

理工学研究科

2011 年 3 月

修士論文

半導体材料の二次元欠陥の第一原理計算

M9321 戸賀瀬 健介
(情報科学専攻)

概 要

本研究では、半導体の結晶表面や積層欠陥等の二次元欠陥を対象に、第一原理計算ソフト VASP (Vienna Ab-initio Simulation Package) を用いてエネルギー計算を行った。具体的には、リン (P) 等のドーパントが、シリコン (Si) 中に含まれる積層欠陥に与える影響について考察した。一方で、シリコンカーバイド (SiC) において、環境依存を考慮した精密な表面エネルギー計算からマイクロパイプ欠陥の生成起源を調べた。

Si は半導体として広く普及している材料である。Czochralski 法 (CZ 法) で生成された Si 単結晶はほとんど転位を含まず、商業化が可能であるが、デバイスの製造工程中に、頻繁に転位が発生する。転位は半導体デバイス中に含まれるとリーク電流の原因となることが知られており、Si テクノロジーにおいて転位の抑制・制御が求められている。一本の完全転位がリボン状に拡張すると、2 本の部分転位に別れ、その間は積層順序の乱れた積層欠陥になることが知られている。最近、東北大金研の Y. Ohno らは、Si 中の積層欠陥が、ドーパされた P が欠陥部に集まることで安定化することを報告した。本研究では、この現象を説明するため、第一原理計算を行った。まず単純に Si バルクにおいて、Diamond 構造と Wurtzite 構造のエネルギー差を計算した。そして Si バルク中の一原子を P に置換し、同様の計算を行い、P が含まれた Si とドーパントを含まない Si の積層欠陥エネルギーを見積もった。次に、Diamond 構造の Si バルク中に、glide-set 転位に積層欠陥を入れたモデルを作成した。そしてモデル中の一層を P に置換し、エネルギーを計算し、P が安定化する位置を探った。結論として Si 中に P が含まれると積層欠陥が安定し、さらに P は積層欠陥部に集まり易いことが示唆され、Y. Ohno らの報告と整合した。

次世代パワー半導体材料として注目されている SiC の単結晶成長には、Lely 法と呼ばれる気相成長法が主に用いられている。しかし、この手法で成長させた SiC 単結晶には、リーク電流の原因となるマイクロパイプ欠陥が {0001} 面上に多数確認されている。最近、関西学院大・金子らが、準安定溶媒エピタキシー (MSE : Metastable Solvent Epitaxy) と呼ばれる新奇な SiC 単結晶成長法を開発した。この手法で成長させた SiC 単結晶の {0001} 面には、マイクロパイプ欠陥は無く、平坦に成長している。マイクロパイプ欠陥の生成機構は、Frank による超格子転位を起源とする説が有力であるが、本研究では両手法には成長環境にの違いに着目した。Lely 法では黒鉛坩堝を用いているので、気体溶媒中に大量の炭素 (C) が溶け出し、成長環境は C-rich と考えられる。一方、MSE 法の成長環境は種結晶が液体 Si に覆われているため、Si-rich である。環境によってその値が変わる物性として表面エネルギーが知られており、これは静的な要因として結晶成長を支配している。本研究では、精密な第一原理計算によって、SiC の環境に依存した表面

エネルギーを計算し、SiC の単結晶成長においてマイクロパイプ欠陥が生じる新たなモデルを提案した．具体的な計算として、SiC の $\{0001\}$ 面とそれに直交する $\{11\bar{2}0\}$ 面、 $\{1\bar{1}00\}$ 面の表面エネルギーを求めた．その手法として、バルクモデルと真空-固体界面を有するスラブモデルのエネルギー差から表面エネルギーを計算した．また Si 面、C 面が交互に現れる $\{0001\}$ 面は、環境による表面エネルギーへの影響が大きい．Si-rich、C-rich 環境下での $\{0001\}$ 面の表面エネルギーを精密に計算するため、本研究では、Si、C それぞれで表面が覆われたスラブモデルを作成し、化学ポテンシャルの概念を利用した．そして、 $\{0001\}$ 面と直交する 2 面の表面エネルギーを比較すると、Si-rich では $\{0001\}$ 面が安定化、C-rich では $\{0001\}$ 面が不安定化するという結果が得られた．この結果は、Si-rich では $\{0001\}$ 面の表面積が大きくなるように結晶成長するため、 $\{0001\}$ 面上に生じたマイクロパイプ欠陥は拡散原子によって埋め立てられ、閉塞していくことを示唆しており、C-rich では、 $\{0001\}$ 面上で拡散原子が結晶に取り込まれ、マイクロパイプ欠陥を維持したまま結晶成長するため、欠陥濃度が高くなることを示唆している．

目次

第1章	序論	3
1.1	研究背景	3
1.2	半導体材料	3
1.2.1	シリコン (Si : Silicon)	3
1.2.2	シリコンカーバイド (SiC : Silicon Carbide)	4
1.3	結晶構造	6
1.3.1	結晶の緒言 [?]	6
1.3.2	fcc 構造と hcp 構造と積層周期	10
1.3.3	Diamond 構造と Wurtzite 構造	13
1.3.4	SiC 結晶多形	14
1.4	転位 (dislocation)	17
1.5	積層欠陥 (SF : stacking fault)	19
1.6	拡張転位 (Extended dislocation)	20
1.7	glide-set 転位と shuffle-set 転位 [?]	22
1.8	SiC 単結晶成長法	24
1.8.1	レイリー法 (Lely-method)	24
1.8.2	準安定溶媒エピタキシー (MSE : Metastable Solvent Epitaxy)	25
1.9	マイクロパイプ欠陥の生成機構	28
第2章	計算原理	30
2.1	第一原理計算	30
2.2	シュレディンガー方程式	31
2.3	ポテンシャル・波動関数・エネルギー	33
2.4	ばねモデル	34
2.5	密度汎関数理論	36
2.6	擬ポテンシャル法	37
2.7	交換相関関数	39
2.8	VASP(Vienna Ab-initio Simulation Package)	39
2.8.1	INCAR	40
2.8.2	POSCAR	43
2.8.3	POTCAR	47

2.8.4	OUTCAR	48
2.8.5	KPOINTS	49
第 3 章	SiC マイクロパイプ欠陥の環境依存性	51
3.1	緒言	51
3.2	表面エネルギー計算	52
3.3	極性面の表面エネルギー計算	55
3.3.1	極性面	55
3.3.2	被覆率を考慮した計算モデル	56
3.3.3	化学ポテンシャル	56
3.4	計算結果	59
3.5	考察	62
第 4 章	総括	64
付 録 A	Maple による結晶格子データの作成	70
付 録 B	Maple によるスラブモデルの作成	75
付 録 C	Maya による視覚化	81
C.1	視覚化の流れ	81
C.2	実行ファイルについて	82
C.3	ライブラリ	94

第1章 序論

1.1 研究背景

半導体デバイスは、単結晶の半導体材料の中で電子等に電界をかけて全体として一定方向へ移動させることによって動作する。このとき電子の運動を微視的に見ると、規則的な結晶格子から散乱を受けながらも全体として一定方向へ移動している。格子配列の乱れた部分では、電子はさらに大きく散乱されるので、平均的な移動速度は低下する[?]。また電界をかけていないにも関わらず電流が流れる（電流がリークする）原因にもなる。そのため半導体テクノロジーにとって、高純度の単結晶を生成すること、結晶中に含まれる格子欠陥を制御することは重要な課題である。

近年、情報通信機器等の高性能化・小型化・軽量化・低消費電力化への社会的ニーズは増々高まっている。情報通信機器の構成部品である各半導体デバイスにおいても高速・高集積・低消費電力化等が求められており、ナノオーダーでの格子欠陥の制御、もしくは新しい半導体材料の開発が重要となっている。

そして物質・材料の研究において、シミュレーションが活用されている。この背景には、近年のコンピュータ性能の急激な向上、シミュレーション手法の進展、ナノテクノロジーに代表される微小領域における実験技術の向上などがある。平成21年の7月に、次世代スーパーコンピュータ戦略委員会が分野2において新物質・エネルギー創成を掲げたこともあり、今後もシミュレーションの活躍が期待されている。一方で、シミュレーション手法の一つである第一原理計算は、原子の格子モデルから電子構造を計算し、系のエネルギーを精確に求める強力な手法として知られている。本研究では、第一原理計算を用いて、半導体材料の二次元欠陥について調べた。

1.2 半導体材料

1.2.1 シリコン（Si : Silicon）

シリコン（Si）は、常温、常圧では、Diamond構造で安定化となり、半導体デバイスとして広く普及している材料である。小節??で述べたように、半導体材料をデバイスとして使用するには、高純度の単結晶として生成する必要がある。

Si 結晶はイレブンナインの純度 (99.99999999%) を持つ、という話がある。半導体デバイスにおいて、それほどの高純度の Si 結晶が求められているが、実際の半導体デバイスにはそこまでの純度は無く、むしろ半導体製造過程において積極的に純度を低下させる。純度の高い Si 単結晶は、絶縁体に近く、Si のみではスイッチの役割を果たさない。そのためドーパントと呼ばれる不純物を入れる (ドープする)。ドーパントには、ホウ素 (B) 等の III 族元素、リン (P) 等の V 族元素がある。絶縁体である Si に適度なドーパントを入れることで、電荷の移動を担うキャリアを生じさせ、Si 中に電流を流れるようにする。

Si は主に Czochralski 法 (CZ 法)、Floting Zone 法 (FZ 法) を用いて単結晶を生成する。この方法で生成された Si 単結晶はほとんど転位を含まず、商業化が可能であるが、デバイスの製造工程中に転位が頻繁に含まれる。そのため転位の制御は、未だ Si テクノロジーにおいて重要な課題の一つである。転位とは、小節??で述べるが、原子配列の局所的な乱れが線状に連なっている格子欠陥のことで、結晶学的には線欠陥に分類される。格子欠陥である転位は、半導体デバイス中に含まれるとリーク電流の原因となることが知られている。そのため半導体製造過程において発生する転位の抑制は重要な課題となっている。

最近、東北大金研の Y. Ohno らによって、P がドープされた Si 中の積層欠陥が、P が欠陥部に集まることで安定するという報告 [?] があった。本研究では、この現象を説明するため、第一原理計算を行った。

1.2.2 シリコンカーバイド (SiC : Silicon Carbide)

シリコンカーバイド (SiC) はすぐれた物性的特徴から次世代パワー半導体の材料として注目されている。少々節??で記した通り、現在、最も多く用いられている半導体材料は Si であり、パワー半導体でも Si が主流である。しかし Si はデバイス性能が理論限界に近づいており、これ以上の改善が期待できない。そこで Si よりもオン抵抗、耐熱性に優れた SiC に大きな期待がかかっている。Si の理論限界の一つは耐熱とオン抵抗のトレードオフである。このような Si の固有特性に対し、SiC はそれらの特性を超える固有特性を有し、パワーデバイスの性能改善に大きく貢献していることが証明されている。表??に SiC と Si の物理定数の比較をまとめた。SiC はワイドバンドギャップ半導体と言われ、価電子帯から伝導帯に電子を持ち上げるのに必要なエネルギー (eV) が Si に比べ約 3 倍である。このワイドバンドギャップにより、Si と比較し、SiC の絶縁破壊電界強度は 10 倍にもなる [?]。また SiC は Si に比べ融点が高い。このためデバイスの高温動作が可能になり (耐熱性)、Si では 200[] 以下であるのに対し、SiC では 600[] 程度まで動作可能との報告がある [?]。また耐圧とオン電圧とのトレードオフは、絶縁破壊電界強度に依存する。破壊強度が 10 倍というのは、同じ耐圧を得るのに n^+ 層の厚みを 1/10 にできる。これは図??に示すように MOSFET 機構のデバイスで考えると、オン抵抗を 1/300 に低減できる可能性をもっている。このように SiC で実現できる低オ

ン抵抗，高温動作，高热伝導は，いずれもパワーデバイスに要求される特性であり，実用化が期待される．

そんな SiC が未だ一般で実用化に至らない原因の一つは，単結晶成長法にある．SiC 以外の半導体材料ウェーハは液相からの成長によって製造されているが，SiC ではこのような低コストでの製造法が未だ確立していない．これは SiC 相を液相と同じ組成から直接に凝固で得られないからである．詳しくは??節に記すが，現在の SiC は Lely 法 [?] と言われる気相成長法で，単結晶を生成するのが主流である．しかし，この手法で成長させた SiC 単結晶には，リーク电流の原因となるマイクロパイプ欠陥が {0001} 面上に多数確認されている [?]．一方で，関西学院大・金子らが，準安定溶媒エピタキシー（MSE : Metastable Solvent Epitaxy）[?] と呼ばれる新奇な SiC 単結晶成長法を開発した．この手法で成長させた SiC 単結晶の {0001} 面には，マイクロパイプ欠陥は無く，平坦に成長している．

表 1.1: SiC と Si の物理定数比較 [?] ．

材料	バンドギャップ (eV)	絶縁破壊電界強度 (MV/cm)	熱伝導率 (W/cmK)
SiC(4H)	3.25	3.0	4.5
Si	1.1	0.3	1.5

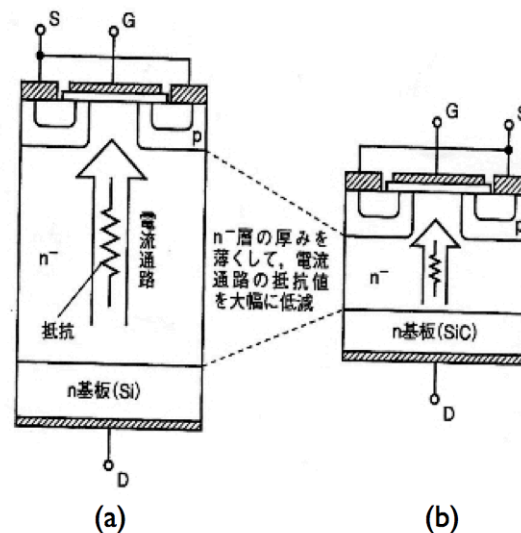


図 1.1: (a)Si-MOSFET と (b)SiC-MOSFET の違い

1.3 結晶構造

1.3.1 結晶の緒言 [?]

