278
4	-3.39848050260530	-2.31518624745876
5	-2.92759927677688	-1.04715645327072
6	-1.61854534224851	0.323334430837932
7	-1.05754533374466	1.47320416882097
8	0.316712913289663	3.10796743142903
9	1.51034569576018	4.11775967158209
10	2.83105480812016	5.69303203860778

表 4.2　非分極モデル, 対称アミンの時のベンゼン中での OH^- の水和クラスターの自由エネルギー $\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$

n_{w}	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =298.15K)	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =348.15K)
1	-2.46967024583996	-2.53604411013300
2	-3.48430055953095	-3.35597982325954
3	-3.51801857494205	-3.55393783591889
4	-3.44339415337967	-3.25326202345590
5	-2.59873360785400	-2.53332793410449
6	-1.65843129129762	-1.44425543441666
7	-0.680751686936603	-0.425571489002948
8	0.568799843251853	0.999900152027621
9	1.55499852983114	2.78216236322523
10	2.53126196690351	4.12265628647100

表 4.3　非分極モデル, 非対称アミンの時のベンゼン中での OH^- の水和クラスターの自由エネルギー $\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$

n_{w}	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =298.15K)	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =348.15K)
1	-6.79312671585725	-7.24222344068414
2	-11.7651627999571	-12.0060512259270
3	-14.9879151986818	-15.6582260008771
4	-16.8008783918037	-17.8114546276887
5	-17.9012130834247	-19.2107079822688
6	-18.6150998442649	-20.2501945120301
7	-18.9393991079328	-20.2388615302590
8	-18.3295583959311	-20.2214020866999
9	-18.0299729244607	-18.0805433711625
10	-17.5697210380778	-17.1352879568396

表 4.4　分極モデル, PTC 無しの時のベンゼン中での OH^- の水和クラスターの自由エネルギー $\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$

n_{w}	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =298.15K)	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =348.15K)
1	-5.42881028204365	-5.46907876867137
2	-9.09661025790244	-8.21780333857931
3	-10.8013280859265	-10.0927707294344
4	-11.2071668757321	-10.1876132969282
5	-10.9841221880237	-10.7784019493212
6	-9.86056881042298	-10.5615141559102
7	-9.09639635891961	-9.99226488956862
8	-8.14577470532878	-9.52538416164718
9	-7.09581383570098	-8.50643294654391
10	-5.54123649795578	-7.32325756955686

表 4.5　分極モデル, 対称アミンの時のベンゼン中での OH^- の水和クラスターの自由エネルギー $\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$

n_{w}	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =298.15K)	$\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$ (T =348.15K)
1	-6.38027957030775	-6.04745160044104
2	-9.81369981772515	-9.80120985836178
3	-11.8954166745226	-11.8122707482393
4	-12.7016085609462	-12.9913618045111
5	-12.6459603931509	-12.7396559288655
6	-12.5312488366447	-12.8786041307829
7	-12.3514390385213	-12.4836127397469
8	-11.6630028062520	-12.0705789781940
9	-11.5055222310041	-11.0319997562342
10	-10.5338593930750	-10.1303811459310

表 4.6　分極モデル, 非対称アミンの時のベンゼン中での OH^- の水和クラスターの自由エネルギー $\Delta G_{n_{\mathrm{w}}} - n_{\mathrm{w}}\mu$

PTC	$n_{\mathrm{w,Average}}$ (非分極)	$n_{\mathrm{w,Average}}$ (分極)
PTC 無し	5.1/4.5	7.1/6.9
対称アミン	3.1/2.4	4.2/5.3
非対称アミン	3.1/3.0	5.4/5.4

表 4.7　平均水和数のまとめ (298.15K/348.15K)

4.2　反応

アリルベンゼン異性化反応の反応は図 4.2 上段の 2 段階で起こると考えられる。また一段階で起こる図 4.2 下段の反応機構も考えられる。またこの反応に水和水 n_{w} 個が OH^- に付き反応はさらに複雑となる。

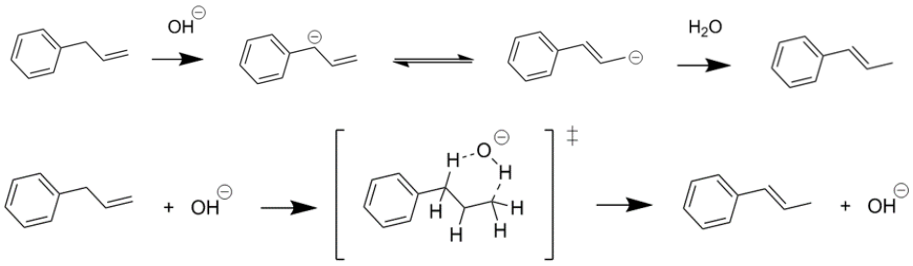


図 4.19　アリルベンゼン異性化反応の予想される反応機構

4.2.1　量子化学計算の精度

実験と比較するために必要な基底関数のレベルを評価するためにいくつかの条件で $\mathrm{p}K_{\mathrm{a}}$ を見積もり実験値と比較することにした。酸 AH が溶媒バルク中で化学式 4.5 の解離平衡にある時、平衡定数は式 4.6 で表される。

$$[\mathrm{HA}] + [\mathrm{SOLVENT}] \rightleftharpoons [\mathrm{A}^-] + [\mathrm{HSOLVENT}^+] \tag{4.5}$$

$$K = \frac{[\mathrm{A}^-][\mathrm{HSOLVENT}^+]}{[\mathrm{HA}][\mathrm{SOLVENT}]} \tag{4.6}$$

このうち、溶媒の水は濃度一定のため定数に含めた形にした酸解離定数 K_{a} は式 4.8 で定義される。

$$K_{\mathrm{a}} = \frac{[\mathrm{A}^-][\mathrm{HSOLVENT}^+]}{[\mathrm{HA}]} \tag{4.7}$$

$$= K[\mathrm{SOLVENT}] \tag{4.8}$$

これを対数で扱った $\mathrm{p}K_{\mathrm{a}}$ は、式 4.12 から計算することができる。

$$\Delta G = -RT \ln K \tag{4.9}$$

$$= -RT \ln K_{\mathrm{a}} + RT \ln [\mathrm{SOLVENT}] \tag{4.10}$$

$$\mathrm{p}K_{\mathrm{a}} = -\log_{10} K_{\mathrm{a}} \tag{4.11}$$

$$= \frac{\Delta G}{2.303RT} - \log [\mathrm{SOLVENT}] \tag{4.12}$$

プロトンの解離の精度を検討するために、実験値のある DMSO 中でのアリルベンゼン (allylbenzene+DMSO=allylbenzene$^-$+HDM
と $\mathrm{H_2O}$($\mathrm{H_2O}^+$DMSO=$\mathrm{OH}^-{}^+$HDMSO$^+$+ΔG) について $\mathrm{p}K_{\mathrm{a}}$ を見積もり、表 4.8〜4.12 にまとめた。

B3LYP/6-31+G(d) で GRRM を実施して、得られた構造の中で重要な構造のみ B3LYP/6-311++G(d,p) で再度 Opt してエネルギーを求めることにした。

酸	pKa(Exp.)	ΔG(Exp.)
allylbenzene	34	49.3
H_2O	31.36	45.7

表 4.8　実験の DMSO 中でのアリルベンゼンと H_2O の pK_a

酸	pKa(Calc.)	ΔG(Calc.)	ΔG(Calc.)-ΔG(Exp.)
allylbenzene	42.6	49.3	11.7
H_2O	35.8	45.7	6.1

表 4.9　B3LYP/6-31++G(d) レベルでの DMSO 中でのアリルベンゼンと H_2O の pK_a

酸	pKa(Calc.)	ΔG(Calc.)	ΔG(Calc.)-ΔG(Exp.)
allylbenzene	41.6	59.7	10.3
H_2O	35.9	51.9	6.2

表 4.10　B3LYP/6-31+G(d) レベルでの DMSO 中でのアリルベンゼンと H_2O の pK_a

酸	pKa(Calc.)	ΔG(Calc.)	ΔG(Calc.)-ΔG(Exp.)
allylbenzene	39.2	49.3	7.1
H_2O	36.4	45.7	6.8

表 4.11　B3LYP/aug-cc-pVTZ レベルでの DMSO 中でのアリルベンゼンと H_2O の pK_a

酸	pKa(Calc.)	ΔG(Calc.)	ΔG(Calc.)-ΔG(Exp.)
allylbenzene	39.0	56.1	6.8
H_2O	35.5	51.4	5.7

表 4.12　B3LYP/6-311++G(d,p) レベルでの DMSO 中でのアリルベンゼンと H_2O の pK_a

4.2.2　遷移状態探索

反応経路がいくつか予想されるため、反応は化学反応経路自動探索プログラム (GRRM)[49] を用いて、B3LYP/6-31+G(d) レベルで、溶媒効果は PCM を用いて Benzene($\varepsilon_r = 2.247$) 中での反応を想定して実施した。OH^- に水和する水和数 $n_w = 0$ 3 の場合についてそれぞれアリルベンゼンと OH^- のプロトン引き抜きが起こりそうな初期構造を用意し、MIN で構造最適化した後に、解離判定に関する UpDC と DownDC オプションをそれぞれ 15 と 13 に設定し Firstonly 探索を行った。得られた遷移状態 (TS) と極小点 (EQ) の個数を表 4.13 に示す。

得られた全ての TS に対して振動数計算を行い、プロトン引き抜きが起こっている遷移状態を抽出した。$n_w = 0$(1 段階目)、 $n_w = 2$(2 段階目) については GRRM で遷移状態が得られなかったため、Gaussian09 で $n_w = 0$(1 段階目) は Opt=(QST3,redundant)、$n_w = 2$(2 段階目) は Opt=(ts,noeigentest,calcfc) によって計算した。得られたアリルベンゼン異性化反応のエネルギーダイアグラムを図 4.14 にまとめた。反応物 R と生成物 P は Gaussian09 でアリル

n_{w}	TS	EQ
0	15	24
1	13	23
2	11	29
3	12	28

表 4.13　水和数 n_{w} ごとの得られた遷移状態と極小点の個数 (B3LYP/6-31+G(d))

ベンゼンと水和クラスターを別々に構造最適化して求めた無限遠でのエネルギーで、反応物 R を基準としてプロットした。

n_{w}	E_{active} [kcal/mol]
0	0
1	4.0
2	9.3
3	16.3
水一分子	53.1

表 4.14　OH^- ($n_{\mathrm{w}}H_2O$) または水一分子によって起こるアリルベンゼン異性化反応の活性化障壁

また、OH^- ではなく水一分子によってベンゼン中で反応が進む場合の活性化障壁も調べたが、53.1 kcal/mol と非常に高く、水では反応は起こらないことがわかった。これまでに OH^- による反応は遷移状態が 2 つある 2 段階の反応しか得られていないが、水一分子による反応では図 4.2 下段のような一段階の反応が得られた。水和数が増えるにつれて活性化障壁が高くなり、反応は不利になった。

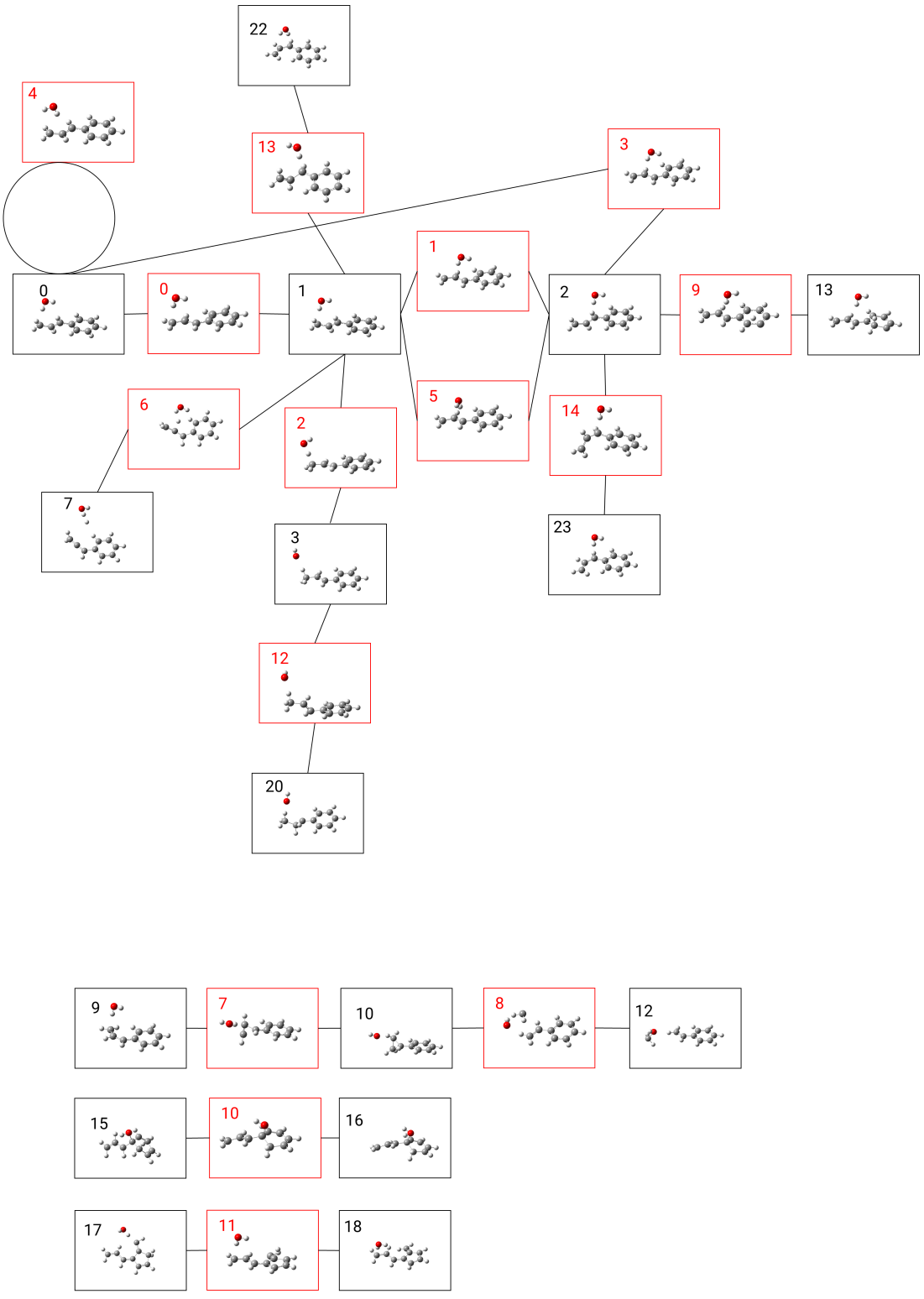


図 4.20　$n_{\mathrm{w}} = 0$ の反応経路。黒は EQ を、赤は TS を表す。TS まわりの EQ のみをマップに記載している。アリルベンゼン異性化反応の TS は赤 2(2 段階目)。

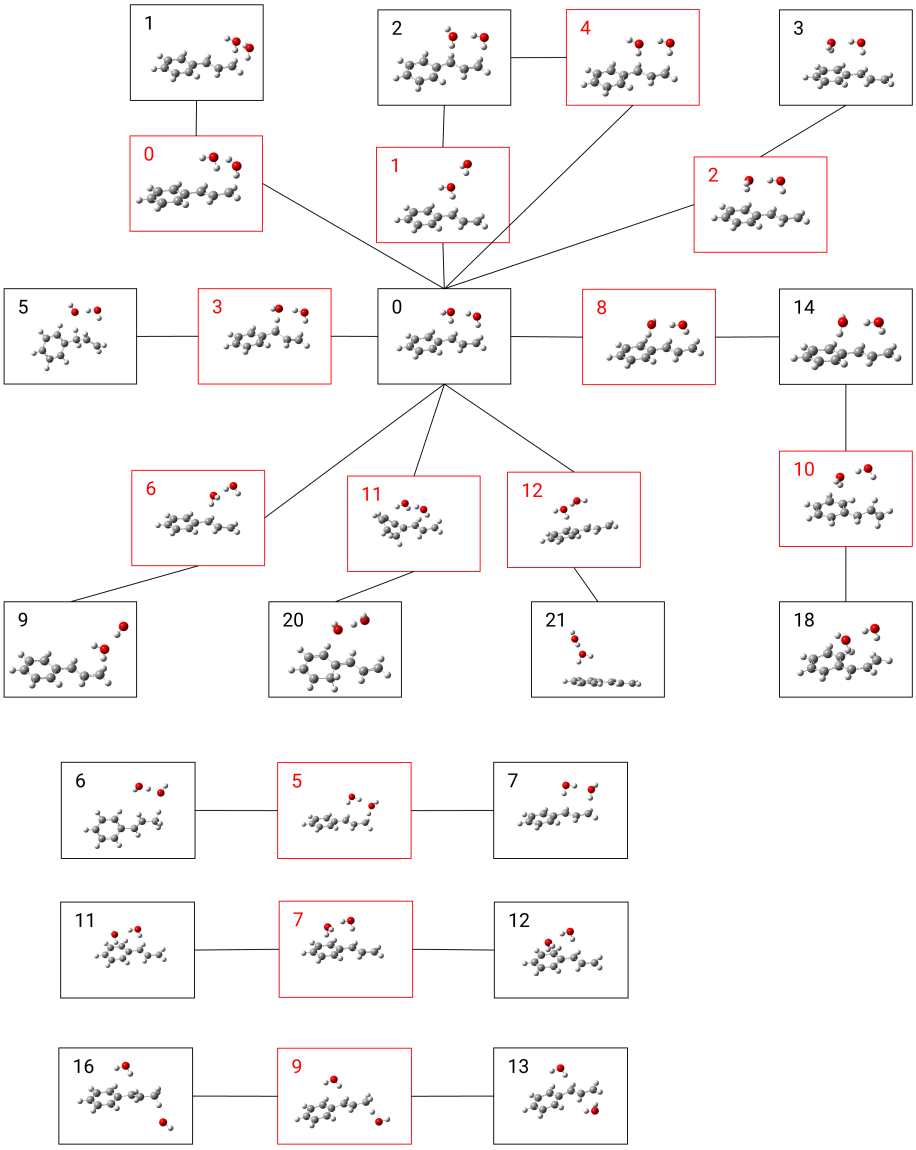


図 4.21　$n_{\mathrm{w}} = 1$ の反応経路。黒は EQ を、赤は TS を表す。TS まわりの EQ のみをマップに記載している。アリルベンゼン異性化反応の TS は赤 3(1 段階目) と赤 5(2 段階目) と赤 9(2 段階目)。

