高精度分子シミュレーションによる転写制御タンパク質ラクトースリプレッサーと DNA 間の特異的相互作用の解析

２０１３年１月

博士（工学）

大山　達也

豊橋技術科学大学

高精度分子シミュレーションによる転写制御タンパク質

ラクトースリプレッサーと DNA 間の特異的相互作用の解析

［要旨］

生体の主な構成要素であり、生体機能を制御しているタンパク質は、DNA に蓄えられた遺伝情報を基に、DNA から RNA への転写機構、及び RNA からタンパク質への翻訳機構を経て合成される。このタンパク質の合成機構、及びこの機構を制御するタンパク質の機能に関しては、これまでの実験により、その概要は明らかにされているが、転写や翻訳を制御するタンパク質と DNA 間の生体内での相互作用機構、あるいは、生体内での環境変化がこれらの機構に及ぼす影響については、未解明である。本研究の対象である転写制御タンパク質ラクトースリプレッサー(LacR)に関しても、生体内での様々なリガンドの有無に応じて、転写機構を巧みに制御することは明らかになっているが、その機構は原子・電子レベルでは未解明である。通常、LacR は DNA に結合し、転写を抑制しているが、周囲に存在するインデューサが LacR に結合すると、LacR と DNA 間の結合親和性が弱まり、LacR が DNA から分離する。その結果、それまで抑制されていた DNA の転写が開始される。一方、アンチインデューサが LacR に結合すると、インデューサとは逆に LacR と DNA 間の結合親和性が強まり、転写をさらに抑制することが分かっている。しかし、LacR のリガンド結合位置と DNA 結合位置は、約 30 Å 以上離れており、LacR にリガンドが結合した影響が、LacR と DNA 間の結合親和性にどのように伝わるかは、原子・電子レベルでは未解明である。この現象が解明できれば、生命現象の中で最も基本的な現象である転写機構に関して、新たしい知見を得ることができると考える。

本研究では、LacR+DNA+リガンド複合体の実験構造を基に、単量体及び二量体の複合体を作成し、さらに、リガンドが結合していない複合体、インデューサ IPTG (Isopropyl-β-D-thiogalactpyranose)、あるいは、アンチインデューサ ONPF (o-nitrophenyl-fucoside) が結合した複合体を作成した。それらの構造を、古典分子力学(MM)及び分子動力学(MD)法により、水中で最適化し、*ab initio* フラグメント分子軌道(FMO)法を用い、最適化構造の電子状態を解析し、LacR へのリガンド結合が LacR と DNA 間の特異的結合特性、結合親和性にどのような影響を与えるかを、電子レベルで解明した。

まず、長時間 30 ns の MD 計算を実行し、複合体の構造変化を広範囲に探索し、LacR に結合するリガンドの種類による、LacR と DNA 間の結合部位周辺の構造変化を明らかにした。さらに、FMO 計算により、LacR の各アミノ酸残基とリガンド及び DNA 間の特異的相互作用を解析し、実験で重要とされていた Arg22、Asp149、Asn246 が、LacR

とリガンド間の結合に重要であることを明らかにした。さらに、LacR と DNA 間の相互作用には、LacR のアミノ酸残基、及び DNA の塩基 Thy14、Gua15 が重要であることを明らかにした。これらの結果を基に、LacR への様々なリガンドの結合が、LacR と DNA 間の特異的相互作用に与える影響に関する新たなモデルを提案した。

Analysis of specific interaction between lactose repressor and DNA by structural and electronic property analysis

[Abstract]

The transcription of genetic information from DNA to mRNA is the first and essential process in a series of mechanisms for gene expression. In all living organisms, the transcription is precisely regulated by many types of regulatory proteins as well as small ligand molecules to respond to the perpetual change of biological conditions in a cell. Lactose repressor (LacR) is one of the regulatory proteins and controls the transcriptional mechanism in a ligand-dependent manner. Although the ligand-binding to LacR was found to change the mechanism drastically, the effect of ligand-binding on the conformation of LacR-DNA complex has not been clarified at atomic and electronic levels. LacR generally regulates the transcription mechanism by binding to the specific site of DNA. On the other hand, the binding of inducer such as lactose and IPTG (Isopropyl-β-D-thiogalactopyranoside) to DNA weakens the binding affinity between LacR and DNA, leading to the separation of LacR from DNA. As a result, the transcription is started, and some lactose metabolizing enzymes are synthesized to metabolize lactose. In contrast, when anti-inducer such as ONPF (*o*-nitrophenylfucoside) binds to LacR, the binding affinity between LacR and DNA is enhanced and the transcription is more regulated by LacR. The transcriptional regulation mechanism of LacR is well known as mentioned above. However, it is difficult to explain the mechanism by the change in interactions between LacR and ligand, because the distance between the ligand binding site and the DNA binding site of LacR is about 30 Å. Hence we considered that it is needed to investigate drastically structural change of LacR for explaining the mechanism.

In this study, we first obtained the structure of the complex with LacR, DNA, anti-inducer ONPF from PDB (PDB code: 1EFA) and constructed the monomer and the dimer of the LacR-ONPF+DNA complex. The structures of the monomer and the dimer of the LacR+DNA complex, which is the complex without ligand, were constructed by deleting ONPF from the monomer and the dimer of LacR-ONPF+DNA. And then the monomer and the dimer of the LacR-IPTG+DNA complex were constructed by adding IPTG to the monomer and the dimer of the LacR+DNA complex. The position of IPTG was determined based on the experimental structure of the complex with the dimer LacR and IPTG (PDB code: 2P9H), which do not include DNA and the DNA binding domain of LacR. Solvating water molecules were added to these six complexes, and the solvated structures were optimized by the classical molecular mechanics (MM) method based on AMBER99SB-ILDN and TIP3P force fields, and the 30 ns molecular dynamics (MD) simulations were performed at 300 K under the periodic boundary condition to elucidate the conformational change of the complex. Furthermore, *Ab initio* fragment molecular orbital (FMO) calculations performed to investigate their electronic states

and specific interactions between LacR and DNA.

After the MM and MD simulations, the structures of the ligand pocket and the DNA binding domain of LacR are changed by the ligand-binding. And we investigated the conformational change and the interaction energies between ligand, amino acid residues of LacR, and solvating water molecules to elucidate the important amino acid residues for binding the ligand. Both of the IPTG and ONPF ligands form some hydrogen bonds to Asp149 and Asn246 of LacR. These residues were also indicated as important amino acid residues by the previous experiments. Especially Asp149 is important because it does not form hydrogen bond to the ligand, but interacts with the ligand electrostatically (the interaction energy is about -40 kcal/mol). And our simulations also clarified that some water molecules are important for binding between the ligand and LacR. For example, Asp274 of LacR interacts repulsively with ONPF (18.1 kcal/mol). However, Asp274 and ONPF interact with a common water molecule by -27.4 kcal/mol and -6.4 kcal/mol energies, respectively. As a result, Asp274 and ONPF interact attractively to each other through the common water molecule. The analysis for the DNA binding domain of LacR clarifies that Arg22 of LacR is important for the binding between LacR and DNA. Arg22 interacts electrostatically with Thymine14 and Guanine15 of DNA. Some experiments also reported that Arg22 is important for the specific recognition between LacR and DNA. Furthermore, we elucidated that the side-chain of Arg22 is also important because the interaction energies between Arg22 and these DNA bases are dependent significantly on the direction of Arg22 side-chain.

The structural analysis of the dimer complex elucidates a new mechanism explaining the effect of ligand-binding on the conformation of LacR+DNA complex. We investigated the conformational change of LacR+DNA induced by MD simulation. As the result, structure of LacR was not changed during 30 ns. However LacR-IPTG widely tilted in the direction of DNA helix at 7.7 ns, while LacR-ONPF widely tilted in the vertical direction of DNA helix at 22.5 ns. To clarify the cause LacR bound ligand tilted, we investigated displacement of Cα in LacR. As result, α-helix which exists near DNA-binding domain in other LacR monomer in LacR-IPTG was changed largely. The α-helix contains Asn125 and Asp149, which are important for binding ligand. These amino acid residues formed some hydrogen bonds with ONPF while they did not form any hydrogen bonds with IPTG. Therefore, we expeted that the α-helix is normally binding on ligand-binding pocket or anti-inducer while the α-helix is free by breaking hydrogen bond between ligand-pocket and the α-helix due to bind IPTG to LacR. And free α-helix forms some hydrogen bond with DNA-binding domain in other LacR monomer and LacR-IPTG tilts in the direction of DNA helix. Thus, these binding states between ligand-binding pocket and the α-helix affect the conformational change and the transcriptional mechanism of LacR.

[目次]

1. 序論

1.1. DNA 遺伝情報の転写制御機構

生物の体は、百個から、数千個以上のアミノ酸がペプチド結合したタンパク質で構成されている。タンパク質は、生物の体を構成するだけでなく、細胞内の様々な化学反応を効率良く進めたり、抑制したりする。そしてそれぞれのタンパク質は、セントラルドグマと呼ばれる転写、翻訳の過程を経て合成される。転写とは各生物が持つ DNA の遺伝情報を基に RNA ポリメラーゼと呼ばれるタンパク質が、メッセンジャーRNA を合成する過程である。翻訳は、転写で合成されたメッセンジャーRNA の情報を基にリボソームと呼ばれるタンパク質が、タンパク質を合成する過程である。

しかし、この機構だけでは、タンパク質は生物の体の中で合成され続けることになり、生命活動の維持ができなくなる。生物の体内では、このような状況にならないように、転写を制御する様々な転写制御タンパク質が存在する。

1.2. ラクトースリプレッサーによる転写制御機構

転写を制御する機構の一つとして、1961 年に Jacob、Monod らにより提唱されたラクトースオペロン(Lactose operon; lac operon)[1]の転写制御機構が挙げられる。これは、大腸菌内のラクトース(β-D-galactopyranosyl-1,4-D-glucose)を代謝する酵素に必要な情報を含んだ構造遺伝子の集合がラクトースリプレッサー(LacR)とオペレータにより、それら酵素の転写が支配されているというものである。

ラクトースオペロンはラクトース代謝酵素の発現に関わる DNA の部位の名称であり、LacR の構造遺伝子である *lacI*、調整領域である Promoter、LacR が結合する Operator、そして、ラクトースの代謝に関わる酵素の構造遺伝子領域である *lacZ*、*lacY*、*lacA* の順で構成される(Fig. 1)。このうち、ラクトースの代謝に直接関係する酵素は、*lacZ*、*lacY*、*lacA* の情報を基に合成される。*lacZ* はラクトース分解酵素(β-ガラクトシダーゼ; β-galactosidase)の情報を持ち、この酵素はラクトースを分解する。*lacY* はβ-ガラクトシドパーミアーゼ(β-galactoside permease)の情報を持ち、この酵素はラクトースを細胞内に引き入れるための透過酵素である。*lacA* は、β-ガラクトシドトランスアセチラーゼ(β-galactoside transacetylase)の情報を持ち、この酵素はラクトースが分解された際に生じるガラクトース(D-galactose)を分解、代謝する。LacR は、*lacI* から転写、翻訳の過程を経て、発現した後、最も親和性の高い Operator と結合する。Operator は転写方向の上流に位置するため、LacR が Operator に結合することにより、遺伝子の情報を転写する RNA ポリメラーゼは DNA に結合できない。これにより、DNA の情報は読み取られず、*lacZ*、*lacY*、*lacA* の発現は抑制される。

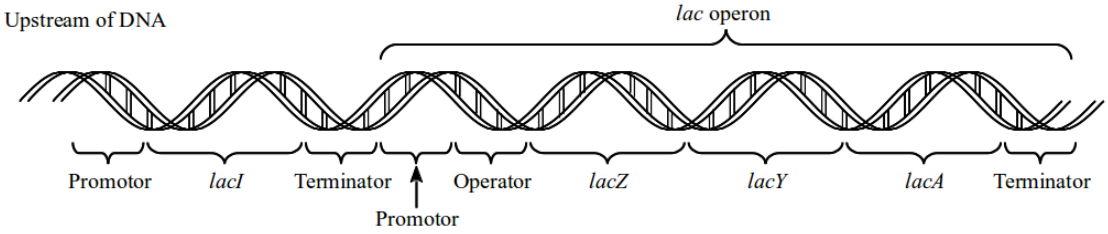


Fig. 1 A simplified overview of the genes and regulatory units involved in the control of lactose metabolism.

大腸菌は、周囲のグルコース(D-glucose)を第一のエネルギー源として代謝しているため、周囲にグルコースが大量に存在する状態では、ラクトースの代謝を必要としない。この時 LacR は DNA の Operator に結合した状態である。しかし、周囲のグルコースを代謝し尽くすと、大腸菌は第二のエネルギー源であるラクトースを代謝し始める。ラクトースはグルコース 1 分子とガラクトース 1 分子で構成されている。そのため、エネルギー源として活用するためには、一度グルコースとガラクトース間の結合を切断する必要がある。そのため、第一ではなく、第二のエネルギー源として活用される。グルコースが存在せず、ラクトースが周囲にある場合、LacR は DNA から分離し、RNA ポリメラーゼが *lacZ*、*lacY*、*lacA* の転写を開始、ラクトース代謝に必要な酵素を発現させる。これにより、β-ガラクトシドパーミアーゼがラクトースを大腸菌細胞内に運搬し、β-ガラクトシダーゼがラクトースをグルコースとガラクトースに分解、その分解で生じたガラクトースをβ-ガラクトシドトランスアセチラーゼが分解、代謝し必要なエネルギーを得る(グルコースは既に合成済みのグルコース代謝酵素により、代謝される)。

1.3. ラクトースリプレッサーの構造と特性

ラクトースリプレッサー単量体は DNA-binding domain (1-58 アミノ酸残基)、Linker (59-61 アミノ酸残基)、Regulatory domain (62-340 アミノ酸残基)、Tetramerization region (341-360 アミノ酸残基)で構成される。DNA binding domain はさらに、Head-piece domain (1-49 アミノ酸残基)と Hinge-helix domain (50-58 アミノ酸残基)に分けられる(Fig. 2)。

Tetramerization region は四量体を形成するために必要な部位である。上記構造を単量体とし、これが二量体(Fig. 3)を形成することで DNA を挟み、さらに、その二量体が二対で四量体を形成することで、DNA の二箇所をループ状にして転写を抑制する(Fig. 4)。Regulatory domain はリガンドと結合し、その影響は Linker を通じて DNA-binding domain へと伝わる。Hinge-helix domain は、DNA の副溝と結合し、LacR を DNA 上にうまく固定させる。Head-piece domain は Operator を認識し、結合する。

また、ラクトースリプレッサーは、アロステリックタンパク質の一つである。アロステリックタンパク質とは、タンパク質の機能が他の化合物によって調整されるタンパク質である。LacR の場合、この他の化合物はラクトースなどのリガンドであり、周囲のラクトースの存在が LacR の DNA 親和性 (DNA と結合、分離)に影響を与える。

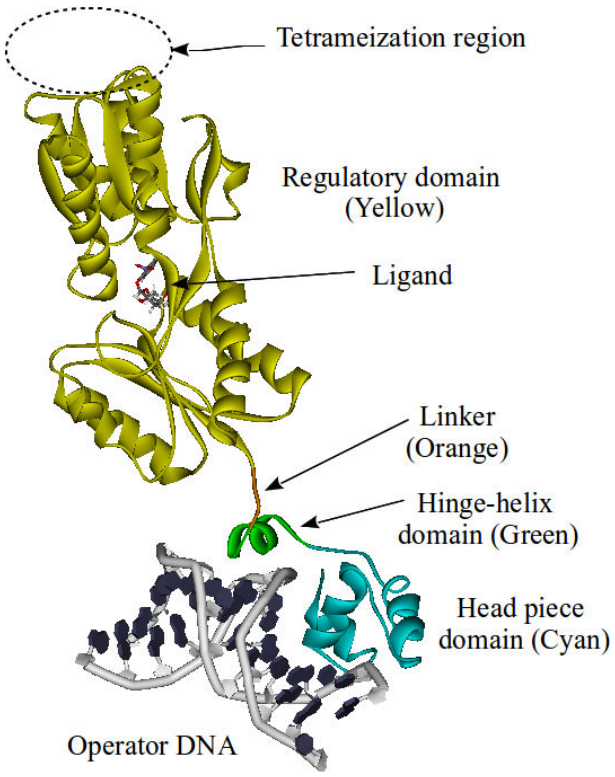


Fig. 2 The structure of LacR monomer + ligand bound to operator DNA [2].

Fig. 3 The structure of LacR dimer + ligands bound to operator DNA [2].

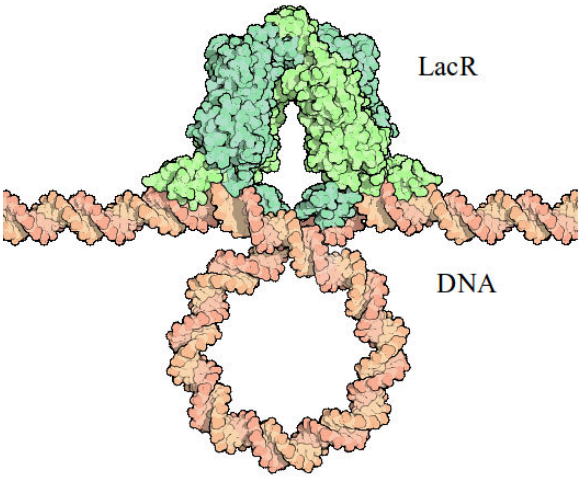


Fig. 4 The predicted structure of LacR tetramer bound to looped operator DNA.

1.4. LacR に対するリガンドの種類とそれらの影響

リガンドは、タンパク質に結合することにより、そのタンパク質の機能や構造を変化させる分子のことである。LacR はリガンド依存型のタンパク質であり、LacR のリガンドは、LacR に結合した際の LacR の DNA に対する親和性で分類される。インデューサは、ラクトースオペロンの転写抑制を解除するリガンドである。これには、ラクトース (allolactose; Fig. 5 (a))や IPTG (Isopropyl-β-D-thiogalactopyranose; Fig. 5 (b))[3]、T-ONPG (T-*o*-nitrophenyl-β-galactoside; Fig. 5 (c)) [4]が挙げられる。T-ONPG は実験[4]のため、ONPG (*o*-nitrophenyl-β-galactoside)のグルコシド結合の酸素原子を硫黄原子に置換したリガンドである。アンチインデューサはインデューサとは逆の機能を示すリガンドであり、これが LacR に結合することにより、LacR の転写抑制機能を促進させる。これには、ONPF (*o*-nitrophenylfucoside; Fig. 5 (d))[3]、T-ONPF (T-*o*-nitrophenylfucoside; Fig. 5 (e))[4]が挙げられる。T-ONPF も、T-ONPG 同様、ONPF のグルコシド結合の酸素原子を硫黄原子に置換したリガンドである。ノンインデューサは、LacR に結合するだけで転写抑制機能に関与しないリガンドである。実験[4]で、インデューサ IPTG とアンチインデューサ ONPF の両方の構造を持つということで導入されたリガンドであり、ONPG (*o*-nitrophenyl-β-galactoside; Fig. 5 (f))が挙げられる。

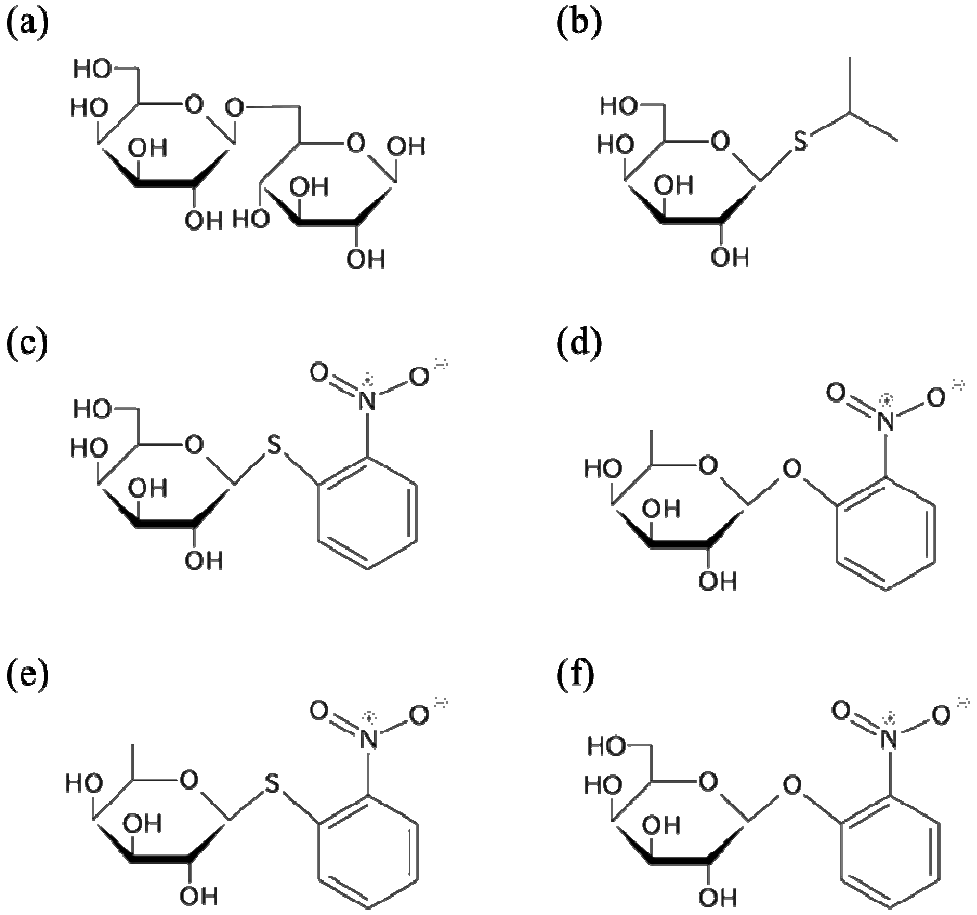


Fig. 5 The structures of ligands for LacR.

1.5. 当研究室でのこれまでの研究と研究課題

当研究室では、半経験的分子軌道法により、LacR の認識ヘリックスと DNA 間の電子状態を解析し、認識ヘリックスにおいて、DNA との結合に重要なアミノ酸を明らかにした [5]。また、LacR の四量体構造を作成し、LacR の四量体形成ドメインが二量体、及び四量体を形成する際に、四量体形成ドメインが各単量体間に及ぼす影響を分子動力学 (MD) 計算により明らかにした [6]。さらに、単量体の LacR とリガンドである IPTG や ONPF との結合に水分子が重要であること、LacR の Helix-turn-helix (HTH)の DNA 認識ドメイン、及びヒンジヘリックスが DNA と強く相互作用していることを明らかにした [7]。最近では、フラグメント分子軌道 (FMO) 法 [8]を用い、単量体 LacR とリガンド (IPTG や ONPF) 間、及び単量体 LacR と DNA 間の電子状態を解析し、リガンドとの結合に重要なアミノ酸、及び LacR と DNA 間の結合に重要なアミノ酸と DNA 塩基を解明した [9]。

しかし、これまでの研究は、tetramerization domain が LacR の構造変化に与える影響やリガンドが LacR のリガンド結合領域の周囲に与える影響について議論しているが、リ

ガンド結合が LacR と DNA 間の結合に与える影響については解析していない。

1.6. 本研究の目的と意義

本研究は、リガンド結合による LacR の構造変化や LacR とリガンド間、LacR と DNA 間、また LacR 単量体間の電子状態を解析し、DNA の転写制御機構を原子・電子レベルで解明することを目的としている。電子状態計算では、第一原理の手法を用い、LacR とリガンド間、及び LacR と DNA 間の相互作用を高精度に扱う。

この研究で、リガンド結合が LacR に与える影響を解明できれば、ラクトースオペロンの転写制御機構の全容の解明に繋がると考えられる。さらに、本研究の手法、着目点を基に、未解明の他の転写制御機構の解明の糸口に繋がる可能性がある。また、この研究は創薬の分野にも活かすことができると考えられる。例えば、アレルギーやガンは転写制御機構がうまく機能せず、過剰に悪性タンパク質が合成されることが原因である。これに対し、現在使用されているガンやアレルギーに対する薬は、転写、翻訳の過程で生成された悪性タンパク質を無効化するというコンセプトの下、開発されてきた。この場合、悪性タンパク質が生成されている間は、薬を飲み続けなければならない。しかし、悪性タンパク質の生成を転写の段階で阻害することができれば、飲む薬の量を減らすことができる。この問題に対し、本研究は最終的に転写制御機構をより巧妙に制御する新規リガンドの提案や転写を人為的に抑制する人工タンパク質の提案で貢献できると考えられる。

2. 計算対象

2.1. LacR-リガンド+DNA 複合体

これまでの実験により、LacR の各ドメインの立体構造は明らかになっている。しかし、LacR 四量体の構造を始め、LacR 単量体全体 (1-360 アミノ酸残基) の完全な状態の構造は明らかになっていない。それは、LacR が、結晶化が困難で X 線解析が容易でないタンパク質だからである。

現在、Protein Data Bank (PDB) に登録されている LacR の構造は、DNA (11 塩基対) 及び 1-51 番のアミノ酸残基を含む LacR 単量体の DNA-binding domain の X 線構造 (PDB code: 1LCC [10]; Fig. 6 (a))、61-356 アミノ酸残基を含む四量体形成ドメインを含む LacR 単量体の X 線構造 (PDB code: 1TLF [11]; Fig. 6 (b))、DNA (21 塩基対) 及び 1-357 番のアミノ酸残基を含む LacR 四量体の X 線構造 (ただし、Cαのみで構成) (PDB code: 1LBG [12]; Fig. 6 (c))、62-330 番のアミノ酸残基を含む LacR 二量体とリガンド (IPTG、ONPF) が結合した X 線構造 (PDB code: 2P9H [4]; Fig. 6 (d)、2PAF [4]; Fig. 6 (e))、DNA (14 塩基対) 及び 2-330 番のアミノ酸残基を含む LacR 三量体とリガンド (ONPF、ONPG) が結合した X 線構造 (PDB code: 1EFA [2]; Fig. 6 (f)、2PE5 [4]; Fig. 6 (g)) である。うち、結晶水が X 線構造内に含まれている構造は、1LCC、2P9H、1EFA である。

本研究では、LacR とリガンド間、LacR と DNA 間、LacR 単量体間の構造変化、及び特異的相互作用を解析することを目的としているため、Regulatory domain、DNA-binding domain、DNA を含んだ構造でなければならない。また、先行研究[7, 9]で結晶水が LacR とリガンド間、LacR と DNA 間に対し、ブリッジした水分子として相互作用に重要であると示唆しているため、構造内には結晶水が含まれていなければならない。これらのことから、LacR のベースとなる構造は、1EFA が妥当であると判断し、1EFA を採用した。

2.2. リガンドの種類

LacR のリガンドとして、Fig. 5 に示す化合物が一般的に知られている。この内、IPTG と ONPF は、LacR の実験でよく用いられている。ラクトースは、ラクトースオペロンの転写が活性化により生成されたβ-ガラクトシドトランスアセチラーゼにより、2 つの糖 (ガラクトースとグルコース) 間のグルコシド結合が切断され、分解される。そのため、実験で LacR の構造を入手するためのリガンドとしては不向きである。一方、IPTG はイソプロピル基とガラクトピラノース (β-D-galactopyranose) が硫黄原子により結合した構造であり、この硫黄はβ-ガラクトシドトランスアセチラーゼにより分解されないため、LacR の構造を入手するために有効なリガンドであり、実験でよく用いられる。従って、本研究ではインデューサとして、実験構造が存在する IPTG を採用した。また、Proten data bank 上には、アンチインデューサデある ONPF と結合した LacR が多数存在する。本研究で採用した LacR の構造である 1EFA も ONPF と結合していた。そのため、本研究で

は、アンチインデューサとして ONPF を採用した。

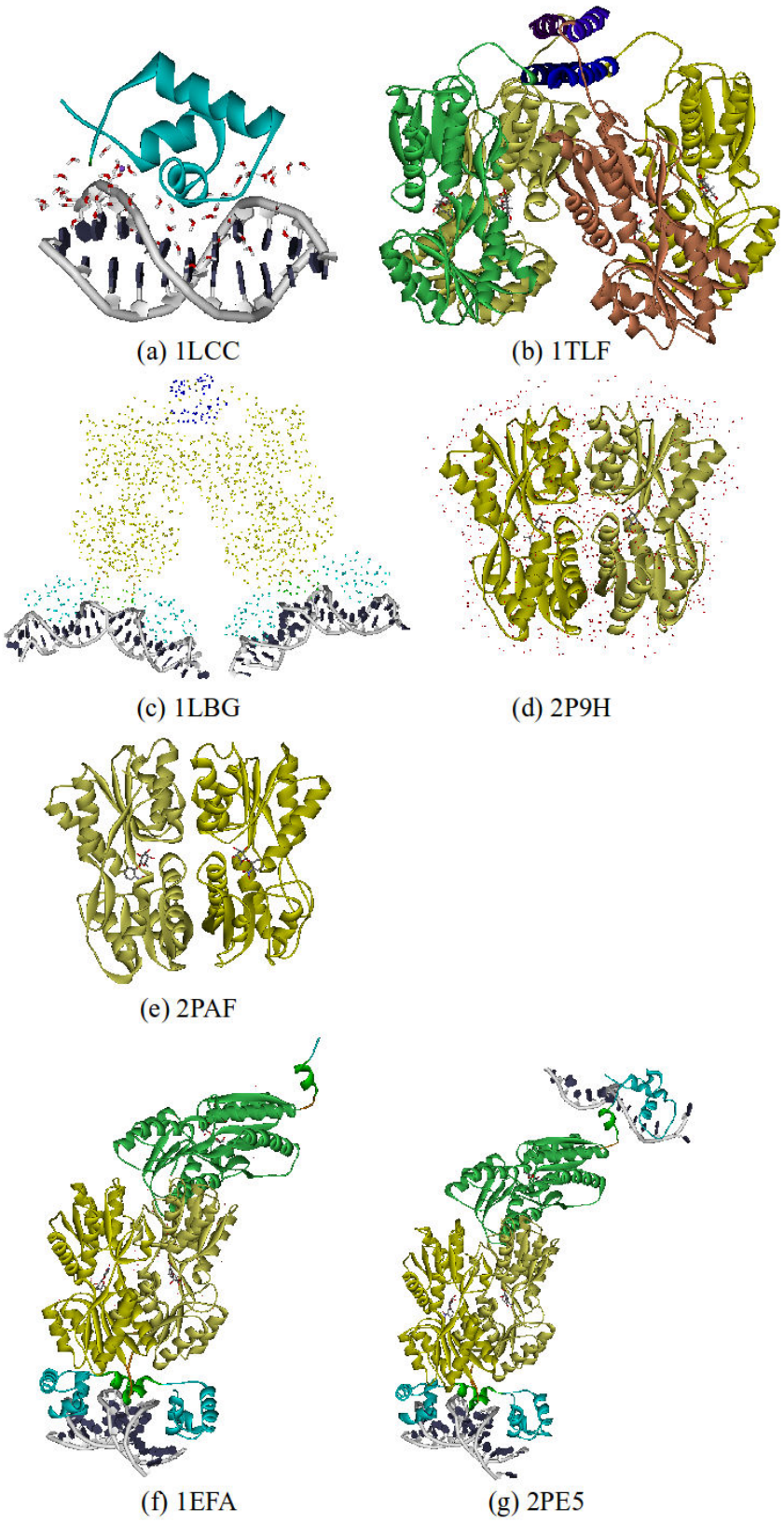


Fig. 6 The structures of the complexes involved LacR registered in PDB.

3. 計算手順

3.1. LacR 単量体-リガンド+DNA 複合体に対する解析

3.1.1. 複合体の初期構造の作成

LacR 単量体複合体の初期構造は PDB の 1EFA の X 線構造から、LacR 単量体、DNA、結晶水、アンチインデューサ ONPF を切り出した。この PDB 構造における DNA は、LacR 二量体それぞれの DNA-binding domain と結合する構造となっているため、LacR 単量体のみを切り出すと、LacR 単量体の DNA-binding domain の割に DNA が長くなってしまう。そのため、DNA は LacR の DNA-binding domain に合うように、10 塩基対の長さに調整した。さらに X 線構造内の DNA には、DNA のリン酸基の負電荷を中和させるカウンターイオンが存在しない。この負電荷を中和させるため、DNA の各リン酸基に対し、カウンターイオン Na^+を配置した。Na の位置は、リン酸基のリン酸基の酸素原子 O_{1P}、O_{2P} の中点(Midpoint)を求め、P 原子から中点を通って、P 原子から中点までの距離の 2 倍の位置にした(Fig. 7)。

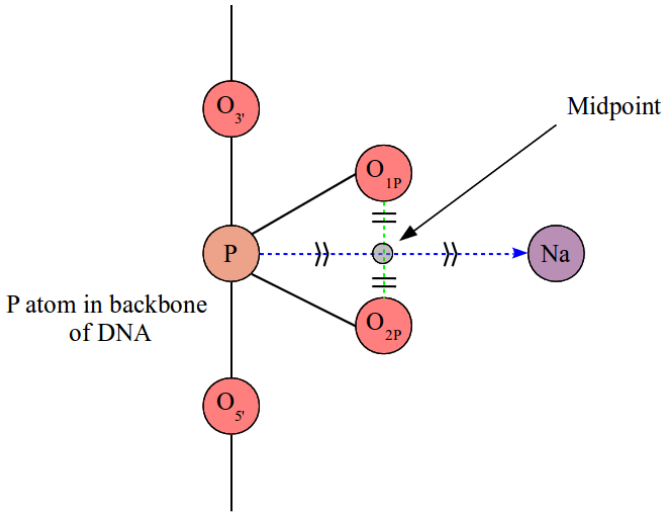


Fig. 7 The relative positioning between oxygen atoms of DNA backbone and Na^+ ion.

この構造は既にアンチインデューサ ONPF が結合しているため、LacR-ONPF+DNA 複合体とした。IPTG と結合した LacR 単量体複合体は、LacR-ONPF+DNA 複合体内に存在する ONPF のフコース (糖) 部位をベースに分子モデリングソフト HyperChem [20]でニトロフェニル基をイソプロピル基に置換することにより、IPTG を作成し、LacR-IPTG+DNA 複合体とした。さらに、LacR-ONPF+DNA 複合体の ONPF を削除することで、リガンドが結合していない状態の LacR+DNA 複合体を作成した。

3.1.2. 複合体の水和構造の作成と最適化

実際の生体内における LacR 単量体複合体の挙動を再現するため、溶媒水を露に考慮する必要がある。そこで、分子力学計算プログラム AMBER9 [21-23]パッケージに付属する tLEaP で、3.1.1 節で作成した複合体構造の周囲 9 Å に水分子を付加した。そして、AMBER9 の MM 法を用い、それぞれの複合体を最適化した。その際、アミノ酸残基、DNA、カウンターイオンには Amber99 力場、水分子には TIP3P [42]力場、リガンドは GAFF 力場を用いた。GAFF 力場作成に必要なリガンドの各原子の電荷は、リガンドを複合体から切り出し、高精度分子軌道計算プログラム Gaussian 03 [14]の MP2 法、基底関数 6-31G(d)で最適化した上で、同じ計算手法を用いた Restrained Electrostatic Potential (RESP) [16-19]計算で求めた。また、構造最適化の収束判定条件は、0.001 kcal mol^{-1} $Å^{-1}$ とした。

構造最適化に用いた Amber99 力場の計算制度を検証するため、実験構造(PDB code: 1EFA)と最適化した水和構造間の Cαの RMSD (Root Mean Square Distance)を求め、0.94 Å で、最適化された水和構造が実験と比較できることを確認した。

3.1.3. LacR、DNA、リガンド間の特異的相互作用の解析

LacR とリガンド間、LacR と DNA 間、単量体 LacR 間の特異的相互作用を解析するため、*ab initio* フラグメント分子軌道プログラム ABINIT-MP ver. 4.3　[24, 25]のフラグメント分子軌道(FMO) [26, 27]法を用い、マルチレイヤーFMO (MFMO)法[28]により、電子状態を計算した。MFMO 法は、系を複数のレイヤーに分け、レイヤー毎に計算手法を設定することができる。これにより、計算全体の計算コストを従来の MO 法に比べ低く抑えつつ、系の重要な部位のみを高精度に計算することができる。

しかし、最適化した水和構造には溶媒水を含め、総原子数が 15,000 原子以上であり、溶質との相互作用に関係のない溶質から離れている溶媒水まで第一原理で計算することは、計算の無駄となってしまう。そのため、特異的相互作用に重要である水分子のみを残し、他の水分子を削除して計算した。

LacR とリガンド間の特異的相互作用では、主にリガンドの周囲だけに着目し、計算する。従って、系の構造は、LacR、リガンド、DNA、Na^+、水分子で、計算に必要な水分子は、リガンドの周囲 5 Å の水分子のみにした。これは、リガンド結合ポケット内に入っている水分子がリガンドの周囲 5 Å 以内に存在し、それ以上の範囲を残すと溶媒水まで含まれ、リガンドとの相互作用に重要でない水分子まで計算し、計算効率が落ちる。それを防ぐため、5 Å に設定した。

以上の構造に対し、リガンド、リガンドから 5 Å のアミノ酸残基、及び水分子を MP2 法で、その他の部位を HF 法に設定した(Fig. 8)。いずれのレイヤーも基底関数は 6-31G を用いた。ここで、MP2[15]法の領域を 5 Å にしている理由は、リガンド結合ポケットを形成しているアミノ酸、またその中に入っている水分子が全て 5 Å の範囲内に存在しているためである。

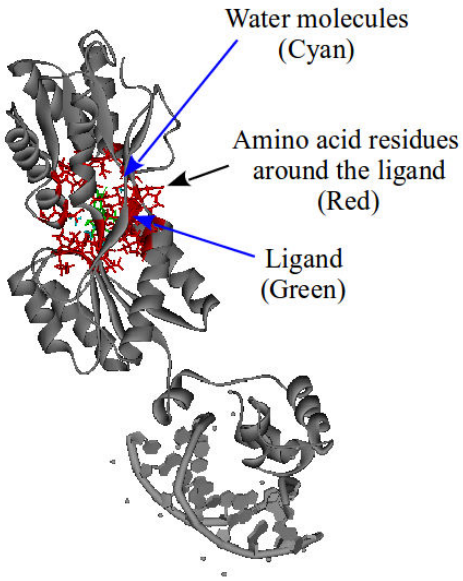


Fig. 8 The domain division in the multi-layer FMO calculations for investigating the specific interactions between LacR monomer and ligand.

一方、LacR と DNA 間の特異的相互作用の解析では、主に LacR の DNA-binding domain と DNA に着目するため、対象とする系は LacR、リガンド、DNA、Na^+、水分子で、LacR、DNA からそれぞれ 5 Å 以内に存在する共通の水分子と Na^+の周囲 3 Å にある水分子を残した。先行研究[7, 9]では、LacR のアミノ酸残基と DNA の塩基が水分子を介し、水素結合を形成して特異的相互作用しており、LacR と DNA 間の水分子が重要であることが示唆された。そこで LacR のアミノ酸残基と DNA 塩基がブリッジすると考えられる水分子を、計算コストも考慮し、最小に残す範囲を試行錯誤した所、LacR、DNA からそれぞれ 5 Å の位置にある水分子が妥当であるという結論に至った。Na^+の周囲の水分子は、Na^+はイオンで存在し、周囲の分子と静電相互作用をすることを考慮し、その静電相互作用が他の相互作用に影響を与えない最小限の領域である第一水和圏の 3 Å とした。

この構造に対し、LacR の DNA-binding domain (2-61 アミノ酸残基)、DNA (17 塩基対)、系内の水分子全てを MP2 法で計算し、その他の領域を HF 法で計算した。なお、基底関数は 6-31G を用いた(Fig. 9)。

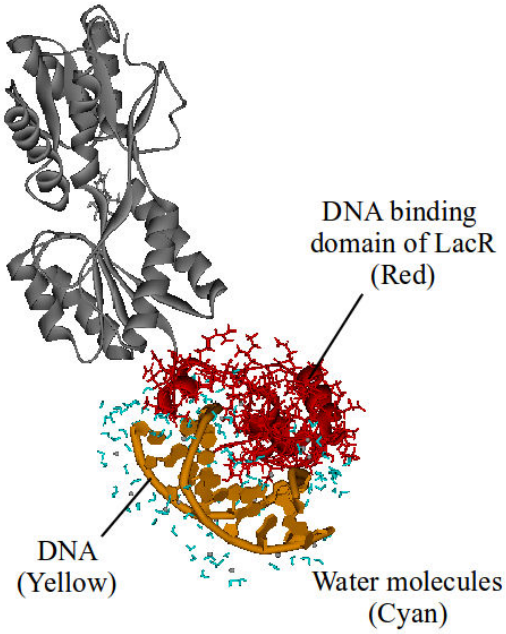


Fig. 9 The domain division in the multi-layer FMO calculations for investigating the specific interactions between LacR and DNA.