本研究では，第一原理計算の計算モデルとなる結晶格子やまた原子レベルの格子欠陥について構造の特徴から議論を行っている．本小節では，結晶格子の特徴や専門用語について簡単に説明する．

結晶格子と格子定数

ここで，代表的な結晶格子を図??に示し，各々について紹介する．はじめに，図??(a)には最も単純な規則配列が示されている．ちょうど赤線で造られる立方格子の格子点に原子が配置しているので，このような結晶構造を単純立方格子 (simple cubic lattice) という．また大きな結晶格子を造る際に，積み上げる単位ブロック (ここでは赤線で示したブロック) を単位胞 (unitcell) という．次に，図??(b)に示す結晶格子は，体心立方格子 (bcc : body centered cubic lattice) と呼ばれ，立方格子の格子点と中心 (重心) に原子が配置する構造となっている．鉄 (Fe) やタングステン (W) の結晶がこの構造をとる．続いて，図??(c)に示すのが，金 (Au)，銀 (Ag)，銅 (Cu)，アルミニウム (Al) といった金属の結晶格子にみられる面心立方格子 (fcc : face centered cubic lattice) である．この格子では，原子が単位胞である立方格子の格子点と全ての面の中心に配置している．この構造では，原子全てが，同じ大きさの真球の場合に最も密な原子配列をとるため，立方最密格子と言われる．面心立方格子の他にもう1つ，最密な原子配列をとる結晶構造がある．それが図??(d)に示す六方最密格子である．この結晶構造は，マグネシウム (Mg) や亜鉛 (Zn) などに見られる．面心立方格子と六方最密格子 (hcp : hexagonal close packed lattice) では，ともに最密構造であるが構造は異なる．詳しくは次小節で述べるが，その違いは積層の順序・周期にある．続いて格子定数について簡単

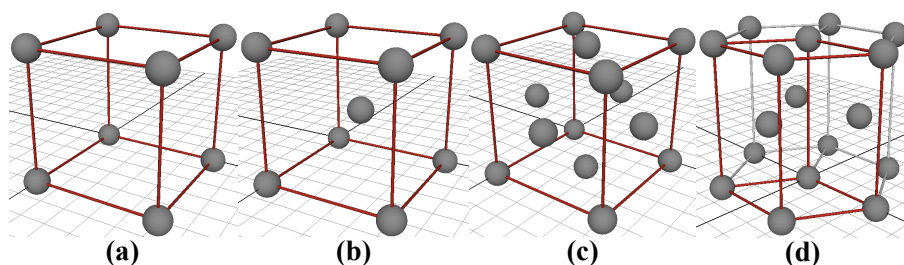


図 1.2: 代表的な結晶格子の格子の形と原子配置の様子．(a) 単純立方格子 (simple cubic lattice) ．(b) 体心立方格子 (bcc : body centered cubic lattice) ．(c) 面心立方格子 (fcc : face centered cubic lattice) ．(d) 六方最密格子 (hcp : hexagonal close packed lattice) ．

に説明する．先に述べた通り，fcc 構造であるが，Cu や Al のように組成が異なれば，結晶格子の大きさ等の形も異なる．つまり結晶に応じて特定の結晶軸の長さや軸間角度があり，それが格子定数である．格子定数は，単位胞の各軸間の角度， α, β, γ と各軸の長さ a, b, c を表す 6 個の定数で成り立つ．また，格子定数の長さの単位は [] を用いるのが一般的であるが，ナノメートル [nm] が用いられることもある（オングストローム：1 [] = 0.1 [nm] = 10^{-10} [m]）．格子定数 a, b, c と α, β, γ の値が，格子空間において指し示す位置を図??に示す．しかし，格子の形状等によっては，格子定数を a のみで表すこともある．例えば，fcc 構造の場合，立方体である以上 $a=b=c$ と $\alpha = \beta = \gamma = 90^\circ$ が成り立つので， a のみで表すことができる．また hcp 構造の格子空間と格子定数について図??に示す．hcp 構造の場合， $a=b \neq c$ となり， $\alpha : \beta : \gamma = 90 : 90 : 120^\circ$ となる．hcp の構造を議論する際， $a \neq c$ となるため， a 軸の長さと c 軸の長さの比を示す c/a の値が出てくることが多い．補足だが，hcp 中にある原子を真球とすると， $c/a=1.63$ が成り立つ．これを c/a の理想軸比という．

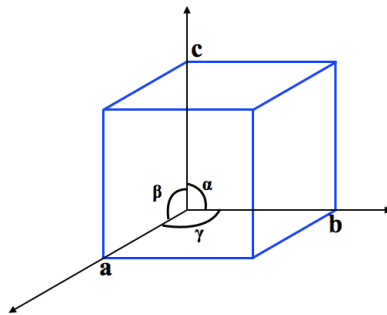


図 1.3: 格子空間と格子定数の関係．青で示した格子空間に対し，格子定数 a, b, c ， α, β, γ は図で示す値となる．

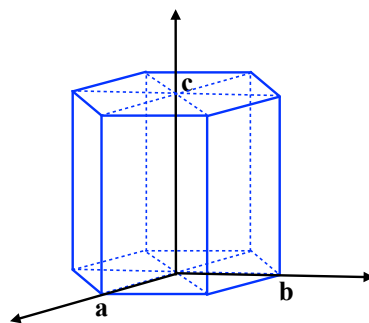


図 1.4: hcp 構造等の六方形の a, b, c 軸の取り方．青で示した格子空間に対し，図のように， a, b, c 軸を取る．この場合の角度は $\alpha : \beta : \gamma = 90 : 90 : 120^\circ$ となる．

ミラー指数と結晶表面

ここでは結晶表面とその向きを指定するミラー指数について簡単に紹介する．結晶表面の構造の異方性を考えるためにミラー指数が用いられる．格子面が $(a : b : c)$ 軸上で切る点 $(x : y : z)$ に対し，ミラー指数は $(1/x, 1/y, 1/z)$ で定まる．(111) 面である図??(a) を例に説明する．(111) 面は， $a=1$ の点， $b=1$ の点と $c=1$ の点を切った面なので， $(x : y : z)$ は $(1 : 1 : 1)$ となる．そうすると $(1/1, 1/1, 1/1)$ から (111) 面と定まる．また (100) 面を示す図??(b) を例に説明すると，(100) 面は， $a=1$ の点を切るが， b, c 軸を切らず，無限に広がっている．つまりミラー指数は $(x : y : z)$ から $(1 : \infty : \infty)$ となり，格子面は $(1/1, 1/\infty, 1/\infty)$ から (100) 面と定まる．また図??(c) に示すような六方形では，4 つのミラー指数で $(1/a_1 : 1/a_2 : 1/a_3 : 1/c)$ として格子面を表すことが多い．

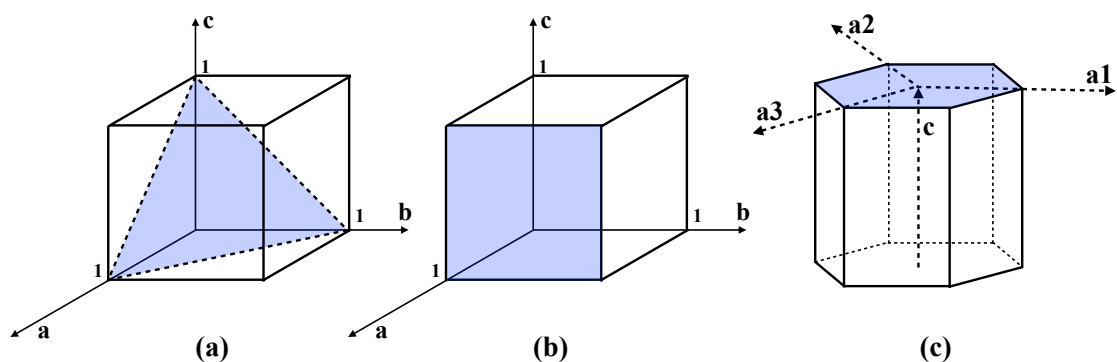


図 1.5: (a) 立方形の (111) 面．(b) 立方形の (100) 面．(c) 六方形の (0001) 面．

ある面が軸を負の側で切る場合は，対応する指数は負で，指数の上にマイナス記号をつけ $(\bar{h} : k : l)$ のように表示する．また異なる方位の面であっても，結晶格子の対称性から面内の原子配列が等価な面がある．そのような面をまとめて表す場合は $\{ \}$ を用いる．例えば，fcc 構造の (100) 面と等価な面は，他に (010), (001), $(\bar{1}00)$, $(0\bar{1}0)$, $(00\bar{1})$ があるが，これらをまとめて表すと $\{100\}$ 面と表記をする．またミラー指数 (hkl) で表される面に垂直な方向を $[hkl]$ で，それと等価な方向の集合を $\langle hkl \rangle$ で表す．最後に立方格子と面の原子配列の様子を，図??にまとめたので，参考にしてほしい．

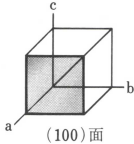
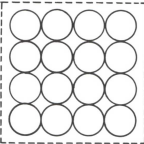
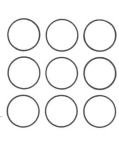
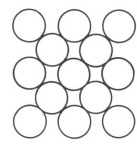
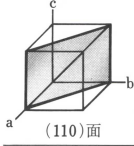
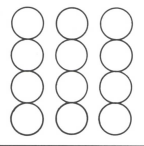
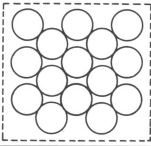
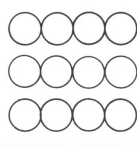
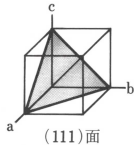
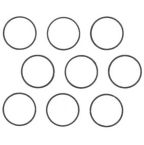
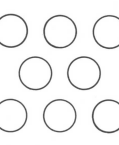
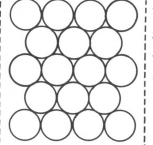
結晶構造 格子面	単純立方格子 (s.c.)	体心立方格子 (b.c.c.)	面心立方格子 (f.c.c.)
 (100)面	 最稠密面		
 (110)面		 最稠密面	
 (111)面			 最稠密面

図 1.6: 立方格子結晶の表面構造の異方性．原子が最も密に充された最稠密面を破線枠で囲っている．

1.3.2 fcc 構造と hcp 構造と積層周期

先の小々節??で、fcc と hcp はともに最密構造でありながら、周期性に違いがあると述べたが、ここでその違いを説明する。fcc 構造と hcp 構造は、両者とも原子を最密に並べた面を有する。その様子は図??より fcc 構造の (111) 面にて確認できる。fcc 構造と hcp 構造の違いは最密面の積み方、つまり積層順序に違いがある。最密面上に最密面を積む方法を図??に示す。青で示した最密面上に最密面を積むパターンは、図??(a) もしくは図??(b) がある。つまり最密面上において原子が置ける場所は、図??(c) の黒で示す菱形の内の順正三角形の重心もしくは逆三角形の重心となる。つまり青の最密面を A 面とするとその上に積める面は、緑の B 面もしくは赤の C 面となる。また B 面の上に積めるのは、A 面もしくは C 面となり、C 面の上に積めるのは、A 面もしくは B 面となる。このように積層順序は、最密面の相対的な位置で A, B, C 面と区別することが出来る。

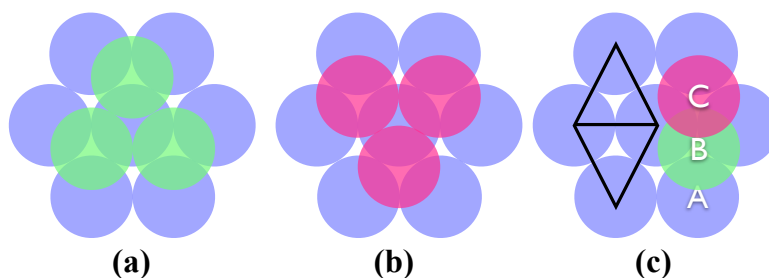


図 1.7: 最密面の積み方の様子。最密面上への積層は、(a), (b) のいずれかである。(c) に先の 2 通りの最密面の積層を示す。(a) は (c) に示す黒い菱形の逆三角形の重心にあたり、(b) は正三角形の重心にあたる。最密面の積層順序は相対的な位置で A, B, C を区別できる。

ここで fcc 構造と hcp 構造の積層順序に着目する。fcc 構造は図??(a) に示す構造をしている。fcc 構造を (110) 面からみると図??(b) に示すように見える。図??(a)(b) の原子の色、対角線、ユニットセルの色は対応している。図??(a)(b) は両者とも、原子の色が単色で成る面が最密面を示している。図??(b) が示すように、fcc 構造は青の A 面、緑の B 面、赤の C 面、... の順で積層しており、それが無限に続いていることから A, B, C 面の 3 層で 1 周期であることが解る。一方で、hcp 構造を図??(a) に示し、(1120) 面からみた hcp 構造を図??(b) に示す。fcc の時と同様に、図??(a)(b) の原子の色、ユニットセルの色は対応している。図??(b) に示すように、hcp 構造では、青の A 面、緑の C 面、青の A 面、... の順で積層している。hcp 構造は fcc 構造と違い、C 面がなく、A, B 面の 2 層で 1 周期となっている。

まとめると、fcc 構造は...|ABC|ABC|... の 3 周期構造であるのに対し、hcp 構造は、...|AB|AB|... の 2 周期構造である。ともに最密格子となる fcc 構造と hcp 構造の違いは、1 周期を構成する積層順序の違いにある。

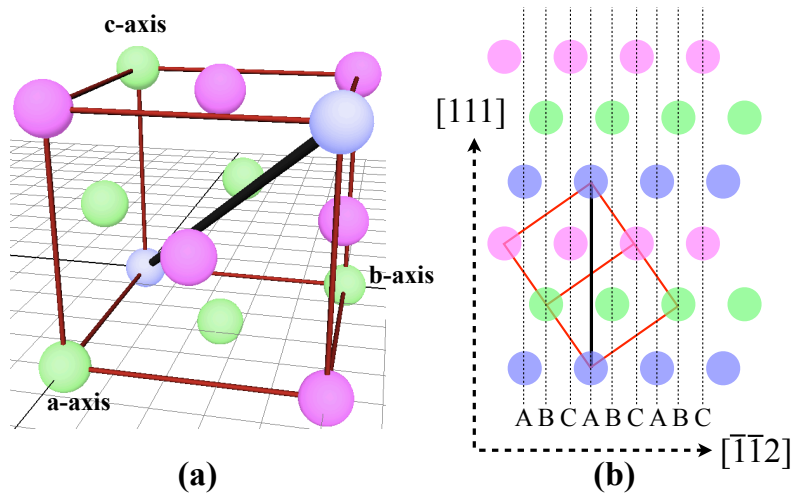


図 1.8: fcc 構造の積層順序 . (a) は fcc 構造を示し , (b) は $(1\bar{1}0)$ 面から見た fcc 構造を示している .

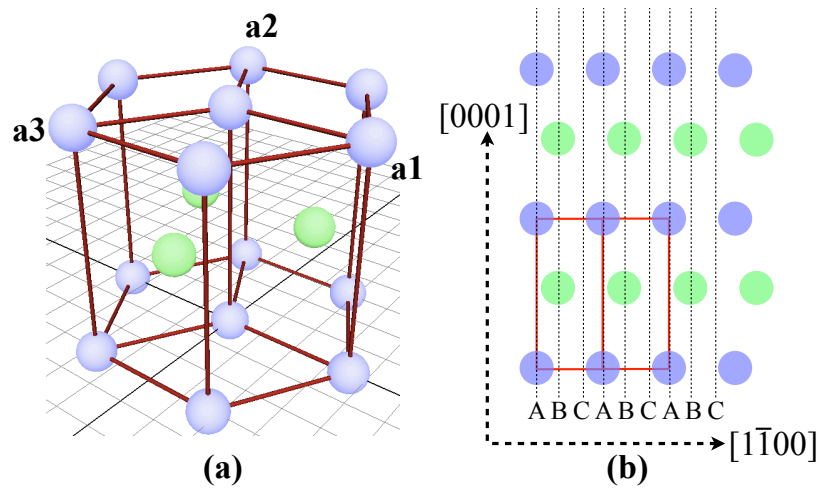


図 1.9: hcp 構造の積層順序 . (a) は hcp 構造を示し , (b) は $(11\bar{2}0)$ 面から見た hcp 構造を示している .