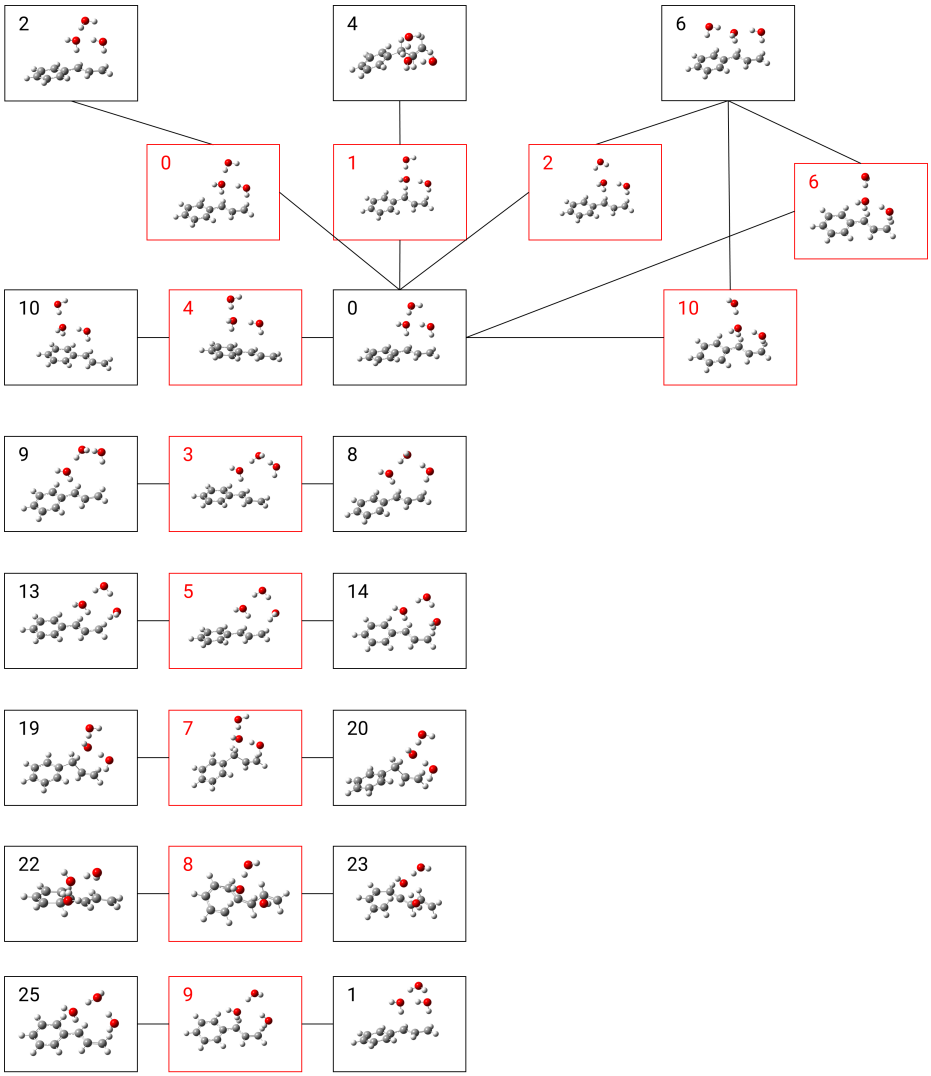


図 4.22　$n_w = 2$ の反応経路。黒は EQ を、赤は TS を表す。TS まわりの EQ のみをマップに記載している。アリルベンゼン異性化反応の TS は赤 1(1 段階目)。

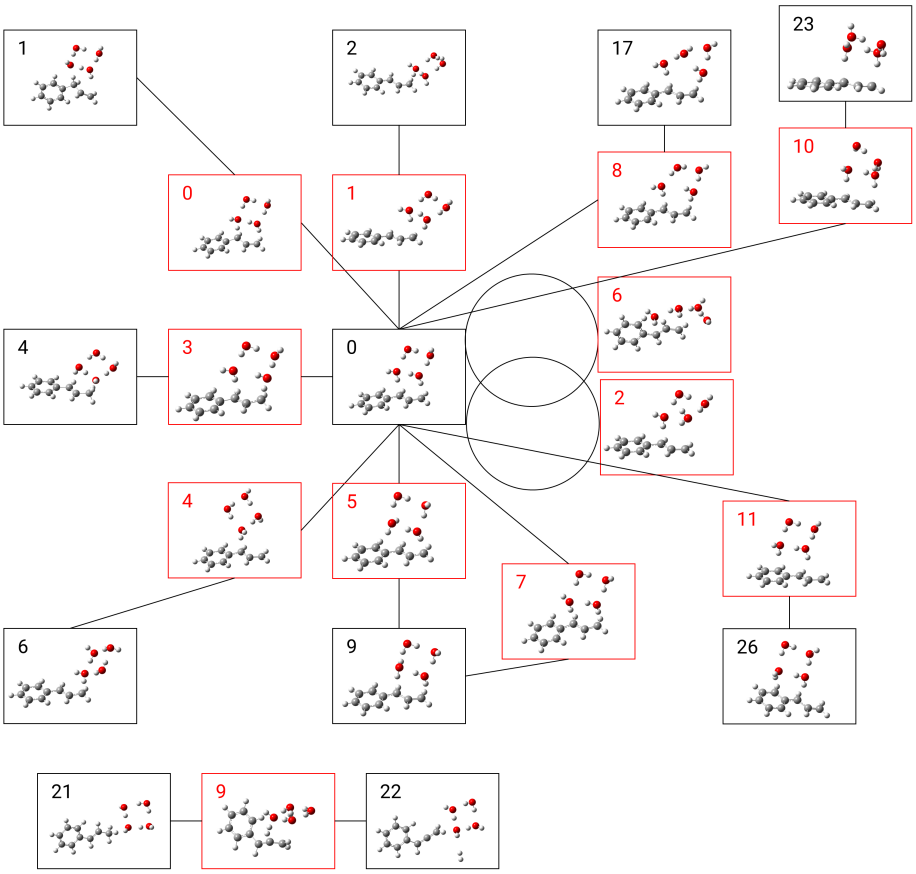


図 4.23　$n_{\mathrm{w}} = 3$ の反応経路。黒は EQ を、赤は TS を表す。TS まわりの EQ のみをマップに記載している。アリルベンゼン異性化反応の TS は赤 0(1 段階目) と赤 3(2 段階目)。

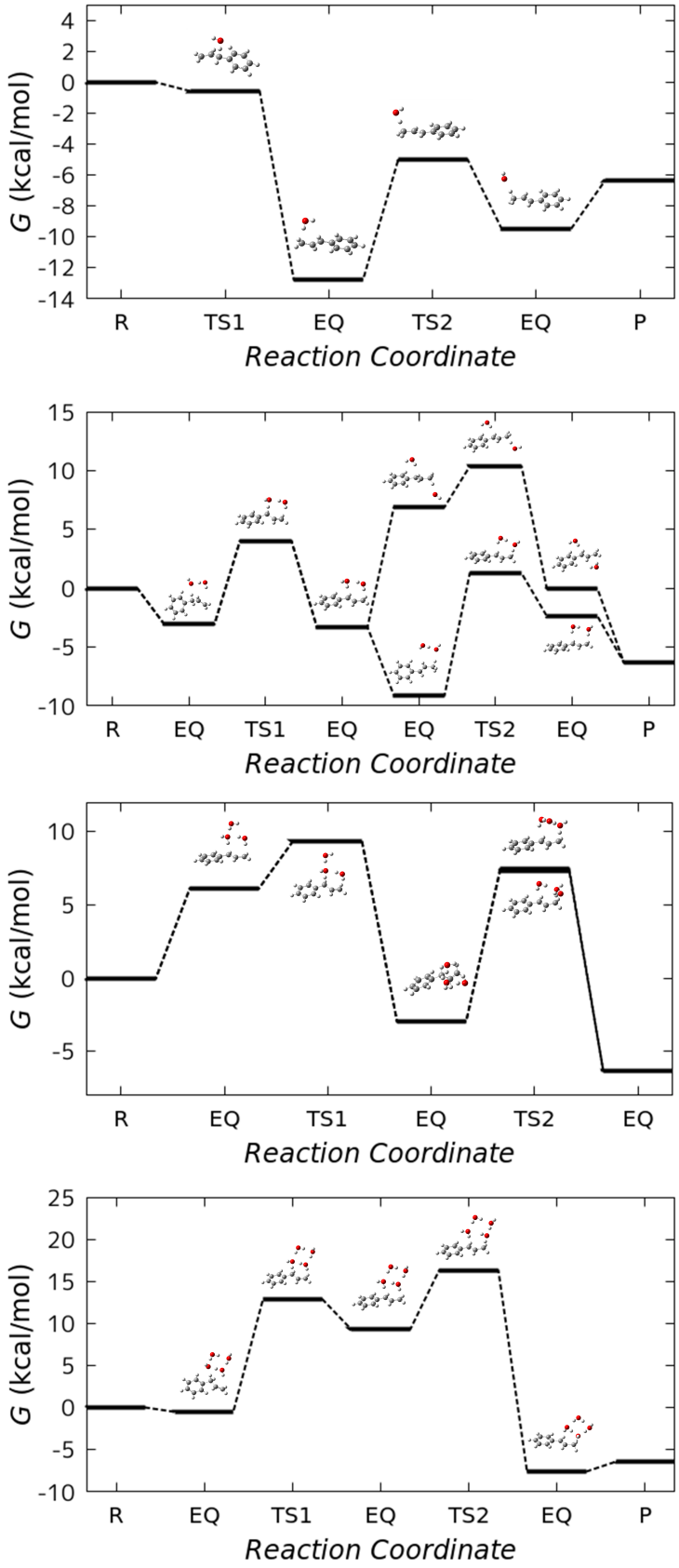


図 4.24　上から順に n_{w} =0, 1, 2, 3 の時のエネルギーダイアグラム

4.3　反応機構解析

ここまでに得られた非分極での { 輸送, 水和, 反応 } のエネルギーを足し合わせた総障壁を図 4.3 に示す。黒は OH^- のみで反応が進行した場合、青は対称 PTC がある場合にで進行した場合であり、それぞれについて、水和数 n_w =0〜2 を比較した。Sym と Asym については差が見られなかった。まず、図 4 に示す PTC の親油性が足りない時には実験的にも反応が進まない事実と一致する。特に障壁を高くしている原因として、輸送が難しいことがわかる。PTC がある場合には、イオン輸送が大きく安定化されている。実験でアレニウスプロットによって見積もられている活性化障壁 17 kcal/mol と比較すると、PTC がある場合には $n_w = 0, 1$ で反応が進行していると考えられる。水和数分布のピークは Asym で n_w=2, 3 あたりにあることから、水和数分布のピークから外れた分布の低い水和数で反応が進行していることになる。また水色のラインは界面通過時のバリアの高さを表しているが、n_w =0,1 の総障壁の高さよりも低いため、反応の律速段階は油相での反応になり、バリアは反応に影響していないと考えている。

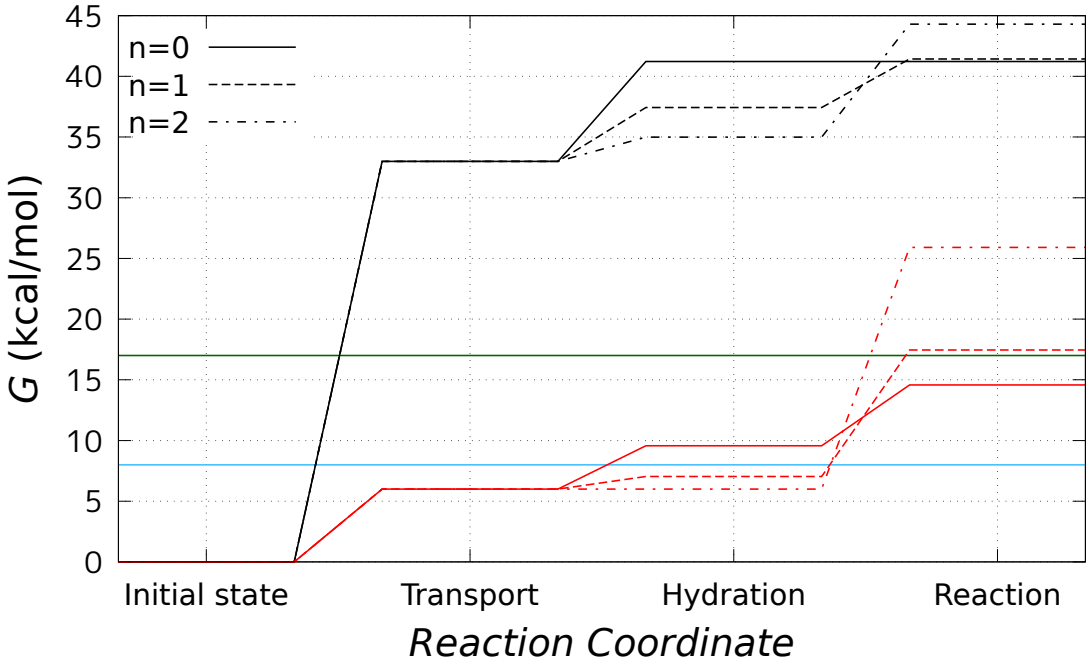


図 4.25　アリルベンゼン異性化反応のイオン輸送、水和、反応から成る総障壁の without と Sym の比較。

第 5 章

油水共存系のためのジクロロメタンの分子モデリング

先行研究で Lingjian Wan らによるジクロロメタン（DCM）中での Cl^- 水和数分布の見積もりが古典分子動力学シミュレーション (MD) で行われていた。[9] しかしその結果は実験での見積もりである 2、3 個を大きく下回り DCM 中で 0 個というものであった。そこで水和数の計算について検討を行い、ジクロロメタンの分子モデルの精度には問題があることがわかり、油相に溶けたイオンの水和数や油水界面など油水共存系に対して実験との比較に耐える分子モデルの開発を行った。既存の DCM 分子のモデルは Dang によって作成されていた [50] が、MD のためのジクロロメタン (DCM) 分子モデルについて、レナード・ジョーンズパラメータ ε と σ のパラメータフィッテングが実施され、油水界面などの油水共存系に適したモデルが作成された。パラメータは DCM 中に水分子 1 個を含む系の第一原理 MD シミュレーション (AIMD) によって計算された動径分布関数 (RDF) と MD によって計算される RDF の差の二乗が最小になるように決定された。得られたパラメータは DCM 中での溶媒和自由エネルギー G_{solv} と水/DCM 界面の界面張力 γ について物性値が確認され、実験値をよく再現した。本研究で複数の分子が存在する系 (界面系や混合系等) に用いる分子モデルのパラメータは AIMD の RDF を指針にして決定することでリーズナブルな物性値を記述することができるという界面系での分子モデリングの方針を示した。さらに作成した分子モデルのパフォーマンスを調べるために、DCM 中での Cl^- イオンの平衡水和数分布について調べた。

5.1　バルクの物性

先行研究でジクロロメタン（DCM）中での Cl^- 水和数分布の見積もりが古典分子動力学シミュレーション (MD) で行われていた。[9] しかしその結果は実験での見積もった結果の水和数 2、3 個を大きく下回り DCM 中で水和数 0 個というものであった。そこで水和数の計算について検討を行い、DCM の分子モデルの精度には問題があることがわかり、油相に溶けたイオンの水和数や油水界面など油水共存系に対して実験との比較に耐える分子モデルの開発を行った。この章ではまず DCM のバルクの物性について確認し、微調整を加えた。

■1 対 1 ポテンシャル　まず、DCM-DCM 間の 1 対 1 ポテンシャルが量子化学計算 (QM) と MD で比較され正しく記述できているか確認された。図 5.1 に、4 種類の配向の異なる DCM-DCM 間のポテンシャルを示す。4 種類の配向を以下に示す。

1. C-Cl—H-C を直線拘束して C-C 間距離 r の各点で構造最適化
2. C-Cl—Cl-C を直線拘束して C-C 間距離 r の各点で構造最適化
3. C-H—H-C を直線拘束して C-C 間距離 r の各点で構造最適化

4. 拘束なしで C-C 間距離 r の各点で構造最適化

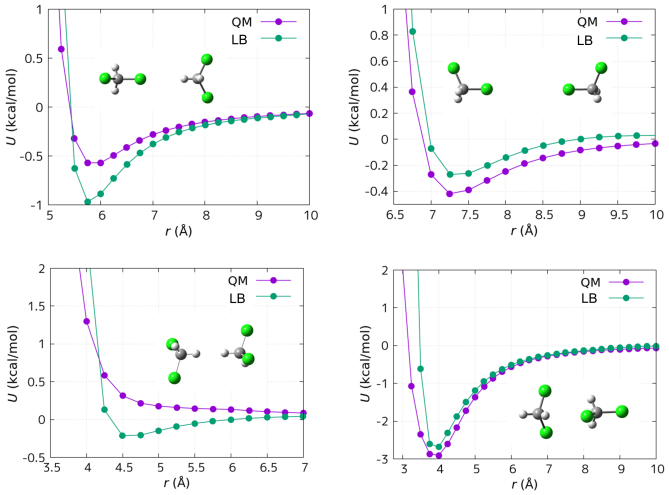


図 5.1　DCM-DCM 間の 1 対 1 ポテンシャル面。緑が Cl^- 原子、灰色が C 原子、白が H 原子を表す。上段から順に配向 1〜4 に対応する。

全ての配向で 2 つの DCM の炭素間距離 r_{C-C} で距離を拘束して B3LYP/aug-cc-pVDZ レベルで構造最適化し、各点で得られた構造に対して CCSD(T)/aug-cc-pVDZ レベルで一点計算をしてポテンシャル面を得た。また BSSE は Counterpoise 法で補正しており、4 番の配向の安定点付近では 2.6 kcal/mol の BSSE が含まれていた。実際には直線拘束と言っているのは、例えば配向 1 番の場合では C-Cl—H の角度を 179˚、 Cl—H-C の角度を 179˚というように厳密には 180˚ではなく、 Gaussian09 で直線拘束ができないことから計算上の都合でわずかにずらしている。また MD でも同様の構造 (1〜4) の条件で計算しポテンシャル面を得た。この時 MD では Lorentz-Berthelot 混合則 (LB) を用いている。ポテンシャル面に関しては 4 つの配向について調べた結果、既存のモデルで LB を用いた時に QM のポテンシャル面をよく記述することが確認された。