3.1.4. 複合体に対する水中での MD 計算

LacR は約 300 残基で構成されるアミノ酸で、これは比較的大きなタンパク質であると言える。このような LacR が DNA から分離する構造変化はドメイン単位で起こると考えられ、その構造変化をシミュレーションで再現するためには、周囲の環境に何らかの影響を与えるか、長時間のシミュレーションをする必要がある。従って、上記の MM 法により得た構造だけでは、構造変化した LacR を得ることは難しいと考えられる。

そこで、古典分子動力学 (MD) 計算を実行した。MM 法により得た複合体の周囲 9 Å に水分子が付加された水和構造を初期構造として採用した。MD の計算条件は、アンサンブルは NPT、全計算時間は 10 ns、時間ステップは 1 fs、温度は初期温度を 0 K とし、その後 25 ps かけて、生体内とほぼ同じ温度である 300 K まで上昇させた。この計算では、周期境界条件を用いなかった。力場は MM 法と同様に、アミノ酸残基、DNA、カウンターイオンには Amber99 力場、水分子には TIP3P 力場、リガンドには GAFF 力場を用いた。

さらに、構造変化が十分に起こったと思われる 10 ns の最後の構造を取得、最適化し、3.1.3 節と同様の条件下で FMO 計算を実行した。

3.2. LacR 二量体-リガンド+DNA 複合体に対する解析

3.2.1. 複合体の初期構造の作成

LacR 二量体も単量体同様、1EFA の PDB から対を形成している LacR 二量体と DNA、

結晶水、アンチインデューサ ONPF を切り出した。LacR 単量体は、それぞれ 2-329 残基、2-331 残基で構成されていたため、単量体の大きさを合わせるため、2-331 残基の LacR 単量体を 2-329 残基に調整した。

次に、LacR 内のヒスチジンのプロトネーションを決定した。LacR 単量体内にはヒスチジンが 7 個存在する。ヒスチジンは側鎖に解離性の原子集団であるイミダゾール基を持っている。このイミダゾール基は、窒素原子に結合したプロトン(H^+)が環境内の水素イオン濃度により、脱着を起こす。これにより、ヒスチジンは His(δ)、His(ε)、His(p; protonated)(Fig. 10)の 3 つの状態に変化する。His(δ)と His(ε)はイミダゾール基に結合しているプロトンの位置の付き方が変化し、立体障害や水素結合に影響を与える。水素イオン濃度が中性に近い、pK_a 6 以上では、ヒスチジンは His(p)に変化する。His(p)は、イミダゾール基に 2 つのプロトンが付くため、構造だけでなく、電荷状態も変化する。電荷が影響を与える静電相互作用は、生体内で重要な役割を担うことが多いため、このヒスチジンのプロトネーションを無視することはできない。そこで、タンパク質内の水素イオン濃度を推定するプログラム PROPKA[32-35]を用い、LacR 内に存在するヒスチジン周囲の水素イオン濃度を推定した。その結果、LacR 単量体内のヒスチジン 7 個のうち、His(p)になる可能性のあるヒスチジンは His29、His202 であった。その他の 5 個のヒスチジンは、周囲の構造を見て、付加されるプロトンが立体障害を起こさないような構造を選択した。最終的に、His(ε)が 5 個、His(p)が 2 個となった。(Fig. 7)

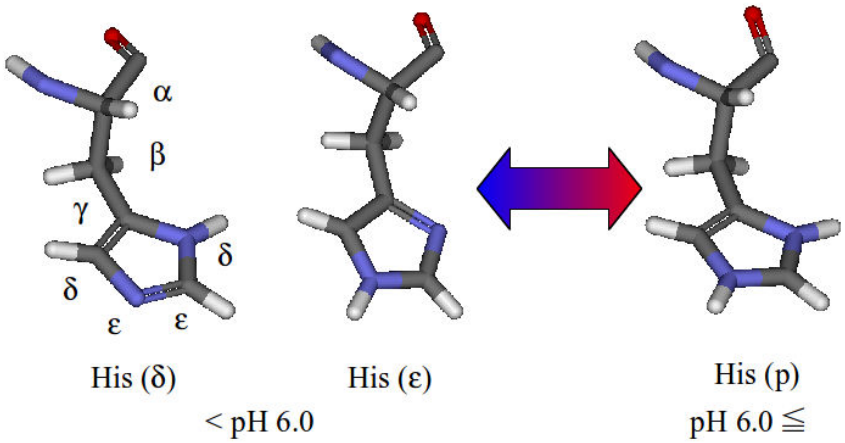


Fig. 10 Three types of protonated structures of histidine residue depending on pH value.

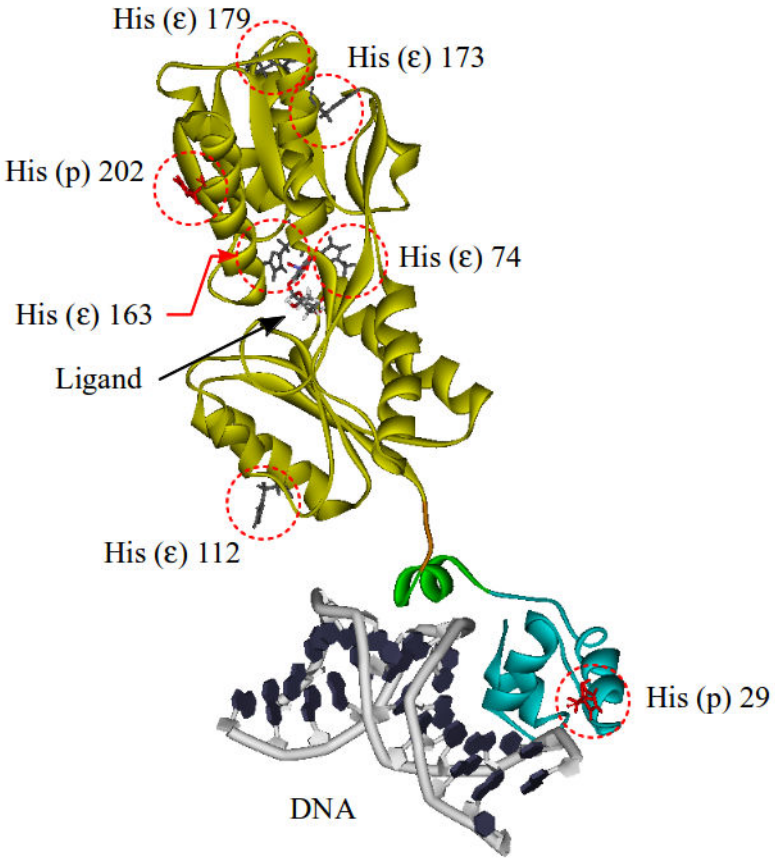
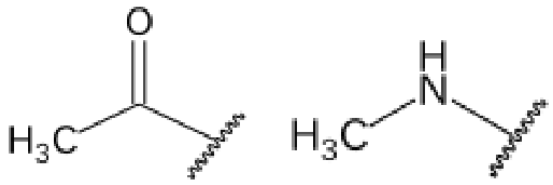


Fig. 11 The positions of histidine residues in LacR.

そして、LacR の電荷(LacR 単量体: -1)を中和するため、LacR の周囲にカウンターイオン Na^+を AMBER12 [36]に付属する Ambertools パッケージ内の tLEaP で付加した。また、LacR のアミノ酸残基の末端は何も加工しない場合、N 末端は NH_3^+、C 末端は COO^-として扱われる。この場合、LacR 全体の電荷の収支は 0 となるが、アミノ酸毎の相互作用エネルギーを解析する際、この末端残基の電荷が他のアミノ酸残基や DNA 塩基に強く影響し、他の相互作用に影響を与えてしまう。そのため、N 末端にアセチル基(Fig. 12 (a))、C 末端にメチルアミン基(Fig. 12 (b))を付加して末端の電荷を 0 にした。



(a) acetyl group (b) methylamine group

Fig. 12 Structures of (a) acetyl group and (b) methylamine group for terminating N- and C-terminals of LacR, respectively.

DNA に関しては、3.1.1 節の LacR 単量体複合体の時と同様の手法を用いて、DNA の各リン酸基に対し、カウンターイオン Na^+を配置した。

結晶水に関しては、PDB に三量体として登録されていたものから、二量体と DNA、結晶水をそのまま取り出したため、三量体目のあった位置に結晶水がいくつか残った状態になっている。この後、溶媒水を溶質の周囲に付加するのであるが、溶媒水を付加するプログラムは与えた構造の周囲に水分子を付加する。そのため、三量体があった位置に存在する結晶水に溶媒水を付加してしまい、不用意に系のサイズが大きくなってしまい、計算効率が落ちてしまう。そこで、LacR 二量体、DNA、Na^+複合体から 5 Å の水分子だけ残して、その他の結晶水は削除した。

この構造は既にリガンドとして、アンチインデューサ ONPF が結合しているため、この構造を LacR-ONPF+DNA 複合体構造とした。

LacR-DNA-ONPF 複合体構造から ONPF を削除することにより、LacR+DNA (リガンドなし)複合体構造を作成した。

インデューサ IPTG と結合した LacR-IPTG+DNA 複合体構造に関しては、二量体 LacR とそれぞれの単量体 LacR に IPTG が結合した構造である 2P9H の X 線構造と LacR+DNA 複合体構造のアミノ酸残基の Cαをフィッティングさせ、2P9H と LacR+DNA 複合体のアミノ酸残基の座標が一致するようにした。そして、そのフィッティングさせた 2P9H から IPTG の構造だけを抽出し、LacR+DNA 複合体に挿入した。その後、IPTG の座標が LacR+DNA 複合体のアミノ酸残基や結晶水と重なっていないことを確認し、この構造を LacR-IPTG+DNA 複合体とした。

3.2.2. 複合体の水和構造の作成と最適化

アロステリックタンパク質の構造変化はドメイン単位で起こるとされ、ドメイン単位の構造変化には、μs、ms オーダーのシミュレーション時間が必要とされている。従って、アロステリックタンパク質である LacR の構造変化を解析するためには、μs、ms オーダーの計算、あるいはそれが困難だとしても、最低数十 ns の計算をしなければ、構造変化の予兆すら観測することができない。LacR 二量体はその周囲に溶媒水を付加すると、系のサイズが LacR 単量体の 2 倍以上になる。これを AMBER で MD 計算しようと試みた所、Intel Core2 Quad Q9550 (2.83 GHz) を 4 CPU 使って、1 ns のトラジェクトリの出力に 3 日かかった。一方、Gromacs [37-40] は、同条件にて 1 ns のトラジェクトリの出力に 1 日で済んだ。そのため、LacR 二量体の計算は、並列化効率が Amber より高い Gromacs を用いた。

溶媒水は、Gromacs Ver. 4.5.3 パッケージ内の editconf でセルサイズを決定し、genbox でセルを水分子で満たした。セルサイズは、対象を座標の中心(0, 0, 0)に移動させつつ、X 軸方向に平行になるように回転、平行移動させた上で、対象の X、Y、Z 軸方向のサイズの 2 倍になるように設定した。これは、古典分子力学の周期境界条件ではセルサイズを十分に確保しないと、隣接するセル内の対象の相互作用が入り、正確な解析ができない。この条件で、セルサイズを設定した所、LacR+DNA、LacR-IPTG+DNA、

LacR-ONPF+DNA、いずれの構造も 188×140×127 $Å^3$ となった。

これらの水和構造に対し、Gromacs の MM 法で、構造最適化をした。力場パラメータとして、アミノ酸残基、DNA、Na^+は Amber99SB-ILDN 力場[41]、水分子に TIP3P 力場を用いた。また、リガンドの力場は GAFF [43]力場を用いた。GAFF 力場作成に必要なリガンドの各原子の電荷は、リガンドを複合体から切り出し、非経験的分子軌道プログラム GAMESS [44]の HF 法と基底関数 6-31G(d)で最適化した上で、同じ手法を用いた RESP 計算で求めた。このリガンドの電荷と構造を ACPYPE (AnteChamber PYthon Parser interfacE) [45]を用い、Gromacs で扱える力場に変換した。

また、カウンターイオンは最適化の過程で、溶質から離れ、カウンターイオンの意味をなくしてしまうことがある。そのため、水中の DNA に対し、カウンターイオンを考慮した MD 計算を行った計算 [46]を参考に、DNA 側の Na^+は DNA 骨格のリン酸から 6 Å 以内に距離拘束し、LacR 側の Na^+はカウンターイオン付近の負に荷電しているアミノ酸残基 Asp129 のカルボキシル基の C 原子から 6 Å 以内に距離拘束した (Fig. 13)。

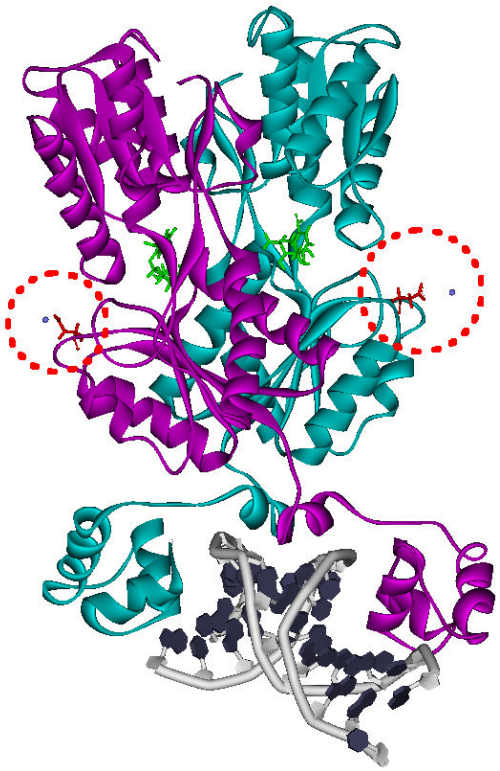


Fig. 13 The positions of counter ions Na^+ for the negatively charged Asp129 residue of LacR dimer; Na^+ ions and the Asp129 side chains are indicated in the dotted circles.

距離拘束に関しては、Na^+を拘束している原子から 0～6 Å にはポテンシャルエネルギーにペナルティを与えず、6 Å 以降に下記の式で表されるペナルティを与えるように設定した(r0、r1、r2 をそれぞれ 0.0、0.6、0.8 に設定した)。

$$V_{dr}(r_{ij}) = \begin{cases} \frac{1}{2}k_{dr}(r_{ij} - r_0)^2 & for\ r_{ij} < r_0 \\ 0 & for\ r_0 \le r_{ij} < r_1 \\ \frac{1}{2}k_{dr}(r_{ij} - r_1)^2 & for\ r_1 \le r_{ij} < r_2 \\ \frac{1}{2}k_{dr}(r_2 - r_1)(2r_{ij} - r_2 - r_1) & for\ r_2 \le r_{ij} \end{cases}$$

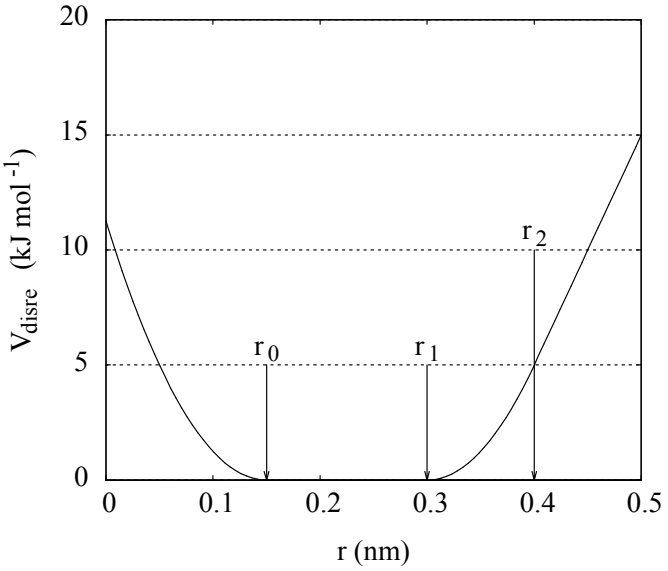


Fig. 14 The potential penalty for distance restraint.

また、MD 計算をする過程で、溶質が平行移動によりセルから外に飛び出してしまう可能性があるため、最低でも溶質の一点を座標拘束する必要がある。LacR は、インデューサと結合すると DNA から分離するため、LacR 側の座標を固定せず、DNA 二重鎖の中心で LacR から離れている方の DNA 鎖の塩基のリン酸基のリン酸を X、Y、Z 座標に対し、下記式の定数 k_x^{pr} を 1000 kJ mol^{-1} にして固定した。

$$V_{pr}(r_i) = \frac{1}{2}\left[k_{pr}^{x}(x_i - X_i)^2\hat{x} + k_{pr}^{y}(y_i - Y_i)^2\hat{y} + k_{pr}^{z}(z_i - Z_i)^2\hat{z}\right]$$

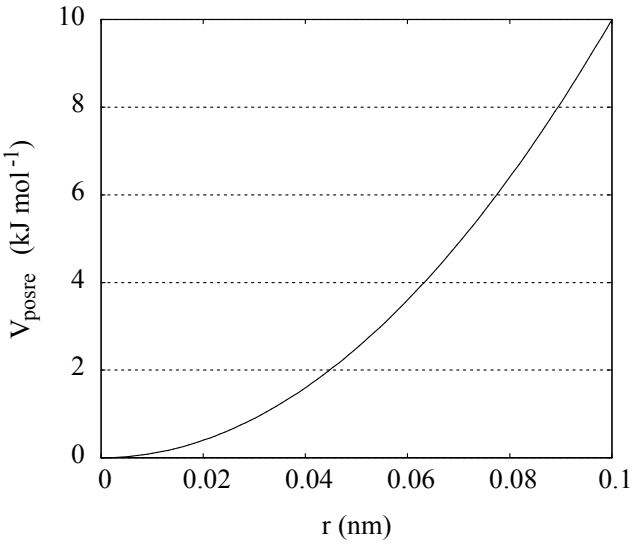


Fig. 15 The potential penalty for position restraint.

以上の条件下、構造最適化は Gromacs Ver. 4.5.3 の最急降下法を用い、収束判定条件を 1.0×10^{-4} kJ mol^{-1} nm^{-1}、周期境界条件下で計算した。

3.2.3. 複合体に対する水中での MD 計算

リガンドが LacR に結合した時、あるいはリガンドがない時の LacR の構造変化を解明するため、生体内の環境で構造を動的に追跡する必要がある。そこで、Gromacs Ver. 4.5.3 の MD 計算を用いて、温度上昇過程、緩和過程、アンサンブル過程の順で、LacR の生体内の構造変化を追跡した。

初期の最適化した構造は、結晶構造をより安定化させた構造であるため、構造に温度が考慮されていない 0 K の構造に近い構造である。したがって、まずは構造に温度を与える必要がある。しかし、急激に温度を上昇させると、構造が破壊され、生体内の構造とは別の構造に変化してしまう可能性がある。そのため、温度上昇過程は焼きなまし法により徐々に温度を上げることにした。計算条件は、leap-frog 積分[47]、周期境界条件、静電相互作用は PME 法[48, 49]を用いて NVT アンサンブルを実行した。時間刻み幅は、水分子の水素原子を SETTLE [50]、その他の水素原子を LINCS [51]で拘束しているため、2 fs とした。温度は Velocity rescale 法[52]で制御し、焼きなまし法により、100 ps までは 0 K を維持させ、100 ps から 400 ps まで 1 K/ps の割合で温度を上昇させ、400 ps から 500 ps まで 300 K を維持させた。

緩和過程では、温度上昇過程と同条件で、温度制御、圧力制御を変更し、NPT アンサンブルを実行した。温度制御は Nose-Hoover 法[53, 54]で 300 K を維持、圧力制御は Parrinello-Rahman 法[55, 56]で 1 bar とし、1 ns 実行した。

アンサンブル過程は、緩和過程と同じ条件で 30 ns 実行した。

4. 計算結果と考察

4.1. LacR 単量体-リガンド+DNA 複合体に対する結果

4.1.1. MM 法により得た複合体の水和構造

リガンド結合による LacR 単量体複合体の構造変化の違いを調べるため、リガンドなしの LacR 単量体複合体とそれぞれのリガンドが結合した複合体構造間の Cαの座標のずれである displacement を比較した。Fig. 16 に示すように、LacR にリガンドが結合すると Gln60 が大きく変化することが明らかになった。特に LacR-ONPF の Gln60 は約 3.5 Å と顕著に変化している。この原因を明らかにするため、Gln60 と水素結合しているアミノ酸残基を解析した。LacR に ONPF が結合することにより、Arg118 の骨格の酸素原子と Gln60 の側鎖の水素原子は水素結合が形成する(Fig. 17 (b))。また Arg118 の displacement がほとんど変化していないことから、Gln60 は水素結合により Arg118 側に引き寄せられたと考えられる。Fig. 18 に示すように、Gln60 の近くには operator が存在し、Arg118 はリガンド結合ポケット付近に存在する。従って、ONPF が LacR+DNA 複合体に結合すると、Gln60 と Arg118 間の水素結合により、これらアミノ酸残基の距離は縮まり、それに合わせて LacR と DNA 間の距離も縮まり、他のアミノ酸残基も DNA と相互作用しやすくなると考えられる。一方、LacR-IPTG の Gln60 の Cαの displacement も大きいが、LacR-ONPF の Gln60 ほどではなく、水素結合形成までには至っていない(Fig. 17 (a))。これらにより、IPTG、ONPF が LacR に与える大きな影響を明らかにすることができた。また、Fig. 16 より、LacR の 70～80 アミノ酸残基の displacement の変化は LacR-IPTG の方が、LacR-ONPF より大きいことも明らかになった。この 70～80 アミノ酸残基は Fig. 18 に示すように、リガンド結合ポケットを形成するα-ヘリックスである。従って、IPTG は LacR に結合することにより、ONPF より LacR 結合ポケットの構造を変化させる効果があると考えられる。

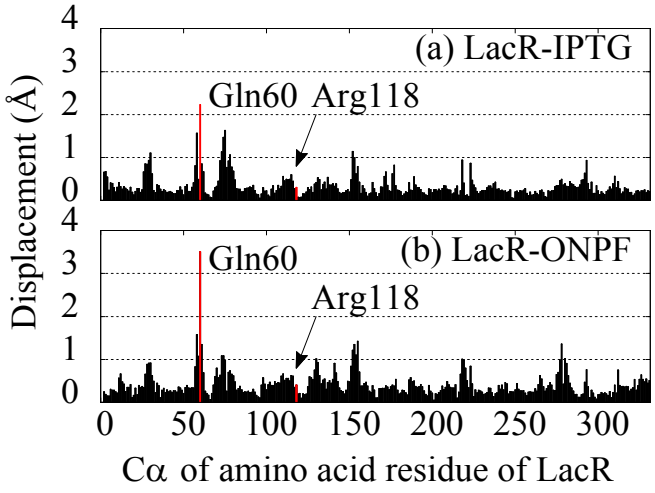


Fig. 16 Displacement of Cα atom of each amino acid residue in LacR induced by the binding of (a) IPTG and (b) ONPF to LacR.

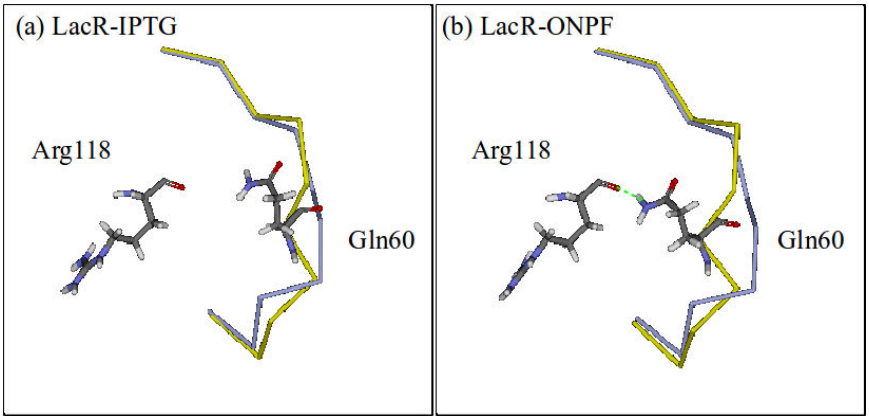


Fig. 17 Interacting structures between Arg118 and Gln60 of LacR in (a) LacR-IPTG and (b) LacR-ONPF complexes. A green dot line indicates a hydrogen bond; blue and yellow lines indicate the backbones of LacR and LacR-ligand, respectively.

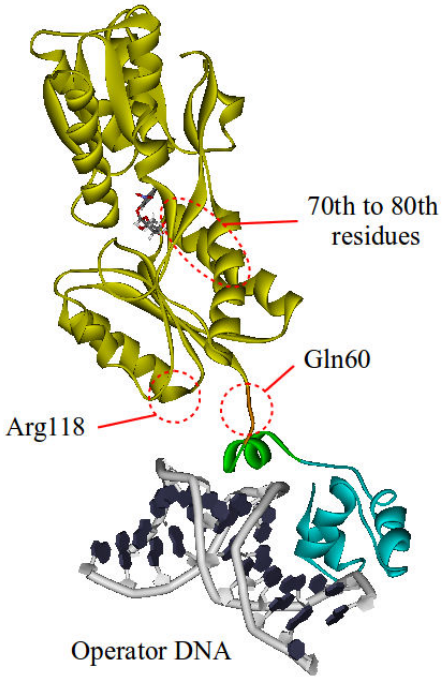


Fig. 18 Positions of Gln60, Arg118 and 70-80th residues of LacR in the LacR-ligand + operator DNA complex.

4.1.2. LacR 単量体とリガンド間の特異的相互作用

LacR+DNA 複合体への IPTG、及び ONPF の結合親和性を明らかにするため、MFMO 法により、水分子を含む全体構造 (Complex)、水分子を含むリガンドを除いた構造 (LacR+DNA+Water)、水和したリガンド単体の構造 (Ligand+Water)、水分子の構造 (Water) の全エネルギーを求め、それら全エネルギーの差から、LacR+DNA とリガンド間の結合エネルギーを求めた。

Table 1 Total eneriges (T.E.) (kcal/mol) of LacR-Ligand + DNA complex and its component parts, and estimated binding energies (B.E.) (kcal/mol) between LacR+DNA and ligand.

B.E. is estimated as

B.E.= –T.E.(Complex) + T.E.(LacR+DNA+Water) + T.E.(ligand+Water) – T.E.(Water).

	IPTG	ONPF
Complex	-97475552.9	-97471651.0
LacR + DNA + Water	-96817077.4	-96817060.2
Ligand + Water	-992735.7	-988839.2
Water	-334325.8	-334323.7
Binding energy	65.6	75.4

Table 1 は、MFMO 法で求めた水和した LacR-Ligand+DNA 複合体 (Complex)、水和した LacR+DNA 複合体 (LacR+DNA+Water)、水和したリガンド (Ligand+Water)、水分子(Water) の全エネルギーとそれらのエネルギーによって求められた結合エネルギーを示している。LacR+DNA 複合体と IPTG 間の結合エネルギー (65.6 kcal/mol) は、LacR+DNA 複合体と ONPF 間の結合エネルギー (75.4 kcal/mol) に比べ、約 10 kcal/mol 小さい。この結合エネルギーの違いの原因を明らかにするため、それぞれの複合体のリガンド周囲の構造を比較した。

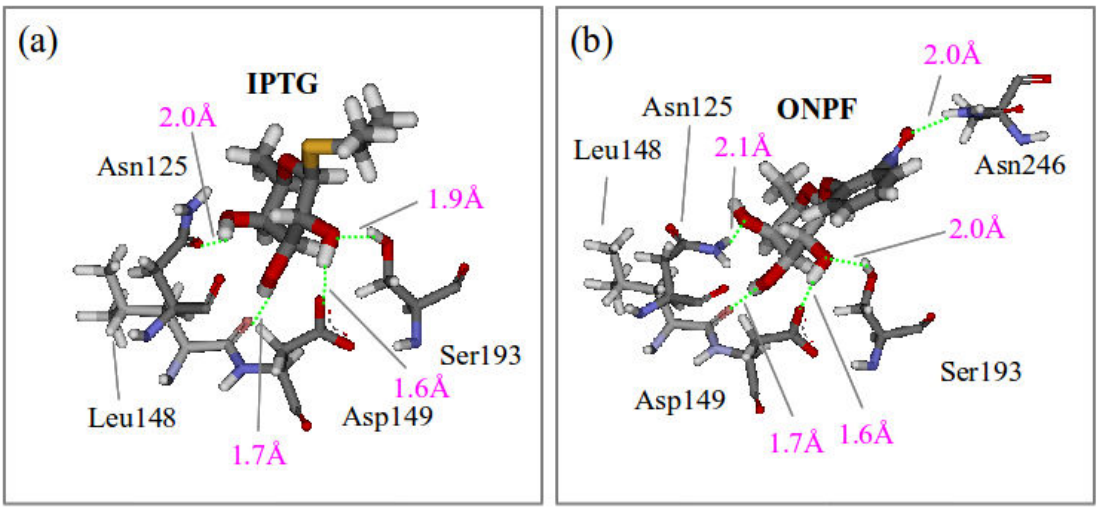


Fig. 19 Structures of hydrogen bondings (green dotted lines) between amino acid residues of LacR and (a) IPTG and (b) ONPF. The distance bnetween the oxygen and the hydrogen atoms contributing to the hydrogen bonding is shown in pink.

Fig. 19 (a) に示すように、IPTG のグルコピラノース (糖) 部位の水酸基は、周囲の LacR のアミノ酸残基と 4 つの水素結合を形成している。Fig. 19 (b) に示すように、ONPF も IPTG と同様にガラクトース部位の水酸基が周囲の LacR のアミノ酸残基が 4 つの水素結合を形成し、さらに ONPF のニトロフェニル基と LacR の Asn246 は水素結合を形成しており、この水素結合が、IPTG と ONPF の結合エネルギーの差の主な原因だと考えられる。実験の先行研究[2, 57]は、IPTG、ONPF が周囲のアミノ酸である Asn246、Arg197、Asp149 と水素結合を形成することを示唆した。今回の計算の結果でも、Fig. 19 に示すように、いずれのリガンドも Asp149 と強く水素結合を形成している。しかし、実験で重要とされていた Arg197 は、今回の計算で IPTG、ONPF からそれぞれ 2.45 Å、2.25 Å 離れており、水素結合を形成していなかった。

さらに、どの LacR のアミノ酸残基、及び水分子が、リガンドと LacR 間の結合に寄与しているかを解明するため、リガンドと LacR の各アミノ酸残基、及び水分子間の相互作用エネルギーを解析した。Fig. 20 に示すように、LacR の Asp149 はいずれのリガンドにおいても、強く引力相互作用をしている。Asp149 は負電荷のアミノ酸残基であるため、Fig. 19 に示すこのアミノ酸残基の側鎖とリガンドの糖部位は強い水素結合を形成してい

る。

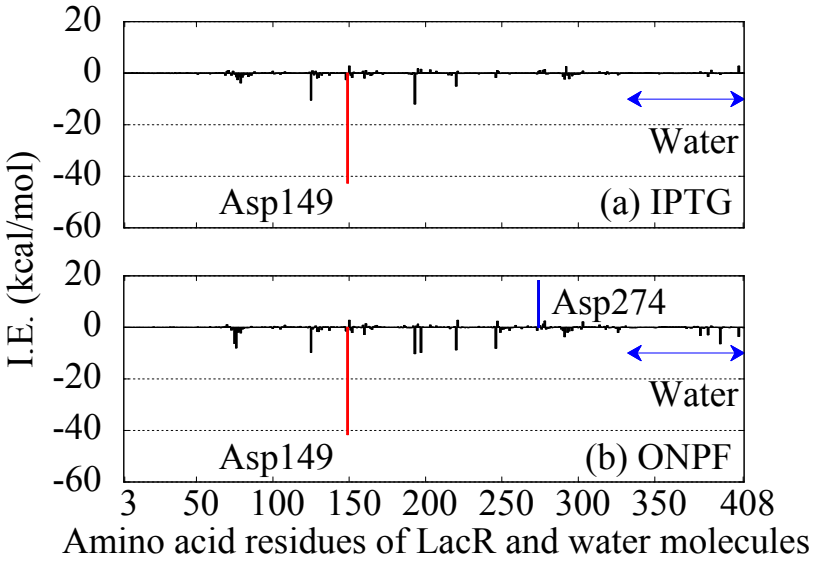


Fig. 20 Interaction energies (I.E.) between each amino acid residue of LacR or water molecules and (a) IPTG and (b) ONPF.

さらに、LacR とリガンドの結合に寄与している荷電アミノ酸について解析するため、リガンドとリガンド周辺の荷電アミノ酸間の距離を測定した。Table 2 に示すように、Asp149 はいずれのリガンドとも約 6〜7 Å と最も近い位置に存在する。そのため、Asp149 は強く引力相互作用していると考えられる。さらに、Asp274 と Arg197 も約 8 Å とリガンドに近い位置に存在している。そのため、IPTG と Asp274 間で-42.6 kcal/mol、ONPF と Asp274 間で 18.1 kcal/mol、ONPF と Arg197 間で-9.7 kcal/mol とリガンドと強く相互作用している。また、荷電アミノ酸だけでなく、水分子もリガンドと LacR のアミノ酸残基間の結合に寄与している。Fig. 21 に示すように、Asp149 や Asp274 の間には架橋水が存在している。Asp149 は ONPF と直接水素結合を形成している上、架橋水を介して、Ser191 への水素結合や ONPF へのもう一つの水素結合を形成している(Fig. 21 (a))。従って、Fig. 20 (b)や Fig. 21 (a)より、この架橋水と Asp149 は LacR と ONPF の結合エネルギーに大きく寄与していることが明らかになった。

Table 2 Distance (Å) between the gravity center of the ligand and the charged amino acid residue existing around the ligand in the LacR-Ligand + DNA complexes.

	IPTG	ONPF
Asp149	6.3	6.7
Arg197	8.4	7.9
Asp219	12.6	11.8
Asp274	7.5	7.3

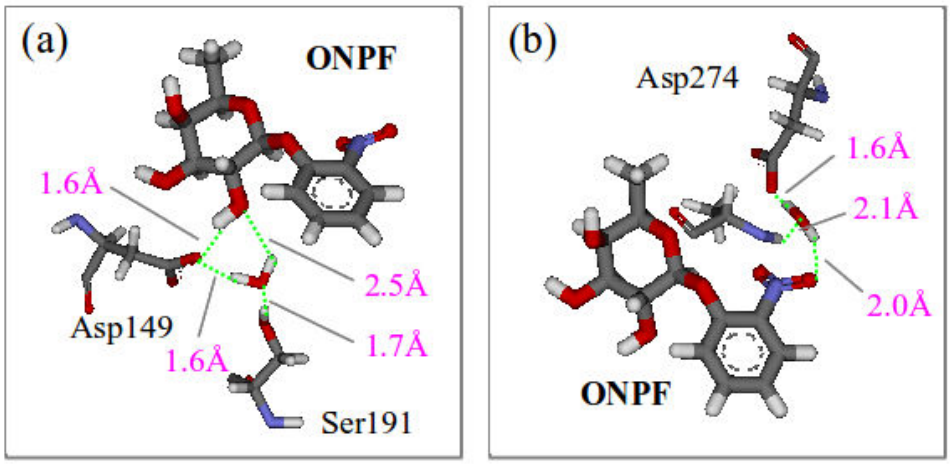


Fig. 21 Structures of hydrogen bondings (green dotted lines) between ONPF and (a) Asp149 or (b) Asp274 of LacR through a bridging water molecule. The distance between the oxygen and the hydrogen atoms contributing to the hydrogen bonding is shown in pink.

さらに、Fig. 20 (b)の結果で特徴的なことは、ONPF と LacR の Asp274 が反発相互作用を示していることである。これは、負電荷を持つ ONPF のニトロフェニル基と、同じく負電荷の側鎖を持つ LacR の Asp274 が静電反発相互作用をしているものだと考えられる。実際、Asp274 と ONPF の構造を示した Fig. 21 (b)では、負電荷を持つ Asp274 の側鎖と ONPF のニトロフェニル基は互いに向い合って、近くに存在する。しかし、この水分子は水分子を介して、Asp274 と ONPF は水素結合により結合している。この水分子の寄与を示した表が Table 3 である。これは、Asp274 とリガンド間の直接の相互作用エネルギー、Asp274 と Asp274 の近くに存在する架橋水間の相互作用エネルギー、リガンドと架橋水間の相互作用エネルギーを示している。いずれのリガンドも、Asp274 とは反発しており、特に ONPF は 18.1 kcal/mol で強い反発相互作用を示している。しかし、Asp274 と架橋水間、ONPF と架橋水間の相互作用エネルギーはそれぞれ、-27.4 kcal/mol、-6.4 kcal/mol で引力相互作用をしている。その結果、ONPF と Asp274 は、Fig. 21 (b)のように架橋水を介して、安定に結合している。逆に IPTG は、イソプロピル基が疎水性を持つため、このような結合を安定化させる架橋水は存在しない。

Table 3 Interaction energies (kcal/mol) between Asp274 and ligand, between Asp274 and the water molecule, and between ligand and the water molecule for the LacR-IPTG+DNA and LacR-ONPF+DNA complexes.

	IPTG	ONPF
Asp274-Ligand	0.7	18.1
Asp274-Water	-27.6	-27.4
Ligand-Water	0	-6.4

以上のことから、今回の水分子を露に考慮した MFMO 法により、LacR のアミノ酸

残基と親水性の ONPF 間に存在する水分子の重要性を明らかにすることができた。

4.1.3. LacR 単量体と DNA 間の特異的相互作用

リガンド結合が LacR と DNA 間の特異的相互作用に及ぼす影響を解明するため、MFMO 法により、LacR+DNA 複合体、及び LacR-Ligand+DNA 複合体の結合エネルギーを求めた。Table 4 は、水和した全体構造(Complex)、水和した LacR-Ligand 構造(LacR-Ligand)、水和した DNA 構造(DNA + Water)、水分子の構造(Water)の全エネルギーを求め、それらの差から LacR と DNA 間の結合エネルギーを求めたものである。実験の結果から、リガンドと結合していない LacR の LacR と DNA 間の結合エネルギーに比べ、インデューサ IPTG が結合した LacR は、LacR と DNA 間の結合が弱く、逆にアンチインデューサと結合した LacR は LacR と DNA 間の結合が強いと考えられる。Table 4 の LacR+DNA 複合体の結合エネルギー(B.E.: 771.0 kcal/mol)と LacR-ONPF+DNA 複合体の結合エネルギー(B.E.: 830.8 kcal/mol)から、アンチインデューサ ONPF が LacR に結合したことを説明することができる。しかし、LacR-IPTG+DNA 複合体の結合エネルギー(B.E.: 789.2 kcal/mol)は、LacR+DNA 複合体より 18 kcal/mol 大きく、インデューサ IPTG が結合したことをうまく説明できない。今回、FMO 計算に用いた LacR-IPTG+DNA 複合体構造は、LacR-ONPF+DNA 複合体構造の X 線構造を基に、水中で構造最適化することにより作成した構造である。従って、今回の LacR-IPTG+DNA 複合体構造は局所的安定構造である可能性がある。水和した LacR+DNA 複合体、及び LacR-IPTG+DNA 複合体、LacR-ONPF+DNA 複合体の大局的安定構造を探索するため、これらの構造に対し、MD 計算を実行する必要がある。

Table 4 Total energies (T.E.) (kcal/mol) of LacR-Ligand+DNA complex and its component parts, and estimated binding energies (B.E.) (kcal/mol) between LacR-Ligand or LacR and DNA. B.E. is estimated as

B.E.= –T.E.(Complex) + T.E.(LacR-Ligand+Water) + T.E.(DNA+Water) – T.E.(Water).