この fcc と hcp の構造の違いから，結晶格子において ABC のような積層順序となる箇所を cubic 構造 (face centered "cubic") として，ABA のような積層順序となる箇所を hexagonal 構造 ("hexagonal" close packed) と見なすことができる．また最密格子は fcc 構造と hcp 構造のみでなく積層順序次第ではほぼ無限にある．...|ABCB|.. のように 4 層で 1 周期となる 4H 構造や...|ABCACB|.. のように 6 層で 1 周期となる 6H 構造，他に 18R 構造などもある．このような結晶多形の表記法は「Ramsdell の表記法」といわれ，先の数字が 1 周期中に含まれる層の数，後のアルファベットは結晶系の頭文字「C：立方晶 (cubic)，H：六方晶 (hexagonal)，R：菱面体 (rhombohedral)」を示す．つまり fcc 構造は 3C として，hcp 構造は 2H として表記できる．例として，3C, 2H, 4H, 6H, 18R の積層順序の様子を図??に示す．赤線は構造の 1 周期ごとに引いている．右側の c,h のアルファベットは，その層の構造を示す．この構造の見分け方は，着目した層の前後の層の位置が異なれば" c" となり，同じであれば，" h" となる．例えば ABC の並びで B に着目すれば前後の積層は A, C となるので" c" 構造，ABA の並びで B に着目すれば前後の積層は A, A となるので" h" 構造となる．

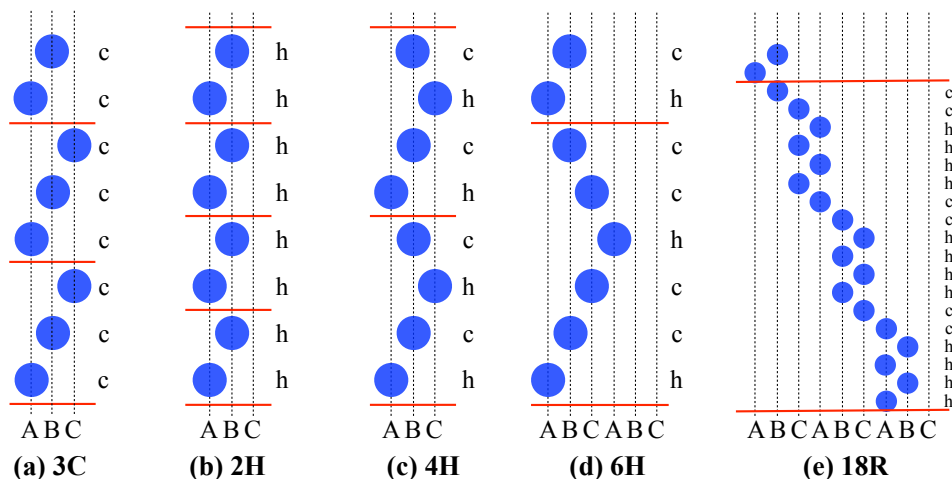


図 1.10: 積層順序の異なる最密格子の例．(a) は 3C, (b) は 2H, (c) は 4H, (d) は 6H, (e) は 18R をそれぞれ示す．図の右の" c", " h" の表記は，左の層が cubic 構造か hexagonal 構造かを示している．この構造の見分け方は，着目した層の前後の層の位置が異なれば" c" となり，同じであれば，" h" となる．例えば ABC の並びで B に着目すれば前後の積層は A, C となるので" c" 構造，ABA の並びで B に着目すれば前後の積層は A, A となるので" h" 構造となる．

1.3.3 Diamond 構造と Wurtzite 構造

Diamond 構造は fcc 構造と密接に関係している．Diamond 構造と fcc 構造の関係は，図??(a) (c) に示すように考えるとイメージしやすい．(a) に示す青の一つの fcc 構造に，(b) に示すように対角線に沿って，対角線長の $1/4$ ズラした位置にもう一つの緑の fcc 構造を重ねる．その内，赤枠内の原子のみを残す．そうすると (c) に示す通り，Diamond 構造となる．

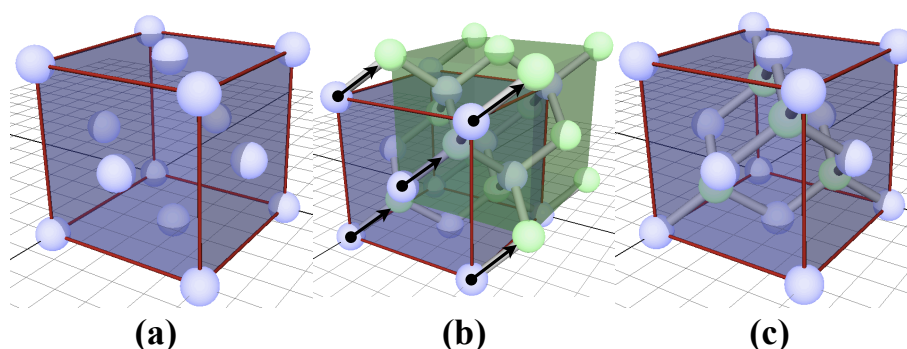


図 1.11: (a)fcc 構造．立方体の頂点の面の中心に原子がある．(b) 青い fcc 構造と緑の fcc 構造を重ねた図．青の立方体の対角線に沿って，対角線長の $1/4$ ズラした立方体が緑の fcc．(c)Diamond 構造．青の原子と緑の原子が同じ元素の場合は Diamond 構造だが，異なる元素の場合は Zinc Blende 構造という．

また Diamond 構造は特徴的な正四面体構造を持つ．その様子を図??(a) に示す．Diamond 構造をつくる Si や C のような 4 配位の元素は，図??(b) に示すように，最近接原子と正四面体構造をしている．この様子を $\{1\bar{1}0\}$ 面から見ると，図??(c) のように見える．Diamond 構造を示す際，正四面体が積み重なっている様子を表現することもある．また，図??(c) から Diamond 構造の積層順序も見取れる．Diamond 構造は， $[111]$ 方向に向かって...|AaBbCc|AaBbCc|... と積層している． $\langle 111 \rangle$ 方向への積層について，A と a, B と b, C と c は同様の位置にいると見積もることが可能であり，Aa のペアを A と，Bb のペアを B と，Cc のペアを C と，それぞれ見なすことが出来る．つまり，Diamond 構造は...|ABC|ABC|... の 3 層で 1 周期の立方晶となり，3C 構造ともとれる．

そして，Diamond 構造と fcc 構造の関係と同様に，Wurtzite 構造は，hcp 構造と密接に関係している．Wurtzite 構造は，図??(a) で示すように，hcp 構造における単位格子の原子の位置に 2 種類の元素を重ねておき，次いでそれらの内の一つを垂直方向（c 軸方向 or $[0001]$ 方向）に $(0, 0, u)$ だけ動かした構造として表さ

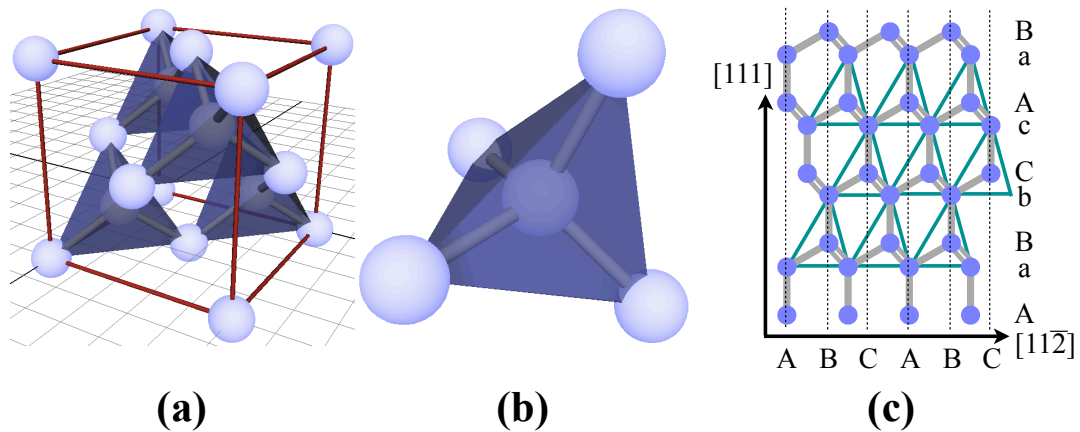


図 1.12: (a)Diamond 構造内にある特徴的な正四面体構造．(b) 正四面体の頂点と重心にそれぞれ原子がある．(c) $\{1\bar{1}0\}$ 面から見た Diamond 構造．緑の枠線が正四面体を示す．

れる．この u を内部パラメータと呼び，構成原子によって，その値は異なる．また Diamond 構造と同様に， Aa のペアを A と， Bb のペアを B と，それぞれ見なし， $\dots|AB|AB|\dots$ の 2H 構造と見れる．

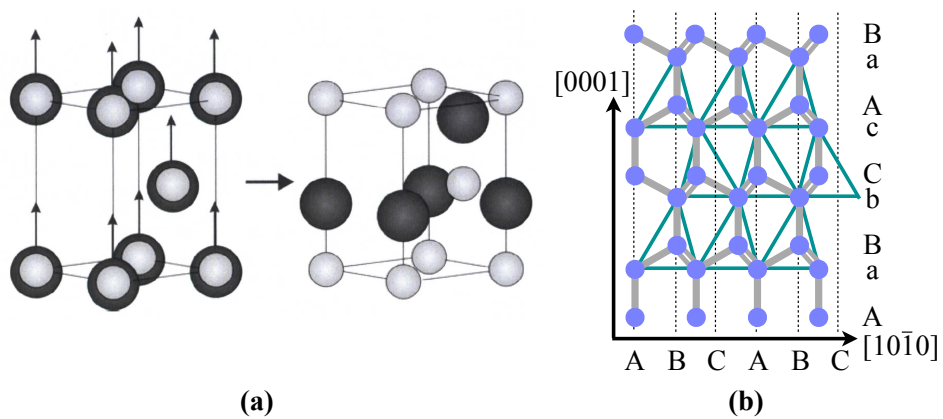


図 1.13: (a)Wurtzite 構造の様子．(b)Wurtzite 構造を $(1\bar{2}10)$ 面から見た様子．緑の枠線で囲まれた箇所は，Wurtzite 構造中に見られる正四面体を示している．

1.3.4 SiC 結晶多形

SiC は IV 族原子同士の結合であるが，Si が C より電気陰性度が大きいことにより若干のイオン性を持つ共有結合型の結晶である．結晶学的には同一の組成で c 軸 方向に対して多様な積層構造をとることができ，100 種類以上の結晶多形が存

在する．その多くの結晶多形のなかでも発生確率が高く，デバイス等に応用する上で重要となる 3C, 4H, 6H-SiC (Ramsdell の表記法) における各結晶構造の単位格子を図??に示す．図??は各結晶構造の単位格子における $(11\bar{2}0)$ 面への投影図を示している．SiC の全ての結晶多形において，Si と C の組成比は 1 : 1 で，Diamond 構造や Wurtzite 構造と同様に， $1/4$ の C($1/4$ の Si) を四つの頂点に配し，中央に Si(or C) 配した正四面体を基本最小構造とする．この正四面体を頂点が重なるように配置すると，a, b, c の 3 種類の配置が考えれる．また正四面体は面内で 180° 回転した配置を取ることができ，このときの配置を a^* , b^* , c^* とする．この 3 種類の配置にどの向きの正四面体を重ねていくかによって SiC の結晶多形が形成される．図??において，正四面体の重心にある原子に焦点を当てて，配置順序を追ってみると，3C では，|bca|bca,... の 3 層で 1 周期を形成していることがわかる．同様に，4H では |bcba|bc... の 4 層で 1 周期，6H では |bcacba| ... の 6 層で 1 周期を形成している．

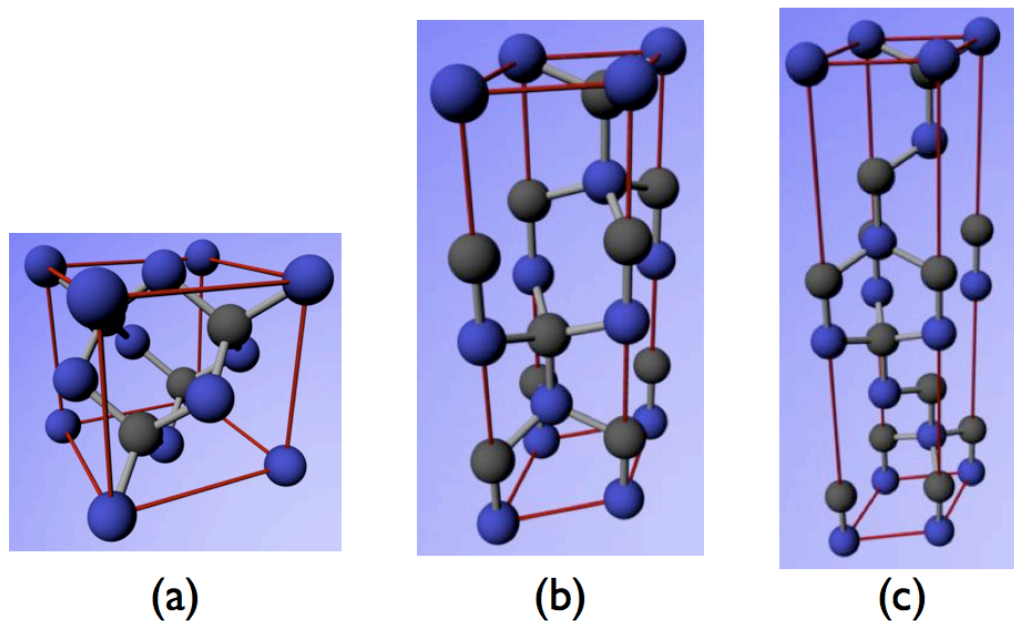


図 1.14: SiC 結晶多形．(a) 3C (b) 4H (c) 6H - SiC．

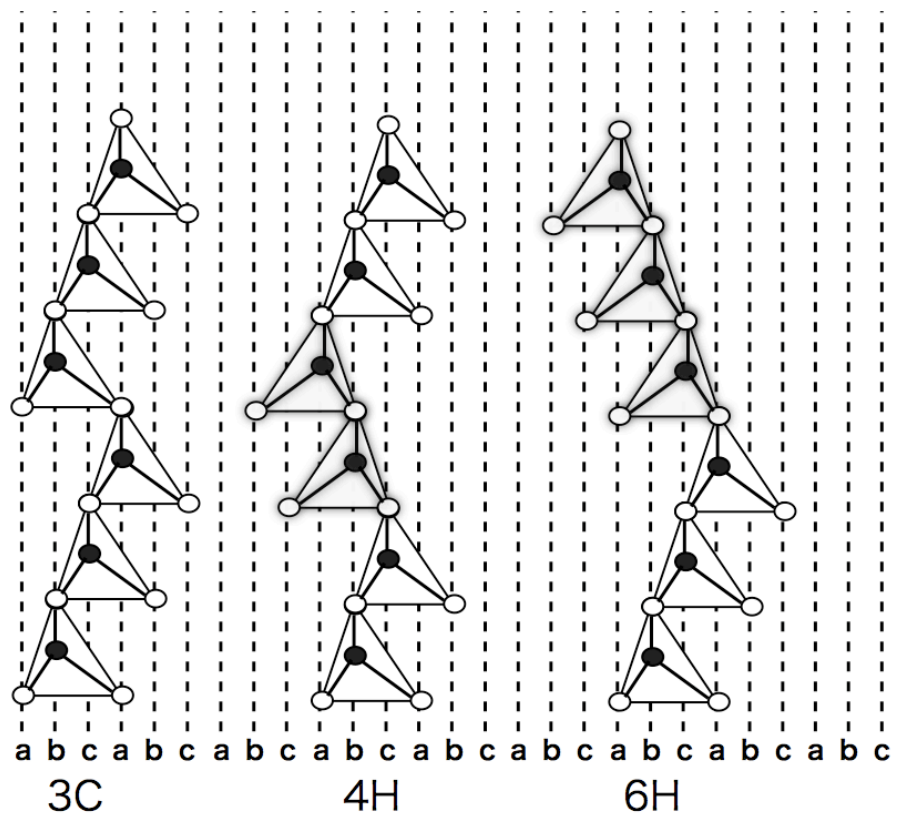


図 1.15: 代表的な SiC 結晶多形の結晶構造の $(11\bar{2}0)$ 面への投影図. がシリコン (Si), がカーボン (C) .