■誘電率　続いて DCM バルクの誘電率を調べ、誘電率を改良するために分極率 α について調整を加えた。誘電率は誘電体に電場を印加してその物質の応答によって定まる。誘電率は次式で求めた。

$$\varepsilon_{\mathrm{r}} = \frac{E_z}{\varepsilon_0 E_z - \frac{\langle M_z \rangle}{V}} \tag{5.1}$$

E_z が印加した電場 $\langle M_z \rangle$ は双極子モーメントの z 成分の統計平均、V はシミュレーションセルの体積である。シミュレーションセルは $L_x \times L_y \times L_z = 30$ Å $\times 30$ Å $\times 30$ Å で、254 個の DCM を含む。また確認のため POL3 水の誘電率を求め 125 となり、正しく計算できていることを確認できた。[51] Dang のモデルで計算された誘電率は 2 つの原因があり実験値を再現しなかった。1 つ目は誤植によってサイトの電荷が間違っていたことが大きな原因であった。Dang の論文 [50] では H と Cl のサイト電荷が逆転していたことで電荷逆転モデルでは本来より極性が大きくなって

いたため誘電率が合わなかった。この誤植によって本来よりも DCM 中でイオンが大きく安定化され Cl^- の水和数予測の結果が DCM 中で 0 個という結果になっていた。[9] しかし、その修正をしても分極率の与え方には修正の余地があった。2 つ目の原因は分極率が小さ過ぎることだった。Dang のモデルと真空中での実験値の分極率の値はそれぞれ 4.968 と 6.48[52] であった。溶媒中では分子の分極率は真空中と比較して小さくなる効果があり、中性分子では溶液中で 9% スケールされることが報告されている。[53] 溶媒中での分子の分極率は減る傾向にあるが、Dang のモデルでは小さ過ぎた。Dang の使用したサイト分極率は Applequist の論文 [54] から引用したものであるが、サイトにおける等方的な分極を記述する式に interaction モデルのパラメータを当てはめたものとなっていた。このため、分子の分極率が小さくなり過ぎていた。したがって、interaction モデルのパラメータではなく additive モデルのパラメータ (加算して分子の分極率を再現する) をこの場合には適用する方がよいと思われる。今回は additive パラメータを 9% スケールしてサイトの分極率を得た。改良後のサイトごとの DCM のパラメータは表 5.2 にまとめた。また改良前後の誘電率の値を表 5.2 にまとめた。改良後では実験値をよく再現することが確認できた。

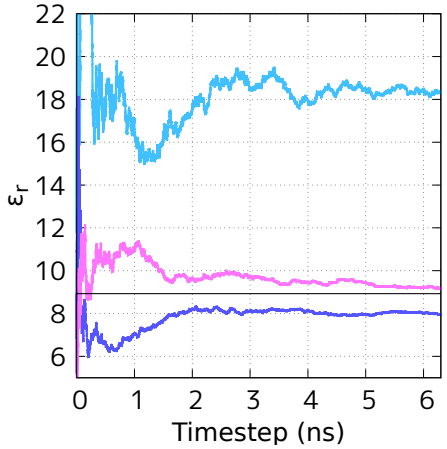


図 5.2　改良前後の DCM{typo (水色)、old (青)、new (ピンク)} と実験値 (黒) の比較。

model	polarizability	Dielectric constant
Dang's model	4.968	18.0(typo), 7.97(correct)
Improved model	6.00	9.18
Exp.	6.48	8.93

表 5.1　分極率と誘電率の関係

site	q [e]	α [Å^3]	σ [Å]	ε [kcal/mol]
C	-0.2720	0.951	3.41	0.137
Cl	-0.0537	2.148	3.45	0.280
H	0.1897	0.377	2.40	0.040

表 5.2　分極率について修正された DCM のパラメータ

■蒸発エンタルピー　続いて、蒸発エンタルピー ΔH_{vap} が式 5.2 にしたがって調べられた。

$$\Delta H_{\mathrm{vap}}(T) = -\langle U(T)\rangle + RT \tag{5.2}$$

結果は $\Delta H_{\mathrm{vap}} = 6.79$ kcal/mol で実験値の 6.7 kcal/mol とよく一致した。[52]

以上の結果から、DCM のバルクの物性値はよく記述できていることが確認された。

5.2　油水共存系のための分子モデル改良

これまでの分子モデルは蒸発エンタルピーなどのあるバルク物性の実験値を再現するようにして作成されてきた。そして不均一系では LJ パラメータ σ, ε は Lorentz-Berthelot 混合則 (LB) が適用されてきた。多くの場合において、LB 則はリーズナブルな結果を与えるが、不均一な混合系や界面系などでは精度が不足していた。水/DCM 界面の場合では、水と DCM はそのバルクについては十分な精度で記述できるが LB を適用して計算された DCM 中に水分子が溶ける溶媒和自由エネルギーや水/DCM 界面の界面張力についてはあまりよく記述できていなかった。本研究では水分子と DCM 分子の間の LJ パラメータについて改良を加えて油水共存系に適したモデルを作成した。改良は DCM バルクに水 1 分子存在する系について MD で計算された RDF が第一原理分子動力学シミュレーション (AIMD) によって計算された動径分布関数 (RDF) に合うように LJ パラメータについて行われた。この章ではまず水-DCM 間の 1 対 1 ポテンシャルを量子化学計算 (QM) と LB を適用した MD で調べた結果から安定化が不足していることを示し、次に DCM 中に水一分子が溶ける溶媒和自由エネルギーと界面張力の物性を確認した結果を示し、最後に分子モデル改良とそのパフォーマンスの順に説明する。

■1 対 1 ポテンシャル　水-DCM 間の 1 対 1 ポテンシャルが量子化学計算 (QM) と MD で比較され正しく記述できているか確認された。図 5.3 に、5 種類の配向の異なる水-DCM 間のポテンシャルを示す。5 種類の配向を以下に示す。

a. C-Cl—HW-OW を直線拘束して C-OW 間距離 r の各点で構造最適化
b. C-Cl—OW-HW を直線拘束して C-OW 間距離 r の各点で構造最適化
c. C-H—OW-HW を直線拘束して C-OW 間距離 r の各点で構造最適化
d. C-H—HW-OW を直線拘束して C-OW 間距離 r の各点で構造最適化
e. 拘束なしで C-OW 間距離 r の各点で構造最適化

全ての配向で 2 つの DCM の炭素間距離 $r_{\mathrm{C-OW}}$ で距離を拘束して B3LYP/aug-cc-pVDZ レベルで構造最適化し、各点で得られた構造に対して CCSD(T)/aug-cc-pVDZ レベルで一点計算をしてポテンシャル面を得た。また BSSE は Counterpoise 法で補正した。また MD では LB を用いている。図 5.3 中央のポテンシャルに注目すると、a,b,d の配向は QM と LB の差が小さいが、特に (e) は、QM と LB の差が大きい。この差について調べるために北浦・諸隈エネルギー分解 [55] が GAMESS[56] を用いて RHF/6-311+G(d,p) レベルで実施された。結果は図 5.3 の右列に示した。相互作用は静電相互作用エネルギー (ES)、交換反発エネルギー (EX), 分極エネルギー (PL), 電荷移動エネルギー (CT)、更にこれら 4 つの摂動の高次の項 (MIX) に分解される。MIX の値は小さい。図 5.3(e) 右列に注目すると、図 5.3(e) 中央のポテンシャル (LB) の安定点での安定化の不足は CT で説明できる。MD では CT の効果は考慮していない為、LJ パラメータに CT の効果を加えることにした。また 1 対 1 ポテンシャルだけではなく確認のために DCM 中に水一分子が溶けた時の溶媒和自由エネルギー ΔG_{solv} と界面張力を調べた。

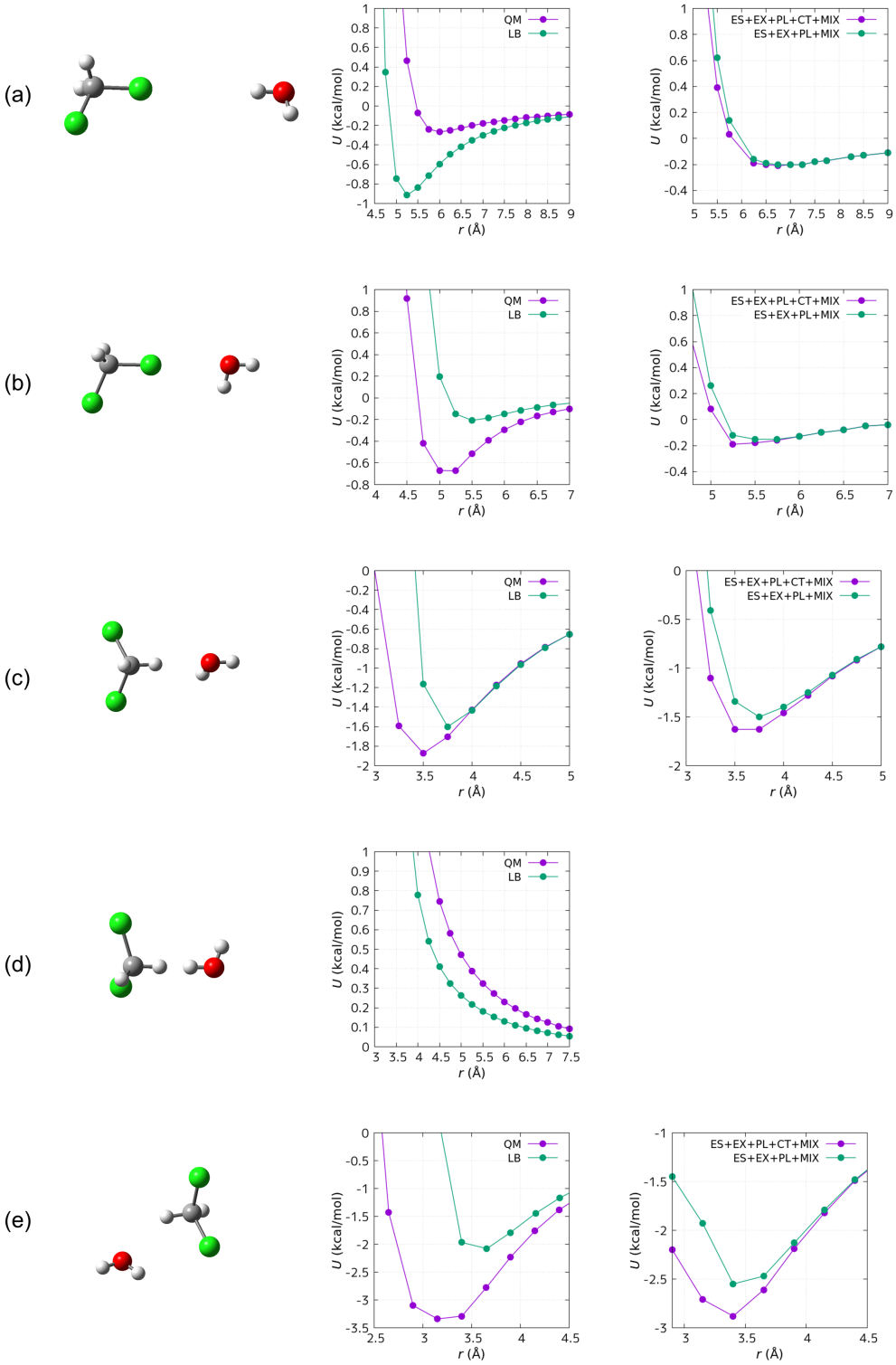


図 5.3　水-DCM 間の配向 (左列) とそれに対応する 1 対 1 ポテンシャル面 (中央列)。緑が Cl^- 原子、灰色が C 原子、白が H 原子、赤が O 原子を表す。上段から順に配置 a〜e に対応する。また右列はそれぞれの配向でのエネルギー分解後の全エネルギー ES + EX + PL + CT + MIX(紫) と CT だけ除いたもの ES + EX + PL + MIX(緑) の比較。

■溶媒和自由エネルギー　溶媒和自由エネルギーは 2 章の 2.3 の方法で実験値または MD で見積もった。MD は DCM 中に水 1 分子を溶かした時の溶媒和自由エネルギー μ^* を計算した。カップリングパラメータ λ は”0, 0.000001, 0.00001, 0.0001, 0.0005 , 0.001, 0.01, 0.1, 0.333, 0.666, 0.833, 1” で 0 から 1 まで変化させ真空中から DCM 中に水一分子が溶ける時の溶媒和自由エネルギーを計算した。セルサイズは $L_x \times L_y \times L_z = 30$ Å $\times 30$ Å $\times 30$ Å で 1 個の水と 254 個の DCM を含む系で平衡化 50ps の後サンプリング 50ps 実施した。LJ パラメータは LB 則を適用した。計算結果は表 5.3 にまとめた。

表 5.3　溶媒和自由エネルギーの実験値と MD の結果の比

	G_{solv} [kcal/mol]
LB	-1.10 ± 0.39
Exp.	-2.73

1 対 1 ポテンシャルで不足していた安定性が溶媒和自由エネルギーからも確認できた。

■界面張力　水/DCM 界面の界面張力 γ はセルサイズ $L_x \times L_y \times L_z = 25$ Å $\times 25$ Å $\times 85$ Å で 523 個の水と 353 個の DCM から成る z 軸方向に沿った水/DCM のスラブ系で計算された。界面の厚みは z 軸方向に水相が 25 Å で DCM 相が 60 Å である。界面張力 γ は圧力テンソル P から計算された。

$$\gamma = \frac{1}{2} L_z \left[P_{zz} - \frac{1}{2}(P_{xx} + P_{yy}) \right]. \tag{5.3}$$

表 5.4　water/DCM 界面の界面張力の実験値と MD の結果の比較

	γ [mN/m]
LB	36.5 ± 0.5
Exp.	27.93 ± 0.03[57]

界面張力の結果はそこまで悪くないが、図 5.3(e) での水-DCM 間の安定化が足りないことが溶媒和自由エネルギーの結果からも確認できる。界面張力の場合には界面での分子の配向の影響もあるため、溶媒和自由エネルギーと界面張力の 2 つの物性で確認することにする。

■動径分布関数 (RDF) を指針としたパラメータフィッティング　(1)「多くの配置でのポテンシャルをフィッテングに含める」また、1 対 1 ポテンシャルが多体系でも成り立つのかという問題を解決するために、(2)「凝集系での動径分布関数をフィッテングの条件に含める」ことにして、(1)，(2) を同時に満たす「第一原理分子動力学で計算した DCM 中の水の動径分布関数 (RDF)」を指針として LJ のパラメータフィッテングを行った。RDF を参照にした LJ パラメータフィッテングの方法とその結果得られたパラメータを示し、そして改良前後のパラメータとそのパラメータを用いて計算されたいくつかの共存系の物性について検証結果を示す。動径分布関数 (RDF) は次の形で書ける。

$$RDF(r) = \int_0^\infty 4\pi r^2 g_{\mathrm{ab}}(r) dr \tag{5.4}$$

$$g_{\mathrm{ab}}(r) = \frac{\rho_{\mathrm{ab}}(r) dr}{N_{\mathrm{a}} N_{\mathrm{b}}}. \tag{5.5}$$

RDF について、第一原理分子動力学シミュレーション (AIMD) で計算したものを参照し LJ パラメータを決定する。まず AIMD の計算条件について記す。AIMD は CP2K の Quickstep を用いて、NVT アンサンブルで時間刻み

Δt が 0.5 fs、温度は $T = 298.15$ K で Nosé-Hoover thermostats で制御された。PBE 関数でダブルゼータ分極基底 (DZVP)、Goedecker–Teter–Hutter 擬ポテンシャル (GTH) で計算された。平面波エネルギーカットオフは 280 Ry に設定された。セルサイズは (30Å, 30Å, 30Å) で系には 254 個の DCM と 1 個の水が含まれる。異なるシードを与えた複数の系を初期構造にしてサンプリングされた。最初に MD で平衡化 50ps、続いて 2 ps DZVP-GTH で平衡化の後、サンプリング 43.7 ps が実施された。また DZVP-MOLOPT-GTH (valence double-zeta plus polarization, molecularly optimized, Goedecker–Teter–Hutter) 基底系でも計算し、比較された。どちらの条件でも同じ形を与えることが確認され、より計算コストの低い DZVP-GTH の結果をフィッテングの条件に使用することにした。

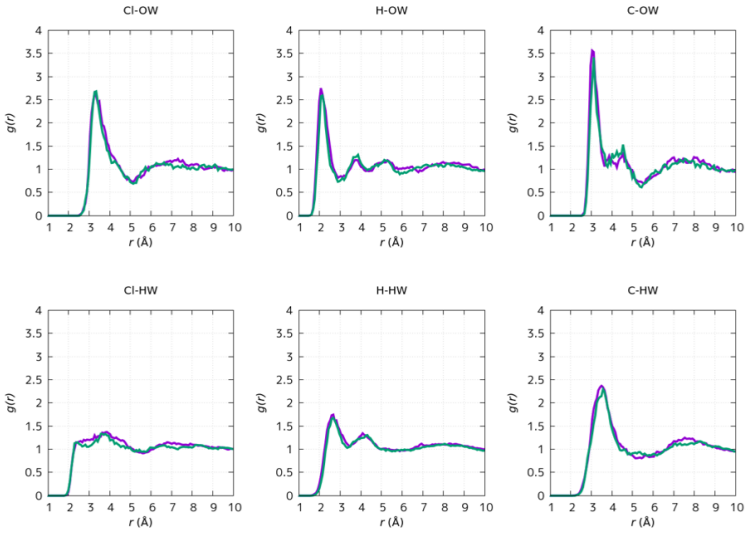


図 5.4　DZVP-GTH (紫) と DZVP-MOLOPT-GTH (緑) で計算された RDF

フィッテングする変数は C-OW, Cl-OW, H-OW の 3 つのペアに関する σ, ε で、C-HW, Cl-HW, H-HW に関しては値が小さく影響が小さいため無視した。3 つのペアに関して MD と AIMD の RDF の差 ($_D eltaRDF$) の二乗が最小になるようにパラメータの最適化を実施した。図 5.5 に示すように、ΔRDF は AIMD で計算された RDF と MD で計算された RDF に囲まれた面積として定義された。