	No-ligand	IPTG	ONPF
Complex	-103831596.7	-104489133.3	-104484705.4
LacR-Ligand + Water	-85414086.3	-86071583.5	-86067279.5
DNA + Water	-25724383.0	-25724383.0	-25724287.3
Water	-7307643.7	-7307622.4	-7307692.1
Binding energy	771.0	789.2	830.8
Difference	0.0	18.2	59.8

これら構造の結合エネルギーの詳細とリガンド結合による LacR と DNA 間の特異的相互作用の変化を明らかにするため、MFMO 法により、LacR の各アミノ酸残基と DNA 間の相互作用エネルギーを解析した。Fig. 22 に示すように、DNA-binding domain のアミノ

酸残基 (3～58 アミノ酸残基[1]) と Linker (59～61 アミノ酸残基) は DNA と大きく引力相互作用をしている。さらに LacR-ONPF+DNA 複合体に関しては、LacR の Arg118 が DNA と大きく引力相互作用をしている。まず、Fig. 22 で最も大きく引力相互作用を示している LacR の Arg22 について解析を行った。Arg22 はいずれの複合体構造においても DNA と大きく引力相互作用を示している。実験の先行研究[29-31]は、Arg22 が operator DNA を認識するために重要なアミノ酸残基であることを示唆している。従って、今回の計算結果とこれらの実験結果は比較可能である。

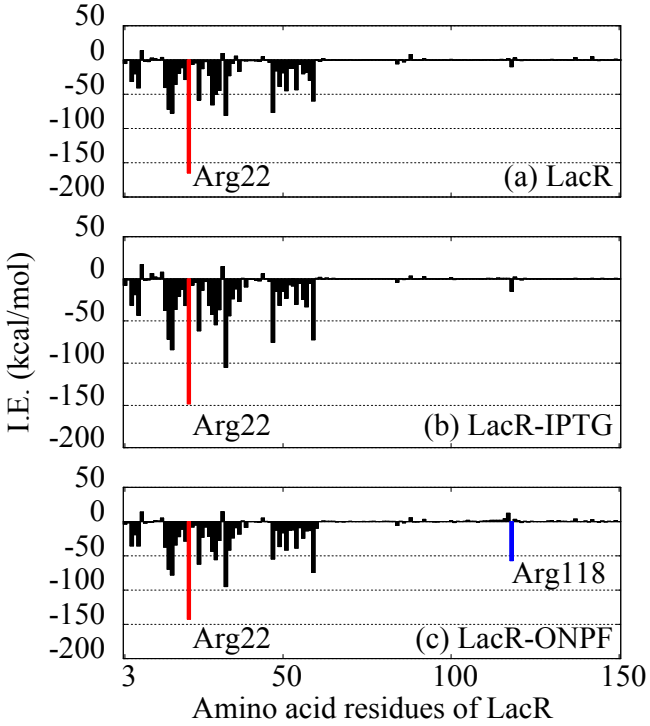


Fig. 22 Interaction energies between DNA + Na^+ + water and each amino acid residue of LacR: (a) LacR (no-ligand), (b) LacR-IPTG, and (c) LacR-ONPF.

さらに、我々は Arg22 が DNA 二重鎖、及びカウンターイオン Na^+のどの部位と相互作用しているのかを明らかにするため、Arg22 と各 DNA 塩基間の相互作用を解析した。Fig. 23 は Arg22 と(a) 各 DNA 塩基+Na^+、(b) 各 DNA 塩基、(c) 各 Na^+間の相互作用エネルギーを示したグラフで、それぞれの複合体の違いを比較した。Fig. 23 (a)の DNA 塩基+Na+において、Arg22 は 2 番目のチミン(Thy2)と 15 番目のグアニン(Gua15)と強く引力相互作用を示している。Fig. 23 (b)より、これらの-50 kcal/mol 以上の引力相互作用は

[1] LacR の DNA-binding domain は 1～58 アミノ酸残基で構成される。しかし、FMO 計算では、フラグメント分割により、N 末端のアミノ酸残基の電荷は本来のアミノ酸残基の電荷に 1 が加えられる。そのため、N 末端のアミノ酸残基は、他のアミノ酸残基や DNA 塩基と、本来ならば存在しない静電相互作用をすることがあるため、N 末端のアミノ酸残基は無視して解析するのが一般的である[58]。

DNA 塩基に由来していることが明らかになった。さらに、Fig. 23 (a)において、複合体間で比較すると、リガンドが結合していない LacR 複合体(LacR+DNA)の Arg22 と Gua15 間の相互作用エネルギーは、リガンドが結合した LacR 複合体のそれらの相互作用エネルギーに比べ、約 10 kcal/mol 大きく、逆にリガンドが結合していない LacR 複合体(LacR+DNA)の Arg22 と Thy14 間の相互作用エネルギーは、リガンドが結合した LacR 単量体のそれらの相互作用エネルギーに比べ、約 10 kcal/mol 小さい。この相互作用エネルギーの違いの原因を明らかにするため、Fig. 24 に示すこれら複合体内の Arg22 と DNA 間の水素結合の構造を比較した。

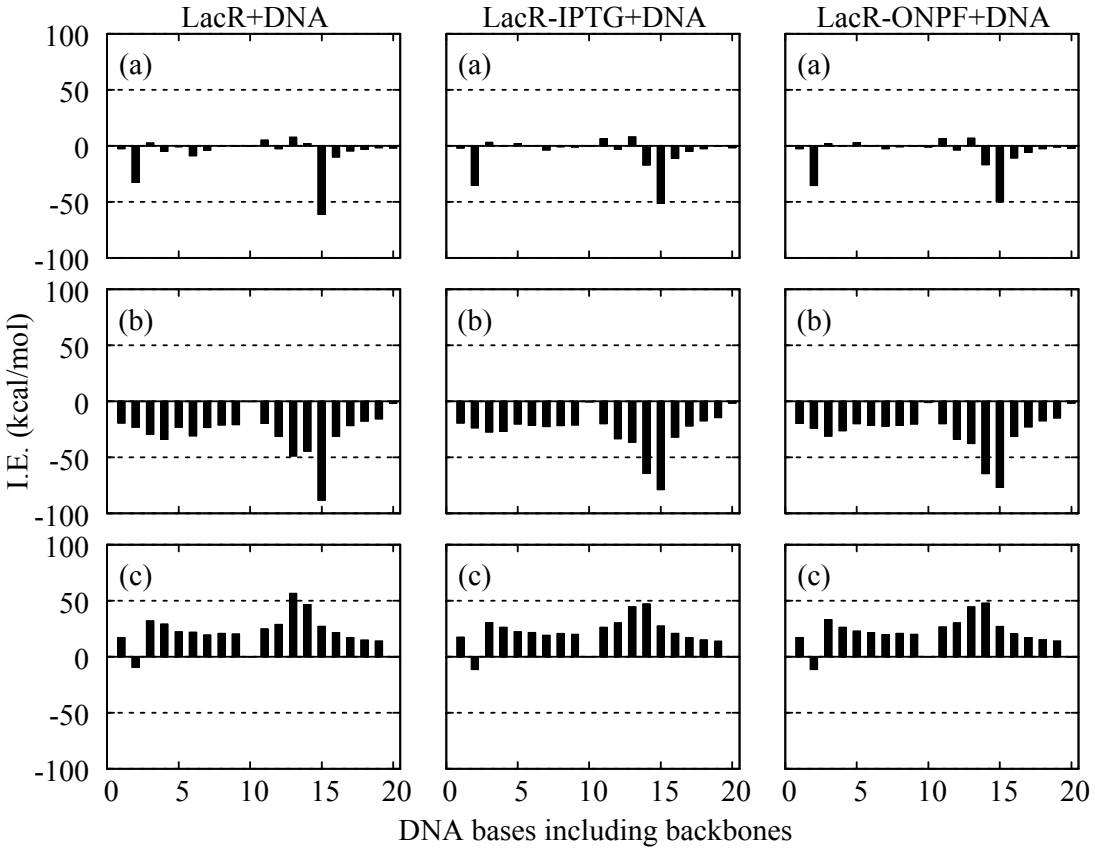


Fig. 23 Interaction energies between Arg22 of LacR and (a) each DNA base + Na^+, (b) each DNA base, and (c) each Na^+.

リガンドが結合した LacR 複合体では、Arg22 と Gua15 間に 1 つの水素結合が形成されている。一方、リガンドが結合していない LacR 複合体(LacR+DNA)の Arg22 は Gua15 に対し、2 つの水素結合を形成している。その結果、リガンドが結合していない LacR 複合体(LacR+DNA)の Arg22 と Gua15 間の相互作用エネルギーは、リガンドが結合している LacR 複合体に比べ、大きくなる。さらに、Fig. 24 は、リガンドが結合している LacR 単量体では、Arg22 と Thy14 間に水素結合が形成されているが、リガンドが結合してい

ないLacR単量体(LacR+DNA)ではそれが存在しないことも示した。Fig. 24に示すように、Arg22とGua15 / Thy14間のこれらの水素結合はArg22の側鎖の向きに依存する。従って、Arg22の側鎖の向きで、LacRとDNA間の特異的な相互作用が大きく変化すると考えられる。

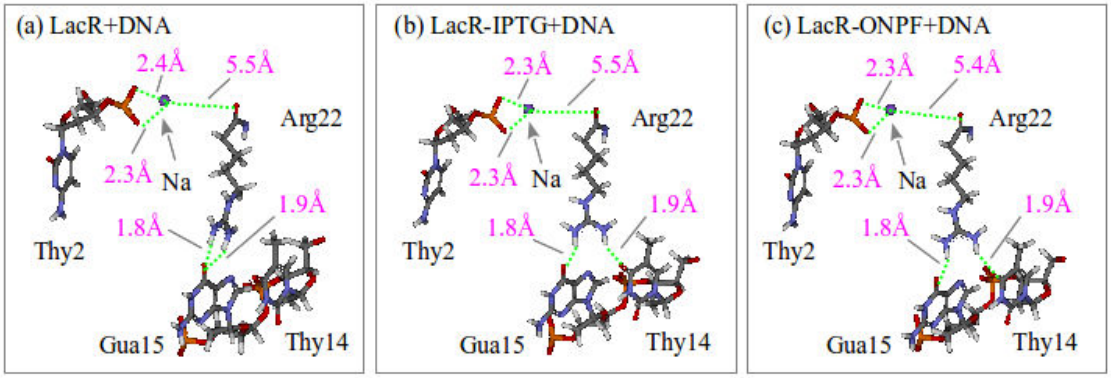


Fig. 24 Structures of hydrogen bondings (green dotted lines) between Arg22 of LacR and DNA bases (Thy2, Thy14, and Gua15) in (a) LacR+DNA, (b) LacR-IPTG+DNA, and (c) LacR-ONPF+DNA. The distance between the oxygen and the hydrogen atoms contributing to the hydrogen bonding is shown in pink. In addition, the distance between Na^+ and the oxygen atoms of Arg22 and Thy2 is shown.

我々はさらに、Arg118とDNA間の引力相互作用についても解析した。Fig. 22 (c)に示す通り、Arg118はLacR-ONPF+DNA複合体特有の引力相互作用であり、これにより、ONPFがLacRに与える影響が明らかになると考えられる。Fig. 25は、Arg118と(a) DNA塩基+Na^+、(b) DNA塩基、(c) Na^+間の相互作用エネルギーを示したグラフである。Arg118は、DNA塩基+Na^+において大きく相互作用していない。そこで、MFMO法により、Arg118とその周囲の水分子間の相互作用エネルギーを解析した。Fig. 26 (a)に示すように、Arg118はいずれの複合体においても、周囲の水分子と主に引力相互作用をしている。50～80番目の水分子はその傾向が顕著であり、中には-15 kcal/mol以上で引力相互作用している水分子も存在する。また、LacR-ONPF+DNA複合体のArg118は、他の複合体よりも水分子と引力相互作用をしている。これが原因でDNAと強く引力相互作用していると考え、Arg118とこれら50～80番目の水分子が存在する領域の構造の解析をした。Fig. 27に示すように、Arg118はFig. 26 (a)で引力相互作用していた水分子と水素結合を形成していた。また、それらの水分子を介して、DNAのCyt20と結合していた。これは、Fig. 26 (b)のArg118とCyt20が共通の水分子と強く引力相互作用していたことからもArg118は水分子を介してDNAのCyt20と結合していると示されている。従って、Arg118とDNAが約-50 kcal/molの引力相互作用をしていた原因は、Arg118が架橋水を介して、DNAと結合しているためであることが明らかになった。

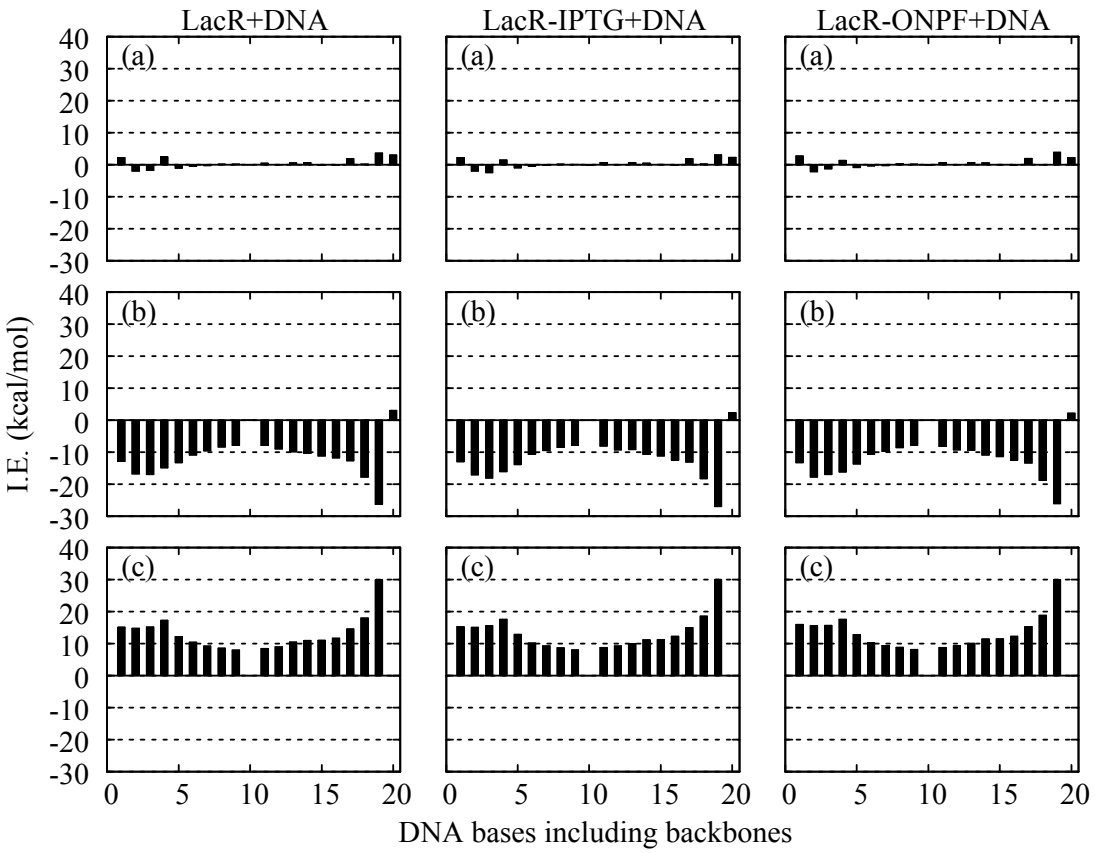


Fig. 25 Interaction energies between Arg118 of LacR and (a) each DNA base + Na^+, (b) each DNA, and (c) each Na^+.

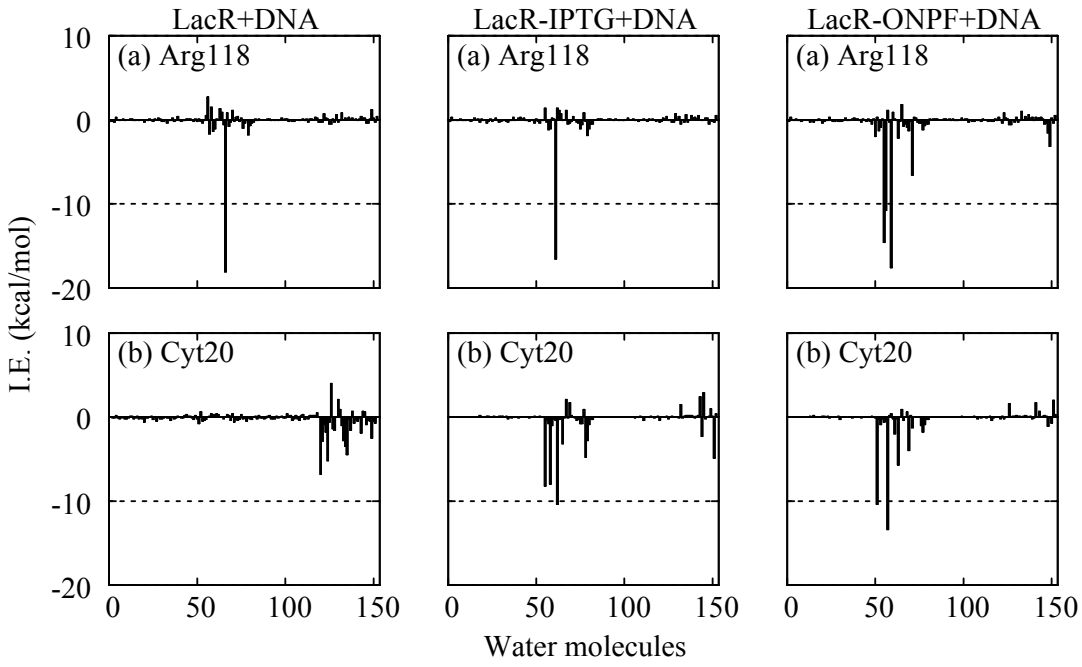


Fig. 26 Interaction energies between (a) Arg118 of LacR, (b) Cyt20 of DNA, and each water molecule for the solvated structures of the LacR+DNA, LacR-IPTG+DNA and LacR-ONPF+DNA complexes.

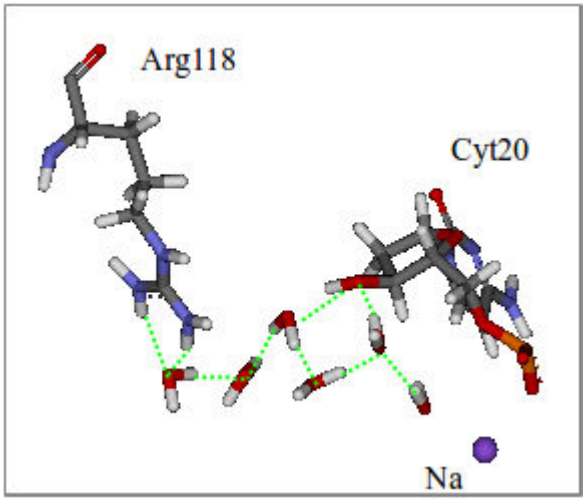


Fig. 27 Structure of hydrogen bondings (green dotted lines) between Arg118 of LacR and Cyt20 of DNA bridged by water molecules.

4.1.4. MM 法により得た複合体構造に対する解析のまとめ

今回、古典分子力学計算によって水和したリガンドが結合していない LacR+DNA 複合体、インデューサ IPTG が結合している LacR+DNA 複合体、アンチインデューサ ONPF が結合している LacR+DNA 複合体構造を得た。これらの構造の比較より、LacR にイン

デューサ、及びアンチインデューサが結合すると、LacR の Gln60 の構造が変化することが明らかになった。特に、LacR-ONPF+DNA 複合体の Gln60 は、Fig. 17 (b)に示すように、Arg118 と水素結合を形成していた。一方、LacR-IPTG+DNA 複合体では、リガンド結合ポケット周囲の構造が変化することが明らかになった。

また、*ab initio* MFMO 計算より、リガンドと LacR のアミノ酸残基間の解析をした。この結果から得られた結合エネルギーから、ONPF が IPTG に比べ、LacR のアミノ酸残基と多くの水素結合を形成していたため、LacR+DNA と ONPF 間の結合エネルギーは、LacR+DNA と IPTG 間の結合エネルギーに比べ強く結合していることが明らかになった。この水素結合数の違いは、ONPF が親水性のニトロフェニル基を持ち、LacR のアミノ酸残基と水素結合を形成している架橋水と水素結合を形成していることに対し、IPTG のイソプロピル基は疎水性で周囲の水分子と水素結合を形成していなかったためである。さらに、LacR のアミノ酸残基とリガンド間の相互作用エネルギーの解析から、Asp149 が IPTG、ONPF との結合に重要であることも明らかにした。その上、リガンドの周囲に存在する水分子が LacR とリガンド間の結合に架橋水という形で寄与していることも明らかにした(Fig. 21)。

最後に、リガンド結合が LacR と DNA 間に与える影響を明らかにするため、MFMO 法により、LacR+DNA 複合体、LacR-IPTG+DNA 複合体、LacR-ONPF+DNA 複合体のアミノ酸残基と DNA 間の相互作用エネルギーを解析した。その結果、LacR と DNA 間の結合に Arg22 が重要であり、Arg22 が DNA の Thy2 と Gua15 と引力相互作用していることを明らかにした。さらに、LacR-ONPF+DNA 複合体において、LacR の Arg118 が、LacR と DNA 間の結合を強めるために重要なアミノ酸残基であることも明らかにした。また、結合エネルギーから LacR-ONPF+DNA 複合体におけるアンチインデューサ ONPF の効果を説明することができたが、LacR-IPTG+DNA 複合体については、結合エネルギーからインデューサ IPTG の効果を説明することができなかった。これは、今回扱った構造が MM 法により局所的安定構造であった可能性を示唆している。この問題を解決するため、MD 法により大局的安定構造を得る必要がある。

4.1.5. MD 法により得た複合体の水和構造及び特異的相互作用

10 ns MD 計算を経て、リガンド結合が LacR の構造に与えた影響を解明するため、MM 法により得た LacR 構造と各時間における LacR 構造のアミノ酸の Cαの RMSD を解析した。Fig. 28 は、MM 法で最適化した複合体構造と MD 計算における各時間の複合体構造の Cαの RMSD を示している。このグラフより、LacR、LacR-IPTG は、LacR-ONPF に比べ、構造変化が大きいことが明らかとなった。さらに LacR のどの部位の構造が大きく変化しているのかを解析するため、MD 計算を実行する前の最適化構造と MD 計算を 10 ns 実行した後の最適化構造の Cαの displacement を比較した。Fig. 29 に示すように、LacR (a) は 40～46 残基の構造が大きく変化し、LacR-IPTG (b) は 30～34 残基の構造が大きく変化することが明らかになった。一方、LacR-ONPF は DNA-binding domain に変化があるものの、LacR や LacR-IPTG ほど大きな変化はなかった。そこで、LacR の 40～46 残

基、LacR-IPTG の 30～34 残基について、どのような構造変化が起こったのかを解析した。Fig. 30 は LacR の構造 (a) とその構造の DNA binding domain 周囲の構造の MD 計算前 (b) と計算後 (c) の構造を示している。これを見ると、DNA から約 9.5 Å の位置に存在した 40～46 のアミノ酸残基が、MD 計算後に DNA から約 13.1 Å の位置まで構造変化している。40～46 のアミノ酸残基は、Fig. 30 (a) に示すように、DNA-binding domain の一部ではあるが、DNA と接しているα-ヘリックスよりも DNA より離れた位置に存在し、DNA と結合していない。また、helix-turn-helix という構造的に揺らぎやすい構造となっている。さらに、周囲に 40～46 のアミノ酸残基に対し、構造変化を誘発する要素が、溶媒水以外ないことから、40～46 のアミノ酸残基の構造変化は熱の揺らぎの影響によるものだと考えられる。また、Fig. 31 に示すように、LacR-IPTG の 30～34 のアミノ酸残基も helix-turn-helix の turn 部位であり、DNA とわずかに相互作用しているだけであるため、この部位も熱の揺らぎの影響を受け、構造が変化したと考えられる。

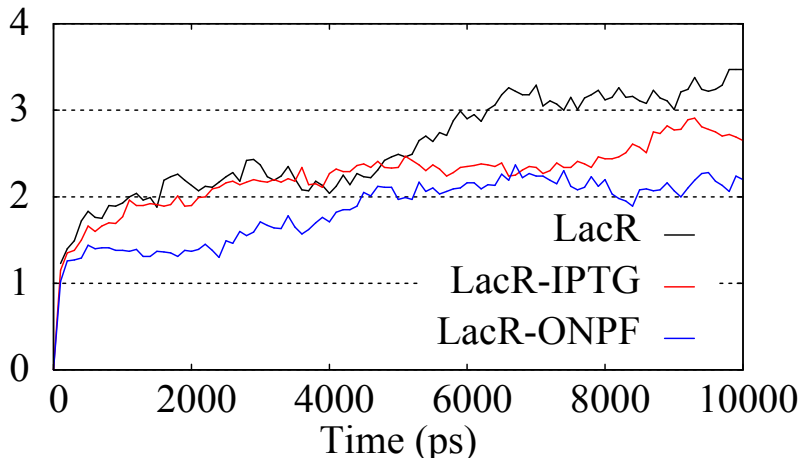


Fig. 28 Change in RMSD of amino acid residues of LacR during MD simulations.

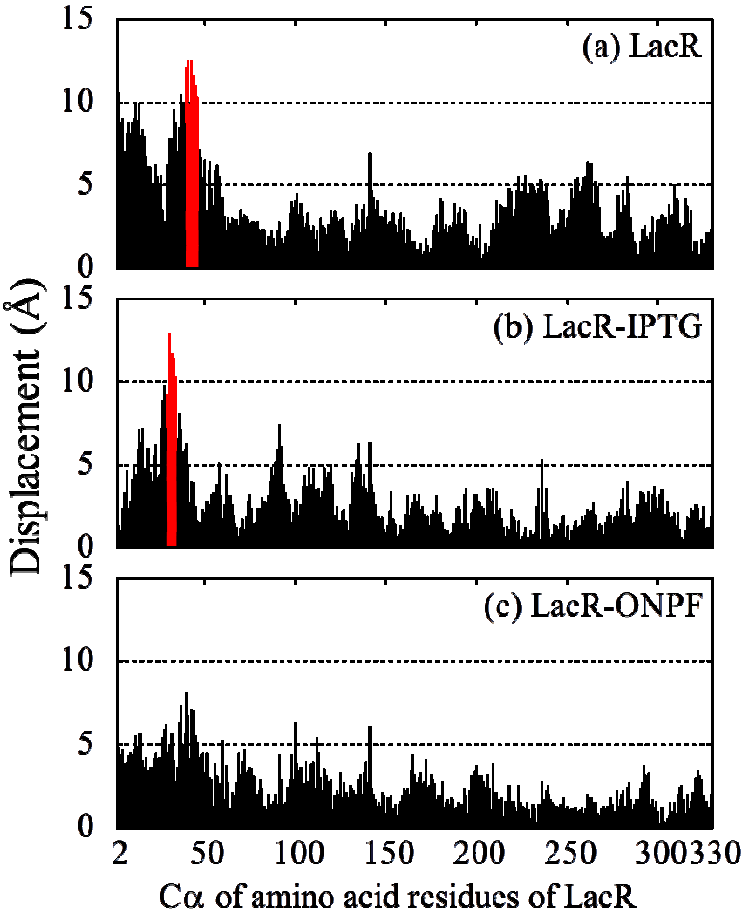


Fig. 29 Displacement of Cα atom of each residue in LacR obtained by MD simulations. The red-marked parts are 40-46 reisudes in LacR and 30-34 residues in LacR-IPTG.

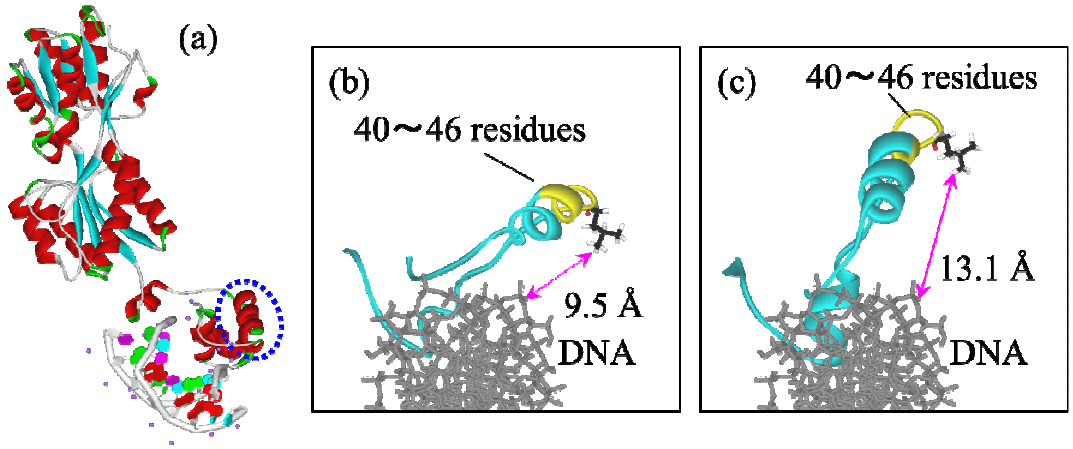


Fig. 30 Structures of (a) LacR+DNA complex, (b) DNA binding domain of LacR and DNA in the MM-optimized structure, and (c) DNA binding domain of LacR and DNA obtained by MD simulation.

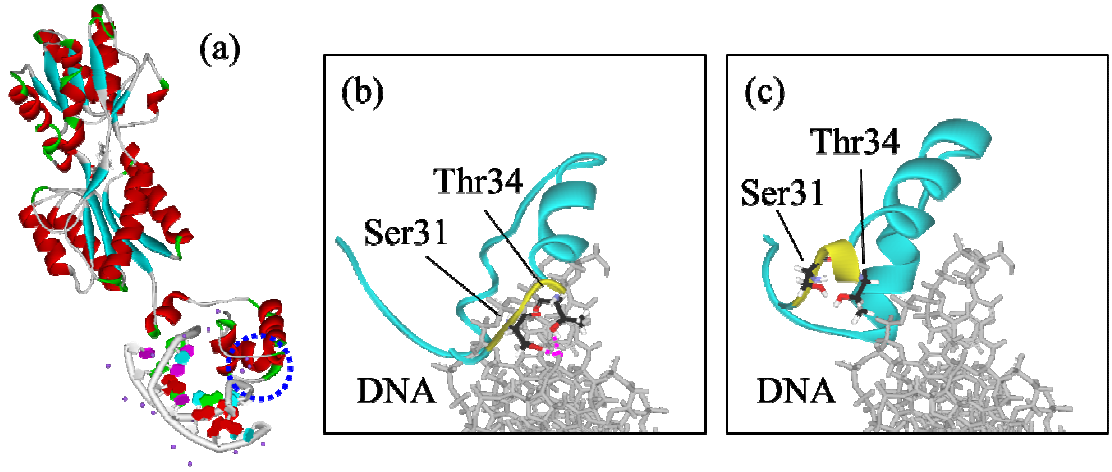


Fig. 31 Structures of (a) LacR-IPTG + DNA complex, (b) DNA binding domain of LacR-IPTG and DNA in the MM-optimized structure, and (c) DNA binding domain of LacR-IPTG and DNA obtained by MD simulation.

リガンド結合に重要な LacR のアミノ酸残基を特定するため、MFMO 法により、LacR-Ligand+DNA 複合体の結合エネルギーを求めた。Table 5 は、水和した全体構造、水和した LacR-ligand+DNA、水和したリガンドと溶媒水の全体エネルギーとその差分の結合エネルギーB.E.を示したものである。また、MM 法により得た構造の結合エネルギーと比較するため、その結合エネルギーも示している。Table 5 を見ると、IPTG、ONPF、共に LacR と結合している。さらに ONPF は IPTG よりも 13.8 kcal/mol 強く結合していることが明らかになった。これらの結果は MM 法により得た構造の結果と定性的に一致する。また、MD 法により得た構造は、MM 法により得た構造よりも大きい結合エネルギーとなっており、MD 法によりリガンドと LacR にとって、安定した構造が得られたと言える。この結合エネルギーに寄与しているアミノ酸残基を特定するため、リガンドと LacR のアミノ酸残基間の相互作用エネルギーを解析した。Fig. 32 に示すように、IPTG、ONPF、両リガンドは Asn125、Asp149、Arg197 と引力相互作用、Asp274 と反発相互作用していることが明らかとなった。また、リガンドは複数の水分子とも相互作用しており、MM 法により得た構造の LacR とリガンド間の相互作用と定性的に一致する結果を得た。また、Fig. 33 に示すように、水素結合を形成している構造も類似していることが明らかとなった。

Table 5 Total eneriges (T.E.) (kcal/mol) of LacR-Ligand+DNA complex and its component parts, and estimated binding energies (B.E.) (kcal/mol) between LacR+DNA and ligand. B.E. is estimated as B.E.= –T.E.(Complex) + T.E.(LacR+DNA+Water) + T.E.(Ligand+Water) – T.E.(Water).

	IPTG	ONPF
Complex	-97997710.1	-97946971.3
LacR + DNA + Water	-97292189.8	-97292371.7
Ligand + Water	-1517443.9	-1466547.4
Water	-811993.8	-812031.7
B.E.	70.1	83.9
Difference	0.0	13.8
B.E. (MM)	65.6	75.4
Difference (MM)	0.0	9.8

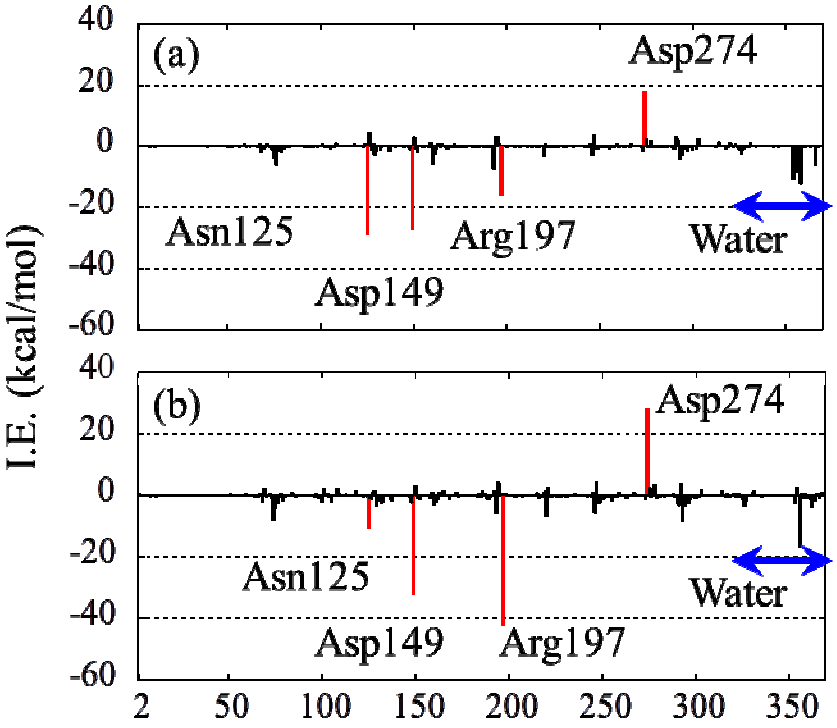


Fig. 32 Interaction energies between each amino acid residue of LacR or water molecule and ligand: (a) IPTG and (b) ONPF.

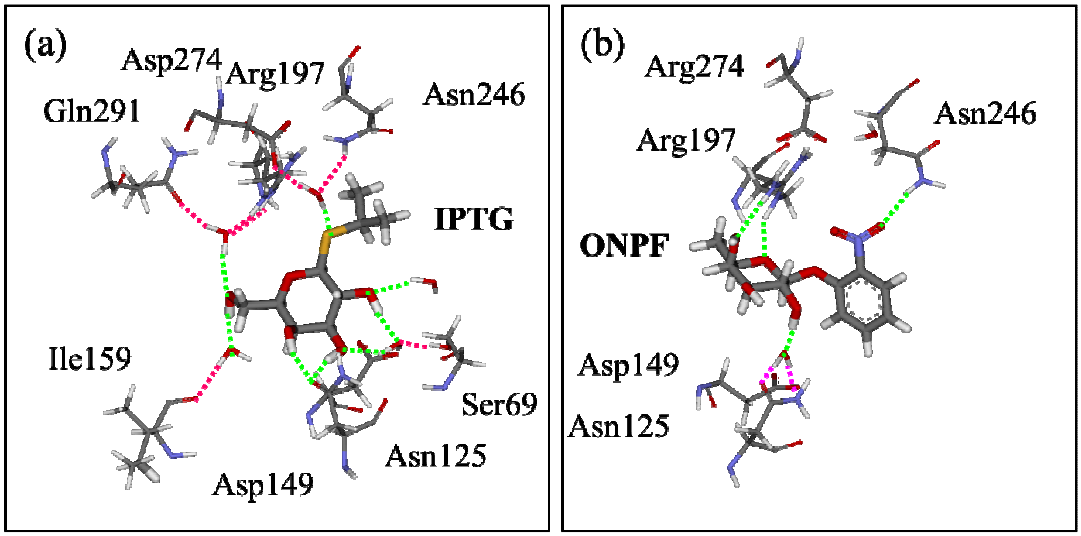


Fig. 33 Structures of hydrogen bonding around (a) IPTG and (b) ONPF. Green and pink dotted lines indicate direct and indirect hydrogen bonds, respectively.

次にリガンド結合が LacR と DNA 間の特異的相互作用に与える影響について解析するため、MFMO 法により、LacR+DNA 複合体、及び LacR-Ligand+DNA 複合体の結合エネルギーを求めた。Table 6 は、水和した LacR-ligand+DNA 複合体、水和した LacR-Ligand 複合体、水和した DNA 複合体、水分子の全体エネルギーと、その差分から求められた結合エネルギー B.E.を示したものである。

Table 6 Total eneriges (T.E.) (kcal/mol) of LacR-ligand+DNA complex and its component parts, and estimated binding energies (B.E.) (kcal/mol) between LacR and DNA. B.E. is estimated as B.E.=–T.E.(Complex) + T.E.(LacR-Ligand+Water) + T.E.(DNA+Water) – T.E.(Water).