1.4 転位 (dislocation)

結晶の塑性変形の多くは、ある結晶面において特定の結晶方向への変形により結晶面の上下の結晶が相対的にある量だけずれることにより起こる。この結晶面をすべり面と呼ぶ。すべり面は通常、最密面であることが多い。この理由として原子密度が大きな面ほど面間距離が大きく原子面の間でずれが生じ易いためである。また最密面では原子距離が小さいことから、原子同士の結合が強く、互いに離れにくいことも理由の一つである。

すべり面の上下の結晶が一斉にずれるためには、非常に大きな応力が必要となるため、実際はすべり面において原子が局所的にずれていく。そして、すべり面上で原子がすべった領域とすべっていない領域との境界部分には、原子配列が不連続となった結晶欠陥が存在し、それを転位 (Dislocation)[?] といい、境界線を転位線 (Dislocation line) という。転位は、局所的な原子配列の乱れが線状に連なっていることから、線欠陥 (1 次欠陥) に分類され、塑性変形の原因として一般に広く知られている。

転位は転位線の方向とバーガースベクトル (Burgers vector) と呼ばれるベクトルとで表される。そしてバーガースベクトルの周りを囲う閉じた回路をバーガースサーキット (Burgers circuit) と呼ぶ。バーガースベクトルは以下のように定義される。

1. 対象にしている転位線の向きを決める。
2. 図??に示すように、まず完全結晶に対し、転位線と平行な方向の周りに右回りの閉じた回路、バーガースサーキットを描く。なおバーガースサーキットの始点を S、終点を F とする。完全結晶では始点と終点は一致する。
3. 図??で示されるように、記号 \vec{b} で表される転位を含む結晶の転位線の周りに同じ回路を描くと、始点 S と終点 F が一致せず、回路は閉じない。
4. 図??のバーガースサーキットを閉じるために必要な向きと量が、バーガースベクトル \vec{b} となる。

ここで、図??で示すように、転位線とバーガースベクトルが垂直な場合は刃状転位 (Edge dislocation)、といい、図??で示すように、転位線とバーガースベクトルが平行な場合は螺旋転位 (Screw dislocation) という。

また転位において、バーガースベクトルがすべり面上において隣接原子間を結ぶベクトル (= 結晶格子の基本ベクトル) と一致する場合、これを完全転位 (perfect dislocation) という。それに対し、バーガースベクトルが、隣接原子間を結ぶベクトルと一致しない場合、転位は複数の部分転位 (partial dislocation) に分かれる。このような転位を不完全転位 (imperfect dislocation) ともいう。完全転位と部分転位のバーガースベクトルの様子を図??に示す。ここで、完全転位のバーガースベクトル \vec{b} と部分転位のバーガースベクトル \vec{b}_1, \vec{b}_2 の間には、 $\vec{b} = \vec{b}_1 + \vec{b}_2$ が成り立つ。

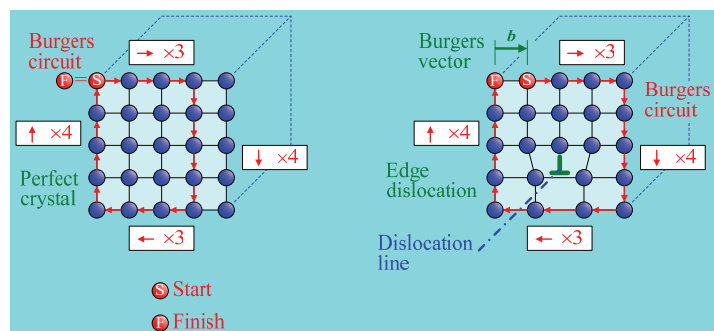


図 1.16: 刃状転位 (Edge dislocation) . 左図が完全結晶 , 右図が転位を含む結晶の様子を示す .

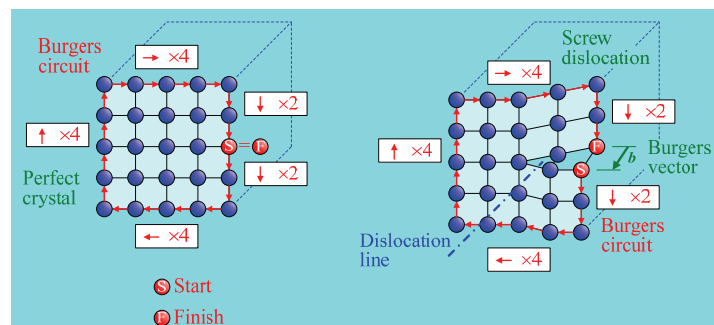


図 1.17: 螺旋転位 (Screw dislocation) . 左図が完全結晶 , 右図が転位を含む結晶の様子を示す .

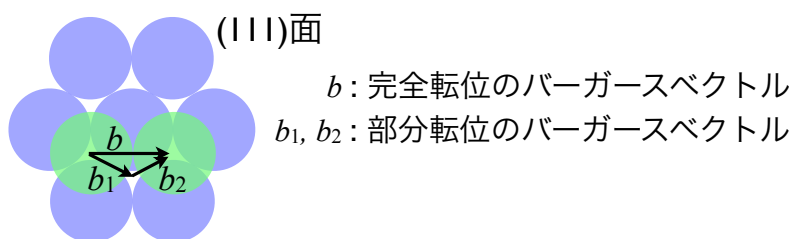


図 1.18: 完全転位と部分転位のバーガースベクトルの様子 .

1.5 積層欠陥 (SF : stacking fault)

積層欠陥とは、完全結晶に対し、積層順序の連続性が局所的に1原子層分だけ乱れた欠陥であり、これはデバイス動作時のリーク源となることが知られている。積層欠陥は面状に形成される格子欠陥であるため、面欠陥（2次元欠陥）に分類される。その様子を図??に示す。fcc 構造では、図??の左のように ABCABC と積層している。赤矢印で示したように、原子層が部分的にずれると、積層順序が ABCABABC となる。その時、図??の赤破線で示した箇所が積層欠陥面となる。また fcc 構造では、全ての層が cubic 構造をしているのに対し、積層欠陥の箇所では、ABA, BAB となる hexagonal 構造が存在することになる。

積層欠陥において、図??(a) に 11/02/03 すように、1 原子層欠落している場合をイントリンシック（空孔）型積層欠陥（intrinsic stacking fault）と呼び、図??(b) に示すように、1 原子層余分に挿入されている場合をエキストリンシック（格子間原子）型積層欠陥（extrinsic stacking fault）と呼ぶ。

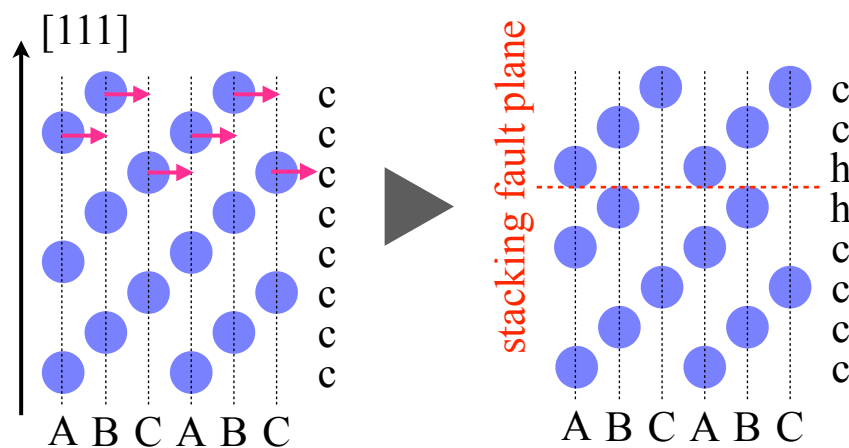


図 1.19: fcc 構造中の積層欠陥の様子。

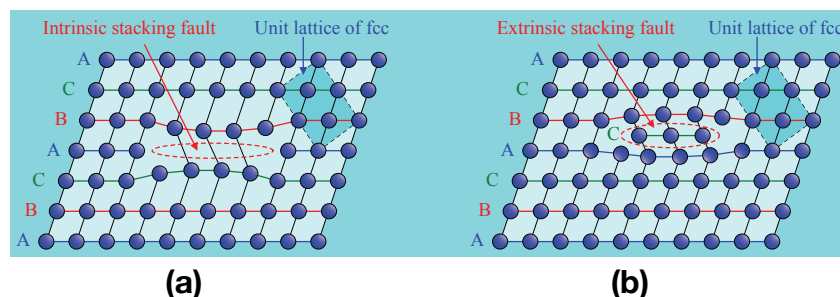


図 1.20: (a)Intrinsic stacking fault(i-SF) (b)Extrinsic stacking fault(e-SF)

1.6 拡張転位 (Extended dislocation)

完全転位は、 $b = b_1 + b_2 + \dots$ のように複数の部分転位に分かれることがある。そして1本の完全転位がリボン状に拡張し、2本の部分転位と積層欠陥になった状態を拡張転位と呼ぶ。拡張転位の様子を図1.21に示す。図の上部から青色で示す完全転位の転位線が走っているが、中程で二本の部分転位に分かれている。そこでは、バーガースベクトルも b でなく、 b_1, b_2 の二本に分かれている。そして二本の部分転位に挟まれた赤で示す領域が積層欠陥となる。この拡張転位において、完全転位および部分転位に挟まれた積層欠陥の原子面における様子を、それぞれ図1.22, 図1.23に示す。図1.22では、B で構成される結晶面の一部がズレて転位（線状の空孔）を形成していることが解る。図1.23では、B で構成される結晶面の一部がズレて、C 面が現れている。そしてC 面がもう一度ズレてB 面に戻っている。つまり、B で構成される結晶面が部分転位に挟まれ、その間はC 面が現れていることが解る。このC 面の領域が積層順序の乱れた積層欠陥となる。

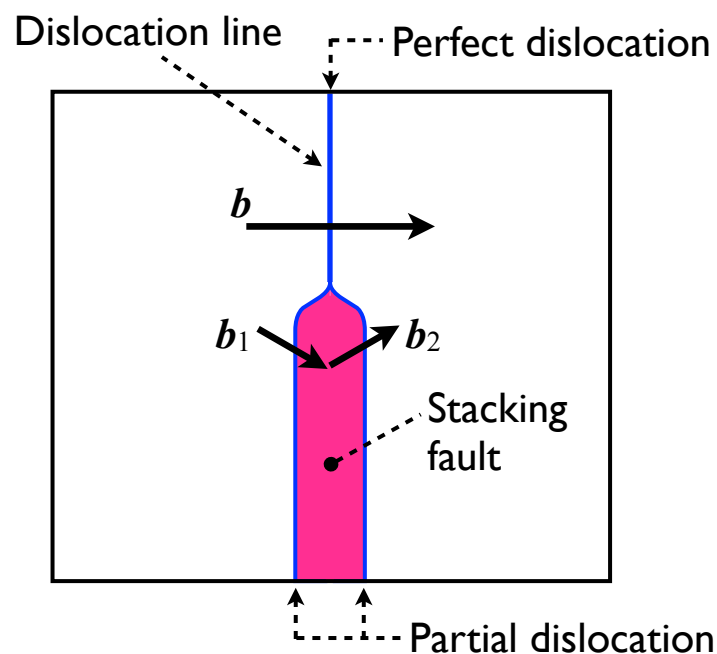


図 1.21: 拡張転位の様子。バーガースベクトル b で表される完全転位が途中で、2 本の部分転位、 b_1, b_2 に分解し、その間に積層欠陥がある。

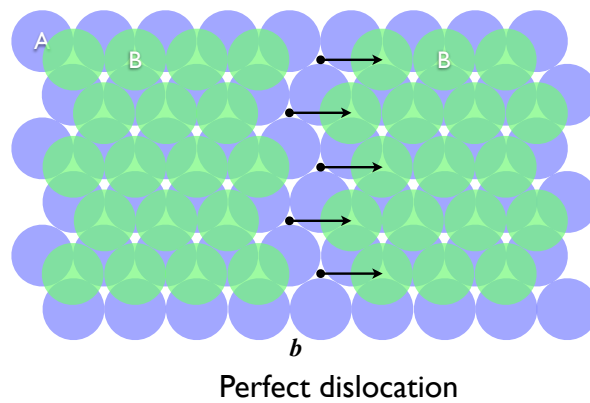


図 1.22: 完全転位の様子．AB と積層した結晶で B の面が b ズれて，図の中心付近に転位を形成している．

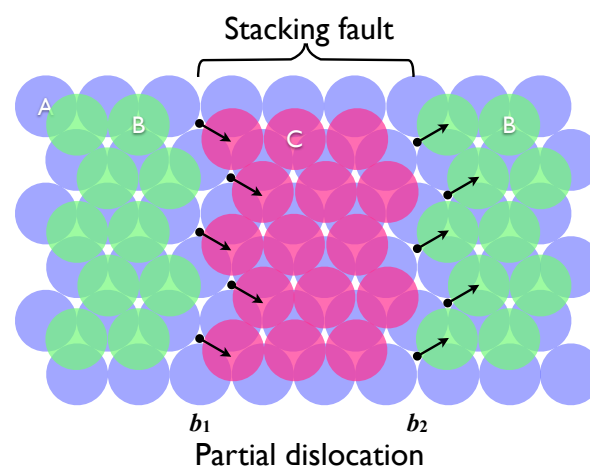


図 1.23: 部分転位に挟まれた積層欠陥の様子．AB と積層した結晶において，B の面が部分的に b_1 ズれて C-site に位置している．そして AB と積層するはずが一部が AC と積層が乱れたので積層欠陥となる．

1.7 glide-set 転位と shuffle-set 転位 [?]

小節??で述べたように，Si は Diamond 構造で安定化する．Diamond 構造は fcc 構造と同様に (111) 面をすべり面とするが，Diamond 構造では，図??に示すように，原子間の距離が広い面間において原子がずれる shuffle-set 転位と原子間の距離が狭い面間において原子がずれる glide-set 転位と呼ばれる 2 種類の転位が存在する．

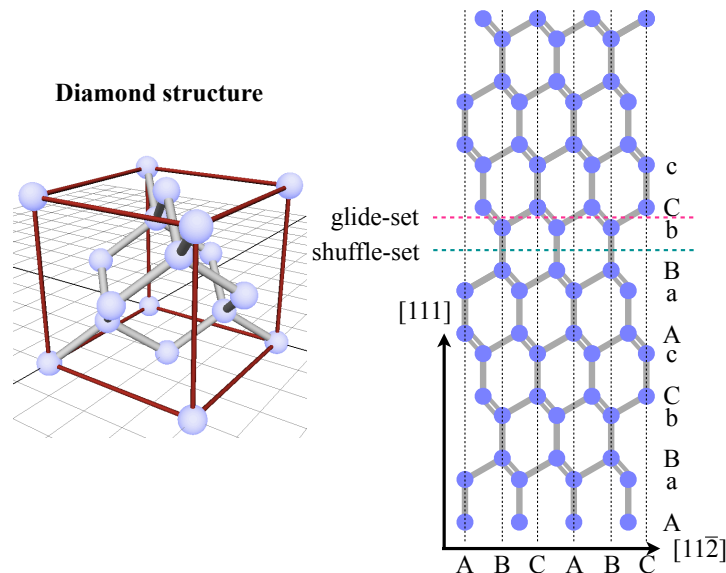


図 1.24: Shuffle-set 転位と Glide-set 転位．

小々節で述べたように，Diamond 構造は，ABCABC... と積層している (A=Aa ペア，B=Bb ペア，C=Cc ペア)．Diamond 構造中の積層欠陥は，一般的に glide-set で安定に存在するといわれている．glide-set での積層欠陥の様子を図??に示す．図??(a) に示すように，glide-set 転位のすべり面より [111] 方向側の全ての原子層を，A B のように 1 site 分ずらすと (b) のように，ABCABAC と積層し，積層欠陥が確認できる．図??(b) にある緑の三角形は，Diamond 構造の特徴の正四面体構造を示しており，glide-set での積層欠陥では，正四面体構造が維持されたまま，積層順序が乱れている．一方で，shuffle-set 転位による結晶の塑性変形を図??に示す．glide-set 転位と同様，図??(a) のように shuffle-set 転位で [111] 方向側の全ての原子層を (A B のように 1 site 分) ズらすと図??(b) 示す構造となる．すると積層順序が AaBbCcAaBcBbCcAa となり，Aa 等の原子対単位で積層順序を議論できない．さらに橙色で示す積層欠陥面の付近のボンドが歪むことで，赤の三角形で示すように，Diamond 構造の特徴である正四面体構造が崩れている様子が伺える．以上のように直感的に，shuffle-set 転位での積層欠陥は不安定化すると示唆される．

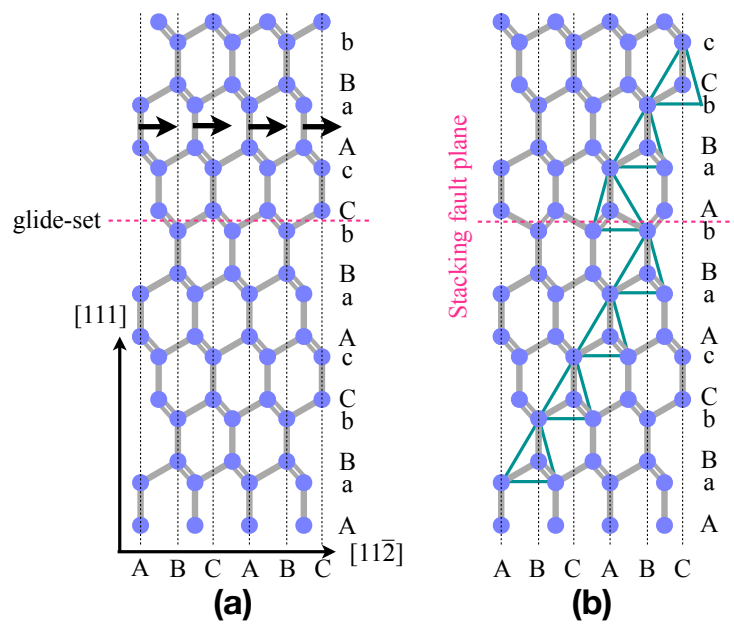


図 1.25: Glide-set 転位での積層欠陥 .