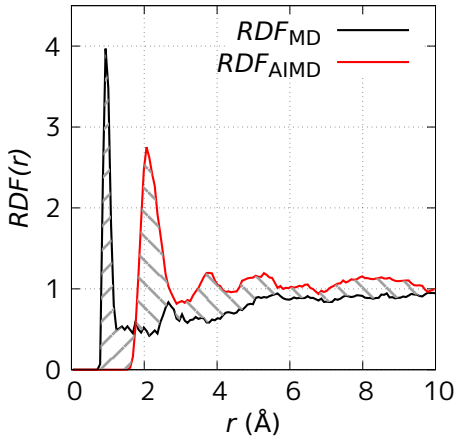


図 5.5　RDF(AIMD) と RDF(MD) の間の面積 (影の部分) で表される ΔRDF

ローレンツベルテロ則 (LB) によるパラメータと差の二乗が最小のパラメータのセットを表 5.2 にまとめた。

LB	σ [Å]	ε [kcal/mol]
C-OW	3.3070	0.14619
Cl-OW	3.3270	0.20899
H-OW	2.8020	0.07899
min	σ [Å]	ε [kcal/mol]
C-OW	3.0570	0.18273
Cl-OW	3.3270	0.15674
H-OW	2.3020	0.05924

表 5.5　ローレンツベルテロ則に従って決定された LJ パラメータと動径分布関数を指針にして決定された LJ パラメータの比較

σ だけに注目すると Cl-OW に関しては min でも LB と変わっておらず、C-OW, H-OW が小さくなっている。σ が小さくなるということはその分斥力が働き始める距離が小さくなるため、引力的になる。5.6 に示すように、C-OW, Cl-OW, H-OW 間の LJ ポテンシャルの LB→min の変化を見た。

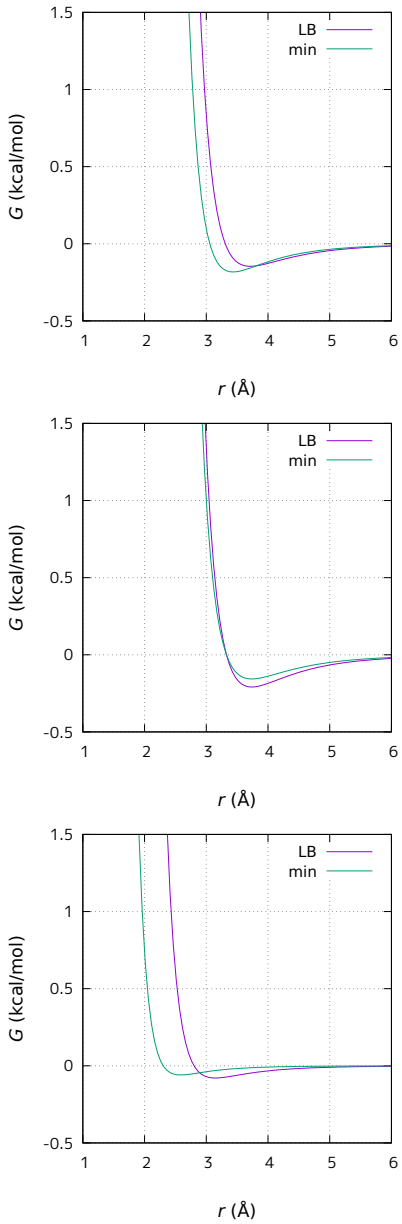


図 5.6　上から順に C-OW, Cl-OW, H-OW 間の LJ ポテンシャルの LB から min の変化

最も変化が大きいのが H-OW で次に C-OW で Cl-OW に関してはほとんど変化がない。次に、得られたパラメータの LB からの差が溶媒和自由エネルギーや界面張力を重水素化した系にて以下の 5 種類のパラメータで比較を行った。

1. LB
2. test1(C-OW のみ LB , 他は min)
3. test2(Cl-OW のみ LB, 他は min)
4. test3(H-OW のみ LB, 他は min)
5. min

これらのパラメータを用いて計算した結果を表 5.6 にまとめた。LB では溶媒和自由エネルギー ΔG_{solv} が実験値と比

parameter	ΔG_{solv} [kcal/mol]	γ [mN/m]
1. LB	-0.75(±0.21)	35.93(±0.54)
2. test1	-2.97(±0.29)	38.60(±0.60)
3. test2	-2.35(±0.27)	28.52(±0.80)
4. test3	-1.95(±0.18)	42.45(±0.69)
5. min	-3.67(±0.40)	33.83(±0.56)
Exp.	-2.73	27.93 ± 0.03 [57]

表 5.6　溶媒和自由エネルギーと界面張力に対する LB, test1, test2, test3, min の比較

較して安定化が大きく不足しているが、AIMD の RDF を目的にしたパラメータフィッテングによって得られた min では溶媒和自由エネルギーと界面張力のどちらも良好に記述できることが確認できた。さらにこの安定化のペアごとの内訳を調べるために、test1〜3 の物性を調べた。test1 は DCM の中心の C と OW のペアであること、test2 はパラメータ変化が最も小さいことから、これらのパラメータ変化による影響が小さいことが理解できる。H-OW 間のみ LB にした test3 では溶媒和自由エネルギー G_{solv} を min と比較すると最も安定化が損なわれているため、-OW 間の LB からのパラメータ変化が最も安定化に寄与していることが理解できる。元々のモデルで不足していた電荷移動による安定化分は H-OW 間のパラメータ変化によって記述することができたことが確認できた。

5.3　水和数分布

油相に存在する親水性のイオンは水和水を持ち水和クラスターとなりやすい。計算方法は [9] に記された方法で、簡単に 2 章の 2.4.1 に記した。シミュレーションセルは $L_x \times L_y \times L_z = 30$ Å $\times 30$ Å $\times 30$ Å で、1 個の Cl^- と n 個の水と $255 - n$ 個の DCM から成る。水和数 $n = 4$ の時を図 5.7 に例として載せる。$n - 1$ 個から n 個の水和数変化の水和自由エネルギー $\Delta G_{(n-1,n)}$ は 2 章の 2.3 の自由エネルギー摂動法で計算された。また $\lambda = 0$ 付近では n 個目の水和水が水和クラスターに付いた状態でのサンプリングが難しくなるため、n 個目の水和水が Cl^- から離れないように $\lambda = 0$ から $\lambda = 1$ に近づくにつれてバイアスがゆるくなる次のバイアスポテンシャルを n 個目の水にかけた。

$$V_{\mathrm{bias}} = (1-\lambda)10k_{\mathrm{B}}T(\frac{|r-r_0|}{r_n^{\mathrm{eff}}})^2 \tag{5.6}$$

$$r_0 = 2^{\frac{1}{6}}\frac{(\sigma_{\mathrm{w}}+\sigma_{\mathrm{ion}})}{2} \tag{5.7}$$

$$r_{\mathrm{ion}} = 2^{\frac{1}{6}}\sigma_{\mathrm{ion}} \tag{5.8}$$

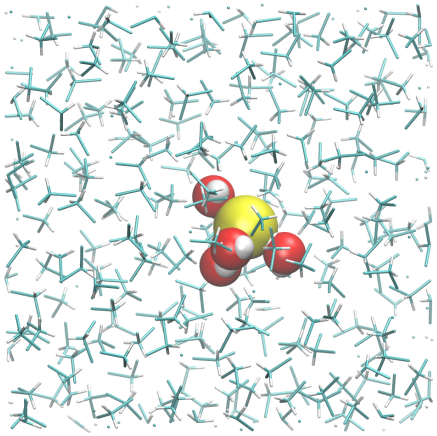


図 5.7　水和数 $n = 4$ の場合の水和数計算のシミュレーションセル。黄色が Cl^- で赤色が水分子を表す。

$$\begin{cases} r_n^{\mathrm{eff}} & = 1.2\left(r_{\mathrm{ion}}^3 + n\frac{3\nu_{\mathrm{w}}}{4\pi}\right)^{\frac{1}{3}} \ (n \geq 4) \\ r_{n=4}^{\mathrm{eff}} & = 1.2\left(r_{\mathrm{ion}}^3 + 4\frac{3\nu_{\mathrm{w}}}{4\pi}\right)^{\frac{1}{3}} \ (n < 4) \end{cases} \tag{5.9}$$

今回は水和数 7 個までを計算した。水和自由エネルギーは、水和数が増えていくとほとんど水-水の相互作用になり一定になる。また水の化学ポテンシャルは平衡に達すると $\mu_{\text{water in water}} = \mu_{\text{water in DCM}}$ が成り立つため、水の蒸気圧曲線から、温度 298.15K の時 $\mu_{\text{water in water,Exp.}} = -10.88$ と計算した値から更に $\mu^*_{\text{Exp.}}$ と $\mu^*_{\text{Calc.}}$ の差で補正した値を用いた。

$$\mu_{\text{water in DCM,MD}} = \mu_{\text{water in water,Exp.}} + \mu^*_{\text{corre}} \tag{5.10}$$

$$mu^*_{\text{corre}} = \mu^*_{\text{water in DCM,MD}} - \mu^*_{\text{water in DCM,Exp.}} \tag{5.11}$$

結果を表 5.7 にまとめた。平均水和数 $n_{\text{Ave.}}$ は $n_{\text{Ave.}} = \sum_i^{\infty} iP_i$ で計算した。

n	w	δw_{bias}	$dG(n-1,n)$	$dG - n\mu$	P_n
0	—	—	—	—	1.51×10^{-7}
1	-9.89 ± 0.44	-6.13	-16.03	-4.26	2.05×10^{-4}
2	-8.84 ± 0.24	-6.13	-14.56	-7.06	2.32×10^{-2}
3	-7.80 ± 0.23	-6.13	-13.28	-8.58	0.30
4	-6.64 ± 0.34	-6.13	-11.96	-8.78	0.41
5	-6.09 ± 0.35	-6.17	-11.31	-8.32	0.19
6	-5.86 ± 0.41	-6.19	-10.99	-7.55	5.28×10^{-2}
7	-5.54 ± 0.64	-6.22	-10.61	-6.39	7.48×10^{-3}

表 5.7　DCM 中での Cl^- の水和自由エネルギーと平衡水和数分布

DCM 中での Cl^- の水和数は $n_{\text{Ave.}} = 3.98^{+2.66}_{-0.82}$ だった。今回の結果と実験値 $n_{\text{Ave.}} = 2.4, 3.12$ [8] を比較すると、MD の見積もり結果は実験値より水和数が少し多い。しかしながら、実験では油相にイオンを溶かすために PTC を用いており、Cl^- だけで DCM 中に存在する場合よりも水和数は減るはずである。したがって今回の計算は以前の結果 [9] と比較して大きく改善されて妥当な結果になったと考えている。

第 6 章

液液界面での熱輸送

界面では二種類の溶媒の熱伝導率の差が生じ、界面をこえた熱輸送には温度ジャンプが生じるはずである。これは水中の酸素ナノバブル界面のような系にも適応できる。数値シミュレーションの先行研究によると、水中の酸素ナノバブルの自己崩壊では気泡内が過熱され、気泡内の温度が約 3000 K まで上昇すると報告されている。[10] しかしこのモデルでは気泡崩壊の瞬間の熱の流入出とその熱抵抗が記述できていない可能性があった。本研究ではより信頼できる気泡崩壊の描像を得るために、数値シミュレーションへ組み込む熱抵抗値を非平衡分子動力学シミュレーションで見積もった。この研究は 2021 年に卒業した伊藤孟さんがメインで進めたテーマで、私は MD での界面熱抵抗の見積もりのため界面系の温度プロファイルの計算方法を確立した。

6.1　界面をまたぐ温度プロファイルの計算の LAMMPS への実装

ここでは 6 章の水/酸素界面の温度プロファイルの計算方法について記述する。水/酸素界面を基準とした界面に垂直な距離 z を定義した時に、z に沿った温度を計算するには、z 方向を $nchunk$ 個の分割した各 chunk での温度の平均を計算する必要がある。この計算は分子動力学シミュレーションプログラム LAMMPS を基にして実施されたが、LAMMPS は界面系の温度プロファイルの計算には非対応であった。温度は後述の式 6.1 で定義されるが、図 6.1 のような chunk の境界に存在する分子では、各 chunk での温度を計算する時に結合による自由度をどのように差し引いて計算すべきか曖昧であるため、LAMMPS では自由度を全く考慮せずに自由度を $3n$ として温度を計算する、もしくはバルク系またはもう一方が気相である場合には単純に各サイトの自由度を一律に減らす (酸素分子であれば 2×3-1 の自由度を持つため各サイトの自由度を 2.5) ことで自由度を考慮した温度の計算が可能であった。界面系ではバルク系と異なり複数の分子が存在するため複数の自由度のパラメータ f を扱った温度の計算を毎ステップ行い統計平均ととる必要があったためで、既存の LAMMPS は未実装であった。また後述する原子温度のみが実装されていた。

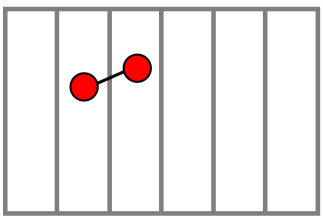


図 6.1　chunk の境界をまたぐ分子

以下に今回 LAMMPS へ実装した二種類の温度の計算方法とコードを記述する。一つ目は原子ごと温度を計算した

もの（以下原子温度と呼ぶ）である。i 番目の chunk での温度 T_i は分子の運動エネルギー K の総和から次のように表される。

$$T_i = \frac{2\sum_{j=1}^{n_i} K_j}{k_B(n_{\mathrm{o},i} f_{\mathrm{o}} + n_{\mathrm{w},i} f_{\mathrm{w}})} \tag{6.1}$$

ただし、j をサイト、n_i を chunk$i$ の総サイト数、n_{o} を酸素のサイト数、n_{w} を水のサイト数、f_{o} を酸素の 1 原子あたりの自由度、f_{w} を水の 1 原子あたりの自由度とした。f_{o} は二原子分子で 1 本の結合の拘束があるため 1 原子あたり $(\frac{2\times3-1}{2} = 2.5)$ の自由度、f_{w} は三原子分子で 2 本の結合と 1 つの角度拘束があるため 1 原子あたり $(\frac{3\times3-3}{3} = 2)$ の自由度を持つ。二つ目は分子ごと温度を計算したもの（以下分子温度と呼ぶ）である。

$$T_{\mathrm{mol},i} = \frac{2\sum_{j=1}^{n_{\mathrm{mol},i}} K_j}{k_B(n_{\mathrm{mol,o},i} f_{\mathrm{mol,o}} + n_{\mathrm{mol,w},i} f_{\mathrm{mol,w}})} \tag{6.2}$$

ただし、j を分子、n_{mol}, i を chunk$i$ の総分子数、$n_{\mathrm{mol,o}}$ を酸素の分子数、$n_{\mathrm{mol,w}}$ を水の分子数、$f_{\mathrm{mol,o}}$ を酸素分子の自由度 $(2\times3-1=5)$、$f_{\mathrm{mol,w}}$ を水分子の自由度 $(3\times3-3=6)$ とした。分子温度では分子の存在する chunk の判定方法がいくつか考えられが、ここでは図 6.1 下段を用いて 2 種類紹介する。一つ目は分子の質量で重み付けした位置 (重心 G) で chunk を判定する方法であり、サイト j =1〜3 から成る分子 l の重心 G_l は次のようになる。

$$G_l = \frac{\sum_{j=1}^{n=3} m_j x_j}{\sum_{j=1}^{n=3} m_j} \tag{6.3}$$

この G_l では、求めた G_l がどの chunk に属するかを判定する必要がある。二つ目は chunk 番号 i で重み付けした重心 G_{chunk} で判定する方法である。分子 l の重心 $G_{l\mathrm{chunk}}$ は次のようになる。

$$G_{l\mathrm{chunk}} = \frac{\sum_{j=1}^{3} m_j i_j}{\sum_{j=1}^{3} m_j} \tag{6.4}$$

今回は LAMMPS に実装する際に都合のよい chunk 番号で重み付けした重心 G_{chunk} を採用した。

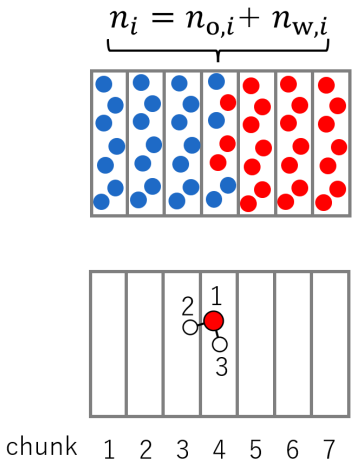


図 6.2　界面系での温度プロファイルの計算

2 種類の温度プロファイルを比較したものを図 6.1 上段に、その時の系の密度を下段に示す。”3n”は LAMMPS のデフォルトで計算できる全サイトの自由度が $3n$ の温度をプロットしてある。バルクで”3n”と chunk の重心温度”mol”

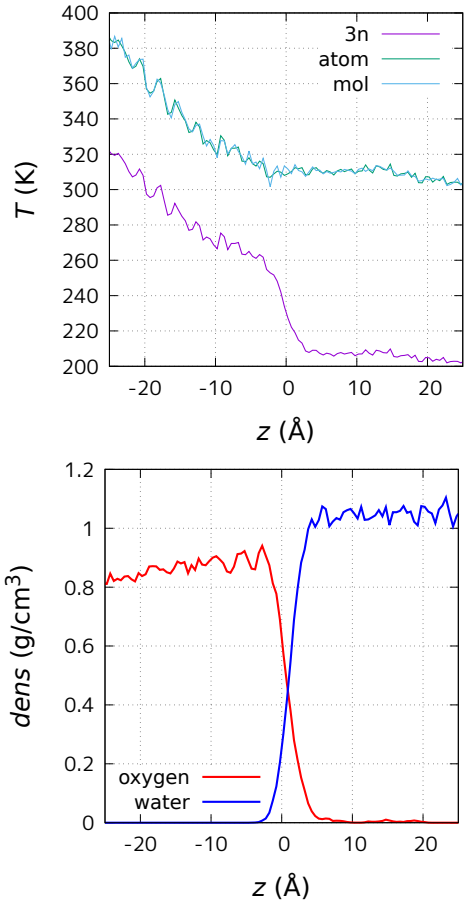


図 6.3　原子温度 (atom) と分子温度 (mol) の温度プロファイルの比較と密度分布

と原子位置での温度”atom”を比較すると、酸素相側は $\frac{3}{2.5} = 1.2$ 倍、水相側では $\frac{3}{2} = 1.5$ 倍となっており正しく実装できていることが確認できる。二種類の温度プロファイルはほとんど同じような結果を与えた。また平面界面以外にも球状のバブル界面の計算にも対応した形で実装した。