	No-ligand	IPTG	ONPF
Complex	-106745710.0	-107449801.2	-107398767.7
LacR-Ligand + Water	-88327320.2	-89031802.2	-88980414.4
DNA + Water	-28638931.0	-28638825.0	-28638577.7
Water	-10221225.0	-10221422.1	-10221076.4
B.E.	683.7	596.1	851.9
Difference	0.0	-87.6	168.2
B.E. (MM)	771.03	789.2	830.79
Difference (MM)	0	18.17	59.76

また、リガンド結合が LacR と DNA 間の特異的相互作用に与える影響をわかりやすくするため、リガンドが結合していない LacR 複合体とリガンドが結合した LacR 複合体の差 Difference も示した。さらに、MM 法により得た LacR 単量体複合体の結果と比較するため、その結合エネルギーも示した。MM 法により得た構造の LacR と DNA 間の結合エネルギーは、IPTG が LacR と DNA 間の結合を弱め、ONPF が LacR と DNA 間の結合を強

める実験の結果をうまく説明することができなかった。しかし、MD 法により得た構造の結合エネルギーは、LacR (リガンドなし) の複合体 (B.E.: 771.03 kcal/mol) に比べ、IPTG と結合した LacR の LacR と DNA 間の結合エネルギーは弱まり、ONPF と結合した LacR の LacR と DNA 間の結合エネルギー (B.E.: 851.9 kcal/mol) が強まる実験結果をうまく説明することができる結果を得ることができた。このことから、LacR と DNA 間の結合を解析するためには、MD 計算が不可欠であることが明らかとなった。

さらに、LacR、DNA のどの部位が LacR と DNA 間の結合に関与しているのかを解明するため、それぞれの相互作用エネルギーI.E.を解析した。Fig. 34 は、水和した DNA+Na^+ 複合体に対する各アミノ酸残基の相互作用エネルギーを示している。MM 法により得た構造で、LacR と DNA 間の結合に重要と指摘した Arg22 は、MD 法により得た構造でも DNA と強く引力相互作用しており、手法によらず、DNA との結合に重要であることが再度確認された。その他のアミノ酸残基の相互作用エネルギーも MM 法により得た構造の相互作用エネルギーと定性的に類似した結果となった。そこで、リガンドなしの LacR の相互作用エネルギーの結果と他の LacR 複合体の相互作用エネルギーの結果の差分を解析した所、IPTG が結合することにより、Glu11、Tyr47 の相互作用エネルギー (Fig. 34 の緑で示したアミノ酸残基) が弱まり、ONPF が結合することにより Pro49 の相互作用エネルギーが強まり、Lys59 の相互作用エネルギーが弱まる (Fig. 34 の青で示したアミノ酸残基) ことが明らかになった。

この原因を解明するため、相互作用エネルギーが変化したアミノ酸残基に対して、DNA の塩基+Na^+、塩基、Na^+フラグメントの相互作用を解析した。Fig. 35 は、Glu11 と DNA の各フラグメント間の相互作用を示したものであり、左がリガンドなしの LacR、右が IPTG が結合した LacR の相互作用エネルギーを示している。Fig. 35 に示すように、Glu11 は、DNA 塩基ではなく、カウンターイオンと引力相互作用している。そして、カウンターイオンとの相互作用は IPTG 結合により弱まっている。この原因を解明するため、Glu11 とその周囲の構造を解析した所、Fig. 36 に示すように、リガンドがない状態 (a) のカウンターイオンは、Glu11 から 4.1 Å の位置存在しているが、IPTG が結合した状態 (b) のカウンターイオンは Glu11 から 8.8 Å の位置に存在しており、カウンターイオンが Glu11 から離れている。カウンターイオンは、熱の揺らぎの影響を受けやすく、水中では負電荷の部位付近に存在するものの、その位置は固定化されていない。従って、この相互作用エネルギーの変化は、IPTG 結合による影響ではなく、熱の揺らぎによるものだと考えられる。

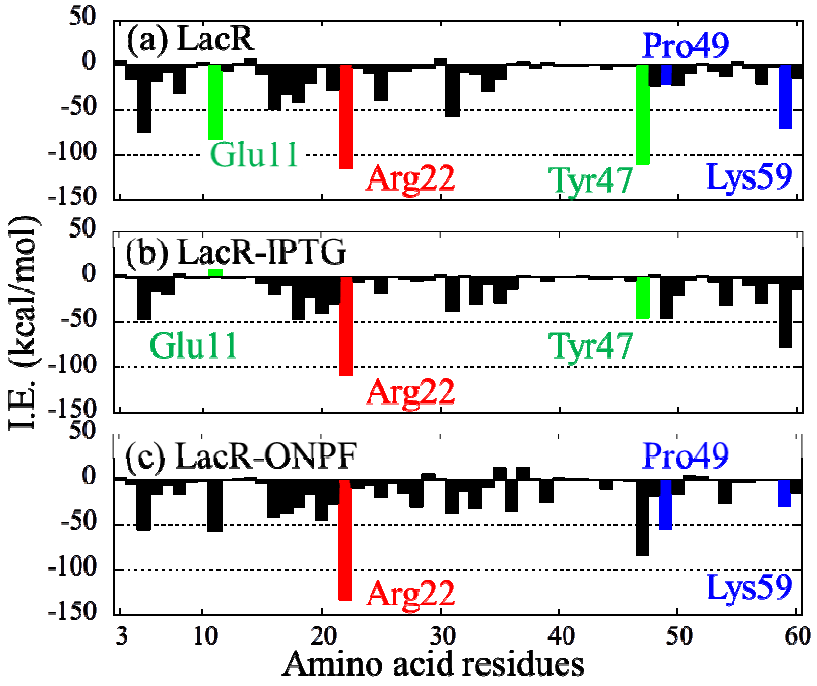


Fig. 34 Interaction energies between each amino acid residue of LacR and DNA+Na^++water: (a) LacR (no-ligand), (b) LacR-IPTG, and (c) LacR-ONPF. A red bar indicates Arg22, green and blue bars indicate the amino acid residues whose interaction energy is changed significantly by the ligand binding.

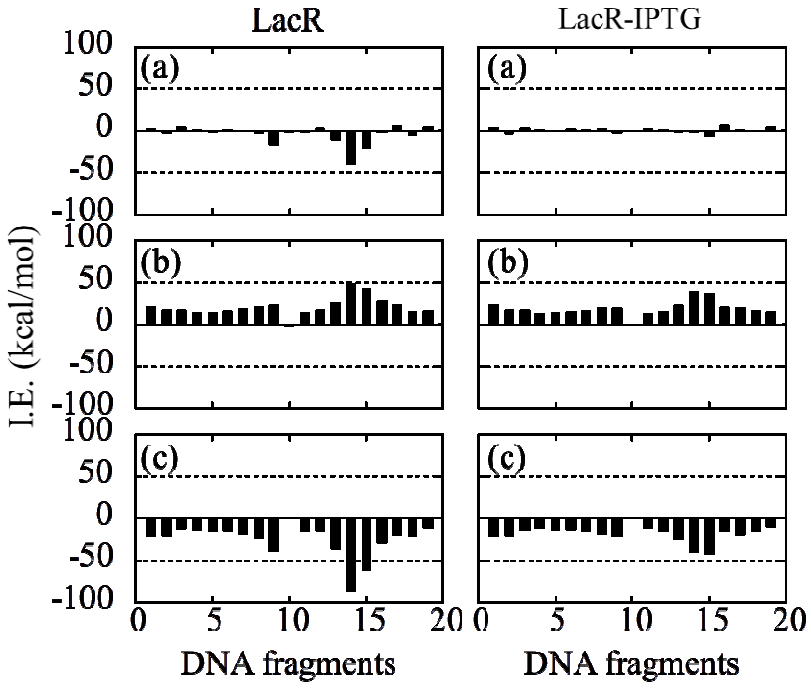


Fig. 35 Interaction energies between Glu11 of LacR and (a) each DNA base + Na^+, (b) each DNA base, and (c) each Na^+.

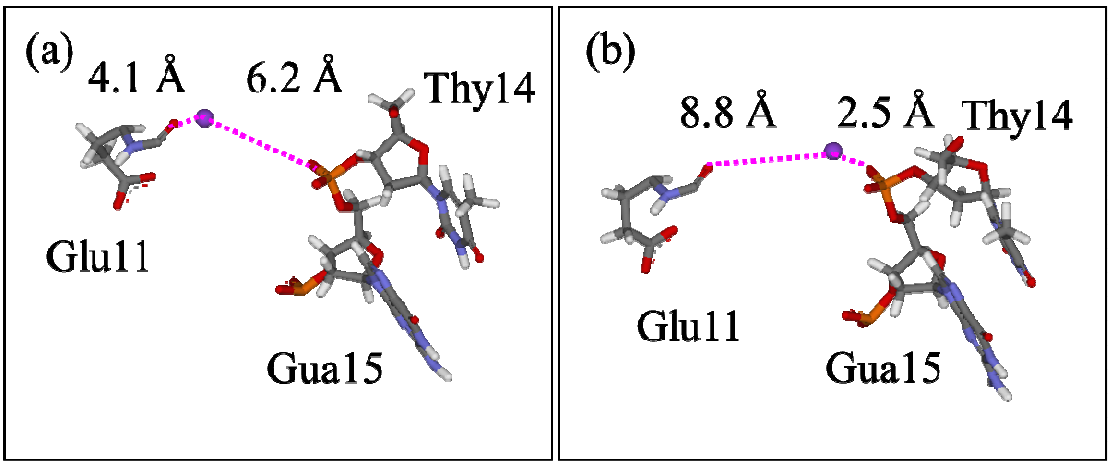


Fig. 36 Structures of electrostatic interactions between Glu11 of LacR and DNA bases (Thy14 and Gua15) in (a) LacR+DNA and (b) LacR-IPTG+DNA complexes. Pink dotted lines indicate the electrostatic interaction.

Glu11 と同様に、Tyr47 も原因を解析した。Fig. 37 に示すように、Tyr47 はカウンターイオンだけでなく、DNA 塩基も引力相互作用に寄与している。最も Tyr47 と DNA 間の引力相互作用に寄与している DNA の Gua1 と Tyr47 間の相互作用は、LacR が-14.2 kcal/mol、LacR-IPTG が-11.7 kcal/mol とあまり変化がない。しかし、Tyr47 とカウンターイオン間の相互作用エネルギーは LacR が-95.4 kcal/mol、LacR-IPTG が-33.2 kcal/mol となっており、Tyr47 と DNA 間の相互作用エネルギーの変化に対し、カウンターイオンの寄与が大きい。そこで、Tyr47 周囲の構造を解析した所、Fig. 38 に示すように、Tyr47 も Glu11 同様、カウンターイオンの位置が大きく変化していた。従って、Tyr47 に関しても IPTG 結合の影響ではなく、熱の揺らぎによりカウンターイオンが移動したため、相互作用エネルギーが変化したと考えられる。

次に、ONPF 結合の影響が原因で相互作用エネルギーが変化したと考えられる Pro49 について解析をした。Fig. 39 に示すように、Pro49 の相互作用エネルギーの変化もカウンターイオンの寄与が大きいことが明らかとなった。そして、Fig. 40 に示す構造から、Pro49 の相互作用エネルギーの変化も熱の揺らぎによるカウンターイオンの位置の変化によるものであり、ONPF 結合の影響ではないことが明らかとなった。

Lys59 と DNA 間の引力相互作用は、Fig. 41 に示すように、Lys59 と DNA の Cyt2 の相互作用が大きく寄与している。そこで、Lys59 と Cyt2 周囲の構造を解析した所、Fig. 42 に示すように、リガンドが結合していない LacR (a) の状態では、Lys59 と Cyt2 は 1.7 Å と近い位置に存在するが、LacR-ONPF の構造では、Lys59 と Cyt2 は 3.8 Å と距離が離れている。

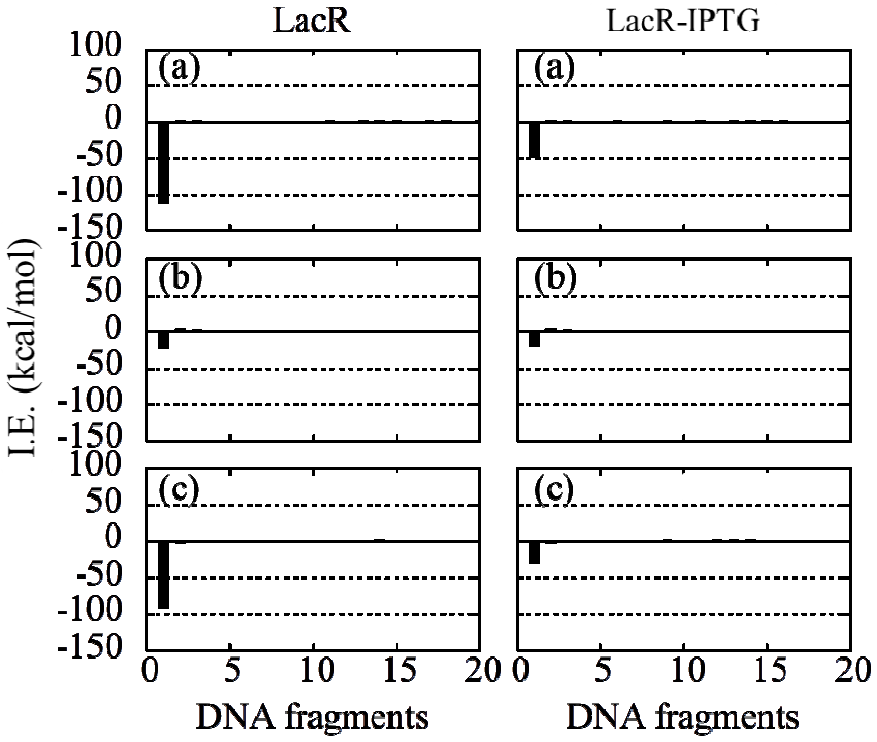


Fig. 37 Interaction energies between Tyr47 of LacR and (a) each DNA base + Na^+, (b) each DNA base, and (c) each Na^+ for the LacR+DNA and the LacR-IPTG+DNA compexes.

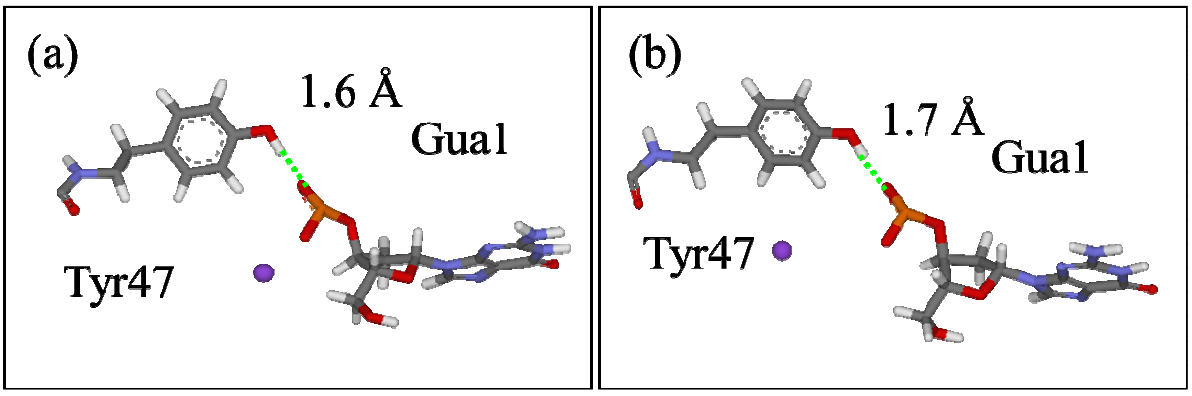


Fig. 38 Structures of hydrogen bonding (green dotted lines) between Tyr47 of LacR and Gua1 of DNA in (a) LacR+DNA and (b) LacR-IPTG+DNA compexes.

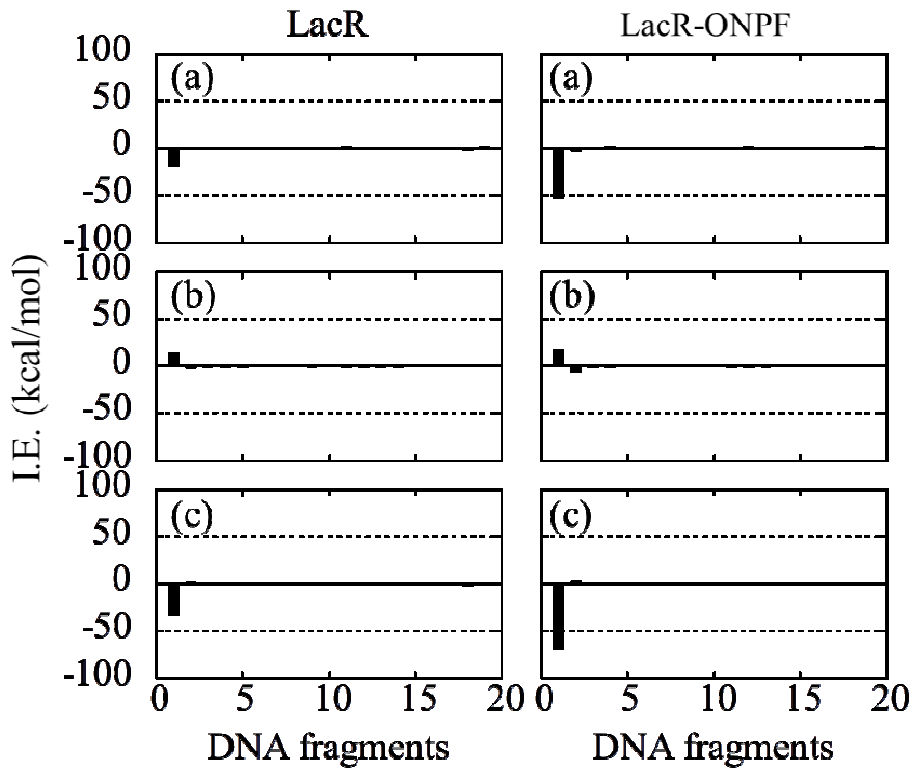


Fig. 39 Interaction energies between Pro49 of LacR and (a) each DNA base + Na^+, (b) each DNA base, and (c) each Na^+ for the LacR+DNA and the LacR-ONPF+DNA compexes.

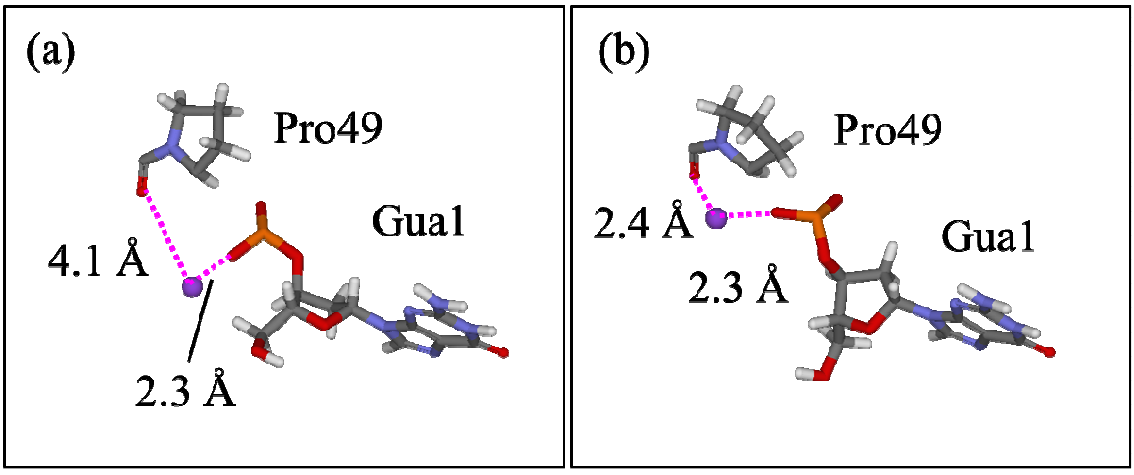


Fig. 40 Structures of electrostatic interactions between Pro49 of LacR and Gua1 of DNA in (a) LacR+DNA and (b) LacR-ONPF+DNA complexes. Pink dotted lines indicate the electrostatic interaction.

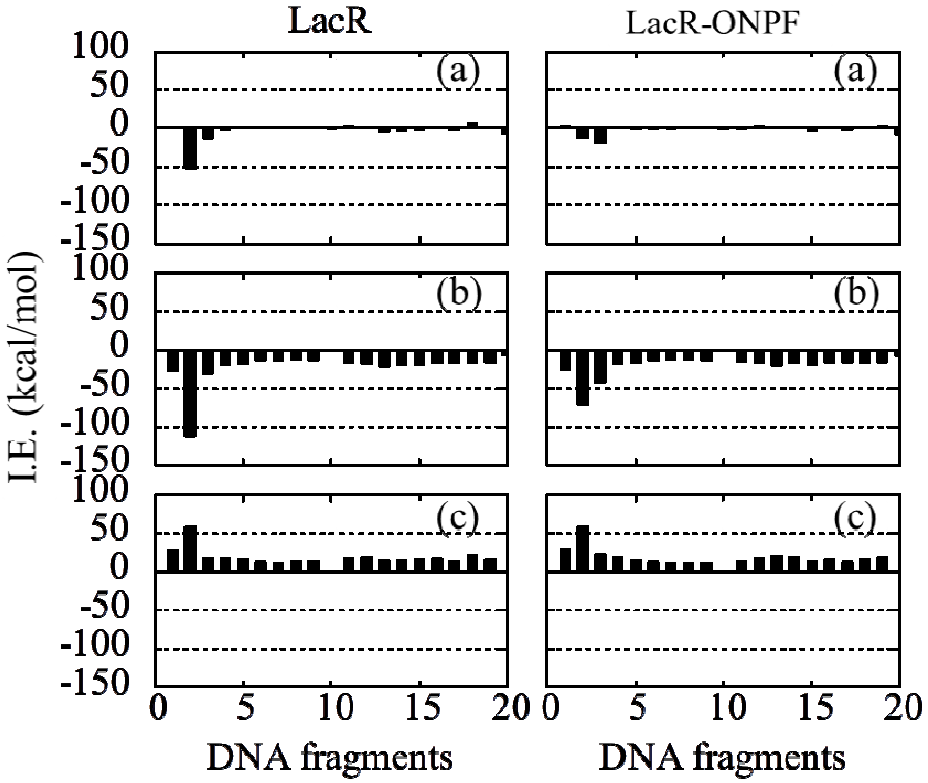


Fig. 41 Interaction energies between Lys59 of LacR and (a) each DNA base + Na^+, (b) each DNA base, and (c) each Na^+ for the LacR+DNA and the LacR-ONPF+DNA compexes.

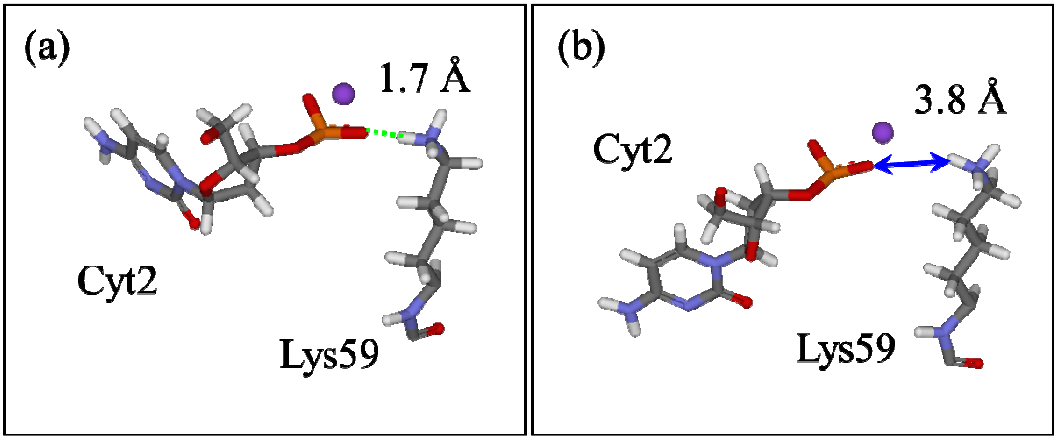


Fig. 42 Structures of hydrogen bonding (green dotted line) between Lys59 of LacR and Cyt2 of DNA in (a) LacR+DNA and (b) LacR-ONPF+DNA complexes.

4.2. LacR 二量体-リガンド+DNA 複合体に対する結果

4.2.1. 複合体の水和構造

30 ns MD 計算を経て、リガンド結合が LacR の構造に与えた影響を解明するため、MM 法により得た LacR 二量体構造と各時間における LacR 二量体構造のアミノ酸の Cαの RMSD を解析した。Fig. 43 は、(a) LacR、(b) LacR-IPTG、(c) LacR-ONPF、それぞれの二量体複合体構造内の Cαについて、二量体の最適化構造とそれぞれの時間における構造間の RMSD を示したものである。Fig. 43 (a) に示す LacR の RMSD は、熱の揺らぎによる構造変化は見られるものの、全体的に大きな構造変化は見られない。MD 計算の始め 0〜3 ns 付近に見られる大きな構造変化は、0 ns からの変化に連続していることから、MD 計算における温度上昇の過程の熱の揺らぎの影響を受けたものであると考えられる。一方、Fig. 43 (b) の LacR-IPTG は、温度上昇過程の影響による構造変化の他に、6〜8 ns 付近に大きな構造変化が観測できた。また、Fig. 43 (c) の LacR-ONPF は、RMSD の値が全体的に大きく、18〜22 ns 付近に大きな構造変化が観測できた。LacR 二量体の MD 計算では、LacR 単量体では観測できなかった特異的な構造変化を観測することができた。

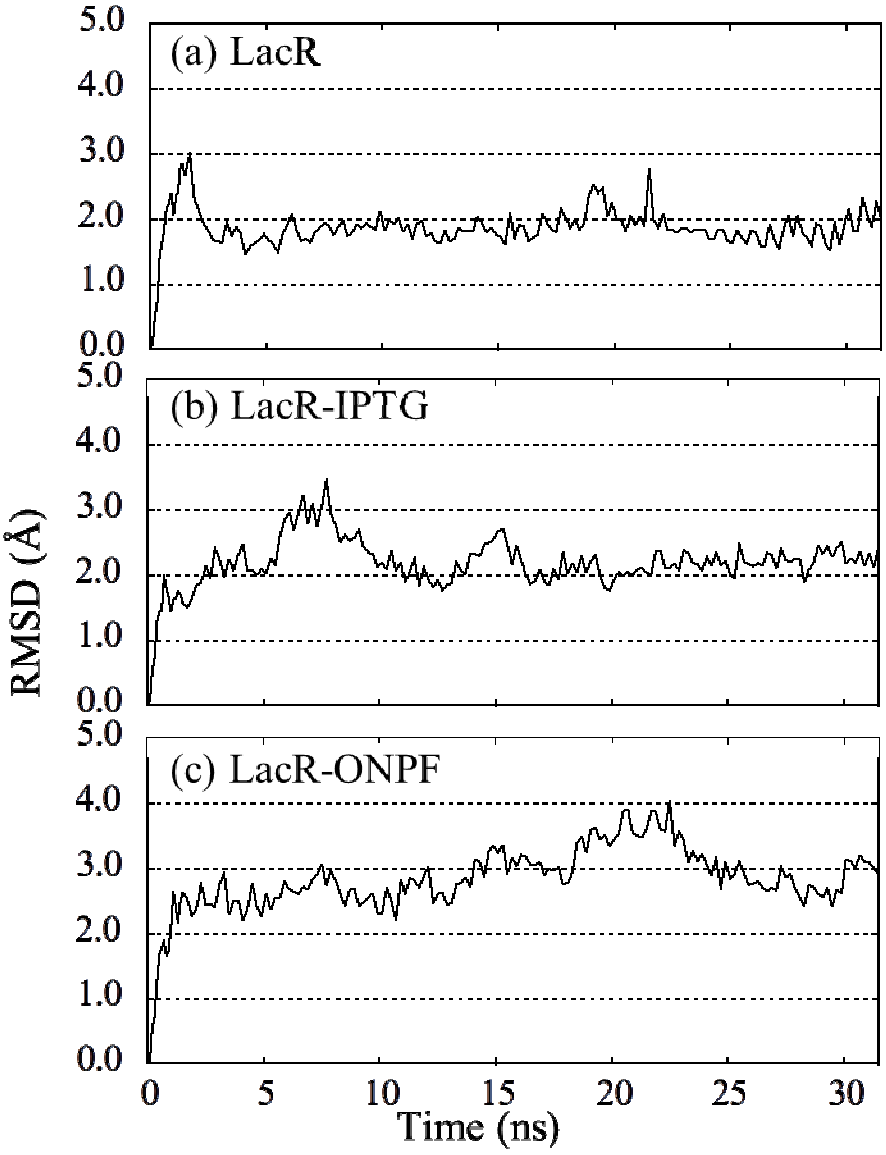


Fig. 43 RMSD of all Cα atoms of LacR between the MM-optimized structure and the structures obtained by the MD simulaton; (a) LacR+DNA, (b) LacR-IPTG+DNA and (c) LacR-ONPF+DNA complexes.

4.2.2. MD 計算で求めた複合体の構造変化

そこで、それぞれの時間における LacR 二量体複合体の構造を解析した。Fig. 44 は、LacR の 1.5、7.5、15.0、22.5、30.0 ns における構造を示している。また、紫、水色の構造はそれぞれ、LacR 単量体の A と B を示している。これと同様に、Fig. 45、Fig. 46 は、それぞれ LacR-IPTG、LacR-ONPF を示している。LacR-ONPF の構造変化は DNA を真横から見た構造だと分かりづらいため、DNA の二重鎖を真上から見た構造 (Fig. 47) を示した。

リガンドが結合していない LacR は、Fig. 44 に示すように、regulatory domain が熱の揺

らぎにより、僅かに揺らいでいるものの、シミュレーション時間全体を通して大きな構造変化をすることはなかった。

LacR-IPTG は Fig. 45 に示すように、熱の揺らぎの影響を受けながら、regulatory domain は徐々に DNA に沿う形で傾き、その傾きは 7.7 ns の構造で約 12.8 度まで傾いた。この 7.7 ns は、Fig. 43 (b) の LacR-IPTG の RMSD が最大となった時間と一致する。その後、ある程度、傾いた状態を保っていたものの 30 ns まで大きな構造変化は見られなかった。

LacR-ONPF は Fig. 43 (c) に示すように LacR-ONPF の RMSD は MD 計算全体を通して、他の LacR 複合体よりも大きな値で推移していた。しかし、Fig. 46 に示すように、複合体の構造からは大きな構造変化を見ることはできなかった。そこで、構造を見る視点を変え、DNA 二重鎖を真上から見るようにした所、Fig. 47 に示すように、LacR-ONPF の regulatory domain に大きな構造変化が観測された。LacR-ONPF の場合、LacR-IPTG の DNA に沿う形で傾く構造変化と異なり、DNA 二重鎖を真上から見た時に、regulatory domain が左右に揺らぐような構造変化をしていた。DNA を中心として LacR 二量体の構造変化の角度を測定した所、1.5 ns の時の LacR-ONPF を基準とすると、LacR-ONPF の regulatory domain は、22.5 ns の構造の約 19.0 度まで傾いた。この 22.5 ns は Fig. 43 (c) の LacR-ONPF の RMSD が最大となった時間と一致する。この時、Regulatory domain の構造が大きく揺らいだにも関わらず、DNA-binding domain は DNA と MD 計算前と同じ状態で結合していた。

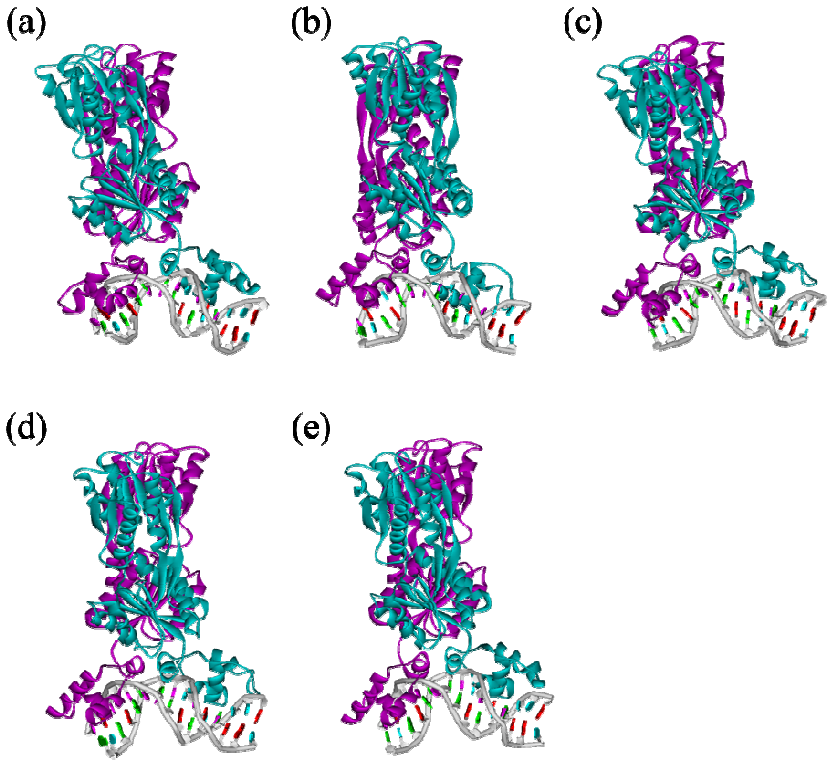


Fig. 44 Change in relative conformation between LacR dimer and DNA in the MD simulation: at (a) 1.5, (b) 7.5, (c) 15.0, (d) 22.5 and (e) 30.0 ns.

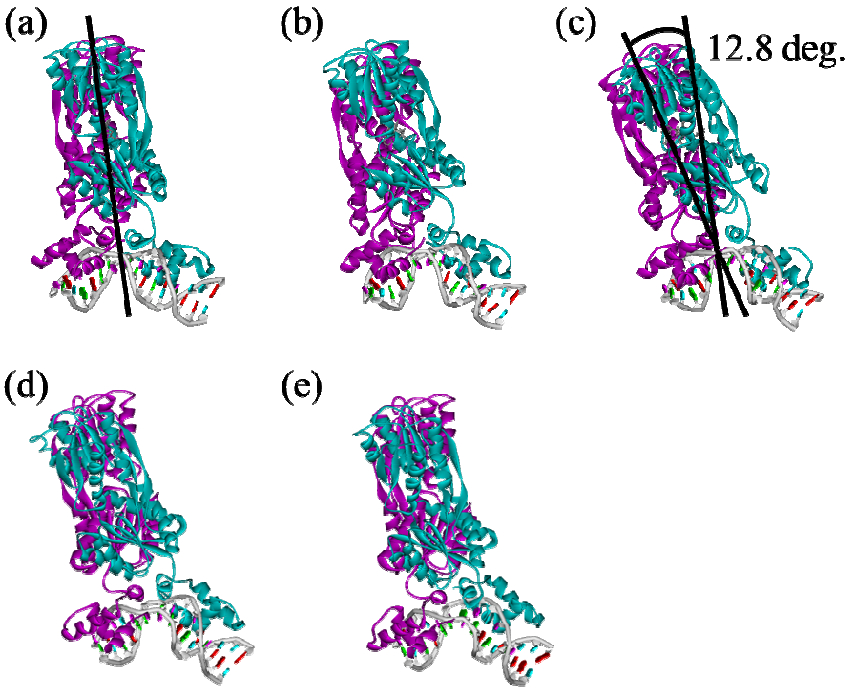


Fig. 45 Change in relatve conformation between LacR-IPTG dimer and DNA in the MD simulation: at (a) 1.5, (b) 5.0, (c) 7.7, (d) 15.0 and (e) 30.0 ns.

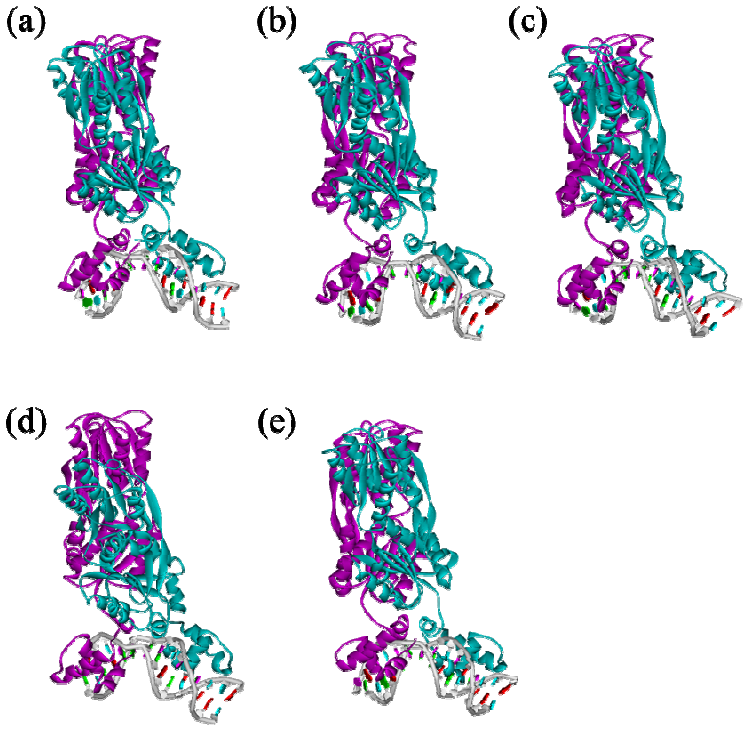


Fig. 46 Change in relatve conformation between LacR-ONPF dimer and DNA in the MD simulation: at (a) 1.5, (b) 7.5, (c) 17.9, (d) 22.5 and (e) 30.0 ns.

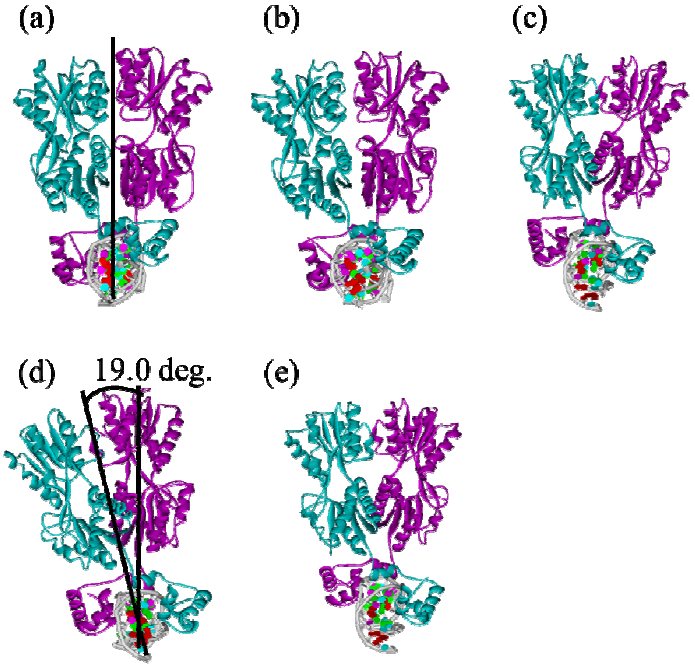


Fig. 47 Change in relatve conformation between LacR-ONPF dimer and DNA in the MD simulation from the other view point of the direction of DNA helix: at (a) 1.5, (b) 7.5, (c) 17.9, (d) 22.5 and (e) 30.0 ns.

4.2.3. リガンド結合による LacR と DNA 間の特異的相互作用の変化

LacR-IPTG の Regulatory domain が傾いた原因を解析するため、リガンドの結合していない LacR と 7.7 ns における LacR-IPTG のアミノ酸残基の Cα の displacement を解析した。Fig. 48 は、その displacement を解析したものであり、(a) と (b) はそれぞれ LacR 単量体の A と B を示している。MD 計算では、計算の途中で複合体が平行移動や回転をしてしまうため、Displacement を比較するにあたり、2 つの構造の Cα が最小になるようにフィッティングした。これにより、2 つの構造間の構造変化以外の座標のずれは無視出来ると考えられる。Fig. 48 よると、DNA-binding domain の構造変化が大きくなっている。これは LacR のアミノ酸残基全体の Cα でフィッティングしたため、330 残基で構成されている LacR 単量体のうち、2-59 残基の DNA-binding domain に LacR-IPTG の傾きによる構造変化の影響が現れたと考えられる。そこで、DNA-binding domain 以外の構造に着目すると、LacR 単量体 A の 310～315 残基、LacR 単量体 B の 100～149 残基、235～240 残基の構造変化が大きい。Fig. 49 を見ると、LacR 単量体 A の 310～315 残基、LacR 単量体 B の 235～240 残基は熱の揺らぎの影響を受けやすい turn 構造であり、構造変化の範囲が数残基程度であるため、これら残基の構造変化は熱の揺らぎによるものだと考えられる。一方、LacR 単量体 B の 100～149 残基は、α-helix 単位の構造変化であるため、リガンドの影響を受けた構造変化であると考えられる。また、100～149 残基の中には、

MM 法や MD 法で得た LacR 単量体複合体を FMO 計算した際の相互作用エネルギーの解析で、リガンドとの結合に重要とされたアミノ酸残基である Asn125、Asp149 が含まれる。これら重要なアミノ酸とリガンドの位置を Fig. 50 に示す。これを見ると、IPTG とα-helix が近い位置に存在していることがわかる。Asn125 と Asp149 は、それぞれ極性アミノ酸、荷電アミノ酸であり、これらアミノ酸残基の方向に対し、IPTG は疎水基であるイソプロピル基を向けている。荷電や極性のような親水性の残基と疎水基は反発相互作用を示す。従って、LacR 単量体 B の 100～149 残基の構造変化は、この反発相互作用によるものであると考えられる。このα-helix の近くには、LacR 単量体 A の DNA-binding domain が存在し、7.7 ns の構造の時点でα-helix と LacR 単量体 A の DNA-binding domain は 3～5 本の水素結合を形成し、結合していた。従って、このα-helix が、Asn125 や Asp149 の反発により、リガンド結合ポケットから押し出されることにより、LacR 単量体 A の DNA-binding domain と相互作用し、最終的に DNA に沿う形で傾いたと考えられる。

一方、LacR-ONPF も ONPF 周囲の構造は LacR-IPTG と類似している。しかし、Fig. 52 に示すように ONPF は、Asn125、Asp149 に親水性のフコースを向けているため、これらアミノ酸残基に対し、水素結合、あるいは静電相互作用をする。従って、これらのアミノ酸残基を含むα-helix をリガンド結合ポケットに引き寄せると考えられる。つまり、IPTG 結合の影響とは逆に、α-helix が LacR 単量体 A の DNA-binding domain と相互作用する確率が低くなると考えられる。

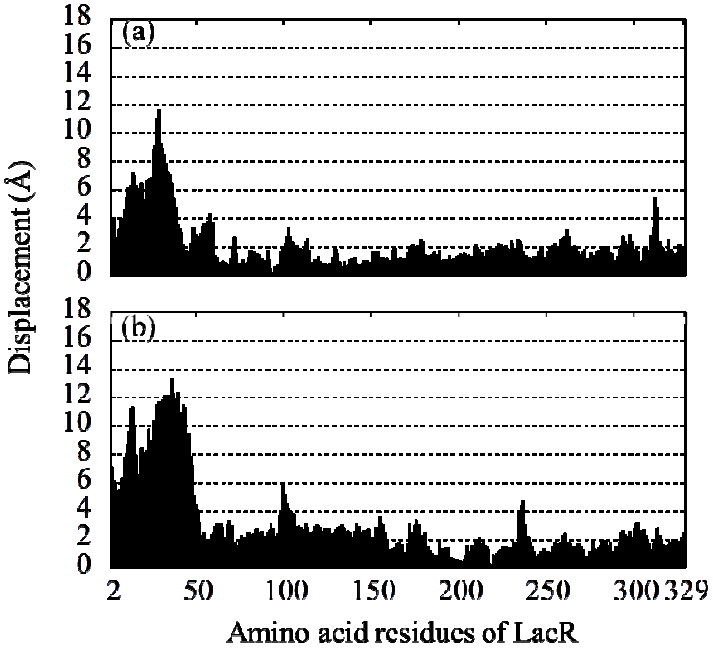


Fig. 48 Displacement of Cα atom of each LacR residue between LacR and LacR-IPTG: (a) in the monomer A and (b) in the monomer B of LacR dimer.