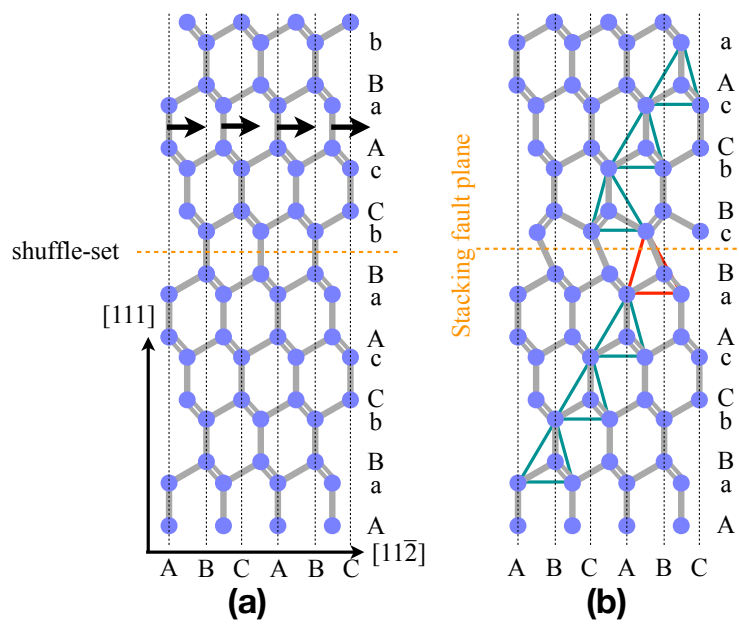


図 1.26: Shuffle-set 転位での積層欠陥 .

1.8 SiC 単結晶成長法

1.8.1 レイリー法 (Lely-method)

現在，SiC のバルク単結晶成長には，ほとんどの結晶メーカー，研究機関で Lely 法 [?] と呼ばれる物理的気相輸送法（昇華再結晶法）が用いられている．Lely 法は，準閉鎖空間内で，原料から昇華した Si と C から成る蒸気が，不活性ガス中を拡散で輸送されて，原料より低温に設定された種結晶上に過飽和となって凝結するという現象を利用したものである．図??に Lely 法の模式図を示す．黒鉛製坩堝は不活性ガスで雰囲気制御された空間内で，通常高周波により誘導加熱される．系の温度制御は，通常断熱材に開けた穴から，放射温度計により坩堝の表面温度を測定することによりなされる場合が多いが（2200～2400 ），シミュレーション等により見積もられた実際の系内の温度は 2500 以上にも達している．このように非常に高いプロセス温度がこの成長法の特徴であり，また結晶成長のプロセス制御，欠陥制御を難しくしている．

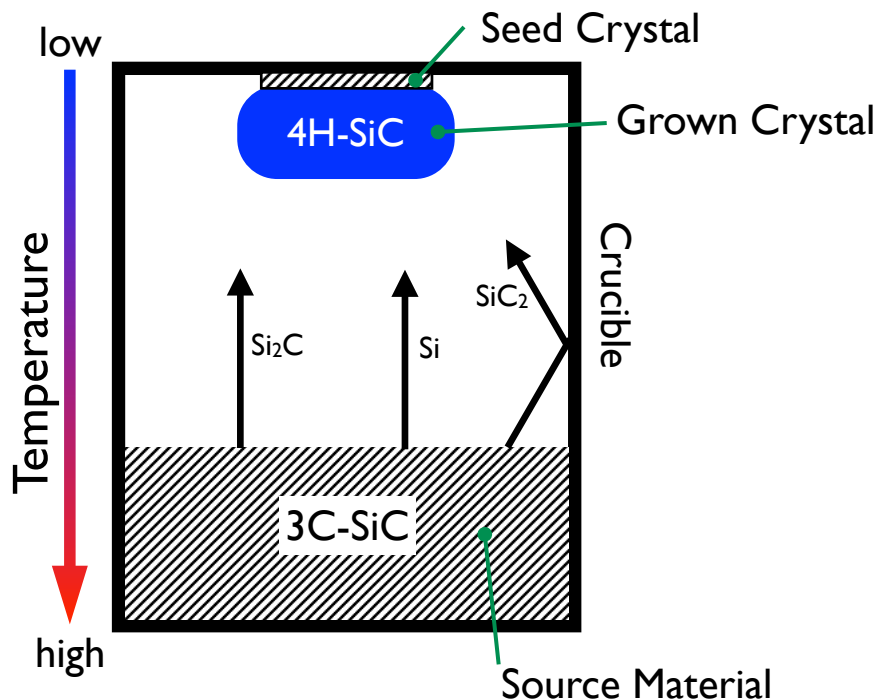


図 1.27: Lely 法による SiC 単結晶成長の模式図．

1.8.2 準安定溶媒エピタキシー (MSE : Metastable Solvent Epitaxy)

準安定溶媒エピタキシー (MSE : Metastable Solvent Epitaxy)[?] は，関西学院大学・金子らが開発した新奇な SiC 単結晶成長法である．図??に MSE 法の模式図を示す．MSE 法は，TaC 坩堝内に，原料となる 2 枚の SiC 板と，その間に溶媒となる液体 Si を配置するサンドイッチ構造を有しており，高真空雰囲気下でタングステンヒーターによって均質に加熱する．MSE 法では，原料 (feed) に 3C-SiC の多結晶を用いて，4H-SiC がエピタキシャル成長の基板 (seed) となる．坩堝を SiC エピタキシャル成長温度である 1800 程まで加熱すると，原料の SiC から Si は溶媒に溶けて溶媒として働き，C は拡散によって基板まで輸送される．溶媒の厚みは極めて薄く，数十から数百 μm である．成長プロセスにおいて，高温で数分間保持した後，冷やされる．

Lely 法では温度勾配を結晶成長の駆動力としていたが，MSE 法では濃度勾配を駆動力としている．図??を用いて，MSE の駆動力を説明する．プロセス過程を順に説明すると，まず溶媒に接している結晶面から Si と C 原子が溶け出す．そして溶媒中の C 濃度は過飽和となり，再結晶化が起こる．このとき，約 1800 というプロセス温度は，4H-SiC が安定となる相である．加えて，図??(a) の実線に示すように，4H の液相域と，固体・液体の共存する準安定域の境界線の方が，破線で示す 3C の境界線より低炭素濃度側にある．3C と 4H とで，境界線がずれる原因は，両構造の化学ポテンシャル差にある．図??(b) に示すように，3C-SiC より 4H-SiC の方が低いエネルギーを有する．そして，先の境界線の状態における SiC の化学ポテンシャル μ は，液相の自由エネルギー曲線と固相の自由エネルギー曲線の共通接線の両端になる．図??(b) では，明らかに 3C での μ_C より 4H の μ_C が低い．この化学ポテンシャル差が濃度勾配を発生させている．これらの要因により，再結晶化したときの結晶構造は 4H となる．そして 4H-SiC の結晶成長が進むと，4H-SiC 結晶周りの C 濃度が低下する．しかし溶媒中の C 濃度を均一に保つため，原料からさらに C が溶け出す．つまり図??(c) に示すように，溶媒である液体 Si 中の C 濃度は，4H-SiC 周りが低く，原料である 3C-SiC 周りが高い状態となり，この濃度勾配が MSE 法の駆動力となっている．

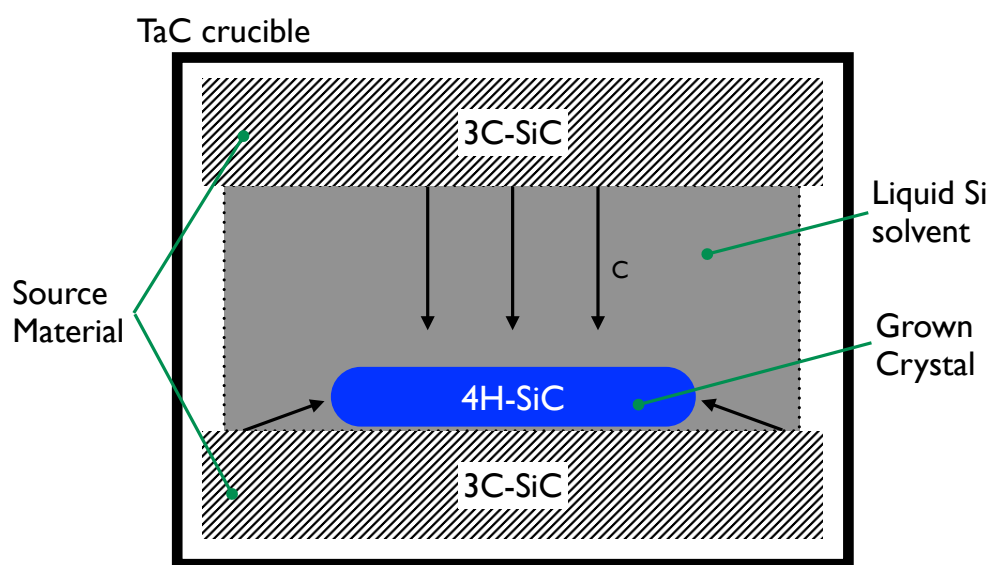


図 1.28: MSE 法による SiC 単結晶成長の模式図 .

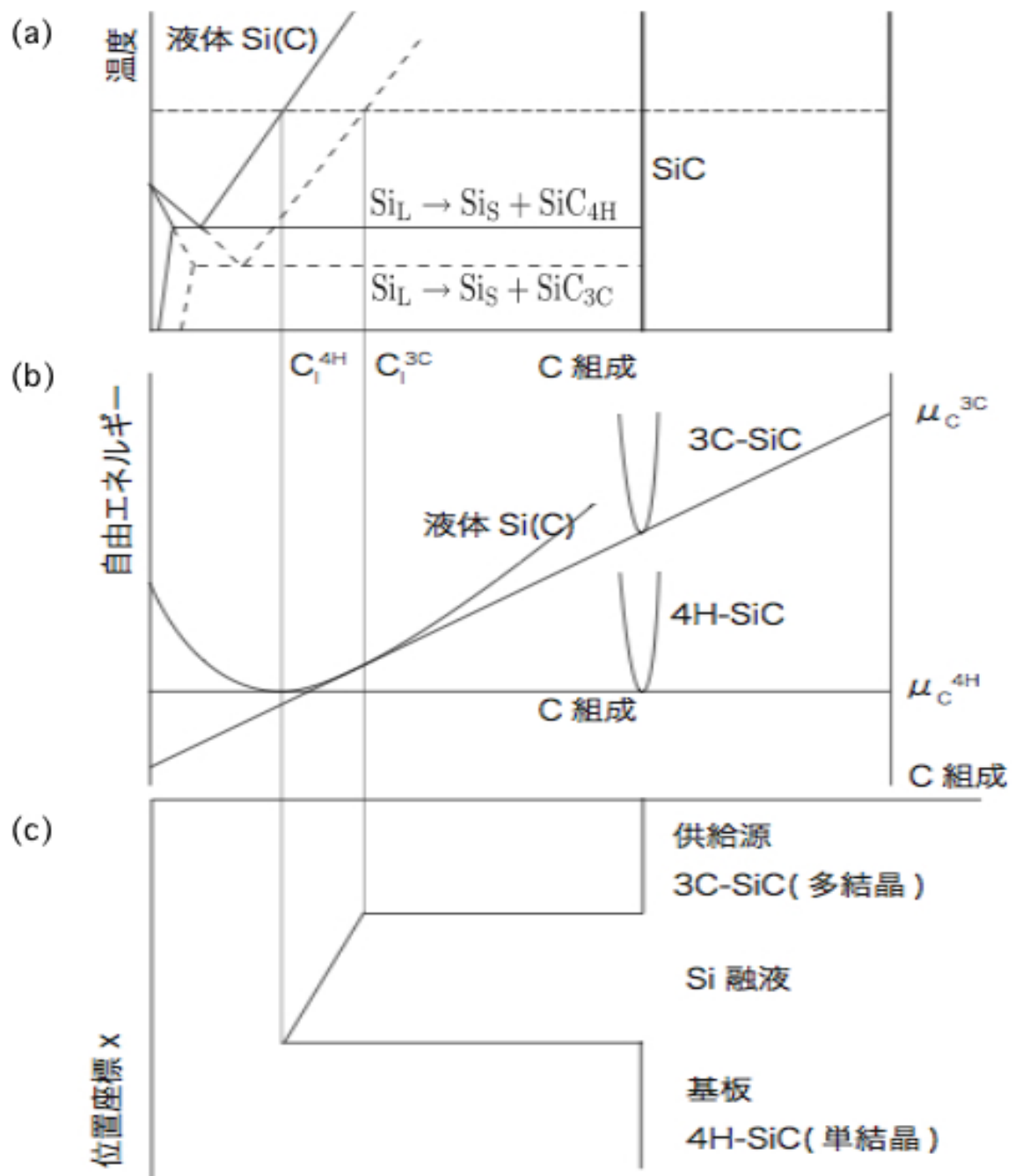


図 1.29: MSE 法の駆動力を説明する模式図．各々の図において，縦軸は図中で示す通り，横軸は C 組成とあるが C 濃度と捉えて良い．(a) 実線が 4H, 破線が 3C を示す状態図．(b) Si-C 組成・自由エネルギー図．(c) 炭素濃度プロファイル．

1.9 マイクロパイプ欠陥の生成機構

現在得られている SiC 単結晶の問題の一つにマイクロパイプと呼ばれる欠陥の存在がある．直径数 μm の中空貫通欠陥であるマイクロパイプ欠陥は，エピタキシャル薄膜成長の際に引き継がれ，デバイス，特に大電力デバイスにとって致命的な欠陥となる [?]．この欠陥は，1951 年に Frank が提唱したホローコア転位 (Hollow core dislocation)[?] を起源とする説が有力である．ホローコア転位は，転位のバーガースベクトルが非常に大きくなったために転位芯が中空状になったものである．螺旋転位の中心は，原子の結合が大きく乱れるため，その付近の原子は不安定となる．その様子を図??に示す．結合の乱れた原子が存在する領域はバーガースベクトルの大きさに比例するので，巨大なバーガースベクトルがあると中空状となる転位芯の領域が大きくなる．