6.1.1　コーディング

LAMMPS は分散メモリ型並列計算に対応している。分散メモリ型並列計算には二種類の流儀がある。

1. 空間分割　(メリット：力の計算がしやすい)
2. 粒子数分割 (メリット：空間分割では密度が高い部分と低い部分で分割後の計算コストが大きく異なり並列化しても速くならない場合があるが、粒子数分割ではその問題がない。)

LAMMPS では空間分割が採用されており、mpi-rank 毎に粒子の位置を出力すると図 6.4 のようになる。

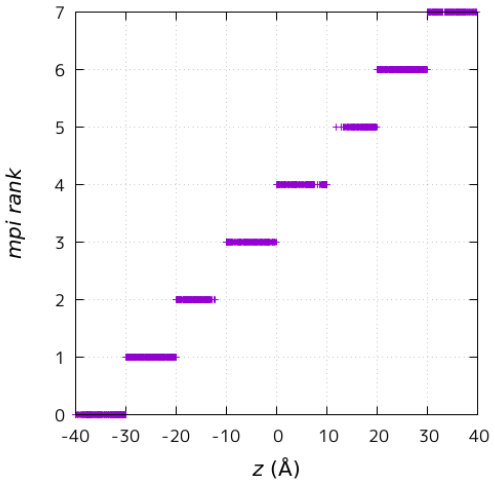


図 6.4　各 mpi-rank での粒子の位置

そして、LAMMPS では空間分割されるため各 mpi-rank の境界で分子は分断されている。したがって分子の重心 G ではなく、chunk で重み付けした重心 G_{chunk} で判定する方が実装上の都合がいい。今回は chunk ごとの計算を扱っている fix_ave_chunk.h, fix_ave_chunk.cpp に機能を追加する形で実装を行った。以下に fix_ave_chunk.h, fix_ave_chunk.cpp を付する。今回追加したコード以外は灰色にしてある。

Listing 6.1　fix_ave_chunk.h

```
1
2  /*----------------------------------------------------------------------
3    LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
4    http://lammps.sandia.gov, Sandia National Laboratories
5    Steve Plimpton, sjplimp@sandia.gov
6
7    Copyright (2003) Sandia Corporation.  Under the terms of Contract
8    DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
9    certain rights in this software.  This software is distributed under
10   the GNU General Public License.
11
```

```
12     See the README file in the top-level LAMMPS directory.
13   ------------------------------------------------------------------------- */
14
15   #ifdef FIX_CLASS
16
17   FixStyle(ave/chunk,FixAveChunk)
18
19   #else
20
21   #ifndef LMP_FIX_AVE_CHUNK_H
22   #define LMP_FIX_AVE_CHUNK_H
23
24   #include <stdio.h>
25   #include "fix.h"
26
27   #include "group.h"
28   #include <vector>
29   #include "dump_xyz.h"
30
31   namespace LAMMPS_NS {
32
33   class FixAveChunk : public Fix {
34    public:
35     FixAveChunk(class LAMMPS *, int, char **);
36     ~FixAveChunk();
37     int setmask();
38     void init();
39     void setup(int);
40     void end_of_step();
41     double compute_array(int,int);
42     double memory_usage();
43
44     int comflag;
45
46    private:
47     int me,nvalues;
48     int nrepeat,nfreq,irepeat;
49     int normflag,scaleflag,overwrite,biasflag,colextra;
50     bigint nvalid,nvalid_last;
51     double cdof;
52   //  double adof;
```

```
53    char *format,*format_user;
54    char *tstring,*sstring,*id_bias;
55    int *which,*argindex,*value2index;
56    char **ids;
57    class Compute *tbias;     // ptr to additional bias compute
58    FILE *fp;
59
60    int ave,nwindow;
61    int normcount,iwindow,window_limit;
62
63    int nchunk,maxchunk;
64    char *idchunk;
65    class ComputeChunkAtom *cchunk;
66    int lockforever;
67
68    long filepos;
69
70    int maxvar;
71    double *varatom;
72
73    // one,many,sum vecs/arrays are used with a single Nfreq epoch
74    // total,list vecs/arrays are used across epochs
75
76    double *count_one,*count_many,*count_sum;
77    double **values_one,**values_many,**values_sum;
78    double *count_total,**count_list;
79    double **values_total,***values_list;
80
81    void allocate();
82    bigint nextvalid();
83
84    int adofflag;
85    int nadof;
86    double *adof;
87    int nname;
88    int ncom;
89    int nmol;
90    int *mol1;
91    int *mol2;
92    double **count_one_add,**count_many_add,**count_sum_add;
```

Listing 6.2　fix_ave_chunk.cpp

```
/* ----------------------------------------------------------------------
   LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
   http://lammps.sandia.gov, Sandia National Laboratories
   Steve Plimpton, sjplimp@sandia.gov

   Copyright (2003) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level LAMMPS directory.
------------------------------------------------------------------------- */

#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "fix_ave_chunk.h"
#include "atom.h"
#include "update.h"
#include "force.h"
#include "domain.h"
#include "modify.h"
#include "compute.h"
#include "compute_chunk_atom.h"
#include "input.h"
#include "variable.h"
#include "memory.h"
#include "error.h"

#include "group.h"
#include <vector>
#include "dump_xyz.h"
std::vector<std::string> name;
std::string str;

using namespace LAMMPS_NS;
using namespace FixConst;