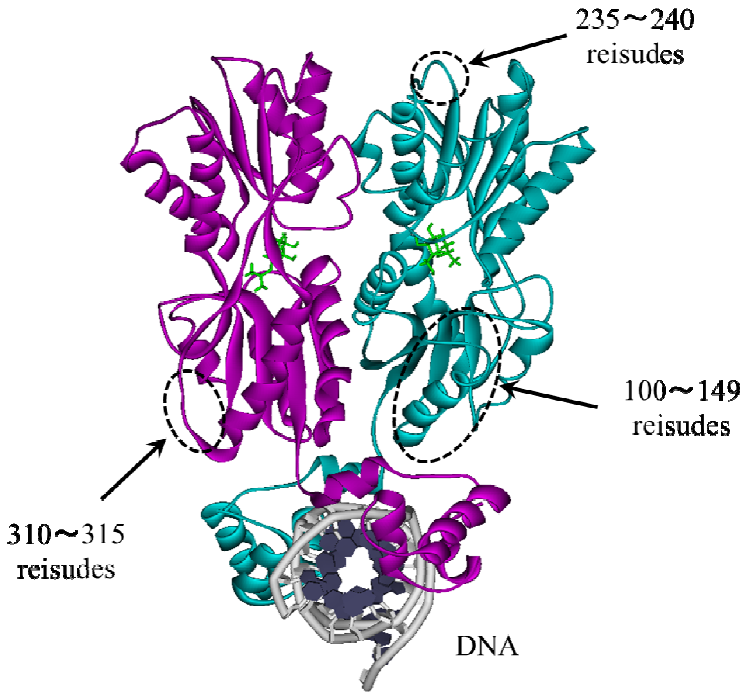


Fig. 49 The positions of residues which conformation changes largely between LacR+DNA and LacR-IPTG+DNA complex.

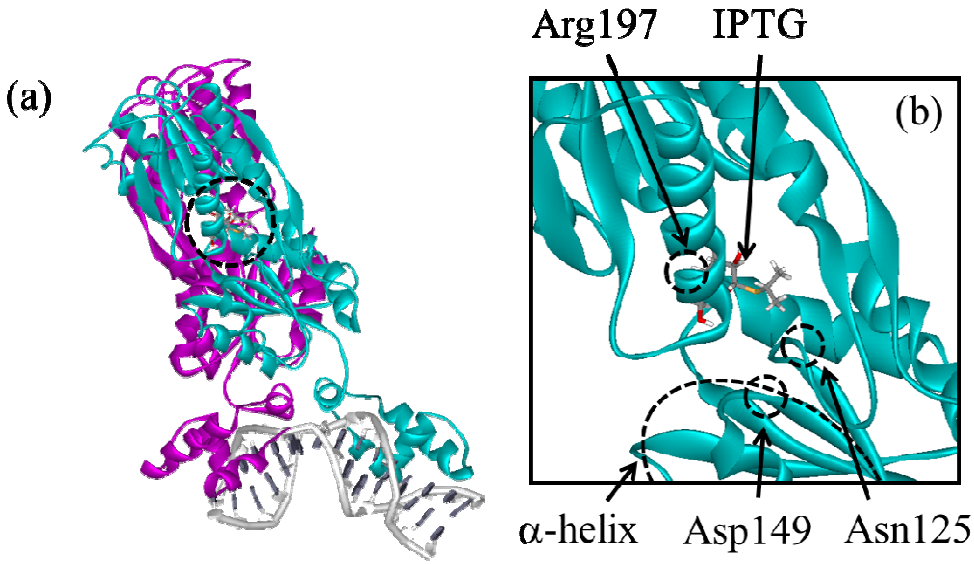


Fig. 50 (a) Structure of the complex with LacR dimer, IPTG and DNA complex obtained by MD simulation at 7.7 ns, and (b) the detailed structure around IPTG in the complex.

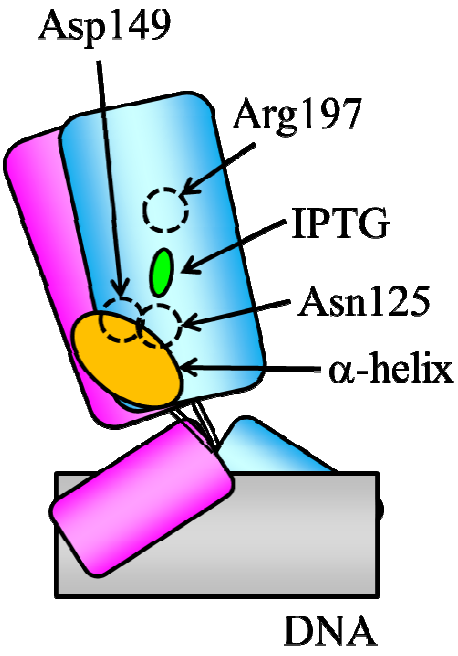


Fig. 51 Schematics view of the complex with LacR-IPTG dimer and DNA.

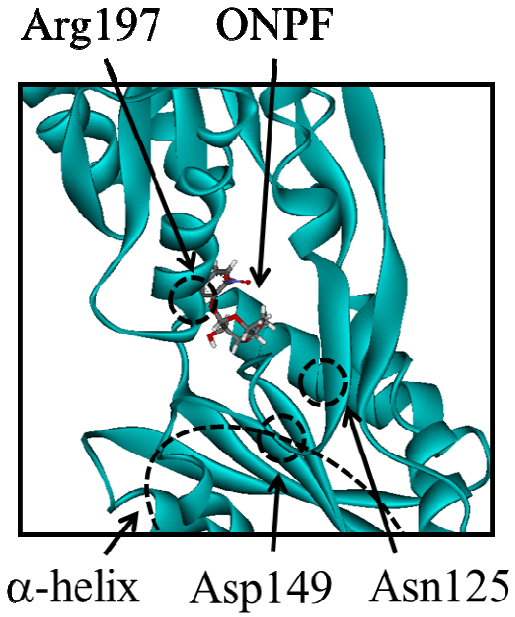


Fig. 52 The detailed structure around ONPF in the LacR-ONPF + DNA complex.

4.2.4. リガンド結合による転写制御機構のモデル

これらの結果より、IPTG と ONPF が LacR の転写制御機構に与える影響を考察する。

LacR は熱の揺らぎの影響を受けながら、DNA に結合している。これは、LacR の regulatory domain が常にある程度、DNA 上で揺れていることを示す。ここに IPTG が結合すると、IPTG のイソプロピル基とリガンド結合に重要なアミノ酸残基である極性アミノ酸 Asn125、荷電アミノ酸 Asp149 が反発する。これらのアミノ酸残基が含まれるα-helix はリガンド結合ポケットから外に押し出され、近くに存在する LacR のもう一方の単量体の DNA-binding domain と水素結合や静電相互作用により結合する。この結合により、もう一方の単量体の DNA-binding domain が regulatory domain に引き寄せられ、二量体で結合していた LacR と DNA の結合が単量体のみの結合となり、不安定化、やがて LacR が DNA から分離すると考えられる。

一方、LacR に ONPF が結合した場合、ONPF のフコースと Asn125、Asp149 は引力相互作用するため、Asn125、Asp149 を含むα-helix がリガンド結合ポケットに引き寄せられる。これにより、α-helix ともう一方の単量体の DNA-binding domain が相互作用しなくなるため、LacR が DNA から分離しないと考えられる。

5. 結論

本研究は、古典分子力学 (MM) 法と古典分子動力学 (MD) 法により、水和したリガンドが結合していない LacR 単量体+DNA 複合体、インデューサ IPTG が結合している LacR 単量体+DNA 複合体、アンチインデューサ ONPF が結合している LacR 単量体+DNA 複合体の構造を得た。これらの構造に対し、フラグメント分子軌道 (FMO) を用い、LacR とリガンド間の結合に LacR の Asn125、Asp149、Arg197 が重要であることを明らかにした。また、LacR と DNA 間の結合に Arg22 が重要であることを明らかにした。さらに、MM 法で得られた構造は LacR、リガンド、DNA 間の相互作用の解析において不十分であり、より安定した構造を解析するために MD 法を利用しなければならないことも明らかにした。また、LacR 単量体+DNA 複合体では、LacR がリガンドの影響により、DNA から分離する過程をうまく説明できないことを明らかにした。

これを踏まえ、LacR 二量体+DNA 複合体、LacR 二量体-IPTG+DNA 複合体、LacR 二量体-ONPF+DNA 複合体に対し、水中で 30 ns の MD 計算をし、その構造を解析した。これにより、リガンドの影響を受けた構造を入手できた。そして、構造解析より、IPTG が結合することにより、IPTG の周囲に存在する親水性のアミノ酸残基を含んだα-helix と IPTG のイソプロピル基が反発することを明らかにした。これをトリガーにして、α-helix がもう一方の単量体の DNA-binding domain と相互作用することにより、DNA と二量体間の結合が DNA と単量体間の結合になり、最終的に DNA から分離するモデルを提案した。一方、ONPF が結合した場合は、ONPF のフコースと親水性のアミノ酸残基を含んだα-helix は引力相互作用をすることを明らかした。これにより、α-helix がリガンド結合ポケット側に引き寄せられ、もう一方の単量体の DNA-binding domain と相互作用しないため、ONPF と結合した LacR が DNA から分離しないモデルを提案した。

6. 謝辞

本研究、本論文を審査して戴き、また別の視点から研究に対しご助言して戴きました豊橋技術科学大学　関野秀男教授、後藤仁志准教に深く感謝致します。

また、本研究は豊橋技術科学大学　栗田典之准教授の御指導の下、行われました。栗田典之准教授には、研究に関する御指導、御助言の他、海外派遣や研究助成金支援書類の作成に関しての御助言を戴きました。また、研究以外の面でも今後の進路の御助言等、お世話になりました。栗田典之准教授には心から感謝申し上げます。

さらに、豊橋技術科学大学　栗田研究室卒業生　出立兼一博士、塚本貴志博士、早川雅人氏には、研究の他、研究生活においても様々な御助言をして戴きました。栗田研究室　宮城慧氏には様々な研究の手法、解析手法、研究生活のスタイルを紹介して戴きました。同じく栗田研究室　岡本晃澄氏には鋭い観点から研究の問題点を指摘して戴きました。その他栗田研究室の皆様にも研究環境、解析プログラムの提供など様々な形でご協力を戴きました。皆様には深く感謝致します。

最後に、私をこれまで支えてくださった両親、及び友人に深く感謝の意を表します。

7. 参考文献

[1] F. Jacob, J. Monod, *J. Mol. Biol.* **3**, 318-356 (1961).

[2] C. E. Bell, M. Lewis, *Nat. Struct. Biol.* **7**, 209-214 (2000).

[3] C. E. Bell, M. Lewis, *Nat. Struct. Biol.* **7**, 209-214 (2004).

[4] R. Daber, S. Stayrook, A. Rosenberg, M. Lewis, *J. Mol. Biol.* **370**, 609-619 (2007).

[5] N. Kurita, M. Matsuoka, Y. Sengoku, *J. Theo. & Comp. Chem.* **5**, 59-74 (2006).

[6] M. Matsuoka, Y. Sengoku, N. Kurita, *J. Comp. Aided. Chem.* **4**, 35-41 (2003).

[7] S. Nishikawa, S. Kozakai, Y. Sengoku, N. Kurita, *J. Comp. Aided. Chem.* **9**, 17-29 (2008).

[8] K. Kitaura, T. Sawai, T. Asada, T. Nakano, M. Uebayasi, *J. Chem. Phys. Lett.* **312**, 319-324 (1999).

[9] T. Ohyama, M. Hayakawa, S. Nishikawa, N. Kurita, *J. Comp. Chem.* **32**, 1661-1670 (2011).

[10] V. P. Chuprina, J. A. Rullmann, R. M. Lamerichs, J. H. van Boom, R. Boelens, R. Kaptein, *J. Mol. Biol.* **234**, 446-462 (1993).

[11] A. M. Friedman, T. O. Tischmann, T. A. Steitz, *Science* **268**, 1721-1727 (1995).

[12] M. Lewis, G. Chang, N.C. Horton, M. A. Kercher, H. C. Pace, M. A. Schumacher, R. G. Brennan, P. Lu, *Science* **271**, 1247-1254 (1996).

[13] B. Müller-Hill, L. Crapo, W. Gilbert, *Proc. Natl. Acad. Sci. USA* **59**, 1259-1264 (1968).

[14] Gaussian 03, Revision C.02, M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, 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, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, J. A. Pople, Gaussian, Inc., Wallingford CT (2004).

[15] C. Møller, M.S. Plesset, Phys. Rev., 46, 618-622 (1934).

[16] C. I. Bayly, P. Cieplak, W. D. Cornell, P. A. Kollman, *J. Phys. Chem.* **97**, 10269-10280 (1993).

[17] W. D. Cornell, P. Cieplak, C. I. Bayly, P. A. Kollman, *J. Am. Chem. Soc.* **115**, 9620-9631 (1993).

[18] T. Fox, P. A. Kollman, *J. Phys. Chem. B* **102**, 8070-8079 (1998).

[19] P. Cieplak, W. D. Cornell, C.I. Bayly, P. A. Kollman, *J. Comp. Chem.* **16**, 1357-1377 (1995).

[20] HyperChem(TM) Professional 7.51, Hypercube, Inc., 1115 NW 4th Street, Gainesville, Florida 32601, USA.

[21] D. A. Case, T. A. Darden, T. E. Cheatham III, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, K. M. Merz, D. A. Pearlman, M. Crowley, R. C. Walker, W. Zhang, B. Wang, S. Hayik, A. Roitberg, G. Seabra, K. F. Wong, F. Paesani, X. Wu, S. Brozell, V. Tsui, H. Gohlke, L. Yang, C. Tan, J. Mongan, V. Hornak, G. Cui, P. Beroza, D. H. Mathews, C. Schafmeister, W. S. Ross, P. A. Kollman, AMBER 9, University of California, San Francisco (2006).

[22] D. A. Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T.E. Cheatham III, S. DeBolt, D. Ferguson, G. Seibel, P. Kollman. *Comp. Phys. Commun.* **91**, 1-41 (1995).

[23] D. A. Case, T. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, Jr., A. Onufriev, C. Simmerling, B. Wang and R. Woods. *J. Comp. Chem.* **26**, 1668-1688 (2005).

[24] K. Kitaura, T. Sawai, T. Asada, T. Nakano, M. Uebayasi, *Chem. Phys. Lett.* **312**, 319-324 (1999).

[25] Y. Mochizuki, K. Yamashita, K. Fukuzawa, K. Takematsu, H. Watanabe, N. Taguchi, Y. Okiyama, M. Tsuboi, T. Nakano, S. Tanaka, *Chem. Phys. Lett.* **493**, 346-352 (2010).

[26] K. Kitaura, E. Ikeo, T. Asada, T. Nakano, M. Uebayasi, *Chem. Phys. Lett.* **313**, 701-706 (1999).

[27] K. Nomura, A. Okamoto, A. Yano, S. Higai, T. Kondo, S. Kamba, N. Kurita, *Chem. Phys. Lett.* **547**, 89-96 (2012).

[28] Y. Mochizuki, S. Koikegami, S. Amari, K. Segawa, K. Kitaura, T. Nakano, *Chem. Phys. Lett.* **406**, 383-288 (2005).

[29] N. Lehming, J. Sartorius, B. Kisters-Woike, B. von Wilcken-Bergmann, B. Müller-Hill, *EMBO J.* **9**, 615-621 (1990).

[30] V. P. Chuprina, J. A. C. Rullmann, R. Lamerichs, J. H. van Boom, R. Boelens, R. Kaptein, *J. Mol. Biol.* **234**, 446-462 (1993).

[31] J. Romanuka, G. E. Folkers, N. Biris, E. Tishchenko, H. Wienk, A. M. J. J. Bonvin, R. Kaptein, R. Boelens, *J. Mol. Biol.* **390**, 478-489 (2009).

[32] H. Li, A. D. Robertson, J. H. Jensen, *Proteins* **61**, 704-721 (2005).

[33] D. C. Bas, D. M. Rogers, J. H. Jensen, *Proteins* **73**, 765-783 (2008).

[34] M. H. M. Olsson, C. R. Søndergard, M. Rostkowski, J. H. Jensen, *J. Chem. Theo. & Comp.* **7**, 525-537 (2011).

[35] C. R. Søndergaard, M. H. M. Olsson, M. Rostkowski, J. H. Jensen, *J. Chem. Theo. & Comp.* **7**, 2284-2295 (2011).

[36] D. A. Case, T. A. Darden, T. E. Cheatham III, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, R. C. Walker, W. Zhang, K. M. Merz, B. Roberts, S. Hayik, A. Roitberg, G. Seabra, J. Swails, A. W. Goetz, I. Kolossváry, K. F. Wong, F. Paesani, J. Vanicek, R. M. Wolf, J. Liu, X. Wu, S. R. Brozell, T. Steinbrecher, H. Gohlke, Q. Cai, X. Ye, J. Wang, M. -J. Hsieh, G.

Cui, D. R. Roe, D. H. Mathews, M. G. Seetin, R. Salomon-Ferrer, C. Sagui, V. Babin, T. Luchko, S. Gusarov, A. Kovalenko, and P. A. Kollman, AMBER 12, University of California, San Francisco (2012).

[37] H. J. C. Berendsen, D. van der Spoel, R. van Drunen, *Comp. Phys. Comm.* **91**, 43-56 (1995).

[38] E. Lindahl, B. Hess, D. van der Spoel., *J. Mol. Model.* **7**, 306-317 (2001).

[39] D. vand der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, H. J. C. Berendsen, *J. Comp. Chem.* **26**, 1701-1718 (2005).

[40] B. Hess, C. Kutzner, D. van der Spoel, E. Hindahl, *J. Chem. Theo. Comp.* **4**, 435-447 (2008).

[41] K. Lindorff-Larsen, S. Piana, K. Palmo, P. Maragakis, J. L. Klepeis, R. O. Dror, D. E. Shaw, *Proteins* **78**, 1950-1958 (2010).

[42] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, M. L. Klein, *J. Chem. Phys.* **79**, 926-936 (1983).

[43] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, D. A. Case, *J. Comp. Chem.* **25**, 1157-1174 (2004).

[44] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, J. A. Montgomery, *J. Comp. Chem.* **14**, 1347-1363 (1993).

[45] A. W. S. da Silva, W. F. Vranken, *BMC Res. Notes* **5**, 367-374 (2012).

[46] T. G. Gantchev, D. J. Hunting, *J. Mol. Model.* **14**, 451-464 (2008).

[47] R. W. Hockney, S. P. Goel, J. W. Eastwood, *J. Comp. Phys.* **14**, 148-158 (1974).

[48] T. Darden, D. York, L. Pedersen, *J. Chem. Phys.* **98**, 10089-10092 (1993).

[49] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, L. G. Pedersen, *J. Chem. Phys.* **103**, 8577-8592 (1995).

[50] S. Miyamoto, P. A. Kollman, *J. Comp. Chem.* **13**, 952-962 (1992).

[51] B. Hess, H. Bekker, H. J. C. Berendsen, J. G. E. M. Fraaije, *J. Comp. Chem.* **18**, 1463-1472 (1997).

[52] G. Bussi, D. Donadio, M. Parrinello, *J. Chem. Phys.* **126**, 014101 (2007).

[53] S. Nosé, *Mol. Phys.* **52**, 255-268 (1984).

[54] W. G. Hoover, *Phys. Rev. A* **31**, 1695-1697 (1985).

[55] M. Parrinello, A. Rahman, *J. Appl. Phys.* **52**, 7182-7190 (1981).

[56] S. Nosé, M. L. Klein, *Mol. Phys.* **50**, 1055-1076 (1983).

[57] M. Lewis, *C. R. Biol.* **328**, 521-548 (2005).

[58] K. Fukuzawa, Y. Komeiji, Y. Mochizuki, A. Kato, T. Nakano, S. Tanaka, *J. Comp. Chem.* **27**, 948-960 (2005).

8. 附録

附録 A: 本研究で作成したプログラム

本研究では、Protein Data Bank 形式の構造データを様々なプログラムに与え、LacR とリガンド、DNA 間の特異的相互作用を解析した。その際、プログラムに合わせ、構造や記述を書き換える必要があった。そこで、PDB を編集するプログラムをいくつか作成した。

なお、PDB 形式のファイルは、PDB を構成する原子のシリアル番号、原子タイプ、Chain 名、残基名、残基番号、原子の座標、原子同士の接続情報が 1 原子あたり 1 行で記述されている。以下に、最小構成の PDB を示す。

```
ATOM      1  C1  NPF A 332     104.060  81.860  61.830
ATOM      2  C2  NPF A 332     102.670  81.770  61.690
ATOM      3  C3  NPF A 332     102.020  82.410  60.640
ATOM      4  C4  NPF A 332     102.770  83.130  59.710
ATOM      5  C5  NPF A 332     104.160  83.180  59.820
  :
  :
CONECT    1    2    6   10
CONECT    2    1    3   21
CONECT    3    2    4   22
CONECT    4    3    5   23
CONECT    5    4    6   24
  :
  :
END
```

PDB は本来、Fortran で利用することを目的としていたため、行中でも記述すべき位置が決まっている。1～6 文字は Record name というその行の情報がどのような情報であるかを示す記述子を記述する。上記の例では、ATOM、CONECT、END がそれに該当し、END は系の終わりを示している。

ATOM 記述子は、それぞれの原子の情報を示す行を示している。この行の 7～11 文字は原子のシリアル番号が記述される。このシリアル番号に割り当てられている文字数は 5 であり、5 桁までのシリアル番号しか記述できない。従って、99999 を超える原子数が記述されている場合、その解釈は読み込むプログラムに依存する。これは場合によっては誤動作の原因となるため、原子数が 99999 を超えていないか確認する必要がある。13～16 文字は原子タイプを記述する。原子タイプは同じ分子の中でユニークに決定される

名前であり、基本的に原子記号が 1 文字の場合は、14 文字目にその原子記号を、原子記号が 2 文字の場合は 13、14 文字目にその原子記号が記述される。さらに、タンパク質、DNA、RNA のような一般的な分子の場合、全ての原子につけられる原子タイプは決まっている。18～20 文字は残基名が記述される。これもタンパク質、DNA、RNA を構成する原子の残基名は決まっている。27 文字は Chain ID を記述する。Chain ID はタンパク質のペプチド鎖に付けられた任意の固有名であり、複数のタンパク質が 1 つの PDB ファイルに記述されている場合、それぞれのタンパク質を識別する際に役に立つ。23～26 文字は残基番号が記述される。タンパク質、DNA、RNA の場合、この残基番号と残基名により、分子や残基の区切りをプログラムが認識することができる。そして、31～38、39～46、47～54 文字は、それぞれ小数点 3 桁の原子の X、Y、Z 座標を示している。

CONECT 記述子は、それぞれの原子がどのように結合・接続しているかを示す行を示している。この行の 7～11 文字は ATOM 記述子の 7～11 文字に対応するシリアル番号が記述され、その行の接続情報がどの原子のものであるかを示している。そして、12～16、17～21、22～26、27～31 文字は 7～11 文字で示されている原子に結合している原子のシリアル番号が記述される。上記の例で説明すると、CONECT 記述子の 1 行目の原子は、2、6、10 の原子と結合していることを示している。

附録 A-1: PDB 編集プログラム

PDB は単なるテキストファイルであるため、テキストエディタで編集することが可能である。しかし、多くのファイルに対し、正確に編集していくことは困難である。そこで、本研究では PDB を編集する CLI プログラムを作成した。本プログラムは、プログラム名 pinch、rashid、uni から成る。これらを組み合わせることにより、大部分の構造を作成することができる。

Pinch は、元の PDB ファイルからシリアル番号、原子タイプ、残基番号、残基名と残基番号、Chain ID のいずれかを指定して任意の構造を抽出するプログラムである。使い方としては、まず、PDB の状態を以下のコマンドで確認する。今回の例では、入力ファイルとして input.pdb を与える。

```
$ pinch --view 3 input.pdb
```

第一引数として、--view、あるいは-V を与え、第二引数には表示する列数を与える。そして、第三引数に入力ファイルを与える。これにより、以下の結果を得ることができる。

```
   1 -    6:ACE C   1 |    7 -   28:LYS C   2 |   29 -   42:PRO C   3
  43 -   58:VAL C   4 |   59 -   72:THR C   5 |   73 -   91:LEU C   6
  92 -  112:TYR C   7 |  113 -  124:ASP C   8 |  125 -  140:VAL C   9
 141 -  150:ALA C  10 |  151 -  165:GLU C  11 |  166 -  186:TYR C  12
 187 -  196:ALA C  13 |  197 -  203:GLY C  14 |  204 -  219:VAL C  15
 220 -  230:SER C  16 |  231 -  251:TYR C  17 |  252 -  268:GLN C  18
 269 -  282:THR C  19 |  283 -  298:VAL C  20 |  299 -  309:SER C  21
```

これは、1 行目を例に取ると、Chain C の ACE 1 の残基がシリアル番号 1～6、Chain C の Lys 2 残基がシリアル 7～28、Chain C の Pro 3 がシリアル番号 29～42 で構成されていることを示している。今回は第二引数に 3 を与えているため、1 行あたり 3 つの残基を示しているが、引数を変更することにより小さい端末でも表示できるようになっている。

次に、先ほどの結果から抽出する対象を見つけ、実際に抽出する。抽出対象を指定する方法は、前述の通り、いくつかの方法がある。それを以下に示す。

method（第一引数）		対象（第二引数）	機能
ロングオプション	ショートオプション		
--atomnumber	-AN	シリアル番号	シリアル番号を指定して抽出
--residuenumber	-AR	残基番号	残基番号を指定して抽出
--residue	-R	残基名	残基名を指定して抽出
--detailatom	-DA	原子タイプ	原子タイプを指定して抽出
--detailresidue	-DR	残基名.残基番号	残基名と残基番号を両方指定して抽出
--unit	-U	Chain ID	Chain IDを指定して抽出

ロングオプションとショートオプションはいずれかを指定する。--atomnumber、--residuenumber は開始番号と終了番号をハイフンで繋いだ範囲指定が可能である。また、全てにおいて対象を「,(コンマ)」で繋ぐことにより、複数の対象を選択することができる。以下にその実行例を示す。ここで、入力ファイルを input.pdb、出力ファイルを output.pdb とする。なお、出力ファイルは省略することができる。その場合、対話的にどのようにするかを尋ねられる。

```
$ pinch --atomnumber 50-60 input.pdb output.pdb
```

```
$ pinch --atomnumber 50-63,100-125 input.pdb output.pdb
```

```
$ pinch --residuenumber 32,35,38 input.pdb output.pdb
```

```
$ pinch --residue GLU,LYS input.pdb output.pdb
```

```
$ pinch --detailatom N,CA,C,O input.pdb output.pdb
```

```
$ pinch --detailresidue PRO.3,VAL.4 input.pdb output.pdb
```

```
$ pinch --unit C input.pdb
```