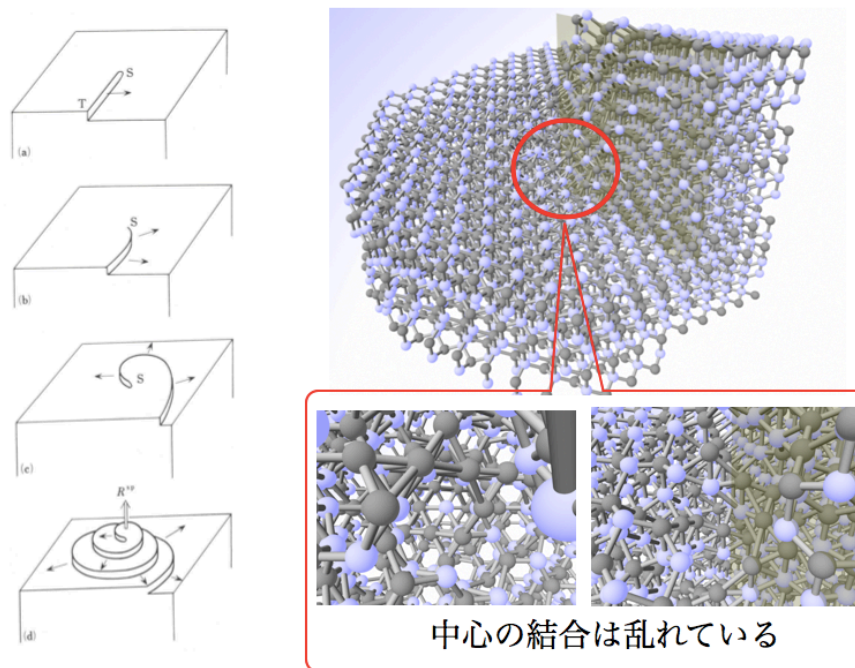


図 1.30: 螺旋転位によって，転位芯付近の原子の結合が乱れた様子．左図の (a),(b),(c),(d) は，螺旋転位の生じた表面のエピタキシャル成長の様子．

図??に 6H-SiC 単結晶の成長表面に現れたマイクロパイプ起因の渦巻き成長模様を示す (原子間顕微鏡 (AFM) 観察: $40\mu\text{m} \times 40\mu\text{m}$)．渦巻きの中心に見える黒い孔がマイクロパイプ欠陥 (巨大なバーガースベクトルを有する螺旋転位) である．渦巻きステップの高さは 13.5nm で，6H-SiC の格子定数 ($c=1.512\text{nm}$) の 9 倍に相当する．

マイクロパイプ欠陥の生成機構については幾つかのモデルが提案されている．こ

れらは2つのグループに大別される．まず一つは，表面に窪みあるいはボイドが発生し，そこに複数の転位がトラップされることによりマイクロパイプ欠陥が生成されるとするモデルである [?][?]．一方で，大きなバーガースベクトルを持つ転位が形成された後，転位芯が中空となりマイクロパイプ欠陥が安定化するというものである [?][?]．

今後，より高性能，大電力の SiC デバイスを実用化していくには，さらなるマイクロパイプ欠陥の低減が必要であり，そのためにマイクロパイプ欠陥の生成機構の解明は重要な課題の一つである．

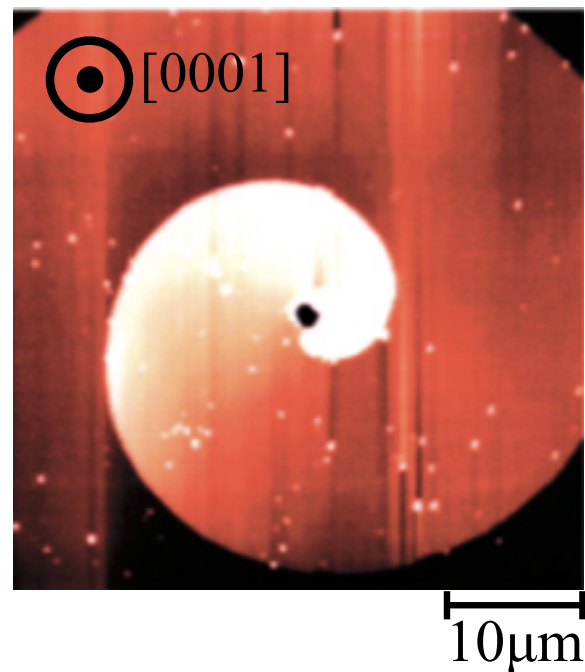


図 1.31: 6H-SiC 単結晶の成長面に現れたマイクロパイプ欠陥に起因した渦巻き成長模様（原子間顕微鏡（AFM）観察： $40\mu\text{m} \times 40\mu\text{m}$ ）．渦巻きの中心に見える黒い孔がマイクロパイプ欠陥．渦巻きステップの高さは 13.5nm で，6H-SiC の格子定数 ($c=1.512\text{nm}$) の9倍． [?]

第2章 計算原理

2.1 第一原理計算

本研究では、原子の種類だけから電子構造を求め様々な物性を予測することのできる第一原理計算 (first principles calculations) を用いた。原子は陽子、中性子、電子で構成され、陽子と中性子 (or 陽子のみ) から成る原子核の周りに電子が存在する。この外側の電子が近接原子の電子と軌道と共にして共有結合をつくる。このような電子の振る舞いを記述するのが量子力学である。そして量子力学を学ぶ上で避けられない方程式がシュレディンガー方程式である。原子番号と原子位置をパラメータとし、シュレディンガー方程式を量子力学的に正確に解くことで、系の電子構造を求め、物性の予測を行うのが第一原理計算である。原子、及び格子の入力から構築した様々なモデルのエネルギーを計算し、その値を比較評価することで理論構築を行った。なお、第一原理計算には VASP と呼ばれる密度汎関数法を用いた平面波基底・擬ポテンシャル法電子構造プログラムを用いている。この手法では、3次元周期的境界条件を満たす平面波の基底関数を用いて電子被占有の軌道を展開し、その波動関数をもとに一電子方程式を解くことにより、電子状態を求める。そもそも物理学の本質はその予測可能性にあり、座標と原子番号を与えて多電子問題のシュレーディンガー方程式を解きさえすればすべての物理力が予測可能となる。しかし、実際の物理系の本質は無限系であり常にある近似に基づいた解しか得られないことと、たとえ完全な解が得られたとしてもその解のもたらす情報はやはり無限にあり常に情報の縮約を行ってできるだけ少ないパラメータで系の本質を記述する作業が必要とされている [?]。物性予測の第一段階のプロセスは第一原理計算による物性のシミュレーションを行うところであり、密度汎関数法に基づく有効ポテンシャルを用いて電子状態を決定し物性予測を行っている。この第一原理計算は並列 PC クラスタを用いることによって、計算時間を早めることができる。詳しくは以下で説明する。

2.2 シュレディンガー方程式

大砲の玉，ボールや惑星の運動を記述するには，位置と速度が解れば可能である．このように物体は，古典力学である Newton の運動方程式に従って，その時間的な動きの変化を追いかけることができる．しかし，この方程式では，電子や格子の動きを説明できないことが解った．20 世紀初めに，電子は波でも粒子でもなく，両者の性質を兼ね備えていることがわかってきた．このような性質を持つものは物質波と呼ばれている．この物質波の動きを説明したのが式??に示したシュレディンガー方程式である．

$$-\frac{\hbar}{2m} \frac{d^2}{dx^2} \psi(x) + V(x) \psi(x) = E \psi(x) \quad (2.1)$$

式??の E はエネルギー準位 (energy level)， $\psi(x)$ は波動関数 (wavefunction) である． $V(x)$ は座標 x における粒子が感じるポテンシャルエネルギーである． \hbar は $h/2\pi$ (h : Planck 定数) であり， $\psi(x)$ は古典的な粒子の軌跡の概念にとってかわって，空間に分布している波を指し，その 2 乗が物質波の存在確率を表すと解釈されている．第一原理計算ではシュレディンガー方程式の固有値を求めることにより，トータルエネルギーが算出される．

しかし，シュレディンガー方程式は水素原子を用いる時にしか解けず，他の原子を用いる場合，厳密な解を算出できない．従って，その場合は近似解を求めることになる．問題はポテンシャル $V(x)$ が波動関数 $\psi(x)$ の関数であることである．これは電子の分布が変化すると，ポテンシャルもその影響を受けることを意味している．つまり， $V(x)$ を計算するためには， $\psi(x)$ を計算する必要があり， $\psi(x)$ を計算するためには $V(x)$ の計算が必要となり，ループしてしまう．従って，第一原理計算には，SCF (self-consistent-field: セルフコンシステントフィールド，自己無撞着) と呼ばれる方法が用いられる．SCF では，はじめに解に近いと思われる適当な $\psi(x)$ とエネルギーで計算結果とするのが，現在のほとんどのアルゴリズムの骨格になっている．もちろん，中身には様々な工夫があり，上記のように単純でない．この SCF 計算の流れを図??に示す．図??の 1 サイクルをイタレーションという．図??は密度汎関数法の例なので， ρ を引き合いにしてイタレーションをしている．最後の収束条件は別に ρ でなくてもよく，系のトータルエネルギーが落ち着いたり，イオンに働く力が落ち着いたりすることによって計算が終わる．このような感じで，収束するまでループが回り，このループをセルフ・コンシステント・ループと呼ぶ．

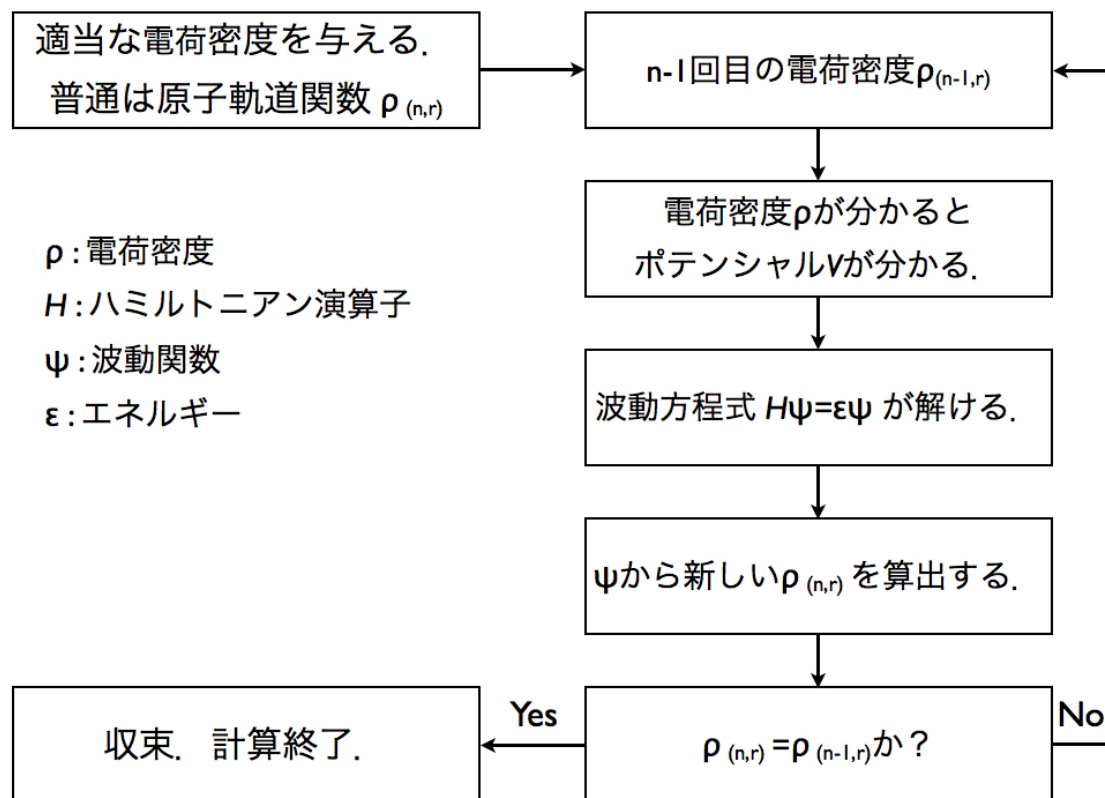


図 2.1: SCF 計算の模式図

2.3 ポテンシャル・波動関数・エネルギー

ここではポテンシャル，波動関数，エネルギーの意味について論じる．シュレディンガー方程式でも含まれるポテンシャル ($V(x)$) は座標 x において，粒子が感じるポテンシャルエネルギーである．また波動関数 $\psi(x)$ は，そのものの 2 乗 ($|\psi(x)|^2$) が粒子の存在確率を示す．

ではエネルギーについて，最も簡単な例である，井戸型ポテンシャルに閉じ込められた電子をもって考える．図??に井戸型ポテンシャルと波動関数，波動関数の 2 乗を模式的に示した．

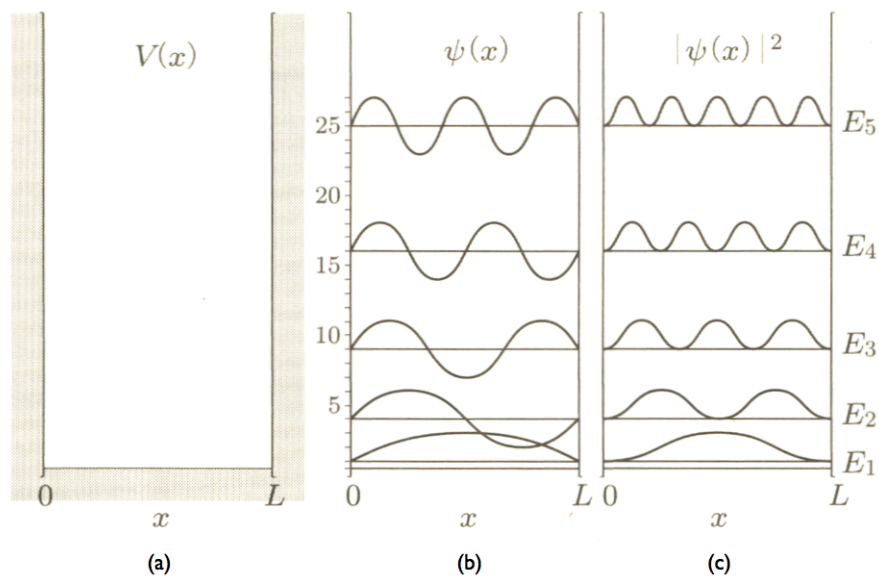


図 2.2: (a) 井戸型ポテンシャル $V(x)$ と (b) 波動関数 $\psi(x)$ および (c) $|\psi(x)|^2$ を示す模式図．それぞれのエネルギー準位 ($E_1 \dots E_5$) に対応した高さに記している．

図??の (a) に関して，無限の高さをもった壁の底にある水の定在波，または両端を固定された弦の動きを考える．この場合のポテンシャルは

$$V(x) = \begin{cases} 0 & (0 \leq x \leq L) \\ \infty & (\text{その他}) \end{cases} \quad (2.2)$$