```

```
39  enum{V,F,DENSITY_NUMBER,DENSITY_MASS,MASS,TEMPERATURE,COMPUTE,FIX,VARIABLE};
40  enum{SAMPLE,ALL};
41  enum{NOSCALE,ATOM};
42  enum{ONE,RUNNING,WINDOW};
43
44  #define INVOKED_PERATOM 8
45
46  /* ---------------------------------------------------------------------- */
47
48  FixAveChunk::FixAveChunk(LAMMPS *lmp, int narg, char **arg) :
49    Fix(lmp, narg, arg)
50  {
51    if (narg < 7) error->all(FLERR,"Illegal fix ave/chunk command");
52
53    MPI_Comm_rank(world,&me);
54
55    nevery = force->inumeric(FLERR,arg[3]);
56    nrepeat = force->inumeric(FLERR,arg[4]);
57    nfreq = force->inumeric(FLERR,arg[5]);
58
59    int n = strlen(arg[6]) + 1;
60    idchunk = new char[n];
61    strcpy(idchunk,arg[6]);
62
63    global_freq = nfreq;
64    no_change_box = 1;
65
66    // parse values until one isn't recognized
67
68    int iarg = 7;
69    which = new int[narg-iarg];
70    argindex = new int[narg-iarg];
71    ids = new char*[narg-iarg];
72    value2index = new int[narg-iarg];
73    nvalues = 0;
74    adofflag=0;
75    comflag=0;
76    nadof=0;
77    nname=0;
78    nmol=0;
79    adof=NULL;
```

```
80    mol1=NULL;
81    mol2=NULL;
82
83    while (iarg < narg) {
84      ids[nvalues] = NULL;
85
86      if (strcmp(arg[iarg],"vx") == 0) {
87        which[nvalues] = V;
88        argindex[nvalues++] = 0;
89      } else if (strcmp(arg[iarg],"vy") == 0) {
90        which[nvalues] = V;
91        argindex[nvalues++] = 1;
92      } else if (strcmp(arg[iarg],"vz") == 0) {
93        which[nvalues] = V;
94        argindex[nvalues++] = 2;
95
96      } else if (strcmp(arg[iarg],"fx") == 0) {
97        which[nvalues] = F;
98        argindex[nvalues++] = 0;
99      } else if (strcmp(arg[iarg],"fy") == 0) {
100       which[nvalues] = F;
101       argindex[nvalues++] = 1;
102     } else if (strcmp(arg[iarg],"fz") == 0) {
103       which[nvalues] = F;
104       argindex[nvalues++] = 2;
105
106     } else if (strcmp(arg[iarg],"density/number") == 0) {
107       which[nvalues] = DENSITY_NUMBER;
108       argindex[nvalues++] = 0;
109     } else if (strcmp(arg[iarg],"density/mass") == 0) {
110       which[nvalues] = DENSITY_MASS;
111       argindex[nvalues++] = 0;
112     } else if (strcmp(arg[iarg],"mass") == 0) {
113       which[nvalues] = MASS;
114       argindex[nvalues++] = 0;
115     } else if (strcmp(arg[iarg],"temp") == 0) {
116       which[nvalues] = TEMPERATURE;
117       argindex[nvalues++] = 0;
118
119     } else if (strncmp(arg[iarg],"c_",2) == 0 ||
120                strncmp(arg[iarg],"f_",2) == 0 ||
```

```
121                 strncmp(arg[iarg],"v_",2) == 0) {
122         if (arg[iarg][0] == 'c') which[nvalues] = COMPUTE;
123         else if (arg[iarg][0] == 'f') which[nvalues] = FIX;
124         else if (arg[iarg][0] == 'v') which[nvalues] = VARIABLE;
125
126         int n = strlen(arg[iarg]);
127         char *suffix = new char[n];
128         strcpy(suffix,&arg[iarg][2]);
129
130         char *ptr = strchr(suffix,'[');
131         if (ptr) {
132           if (suffix[strlen(suffix)-1] != ']')
133             error->all(FLERR,"Illegal fix ave/chunk command");
134           argindex[nvalues] = atoi(ptr+1);
135           *ptr = '\0';
136         } else argindex[nvalues] = 0;
137
138         n = strlen(suffix) + 1;
139         ids[nvalues] = new char[n];
140         strcpy(ids[nvalues],suffix);
141         nvalues++;
142         delete [] suffix;
143
144       } else break;
145
146       iarg++;
147     }
148
149     if (nvalues == 0) error->all(FLERR,"No values in fix ave/chunk command");
150
151     // optional args
152
153     normflag = ALL;
154     scaleflag = ATOM;
155     ave = ONE;
156     fp = NULL;
157     nwindow = 0;
158     biasflag = 0;
159     id_bias = NULL;
160     cdof = 0;
161     overwrite = 0;
```

```
162     format_user = NULL;
163     format = (char *) " %g";
164     char *title1 = NULL;
165     char *title2 = NULL;
166     char *title3 = NULL;
167
168     while (iarg < narg) {
169       if (strcmp(arg[iarg],"norm") == 0) {
170         if (iarg+2 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
171         if (strcmp(arg[iarg+1],"all") == 0) {
172           normflag = ALL;
173           scaleflag = ATOM;
174         } else if (strcmp(arg[iarg+1],"sample") == 0) {
175           normflag = SAMPLE;
176           scaleflag = ATOM;
177         } else if (strcmp(arg[iarg+1],"none") == 0) {
178           normflag = SAMPLE;
179           scaleflag = NOSCALE;
180         } else error->all(FLERR,"Illegal fix ave/chunk command");
181         iarg += 2;
182       } else if (strcmp(arg[iarg],"ave") == 0) {
183         if (iarg+2 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
184         if (strcmp(arg[iarg+1],"one") == 0) ave = ONE;
185         else if (strcmp(arg[iarg+1],"running") == 0) ave = RUNNING;
186         else if (strcmp(arg[iarg+1],"window") == 0) ave = WINDOW;
187         else error->all(FLERR,"Illegal fix ave/chunk command");
188         if (ave == WINDOW) {
189           if (iarg+3 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
190           nwindow = force->inumeric(FLERR,arg[iarg+2]);
191           if (nwindow <= 0) error->all(FLERR,"Illegal fix ave/chunk command");
192         }
193         iarg += 2;
194         if (ave == WINDOW) iarg++;
195
196       } else if (strcmp(arg[iarg],"bias") == 0) {
197         if (iarg+2 > narg)
198           error->all(FLERR,"Illegal fix ave/chunk command");
199         biasflag = 1;
200         int n = strlen(arg[iarg+1]) + 1;
201         id_bias = new char[n];
202         strcpy(id_bias,arg[iarg+1]);
```

```
203         iarg += 2;
204       } else if (strcmp(arg[iarg],"adof") == 0) {
205         if (iarg+2 > narg)
206           error->all(FLERR,"Illegal fix ave/chunk command");
207         adofflag=1;
208         nadof = force->numeric(FLERR,arg[iarg+1]);
209         adof = new double [nadof];
210         if (iarg+2+nadof > narg)
211           error->all(FLERR,"Illegal fix ave/chunk command");
212         for (int in = 0; in < nadof; in++) {
213           adof[in] = force->numeric(FLERR,arg[iarg+2+in]);
214         }
215
216   iarg += 4;
217       } else if (strcmp(arg[iarg],"name") == 0) {
218         if (adofflag!=1) error->all(FLERR,"Illegal fix ave/chunk command");
219         if (iarg+2 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
220         nname = force->numeric(FLERR,arg[iarg+1]);
221         char buff[128];
222         iarg += 2;
223         if (iarg+nname > narg)
224           error->all(FLERR,"Illegal fix ave/chunk command");
225         for (int in = 0; in < nname; in++) {
226         strcpy(buff,arg[iarg+in]);
227         str=buff;
228         name.push_back(str);
229         }
230         iarg += nname;
231       } else if (strcmp(arg[iarg],"com") == 0) {
232         if (adofflag!=1) error->all(FLERR,"Illegal fix ave/chunk command");
233         if (iarg+2 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
234         comflag = 1;
235         nmol = force->numeric(FLERR,arg[iarg+1]);
236         iarg += 1;
237         ncom = force->numeric(FLERR,arg[iarg+1]);
238         iarg += 1;
239         if (iarg+ncom*2+2> narg)
240           error->all(FLERR,"Illegal fix ave/chunk command");
241         mol1 = new int[ncom];
242         mol2 = new int[ncom];
243         for (int in = 0; in < ncom; in++) {
```

```
244         mol1[in] = force->numeric(FLERR,arg[iarg+in+1]);
245       }
246       iarg += ncom;
247       for (int in = 0; in < ncom; in++) {
248         mol2[in] = force->numeric(FLERR,arg[iarg+in+1]);
249       }
250       iarg += ncom+1;
251
252     } else if (strcmp(arg[iarg],"cdof") == 0) {
253       if (iarg+2 > narg)
254         error->all(FLERR,"Illegal fix ave/chunk command");
255       cdof = force->numeric(FLERR,arg[iarg+1]);
256       iarg += 2;
257     } else if (strcmp(arg[iarg],"file") == 0) {
258       if (iarg+2 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
259       if (me == 0) {
260         fp = fopen(arg[iarg+1],"w");
261         if (fp == NULL) {
262           char str[128];
263           sprintf(str,"Cannot open fix ave/chunk file %s",arg[iarg+1]);
264           error->one(FLERR,str);
265         }
266       }
267       iarg += 2;
268     } else if (strcmp(arg[iarg],"overwrite") == 0) {
269       overwrite = 1;
270       iarg += 1;
271     } else if (strcmp(arg[iarg],"format") == 0) {
272       if (iarg+2 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
273       delete [] format_user;
274       int n = strlen(arg[iarg+1]) + 2;
275       format_user = new char[n];
276       sprintf(format_user," %s",arg[iarg+1]);
277       format = format_user;
278       iarg += 2;
279     } else if (strcmp(arg[iarg],"title1") == 0) {
280       if (iarg+2 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
281       delete [] title1;
282       int n = strlen(arg[iarg+1]) + 1;
283       title1 = new char[n];
284       strcpy(title1,arg[iarg+1]);
```

```
285         iarg += 2;
286       } else if (strcmp(arg[iarg],"title2") == 0) {
287         if (iarg+2 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
288         delete [] title2;
289         int n = strlen(arg[iarg+1]) + 1;
290         title2 = new char[n];
291         strcpy(title2,arg[iarg+1]);
292         iarg += 2;
293       } else if (strcmp(arg[iarg],"title3") == 0) {
294         if (iarg+2 > narg) error->all(FLERR,"Illegal fix ave/chunk command");
295         delete [] title3;
296         int n = strlen(arg[iarg+1]) + 1;
297         title3 = new char[n];
298         strcpy(title3,arg[iarg+1]);
299         iarg += 2;
300       } else error->all(FLERR,"Illegal fix ave/chunk command");
301     }
302
303     if (adofflag ==1){
304       if (me ==0){
305         fprintf(logfile,"\n/*----------------------------------------\n");
306         fprintf(logfile, "#GROUP NAME for CALC TEMP in ADOF MODE\n");
307         fprintf(screen,"\n/*----------------------------------------\n");
308         fprintf(screen, "#GROUP NAME for CALC TEMP in ADOF MODE\n");
309         for (int in=0; in<nname;in++){
310           fprintf(logfile,"%s %d %s \n","group",in+1,name[in].c_str());
311           fprintf(screen,"%s %d %s \n","group",in+1,name[in].c_str());
312         }
313
314         if (comflag==1) {
315           fprintf(logfile,"\n#COM\n");
316           fprintf(screen,"\n#COM\n");
317
318           fprintf(logfile,"%s %d \n","nmol",nmol);
319           fprintf(screen,"%s %d \n","nmol",nmol);
320
321           fprintf(logfile,"%s %s %s\n", "#name", "start-ID", "end-ID");
322           fprintf(screen,"%s %s %s\n", "#name", "start-ID", "end-ID");
323           for (int in=0; in<nname;in++){
324             fprintf(logfile,"%s %d %d \n",name[in].c_str(),mol1[in],mol2[in]);
325             fprintf(screen,"%s %d %d \n",name[in].c_str(),mol1[in],mol2[in]);
```

```
326           }
327         }
328
329           fprintf(logfile,"\n#ADOF\n");
330           fprintf(screen,"\n#ADOF\n");
331           fprintf(logfile,"%s %s\n", "#name", "adof");
332           fprintf(screen,"%s %s\n", "#name", "adof");
333         for (int in=0; in<nname;in++){
334           fprintf(logfile,"%s %f\n",name[in].c_str(),adof[in]);
335           fprintf(screen,"%s %f\n",name[in].c_str(),adof[in]);
336         }
337         fprintf(logfile,"----------------------------------------*/\n\n");
338         fprintf(screen,"----------------------------------------*/\n\n");
339       }
340     }
341
342     // setup and error check
343
344     if (nevery <= 0 || nrepeat <= 0 || nfreq <= 0)
345       error->all(FLERR,"Illegal fix ave/chunk command");
346     if (nfreq % nevery || nrepeat*nevery > nfreq)
347       error->all(FLERR,"Illegal fix ave/chunk command");
348     if (ave != RUNNING && overwrite)
349       error->all(FLERR,"Illegal fix ave/chunk command");
350
351     if (biasflag) {
352       int i = modify->find_compute(id_bias);
353       if (i < 0)
354         error->all(FLERR,"Could not find compute ID for temperature bias");
355       tbias = modify->compute[i];
356       if (tbias->tempflag == 0)
357         error->all(FLERR,"Bias compute does not calculate temperature");
358       if (tbias->tempbias == 0)
359         error->all(FLERR,"Bias compute does not calculate a velocity bias");
360     }
361
362     for (int i = 0; i < nvalues; i++) {
363       if (which[i] == COMPUTE) {
364         int icompute = modify->find_compute(ids[i]);
365         if (icompute < 0)
366           error->all(FLERR,"Compute ID for fix ave/chunk does not exist");
```

```
367         if (modify->compute[icompute]->peratom_flag == 0)
368           error->all(FLERR,"Fix ave/chunk compute does not "
369                      "calculate per-atom values");
370         if (argindex[i] == 0 &&
371             modify->compute[icompute]->size_peratom_cols != 0)
372           error->all(FLERR,"Fix ave/chunk compute does not "
373                      "calculate a per-atom vector");
374         if (argindex[i] && modify->compute[icompute]->size_peratom_cols == 0)
375           error->all(FLERR,"Fix ave/chunk compute does not "
376                      "calculate a per-atom array");
377         if (argindex[i] &&
378             argindex[i] > modify->compute[icompute]->size_peratom_cols)
379           error->all(FLERR,
380                      "Fix ave/chunk compute vector is accessed out-of-range");
381
382       } else if (which[i] == FIX) {
383         int ifix = modify->find_fix(ids[i]);
384         if (ifix < 0)
385           error->all(FLERR,"Fix ID for fix ave/chunk does not exist");
386         if (modify->fix[ifix]->peratom_flag == 0)
387           error->all(FLERR,
388                      "Fix ave/chunk fix does not calculate per-atom values");
389         if (argindex[i] == 0 && modify->fix[ifix]->size_peratom_cols != 0)
390           error->all(FLERR,
391                      "Fix ave/chunk fix does not calculate a per-atom vector");
392         if (argindex[i] && modify->fix[ifix]->size_peratom_cols == 0)
393           error->all(FLERR,
394                      "Fix ave/chunk fix does not calculate a per-atom array");
395         if (argindex[i] && argindex[i] > modify->fix[ifix]->size_peratom_cols)
396           error->all(FLERR,"Fix ave/chunk fix vector is accessed out-of-range");
397       } else if (which[i] == VARIABLE) {
398         int ivariable = input->variable->find(ids[i]);
399         if (ivariable < 0)
400           error->all(FLERR,"Variable name for fix ave/chunk does not exist");
401         if (input->variable->atomstyle(ivariable) == 0)
402           error->all(FLERR,"Fix ave/chunk variable is not atom-style variable");
403       }
404     }
405
406     // increment lock counter in compute chunk/atom
407     // only if nrepeat > 1 or ave = RUNNING/WINDOW,
```

```
408   //   so that locking spans multiple timesteps
409
410   int icompute = modify->find_compute(idchunk);
411   if (icompute < 0)
412     error->all(FLERR,"Chunk/atom compute does not exist for fix ave/chunk");
413   cchunk = (ComputeChunkAtom *) modify->compute[icompute];
414   if (strcmp(cchunk->style,"chunk/atom") != 0)
415     error->all(FLERR,"Fix ave/chunk does not use chunk/atom compute");
416
417   if (nrepeat > 1 || ave == RUNNING || ave == WINDOW) cchunk->lockcount++;
418   lockforever = 0;
419
420   // print file comment lines
421
422   if (fp && me == 0) {
423     clearerr(fp);
424     if (title1) fprintf(fp,"%s\n",title1);
425     else fprintf(fp,"# Chunk-averaged data for fix %s and group %s\n",
426                  id,arg[1]);
427     if (title2) fprintf(fp,"%s\n",title2);
428     else fprintf(fp,"# Timestep Number-of-chunks Total-count\n");
429     if (title3) fprintf(fp,"%s\n",title3);
430     else {
431       int compress = cchunk->compress;
432       int ncoord = cchunk->ncoord;
433       if (!compress) {
434         if (ncoord == 0) fprintf(fp,"# Chunk Ncount");
435         else if (ncoord == 1) fprintf(fp,"# Chunk Coord1 Ncount");
436         else if (ncoord == 2) fprintf(fp,"# Chunk Coord1 Coord2 Ncount");
437         else if (ncoord == 3)
438           fprintf(fp,"# Chunk Coord1 Coord2 Coord3 Ncount");
439       } else {
440         if (ncoord == 0) fprintf(fp,"# Chunk OrigID Ncount");
441         else if (ncoord == 1) fprintf(fp,"# Chunk OrigID Coord1 Ncount");
442         else if (ncoord == 2) fprintf(fp,"# Chunk OrigID Coord1 Coord2 Ncount");
443         else if (ncoord == 3)
444           fprintf(fp,"# Chunk OrigID Coord1 Coord2 Coord3 Ncount");
445       }
446       for (int i = 0; i < nvalues; i++) fprintf(fp," %s",arg[7+i]);
447       fprintf(fp,"\n");
448     }
```

```
    if (ferror(fp))
      error->one(FLERR,"Error writing file header");

    filepos = ftell(fp);
  }

  delete [] title1;
  delete [] title2;
  delete [] title3;

  // this fix produces a global array
  // size_array_rows is variable and set by allocate()

  int compress = cchunk->compress;
  int ncoord = cchunk->ncoord;
  colextra = compress + ncoord;

  array_flag = 1;
  size_array_cols = colextra + 1 + nvalues;
  size_array_rows_variable = 1;
  extarray = 0;

  // initializations

  irepeat = 0;
  iwindow = window_limit = 0;
  normcount = 0;

  maxvar = 0;
  varatom = NULL;

  count_one = count_many = count_sum = count_total = NULL;
  if (adofflag==1){
    count_one_add = count_many_add = count_sum_add = NULL;
  }

  count_list = NULL;
  values_one = values_many = values_sum = values_total = NULL;
  values_list = NULL;

  maxchunk = 0;
```

```
490     nchunk = 1;
491
492     allocate();
493
494     // nvalid = next step on which end_of_step does something
495     // add nvalid to all computes that store invocation times
496     // since don't know a priori which are invoked by this fix
497     // once in end_of_step() can set timestep for ones actually invoked
498
499     nvalid_last = -1;
500     nvalid = nextvalid();
501     modify->addstep_compute_all(nvalid);
502
503   }
504
505   /* ---------------------------------------------------------------------- */
506
507   FixAveChunk::~FixAveChunk()
508   {
509     delete [] which;
510     delete [] argindex;
511     for (int i = 0; i < nvalues; i++) delete [] ids[i];
512     delete [] ids;
513     delete [] value2index;
514
515     if (fp && me == 0) fclose(fp);
516
517     memory->destroy(varatom);
518
519     memory->destroy(count_one);
520     memory->destroy(count_many);
521     memory->destroy(count_sum);
522
523     if (adofflag==1){
524       memory->destroy(count_one_add);
525       memory->destroy(count_many_add);
526       memory->destroy(count_sum_add);
527     }
528
529     memory->destroy(count_total);
530     memory->destroy(count_list);
```

```
531   memory->destroy(values_one);
532   memory->destroy(values_many);
533   memory->destroy(values_sum);
534   memory->destroy(values_total);
535   memory->destroy(values_list);
536
537   // decrement lock counter in compute chunk/atom, it if still exists
538
539   if (nrepeat > 1 || ave == RUNNING || ave == WINDOW) {
540     int icompute = modify->find_compute(idchunk);
541     if (icompute >= 0) {
542       cchunk = (ComputeChunkAtom *) modify->compute[icompute];
543       if (ave == RUNNING || ave == WINDOW) cchunk->unlock(this);
544       cchunk->lockcount--;
545     }
546   }
547
548   delete [] idchunk;
549 }
550
551 /* ---------------------------------------------------------------------- */
552
553 int FixAveChunk::setmask()
554 {
555   int mask = 0;
556   mask |= END_OF_STEP;
557   return mask;
558 }
559
560 /* ---------------------------------------------------------------------- */
561
562 void FixAveChunk::init()
563 {
564   // set indices and check validity of all computes,fixes,variables
565   // check that fix frequency is acceptable
566   int icompute = modify->find_compute(idchunk);
567   if (icompute < 0)
568     error->all(FLERR,"Chunk/atom compute does not exist for fix ave/chunk");
569   cchunk = (ComputeChunkAtom *) modify->compute[icompute];
570
571   if (biasflag) {
```

```
572       int i = modify->find_compute(id_bias);
573       if (i < 0)
574         error->all(FLERR,"Could not find compute ID for temperature bias");
575       tbias = modify->compute[i];
576     }
577
578     for (int m = 0; m < nvalues; m++) {
579       if (which[m] == COMPUTE) {
580         int icompute = modify->find_compute(ids[m]);
581         if (icompute < 0)
582           error->all(FLERR,"Compute ID for fix ave/chunk does not exist");
583         value2index[m] = icompute;
584
585       } else if (which[m] == FIX) {
586         int ifix = modify->find_fix(ids[m]);
587         if (ifix < 0)
588           error->all(FLERR,"Fix ID for fix ave/chunk does not exist");
589         value2index[m] = ifix;
590
591         if (nevery % modify->fix[ifix]->peratom_freq)
592           error->all(FLERR,
593                      "Fix for fix ave/chunk not computed at compatible time");
594
595       } else if (which[m] == VARIABLE) {
596         int ivariable = input->variable->find(ids[m]);
597         if (ivariable < 0)
598           error->all(FLERR,"Variable name for fix ave/chunk does not exist");
599         value2index[m] = ivariable;
600
601       } else value2index[m] = -1;
602     }
603
604     // need to reset nvalid if nvalid < ntimestep b/c minimize was performed
605
606     if (nvalid < update->ntimestep) {
607       irepeat = 0;
608       nvalid = nextvalid();
609       modify->addstep_compute_all(nvalid);
610     }
611   }
612
```

```
613 /* ----------------------------------------------------------------------
614    only does averaging if nvalid = current timestep
615    do not call setup_chunks(), even though fix ave/spatial called setup_bins()
616    b/c could cause nchunk to change if Nfreq epoch crosses 2 runs
617    does mean that if change_box is used between runs to change box size,
618      that nchunk may not track it
619 ------------------------------------------------------------------------- */
620
621 void FixAveChunk::setup(int vflag)
622 {
623   end_of_step();
624 }
625
626 /* ---------------------------------------------------------------------- */
627
628 void FixAveChunk::end_of_step()
629 {
630   int i,j,m,n,index;
631
632   // skip if not step which requires doing something
633   // error check if timestep was reset in an invalid manner
634
635   bigint ntimestep = update->ntimestep;
636   if (ntimestep < nvalid_last || ntimestep > nvalid)
637     error->all(FLERR,"Invalid timestep reset for fix ave/chunk");
638   if (ntimestep != nvalid) return;
639   nvalid_last = nvalid;
640
641   // first sample within single Nfreq epoch
642   // zero out arrays that accumulate over many samples, but not across epochs
643   // invoke setup_chunks() to determine current nchunk
644   //   re-allocate per-chunk arrays if needed
645   // invoke lock() in two cases:
646   //   if nrepeat > 1: so nchunk cannot change until Nfreq epoch is over,
647   //     will be unlocked on last repeat of this Nfreq
648   //   if ave = RUNNING/WINDOW and not yet locked:
649   //     set forever, will be unlocked in fix destructor
650   // wrap setup_chunks in clearstep/addstep b/c it may invoke computes
651   //   both nevery and nfreq are future steps,
652   //   since call below to cchunk->ichunk()
653   //     does not re-invoke internal cchunk compute on this same step
```

```
654
655   if (irepeat == 0) {
656     if (cchunk->computeflag) modify->clearstep_compute();
657     nchunk = cchunk->setup_chunks();
658     if (cchunk->computeflag) {
659       modify->addstep_compute(ntimestep+nevery);
660       modify->addstep_compute(ntimestep+nfreq);
661     }
662     allocate();
663     if (nrepeat > 1 && ave == ONE)
664       cchunk->lock(this,ntimestep,ntimestep+(nrepeat-1)*nevery);
665     else if ((ave == RUNNING || ave == WINDOW) && !lockforever) {
666       cchunk->lock(this,update->ntimestep,-1);
667       lockforever = 1;
668     }
669
670     for (m = 0; m < nchunk; m++) {
671       if (adofflag==1){
672         for (int in = 0; in < nname; in++)
673           count_many_add[in][m] = count_sum_add[in][m] = 0;
674         }
675         count_many[m] = count_sum[m] = 0;
676         for (i = 0; i < nvalues; i++) values_many[m][i] = 0;
677     }
678   }
679
680   // zero out arrays for one sample
681
682   for (m = 0; m < nchunk; m++) {
683     if (adofflag==1){
684       for (int in = 0; in < nname; in++) {
685         count_one_add[in][m] = 0;
686       }
687     }
688       count_one[m] = 0;
689       for (i = 0; i < nvalues; i++) values_one[m][i] = 0;
690   }
691
692   // compute chunk/atom assigns atoms to chunk IDs
693   // extract ichunk index vector from compute
694   // ichunk = 1 to Nchunk for included atoms, 0 for excluded atoms
```

```
695   // wrap compute_ichunk in clearstep/addstep b/c it may invoke computes
696
697   if (cchunk->computeflag) modify->clearstep_compute();
698
699   cchunk->compute_ichunk();
700   int *ichunk = cchunk->ichunk;
701
702   if (cchunk->computeflag) modify->addstep_compute(ntimestep+nevery);
703
704   // perform the computation for one sample
705   // count # of atoms in each bin
706   // accumulate results of attributes,computes,fixes,variables to local copy
707   // sum within each chunk, only include atoms in fix group