以下に pinch のソースコード　(プログラム言語: Perl) を示す。

```
#! /usr/bin/env perl

use strict;
use File::Temp;

my $option = shift(@ARGV);
my $apo = shift(@ARGV);
my $in = shift(@ARGV);
my $out = shift(@ARGV);
my $tmp1 = File::Temp->new(TEMPLATE => '.pinch-1XXXXXX');
my $tmp2 = File::Temp->new(TEMPLATE => '.pinch-2XXXXXX');
$SIG{'TERM'} = $SIG{'PIPE'} = $SIG{'HUP'} = $SIG{'INT'} = sub {
        unlink $tmp1;
        unlink $tmp2;
        exit;
};      # プログラムが中断した場合の処理

if(($option =~ /^--help$/i) || ($option =~ /^-h$/i)){
        &help;
        exit;
}

if(($option =~ /^--view$/i) || ($option =~ /^-V$/i)){
        if($apo =~ /^¥d+$/){
                &file($in);
                &view($apo, $in);
        }
        else{
                print " ERROR: 表示列数を指定してください. ¥n";
        }
        exit;
}
elsif(($option =~ /^--information$/i) || ($option =~ /^-I$/i)){
        &file($apo);
        &info($apo);
        exit;
}
elsif(($option =~ /^--atomnumber$/i) || ($option =~ /^-AN$/i)){
        &file($in);
        &atomorder($apo, $in, $tmp1);
}
elsif(($option =~ /^--residuenumber$/i) || ($option =~ /^-RN$/i)){
        &file($in);
        &residueorder($apo, $in, $tmp1);
}
elsif(($option =~ /^--residue$/i) || ($option =~ /^-R$/i)){
        &file($in);
        &residue($apo, $in, $tmp1);
}
elsif(($option =~ /^--unit$/i) || ($option =~ /^-U$/i)){
        if($apo =~ /^none$/i){
                $apo = " ";
        }
        &file($in);
        &unit($apo, $in, $tmp1);
}
elsif(($option =~ /^--detailresidue$/i) || ($option =~ /^-DR$/i)){
        &file($in);
        my @check = split(/,/, $apo);
        my $checkflag = 0;
        foreach(@check){
                if($_ !~ /^.+¥..+$/){
                        $checkflag = 1;
                        last;
                }
        }
        undef @check;
        if($checkflag == 1){
                print " ERROR: 指定残基の表現が違います. [残基名.残基番号]で指定する. ¥n";
                exit;
```

```
        }
        else{
                &detailresidue($apo, $in, $tmp1);
        }
}
elsif(($option =~ /^--detailatom$/i) || ($option =~ /^-DA$/i)){
        &file($in);
        &detailatom($apo, $in, $tmp1);
}
else{
        print " ERROR: 未定義オプションです. ¥n";
        exit;
}
&terdel($tmp1, $tmp2);
&crecone($tmp2, $tmp1, 1.9);
if($out =~ /^$/){
        print " INFORMATION: 出力先が指定されていません. ¥n";
        print " INFORMATION: 上書き(O), 別名で保存(S), 何もしない(C)¥n > ";
        my $user = <STDIN>;
        $user =~ s/¥n//;
        if($user =~ /^O$/i){
                unlink $in;
                rename $tmp1, $in;
        }
        elsif($user =~ /^S$/i){
                print " ファイル名: ";
                $user = <STDIN>;
                $user =~ s/¥n//;
                rename $tmp1, $user;
        }
        else{
                unlink $tmp1;
                exit;
        }
}
else{
        rename $tmp1, $out;
}
exit;

# =============== view =============== #
## VIEWモード
sub view{
my $column = shift(@_);
my $in = shift(@_);

my $residue1 = "";
my $residue2 = "";
my $atom1 = "";
my $atom2 = "";
my $start = 0;
my $ncolumn = 0;
my $lastatom = 0;
open(IN, "$in");
while(<IN>){
        if((/^atom/i) || (/^hetatm/i)){
                my $before = $lastatom;
                $lastatom = substr($_, 6, 5);
                $residue1 = substr($_, 17, 9);
                if($residue1 eq $residue2){
                        next;
                }
                elsif($start == 0){
                        $residue2 = substr($_, 17, 9);
                        $atom1 = substr($_, 6, 5);
                        $atom1 =~ s/¥s//g;
                        $start = 1;
                        print " ";
                }
                else{
                        $atom2 = $before;
                        $atom1 = &adjust(5, $atom1);
                        $atom2 = &adjust(5, $atom2);
                        print "$atom1 - $atom2:$residue2";
                        $atom1 = $lastatom;
                        $residue2 = substr($ , 17, 9);
```

```
                        $ncolumn++;
                        if($ncolumn >= $column){
                                print "¥n ";
                                $ncolumn = 0;
                        }
                        else{
                                print " | ";
                        }
                }
        }
}
close(IN);

$atom2 = $lastatom;
$atom1 = &adjust(5, $atom1);
$atom2 = &adjust(5, $atom2);
print "$atom1 - $atom2:$residue2¥n";
}

# =============== info =============== #
## 系の情報を調べる
sub info{
my $in = shift(@_);

my %charge_residues = (#  電荷情報
        "LYS" => "1",
        "ARG" => "1",
        "ASP" => "-1",
        "GLU" => "-1",
        "HIP" => "1",
        "CYM" => "1",
        "Na" => "1",    # Na+
        "K" => "1",             # K+
        "Zn" => "2",    # Zn+
        "ZIN" => "2",   # Zn+
        "Ca" => "2",    # Ca+
        "CAL" => "2",   # Ca+
        "DA" => "-1",
        "DG" => "-1",
        "DC" => "-1",
        "DT" => "-1",
        "DA3" => "-1",
        "DG3" => "-1",
        "DC3" => "-1",
        "DT3" => "-1",
        "RA" => "-1",
        "RG" => "-1",
        "RC" => "-1",
        "RU" => "-1",
        "RA3" => "-1",
        "RG3" => "-1",
        "RC3" => "-1",
        "RU3" => "-1"
);
my @noncharge_residues = (
        "ACE", "ALA", "ASN", "CYS", "GLN", "GLY", "HIS", "HID", "HIE", "ILE",
        "LEU", "MET", "NME", "PHE", "PRO", "SER", "THR", "TRP", "TYR", "VAL",
        "SOL", "HOH", "WAT", "DT5", "DA5", "DC5", "DG5", "RU5", "RA5", "RC5", "RG5"
);

my $resnum = 0;
my $atomnum = 0;
my $charge = 0;
my $water = 0;
my $residue1 = "";
my $start = 0;
my @unknowns = ();
my @hiss = (0, 0, 0, 0);        # His の状態(HIS, HID, HIE, HIP)
open(IN, "$in");
while(<IN>){
        if((/^HETATM/i) || (/^ATOM/)){
                $atomnum ++;    #  原子数+1
                my $residue2 = substr($_, 17, 9);       #  現在の残基情報
                if($residue1 ne $residue2){
                #  前の残基情報と一致しない場合(残基が変わった場合)
                        $resnum ++;     #  残基数+1
```

```
                        if($residue2 =~ /^HI(S|D|E|P)/i){
                                if($residue2 =~ /^HIS/i){
                                        $hiss[0] ++;
                                }
                                elsif($residue2 =~ /^HID/i){
                                        $hiss[1] ++;
                                }
                                elsif($residue2 =~ /^HIE/i){
                                        $hiss[2] ++;
                                }
                                elsif($residue2 =~ /^HIP/i){
                                        $hiss[3] ++;
                                }
                                else{
                                        print " WARNING: Unknown His exists¥n";
                                }
                        }
                        if(($residue2 =~ /^WAT/i) || ($residue2 =~ /^HOH/i) ||
                        ($residue2 =~ /^SOL/i)){
                        #  残基名が WAT あるいは HOH の場合
                                $water ++;      #  水分子数+1
                        }
                        if($start == 0){        #  最初の残基の異なりは無効
                                $start = 1;     #  フラグを通常に戻す
                                $residue1 = $residue2;
                                #  現在の残基情報を前の残基情報として登録
                                next;
                        }
                        else{   #  通常状態
                                $residue1 = substr($residue1, 0, 3);
                                #  残基名のみ取得
                                $residue1 =~ s/¥s+//g;
                                $residue1 =~ s/[¥+¥*¥/]//g;
                                #  + や *, /は _ に変換
                                my @data = keys(%charge_residues);
                                #  電荷情報からキーを配列に格納
                                my $charge_flag = 0;
                                foreach(@data){
                                        if($residue1 =~ /^$_$/i){
                                                #  キーが一致した場合
                                                $charge += $charge_residues{$_};
                                                #  キーの値(電荷)を現在の電荷に加算
                                                $charge_flag = 1;
                                                last;
                                        }
                                }
                                if(($charge_flag == 0) &&
                                (grep(/^$residue1$/, @noncharge_residues) == 0)){
                                        push(@unknowns, $residue1);
                                }
                                $residue1 = $residue2;
                                #  現在の残基を前の残基として登録
                        }
                }
        }
}
close(IN);

$residue1 = substr($residue1, 0, 3);
#  ループできなかった情報の処理      #  残基名のみ取得
$residue1 =~ s/[¥+¥*¥/]/_/g;    #  + や *, /は _ に変換
my @data = keys(%charge_residues);      #  電荷情報からキーを配列に格納
my $charge_flag = 0;
foreach(@data){
        if($residue1 =~ /^$_$/i){       #  キーが一致した場合
                $charge += $charge_residues{$_};
                #  キーの値(電荷)を現在の電荷に加算
                $charge_flag = 1;
                last;
        }
}
if(($charge_flag == 0) && (grep(/^$residue1$/, @noncharge_residues) == 0)){
        push(@unknowns, $residue1);
}

my $unknown_residue = join(" + ", @unknowns);
```

```
if($#unknowns != -1){
        $unknown_residue = " + " . $unknown_residue;
}

my $notwater = $resnum - $water;
print << "INFO";
 INFORMATION for $in
   All of Atom         : $atomnum
   All of Residue      : $resnum
    * Not water molecule: $notwater
    * Water molecule    : $water
   System charge       : $charge $unknown_residue

   HIS : HID : HIE: HIP = $hiss[0] : $hiss[1] : $hiss[2] : $hiss[3]

INFO
}

# =============== atomorder =============== #
## 原子順序番号指定モード
sub atomorder{
my $atomrange = shift(@_);      # 指定された原子範囲
my $in = shift(@_);             # 入力ファイル
my $out = shift(@_);            # 出力ファイル

my $infinity = 0;       # 無限判定（以降全て）
my @atomarray = &range($atomrange);     # 原子順序番号範囲を全て配列に

my $check = pop(@atomarray);    # 配列末端を$check に
if($check == -1){       # 末端が-1 なら無限である
        $infinity = 1;
        $check = $atomarray[0];
}
else{
        push(@atomarray, $check);       # 通常ならば，判定に使った末端情報を元に戻す
        @atomarray = sort {$a <=> $b} (@atomarray);     # ソート
        $check = 0;     # 以降の無限判定にひっかからないように 0
}

my $write = 0;                  # 無限用書き込み判定
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        my $line = $_;  # バックアップ
        if((/^ATOM/) || (/^HETATM/)){
                my $nowatom = substr($line, 6, 5);      # 原子順序番号を取得
                $nowatom =~ s/¥s+//;    # スペースを消して完全に数字扱い
                if(($check != 0) && ($check <= $nowatom)){
                        # 無限の開始位置のチェック
                        print OUT $line;
                }
                else{
                        my @newatomarray = ();  # 新しい原子順序番号の配列
                        foreach(@atomarray){
                                if($_ == $nowatom){     # 一致したら書き込み
                                        print OUT $line;
                                }
                                else{
                                        # 外れは次の検知に備えて，配列のスリム化
                                        push(@newatomarray, $_);
                                }
                        }
                        @atomarray = @newatomarray;
                }
        }
        elsif(/^TER/){
                print OUT "TER¥n";
        }
        elsif(/^END/){
                print OUT "END¥n";
        }
}
close(IN);
close(OUT);
}
```

```
# =============== residueorder =============== #
## 残基順序番号指定モード
sub residueorder{
my $resrange = shift(@_);       # 指定された残基番号範囲
my $in = shift(@_);             # 入力ファイル
my $out = shift(@_);            # 出力ファイル

my @resarray = &range($resrange);       # 残基順序番号範囲を全て配列に

my $check = pop(@resarray);     # 配列末端を$checkに
if($check == -1){       # 末端が-1なら無限である
        $check = $resarray[0];
        $check = pop(@resarray);
}
else{
        push(@resarray, $check);        # 通常ならば，判定に使った末端情報を元に戻す
        @resarray = sort {$a <=> $b} (@resarray);       # ソート
        $check = 0;
}

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        my $line = $_;  # バックアップ
        if((/^ATOM/) || (/^HETATM/)){
                my $nowres = substr($line, 22, 4);      # 残基順序番号を取得
                $nowres =~ s/¥s+//g;    # スペースを消して完全に数字扱い
                if(($check != 0) && ($check <= $nowres)){
                        print OUT $line;
                }
                else{
                        foreach(@resarray){
                                if($_ == $nowres){      # 一致したら書き込み
                                        print OUT $line;
                                }
                        }
                }
        }
        elsif(/^TER/){
                print OUT "TER¥n";
        }
        elsif(/^END/){
                print OUT "END¥n";
        }
}
close(IN);
close(OUT);
}

# =============== residue =============== #
## 残基指定モード
sub residue{
my $residue = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @residues = split(/,/, $residue);

my $first = 0;          # TER判定による初期スキップフラグ
my $beforeres = "";
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        my $line = $_;  # 現在行のバックアップ
        if((/^ATOM/) || (/^HETATM/)){
                my $nowdata = substr($_, 17, 9);                # 全ての残基情報
                my $nowres = substr($nowdata, 0, 3);    # 残基名取得
                foreach(@residues){
                        if($nowres eq $_){      # 残基名が一致したら書き込み
                                if($beforeres ne $nowdata){
                                        if($first == 1){
                                                print OUT "TER¥n";
                                        }
                                        else{
                                                $first = 1;
                                        }
```

```
                                }
                                print OUT $line;
                                $beforeres = $nowdata;
                                last;
                        }
                }
        }
        elsif((/^TER/) && ($first == 1)){
                # 以前に書き込みが有り，かつ TER なら TER を
                print OUT "TER¥n";
        }
        elsif(/^END/){
                print OUT "END¥n";
        }
}
close(IN);
close(OUT);
}

# =============== unit =============== #
## ユニット指定モード
sub unit{
my $unit = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @units = split(/,/, $unit);          # ユニットを分割

my $first = 0;  # 書き込みに関するフラグ
my $beforeunit = "";
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        my $line = $_;  # バックアップ
        if((/^ATOM/) || (/^HETATM/)){
                my $nowunit = substr($_, 21, 1);        # 現在のユニット取得
                foreach(@units){
                        if($nowunit eq $_){
                                # 指定ユニットと一致したら書き込む
                                if($nowunit ne $beforeunit){
                                        # 前に書き込んだユニットと違った場合
                                        $beforeunit = $nowunit;
                                        if($first == 1){
                                                # TER を記述
                                                print OUT "TER¥n";
                                        }
                                        else{           # 初回はスキップ
                                                $first = 1;
                                        }
                                }
                                print OUT $line;
                        }
                }
        }
        elsif((/^TER/) && ($first == 1)){       # 指定ユニット内の TER は書き込む
                print OUT "TER¥n";
        }
        elsif(/^END/){
                print OUT "END¥n";
        }
}
close(IN);
close(OUT);
}

# =============== detailresidue =============== #
## 詳細残基指定モード
sub detailresidue{
my $apoint = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @res = split(/,/, $apoint);

my $first = 0;
my $beforeres = "";
```

```
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        my $line = $_;
        if((/^ATOM/) || (/^HETATM/)){
                my $nowdata = substr($line, 17, 9);     # 全体の残基情報
                my $nowres = substr($nowdata, 0, 3);    # 残基名取得
                $nowres =~ s/¥s//g;
                my $nowresnum = substr($nowdata, 5, 4);         # 残基番号取得
                $nowresnum =~ s/¥s//g;
                foreach(@res){
                        my $residue = $_;
                        ($residue, my $resnum) = split(/¥./, $residue);
                        if(($nowres eq $residue) && ($nowresnum == $resnum)){
                                if($beforeres ne $nowdata){
                                        if($first == 1){
                                                print OUT "TER¥n";
                                        }
                                        else{
                                                $first = 1;
                                        }
                                }
                                print OUT $line;
                                $beforeres = $nowdata;
                                last;
                        }
                }
        }
        elsif(/^TER/){
                print OUT "TER¥n";
        }
        elsif(/^END/){
                print OUT "END¥n";
        }
}
close(IN);
close(OUT);
}

# =============== detailatom =============== #
## 詳細原子指定モード
sub detailatom{
my $atom = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @atomarray = split(/,/, $atom);

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^ATOM/i) || (/^HETATM/i)){
                my $line = $_;
                my $nowatom = substr($line, 12, 4);     # 詳細原子名取得
                foreach(@atomarray){
                        my $checkatom = $_;
                        if($nowatom =~ /$checkatom/){   # 照らし合わせ
                                print OUT $line;
                                print OUT "TER¥n";
                                last;
                        }
                }
        }
        elsif(/^TER/){
                print OUT "TER¥n";
        }
        elsif(/^END/){
                print OUT "END¥n";
        }
}
close(IN);
close(OUT);
}

# =============== terdel =============== #
## TER の重複及び重複行の削除
```

```
sub terdel{
my $in = shift(@_);
my $out = shift(@_);

open(IN, "$in");
open(OUT, "> $out");
my $line1 = ""; # 現在行
my $line2 = ""; # 前の行
my $flag = 0;
while(<IN>){
        if(($flag == 0) && ((/^ATOM/i) || (/^HETATM/i))){
                $flag = 1;
        }
        if(($flag == 0) && (/^TER/i)){
                next;
        }
        $line1 = $_;    # 現在行取得
        $line1 =~ s/¥n//;
        $line1 =~ s/¥s+//g;
        if($line1 eq $line2){   # 前の行と同じならスキップ
                next;
        }
        $line2 = $line1;        # 引き継ぎ
        print OUT;
}
close(IN);
close(OUT);
}

# ============== range ============== #
sub range{
my $numarray = shift(@_);               # 削除原子群情報

my @numarray1 = split(/,/, $numarray);
my @numarray2 = ();             # 範囲指定情報も含めた最終的情報を格納する
my $count = 0;                          # numarray2 の配列番号
foreach(@numarray1){
        if(/-/){        # x-y 等の範囲指定を認識
                my @wide = split(/-/, $_);      # 開始と終了に分ける
                if($wide[0] =~ /^$/){   # 開始がない場合
                        $wide[0] = 1;
                }
                elsif($wide[1] =~ /^$/){        # 終了がない場合
                        $numarray2[$count++] = $wide[0];
                        $numarray2[$count++] = -1;
                        last;
                }
                my $i = 0;
                for($i = $wide[0]; $i <= $wide[1]; $i++){
                        # 範囲をすべて数字にする
                        $numarray2[$count++] = $i;
                }
        }
        else{
                $numarray2[$count++] = $_;
        }
}
return(@numarray2);
}

# ============== adjust ============== #
sub adjust{
my $max = shift(@_);    # 調整後文字数
my $line = shift(@_);   # 文字列

my $space = length($line);      # 文字列の長さ
$space = $max - $space;         # 追加するスペース数
$space = " " x $space;                  # スペース
$line = $space . $line;         # スペース追加
return($line);
}

# ============== crecone ============== #
sub crecone{
my $in = shift(@_);
my $out = shift(@_);
```

```
my $d = shift(@_);

my @arrayx = ();        # 対象原子の X 座標
my @arrayy = ();        # 対象原子の Y 座標
my @arrayz = ();        # 対象原子の Z 座標
my @atom = ();          # 対象原子の原子順序番号
my @atomname = ();      # 対象原子の原子名
my @amino = ("SOL", "WAT", "HOH", "ALA", "ARG", "ASN", "ASP", "CYS", "GLN", "GLU",
"GLY", "HIS", "HIP", "HID", "HIE", "ILE", "LEU", "LYS", "MET", "PHE", "PRO", "SER",
"THR", "TRP", "TYR", "VAL");
        # 除外残基

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        my $line = $_;
        if(/^TER/i){
                print OUT "TER¥n";
        }
        elsif(/^END/i){
                last;
        }
        elsif(/^CONECT/i){
                next;
        }
        elsif((/^HETATM/i) || (/^ATOM/i)){
                my $resc = substr($line, 17, 3);        # 残基情報取得
                my $hantei = 1;
                foreach(@amino){
                        # アミノ酸残基以外（非標準残基）はフラグを 1（作業マーク）のままに
                        if($resc =~ /^$_$/i){
                                $hantei = 0;
                        }
                }
                if($hantei == 1){
                        my $x = substr($line, 30, 8);
                        $x =~ s/¥s//g;
                        $x =~ s/¥n//;
                        push(@arrayx, $x);      # 対象原子の X 座標を格納
                        my $y = substr($line, 38, 8);
                        $y =~ s/¥s//g;
                        $y =~ s/¥n//;
                        push(@arrayy, $y);      # 対象原子の Y 座標を格納
                        my $z = substr($line, 46, 8);
                        $z =~ s/¥s//g;
                        $z =~ s/¥s//;
                        push(@arrayz, $z);      # 対象原子の Z 座標を格納
                        $x = substr($line, 6, 5);
                        $x =~ s/¥s//g;
                        $x =~ s/¥n//;
                        push(@atom, $x);        # 対象原子の原子順序番号を格納
                        my $name = substr($line, 13, 1);
                        push(@atomname, $name);# 対象原子の原子を格納
                }
                print OUT $line;
        }
}
close(IN);

my $targetcon = 0;      # 標的原子のループカウンタ
my $othercon = 0;               # 比較対象原子のループカウンタ
my $str = "";   # 出力用の接続する原子の原子順序番号
foreach(@atom){
        my $now = $_;   # 標的原子
        my $x1 = $arrayx[$targetcon];   # 標的原子の X 座標
        my $y1 = $arrayy[$targetcon];   # 標的原子の Y 座標
        my $z1 = $arrayz[$targetcon];   # 標的原子の Z 座標
        my $name1 = $atomname[$targetcon];      # 標的原子の原子
        foreach(@atom){
                my $x2 = $arrayx[$othercon];    # 比較対象原子の X 座標
                my $y2 = $arrayy[$othercon];    # 比較対象原子の Y 座標
                my $z2 = $arrayz[$othercon];    # 比較対象原子の Z 座標
                my $name2 = $atomname[$othercon];       # 比較対象原子の原子
                my $distance = ((($x2 - $x1) ** 2) + (($y2 - $y1) ** 2) +
                                ((($z2 - $z1) ** 2)) ** 0.5;
                        # 距離を演算
```

```
                if($distance <= $d){    # 距離がこの範囲内だったら接続していると見なす
                        if((($name1 eq "H") || ($name2 eq "H"))
                                && ($distance > 1.20)){
                                # ただし，水素原子との距離は 1.20Åまでとする.
                                $othercon++;
                                next;
                        }
                        elsif($distance == 0){
                                # 0 距離は自身であるのでスキップ
                                $othercon++;
                                next;
                        }
                        my $line = &adjust(5, $_);
                        $str = $str . $line;    # 接続原子を出力用に
                }
                $othercon++;
        }
        $othercon = 0;
        if($str !~ /^$/){
                $now = &adjust(5, $now);
                print OUT "CONECT$now$str¥n";  # 出力
                $str = "";
        }
        $targetcon++;
}
print OUT "END¥n";
close(OUT);
}

# =============== file =============== #
## ファイルの入力と存在を確認
sub file{
my $file = shift(@_);

if($file =~ /^$/){
        print " ERROR: 入力ファイルが指定されていません. ¥n";
        exit;
}
if(-f $file){
}
else{
        print " ERROR: 指定された入力ファイルは存在しません. ¥n";
        exit;
}
}

# =============== help =============== #
sub help{
print << "help";
 PDB 抽出プログラム
  *VIEW モード
   ¥$ pinch -V 表示列数 入力ファイル
  *電荷計算モード
   ¥$ pinch -I 入力ファイル
  *原子順序番号指定モード
   ¥$ pinch -AN 番号(範囲指定可能) 入力ファイル 出力ファイル
  *残基順序番号指定モード
   ¥$ pinch -RN 残基番号(範囲指定可能) 入力ファイル 出力ファイル
  *残基指定モード
   ¥$ pinch -R 残基名 入力ファイル 出力ファイル
  *ユニット指定モード
   ¥$ pinch -U ユニット名 入力ファイル 出力ファイル
  *詳細原子指定モード
   ¥$ pinch -DA 原子名 入力ファイル 出力ファイル
  *詳細残基指定モード
   ¥$ pinch -DR 残基名.残基番号 入力ファイル 出力ファイル
     *** 入力ファイルの引数を入出力ファイルにすると上書き可能 ***

help
}
```

次に PDB 部分削除プログラム rashid について説明する。rashid は pinch 同様、任意の対象を指定し、その対象を削除するプログラムである。以下に、対象を選択する手法を

示す。

method（第一引数）		対象（第二引数）	機能
ロングオプション	ショートオプション		
--atomnumber	-AN	シリアル番号	シリアル番号を指定して削除
--residuenumber	-AR	残基番号	残基番号を指定して削除
--atom	-A	原子記号	原子記号を指定して削除
--residue	-R	残基名	残基名を指定して削除
--detailatom	-DA	原子タイプ	原子タイプを指定して削除
--detailresidue	-DR	残基名.残基番号	残基名と残基番号を両方指定して削除
--unit	-U	Chain ID	Chain IDを指定して削除

選択手法のほとんどは pinch と同じで、--atom という手法が追加されているだけである。コマンドも以下のように、pinch が rashid に変わっただけである。

```
$ rashid <method> <target> input.pdb [output.pdb]
```

以下に rashid のソースコード (プログラム言語: Perl) を示す。

```
#! /usr/bin/perl

use strict;
use File::Temp;

my $option = shift(@ARGV);
my $select = shift(@ARGV);
my $in = shift(@ARGV);
my $out = shift(@ARGV);

if(($option =~ /^--help$/i) || ($option =~ /^-h$/i)){
        &help;
        exit;
}
elsif(($option =~ /^--information$/i) || ($option =~ /^-I$/i)){
        if($select !~ /^$/){
                if(-f $select){
                        &info($select);
                        exit;
                }
                else{
                        print " ERROR: 指定された入力ファイルは存在しません. ¥n";
                }
        }
        else{
                print " ERROR: 入力ファイルを指定してください. ¥n";
        }
}

if($in !~ /^$/){
        if(-f $in){
                my $tmp1 = File::Temp->new(TEMPLATE => '.rashid-1XXXXXX');
                my $tmp2 = File::Temp->new(TEMPLATE => '.rashid-2XXXXXX');
                $SIG{'TERM'} = $SIG{'PIPE'} = $SIG{'HUP'} = $SIG{'INT'} = sub {
                        unlink $tmp1;
                        unlink $tmp2;
                        exit;
                };      # プログラムが中断した場合の処理

                if(($option =~ /^--view$/i) || ($option =~ /^-V$/i)){
                        &view($select, $in);
                        exit;
                }
                elsif(($option =~ /^--atomnumber$/i) || ($option =~ /^-AN$/i)){
                        &atomorder($select, $in, $tmp1);
                }
                elsif(($option =~ /^--residuenumber$/i) || ($option =~ /^-RN$/i)){
                        &residueorder($select, $in, $tmp1);
                }
                elsif(($option =~ /^--atom$/i) || ($option =~ /^-A$/i)){
```

```
                        &atom($select, $in, $tmp1);
                }
                elsif(($option =~ /^--residue$/i) || ($option =~ /^-R$/i)){
                        &residue($select, $in, $tmp1);
                }
                elsif(($option =~ /^--unit$/i) || ($option =~ /^-U$/i)){
                        if($select =~ /^none$/i){
                                $select = " ";
                        }
                        &unit($select, $in, $tmp1);
                }
                elsif(($option =~ /^--detailresidue$/) || ($option =~ /^-DR$/i)){
                        my @check = split(/,/, $select);
                        my $checkflag = 0;
                        foreach(@check){
                                if($_ !~ /^.+¥..+$/){
                                        $checkflag = 1;
                                        last;
                                }
                        }
                        undef @check;
                        if($checkflag == 1){
                                print " ERROR: 指定残基の表現が違います.
                                 [残基名.残基番号]で指定する. ¥n";
                                exit;
                        }
                        else{
                                &detailresidue($select, $in, $tmp1);
                        }
                }
                elsif(($option =~ /^--detailatom$/) || ($option =~ /^-DA$/i)){
                        &detailatom($select, $in, $tmp1);
                }
                else{
                        print " ERROR: 未定義のオプションです. ¥n";
                        exit;
                }
                &terdel($tmp1, $tmp2);
                &cleanatom($tmp2, $tmp1);
                &crecone($tmp1, $tmp2, 1.9);
                if($out =~ /^$/){
                        print " INFORMATION: 出力先が指定されていません. ¥n";
                        print " INFORMATION: 上書き(O), 別名で保存(S),
                                何もしない(C)¥n > ";
                        my $user = <STDIN>;
                        $user =~ s/¥n//;
                        if($user =~ /^O$/i){
                                unlink $in;
                                rename $tmp2, $in;
                        }
                        elsif($user =~ /^S$/i){
                                print " ファイル名: ";
                                $user = <STDIN>;
                                $user =~ s/¥n//;
                                rename $tmp2, $user;
                        }
                        else{
                                exit;
                        }
                }
                else{
                        rename $tmp2, $out;
                }
        }
        else{
                print " ERROR: 指定された入力ファイルは存在しません. ¥n";
        }
}
else{
        if(($option =~ /^--view$/i) || ($option =~ /^-V$/i)){
                print " ERROR: 表示する列数を入力してください. ¥n";
        }
        else{
                print " ERROR: 入力ファイルを指定してください. ¥n";
        }
}
```

```
exit;

# =============== view =============== #
## VIEWモード
sub view{
my $column = shift(@_);         # 列数
my $in = shift(@_);                     # 入力ファイル

my $residue1 = "";      # 現在の残基名
my $residue2 = "";      # 前の行の残基名
my $atom1 = "";
my $atom2 = "";
my $start = 0;                  # 最初の処理のためのブランク処理フラグ
my $ncolumn = 0;                # 現在の列数
my $lastatom = 0;
open(IN, "$in");
while(<IN>){
        if((/^atom/i) || (/^hetatm/i)){
                my $before = $lastatom;                 # 前回の原子順序番号
                $lastatom = substr($_, 6, 5);   # 現在の原子順序番号
                $residue1 = substr($_, 17, 9); # 現在の残基名
                if($residue1 eq $residue2){     # 残基名同じだったら何もしない
                        next;
                }
                elsif($start == 0){             # 最初のブランク処理
                        $residue2 = substr($_, 17, 9); # 記憶すべき残基名
                        $atom1 = substr($_, 6, 5);      # 開始原子順序番号
                        $atom1 =~ s/¥s//g;
                        $start = 1;
                        print " ";
                }
                else{
                        $atom2 = $before;               # 前回の原子順序番号が必要
                        $atom1 = &adjust1(5, $atom1);   # 開始原子順序番号
                        $atom2 = &adjust1(5, $atom2);   # 終了原子順序番号
                        print "$atom1 - $atom2:$residue2";
                        $atom1 = $lastatom;
                        # 開始原子順序番号に現在の原子順序番号を
                        $residue2 = substr($_, 17, 9); # 現在の残基名を記憶
                        $ncolumn++;                     # 列数を増やす
                        if($ncolumn >= $column){        # 指定列数に達したら改行
                                print "¥n ";
                                $ncolumn = 0;
                        }
                        else{
                                print " | ";    # それ以外は区切るだけ
                        }
                }
        }
}
close(IN);

$atom2 = $lastatom;             # ファイル読み込み後に残ったデータ処理
$atom1 = &adjust1(5, $atom1);
$atom2 = &adjust1(5, $atom2);
print "$atom1 - $atom2:$residue2¥n";
}

# =============== info =============== #
## 系の情報を調べる
sub info{
my $in = shift(@_);

my %charge_residues = (#  電荷情報
        "LYS" => "1",
        "ARG" => "1",
        "ASP" => "-1",
        "GLU" => "-1",
        "HIP" => "1",
        "CYM" => "1",
        "Na" => "1",    # Na+
        "K" => "1",             # K+
        "Zn" => "2",    # Zn+
        "ZIN" => "2",   # Zn+
        "Ca" => "2",    # Ca+
        "CAL" => "2",   # Ca+
```

```
        "DA" => "-1",
        "DG" => "-1",
        "DC" => "-1",
        "DT" => "-1",
        "DA3" => "-1",
        "DG3" => "-1",
        "DC3" => "-1",
        "DT3" => "-1",
        "RA" => "-1",
        "RG" => "-1",
        "RC" => "-1",
        "RU" => "-1",
        "RA3" => "-1",
        "RG3" => "-1",
        "RC3" => "-1",
        "RU3" => "-1"
);
my @noncharge_residues = (
        "ACE", "ALA", "ASN", "CYS", "GLN", "GLY", "HIS", "HID", "HIE", "ILE",
        "LEU", "MET", "NME", "PHE", "PRO", "SER", "THR", "TRP", "TYR", "VAL",
        "SOL", "HOH", "WAT", "DT5", "DA5", "DC5", "DG5", "RU5", "RA5", "RC5", "RG5"
);

my $resnum = 0;
my $atomnum = 0;
my $charge = 0;
my $water = 0;
my $residue1 = "";
my $start = 0;
my @unknowns = ();
my @hiss = (0, 0, 0, 0);        # His の状態(HIS, HID, HIE, HIP)
open(IN, "$in");
while(<IN>){
        if((/^HETATM/i) || (/^ATOM/)){
                $atomnum ++;    #  原子数+1
                my $residue2 = substr($_, 17, 9);       #  現在の残基情報
                if($residue1 ne $residue2){
                        #  前の残基情報と一致しない場合(残基が変わった場合)
                        $resnum ++;     #  残基数+1
                        if($residue2 =~ /^HI(S|D|E|P)/i){
                                if($residue2 =~ /^HIS/i){
                                        $hiss[0] ++;
                                }
                                elsif($residue2 =~ /^HID/i){
                                        $hiss[1] ++;
                                }
                                elsif($residue2 =~ /^HIE/i){
                                        $hiss[2] ++;
                                }
                                elsif($residue2 =~ /^HIP/i){
                                        $hiss[3] ++;
                                }
                                else{
                                        print " WARNING: Unknown His exists¥n";
                                }
                        }
                        if(($residue2 =~ /^WAT/i) || ($residue2 =~ /^HOH/i)
                || ($residue2 =~ /^SOL/i)){     #  残基名が WAT あるいは HOH の場合
                                $water ++;      #  水分子数+1
                        }
                        if($start == 0){        #  最初の残基の異なりは無効
                                $start = 1;     #  フラグを通常に戻す
                                $residue1 = $residue2;
                                #  現在の残基情報を前の残基情報として登録
                                next;
                        }
                        else{   #  通常状態
                                $residue1 = substr($residue1, 0, 3);
                                #  残基名のみ取得
                                $residue1 =~ s/¥s+//g;
                                $residue1 =~ s/[¥+¥*¥/]//g;
                                #  + や *, /は _ に変換
                                my @data = keys(%charge_residues);
                                #  電荷情報からキーを配列に格納
                                my $charge_flag = 0;
                                foreach(@data){
```

```
                                        if($residue1 =~ /^$_$/i){
                                                #  キーが一致した場合
                                                $charge += $charge_residues{$_};
                                                #  キーの値(電荷)を現在の電荷に加算
                                                $charge_flag = 1;
                                                last;
                                        }
                                }
                                if(($charge_flag == 0) &&
                                (grep(/^$residue1$/, @noncharge_residues) == 0)){
                                        push(@unknowns, $residue1);
                                }
                                $residue1 = $residue2;
                                #  現在の残基を前の残基として登録
                        }
                }
        }
}
close(IN);

$residue1 = substr($residue1, 0, 3);
#  ループできなかった情報の処理      #  残基名のみ取得
$residue1 =~ s/[¥+¥*¥/]/_/g;    #  + や *, /は    に変換
my @data = keys(%charge_residues);      #  電荷情報からキーを配列に格納
my $charge_flag = 0;
foreach(@data){
        if($residue1 =~ /^$_$/i){       #  キーが一致した場合
                $charge += $charge_residues{$_};
                #  キーの値(電荷)を現在の電荷に加算
                $charge_flag = 1;
                last;
        }
}
if(($charge_flag == 0) && (grep(/^$residue1$/, @noncharge_residues) == 0)){
        push(@unknowns, $residue1);
}

my $unknown_residue = join(" + ", @unknowns);
if($#unknowns != -1){
        $unknown_residue = " + " . $unknown_residue;
}

my $notwater = $resnum - $water;
print << "INFO";
 INFORMATION for $in
   All of Atom         : $atomnum
   All of Residue      : $resnum
    * Not water molecule: $notwater
    * Water molecule    : $water
   System charge       : $charge $unknown_residue

   HIS : HID : HIE: HIP = $hiss[0] : $hiss[1] : $hiss[2] : $hiss[3]

INFO
}

# =============== atom =============== #
## 原子指定モード
sub atom{
my $atom = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @atoms = split(/,/, $atom);

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^ATOM/i) || (/^HETATM/i)){
                my $line = $_;
                my $flag = 0;
                my $nowatom = substr($_, 12, 2);        # 原子名取得
                $nowatom =~ s/¥d//g;
                $nowatom =~ s/¥s+//g;
                foreach(@atoms){
                        if($nowatom eq $_ ){     # 照らし合わせ
```

```
                                $flag = 1;
                                last;
                        }
                }
                if($flag == 0){
                        print OUT $line;
                }
        }
        elsif(/^TER/){
                print OUT "TER¥n";
        }
}
close(IN);
close(OUT);
}

# =============== detailatom =============== #
## 詳細原子指定モード
sub detailatom{
my $atom = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @atoms = split(/,/, $atom);

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^ATOM/i) || (/^HETATM/i)){
                my $line = $_;
                my $flag = 0;
                my $nowatom = substr($_, 12, 4);        # 詳細原子名取得
                $nowatom =~ s/¥s//g;
                foreach(@atoms){
                        if($nowatom =~ /^$_$/){ # 照らし合わせ
                                $flag = 1;
                                last;
                        }
                }
                if($flag == 0){
                        print OUT $line;
                }
        }
        elsif(/^TER/){
                print OUT "TER¥n";
        }
}
close(IN);
close(OUT);
}

# =============== residue =============== #
## 残基指定モード
sub residue{
my $residue = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @residues = split(/,/, $residue);

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^ATOM/i) || (/^HETATM/i)){
                my $line = $_;
                my $flag = 0;
                my $nowres = substr($_, 17, 3);# 残基名取得
                foreach(@residues){
                        if($nowres eq $_){      # 照らし合わせ
                                $flag = 1;
                                last;
                        }
                }
                if($flag == 0){
                        print OUT $line;
                }
```

```
                else{
                        print OUT "TER¥n";      # 消した後には TER を
                }
        }
        elsif(/^TER/){
                print OUT "TER¥n";
        }
}
close(IN);
close(OUT);
}

# =============== atomorder =============== #
## 原子順序指定モード
sub atomorder{
my $atomrange = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @atomorder = &range($atomrange);     # 範囲指定を分解

my $infinity = 0;       # 無限判定
my $check = pop(@atomorder);
if($check == -1){       # range ルーチンで無限まで続くのなら-1 が返されているはず
        $infinity = 1;
}
else{
        push(@atomorder, $check);       # 配列の最後を取り出したから元に戻す
}

@atomorder = sort {$a <=> $b} (@atomorder);     # ソート

my $find = 0;   # 指定された原子か？
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^ATOM/i) || (/^HETATM/i)){
                my $line = $_;
                my $nowatomorder = substr($_, 6, 5);    # 原子順序番号取得
                $nowatomorder =~ s/¥s+//g;
                if(($infinity == 1) && ($nowatomorder >= $atomorder[0])){
        # 無限かつ指定された原子より大きければ，消去し，TER を
                        print OUT "TER¥n";
                        next;
                }
                else{
                        my $find = 0;
                        foreach(@atomorder){    # 範囲内に現在の原子順序があるか
                                if($_ == $nowatomorder){
                                        $find = 1;
                                        last;
                                }
                        }
                        if($find == 0){ # 指定されたものでない
                                print OUT "$line";
                        }
                        else{
                                print OUT "TER¥n";      # 指定されたものなら TER を
                        }
                }
        }
        elsif(/^TER/i){
                print OUT "TER¥n";
        }
}
close(IN);
close(OUT);
}

# =============== resiueorder =============== #
## 残基順序指定モード
sub residueorder{
my $resrange = shift(@_);
my $in = shift(@_);
my $out = shift(@_);
```

```
my @resorder = &range($resrange);       # 範囲指定を分解

my $check = pop(@resorder);
if($check == -1){       # range ルーチンで無限まで続くのなら-1 が返されているはず
        $check = pop(@resorder);
}
else{
        push(@resorder, $check);        # 配列の最後を取り出したから元に戻す
        $check = 0;
}

@resorder = sort {$a <=> $b} (@resorder);       # ソート

my $find = 0;   # 指定された残基番号か？
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^ATOM/i) || (/^HETATM/i)){
                my $line = $_;
                my $nowresorder = substr($_, 22, 4);    # 残基順序番号取得
                $nowresorder =~ s/¥s+//g;
                if(($check != 0) && ($check <= $nowresorder)){
                        print OUT "TER¥n";
                }
                else{
                        my $find = 0;
                        foreach(@resorder){     # 範囲内に現在の残基順序番号があるか
                                if($_ == $nowresorder){
                                        $find = 1;
                                        last;
                                }
                        }
                        if($find == 0){ # 指定されたものでない
                                print OUT "$line";
                        }
                        else{
                                print OUT "TER¥n";      # 指定されたものなら TER を
                        }
                }
        }
        elsif(/^TER/i){
                print OUT "TER¥n";
        }
}
close(IN);
close(OUT);
}

# =============== unit =============== #
## ユニット指定モード
sub unit{
my $unit = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @units = split(/,/, $unit);

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^ATOM/i) || (/^HETATM/i)){
                my $line = $_;
                my $flag = 0;
                my $nowunit = substr($_, 21, 1);        # ユニット名取得
                foreach(@units){
                        if($nowunit eq $_){
                                $flag = 1;
                                last;
                        }
                }
                if($flag == 0){
                        print OUT $line;
                }
                else{
                        print OUT "TER¥n";      # 消した後には TER を
                }
```

```
        }
        elsif(/^TER/){
                print OUT "TER¥n";
        }
}
close(IN);
close(OUT);
}

# =============== detailresidue =============== #
## 詳細残基指定モード
sub detailresidue{
my $select = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @residues = split(/,/, $select);

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^ATOM/i) || (/^HETATM/i)){
                my $line = $_;
                my $flag = 0;
                my $nowdata = substr($line, 17, 9);
                my $nowres = substr($nowdata, 0, 3);   # 残基名取得
                $nowres =~ s/¥s//g;
                my $nowresnum = substr($nowdata, 5, 4);        # 残基番号取得
                $nowresnum =~ s/¥s//g;
                foreach(@residues){
                        (my $checkres, my $checkresnum) = split(/¥./, $_);
                        if(($nowres eq $checkres)
                                && ($nowresnum == $checkresnum)){
                                $flag = 1;
                                last;
                        }
                }
                if($flag == 0){
                        print OUT $line;
                }
                else{
                        print OUT "TER¥n";
                }
        }
        elsif(/^TER/){
                print OUT "TER¥n";
        }
}
close(IN);
close(OUT);
}

# =============== terdel =============== #
sub terdel{
my $in = shift(@_);
my $out = shift(@_);

my $beforeline = "";
my $flag = 0;
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        my $line = $_;  # 現在行の改行とスペースを除去して，データを判別できる形にして
$line1 へ
        if(($flag == 0) && ($line =~ /^TER/)){
                next;
        }
        if($line ne $beforeline){
                # 前の行と同じ（おそらく重複 TER）でなかったら記述
                print OUT $line;
                $beforeline = $line;
                $flag = 1;
        }
}
close(IN);
close(OUT);
```

```
}

# ============== cleanatom ============== #
## 原子順序番号の正規化
sub cleanatom{
my $in = shift(@_);
my $out = shift(@_);

my $count = 0;
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^ATOM/i) || (/^HETATM/)){
                $count++;
                my $line = &adjust1(5, $count);
                substr($_, 6, 5) = $line;
                print OUT;
        }
        else{
                print OUT;
        }
}
close(IN);
close(OUT);
}

# ============== crecone ============== #
sub crecone{
my $in = shift(@_);
my $out = shift(@_);
my $d = shift(@_);

my @arrayx = ();        # 対象原子の X 座標
my @arrayy = ();        # 対象原子の Y 座標
my @arrayz = ();        # 対象原子の Z 座標
my @atom = ();          # 対象原子の原子順序番号
my @atomname = ();      # 対象原子の原子名
my @amino = ("SOL", "WAT", "HOH", "ALA", "ARG", "ASN", "ASP", "CYS", "GLN", "GLU",
"GLY", "HIS", "HIP", "HID", "HIE", "ILE", "LEU", "LYS", "MET", "PHE", "PRO", "SER",
"THR", "TRP", "TYR", "VAL");
        # 除外残基

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        my $line = $_;
        if(/^TER/i){
                print OUT "TER¥n";
        }
        elsif(/^END/i){
                last;
        }
        elsif(/^CONECT/i){
                next;
        }
        elsif((/^HETATM/i) || (/^ATOM/i)){
                my $resc = substr($line, 17, 3);        # 残基情報取得
                my $hantei = 1;
                foreach(@amino){
                        # アミノ酸残基以外（非標準残基）はフラグを 1（作業マーク）のままに
                        if($resc =~ /^$_$/i){
                                $hantei = 0;
                        }
                }
                if($hantei == 1){
                        my $x = substr($line, 30, 8);
                        $x =~ s/¥s//g;
                        $x =~ s/¥n//;
                        push(@arrayx, $x);      # 対象原子の X 座標を格納
                        my $y = substr($line, 38, 8);
                        $y =~ s/¥s//g;
                        $y =~ s/¥n//;
                        push(@arrayy, $y);      # 対象原子の Y 座標を格納
                        my $z = substr($line, 46, 8);
                        $z =~ s/¥s//g;
                        $z =~ s/¥s//;
```

```
                        push(@arrayz, $z);      # 対象原子の Z 座標を格納
                        $x = substr($line, 6, 5);
                        $x =~ s/¥s//g;
                        $x =~ s/¥n//;
                        push(@atom, $x);        # 対象原子の原子順序番号を格納
                        my $name = substr($line, 13, 1);
                        push(@atomname, $name);# 対象原子の原子を格納
                }
                print OUT $line;
        }
}
close(IN);

my $targetcon = 0;      # 標的原子のループカウンタ
my $othercon = 0;               # 比較対象原子のループカウンタ
my $str = "";   # 出力用の接続する原子の原子順序番号
foreach(@atom){
        my $now = $_;   # 標的原子
        my $x1 = $arrayx[$targetcon];  # 標的原子の X 座標
        my $y1 = $arrayy[$targetcon];  # 標的原子の Y 座標
        my $z1 = $arrayz[$targetcon];  # 標的原子の Z 座標
        my $name1 = $atomname[$targetcon];      # 標的原子の原子
        foreach(@atom){
                my $x2 = $arrayx[$othercon];    # 比較対象原子の X 座標
                my $y2 = $arrayy[$othercon];    # 比較対象原子の Y 座標
                my $z2 = $arrayz[$othercon];    # 比較対象原子の Z 座標
                my $name2 = $atomname[$othercon];       # 比較対象原子の原子
                my $distance = ((($x2 - $x1) ** 2) + (($y2 - $y1) ** 2)
                        + (($z2 - $z1) ** 2)) ** 0.5;
                        # 距離を演算
                if($distance <= $d){    # 距離がこの範囲内だったら接続していると見なす
                        if((($name1 eq "H") || ($name2 eq "H"))
                                && ($distance > 1.20)){
                                # ただし，水素原子との距離は 1.20Åまでとする.
                                $othercon++;
                                next;
                        }
                        elsif($distance == 0){
                                # 0 距離は自身であるのでスキップ
                                $othercon++;
                                next;
                        }
                        my $line = &adjust1(5, $_);
                        $str = $str . $line;    # 接続原子を出力用に
                }
                $othercon++;
        }
        $othercon = 0;
        if($str !~ /^$/){
                $now = &adjust1(5, $now);
                print OUT "CONECT$now$str¥n";  # 出力
                $str = "";
        }
        $targetcon++;
}
print OUT "END¥n";
close(OUT);
}

# =============== range =============== #
sub range{
my $numarray = shift(@_);

$numarray =~ s/¥s//g;
my @numarray1 = split(/,/, $numarray);
my @numarray2 = ();
my $count = 0;
foreach(@numarray1){
        if(/-/){
                my @wide = split(/-/, $_);
                if($wide[0] =~ /^$/){
                        $wide[0] = 1;
                }
                elsif($wide[1] =~ /^$/){
                        $numarray2[$count++] = $wide[0];
                        $numarray2[$count++] = -1;
```

```
                        last;
                }
                my $i = 0;
                for($i = $wide[0]; $i <= $wide[1]; $i++){
                        $numarray2[$count++] = $i;
                }
        }
        else{
                $numarray2[$count++] = $_;
        }
}
return(@numarray2);
}

# =============== adjust1 =============== #
sub adjust1{
my $max = shift(@_);    # 調整後文字数
my $line = shift(@_);   # 文字列

my $space = length($line);      # 文字列の長さ
$space = $max - $space;         # 追加するスペース数
$space = " " x $space;                  # スペース
$line = $space . $line;         # スペース追加
return($line);
}

# =============== help =============== #
sub help{
print << "help";
 PDB 部分削除プログラム
  *VIEW モード
    ¥$ rashid --view 表示列数 入力ファイル
  *情報表示モード
    ¥$ rashid --information 入力ファイル
  *原子指定削除
    ¥$ rashid --atom 原子記号 入力ファイル 出力ファイル
  *原子順序番号指定削除
    ¥$ rashid --atomnumber 原子順序(範囲指定可能) 入力ファイル 出力ファイル
  *残基順序番号指定削除
    ¥$ rashid --residuenumber 残基順序(範囲指定可能) 入力ファイル 出力ファイル
  *残基指定削除
    ¥$ rashid --residue 残基名 入力ファイル 出力ファイル
  *詳細原子指定削除
    ¥$ rashid --detailatom 原子名 入力ファイル 出力ファイル
  *詳細残基指定削除
    ¥$ rashid --detailresidue 残基名.残基番号 入力ファイル 出力ファイル
  *ユニット指定削除
    ¥$ rashid --unit ユニット名 入力ファイル 出力ファイル
     *** 入力ファイルの引数を入出力ファイルにすると上書き可能 ***
help
}
```