で与えられる．したがって，式 (??) は

$$-\frac{\hbar}{2m} \frac{d^2}{dx^2} \psi(x) = E \psi(x) \quad (2.3)$$

と簡単になる．この微分方程式の一般解は， A ， B を定数として

$$\psi(x) = A \sin kx + B \cos kx \quad (2.4)$$

で与えられる．この2次微分をとると

$$\frac{d^2}{dx^2}\psi(x) = -Ak^2 \sin kx - Bk^2 \cos kx \quad (2.5)$$

となり， k は

$$\frac{\hbar^2}{2m}k^2 = E \quad (2.6)$$

より求められる．両端が固定されているため，

$$\psi(0) = 0 \quad \text{より} \quad B = 0 \quad (2.7)$$

$$\psi(L) = 0 \quad \text{より} \quad kL = n\pi \quad (2.8)$$

となる．これらより，系がとり得るエネルギー E_n は，

$$E_n = \frac{\hbar^2}{2m} \frac{n^2\pi^2}{L^2} = \frac{h^2}{8mL^2} n^2 \quad (2.9)$$

に制限される．古典的な系は，任意のエネルギーをとることができるが，量子的な系は，ある決まったとびとびのエネルギーしかとることができない．これを系のエネルギー準位 (E_1, E_2, E_3, E_4, E_5) という．図??の (b)，(c) には，こうして得られた波動関数 $\psi(x)$ とその2乗 $|\psi(x)|^2$ を，それぞれのエネルギー準位に対応した高さに示した．

第一原理計算では，シュレディンガー方程式 (式??) に電子が感じるポテンシャルを入れ，電子のエネルギー準位と波動関数とを計算する．

2.4 ばねモデル

図??(a) の上段は，A，B 原子のまわりにある球対称のポテンシャルによって引きつけられた電子の軌道 (ψ_A, ψ_B) を示す．また (b) にそれぞれの電子エネルギー準位 (E_A, E_B) を示す．この2原子同士を近づけてくると電子の波動関数は重なり合い，新たな軌道 ((a) の下段) とそれに対応した準位を作る．これを結合準位 E^+ と反結合準位 E^- と呼ぶ．電子がそれぞれ1個しかない場合には，遠く離れた自由電子でのエネルギー準位と，近づくにつれて下がってくる結合軌道のエネルギー準位との差分が，分子の原子間の結合力となる．(c) はこのエネルギーの変化を模式的に示した図である．2原子が十分離れた距離でのエネルギーを基準としてゼロとすると，2原子を近づけると結合準位の低下にともなって，ある距離までエネルギーは下がってくる．近づきすぎると電子同士が重なり，反発する力が強くなる．するとエネルギーは急激に上昇する．

固体でも、2原子分子と同様の相互作用エネルギーの距離依存性が存在する．エネルギーが最も低い距離を平衡原子間距離，その時のエネルギーの値を凝集エネルギーという．また，結合エネルギー曲線の2次微分 (曲率) から固体の固さが求まる．この近傍の曲率は，ほぼ2次曲線で近似できる．よって，最も単純な固体のモデルとして図??に示すばねモデルが得られる．

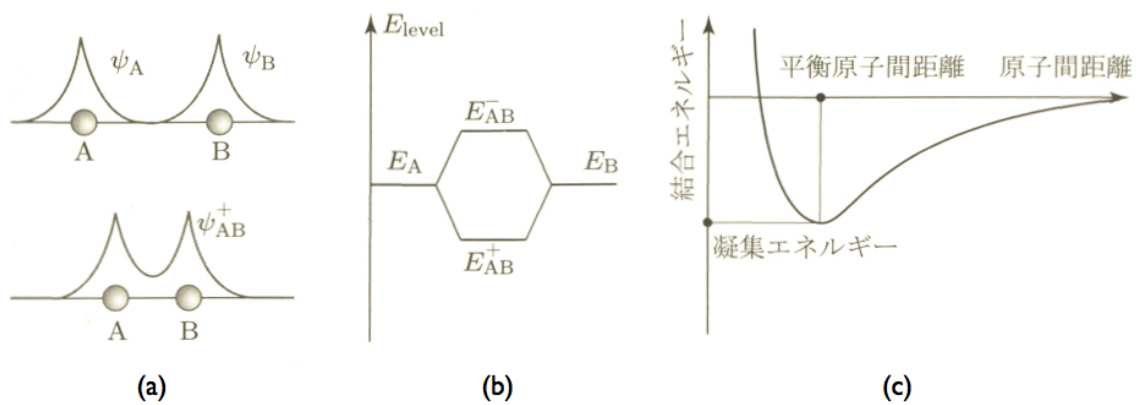


図 2.3: 2 原子分子の (a) 波動関数 , (b) エネルギー準位 , (c) 相互作用エネルギーの距離依存性 .

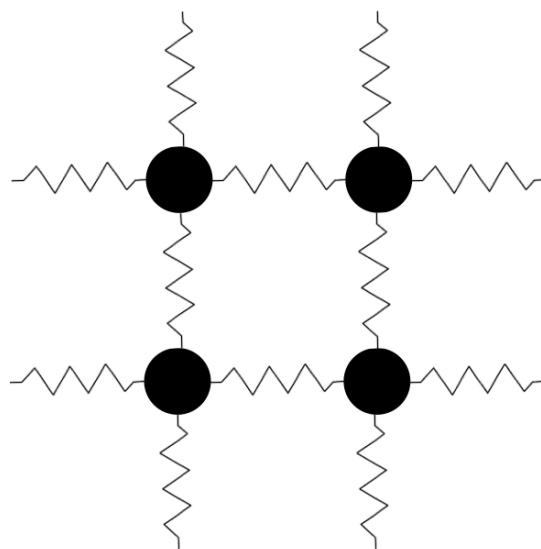


図 2.4: 固体の原子のばねモデル .

2.5 密度汎関数理論

密度汎関数理論 (DFT : Density Functional Theory) [?] は電子系のエネルギーなどの物性を電子密度から計算することが可能であるとするのが真意である。

このような計算が原理的に可能であることは1964年にヴァルター・コーンとピーエール・ホーヘンベルクによって示された。1965年にヴァルター・コーンとリュウ・シャムによりそれに基づいた実際の計算手法が示され応用が可能となった。

電子 N 個からなる系において、それらの電子に対する外部ポテンシャルが決まっているとする (例えば分子の原子核の配置が決まれば、それらの原子核が電子に及ぼす静電ポテンシャルは決まる。) するとその外部ポテンシャルから導かれるハミルトニアン H のシュレディンガー方程式を解けば、その外部ポテンシャルのもとで許される電子系の波動関数とそのエネルギー E の組が求まる。特に求めたい状態を許される状態の中で最もエネルギーの低い基底状態に限定し、基底状態に縮退がないとすれば (電子数と外部ポテンシャル)-(波動関数)-(エネルギー) は1対1に対応する。

一方、ある系の電子密度 ρ が決まると、それを基底状態とする外部ポテンシャルがもし存在するとすればただ1通りに定まる。(ホーヘンベルク・コーンの第1定理)。また電子数 N も電子密度を全空間に渡って積分することで求めることができる。すると先ほどの手順を踏むことで (電子密度)-(波動関数)-(エネルギー) も1対1に定まる。すなわち、ある電子系の波動関数やエネルギーは関数 ρ の関数、すなわち電子密度の汎関数である。

電子数が N となるような適当な電子密度 ρ' を仮定する。するとこの電子密度に対応する基底状態の波動関数 Ψ' が存在するとすればただ1つ定まる。この電子密度に対応する状態のエネルギーの期待値 $\langle E' \rangle$

$$\langle E' \rangle = \int \Psi'^* \hat{H} \Psi' dv \quad (2.10)$$

は真の基底状態のエネルギーよりも必ず大きい (ホーヘンベルク・コーンの第2定理)。よって電子密度についても変分原理が成立し、電子密度関数を変化させて最小のエネルギーを与える電子密度を探索することで、電子系の基底状態のエネルギーを求めることができる。

密度汎関数理論に基づいたこのような方法で電子系のエネルギーを求める方法を密度汎関数法という。3次元空間内の電子 N 個の系の波動関数は各電子について3個、合計 $3N$ 個の座標変数に依存する関数となる。一方、電子密度は電子が何個になろうとも3個の座標変数に依存するだけであり、取り扱い易さに雲泥の差がある。そのため、密度汎関数法は多電子系の物性を求める計算化学の手法として良く用いられている。

つまり、基底状態に関しては、量子力学で規定されるシュレディンガー方程式と等価である。このことから、密度汎関数理論に基づく電子状態計算は第一原理計算と呼ばれている。

2.6 擬ポテンシャル法

本研究では，第一原理計算を行う上で，擬ポテンシャル法としてPAW(Projector Augmented Wave) 法を用いた．擬ポテンシャル法には，フルポテンシャル（全電子），PAW ポテンシャル，擬ポテンシャル（ウルトラソフト型）の3つに分類され，その特徴を表??に示す．PAW ポテンシャル[?] は，Blochl が考案した全電子計算法であり，フルポテンシャルの精度と擬ポテンシャルの計算速度を兼ね備えた方法である．各々のポテンシャルの模式図を図??に示す．通常ポテンシャル V

表 2.1: 擬ポテンシャル法とフルポテンシャル法の比較．

フルポテンシャル	精度が高い 全元素対応 × 計算時間がかかるため，小さな系のみ × 原子半径等，パラメータ設定に熟練が必要
PAW ポテンシャル	フルポテンシャルの精度を維持しながら計算時間を軽減 全元素対応
(ウルトラソフト型) 擬ポテンシャル	計算時間を軽減 × アルカリ金属，アルカリ土類，希土類に難

は，無限に深いものであると考える．そのポテンシャルを計算する際，擬ポテンシャル法では，図??(a) に示す赤線の様なポテンシャルを仮想して計算する．フルポテンシャル法では，図??(c) に示す緑線の様に奥底まで考慮して計算する．そのため表??のような特徴が表れる．PAW 法では，図??(b) に示す青線の様に擬ポテンシャルとフルポテンシャルの中間のポテンシャルを用いて計算する．よって計算精度を維持しながら計算時間の短縮を図れる．

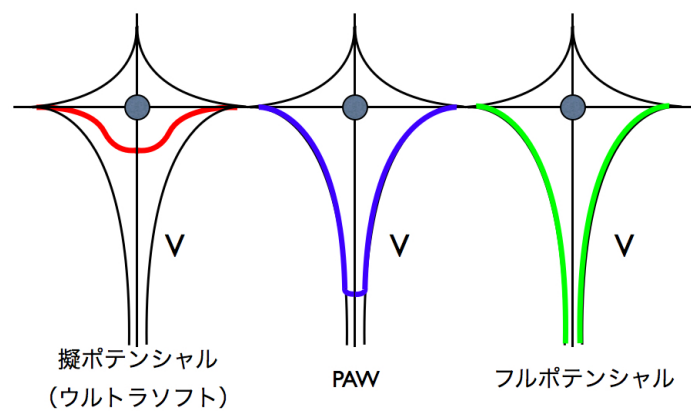


図 2.5: ポテンシャルの計算における各手法の模式図 .

2.7 交換相関関数

??で電子系の計算を行うには、電子交換と電子相関に由来するポテンシャル V_{xc} (交換相互作用ポテンシャル) の厳密な表式が与えられていない。この項は、本質的に電子と電子の相互作用や、自分自身の電子との相互作用もあり表式が難しいが、現在はいろいろな近似法が提案されている。それが LDA(local density approximation) や GGA(generalized gradient approximation) と言われるポテンシャルである。LDA は、交換・相関項を局所的に一樣な電子の電荷密度による交換相関エネルギー ϵ_{xc} の関数形で記述する近似である。交換・相関項 $E_{xc}[n]$ の厳密な表式を得ることは多体効果が絡むため不可能であり、式??のような近似として求められる。

$$E_{xc}[n] = \int \epsilon_{xc}(n(r))n(r)dr \quad (2.11)$$

ここで、式??の右辺の $n(r)$ は一樣な電荷密度(電子密度)とする。同じく右辺の ϵ_{xc} は電子密度 $n(r)$ から局所的に得られるとする。LDA は経験的にわりと現実を反映する(計算と実験が整合する)ので、多々用いられてきた。しかし、平均かした電子密度を使うので注目する電子自身との相互作用を計算してしまう等の問題がある。また電子・電子の相互作用が大きくなってくると計算結果がズれてくる傾向がある。このような問題を克服する手段として、GGA が提案・試行されている。

GGA は、電子状態計算で用いられる局所密度近似を超える試みの一つである。密度汎関数法では電子間の相互作用である交換相関項は電荷密度で表現される。その電荷密度を一樣な電子ガスとして解かれた表式に利用するのが局所密度近似である。現実の電荷密度の分布は一樣でないので、この一樣とする近似を超えて電荷密度の勾配の効果を導入することにより局所密度近似の精度を上げようとする試みは古くからあった。1985 年、Perdew 等による改良により精度が向上し、実際のバンド計算にも利用されるようになった。この Perdew 等による改良版とそれ以降の派生版が GGA と呼ばれている。GGA により、系の凝集エネルギー等の精度が改善された。GGA には、PW91, PBE, B3LYP など幾つかの派生版が存在する。LDA や GGA の他にも LDA+U や GW 等があるが、本書では割愛する。

2.8 VASP(Vienna Ab-initio Simulation Package)

第一原理計算ソフト VASP[?] は平面波・擬ポテンシャル法(ならびに PAW 法)による第一原理バンド計算プログラムである。第一原理計算はシュレディンガー方程式の固有値を求めることにより、トータルエネルギーを算出する。平面波基底の重ね合わせで波動関数を表現し、密度汎関数理論に基づいて電子状態の計算を行う。平面波を使用する利点として、その系の原子にかかる力の計算を正確かつ高速に行える点が挙げられる。このことから、VASP は構造最適化や第一原理

分子動力学計算のツールとして幅広く用いられている．また，擬ポテンシャル法により内殻電子をポテンシャルに置き換えて取り扱うので，波動関数の表現に用いる平面波基底の数を大幅に減らし，計算量を軽減する．内殻電子の取り扱いについては，擬ポテンシャル法の他に，全電子計算 PAW 法を採用しており，擬ポテンシャル法と比べさほど計算量を増やすことなく，精度を上げることができる．バルク構造，表面，界面など広範に渡る問題に適用できる汎用的なソフトウェアである．次小節以降に VASP に必要な入力ファイルや設定，計算条件について簡単に記述する．

2.8.1 INCAR

INCAR は VASP におけるコアな入力ファイルであり，主に VASP の計算条件を設定するファイルである．以下に INCAR のサンプルを記載させている．これらのタグの入力に応じ，計算条件は大きく変わる．本論では，研究を進めるにあたって，頻繁に変更したタグについていくつか取り上げ説明する．

— INCAR —

```
# SCF input for VASP
# Note that VASP uses the FIRST occurrence of a keyword
SYSTEM = Diamond-Si_unitcell_fix_toga

PREC = Accurate
ENCUT = 1000
IBRION = 2
NSW = 100
ISIF = 3
ALGO = Normal (blocked Davidson)
NELM = 60
NELMIN = 2
EDIFF = 1.0e-05
EDIFFG = -0.01
VOSKOWN = 1
NBLOCK = 1
ISPIN = 1
INIWAV = 1
ISTART = 0
ICHARG = 2
LWAVE = .FALSE.
LCHARG = .FALSE.
ADDGRID = .FALSE.
ISMEAR = 1
SIGMA = 0.2
LREAL = .FALSE.
RWIGS = 1.11
```

PREC 計算精度を設定しているタグの一つである。入力として Low/Medium/Normal/Accurate 等がある。なお、Normal と Accurate は VASP4.5 以降の ver. でのみ使用できる。構造最適化を目的とした計算を行うなら Accurate とするのが望ましい。

ENCUT 計算する際、考慮する平面波の範囲を設定するタグである。一般に、カットオフエネルギー (Cutoff Energy) と呼ばれる値を入力する。通常は [eV] 単位の値を記述する。上記の INCAR ファイルの場合、Cutoff Energy は 1000[eV] という入力になる。入力値が大きい程、短い波長の平面波を考慮し、より精密な計算を行える。