708   // compute/fix/variable may invoke computes so wrap with clear/add
709
710   int *mask = atom->mask;
711   int nlocal = atom->nlocal;
712
713    for (i = 0; i < nlocal; i++){
714      if (mask[i] & groupbit && ichunk[i] > 0)
715        count_one[ichunk[i]-1]++;
716    }
717
718   if(adofflag==1 && comflag==0){
719     int gid_add[nname];
720     int groupbit_add[nname];
721
722     for (int in = 0; in < nname; in++){
723       gid_add[in]=group->find(name[in].c_str());
724       groupbit_add[in] = group->bitmask[gid_add[in]];
725     }
726
727     for (i = 0; i < nlocal; i++){
728       for (int in = 0; in < nname; in++)
729       if (mask[i] & groupbit_add[in] && ichunk[i] > 0)
730         count_one_add[in][ichunk[i]-1]++;
731     }
732
733   }else if (adofflag==1 && comflag==1){
734     double com[nmol+1], comall[nmol+1];
735     double Tmol[nmol+1], Tmolall[nmol+1];
```

```
736       double masstotal[nmol+1], masstotalall[nmol+1];
737       int *tag = atom->tag;
738       int *type = atom->type;
739       int *molecule = atom->molecule;
740       double **v = atom->v;
741       double onemass;
742       double vx2, vy2, vz2;
743       int im;
744
745       for (int m = 0; m < nvalues; m++) {
746         if (which[m] == TEMPERATURE) im=m;
747       }
748
749       // make com[nmol], Tmol[nmol], masstotal[nmol]
750       for (int imol=1; imol<=nmol; imol++){
751         com[imol]=comall[imol]=Tmol[imol]=Tmolall[imol]=masstotal[imol]=masstotalall[imol]=0;
752       }
753
754       for (int i=0; i<nlocal; i++){
755         onemass=atom->mass[type[i]];
756         vx2 = v[i][0]*v[i][0];
757         vy2 = v[i][1]*v[i][1];
758         vz2 = v[i][2]*v[i][2];
759
760         int imol = molecule[i];
761        com[imol]       +=ichunk[i]*onemass;
762        Tmol[imol]      += (vx2 + vy2 + vz2)*onemass;
763        masstotal[imol] +=onemass;
764       }
765
766       // make comall[nmol], Tmolall[nmol], masstotalall[nmol]
767      MPI_Allreduce(com,comall,(nmol+1),MPI_DOUBLE,MPI_SUM,world);
768      MPI_Allreduce(Tmol,Tmolall,(nmol+1),MPI_DOUBLE,MPI_SUM,world);
769      MPI_Allreduce(masstotal,masstotalall,(nmol+1),MPI_DOUBLE,MPI_SUM,world);
770
771       // chunk
772       for (int in = 0; in< ncom; in++) {
773         for (int imol=mol1[in];imol<=mol2[in];imol++){
774          int ichunk = round(comall[imol] / masstotalall[imol]);
775          count_one_add[in][ichunk-1]++;
776          values_one[ichunk-1][im] += Tmolall[imol];
```

```
777         }
778       }
779     }
780
781     modify->clearstep_compute();
782
783     for (int m = 0; m < nvalues; m++) {
784       n = value2index[m];
785       j = argindex[m];
786
787       // V,F adds velocities,forces to values
788
789       if (which[m] == V || which[m] == F) {
790         double **attribute;
791         if (which[m] == V) attribute = atom->v;
792         else attribute = atom->f;
793
794         for (i = 0; i < nlocal; i++)
795           if (mask[i] & groupbit && ichunk[i] > 0) {
796             index = ichunk[i]-1;
797             values_one[index][m] += attribute[i][j];
798           }
799
800       // DENSITY_NUMBER adds 1 to values
801
802       } else if (which[m] == DENSITY_NUMBER) {
803
804         for (i = 0; i < nlocal; i++)
805           if (mask[i] & groupbit && ichunk[i] > 0) {
806             index = ichunk[i]-1;
807             values_one[index][m] += 1.0;
808           }
809
810       // DENSITY_MASS or MASS adds mass to values
811
812       } else if (which[m] == DENSITY_MASS || which[m] == MASS) {
813         int *type = atom->type;
814         double *mass = atom->mass;
815         double *rmass = atom->rmass;
816
817         if (rmass) {
```

```
818   for (i = 0; i < nlocal; i++)
819     if (mask[i] & groupbit && ichunk[i] > 0) {
820       index = ichunk[i]-1;
821       values_one[index][m] += rmass[i];
822           }
823        } else {
824          for (i = 0; i < nlocal; i++)
825     if (mask[i] & groupbit && ichunk[i] > 0) {
826       index = ichunk[i]-1;
827       values_one[index][m] += mass[type[i]];
828           }
829        }
830
831      // TEMPERATURE adds KE to values
832      // subtract and restore velocity bias if requested
833
834      } else if (which[m] == TEMPERATURE) {
835
836        if (comflag==0){
837
838         if (biasflag) {
839           if (tbias->invoked_scalar != ntimestep) tbias->compute_scalar();
840           tbias->remove_bias_all();
841         }
842
843          double **v = atom->v;
844          int *type = atom->type;
845          double *mass = atom->mass;
846          double *rmass = atom->rmass;
847
848         if (rmass) {
849           for (i = 0; i < nlocal; i++)
850             if (mask[i] & groupbit && ichunk[i] > 0) {
851               index = ichunk[i]-1;
852               values_one[index][m] +=
853                 (v[i][0]*v[i][0] + v[i][1]*v[i][1] + v[i][2]*v[i][2]) * rmass[i];
854             }
855         } else {
856           for (i = 0; i < nlocal; i++)
857             if (mask[i] & groupbit && ichunk[i] > 0) {
858               index = ichunk[i]-1;
```

```
            values_one[index][m] +=
              (v[i][0]*v[i][0] + v[i][1]*v[i][1] + v[i][2]*v[i][2]) *
              mass[type[i]];
          }
        }

      }

      if (biasflag) tbias->restore_bias_all();

    // COMPUTE adds its scalar or vector component to values
    // invoke compute if not previously invoked

    } else if (which[m] == COMPUTE) {
      Compute *compute = modify->compute[n];
      if (!(compute->invoked_flag & INVOKED_PERATOM)) {
        compute->compute_peratom();
        compute->invoked_flag |= INVOKED_PERATOM;
      }
      double *vector = compute->vector_atom;
      double **array = compute->array_atom;
      int jm1 = j - 1;

      for (i = 0; i < nlocal; i++)
        if (mask[i] & groupbit && ichunk[i] > 0) {
          index = ichunk[i]-1;
          if (j == 0) values_one[index][m] += vector[i];
          else values_one[index][m] += array[i][jm1];
        }

    // FIX adds its scalar or vector component to values
    // access fix fields, guaranteed to be ready

    } else if (which[m] == FIX) {
      double *vector = modify->fix[n]->vector_atom;
      double **array = modify->fix[n]->array_atom;
      int jm1 = j - 1;

      for (i = 0; i < nlocal; i++)
        if (mask[i] & groupbit && ichunk[i] > 0) {
          index = ichunk[i]-1;
```

```
900           if (j == 0) values_one[index][m] += vector[i];
901           else values_one[index][m] += array[i][jm1];
902         }
903
904       // VARIABLE adds its per-atom quantities to values
905       // evaluate atom-style variable
906
907       } else if (which[m] == VARIABLE) {
908         if (atom->nmax > maxvar) {
909           maxvar = atom->nmax;
910           memory->destroy(varatom);
911           memory->create(varatom,maxvar,"ave/chunk:varatom");
912         }
913
914         input->variable->compute_atom(n,igroup,varatom,1,0);
915
916         for (i = 0; i < nlocal; i++)
917           if (mask[i] & groupbit && ichunk[i] > 0) {
918             index = ichunk[i]-1;
919             values_one[index][m] += varatom[i];
920           }
921       }
922     }
923
924     // process the current sample
925     // if normflag = ALL, accumulate values,count separately to many
926     // if normflag = SAMPLE, one = value/count, accumulate one to many
927     //   count is MPI summed here, value is MPI summed below across samples
928     //   exception is TEMPERATURE: normalize by DOF
929     //   exception is DENSITYs: no normalize by atom count
930     //   exception is scaleflag = NOSCALE : no normalize by atom count
931     //     check last so other options can take precedence
932
933     double mvv2e = force->mvv2e;
934     double boltz = force->boltz;
935
936     if (normflag == ALL) {
937       for (m = 0; m < nchunk; m++) {
938         count_many[m] += count_one[m];
939         if (adofflag==1){
940           for (int in = 0; in < nname; in++) {
```

```
941             count_many_add[in][m] += count_one_add[in][m];
942           }
943         }
944         for (j = 0; j < nvalues; j++)
945           values_many[m][j] += values_one[m][j];
946       }
947
948     } else if (normflag == SAMPLE) {
949       MPI_Allreduce(count_one,count_many,nchunk,MPI_DOUBLE,MPI_SUM,world);
950
951       if (adofflag==1){
952         for (int in = 0; in < nname; in++){
953           MPI_Allreduce(count_one_add[in],count_many_add[in],nchunk,MPI_DOUBLE,MPI_SUM,world);
954         }
955       }
956
957       for (m = 0; m < nchunk; m++) {
958           for (j = 0; j < nvalues; j++) {
959             if (which[j] == TEMPERATURE)
960               if (adofflag==1){
961                 double dof_temp=cdof;
962                 for (int in =0; in < nname; in++){
963                   dof_temp+=adof[in]*count_many_add[in][m];
964                 }
965                 if (dof_temp > 0)
966                   values_many[m][j] += mvv2e*values_one[m][j] /(dof_temp * boltz);
967
968               }else{
969                 double dof_temp;
970                 dof_temp=domain->dimension;
971                 if (count_many[m] > 0)
972                 values_many[m][j] += mvv2e*values_one[m][j] /
973                   ((cdof + dof_temp*count_many[m]) * boltz);
974               }
975
976             else if (which[j] == DENSITY_NUMBER || which[j] == DENSITY_MASS ||
977                      scaleflag == NOSCALE)
978               values_many[m][j] += values_one[m][j];
979             else {
980               if (adofflag==1){
981                 double count_many_temp=0;
```

```
982                 for (int in =0; in < nname; in++){
983                   count_many_temp += count_many_add[in][m];
984                 }
985                 if (count_many_temp>0) values_many[m][j] += values_one[m][j]/count_many_temp;
986               }else{
987                 if (count_many[m]>0) values_many[m][j] += values_one[m][j]/count_many[m];
988               }
989             }
990             }
991
992             if (adofflag==1){
993               for (int in =0; in < nname; in++){
994                 count_sum_add[in][m] += count_many_add[in][m];
995               }
996             }else{
997               count_sum[m] += count_many[m];
998             }
999       }
1000    }
1001
1002    // done if irepeat < nrepeat
1003    // else reset irepeat and nvalid
1004
1005    irepeat++;
1006    if (irepeat < nrepeat) {
1007      nvalid += nevery;
1008      modify->addstep_compute(nvalid);
1009      return;
1010    }
1011
1012    irepeat = 0;
1013    nvalid = ntimestep+nfreq - (nrepeat-1)*nevery;
1014    modify->addstep_compute(nvalid);
1015
1016    // unlock compute chunk/atom at end of Nfreq epoch
1017    // do not unlock if ave = RUNNING or WINDOW
1018
1019    if (nrepeat > 1 && ave == ONE) cchunk->unlock(this);
1020
1021    // time average across samples
1022    // if normflag = ALL, final is total value / total count
```

```
1023    //   exception is TEMPERATURE: normalize by DOF for total count
1024    //   exception is DENSITYs: normalize by repeat, not total count
1025    //   exception is scaleflag == NOSCALE: normalize by repeat, not total count
1026    //     check last so other options can take precedence
1027    // if normflag = SAMPLE, final is sum of ave / repeat
1028
1029    double repeat = nrepeat;
1030    double mv2d = force->mv2d;
1031
1032    if (normflag == ALL) {
1033      MPI_Allreduce(count_many,count_sum,nchunk,MPI_DOUBLE,MPI_SUM,world);
1034      if(adofflag==1){
1035        for (int in = 0; in < nname; in++){
1036          MPI_Allreduce(count_many_add[in],count_sum_add[in],nchunk,MPI_DOUBLE,MPI_SUM,world);
1037        }
1038      }
1039      MPI_Allreduce(&values_many[0][0],&values_sum[0][0],nchunk*nvalues,
1040                    MPI_DOUBLE,MPI_SUM,world);
1041
1042      for (m = 0; m < nchunk; m++) {
1043          for (j = 0; j < nvalues; j++) {
1044            if (which[j] == TEMPERATURE)
1045              if (adofflag==1){
1046
1047                double dof_temp=cdof;
1048                for (int in =0; in < nname; in++){
1049                  dof_temp += adof[in]*count_sum_add[in][m];
1050                }
1051
1052                if (dof_temp > 0)
1053                values_sum[m][j] *= mvv2e /(dof_temp * boltz);
1054
1055              }else{
1056                double dof_temp;
1057                if (count_sum[m] > 0)
1058                dof_temp=domain->dimension;
1059                values_sum[m][j] *= mvv2e / (( cdof + dof_temp*count_sum[m] ) * boltz);
1060              }
1061            else if (which[j] == DENSITY_MASS)
1062              values_sum[m][j] *= mv2d/repeat;
1063            else if (which[j] == DENSITY_NUMBER || scaleflag == NOSCALE)
```

```
            values_sum[m][j] /= repeat;
          else {

            if (adofflag==1){
              double count_sum_temp=0;
              for (int in =0; in < nname; in++){
                count_sum_temp += count_sum_add[in][m];
              }
              if (count_sum_temp>0) values_sum[m][j] /= count_sum_temp;
            }else{
              if (count_sum[m]>0) values_sum[m][j] /= count_sum[m];
            }

          }
        }
        if (adofflag==1){
          for (int in =0; in < nname; in++)
            count_sum_add[in][m] /= repeat;
        }else{
          count_sum[m] /= repeat;
        }
    }
  } else if (normflag == SAMPLE) {
    MPI_Allreduce(&values_many[0][0],&values_sum[0][0],nchunk*nvalues,
                  MPI_DOUBLE,MPI_SUM,world);
    for (m = 0; m < nchunk; m++) {
      for (j = 0; j < nvalues; j++) values_sum[m][j] /= repeat;
        if (adofflag==1){
          for (int in =0; in < nname; in++)
            count_sum_add[in][m] /= repeat;
        }else{
          count_sum[m] /= repeat;
        }
    }
  }

  // DENSITYs are additionally normalized by chunk volume
  // use scalar or vector values for volume(s)
  // if chunks are not spatial bins, chunk_volume_scalar = 1.0

  for (j = 0; j < nvalues; j++)
```

```
1105      if (which[j] == DENSITY_NUMBER || which[j] == DENSITY_MASS) {
1106        if (cchunk->chunk_volume_vec) {
1107          double *chunk_volume_vec = cchunk->chunk_volume_vec;
1108          for (m = 0; m < nchunk; m++)
1109            values_sum[m][j] /= chunk_volume_vec[m];
1110        } else {
1111          double chunk_volume_scalar = cchunk->chunk_volume_scalar;
1112          for (m = 0; m < nchunk; m++)
1113            values_sum[m][j] /= chunk_volume_scalar;
1114        }
1115      }
1116
1117    // if ave = ONE, only single Nfreq timestep value is needed
1118    // if ave = RUNNING, combine with all previous Nfreq timestep values
1119    // if ave = WINDOW, comine with nwindow most recent Nfreq timestep values
1120
1121    if (ave == ONE) {
1122      for (m = 0; m < nchunk; m++) {
1123        for (i = 0; i < nvalues; i++)
1124          values_total[m][i] = values_sum[m][i];
1125
1126        if (adofflag==1){
1127          double count_sum_temp=0;
1128          for (int in =0; in < nname; in++){
1129            count_sum_temp += count_sum_add[in][m];
1130          }
1131            count_total[m] = count_sum_temp;
1132        }else{
1133          count_total[m] = count_sum[m];
1134        }
1135
1136      }
1137      normcount = 1;
1138
1139    } else if (ave == RUNNING) {
1140      for (m = 0; m < nchunk; m++) {
1141        for (i = 0; i < nvalues; i++)
1142          values_total[m][i] += values_sum[m][i];
1143
1144        if (adofflag==1){
1145          double count_sum_temp=0;
```

```
1146          for (int in =0; in < nname; in++){
1147            count_sum_temp += count_sum_add[in][m];
1148          }
1149            count_total[m] += count_sum_temp;
1150        }else{
1151          count_total[m] += count_sum[m];
1152        }
1153
1154      }
1155      normcount++;
1156
1157    } else if (ave == WINDOW) {
1158      for (m = 0; m < nchunk; m++) {
1159        for (i = 0; i < nvalues; i++) {
1160          values_total[m][i] += values_sum[m][i];
1161          if (window_limit) values_total[m][i] -= values_list[iwindow][m][i];
1162          values_list[iwindow][m][i] = values_sum[m][i];
1163        }
1164
1165        if (adofflag==1){
1166          double count_sum_temp=0;
1167          for (int in =0; in < nname; in++){
1168            count_sum_temp += count_sum_add[in][m];
1169          }
1170          count_total[m] += count_sum_temp;
1171        }else{
1172          count_total[m] += count_sum[m];
1173        }
1174
1175        if (window_limit) count_total[m] -= count_list[iwindow][m];
1176        if (adofflag==1){
1177          double count_sum_temp=0;
1178          for (int in =0; in < nname; in++){
1179            count_sum_temp += count_sum_add[in][m];
1180          }
1181            count_list[iwindow][m] = count_sum_temp;
1182        }else{
1183          count_list[iwindow][m] = count_sum[m];
1184        }
1185      }
1186
```

```
1187      iwindow++;
1188      if (iwindow == nwindow) {
1189        iwindow = 0;
1190        window_limit = 1;
1191      }
1192      if (window_limit) normcount = nwindow;
1193      else normcount = iwindow;
1194    }
1195
1196    // output result to file
1197
1198    if (fp && me == 0) {
1199      clearerr(fp);
1200      if (overwrite) fseek(fp,filepos,SEEK_SET);
1201      double count = 0;
1202      for (m = 0; m < nchunk; m++) count += count_total[m];
1203      fprintf(fp,BIGINT_FORMAT " %d %g\n",ntimestep,nchunk,count);
1204
1205      int compress = cchunk->compress;
1206      int *chunkID = cchunk->chunkID;
1207      int ncoord = cchunk->ncoord;
1208      double **coord = cchunk->coord;
1209
1210      if (!compress) {
1211        if (ncoord == 0) {
1212          for (m = 0; m < nchunk; m++) {
1213            fprintf(fp,"  %d %g",m+1,count_total[m]/normcount);
1214            for (i = 0; i < nvalues; i++)
1215              fprintf(fp,format,values_total[m][i]/normcount);
1216            fprintf(fp,"\n");
1217          }
1218        // bin/1d
1219        } else if (ncoord == 1) {
1220          for (m = 0; m < nchunk; m++) {
1221            fprintf(fp,"  %d %g %g",m+1,coord[m][0],
1222                    count_total[m]/normcount);
1223            for (i = 0; i < nvalues; i++)
1224                fprintf(fp,format,values_total[m][i]/normcount);
1225            fprintf(fp,"\n");
1226          }
1227        // bin/2d
```

```
1228        } else if (ncoord == 2) {
1229          for (m = 0; m < nchunk; m++) {
1230            fprintf(fp,"  %d %g %g %g",m+1,coord[m][0],coord[m][1],
1231                    count_total[m]/normcount);
1232            for (i = 0; i < nvalues; i++)
1233              fprintf(fp,format,values_total[m][i]/normcount);
1234            fprintf(fp,"\n");
1235          }
1236        // bin/3d
1237        } else if (ncoord == 3) {
1238          for (m = 0; m < nchunk; m++) {
1239            fprintf(fp,"  %d %g %g %g %g %g %g",m+1,
1240                    coord[m][0],coord[m][1],coord[m][2],count_total[m]/normcount);
1241            for (i = 0; i < nvalues; i++)
1242              fprintf(fp,format,values_total[m][i]/normcount);
1243            fprintf(fp,"\n");
1244          }
1245        }
1246      } else {
1247        int j;
1248        if (ncoord == 0) {
1249          for (m = 0; m < nchunk; m++) {
1250            fprintf(fp,"  %d %d %g",m+1,chunkID[m],count_total[m]/normcount);
1251            for (i = 0; i < nvalues; i++)
1252              fprintf(fp,format,values_total[m][i]/normcount);
1253            fprintf(fp,"\n");
1254          }
1255        } else if (ncoord == 1) {
1256          for (m = 0; m < nchunk; m++) {
1257            j = chunkID[m];
1258            fprintf(fp,"  %d %d %g %g",m+1,j,coord[j-1][0],
1259                    count_total[m]/normcount);
1260            for (i = 0; i < nvalues; i++)
1261              fprintf(fp,format,values_total[m][i]/normcount);
1262            fprintf(fp,"\n");
1263          }
1264        } else if (ncoord == 2) {
1265          for (m = 0; m < nchunk; m++) {
1266            j = chunkID[m];
1267            fprintf(fp,"  %d %d %g %g %g",m+1,j,coord[j-1][0],coord[j-1][1],
1268                    count_total[m]/normcount);
```

```
            for (i = 0; i < nvalues; i++)
              fprintf(fp,format,values_total[m][i]/normcount);
            fprintf(fp,"\n");
          }
        } else if (ncoord == 3) {
          for (m = 0; m < nchunk; m++) {
            j = chunkID[m];
            fprintf(fp,"  %d %d %g %g %g %g",m+1,j,coord[j-1][0],
                    coord[j-1][1],coord[j-1][2],count_total[m]/normcount);
            for (i = 0; i < nvalues; i++)
              fprintf(fp,format,values_total[m][i]/normcount);
            fprintf(fp,"\n");
          }
        }
      }
      if (ferror(fp))
        error->one(FLERR,"Error writing averaged chunk data");

      fflush(fp);

      if (overwrite) {
        long fileend = ftell(fp);
        if (fileend > 0) ftruncate(fileno(fp),fileend);
      }
    }
  }

  /* ----------------------------------------------------------------------
     allocate all per-chunk vectors
  ------------------------------------------------------------------------ */

  void FixAveChunk::allocate()
  {
    size_array_rows = nchunk;

    // reallocate chunk arrays if needed

    if (nchunk > maxchunk) {
      maxchunk = nchunk;
      memory->grow(count_one,nchunk,"ave/chunk:count_one");
      memory->grow(count_many,nchunk,"ave/chunk:count_many");
```

```
    memory->grow(count_sum,nchunk,"ave/chunk:count_sum");

    if (adofflag==1){
        memory->grow(count_many_add,nadof,nchunk,"ave/chunk:count_many_add");
        memory->grow(count_one_add,nadof,nchunk,"ave/chunk:count_one_add");
        memory->grow(count_sum_add,nadof,nchunk,"ave/chunk:count_sum_add");
    }

    memory->grow(count_total,nchunk,"ave/chunk:count_total");
    memory->grow(values_one,nchunk,nvalues,"ave/chunk:values_one");
    memory->grow(values_many,nchunk,nvalues,"ave/chunk:values_many");
    memory->grow(values_sum,nchunk,nvalues,"ave/chunk:values_sum");
    memory->grow(values_total,nchunk,nvalues,"ave/chunk:values_total");

    // only allocate count and values list for ave = WINDOW

    if (ave == WINDOW) {
      memory->create(count_list,nwindow,nchunk,"ave/chunk:count_list");
      memory->create(values_list,nwindow,nchunk,nvalues,
                     "ave/chunk:values_list");
    }

    // reinitialize regrown count/values total since they accumulate

    int i,m;
    for (m = 0; m < nchunk; m++) {
      for (i = 0; i < nvalues; i++) values_total[m][i] = 0;
      count_total[m] = 0;
    }
  }
}

/* ----------------------------------------------------------------------
   return I,J array value
   if I exceeds current nchunks, return 0.0 instead of generating an error
   columns 1 to colextra = chunkID + ncoord
   next column = count, remaining columns = Nvalues
------------------------------------------------------------------------- */

double FixAveChunk::compute_array(int i, int j)
{
```

```
1351    if (values_total == NULL) return 0;
1352    if (i >= nchunk) return 0;
1353    if (j < colextra) {
1354      if (cchunk->compress) {
1355        if (j == 0) return (double) cchunk->chunkID[i];
1356        return cchunk->coord[i][j-1];
1357      } else return cchunk->coord[i][j];
1358    }
1359    j -= colextra + 1;
1360    if (!normcount) return 0;
1361    if (j < 0) return count_total[i]/normcount;
1362    return values_total[i][j]/normcount;
1363  }
1364
1365  /* ----------------------------------------------------------------------
1366     calculate nvalid = next step on which end_of_step does something
1367     can be this timestep if multiple of nfreq and nrepeat = 1
1368     else backup from next multiple of nfreq
1369  ------------------------------------------------------------------------- */
1370
1371  bigint FixAveChunk::nextvalid()
1372  {
1373    bigint nvalid = (update->ntimestep/nfreq)*nfreq + nfreq;
1374    if (nvalid-nfreq == update->ntimestep && nrepeat == 1)
1375      nvalid = update->ntimestep;
1376    else
1377      nvalid -= (nrepeat-1)*nevery;
1378    if (nvalid < update->ntimestep) nvalid += nfreq;
1379    return nvalid;
1380  }
1381
1382  /* ----------------------------------------------------------------------
1383     memory usage of varatom and bins
1384  ------------------------------------------------------------------------- */
1385
1386  double FixAveChunk::memory_usage()
1387  {
1388    double bytes = maxvar * sizeof(double);            // varatom
1389    bytes += 4*maxchunk * sizeof(double);              // count_one,many,sum,total
1390    if (adofflag==1){
1391      bytes += 3*nadof*maxchunk * sizeof(double);    // (count_one_add,many_add,sum_add)*nadof
```

```
1392      if (comflag==1){
1393        bytes += 6*maxchunk * sizeof(double);          // com, comall, Tmol, Tmolall
1394                                                       // masstotal, masstotalall
1395      }
1396    }
1397    bytes += nvalues*maxchunk * sizeof(double);          // values one,many,sum,total
1398    bytes += nwindow*maxchunk * sizeof(double);          // count_list
1399    bytes += nwindow*maxchunk*nvalues * sizeof(double);  // values_list
1400    return bytes;
1401  }
```