最後に、PDB 結合プログラム uni について説明する。uni は pinch や rashid で編集した PDB を一つの PDB にまとめることができる。ここで、入力ファイルを input1.pdb、input2.pdb、input3.pdb とし、出力ファイルを output.pdb とすると、コマンドは以下のようになる。

```
$ uni output.pdb input1.pdb input2.pdb input3.pdb
```

なお、pinch や rashid では、出力ファイルを省略することができたが、uni では仕様上省略することができない。また、与える入力ファイルは無限に指定することができる。以下に uni のソースコード (プログラム言語: Perl) を示す。

```
#! /usr/bin/env perl

use strict;
use File::Temp;
use File::Basename;
```

```
my $option_flag = 0;
# 入力ファイルの途中にオプションに類似するファイルがあった場合の対処フラグ
my $renum_option = 0;   # 残基番号リセットフラグ
my $out = "";
my @inputs = ();
foreach(@ARGV){
        if((/^--help$/i) || (/^-H$/i)){
                &help;
                exit;
        }
        elsif(($option_flag < 2) && ((/^--ignore$/i) || (/^-I$/i))){
                # 残基情報リセットしない(デフォルト)
                $renum_option = 0;
                $option_flag = 1;
        }
        elsif(($option_flag < 2) && ((/^--force$/i) || (/^-F$/i))){
                # 残基情報をリセットする
                $renum_option = 1;
                $option_flag = 1;
        }
        elsif($out =~ /^$/){
                $out = &check_overwrite($_);
                $option_flag = 2;
        }
        else{
                &check_file($_);
                push(@inputs, $_);
        }
}

my $count = 0;
my @datas = ();
foreach(@inputs){
        my @tmps = &load_file($_);
        # ファイル読み込み(接続情報の除去，その他の情報除去)
        push(@datas, @tmps);
        if($#inputs > $count){  # 入力ファイルの最後でない場合のつなぎ目
                if($datas[$#datas] !~ /^TER/){
                        push(@datas, "TER¥n");
                }
        }
        else{                                                   # 最後の入力ファイル
                push(@datas, "END¥n");
        }
        $count ++;
}
my $flag = &check_atomorder(@datas);
if($flag == 1){
        @datas = &clean_atom(@datas);
}
if($renum_option == 1){
        $flag = &check_residueorder(@datas);
        if($flag == 1){
                @datas = &clean_residue(@datas);
        }
}
@datas = &terdel(@datas);
@datas = &crecone(1.9, @datas);

open(OUT, "> $out");
foreach(@datas){
        print OUT $_;
}
close(OUT);

exit;

# =============== load_file =============== #
sub load_file{
my $in = shift(@_);

my @datas = ();
open(IN, $in);
while(<IN>){
        if(/^(ATOM)|(HETATM)/){
```

```
                push(@datas, $_);
        }
        elsif(/^TER/){
                push(@datas, "TER¥n");
        }
}
close(IN);

return @datas;
}

# =============== check_atomorder =============== #
## 原子順序を振り直すかどうかのチェック
sub check_atomorder{
my @datas = @_;

my $flag = 0;
my $beforenum = 0;
foreach(@datas){
        if((/^(HETATM)|(ATOM)/)){
                my $nownum = substr($_, 6, 5);
                if($nownum <= $beforenum){
                        $flag = 1;
                        print "WARNING: Since atom order is conflict,
                                it will be fixed forcibly.¥n";
                        last;
                }
                $beforenum = $nownum;
        }
}

return $flag;
}

# =============== check_residueroder =============== #
## 残基番号を振り直すかどうかのチェック
sub check_residueorder{
my @datas = @_;

my $flag = 0;           # 振り直すフラグ  1:振り直す，0:振り直さない
my $before_residue = "";
my $before_residuenum = 0;
foreach(@datas){
        if((/^(ATOM)|(HETATM)/)){
                my $now_residue = substr($_, 17, 9);    # 残基情報取得
                if($now_residue ne $before_residue){
                        # 前の残基情報と異なる場合(新しい残基)
                        $before_residue = $now_residue;# 前の残基として登録
                        my $now_residuenum = substr($now_residue, 5, 4);
                        # 残基番号取得
                        $now_residuenum =~ s/^¥s*//;
                        if($now_residuenum < $before_residuenum){
                                $flag = 1;
                                print "WARNING: Since residue order is conflict,
                                        it will be fixed.¥n";
                                last;
                        }
                }
        }
}

return $flag;
}

# =============== clean_atom =============== #
## 原子順序番号の正規化
sub clean_atom{
my @datas = @_;

my @new_datas = ();
my $count = 0;
foreach(@datas){
        if((/^(HETATM)|(ATOM)/)){
                $count ++;
                my $line = &adjust(5, $count);
                substr($_ , 6, 5) = $line;
```

```
                push(@new_datas, $_);
        }
        elsif(/^TER/){
                push(@new_datas, "TER¥n");
        }
}

return @new_datas;
}

# =============== clean_residue =============== #
## 残基番号の正規化
sub clean_residue{
my @datas = @_;

my @new_datas = ();
my $count = 0;
my $num = 0;
my $residue1 = "";
my $residue2 = "";
foreach(@datas){
        if((/^(ATOM)|(HETATM)/)){
                $residue1 = substr($_, 17, 9);
                # 残基名 残基グループ 残基番号を取り出す
                if($residue1 ne $residue2){
                        # 残基情報が違ったらリセットし，残基番号カウントを+1
                        $residue2 = $residue1;
                        $count++;
                }
                $num = &adjust(4, $count);
                substr($_, 22, 4) = $num;               # 新しい残基番号を挿入
                push(@new_datas, $_);
        }
        else{
                push(@new_datas, $_);
        }
}

return @new_datas;
}

# =============== terdel =============== #
sub terdel{
my @datas = @_;

my $line1 = "";
my $line2 = "";
my @new_datas = ();
foreach(@datas){
        $line1 = $_;
        # 現在行の改行とスペースを除去して，データを判別できる形にして$line1 へ
        $line1 =~ s/¥n//;
        $line1 =~ s/¥s+//g;
        if($line1 eq $line2){
                # 前の行と同じ（おそらく重複 TER）だったらスキップすることで，削除
                next;
        }
        $line2 = $line1;
        # 重複していなければ，$line2 に保存し，重複していない現在行を出力
        push(@new_datas, $_);
}

return @new_datas;
}

# =============== crecone =============== #
sub crecone{
my $d = shift(@_);      # 結合距離
my @datas = @_;         # データ

my @arrayx = ();        # 対象原子の X 座標
my @arrayy = ();        # 対象原子の Y 座標
my @arrayz = ();        # 対象原子の Z 座標
my @atom = ();          # 対象原子の原子順序番号
my @atomname = ();      # 対象原子の原子名
my @amino = ("SOL", "WAT", "HOH", "ALA", "ARG", "ASN", "ASP", "CYS", "GLN", "GLU",
```

```
"GLY", "HIS", "HIP", "HID", "HIE", "ILE", "LEU", "LYS", "MET", "PHE", "PRO", "SER",
"THR", "TRP", "TYR", "VAL", "DA5", "DT5", "DG5", "DC5", "DA3", "DT3", "DG3", "DC3",
"DA", "DT", "DG", "DC", "RA5", "RU5", "RG5", "RC5", "RA3", "RU3", "RG3", "RC3",
"RA", "RU", "RG", "RC");
        # 除外残基

my @new_datas = ();
foreach(@datas){
        my $line = $_;
        if(/^END/){
                last;
        }
        elsif((/^(HETATM)|(ATOM)/)){
                my $resc = substr($line, 17, 3);        # 残基情報取得
                $resc =~ s/¥s//g;
                my $hantei = 1;
                foreach(@amino){
                        # アミノ酸残基以外（非標準残基）はフラグを 1（作業マーク）のままに
                        if($resc eq $_){
                                $hantei = 0;
                        }
                }
                if($hantei == 1){
                        my $x = substr($line, 30, 8);
                        $x =~ s/¥s//g;
                        $x =~ s/¥n//;
                        push(@arrayx, $x);      # 対象原子の X 座標を格納
                        my $y = substr($line, 38, 8);
                        $y =~ s/¥s//g;
                        $y =~ s/¥n//;
                        push(@arrayy, $y);      # 対象原子の Y 座標を格納
                        my $z = substr($line, 46, 8);
                        $z =~ s/¥s//g;
                        $z =~ s/¥s//;
                        push(@arrayz, $z);      # 対象原子の Z 座標を格納
                        $x = substr($line, 6, 5);
                        $x =~ s/¥s//g;
                        $x =~ s/¥n//;
                        push(@atom, $x);        # 対象原子の原子順序番号を格納
                        my $name = substr($line, 13, 1);
                        push(@atomname, $name);# 対象原子の原子を格納
                }
        }
        push(@new_datas, $line);
}
close(IN);
undef @datas;

my $targetcon = 0;      # 標的原子のループカウンタ
my $othercon = 0;               # 比較対象原子のループカウンタ
my $str = "";   # 出力用の接続する原子の原子順序番号
foreach(@atom){
        my $now = $_;   # 標的原子
        my $x1 = $arrayx[$targetcon];  # 標的原子の X 座標
        my $y1 = $arrayy[$targetcon];  # 標的原子の Y 座標
        my $z1 = $arrayz[$targetcon];  # 標的原子の Z 座標
        my $name1 = $atomname[$targetcon];      # 標的原子の原子
        foreach(@atom){
                my $x2 = $arrayx[$othercon];    # 比較対象原子の X 座標
                my $y2 = $arrayy[$othercon];    # 比較対象原子の Y 座標
                my $z2 = $arrayz[$othercon];    # 比較対象原子の Z 座標
                my $name2 = $atomname[$othercon];       # 比較対象原子の原子
                my $distance = ((($x2 - $x1) ** 2) + (($y2 - $y1) ** 2)
                        + (($z2 - $z1) ** 2)) ** 0.5;
                        # 距離を演算
                if($distance <= $d){    # 距離がこの範囲内だったら接続していると見なす
                        if((($name1 eq "H") || ($name2 eq "H"))
                                && ($distance > 1.20)){
                                # ただし，水素原子との距離は 1.20Åまでとする.
                                $othercon++;
                                next;
                        }
                        elsif($distance == 0){
                                        # 0 距離は自身であるのでスキップ
                                $othercon++;
                                next;
```

```
                        }
                        my $line = &adjust(5, $_);
                        $str = $str . $line;    # 接続原子を出力用に
                }
                $othercon++;
        }
        $othercon = 0;
        if($str !~ /^$/){
                $now = &adjust(5, $now);
                push(@new_datas, "CONECT$now$str¥n");
                $str = "";
        }
        $targetcon++;
}
push(@new_datas, "END¥n");

return @new_datas;
}

# =============== check_file =============== #
# ファイルチェック
sub check_file{
my $file = shift(@_);

if(! -f $file){
        print " ERROR: No such file: $file¥n";
        exit;
}

return $file;
}

# =============== check_overwrite =============== #
# 上書き確認
sub check_overwrite{
my $file = shift(@_);

if(-f $file){
        print "$file exist. Do you want to overwrite it? (y/N) > ";
        my $user = <STDIN>;
        $user =~ s/¥n//;
        if($user !~ /^y$/i){
                print "Canceled.¥n";
                exit;
        }
}

return $file;
}

# =============== adjust =============== #
sub adjust{
my $max = shift(@_);    # 調整後文字数
my $line = shift(@_);   # 文字列

my $space = length($line);      # 文字列の長さ
$space = $max - $space;         # 追加するスペース数
$space = " " x $space;                  # スペース
$line = $space . $line;         # スペース追加
return($line);
}

# =============== help =============== #
sub help{
my $script_name = basename($0);
print << "HELP";
 PDB binding program
  ¥$ $script_name [OPTION] OUTPUT INPUT1 INPUT2 ...

  OPTION
   -I, --ignore   DO NOT fix residue order (default)
   -F, --force    Fix residue order

HELP
}
```

附録 A-2: PDB 接続情報編集プログラム

PDB 形式のファイルの原子同士の接続情報は、タンパク質や DNA のような一般的な分子の場合、それぞれの原子にユニークな原子タイプが割り振られるため、記述する必要がない。しかし、リガンドのような低分子の場合は、ファイルの各原子の情報の後に記述する必要がある。その場合、構造を見ながら、記述していくことになるが、それは非常に困難である。そこで、その作業を自動化するプログラムを作成した。また、接続情報を一発で削除するプログラムも作成した。前者を crecone、後者を discone と名付けた。

crecone のコマンドを以下に示す。

```
$ crecone [option] input.pdb [output.pdb]
```

option と出力ファイル output.pdb は省略することができる。出力ファイルを省略した場合、実行中にどのように処理するかを尋ねられる。さらに以下にオプションとその機能を示す。

option（第一引数）		機能
ロングオプション	ショートオプション	
--distance	-D	結合しているとみなす距離を指定（デフォルト 1.9 Å）
--ignore	-I	水素原子と結合している距離（1.2 Å）を無視
--force	-F	タンパク質、DNA、RNAに対しても、接続情報を出力

このプログラムは、結合している原子を原子間の距離で判断する。従って、不安定な構造の場合、誤った接続情報となる。その場合、--distance オプションを用いて、ユーザが結合距離を指定する必要がある。以下に crecone のソースコード (プログラム言語: Perl) を示す。

```
#! /usr/bin/env perl

use strict;
use File::Temp;

my $in = "";
my $out = "";
my $bond_length_flag = 0;
my $bond_length = 1.9;
my $option_force = 0;
my $option_ignore = 0;
foreach(@ARGV){
        if((/^--force$/i) || (/^-F$/i)){        #
                $option_force = 1;
        }
        elsif((/^--ignore$/i) || (/^-I$/i)){
                $option_ignore = 1;
        }
        elsif((/^--distance$/i) || (/^-D$/i)){
                $bond_length_flag = 1;
        }
        elsif($bond_length_flag == 1){
                $bond_length = &check_float($_);
                $bond_length_flag = 0;
        }
        else{
                if($in eq ""){
                        $in = $_;
                }
                elsif($out eq ""){
```

```
                        $out = $_;
                }
                else{
                        print " ERROR: Too many arguments¥n";
                        exit;
                }
        }
}

if(($in =~ /^--help$/i) || ($in =~ /^-H$/i)){
        &help;
        exit;
}
if($in !~ /^$/){
        if(-f $in){
                my $tmp = File::Temp->new(TEMPLATE => '.crecone-XXXXXX');
                $SIG{'TERM'} = $SIG{'PIPE'} = $SIG{'HUP'} = $SIG{'INT'} = sub {
                        unlink $tmp;
                        exit;
                };      # プログラムが中断した場合の処理

                &work($option_force, $option_ignore, $bond_length, $in, $tmp);

                if($out =~ /^$/){
                        print " INFORMATION: 出力先が指定されていません. ¥n";
                        print " INFORMATION: 上書き(O), 別名で保存(S),
                                                何もしない(C)¥n > ";
                        my $user = <STDIN>;
                        $user =~ s/¥n//;
                        if($user =~ /^O$/i){
                                unlink $in;
                                rename $tmp, $in;
                        }
                        elsif($user =~ /^S$/i){
                                print " ファイル名 : ";
                                $user = <STDIN>;
                                $user =~ s/¥n//;
                                rename $tmp, $user;
                        }
                        else{
                                exit;
                        }
                }
                else{
                        rename $tmp, $out;
                }
        }
        else{
                print " ERROR: No such file: $in¥n";
        }
}
else{
        print " ERROR: Invalid argument¥n";
}
exit;

# --------------- work --------------- #
sub work{
my $option_force = shift(@_);
my $option_ignore = shift(@_);
my $bond_length = shift(@_);
my $in = shift(@_);
my $out = shift(@_);

my @arrayx = ();        #対象原子の X 座標
my @arrayy = ();        #対象原子の Y 座標
my @arrayz = ();        #対象原子の Z 座標
my @atom = ();          #対象原子の原子順序番号
my @atomname = ();      #対象原子の原子名
my @amino = ("SOL", "WAT", "HOH", "ACE", "ALA", "ARG", "ASN", "ASP", "ASH", "CYS",
"CYM", "CYX", "GLN", "GLU", "GLH", "GLY", "HIS", "HIP", "HID", "HIE", "ILE", "LEU",
"LYS", "MET", "NME", "PHE", "PRO", "SER", "THR", "TRP", "TYR", "VAL", "DA5", "DT5",
"DG5", "DC5", "DA3", "DT3", "DG3", "DC3", "DA", "DT", "DG", "DC", "RA5", "RU5",
"RG5", "RC5", "RA3", "RU3", "RG3", "RC3", "RA", "RU", "RG", "RC");
        #除外残基
```

```
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        my $line = $_;
        if(/^TER/i){
                print OUT "TER¥n";
        }
        elsif(/^END/i){
                last;
        }
        elsif(/^CONECT/i){
                next;
        }
        elsif((/^HETATM/i) || (/^ATOM/i)){
                my $resc = substr($line, 17, 3);        #残基情報取得
                $resc =~ s/¥s//g;
                my $hantei = 1;
                if($option_force == 0){
                        my @results = grep(/^$_$/, @amino);
                        # 非標準残基は処理しない
                        if($#results == -1){    # 標準残基に項目がない場合
                                $hantei = 0;
                        }
                }
                if($hantei == 1){
                        my $x = substr($line, 30, 8);
                        $x =~ s/¥s//g;
                        $x =~ s/¥n//;
                        push(@arrayx, $x);      #対象原子の X 座標を格納
                        my $y = substr($line, 38, 8);
                        $y =~ s/¥s//g;
                        $y =~ s/¥n//;
                        push(@arrayy, $y);      #対象原子の Y 座標を格納
                        my $z = substr($line, 46, 8);
                        $z =~ s/¥s//g;
                        $z =~ s/¥s//;
                        push(@arrayz, $z);      #対象原子の Z 座標を格納
                        $x = substr($line, 6, 5);
                        $x =~ s/¥s//g;
                        $x =~ s/¥n//;
                        push(@atom, $x);        #対象原子の原子順序番号を格納
                        my $name = substr($line, 13, 1);
                        push(@atomname, $name);#対象原子の原子を格納
                }
                print OUT $line;
        }
}
close(IN);

my $targetcon = 0;      #標的原子のループカウンタ
my $othercon = 0;               #比較対象原子のループカウンタ
my $str = "";   #出力用の接続する原子の原子順序番号
foreach(@atom){
        my $now = $_;   #標的原子
        my $x1 = $arrayx[$targetcon];  #標的原子の X 座標
        my $y1 = $arrayy[$targetcon];  #標的原子の Y 座標
        my $z1 = $arrayz[$targetcon];  #標的原子の Z 座標
        my $name1 = $atomname[$targetcon];      #標的原子の原子
        foreach(@atom){
                my $x2 = $arrayx[$othercon];    #比較対象原子の X 座標
                my $y2 = $arrayy[$othercon];    #比較対象原子の Y 座標
                my $z2 = $arrayz[$othercon];    #比較対象原子の Z 座標
                my $name2 = $atomname[$othercon];       #比較対象原子の原子
                my $bond_lengthistance = ((($x2 - $x1) ** 2)
                        + (($y2 - $y1) ** 2) + (($z2 - $z1) ** 2)) ** 0.5;
                        #距離を演算
                if($bond_lengthistance <= $bond_length){
                        #距離がこの範囲内だったら接続していると見なす
                        if(($option_ignore == 0) && (($name1 eq "H")
                        || ($name2 eq "H")) && ($bond_lengthistance > 1.20)){
                                #ただし，水素原子との距離は 1.20Åまでとする.
                                $othercon++;
                                next;
                        }
                        elsif($bond_lengthistance == 0){
                                        #0 距離は自身であるのでスキップ
```

```
                                $othercon++;
                                next;
                        }
                        my $line = &adjust(5, $_);
                        $str = $str . $line;    #接続原子を出力用に
                }
                $othercon++;
        }
        $othercon = 0;
        if($str !~ /^$/){
                $now = &adjust(5, $now);
                print OUT "CONECT$now$str¥n";   #出力
                $str = "";
        }
        $targetcon++;
}
print OUT "END¥n";
close(OUT);
}

# =============== adjust =============== #
sub adjust{
my $max = shift(@_);    # 調整後文字数
my $line = shift(@_);   # 文字列

my $space = length($line);      # 文字列の長さ
$space = $max - $space;         # 追加するスペース数
$space = " " x $space;                  # スペース
$line = $space . $line;         # スペース追加
return($line);
}

# =============== check_float =============== #
sub check_float{
my $value = shift(@_);

if($value !~ /^-?¥d+(¥.¥d+)?$/){
        print " ERROR: Invalid argument: distance_option¥n";
        exit;
}

return $value;
}

# =============== help =============== #
sub help{
print << "HELP";
 PDB 結合情報追記プログラム
  ¥$ crecone [オプション] 入力ファイル [出力ファイル]
    -F, --force      アミノ酸や DNA のような標準残基であっても計算
    -I, --ignore     水素の結合距離条件を無視 (デフォルトは 1.20 Å)
    -D, --distance   結合距離を指定（デフォルトは 1.9 Å）

HELP
}
```

一方、接続情報を削除するプログラム discone は入力ファイルと出力ファイルを指定するだけで良い。なお、出力ファイルは他のプログラム同様、指定しない場合、プログラムの途中でどのように処理するかを尋ねられる。

```
$ discone input.pdb [output.pdb]
```

以下に discone のソースコード (プログラム言語: Perl) を示す。

```
#! /usr/bin/env perl

use strict;
use File::Temp;

my $in = shift(@ARGV);
my $out = shift(@ARGV);
```

```
if(($in =~ /^--help$/i) || ($in =~ /^-H$/i)){
        &help;
        exit;
}

if($in !~ /^$/){
        if(-f $in){
                my $tmp = File::Temp->new(TEMPLATE => '.discone-XXXXXX');
                $SIG{'TERM'} = $SIG{'PIPE'} = $SIG{'HUP'} = $SIG{'INT'} = sub {
                        unlink $tmp;
                        exit;
                };      # プログラムが中断した場合の処理
                &work($in, $tmp);
                if($out =~ /^$/){
                        print " INFORMATION: 出力先が指定されていません. ¥n";
                        print " INFORMATION: 上書き(O), 別名で保存(S),
                                                何もしない(C)¥n > ";
                        my $user = <STDIN>;
                        $user =~ s/¥n//;
                        if($user =~ /^O$/i){
                                unlink $in;
                                rename $tmp, $in;
                        }
                        elsif($user =~ /^S$/i){
                                print " ファイル名: ";
                                $user = <STDIN>;
                                $user =~ s/¥n//;
                                rename $tmp, $user;
                        }
                        else{
                                unlink $tmp;
                                exit;
                        }
                }
                else{
                        rename $tmp, $out;
                }
        }
        else{
                print " ERROR: 指定された入力ファイルは存在しません. ¥n";
        }
}
else{
        print " ERROR: 入力ファイルを指定してください. ¥n";
}
exit;

# =============== <work> ===============
sub work{
my $in = shift(@_);
my $out = shift(@_);

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if(/^CONECT/){
                next;
        }
        s/¥r¥n/¥n/;
        print OUT;
}
close(IN);
close(OUT);
}

# =============== <help> ===============
sub help{
print " PDB 接続情報削除プログラム¥n";
print "  ¥$ discone 入力ファイル 出力ファイル¥n";
print "  ¥$ discone 入出力ファイル¥n";
}
```

附録 A-3: シリアル番号、残基番号調整プログラム

PDB ファイルを編集していると、原子のシリアル番号、及び残基番号が不規則になる。そして、PDB は仕様上、99999 原子、9999 残基までしか記述することができない。そのため、編集を完了したら、シリアル番号、及び残基数をリセットする必要がある。しかし、7～11 文字、及び 23～26 文字に対し、シリアル番号、及び残基数を規則正しくエディタで編集するのは困難である。そこで、これらの数値をリセットするプログラム prenum を作成した。

prenum で原子のシリアル番号をリセットする場合のコマンドを以下に示す。

```
$ prenum --atom input.pdb output.pdb [--start FIRST_NUM]
```

--atom は-A と指定することも可能である。--start は、シリアル番号の開始番号を指定する。--start を指定しない場合、シリアル番号の開始番号は 1 となる。これにより、ファイルに記述された順に、シリアル開始番号が割り振られていく。シリアル番号が 99999 を超えた場合、シリアル番号は読み込むプログラムの解釈に依存し、誤動作の原因となるため、100000 以降の原子のシリアル番号は「*****」と記述するようにした。

また、prenum で残基番号をリセットする場合のコマンドを以下に示す。

```
$ prenum --residue input.pdb output.pdb [--start FIRST_NUM]
```

--residue は-R と指定することも可能である。なお、--residue には残基数が 9999 を超えた場合の処理を指定する方法として、--residue1 (-R1 と指定可) と--residue2 (-R2 と指定可) の 2 通りの手法が存在する。--residue1 は 9999 を超えた残基番号を「****」とする方法である。一方、--residue2 は、残基番号が 9999 を超えると再び 1 から残基番号を振り始める。--residue を指定した場合、自動的に--residue2 が指定される。

以下に prenum のソースコード (プログラム言語: Perl) を示す。

```
#! /usr/bin/env perl

use strict;
use File::Temp;

my $mode = 0;
my $in = "";
my $out = "";
my $start = 1;
my $option = 0;
my $flag = 0;
foreach(@ARGV){
        if((/^--help$/i) || (/^-H$/i)){# ヘルプ
                $mode = 0;
                last;
        }
        elsif((/^--atom$/i) || (/^-A$/i)){      # 原子順序番号
                $mode = 1;
        }
        elsif((/^--residue$/i) || (/^-R$/i) || (/^--residue2$/i)
                || (/^-R$/i)){
                # 残基番号(最新法；デフォルト)
                $mode = 2;
        }
        elsif((/^--residue1$/i) || (/^-R1$/i)){         # 残基番号
                $mode = 3;
```

```
        }
        elsif((/^--clear$/i) || (/^-C$/i)){
                # 残基番号が後ろの領域を使っていた場合削除する
                $option = 1;
        }
        elsif((/^--start$/i) || (/^-S$/i)){     # 開始値
                $flag = 1;
        }
        elsif($flag == 1){      # 開始値
                if(/^-?¥d+$/){
                        $start = $_;
                }
                else{
                        print " ERROR: Start number must be integer";
                        exit;
                }
        }
        else{
                if($in eq ""){  # 入力ファイル
                        $in = $_;
                }
                else{                   # 出力ファイル
                        $out = $_;
                }
        }
}

if($mode == 0){ # ヘルプ(あるいは未定義オプションを指定した場合)
        &help;
}
else{
        &check_file($in);

        my $tmp = File::Temp->new(TEMPLATE => '.prenum-XXXXXX');
        $SIG{'TERM'} = $SIG{'PIPE'} = $SIG{'HUP'} = $SIG{'INT'} = sub {
                unlink $tmp;
                exit;
        };      # プログラムが中断した場合の処理

        if($mode == 1){         # 原子順序番号
                &atomorder($in, $tmp, $start);
        }
        elsif($mode == 2){              # 残基番号(最新法; デフォルト)
                &residueorder($in, $tmp, $start, 2, $option);
        }
        elsif($mode == 3){              # 残基番号(従来法)
                &residueorder($in, $tmp, $start, 1, $option);
        }
        &overwrite($in, $out, $tmp);
}
exit;

# =============== atomorder =============== #
# 原子順序番号リセット
sub atomorder{
my $in = shift(@_);     # 入力ファイル
my $out = shift(@_);    # 出力ファイル
my $start = shift(@_);  # 開始値

my $num = $start - 1;

open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^HETATM/i) || (/^ATOM/i)){
                $num ++;
                my $atomnum = "";
                if($num >= 100000){     # 原子順序番号が 10000 を超えたら，*に変換
                        $atomnum = "*****";
                }
                else{   # 通常の原子順序番号
                        $atomnum = $num;
                }
                my $line = &adjust(5, $atomnum);
                substr($_, 6, 5) = $line;
                print OUT;
```

```
        }
        elsif(/^TER/i){
                print OUT "TER¥n";
        }
        elsif(/^CONECT/i){
                print OUT;
        }
}
close(IN);
print OUT "END¥n";
close(OUT);
}

# =============== residueorder =============== #
# 残基番号リセット
sub residueorder{
my $in = shift(@_);             # 入力ファイル
my $out = shift(@_);            # 出力ファイル
my $start = shift(@_);          # 開始値
my $flag = shift(@_);           # 残基オーバーの挙動
my $option = shift(@_);# 残基番号の 4 桁目が後ろに伸びていた場合

my $num = $start - 1;
my $contain = 0;        # 残基を構成する原子数(1-3 個で構成されていたら Warning を出す)

my $beforeres = "";             # 前の残基名
open(IN, "$in");
open(OUT, "> $out");
while(<IN>){
        if((/^HETATM/i) || (/^ATOM/i)){
                my $line = $_;
                my $nowres = substr($line, 17, 12);
                my $amino_flag = &rec_amino($nowres);
                my $nowatom = substr($line, 12, 4);
                if(($beforeres ne $nowres) || (($amino_flag == 1)
                && ($nowatom eq " N  "))){
                # 前の残基と異なる場合，あるいはアミノ酸で atomtype が N の時に処理
                        if(($contain <= 2)
                                && (($beforeres !~ /^((HOH)|(SOL)|(WAT))/)
                                && ($beforeres !~ /^$/))){
                                # 最初と水分子を除き，数原子で残基を作っているものを警告
                                print " WARNING: Residue number changed
                                        with a few atoms¥n";
                                print "   Current residue number: $num¥n";
                                print "   Residue name: ¥"$beforeres¥"¥n¥n";
                        }
                        $num ++;
                        $beforeres = $nowres;
                        $contain = 0;
                }

                my $resnum = "";
                if($num >= 10000){      # 残基番号が 10000 を超えたら，*に変換(従来法)
                        if($flag == 1){
                                $resnum = "****";
                        }
                        elsif($flag == 2){      # 残基番号をリセットする
                                $resnum = 0;
                                $num = 0;
                        }
                }
                else{   # 通常の残基番号
                        $resnum = $num;
                }

                my $strnum = &adjust(4, $resnum);
                my $check = substr($line, 20, 6);
                if($check =~ /^[¥s¥D]{2}¥s*¥d+$/){      # 通常の場合
                        substr($line, 22, 4) = $strnum;
                }
                else{   # 残基番号がユニットの部分まで進出していた場合削除する
                        substr($line, 20, 6) = "  $strnum";
                }

                if($option == 1){       # 残基番号が後ろの領域を使っていた場合削除する
                        substr($line, 26, 4) = "    ";
```

```
                }

                print OUT $line;
                $contain ++;
        }
        elsif(/^TER/i){
                print OUT "TER¥n";
                $beforeres = "TER";
        }
        elsif(/^CONECT/i){
                print OUT;
        }
}
close(IN);
print OUT "END¥n";
close(OUT);
}

# ============== adjust ============== #
sub adjust{
my $max = shift(@_);    # 調整後文字数
my $line = shift(@_);   # 文字列

my $space = length($line);      # 文字列の長さ
$space = $max - $space;         # 追加するスペース数
$space = " " x $space;                  # スペース
$line = $space . $line;         # スペース追加
return($line);
}

# ============== overwrite ============== #
## 上書き確認
sub overwrite{
my $in = shift(@_);
my $out = shift(@_);
my $tmp = shift(@_);

if($out =~ /^$/){
        print " INFORMATION: 出力先が指定されていません. ¥n";
        print " INFORMATION: 上書き(O), 別名で保存(S), 何もしない(C)¥n > ";
        my $user = <STDIN>;
        $user =~ s/¥n//;
        if($user =~ /^O$/i){
                unlink $in;
                rename $tmp, $in;
        }
        elsif($user =~ /^S$/i){
                print " ファイル名: ";
                $user = <STDIN>;
                $user =~ s/¥n//;
                rename $tmp, $user;
        }
        else{
                unlink $tmp;
                exit;
        }
}
else{
        rename $tmp, $out;
}
}

# ============== rec_amino ============== #
# アミノ酸残基かを判定
sub rec_amino{
my $res = shift(@_);

my $flag = 0;
if($res =~ /(ALA)|(ARG)|(ASN)|(ASP)|(CYS)|(GLN)|(GLU)|(GLY)|(HIS)
        |(HIE)|(HID)|(HIP)|(ILE)|(LEU)|(LYS)|(MET)|(PRO)|(SER)|(THR)
        |(TRP)|(TYR)|(VAL)/i){
        $flag = 1;
}

return $flag;
}
```

```
# ================ check_file ================ #
# ファイルチェック
sub check_file{
my $file = shift(@_);

if(! -f $file){
        print " ERROR: No such file: $file¥n";
        exit;
}

return $file;
}

# ---------------- help ---------------- #
sub help{
print << "HELP";
 Re-numbering atom or residue order program for PDB
  ¥$ prenum <MODE> <INPUT> [OUTPUT] [--start INIT_NUMBER] [--clear]

  MODE
     -A, --atom       Re-numbering atom order
     -R, --residue     Re-numbering residue order(Newer method)
     -R1, --residue1   Re-numbering residue order(conventional method)
     -R2, --residue2   Re-numbering residue order(Newer method)

     * When digit number of residue order is over 4,
       residue order is overwritten with "****" at conventional method.
       Newer method is that residue order is overwritten with
       new number reset from 0 at above situation.

  OTHER OPTION
     -S, --start      Start with specified number
     -C, --clear      When residue number is displayed at 24-30 column,
                    correct to 23-26 (for moltools bug)

HELP
}
```

附録 A-4: 水分子数調整プログラム

結合エネルギーは、FMO 計算などで各部位の全体エネルギーを計算し、その差分により求めることができる。しかし、系に含まれる分子数が異なると全体エネルギーが変化し、最終的に結合エネルギーにも影響を与えてしまう。従って、結合エネルギーを比較する場合、最低でも比較する系の水分子を調整し、同じ水分子数にする必要がある。そこで、水分子数を調整するプログラム wtune を作成した。

以下に wtune のコマンドとそのオプションを示す。なお、オプションの赤字は必ず指定しなければならない項目を示している。

```
$ wtune <option> input.pdb [output.pdb]
```

option		機能
ロングオプション	ショートオプション	
--view	-V	ファイルに含まれる水分子数を表示
--measure	-M	各水分子の溶質からの距離を表示（サブオプションが必要）
--distance	-D	水分子数を溶質からの距離を指定
--number	-N	水分子数を数を指定
--thread	-T	実行時に使用するスレッド数を指定
--hydrogen	-Y	水分子の水素原子を考慮
--part	-P	溶質を指定

--view は、ファイルに含まれる水分子を端末上に表示する。これにより、端末上で水分子数を確認することができる。--measure は、各水分子の溶質 (水以外の原子・分子) からの距離を端末上に表示する。これには ”long” あるいは、”short” のサブオプションが必要である。wtune は、溶質原子と水分子のペアの距離を総当りで計算する。そのため、計算により溶質原子からの最長距離 (long)と最短距離 (short) が算出される。最短距離は溶質表面からの距離を示しており、後述する--distance や--number もこれを参考に水分子を調整している。--distance は溶質からの距離を指定することにより、その範囲内の水分子を残した PDB ファイルを出力する。一方、--number は、溶質から近い順に指定した水分子数を残した PDB ファイルを出力する。本研究では、--distance で大体の水分子数を決定し、--number により微調整することにより、FMO 計算に加える水分子を決定した。また、近年のコンピュータは 1 ノードあたり、複数の CPU を持っており、これを利用し並列化することにより、計算時間を短縮することができる。--thread は実行時に使用する 1 ノード内の CPU 数を指定し、実行する並列処理するオプションである。また、wtune では計算コストを抑えるため、水分子の水素原子の座標を無視して計算している。--hydrogen オプションを加えると、水分子の水素原子の座標も考慮して計算するため、厳密な計算をすることが可能となる。さらに、wtune は通常、溶質を水分子以外の原子・分子と定義しているが、--part オプションを指定することにより、その溶質を部分的に選択することが可能である。溶質は「L:PDB 内の行数」「AN:原子のシリアル番号」「RN:残基番号」の 3 つのサブオプションによって選択する。