6.1.2　インプットファイル

Listing 6.3　input

```
1  //原子温度
2  compute cchT all chunk/atom bin/1d z lower ${dz}
3  fix fchT all ave/chunk n1 n2 n3 cchT temp adof 2 2.5 2
4                                  name 2 oxygen water file "./output/out.temp"
5
6  //分子温度
7  fix fchT all ave/chunk n1 n2 n3 cchT temp adof 2 5 6
8      name 2 oxygen water com 6332 2 1 2149 2148 6332 file "./output/out.temp"
```

参考文献

[1] Ilan Benjamin. I. mechanism and dynamics of ion transfer across a liquid-liquid interface. *Science*, Vol. 261, pp. 1558–1560, 1993.

[2] KJ Schweighofer and Ilan Benjamin. Transfer of small ions across the water/1, 2-dichloroethane interface. *The Journal of Physical Chemistry*, Vol. 99, pp. 9974–9985, 1995.

[3] Mária Darvas, Miguel Jorge, M Natalia DS Cordeiro, and Pál Jedlovszky. Solvation free energy profile of the scn–ion across the water–1, 2-dichloroethane liquid/liquid interface. a computer simulation study. *J. Phys. Chem. C*, Vol. 115, No. 22, pp. 11140–11146, 2011.

[4] Mária Darvas, Miguel Jorge, M Natalia DS Cordeiro, Sofia S Kantorovich, Marcello Sega, and Pál Jedlovszky. Calculation of the intrinsic solvation free energy profile of an ionic penetrant across a liquid–liquid interface with computer simulations. *J. Phys. Chem. B*, Vol. 117, No. 50, pp. 16148–16156, 2013.

[5] Nobuaki Kikkawa, Lingjian Wang, and Akihiro Morita. Microscopic barrier mechanism of ion transport through liquid–liquid interface. *Journal of the American Chemical Society*, Vol. 137, No. 25, pp. 8022–8025, 2015.

[6] Nobuaki Kikkawa, Lingjian Wang, and Akihiro Morita. Computational study of effect of water finger on ion transport through water-oil interface. *The Journal of chemical physics*, Vol. 145, No. 1, p. 014702, 2016.

[7] François O Laforge, Peng Sun, and Michael V Mirkin. Shuttling mechanism of ion transfer at the interface between two immiscible liquids. *Journal of the American Chemical Society*, Vol. 128, No. 46, pp. 15019–15025, 2006.

[8] T Kenjo and R.M. Diamond. ydration of the halide ions in certain organic solvents. *Journal of Inorganic and Nuclear Chemistry*, Vol. 36, pp. 183–188, 1974.

[9] Lingjian Wang, Nobuaki Kikkawa, and Akihiro Morita. Hydrated ion clusters in hydrophobic liquid: Equilibrium distribution, kinetics, and implications. *The Journal of physical chemistry B*, Vol. 122, pp. 3562–3571, 2018.

[10] Kyuichi Yasui, Toru Tuziuti, and Wataru Kanematsu. High temperature and pressure inside a dissolving oxygen nanobubble. *Ultrasonics sonochemistry*, Vol. 55, pp. 308–312, 2019.

[11] Shuichi Nosé. A unified formulation of the constant temperature molecular dynamics methods. *The Journal of chemical physics*, Vol. 81, No. 1, pp. 511–519, 1984.

[12] William G Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Physical review A*, Vol. 31, No. 3, p. 1695, 1985.

[13] Daan Frenkel, Berend Smit, Jan Tobochnik, Susan R McKay, and Wolfgang Christian. Understanding molecular simulation. *Computers in Physics*, Vol. 11, No. 4, pp. 351–354, 1997.

[14] Hans C Andersen. Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations.

Journal of computational Physics, Vol. 52, No. 1, pp. 24–34, 1983.

[15] Michael P Allen and Dominic J Tildesley. *Some tricks of the trade*. Oxford University Press, 1987.

[16] Tom Darden, Darrin York, and Lee Pedersen. Particle mesh ewald: An n log (n) method for ewald sums in large systems. *The Journal of chemical physics*, Vol. 98, No. 12, pp. 10089–10092, 1993.

[17] Abdulnour Toukmaji, Celeste Sagui, John Board, and Tom Darden. Efficient particle-mesh ewald based approach to fixed and induced dipolar interactions. *The Journal of chemical physics*, Vol. 113, No. 24, pp. 10913–10927, 2000.

[18] Y Sugita, A Kitao, and Y Okamoto. Multidimensional replica-exchange method for free-energy calculations. *Journal of Chemical Physics*, Vol. 113, No. 15, pp. 6042–6051, 2000.

[19] S Kumar, J. M. Rosenberg, Djamal Bouzida, Robert H. Swendsen, and Peter A. Kollman. The weighted histogram analysis method for free-energy calculations on biomolecules. i. the method. *Journal of Computational Chemistry*, Vol. 13, No. 2, pp. 1011–1021, 1992.

[20] Hubert H. Girault. Dynamics and structure of the liquid–liquid interface. *Faraday Discussions*, Vol. 129, pp. 1–370, 2005.

[21] Hubert H. Girault. *Electrochemistry at Liquid-Liquid Interfaces*, Vol. 23, pp. 1–104. CRC Press, Boca Raton, FL, 1st ed edition, 2010.

[22] R. A. W. Dryfe. The electrified liquid-liquid interface. *Advances in Chemical Physics*, Vol. 141, pp. 153–215, 2009.

[23] Shujuan Liu, Qing Li, and Yuanhua Shao. Electrochemistry at micro-and nanoscopic liquid/liquid interfaces. *Chemical Society Reviews*, Vol. 40, No. 5, pp. 2236–2253, 2011.

[24] Z. Samec and T. Kakiuchi. *Charge Transfer Kinetics at Water-Organic Solvent Phase Boundaries*, Vol. 4, pp. 297–361. Wiley, VCH, New York, 2008.

[25] Eric Bakker, Philippe Bühlmann, and Ernö Pretsch. Carrier-based ion-selective electrodes and bulk optodes. 1. general characteristics. *Chemical Reviews*, Vol. 97, No. 8, pp. 3083–3132, 1997.

[26] Charles M Starks, Charles L Liotta, and Marc E Halpern. *Phase-Transfer Catalysts*. Springer, 1994.

[27] Philippe Bühlmann, Ernö Pretsch, and Eric Bakker. Carrier-based ion-selective electrodes and bulk optodes. 2. ionophores for potentiometric and optical sensors. *Chemical Reviews*, Vol. 98, No. 4, pp. 1593–1688, 1998.

[28] Johan Bobacka, Ari Ivaska, and Andrzej Lewenstam. Potentiometric ion sensors. *Chemical reviews*, Vol. 108, No. 2, pp. 329–351, 2008.

[29] Philippe Bühlmann and Li D Chen. *Ion-Selective Electrodes With Ionophore-Doped Sensing Membranes*, Vol. 5, pp. 2539–2579. Wiley, 2012.

[30] Frédéric Reymond, David Fermın, Hye Jin Lee, and Hubert H Girault. Electrochemistry at liquid/liquid interfaces: methodology and potential applications. *Electrochimica acta*, Vol. 45, No. 15-16, pp. 2647–2662, 2000.

[31] Biao Liu and Michael V Mirkin. Electrochemistry at microscopic liquid–liquid interfaces. *Electroanalysis: An International Journal Devoted to Fundamental and Practical Aspects of Electroanalysis*, Vol. 12, No. 18, pp. 1433–1446, 2000.

[32] Celeste A Morris, Alicia K Friedman, and Lane A Baker. Applications of nanopipettes in the analytical sciences. *Analyst*, Vol. 135, No. 9, pp. 2190–2202, 2010.

[33] D Homolka, Le Quoc Hung, A Hofmanova, MW Khalil, J Koryta, V Marecek, Z Samec, SK Sen, P Vanysek, et al. Faradaic ion transfer across the interface of two immiscible electrolyte solutions: chronopotentiometry

and cyclic voltammetry. *Analytical Chemistry*, Vol. 52, No. 11, pp. 1606–1610, 1980.

[34] Zdeněk Samec, Daniel Homolka, and Vladimir Mareček. Charge transfer between two immiscible electrolyte solutions part viii. transfer of alkali and alkaline earth-metal cations across the water/nitrobenzene interface facilitated by synthetic neutral ion carriers. *Journal of Electroanalytical Chemistry and Interfacial Electrochemistry*, Vol. 135, No. 2, pp. 265–283, 1982.

[35] Tadaaki Kakutani, Yoshinori Nishiwaki, Toshiyuki Osakai, and Mitsugi Senda. On the mechanism of transfer of sodium ion across the nitrobenzene/water interface facilitated by dibenzo-18-crown-6. *Bulletin of the Chemical Society of Japan*, Vol. 59, No. 3, pp. 781–788, 1986.

[36] Y Shao, MD Osborne, and HH Girault. Assisted ion transfer at micro-ities supported at the tip of micropipettes. *Journal of electroanalytical chemistry and interfacial electrochemistry*, Vol. 318, No. 1-2, pp. 101–109, 1991.

[37] Lin Sinru, Zhao Zaofan, and Henry Freiser. Potassium ion transport processes across the interface of an immiscible liquid pair in the presence of crown ethers. *Journal of electroanalytical chemistry and interfacial electrochemistry*, Vol. 210, No. 1, pp. 137–146, 1986.

[38] Shigeru Amemiya, Jiyeon Kim, Anahita Izadyar, Benjamin Kabagambe, Mei Shen, and Ryoichi Ishimatsu. Electrochemical sensing and imaging based on ion transfer at liquid/liquid interfaces. *Electrochimica acta*, Vol. 110, pp. 836–845, 2013.

[39] MHM Caçote, CM Pereira, L Tomaszewski, HH Girault, and F Silva. Ag+ transfer across the water/1, 2-dichloroethane interface facilitated by complex formation with tetraphenylborate derivatives. *Electrochimica acta*, Vol. 49, No. 2, pp. 263–270, 2004.

[40] Brian J Kirby and Pavel Jungwirth. Charge scaling manifesto: A way of reconciling the inherently macroscopic and microscopic natures of molecular simulations. *The journal of physical chemistry letters*, Vol. 10, No. 23, pp. 7531–7536, 2019.

[41] Lijiang Yang, Chun-hu Tan, Meng-Juei Hsieh, Junmei Wang, Yong Duan, Piotr Cieplak, James Caldwell, Peter A Kollman, and Ray Luo. New-generation amber united-atom force field. *The journal of physical chemistry B*, Vol. 110, No. 26, pp. 13166–13176, 2006.

[42] Xiangwen Wang, Dimitrios Toroz, Seonmyeong Kim, Simon L Clegg, Gun-Sik Park, and Devis Di Tommaso. Density functional theory based molecular dynamics study of solution composition effects on the solvation shell of metal ions. *Physical Chemistry Chemical Physics*, Vol. 22, No. 28, pp. 16301–16313, 2020.

[43] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, Jr. J. A. Montgomery, J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, T. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, O. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, and D. J. Fox. Gaussian 09, revision A.02, 2016. Gaussian, Inc., Wallingford CT.

[44] DS Otkidach and IV Pletnev. Conformational analysis of boron-containing compounds using gillespie–kepert

version of molecular mechanics. *Journal of Molecular Structure: THEOCHEM*, Vol. 536, No. 1, pp. 65–72, 2001.

[45] HJC Berendsen, JR Grigera, and TP Straatsma. The missing term in effective pair potentials. *Journal of Physical Chemistry*, Vol. 91, No. 24, pp. 6269–6271, 1987.

[46] Swaroop Chatterjee, Pablo G Debenedetti, Frank H Stillinger, and Ruth M Lynden-Bell. A computational investigation of thermodynamics, structure, dynamics and solvation behavior in modified water models. *The Journal of chemical physics*, Vol. 128, No. 12, p. 124511, 2008.

[47] Liem X Dang and David Feller. Molecular dynamics study of water- benzene interactions at the liquid/vapor interface of water. *The Journal of Physical Chemistry B*, Vol. 104, No. 18, pp. 4403–4407, 2000.

[48] James W Caldwell and Peter A Kollman. Structure and properties of neat liquids using nonadditive molecular dynamics: water, methanol, and n-methylacetamide. *The Journal of Physical Chemistry*, Vol. 99, No. 16, pp. 6208–6219, 1995.

[49] Koichi Ohno and Satoshi Maeda. Global reaction route mapping on potential energy surfaces of formaldehyde, formic acid, and their metal-substituted analogues. *The Journal of Physical Chemistry A*, Vol. 110, No. 28, pp. 8933–8941, 2006.

[50] Liem X Dang. A mechanism for ion transport across the water/dichloromethane interface: a molecular dynamics study using polarizable potential models. *The Journal of Physical Chemistry B*, Vol. 105, pp. 804–809, 2001.

[51] Jun Wang, Piotr Cieplak, Qin Cai, Meng-Juei Hsieh, Junmei Wang, Yong Duan, and Ray Luo. Development of polarizable models for molecular mechanical calculations. 3. polarizable water models conforming to thole polarization screening schemes. *The Journal of Physical Chemistry B*, Vol. 116, No. 28, pp. 7999–8008, 2012.

[52] David R. Lide. *CRC Handbook of Chemistry and Physics*. CRC Press, 85th ed edition, 2004.

[53] Akihiro Morita and Shigeki Kato. An ab initio analysis of medium perturbation on molecular polarizabilities. *J. Chem. Phys.*, Vol. 110, pp. 11987–11998, 1999.

[54] Jon Applequist, James R. Carl, and Kwok-Keung Fung. Atom dipole interaction model for molecular polarizability. application to polyatomic molecules and determination of atom polarizabilities. *J. Am. Chem. Soc.*, pp. 2952–2960, 1972.

[55] Kazuo Kitaura and Keiji Morokuma. A new energy decomposition scheme for molecular interactions within the hartree-fock approximation. *International Journal of Quantum Chemistry*, Vol. 10, No. 2, pp. 325–340, 1976.

[56] Michael W. Schmidt, Kim K. Baldridge, Jerry A. Boatz, Steven T. Elbert, Mark S. Gordon, Jan H. Jensen, Shiro Koseki, Nikita Matsunaga, Kiet A. Nguyen, Shujun Su, Theresa L. Windus, Michel Dupuis, and John A. Montgomery Jr. General atomic and molecular electronic structure system. *J. Comput. Chem.*, Vol. 14, pp. 1347–1363, 1993.

[57] Takashi Kakiuchi, Masatoshi Nakanishi, and Mitsugi Senda. The electrocapillary curves of the phosphatidylcholine monolayer at the polarized oil–water interface. i. measurement of interfacial tension using a computer-aided pendant-drop method. *Bulletin of the Chemical Society of Japan*, Vol. 61, No. 6, pp. 1845–1851, 1988.