以下に wtune のソースコード (プログラム言語: Perl)を示す。

```
#! /usr/bin/env perl

use strict;
use File::Temp;
use threads;
use File::Basename;

my $flag = 0;           # オプション振り分けフラグ
my $option = 0;         # 操作(モード)
my $value = 0;
my $thread = 1;         # スレッド数
my $range = 0;          # 距離ソートフラグ(最短/最長)
my $hydrogen = 1;       # 水素原子考慮フラグ
my $parttype = 0;       # 部分指定のタイプ (1: 行, 2: 原子順序番号, 3: 残基番号)
my $partpoint = "";     # 部分指定場所
my $in = "";            # 入力ファイル
my $out = "";           # 出力ファイル
foreach(@ARGV){
        if($flag == 1){ # thread
                $thread = &check_int($_);
                if($thread == 0){
                        print " ERROR: Invalid argument (Not 0)¥n";
                        exit;
                }
                $flag = 0;
        }
        elsif($flag == 2){              # view
                my $in = &check_file($_);
                &view($in);
                exit;
        }
        elsif($flag == 3){              # distance
                $value = &check_float($_);
                $flag = 0;
        }
        elsif($flag == 4){              # mount
```

```
                $value = &check_int($_);
                $flag = 0;
        }
        elsif($flag == 5){              # search
                if((/^s$/i) || (/^short$/i)){
                        $range = 0;
                }
                elsif((/^l$/i) || (/^long$/i)){
                        $range = 9;
                }
                else{
                        print " ERROR: Need to sub-option¥n";
                        exit;
                }
                $flag = 0;
        }
        elsif($flag == 6){
                if(/^(l|(AN)|(RN)):¥d+((,¥d+)|(-¥d+))*$/i){
                        ($parttype, $partpoint) = split(/:/, $_, 2);
                        if($parttype =~ /^l$/i){                # 行指定
                                $parttype = 1;
                        }
                        elsif($parttype =~ /^AN$/i){    # 原子順序番号指定
                                $parttype = 2;
                        }
                        elsif($parttype =~ /^RN$/i){    # 残基番号指定
                                $parttype = 3;
                        }
                }
                else{
                        print " WARNING: Invalid syntax; Ignored¥n";
                }
                $flag = 0;
        }
        elsif($flag == 10){     # output
                $out = $_;
                $flag = 0;
        }
        # * # * # * # * # * # * # * # * # * #
        elsif((/^--help$/i) || (/^-H$/i)){      # help
                &help;
                exit;
        }
        elsif((/^--thread$/i) || (/^-T$/i)){
                $flag = 1;
        }
        elsif((/^--view$/i) || (/^-V$/i)){
                $flag = 2;
        }
        elsif((/^--distance$/i) || (/^-D$/i)){
                $flag = 3;
                &check_option($option);
                $option = 3;
                $range = 0;
        }
        elsif((/^--number$/i) || (/^-N$/i)){
                $flag = 4;
                &check_option($option);
                $option = 4;
                $range = 0;
        }
        elsif((/^--measure$/i) || (/^-M$/i)){
                $flag = 5;
                &check_option($option);
                $option = 5;
        }
        elsif((/^--hydrogen$/i) || (/^-Y$/i)){
                $hydrogen = 3;
        }
        elsif((/^--part$/i) || (/^-P$/i)){
                $flag = 6;
        }
        else{   # input
                $in = &check_file($_);
                $flag = 10;
        }
```

```
}

if($in !~ /^$/){
        my  ($pro,  $wat,  $target)  =  &separate($in,  $parttype,  $partpoint);
        # 構造を水とタンパク質，対象に分解

        my @wats = &return_array($wat);
        my $wat_line = $#wats + 1;      # 行数取得
        undef @wats;
        $wat_line /= $thread;
        $wat_line =~ s/¥.¥d+$//;                # 1 ファイルあたりの行数
        my @wats = &thread_separator($thread, $wat_line, $wat);
        # 水分子を分割

        my @thread_datas = ();
        if($thread == 1){
                $thread_datas[0] =
                &discalc($hydrogen, $range, $target, $wats[0]);
                # "距離¥t 行数¥t 残基名"
        }
        else{
                # スレッドが指定されていた場合，スレッド処理
                for(my $i = 1; $i <= $thread; $i ++){  # スレッド作成
                        push(@thread_datas, "");
                        $thread_datas[$i - 1] =
                                threads -> new(¥&discalc, $hydrogen,
                                $range, $target, $wats[$i - 1]);
                        # "距離¥t 行数¥t 残基名"
                }
                for(my $i = 1; $i <= $thread; $i ++){  # スレッド待機
                        $thread_datas[$i - 1] = $thread_datas[$i - 1] -> join();
                }
        }

        my @datas = ();
        foreach(@thread_datas){ # スレッドデータ統合
                my @data_fragmented = &return_array($_);
                push(@datas, @data_fragmented);
        }
        undef @thread_datas;

        if($option == 5){               # search mode
                if($hydrogen == 1){
                        my @new_datas = ();
                        foreach(@datas){
                                my $line = $_;
                                my @lines = split(/¥t/, $_, 3);
                                # "距離¥t 行数¥t 残基名"
                                $lines[2] = substr($lines[2], 11, 1);
                                # 原子名取得
                                if($lines[2] ne "H"){
                                        push(@new_datas, $line);
                                }
                        }
                        @datas = @new_datas;
                        undef @new_datas;
                }
                if($range == 0){
                        @datas = sort {$a <=> $b} @datas;
                        # 距離でソート(昇順): 最短
                }
                elsif($range == 9){
                        @datas = sort {$a <=> $b} @datas;
                        # 距離でソート(降順): 最長
                }
                else{
                        print " ERROR: Subroutine error :: range is not correct¥n";
                        exit;
                }
                &measure(@datas);
                exit;
        }
        else{
                @datas = &determine_main(@datas);
                # "距離¥t 水分子順序リスト"
                @datas = sort {$a <=> $b} @datas;
```

```
                # 距離でソート

                my @new_datas = ();
                if($option == 3){       # 距離で処理
                        foreach(@datas){
                                my ($distance, $water_order) = split(/¥t+/, $_);
                                # 距離と水分子順序リストに分解
                                if($distance <= $value){
                                        # 距離以下の場合
                                        my @lines = split(/,/, $water_order);
                                        # 水分子順序リストを分解
                                        push(@new_datas, @lines);
                                }
                        }
                }
                elsif($option == 4){    # 数で処理
                        for(my $i = 0; $i < $value; $i ++){
                                # 指定された個数だけ処理
                                my ($distance, $water_order) =
                                        split(/¥t+/, $datas[$i]);
                                # 距離と水分子順序リストに分解
                                my @lines = split(/,/, $water_order);
                                # 水分子順序リストに分解
                                push(@new_datas, @lines);
                        }
                }
                @datas = @new_datas;
                undef @new_datas;
                @datas = sort {$a <=> $b} @datas;

                my $tmp = File::Temp->new(TEMPLATE => '.wtune-XXXXXX');
                $SIG{'TERM'} = $SIG{'PIPE'} = $SIG{'HUP'} = $SIG{'INT'} = sub {
                        unlink $tmp;
                        exit;
                };      # プログラムが中断した場合の処理

                &create($pro, $wat, $tmp, @datas);
                if($out =~ /^$/){       # 出力先が指定されていない場合
                        print " INFORMATION: 出力先が指定されていません. ¥n";
                        print " INFORMATION: 上書き(O), 別名で保存(S),
                                何もしない(C)¥n > ";
                        my $user = <STDIN>;
                        $user =~ s/¥n//;
                        if($user =~ /^O$/i){
                                unlink $in;
                                rename $tmp, $in;
                        }
                        elsif($user =~ /^S$/i){
                                print " ファイル名: ";
                                $user = <STDIN>;
                                $user =~ s/¥n//;
                                rename $tmp, $user;
                        }
                        else{
                                exit;
                        }
                }
                else{
                        rename $tmp, $out;
                }
        }
}
else{
        print " ERROR: No input file¥n";
        exit;
}
exit;

# =============== view =============== #
## PDB のステータス表示
sub view{
my $in = $_[0];

my $water = 0;
my $atom = 0;
my $wateratom = 0;
```

```
open(IN, "$in");
while(<IN>){
        if(((/^atom/i) || (/^hetatm/i)){
                $atom++;        # 総原子数
                my $line1 = substr($_, 11, 9);
                my $line2 = substr($_, 17, 3);
                if(($line2 =~ /HOH/) || ($line2 =~ /WAT/)
                        || ($line2 =~ /SOL/)){ # 水原子数
                        $wateratom++;
                }
                if(($line1 =~ /¥s+O¥s+HOH/) || ($line1 =~ /¥s+O¥s+WAT/)
                        || ($line1 =~ /¥s+OW¥s+SOL/)){ # 水分子数
                        $water++;
                }
        }
}
close(IN);
$wateratom = $atom - $wateratom;        # 他の原子数算出
print " $in¥n";
print "  水分子数: $water¥n";
print "  他の原子: $wateratom¥n";
print "  総原子数: $atom¥n";
}

# =============== separate =============== #
## 水とタンパク質に分割
sub separate{
my $in = shift(@_);                     # 元 PDB
my $parttype = shift(@_);               # 部分指定のタイプ
my $partpoint = shift(@_);      # 部分指定場所

my @partpoints = ();
if($parttype != 0){
        @partpoints = &range($partpoint);
        $partpoint = "";
}

my @pros = ();
my @wats = ();
my @targets = ();
my $wat_count = 0;
my $beforeline = "";
open(IN, "$in");
while(<IN>){
        s/¥n//;
        if(((/^HETATM/i) || (/^ATOM/i)){
                if($parttype == 2){             # 原子順序番号指定
                        $partpoint = substr($_, 6, 5);
                        $partpoint =~ s/¥s+//g;
                }
                elsif($parttype == 3){ # 残基番号指定
                        $partpoint = substr($_, 22, 4);
                        $partpoint =~ s/¥s+//g;
                }

                my $data = substr($_, 17, 3);
                if(($data =~ /^HOH$/i) || ($data =~ /^WAT$/)
                        || ($data =~ /^SOL$/)){# 水分子
                        my $line_num = &adjust(10, $wat_count);
                        push(@wats, "$line_num¥t$_");
                        $wat_count ++;
                }
                else{   # その他の分子
                        push(@pros, $_);

                        if(($parttype == 1) && (grep(/^$.$/, @partpoints))){
                                push(@targets, $_);
                        }
                        elsif((($parttype == 2) || ($parttype == 3))
                                && (grep(/^$partpoint$/, @partpoints))){
                                push(@targets, $_);
                        }
                }
        }
        elsif(/^TER/){  # TER は PRO に記述
                if($beforeline ne $ ){  # TER が連続しないようにする
```

```
                        push(@pros, "TER");
                }
        }
        $beforeline = $_;
}
close(IN);

my $pro = join("¥n", @pros);
my $wat = join("¥n", @wats);
my $target = "";
if($parttype == 0){
        $target = $pro;
}
else{
        $target = join("¥n", @targets);
}

return $pro, $wat, $target;
}

# =============== discalc =============== #
## 距離を計算し，距離を出力
sub discalc{
my $option1 = shift(@_);
# 水分子の扱い(1: 酸素原子で判断; 3: 水素原子も含めて判断)
my $option2 = shift(@_);        # 距離の扱い(0: 最短距離; 9: 最長距離)
my $pro = shift(@_);            # 水以外の分子の PDB（入力）
my $wat = shift(@_);            # 水分子の PDB（入力）

my @pros = &return_array($pro);
my @aminox = ();        # 水以外の分子の x 座標
my @aminoy = ();        # 水以外の分子の y 座標
my @aminoz = ();        # 水以外の分子の z 座標
foreach(@pros){
        if((/^ATOM/i) || (/^HETATM/i)){
                s/¥n//;
                my $prox = substr($_, 30, 8);
                my $proy = substr($_, 38, 8);
                my $proz = substr($_, 46, 8);
                push(@aminox, $prox);   # その他の原子 x 座標取得
                push(@aminoy, $proy);   # その他の原子 y 座標取得
                push(@aminoz, $proz);   # その他の原子 z 座標取得
        }
}
undef $pro, @pros;

my @wats = &return_array($wat);
my @results = ();               # 最終的な距離と水分子情報
my @distances = ();             # 距離
my @hydrogens = ();
# 水素原子を扱う場合に水分子の代表値となるものを決定するために必要な配列
my @hydrogen_lines = ();        # 水素原子情報
my $flag = 0;                   # 最初
my $before_residue = "";        # 前の残基名
my $before_distance = 0;        # 水素原子のための前の距離保存用
foreach(@wats){
        my ($line_num, $line) = split(/¥t+/, $_, 2);    # 水分子順序番号と分割
        my $atom = substr($line, 12, 2);
        my $atom_name = substr($line, 12, 4);
        $atom =~ s/¥d+//g;
        $atom =~ s/¥s//g;
        if(($option1 == 1) && ($atom eq "H")){
                # 酸素原子のみ扱う場合，水素原子はお断り
                $line = substr($line, 17, 9);   # 残基データ取得
                push(@hydrogen_lines, "$line_num¥t$line¥t$atom_name");
                next;
        }
        my $watx = substr($line, 30, 8);
        my $waty = substr($line, 38, 8);
        my $watz = substr($line, 46, 8);
        $line = substr($line, 17, 9);   # 残基データ取得
        my $count = 0;
        foreach(@aminox){
                my $ax = $_;
                # 水以外の分子の x 座標
                my $ay = $aminoy[$count];               # 水以外の分子の y 座標
```

```
                my $az = $aminoz[$count];               # 水以外の分子の z 座標
                my $calcdis = sqrt(($watx - $ax) ** 2 + ($waty - $ay) ** 2
                        + ($watz - $az) ** 2); # 距離の演算
                push(@distances, $calcdis);
                $count ++;
        }
        @distances = sort {$a <=> $b} @distances;
        # 最短/最長距離算出
        if($option2 == 0){
                if(($option1 == 1) && ($#hydrogen_lines != -1)){
                        foreach(@hydrogen_lines){
                                push(@results, "$before_distance¥t$_");
                        }
                        @hydrogen_lines = ();
                }
                push(@results, "$distances[0]¥t$line_num¥t$line¥t$atom_name");
                # "距離¥t 行数¥t 残基名"
                $before_distance = $distances[0];
                # 水素原子は酸素原子のあとなので
        }
        elsif($option2 == 9){
                if(($option1 == 1) && ($#hydrogen_lines != -1)){
                        foreach(@hydrogen_lines){
                                push(@results, "$before_distance¥t$_");
                        }
                        @hydrogen_lines = ();
                }
                push(@results,
                "$distances[$#distances]¥t$line_num¥t$line¥t$atom_name");
                # "距離¥t 行数¥t 残基名"
                $before_distance = $distances[$#distances];
                # 水素原子は酸素原子のあとなので
        }
        else{
                print " ERROR: Subroutine error: discalc - option2¥n";
                exit;
        }
        @distances = ();
}

if($#hydrogen_lines != -1){     # 後処理
        foreach(@hydrogen_lines){
                push(@results, "$before_distance¥t$_");
        }
}

my $result = join("¥n", @results);
return $result;
}

# =============== create =============== #
## リストを元に PDB を作成する
sub create{
my $pro = shift(@_);    # タンパク質のデータ
my $wat = shift(@_);    # 水分子のデータ
my $out = shift(@_);    # 出力する PDB ファイル（出力）
my @lists = @_;                 # 残す水分子リスト(行数-1)

my @out = &return_array($pro);
undef $pro;

my @wats = &return_array($wat);
undef $wat;

my $wat_count = 0;
my $now_wat = shift(@lists);
if($#lists != -1){              # 水分子が存在するときのみ実行
        foreach(@wats){
                if($wat_count == $now_wat){     # リストと一致した場合
                        my ($line_num, $residue) = split(/¥t/, $_, 2);
                        # 行数を除去するため分解
                        push(@out, $residue);
                        $now_wat = shift(@lists);       # 一致対象を更新
                }
                $wat_count ++;
        }
```

```
}

my $beforedata = "";
my @new_out = ();
foreach(@out){  # 重複するデータがないようにする
        if($beforedata ne $_){
                push(@new_out, $_);
                $beforedata = $_;
        }
}
push(@new_out, "END¥n");
@out = @new_out;
undef @new_out;

my $file = join("¥n", @out);
open(OUT, "> $out");
print OUT $file;
close(OUT);
}

# =============== search =============== #
## 距離を計算．タンパク質からの距離を表示.
sub measure{
my @datas = @_;                 # データ

foreach(@datas){
        my @lines = split(/¥t+/, $_);
        $lines[0] = &nearest($lines[0], 2);
        $lines[0] = &adjust(10, $lines[0]);
        print "$lines[0]¥t$lines[2] $lines[3]¥n";
}
}

# =============== nearest =============== #
# 四捨五入
sub nearest{
my $data = shift(@_);   # 値
my $round = shift(@_);  # 小数点以下の桁数

if($round > 0){ # 桁数が設定されている場合
        $data = $data * 10 ** $round;
        # 出力すべき値を整数部にする．そして小数点以下切り捨て.
        if($data > 0){  # 繰り上げ・繰り下げの判定(正負の値により変わる)
                $data += 0.5;
        }
        else{
                $data -= 0.5;
        }
        if($data !~ /^-?¥d+¥.¥d+$/){
                $data = $data . ".0";
        }
        $data = int($data);     # 小数点以下切り捨て
        $data = $data / (10 ** $round);# 桁を元に戻す
        if($data !~ /¥.¥d+/){   # 結果が整数のみ(小数点がない場合)の場合
                my $zero = "";
                for(my $i = 1; $i <= $round; $i ++){
                        # 小数点以下に付く 0 を用意する
                        $zero = "0$zero";
                }
                $data = "$data.$zero";  # 結果に 0 を付加する
        }
        else{   # 小数点はあるもの(最後の桁が 0 で切り捨てられている場合がある)
                my $length = $&;        # 小数点以下のデータを取得
                $length = length($length) - 1; # 小数点以下何桁あるか調べる
                if($length != $round){  # 指定した桁数ない場合
                        my $addzero = $round - $length;
                        my $zero = "";
                        for(my $i = 1; $i <= $addzero; $i ++){
                        # 小数点以下に付く 0 を用意する
                                $zero = "0$zero";
                        }
                        $data = "$data$zero";   # 結果に 0 を付加する
                }
        }
}
```

```
return $data;
}

# ============== adjust ============== #
sub adjust{
my $max = shift(@_);    # 調整後文字数
my $line = shift(@_);   # 文字列

my $space = length($line);      # 文字列の長さ
$space = $max - $space;         # 追加するスペース数
$space = " " x $space;                  # スペース
$line = $space . $line;         # スペース追加
return($line);
}

# ============== check_file ============== #
# ファイルチェック
sub check_file{
my $file = shift(@_);

if(! -f $file){
        print " ERROR: No such file: $file¥n";
        exit;
}

return $file;
}

# ============== check_float ============== #
# 浮動小数チェック
sub check_float{
my $value = shift(@_);

if($value !~ /^-?¥d+(¥.¥d+)?$/){
        print " ERROR: Invalid argument: $value¥n";
        exit;
}

return $value;
}

# ============== check_int ============== #
# 整数チェック
sub check_int{
my $value = shift(@_);

if($value !~ /^-?¥d+$/){
        print " ERROR: Invalid argument: $value¥n";
        exit;
}

return $value;
}

# ============== check_option ============== #
# オプションをチェック
sub check_option{
my $value = shift(@_);

if($value != 0){
        print " ERROR: Option is conflict¥n";
        exit;
}

return $value;
}

# ============== thread_separator ============== #
# スレッド用データ生成
 sub thread_separator{
my $thread = shift(@_);# 生成するデータ数
my $line_max = shift(@_);       # ファイルの行数(分割後)
my $data = shift(@_);   # 元データ

my @datas = &return_array($data);
```

```
my @results = ();
my @result_tmps = ();
my $line_num = 1;
my $count = 1;
my $before_atom = "";
foreach(@datas){
        my $line = $_;
        my @lines = split(/¥t/, $line, 2);      # 行を分割
        my $now_atom = substr($lines[1], 12, 2);        # 原子名取得
        $now_atom =~ s/^¥d//;
        $now_atom =~ s/¥s+//g;
        push(@result_tmps, $line);
        if(($line_max <= $line_num) && ($count < $thread)
                && (($now_atom eq "H") && ($before_atom eq "H"))){
                # ファイルが指定行数を超え，次のファイルが存在する場合
                my $result = join("¥n", @result_tmps);
                push(@results, $result);
                @result_tmps = ();
                $line_num = 0;
                $count ++;
        }
        $before_atom = $now_atom;
        $line_num ++;
}
my $result = join("¥n", @result_tmps);
push(@results, $result);
return @results;
}

# =============== return_array =============== #
# 文字列を配列に戻す
sub return_array{
my $str = shift(@_);

my @array = split(/¥n/, $str);

return @array;
}

# =============== determine_main =============== #
# 残基データから代表値を決定
sub determine_main{
my @datas = @_;

my @distances = ();
my @line_nums = ();
my @results = ();
my $before = "";
foreach(@datas){
        my @lines = split(/¥t+/, $_);  # "距離¥t 行数¥t 残基情報"
        if($before ne $lines[2]){
                if($before ne ""){
                        @distances = sort {$a <=> $b} @distances;
                        my $line_num = join(",", @line_nums);
                        $line_num =~ s/¥s+//g;
                        push(@results, "$distances[0]¥t$line_num");
                        # "距離¥t 行数"
                }
                $before = $lines[2];
                @distances = ();
                @line_nums = ();
        }
        push(@distances, $lines[0]);
        push(@line_nums, $lines[1]);
}

if($#distances != -1){
        @distances = sort {$a <=> $b} @distances;
        my $line_num = join(",", @line_nums);
        $line_num =~ s/¥s+//g;
        push(@results, "$distances[0]¥t$line_num");    # "距離¥t 行数"
}

return @results;
}
```

```
# =============== range =============== #
sub range{
my $numarray = shift(@_);               # 削除原子群情報

my @numarray1 = split(/,/, $numarray);
my @numarray2 = ();             # 範囲指定情報も含めた最終的情報を格納する
my $count = 0;                          # numarray2 の配列番号
foreach(@numarray1){
        if(/-/){        # x-y 等の範囲指定を認識
                my @wide = split(/-/, $_);      # 開始と終了に分ける
                if($wide[0] =~ /^$/){   # 開始がない場合
                        $wide[0] = 1;
                }
                elsif($wide[1] =~ /^$/){        # 終了がない場合
                        $numarray2[$count++] = $wide[0];
                        $numarray2[$count++] = -1;
                        last;
                }
                my $i = 0;
                for($i = $wide[0]; $i <= $wide[1]; $i++){
                        # 範囲をすべて数字にする
                        $numarray2[$count++] = $i;
                }
        }
        else{
                $numarray2[$count++] = $_;
        }
}
return(@numarray2);
}

# =============== help =============== #
sub help{
my $script_name = basename($0);
print << "HELP";
 Program of controlling water molecule

  ¥$ $script_name OPTIONS [SUB_OPTION] INPUT [OUTPUT]

  OPTIONS
   -V, --view     Show the number of water molecules and all atoms
   -N, --number    Control the number of water molecule (Need the number)
   -D, --distance  Control the distance of water molecule (Need the distance)
   -M, --measure   Measurement distance between solute and water molecule
                  (Need sub-option)
      s, short    Calculate shortest distance (in general)
      l, long     Calculate longest distance
   -T, --thread    Calculate in parallel (Need the number of thread)
   -Y, --hydrogen  Also calculate hydrogen (Default: Oxygen only)
   -P, --part      Target as specified area (Need sub-option and range)
      L:<RANGE>   Specify as line number
      AN:<RANGE>  Specify as atom order
      RN:<RANGE>  Specify as residue number
   -H, --help      Show help

HELP
}
```

附録 A-5: アミノ酸末端付加プログラム

FMO 計算はその仕様上、末端の電荷が直前、あるいは直後の残基 (フラグメント) に移動させて電子状態を計算する。また、PDB はデフォルトでは、タンパク質の N 末端、C 末端をそれぞれ NH_3^+、COO^-として扱うため、FMO 計算と PDB の仕様により、末端の電子状態が過大、あるいは過小評価されてしまう。そのため、タンパク質の N 末端、及び C 末端にそれぞれアセチル基、及びメチルアミン基を付加する必要がある。

アセチル基は、N 末端残基の Cα原子から N 原子へのベクトル分を、N 原子から伸ば

した位置に ACE 残基の原子タイプ C の原子を配置し、AMBER や Gromacs、OpenBabel でその他の原子を付加することにより作成することができる。また、メチルアミン基は、C 末端残基の C 原子から O 原子へのベクトル分を O 原子に加えた位置に NME 残基の原子タイプ N の原子を配置し、アセチル基同様、AMBER などで他の原子を付加することにより作成できる。本研究では、末端にアセチル基やメチルアミン基の基準となる原子を配置するプログラム aminocap を作成した。

aminocap のコマンドを以下に示す。

```
$ aminocap input.pdb <terminal_serial> <terminal_type>
[output.pdb]
```

ここで、terminal_serial は、処理したい末端残基を構成する原子のシリアル番号をハイフンで繋いだ範囲で指定する。terminal_type は、その末端残基に対し、付加したい官能基を指定する。本プログラムでは、アセチル基 (ACE)、メチルアミン基 (NME)、アミン基 (NHH) が指定可能である。出力ファイル output.pdb は省略可能であり、省略した場合、途中で出力をどのようにするか尋ねられる。

以下に aminocap のソースコード (プログラム言語: Perl)を示す。

```
#! /usr/bin/env perl

use strict;
use File::Basename;
use File::Temp;

my $in = shift(@ARGV);          # 入力ファイル
my $range = shift(@ARGV);       # 原子順序番号範囲
my $type = shift(@ARGV);        # キャップタイプ
my $out = shift(@ARGV);# 出力ファイル

if(($in =~ /^--help$/i) || ($in =~ /^-H$/i)){  # ヘルプ
        &help;
}
elsif(-f $in){  # 入力ファイルのチェック
        if($range =~ /^¥d+-¥d+$/){
        # 原子順序番号範囲をチェック
                if(($type =~ /^NME$/i) || ($type =~ /^NHH$/i)
                        || ($type =~ /^ACE$/i)){
                        # キャップタイプのチェック
                        my $tmp = File::Temp->new(TEMPLATE => '.aminocap-XXXXXX');
                        $SIG{'TERM'} = $SIG{'PIPE'} =
                                $SIG{'HUP'} = $SIG{'INT'} = sub {
                                unlink $tmp;
                                exit;
                        };      # プログラムが中断した場合の処理
                        my @atoms = &range($range);     # 原子順序番号を配列に変換
                        my $status = &work($in, $tmp, $type, @atoms);  # 処理
                        if($status == 0){
                                # 指定した原子順序番号が PDB 内に存在しない場合
                                unlink $tmp;
                                print " WARNING: Not modified by this program.¥n";
                        }
                        else{
                                if($out =~ /^$/){
                                        print " INFORMATION: No output name.¥n";
                                        print " INFORMATION: Overwrite(O),
                                                Save as(S), Cancel(C)¥n > ";
                                        my $user = <STDIN>;
                                        $user =~ s/¥n//;
                                        if($user =~ /^O$/i){
                                                unlink $in;
                                                rename $tmp, $in;
                                        }
```

```
                                        elsif($user =~ /^S$/i){
                                                print " File name: ";
                                                $user = <STDIN>;
                                                $user =~ s/¥n//;
                                                rename $tmp, $user;
                                        }
                                        else{
                                                exit;
                                        }
                                }
                                else{
                                        rename $tmp, $out;
                                }
                        }
                }
                else{
                        print " ERROR: Invalid argument: cap type.¥n";
                }
        }
        else{
                print " ERROR: Invalid argument: range.¥n";
        }
}
else{
        print " ERROR: No such file or directory: $in¥n";
}
exit;

# =============== work =============== #
sub work{
my $in = shift(@_);     # 入力ファイル
my $out = shift(@_);    # 出力ファイル
my $type = shift(@_);   # キャップタイプ
my @atoms = @_;

my $force_write = 0;    # キャップ対象への処理が終了したフラグ
my $search = 0;         # 指定した原子順序番号が存在するかフラグ
my $last = 0;                   # 範囲の最後が指定されていないフラグ
my @target_atoms = ();  # 対象原子群情報を格納
open(OUT, "> $out");
open(IN, $in);
while(<IN>){
        my $line = $_;          # バックアップ
        if($force_write == 1){  # 指定した末端を処理し終えた場合，ひたすら出力
                print OUT;
        }
        elsif((/^ATOM/) || (/^HETATM/)){        # 原子の場合
                my $order = substr($line, 6, 5);        # 原子順序番号を取得
                my $flag = 0;           # 指定した原子順序番号と一致したかフラグ
                my $i = 0;                      # 配列から要素を削除するためのカウンタ
                if($last == 1){
                        $flag = 1;
                }
                else{
                        foreach(@atoms){
                                # 指定した原子順序番号に対し，総当たりで検索
                                if($order == $_){       # 一致した場合
                                        $flag = 1;      # フラグ ON
                                        splice(@atoms, $i, 1);
                                        # 該当した原子順序を削除
                                        last;   # ループ終了
                                }
                                elsif($_ == -1){
                                # 範囲に -1 が含まれていた場合，PDB の最後までが対象
                                        $last = 1;
                                }
                                $i ++;  # 配列番号カウンタプラス
                        }
                }
                if($flag == 1){ # 一致している
                        push(@target_atoms, $line);     # 配列に格納
                        $search = 1;    # PDB 内に指定した原子順序番号が存在
                }
                if(($search == 1) && ($#atoms == -1)){
                # 指定した原子順序番号が存在し，かつ指定した原子順序番号がなくなった場合
                        @target_atoms = &cap($type, @target_atoms);
```

```
                        # キャップの計算
                        foreach(@target_atoms){# 算出した末端情報出力
                                print OUT;
                        }
                        $force_write = 1;       # 末端処理終了
                }
                elsif($search != 1){
                        print OUT;
                }
        }
        else{
                if(($search == 1) && ($#atoms == -1)){
                # 指定した原子順序番号が存在し，かつ指定した原子順序番号がなくなった場合
                        @target_atoms = &cap($type, @target_atoms);
                        # キャップの計算
                        foreach(@target_atoms){# 算出した末端情報出力
                                print OUT;
                        }
                        $force_write = 1;       # 末端処理終了
                }
                print OUT;      # TER や END 等出力
        }
}
close(IN);
close(OUT);

return $search;
}

# =============== cap =============== #
# キャップ算出
sub cap{
my $type = shift(@_);   # キャップタイプ
my @datas = @_;                 # 末端情報

my $cap_atom_num = -5000;       # 末端の原子順序番号
my $cap_res_num = -5000;        # 末端の残基順序番号
my @ca_datas = ();      # 末端の CA の情報
my @c_datas = ();       # 末端の C の情報
my @sec_datas = ();     # 末端の算出に必要な情報(C 末端→ N，N 末端→ O)
foreach(@datas){        # データ読み込み
        if(($type =~ /^ACE/i) && ($cap_atom_num == -5000)){
                # 末端の原子，残基番号取得(ACE)
                $cap_atom_num = substr($_, 6, 5);
                $cap_res_num = substr($_, 22, 4);
                $cap_res_num =~ s/¥D//g;
        }
        elsif($type =~ /^N/){           # 末端の原子，残基番号取得(NME，NHH)
                $cap_atom_num = substr($_, 6, 5);
                $cap_res_num = substr($_, 22, 4);
                $cap_res_num =~ s/¥D//g;
        }
        my $atom = substr($_, 13, 2);           # 原子名取得
        if($atom eq "CA"){              # CA の場合
                my $x = substr($_, 30, 8);
                my $y = substr($_, 38, 8);
                my $z = substr($_, 46, 8);
                push(@ca_datas, $x);
                push(@ca_datas, $y);
                push(@ca_datas, $z);
        }
        elsif($atom eq "C "){   # C の場合
                my $x = substr($_, 30, 8);
                my $y = substr($_, 38, 8);
                my $z = substr($_, 46, 8);
                push(@c_datas, $x);
                push(@c_datas, $y);
                push(@c_datas, $z);
        }
        elsif(($type =~ /ACE/i) && ($atom eq "N ")){    # N の場合(ACE)
                my $x = substr($_, 30, 8);
                my $y = substr($_, 38, 8);
                my $z = substr($_, 46, 8);
                push(@sec_datas, $x);
                push(@sec_datas, $y);
                push(@sec_datas, $z);
```

```
        }
        elsif(($type =~ /NME/i) || ($type =~ /NHH/i)) && ($atom eq "O ")){
        # O の場合(NME, NHH)
                my $x = substr($_, 30, 8);
                my $y = substr($_, 38, 8);
                my $z = substr($_, 46, 8);
                push(@sec_datas, $x);
                push(@sec_datas, $y);
                push(@sec_datas, $z);
        }
}

my $new_atom = "";              # キャップの情報
if($type =~ /ACE/i){    # ACE の場合
        my @new_add_atoms = (0, 0, 0);
        for(my $i = 0; $i <= 2; $i ++){# X, Y, Z に適用
                my $distance = $sec_datas[$i] - $ca_datas[$i];
                # CA と N の距離
                $new_add_atoms[$i] = $sec_datas[$i] + $distance;
                # N に距離を追加
                $new_add_atoms[$i] = &keta(3, $new_add_atoms[$i]);
                # 桁を合わせる
                $new_add_atoms[$i] = &adjust(8, $new_add_atoms[$i]);
                # 文字列調整
        }
        $new_atom = join("", @new_add_atoms);
        $cap_atom_num --;       # キャップ原子順序番号調整
        $cap_atom_num = &adjust(5, $cap_atom_num);
        $cap_res_num --;        # キャップ残基番号調整
        $cap_res_num = &adjust(3, $cap_res_num);
        $new_atom = "ATOM  $cap_atom_num  C   ACE   $cap_res_num    $new_atom¥n";
        unshift(@datas, $new_atom);     # 末端の前にキャップ情報追加
}
elsif($type =~ /^N/i){
        my @new_add_atoms = (0, 0, 0);
        for(my $i = 0; $i <= 2; $i ++){
                my $distance = ($c_datas[$i] - $ca_datas[$i]) / 2;
                # CA と C の距離の半分
                $new_add_atoms[$i] = $c_datas[$i] + $distance;
                # C に距離の半分を追加(仮想原子の座標)
                $distance = $new_add_atoms[$i] - $sec_datas[$i];
                # 仮想原子と O の距離
                $new_add_atoms[$i] = $new_add_atoms[$i] + $distance;
                # 仮想原子に距離を追加
                $new_add_atoms[$i] = &keta(3, $new_add_atoms[$i]);
                # 桁を合わせる
                $new_add_atoms[$i] = &adjust(8, $new_add_atoms[$i]);
                # 文字列調整
        }
        $new_atom = join("", @new_add_atoms);
        $cap_atom_num ++;       # キャップ原子順序番号調整
        $cap_atom_num = &adjust(5, $cap_atom_num);
        $cap_res_num ++;        # キャップ残基番号調整
        $cap_res_num = &adjust(3, $cap_res_num);
        if($type =~ /^NME$/i){
                $new_atom =
                "ATOM  $cap_atom_num  N   NME   $cap_res_num    $new_atom¥n";
        }
        elsif($type =~ /^NHH$/i){
                $new_atom =
                "ATOM  $cap_atom_num  N   NHH   $cap_res_num    $new_atom¥n";
        }
        else{
                print " ERROR: Subroutine error: cap(1)¥n";
                exit;
        }
        push(@datas, $new_atom);        # 末端の後にキャップ情報追加
}
else{
        print " ERROR: Subroutine error: cap(2)¥n";
        exit;
}

return @datas;
}
```

```
# =============== range =============== #
sub range{
my $numarray = shift(@_);               # 削除原子群情報

my @numarray1 = split(/,/, $numarray);
my @numarray2 = ();             # 範囲指定情報も含めた最終的情報を格納する
my $count = 0;                          # numarray2 の配列番号
foreach(@numarray1){
        if(/-/){        # x-y 等の範囲指定を認識
                my @wide = split(/-/, $_);      # 開始と終了に分ける
                if($wide[0] =~ /^$/){   # 開始がない場合
                        $wide[0] = 1;
                }
                elsif($wide[1] =~ /^$/){        # 終了がない場合
                        $numarray2[$count++] = $wide[0];
                        $numarray2[$count++] = -1;
                        last;
                }
                my $i = 0;
                for($i = $wide[0]; $i <= $wide[1]; $i++){
                        # 範囲をすべて数字にする
                        $numarray2[$count++] = $i;
                }
        }
        else{
                $numarray2[$count++] = $_;
        }
}
return(@numarray2);
}

# =============== adjust =============== #
sub adjust{
my $max = shift(@_);    # 調整後文字数
my $line = shift(@_);   # 文字列

my $space = length($line);      # 文字列の長さ
$space = $max - $space;         # 追加するスペース数
$space = " " x $space;                  # スペース
$line = $space . $line;         # スペース追加
return($line);
}

# =============== keta =============== #
# 桁合わせ
sub keta{
my $keta = shift(@_);           # 小数点以下の桁
my $value = shift(@_);          # 数値

my $tmp = $keta + 1;    # 桁以上の場合の
foreach($keta, $value){# 与えられた引数のチェック
        if($_ !~ /^-?¥d+(¥.¥d+)?$/){
                print " ERROR: Subroutine error: keta¥n";
                exit;
        }
}

if($value !~ /¥./){     # 数値が整数の場合，桁数分 0 を追加
        my $zero = "";
        for(my $i = 1; $i <= $keta; $i ++){
                $zero .= "0";
        }
        $value = $value . "." . $zero;
}
elsif($value =~ /^-?¥d+¥.¥d{$tmp,}$/){
        # 小数点以下の桁が指定されたものより大きい場合，四捨五入
        $value = $value * 10 ** $keta;
        $value = int($value + 0.5) / 10 ** $keta;
}
elsif($value !~ /^-?¥d+¥.¥d{$keta}$/){
        # 小数点以下の桁が指定されたものより小さい場合，0 を追加
        my $tmp = $value;
        $tmp =~ s/-?¥d+¥.//;
        $tmp = length($tmp);
        $tmp = $keta - $tmp;
        my $zero = "";
```

```
        for(my $i = 1; $i <= $tmp; $i ++){
                $zero .= "0";
        }
        $value .= $zero;
}

return $value;
}

# =============== help =============== #
sub help{
my $name = basename($0);
print << "HELP";
 Amino acid terminal neutralization program
  ¥$ $name INPUT_FILE ATOM_ORDER CAP_TYPE [OUTPUT_FILE]
    INPUT_FILE  PDB input file
    ATOM_ORDER  Atom order range of N-terminal or C-terminal
    CAP_TYPE    Terminal type (ACE, NME, NHH)
    OUTPUT_FILE PDB output file (if not appointed, input file is overwritten.)

HELP
}
```

附録 B: 本論文の基礎となる研究成果

附録 B-1: 学術雑誌に発表した論文

第一著者として、以下の論文を学術雑誌に発表した。

"Specific interactions between lactose repressor protein and DNA affected by ligand binding: *ab initio* molecular orbital calculations", **T. Ohyama**, M. Hayakawa, S. Nishikawa, N. Kurita, *Journa of Computational Chemistry* **32**, 1661-1670 (2011).

以下に示す論文を作成中であり、国際学術雑誌への投稿を準備している。

"Analysis of the effects of ligand-binding on conformational change for lactose repressor using molecular dynamics", **T. Ohyama**, Y. Matsushita, N. Kurita.

また、共著者として、以下の論文を学術雑誌に発表した。

1. "The effects of vitronectin on specific interactions between urokinase-type plasminogen activator and its receptor: *Ab initio* molecular orbital calculations", T. Kasumi, K. Araki, **T. Ohyama**, S. Tsuji, E. Yoshikawa, H. Kobayashi, N. Kurita, *Molecular simulation*, (2013), in press.
2. "Specific interactions and binding energies between thermolysin and potent inhibitors: Molecular simulations based on *ab initio* molecular orbital method", T. Hirakawa, S. Fujita, **T. Ohyama**, K. Dedachi, M. T. H. Khan, I. Sylte, N. Kurita, *Journal of Molecular Graphics and Modeling* **33**, 1-11 (2012).
3. "Structure-based analysis of the molecular recognitions between HIV-1 TAR-RNA and transcription factor nuclear factor kappaB (NFkB)", M.T.H. Khan, C. Mischiati, A. Ather, **T. Ohyama**, K. Dedachi, M. Borgatti, N. Kurita, R. Gambari, *Journal Current Topics in Medicinal Chemistry* **12**, 814-27 (2012).
4. "*Ab initio* fragment molecular orbital calculations on the specific interactions between human, mouse and rat PPARα and GW409544", M. Hayakawa, **T. Ohyama**, Y. Yamaguchi, S. Iwabuchi, T. Nakagawa, T. Nakajima, N. Kurita, *Molecular Simulation* **36**, 644-656 (2010).

附録 B-2: 国際学会における発表

1. "Analysis of the effects of ligand-binding on conformational change for lactose repressor using molecular dynamics", T. Ohyama, Y. Matsushita, N. Kurita, The53rd Sanibel Symposium, 2013, Feb. 17 - Feb. 22, Georgia, U.S.A.
2. "Specific interactions between transcriptional regulatory protein and DNA: molecular simulations combined with MD and *ab initio* FMO methods", **T. Ohyama**, K. Nomura, N. Noriyuki, 2nd AICS International Symposium, 2012, March 1-2, Kobe, Japan.
3. "Analysis of specific interactions between transcriptional regulatory protein and DNA: molecular simulations combined with MD and *ab initio* FMO methods", **T. Ohyama**, K. Nomura, N. Kurita, The52nd Sanibel Symposium, 2012, Feb. 19-24, Georgia, U.S.A.

4. "Allosteric effect on specific interaction between LacR and DNA induced by ligand-binding to LacR: molecular simulations combined with molecular dynamics and *ab initio* fragment molecular orbital methods" **T. Ohyama**, M. Hayakawa, S. Nishikawa, N. Kurita, The 51st Sanibel Symposium, 2011, Feb. 25 - March 1, Georgia, U.S.A.
5. "Specific interactions between lactose repressor protein and DNA affected by ligand binding: *ab initio* fragment molecular orbital calculations", **T. Ohyama**, M. Hayakawa, S. Nishikawa, N. Kurita, CBI Annual Meeting 2010, 2010, September 15-17, Tokyo, Japan.
6. "Specific interactions between lactose repressor protein and its inducer and anti-inducer molecules as well as DNA: molecular mechanics and molecular orbital simulations", **T. Ohyama**, S. Nishikawa, Y. Sengoku and N. Kurita, International Symposium on Multi-scale Simulations of Biological and Soft Materials , 2008, June. 18-20, Tokyo, Japan.

附録 B-3 国内学会における発表

1. "分子シミュレーションによる転写制御タンパク質ラクトースリプレッサー二量体と DNA 間の特異的相互作用の解析"、**大山達也**、松下祐貴、栗田典之、第 6 回分子科学討論会, 2012, 東京都.
2. "MD 及び FMO 計算による転写制御タンパク質と DNA 間の特異的相互作用の解析"、**大山達也**、野村和哉、栗田典之、第 25 回分子シミュレーション討論会、2011、東京都。
3. "分子シミュレーションによるアロステリックタンパク質と DNA 間の特異的相互作用の解析"、**大山達也**、野村和哉、栗田典之、第 5 回分子科学討論会, 2011、北海道。
4. "転写制御タンパク質ラクトースリプレッサーとリガンド及び DNA との特異的相互作用に関する高精度分子シミュレーション"、**大山達也**、西川真、栗田典之、第 3 回分子科学討論会、2009、愛知県。

附録 B-4: 研究助成

申請者がこれまでに受けた研究助成の名称とその内容をいかに示す。これらのご支援を受け、研究に専念し、本論文に至る研究成果を得ることができた。

1. 豊橋技術科学大学　平成 24 年度卓越した大学院拠点形成プログラム
   リサーチ・アシスタント(卓越拠点 RA)
   (給与: 18 万円、平成 24 年度)
2. 公益財団法人　堀科学芸術振興財団
   (渡航費及びその他の研究支援: 35 万円、平成 24 年度)
3. 吉田科学財団
   (渡航費: 25 万円、平成 23 年度)
4. 豊橋技術科学大学　平成 23 年度外部資金獲得支援経費
   (研究支援: 19 万 2 千円、平成 23 年度)