NBANDS バンド数を設定するタグである。通常、記述しなくても適当な値をとって計算される。しかし、大きな系で計算する際、この値が原因となるエラーがある。そのエラーの解決は単純に数値を大きくするだけだが、計算に時間がかかる。最適なバンド数を指定するには、VASP の出力ファイルである OUTCAR を確認すれば良い。エラーが起きようと起きまいと OUTCAR 中に計算に必要なバンド数が記載されており、その値をヒントにバンド数を指定すれば良い。

IBRION 原子の緩和 (relax) の手法を設定するタグである。表??に IBRION タグへの入力値とそれに対応した緩和手法をまとめた。構造最適化を行わず計算 (fix で計算) したい場合は IBRION=-1 とすると良い。

表 2.2: IBRION の入力値と緩和手法の対応表

IBRION の入力値	緩和の手法
IBRION = 0	分子動力学法 (MD : molecular dynamics)
IBRION = 1	準ニュートン法 (quasi-Newton)
IBRION = 2	共役勾配法 (conjugate-gradient)
IBRION = 3	最急降下法
IBRION = -1	緩和をしない (fix で計算)

ISIF 応力テンソルをどのように計算するかを決めるタグである．force や応力テンソル，イオンの緩和となる内部緩和，格子の空間の形や体積の変化を考慮するかを設定する．入力値に対する計算条件を表に示す．応力テンソルの計算は比較的時間がかかる．

内部緩和のみ行う場合．

IBRION = 2, ISIF = 2

内部・外部緩和を行う場合（西谷研究室では fullrelax と呼んでいるが一般呼称では無い）．

IBRION = 2, ISIF = 3

表 2.3: ISIF(0 ~ 7) による相違点

ISIF	calculate force	calculate stress tensor	relax ions	change cell shape	change cell volume
0	yes	no	yes	no	no
1	yes	trace only	yes	no	no
2	yes	yes	yes	no	no
3	yes	yes	yes	yes	yes
4	yes	yes	yes	yes	no
5	yes	yes	no	yes	no
6	yes	yes	no	yes	yes
7	yes	yes	no	no	yes

NSW イオンの緩和計算のステップ数の上限を設定するタグである．この入力値が小さいと収束せずに計算が止まってしまう可能性がある．例えば，NSW=100 で計算し，緩和ステップが 100 で止まっている場合，最適な構造緩和の余地があり，十分な計算ができていないことになる．逆に，10 ステップで収束した場合，そこで計算は打ち切られる．そのため，この値は大きくすることを勧める．

fix(構造緩和しない) で計算する場合，NSW = 0 と設定することを強く勧める．fix 計算で NSW=100 としていると，同じ fix 計算を 100 回繰り返してしまうためである．

NELM 小節??で記述したセルフコンシステントループのループする回数の上限を設定するタグである．この入力値が小さいと NSW と同様に，十分に収束しないまま計算が打ち切られる．計算が収束すると，上限に達せずとも計算は修了する．(default=60)

NELMIN セルフコンシステントループのループする回数の上限を設定するNELM
対して、下限を設定するタグである。(default=2)

ISMear 波動関数をどのような手法で表示するのかを設定するタグである。0
でガウスモデル，-1 でフェルミモデル，1 以上は Methfessel-Paxton 法を用いる。
金属のリラクゼーションを考慮する場合は 1 か 2 を用いるが，この二つは比較的
同じような解を算出する。(default=1)

EDIFF 電子計算の際，どの程度の差で計算を終了するかを設定するタグである。
イタレーションごとに前の結果とのエネルギー差を算出し，その値がEDIFFで指
定した値以下になれば，計算が終了する。よって，より細かなデータが必要な時
にこの値を小さくすればよい。(default= 10^{-4})

2.8.2 POSCAR

POSCAR は計算させる原子モデルのデータを格納する入力ファイルである。以
下は実際に計算で用いた 4H-SiC の POSCAR である。POSCAR には，入力され
た格子空間の倍率，格子空間を指定する基本並進ベクトル (Primitive Vector)，原
子数，格子空間に対する原子の相対座標が記述されている。

```
POSCAR
4H-SiC
1.0000000000000000
3.0935700000000000 0.0000000000000000
0.0000000000000000
-1.5467850000000000 2.6791102083854259
0.0000000000000000
0.0000000000000000 0.0000000000000000
10.1287000000000000
4 4
Direct
0.0000000000000000 0.0000000000000000 0.0000000000000000
0.0000000000000000 0.0000000000000000 0.5000000000000000
0.3333333333333333 0.6666666666666667 0.2500000000000000
0.6666666666666667 0.3333333333333333 0.7500000000000000
0.0000000000000000 0.0000000000000000 0.1874200000000000
0.0000000000000000 0.0000000000000000 0.6874200000000000
0.3333333333333333 0.6666666666666667 0.4374200000000000
0.6666666666666667 0.3333333333333333 0.9374200000000000
```

以降，POSCAR に含まれる数字の意味を説明する。

まず格子空間を設定する。

格子空間の設定

```
4H-SiC
1.0000000000000000
3.0935700000000000 0.0000000000000000
0.0000000000000000
-1.5467850000000000 2.6791102083854259
0.0000000000000000
0.0000000000000000 0.0000000000000000
10.1287000000000000
```

1行はタイトルなので何でも良い．2行目が格子空間の倍率を設定する数字である．例では格子空間の倍率が1.0とあるが，ここが0.5の場合，格子空間のa,b,c軸の全てを0.5倍することになる（3～5行目にある Primitive Vector を全て0.5倍する）．

3～5行目に記された数字が格子定数より決まった Primitive Vector である．上から a,b,c 軸を表すベクトルである．

4H-SiC の格子定数は

$a : b : c = 3.09357 : 3.09357 : 10.1287$ [Angst.]

$\alpha : \beta : \gamma = 90 : 90 : 120$ [度]

続いて，原子数を設定する．

原子数の設定

```
4 4
Direct
```

1行目が原子数の入力となる．モデル中に含まれる原子数を入力する．この例では，Si が4個，C が4個なので上記のように設定している．単元素で構成しているモデルなら，4 4でなく8といったように記述する．ここで注意すべき点は，4 4の内，どちらがSiでどちらがCなのかをわかっている必要がある．例えば，Siを5個，Pを1個のモデルの場合，5 1となるか1 5となるかがわからない．これは後述する POTCAR に記されている原子のポテンシャルのデータの並びに対応している．例えば，POTCAR 内に，Si のポテンシャル，P のポテンシャルの順で記述されていれば，原子数は5 1と記述する．

最後に，原子の座標について説明する．

— 原子のポジション —

0.0000000000000000	0.0000000000000000	0.0000000000000000
0.0000000000000000	0.0000000000000000	0.5000000000000000
0.3333333333333333	0.6666666666666667	0.2500000000000000
0.6666666666666667	0.3333333333333333	0.7500000000000000
0.0000000000000000	0.0000000000000000	0.1874200000000000
0.0000000000000000	0.0000000000000000	0.6874200000000000
0.3333333333333333	0.6666666666666667	0.4374200000000000
0.6666666666666667	0.3333333333333333	0.9374200000000000

上記の座標点 (x,y,z) の通り，左から x,y,z の数値を表す．またこの座標は格子空間に対する相対座標である．上記の原子数の記述が Si が 4 個，C が 4 個を示す場合，上記 4 行の座標点が Si の座標，下記 4 行の座標が C の座標となる．3 行目の $(1/3, 2/3, 1/4)$ を例に説明すると， a 軸に対し， $1/3$ 倍， b 軸に対して $2/3$ 倍， c 軸に対して $1/4$ 倍の位置に原子があることを示す．つまり相対座標 $(1/3, 2/3, 1/4)$ 点は，原点 $(0, 0, 0)$ に対し， $(0.773, 1.786, 2.532)$ 点を表す．3 つの Primitive Vector をそれぞれ P_0, P_1, P_2 とすると相対座標 (x, y, z) に対し，原点からの座標 (X, Y, Z) は以下のように求まる．

$$P_0 = \begin{bmatrix} p_{0,0} \\ p_{0,1} \\ p_{0,2} \end{bmatrix}, P_1 = \begin{bmatrix} p_{1,0} \\ p_{1,1} \\ p_{1,2} \end{bmatrix}, P_2 = \begin{bmatrix} p_{2,0} \\ p_{2,1} \\ p_{2,2} \end{bmatrix} \quad (2.12)$$

$$B = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.13)$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = P_0 x + P_1 y + P_2 z \quad (2.14)$$

また構造緩和計算を行う上で，POSCAR で必要な記述もある．VASP ではモデル中の原子で，緩和させる原子と緩和させない原子を，また緩和させる方向を指定できる．その例を以下に示す．まず 7 行目に Selective dynamics という記述を追加する．そして原子の相対座標の右端に緩和させるか否か，そして緩和させる方向を記述する． a, b, c 軸に対し緩和させるなら T と，緩和させないなら F と記述する．例の 11 行目の $(1/3, 2/3, 1/4)$ を例に説明すると，右に F F T という記述が追加されている．これはこの原子を c 軸方向のみ緩和させるという意味となる．そして他の原子には，全て F F F と記述されているので， $(1/3, 2/3, 1/4)$ 点の原子のみ c 軸方向に限り緩和し，他の原子は緩和しないという POSCAR になる．

緩和する原子を指定する POSCAR

```

4H-SiC
  1.0000000000000000
    3.0935700000000000    0.0000000000000000
0.0000000000000000
  -1.5467850000000000    2.6791102083854259
0.0000000000000000
    0.0000000000000000    0.0000000000000000
10.1287000000000000
   4   4
Selective dynamics
Direct
  0.0000000000000000  0.0000000000000000  0.0000000000000000 F F
F
  0.0000000000000000  0.0000000000000000  0.5000000000000000 F F
F
  0.3333333333333333  0.6666666666666667  0.2500000000000000 F F
T
  0.6666666666666667  0.3333333333333333  0.7500000000000000 F F
F
  0.0000000000000000  0.0000000000000000  0.1874200000000000 F F
F
  0.0000000000000000  0.0000000000000000  0.6874200000000000 F F
F
  0.3333333333333333  0.6666666666666667  0.4374200000000000 F F
F
  0.6666666666666667  0.3333333333333333  0.9374200000000000 F F
F

```

2.8.3 POTCAR

計算モデルで使用する元素のポテンシャルを明記したファイルである．例として，4H-SiC の POTCAR を図??に示す．少々節??で，記した通り，POSCAR の原子数の記述と POTCAR のポテンシャルの記述順は対応している．図??では，先に Si のポテンシャル，後に C のポテンシャルが記されている．そのため，原子数 4 4 の内，前者の 4 は Si が 4 個を示し，後者の 4 は C 個を示す．POTCAR は，非常に大きなファイルなので，元素の確認は困難となるが，コマンドプロンプト（Mac ではターミナル）で，コマンドラインに unix コマンドの "grep 'Si' POTCAR" と打ち込めば容易に検索できる．またカットオフエネルギーの値も POTCAR の中に存在する．??で記述したが，INCAR で直接的にカットオフエネルギーの値を指定することが出来る．ファイルサイズの小さい INCAR で操作する方が容易である．

```
PAW_PBE Si 05Jan2001
4.0000000000000000
parameters from PSCTR are:
VRHFIN =Si: s2p2
LEXCH = PE
EATOM = 103.0669 eV, 7.5752 Ry
.
.
.
-.631516436073E-01 -.757171941354E-01 -.896411710603E-01 -.105014918512E+00 -.121930338317E+00
-.140479159189E+00 -.160752140242E+00 -.182838197398E+00
End of Dataset
PAW_PBE C 08Apr2002
4.0000000000000000
parameters from PSCTR are:
VRHFIN =C: s2p2
LEXCH = PE
EATOM = 147.1560 eV, 10.8157 Ry
.
.
.
.647658834662E+00 .716950416187E+00 .789437621611E+00 .864907568857E+00 .943081468528E+00
.102360696409E+01 .110605024916E+01 .118988811168E+01
End of Dataset
```

図 2.6: POTCAR．赤破線枠の箇所で元素を確認できる．青破線枠がポテンシャルデータの終端を表す記述です．2 元素以上のモデルの POTCAR の中身では，一つ目の元素のポテンシャルが記述され End of Dataset で終端した後にはすぐ次の元素のポテンシャルが記述されている．

2.8.4 OUTCAR

OUTCAR は計算終了後に作成されるファイルである。OUTCAR には、計算結果が出力されており、OUTCAR のデータは VASP 計算にとって非常に重要となる。少々節々の NBANDS でも記した通り、必要なバンド数が記述されている。また原子にかかるフォースの大きさや計算時間等も出力される。そしてエネルギーはもちろん、構造緩和計算を行ってれば緩和後の格子定数、原子のポジションも出力される。図 2.7 に 4H-SiC の計算を行った OUTCAR を例として示す。OUTCAR は非常に大きなファイルであるが、計算結果はファイルの底部に記述されている。

```

FREE ENERGIE OF THE ION-ELECTRON SYSTEM (eV)
-----
free energy TOTEN = -60.266777 eV

energy without entropy= -60.268783 energy(sigma->0) = -60.267446
.
.
.
VOLUME and BASIS-vectors are now :
-----
energy-cutoff : 1000.00
volume of cell : 83.95
direct lattice vectors      reciprocal lattice vectors
3.093570000 0.000000000 0.000000000 0.323251131 0.186629127 0.000000000
-1.546785000 2.679110208 0.000000000 0.000000000 0.373258255 0.000000000
0.000000000 0.000000000 10.128700000 0.000000000 0.000000000 0.098729353

length of vectors
3.093570000 3.093570000 10.128700000 0.373258255 0.373258255 0.098729353
.
.
.
POSITION      TOTAL-FORCE (eV/Angst)
-----
Si { 0.00000 0.00000 0.00000 0.000000 0.000000 -0.065260
     0.00000 0.00000 5.06435 0.000000 0.000000 -0.065260
     0.00000 1.78607 2.53218 0.000000 0.000000 -0.045876
     1.54678 0.89304 7.59652 0.000000 0.000000 -0.045876
C { 0.00000 0.00000 1.89832 0.000000 0.000000 -0.039491
     0.00000 0.00000 6.96267 0.000000 0.000000 -0.039491
     0.00000 1.78607 4.43050 0.000000 0.000000 0.150628
     1.54678 0.89304 9.49485 0.000000 0.000000 0.150628
total drift:      0.000000 0.000000 0.000468

```

図 2.7: OUTCAR。赤端線枠に緩和後の Primitive Vector が記述される。POSITION については POSCAR の原子座標と対応しており、ここでは上記 4 つが Si の、下記 4 つが C の座標を示す。またこの座標は原点 (0, 0, 0) からの座標であり、Primitive Vector に対する相対座標でないことに注意する。

2.8.5 KPOINTS

k 点メッシュは 3 次元空間で表される．その k 点メッシュの中でも最も簡単な例である，正方格子における k 点メッシュ，及び k 点を図??に示す．黒点が，実空間における面に対応する逆格子点（実空間における表面の面方位や面間隔といった情報を保持している）を表しており，中心の逆格子点（ Γ 点）から第一近接の逆格子点に線分を引き，その線分の垂直二等分線に囲まれた領域をブリルアンゾーンという．そのブリルアンゾーンを等分割するメッシュを k 点メッシュといい，そのメッシュ上の交点を k 点という．

密度汎関数法を用いて全エネルギーと電荷密度の計算をするには，ブリルアンゾーン領域全体での積分を必要とする．しかし通常のバンド計算法では，有限の k 点での回を重み付けして積算することでこれらの解に近似する．また使用する k 点の数から可能な最大限の精度を得るために，最適な k 点を選択する Monkhorst-Pach の特殊点法を採用する．Monkhorst-Pach 法は，先述したように，逆格子空間を一定のメッシュに区切って，k 点を生成する．非金属系では電子的性質はブリルアンゾーン内でゆっくり変化するため，さほど多くの k 点を必要としないが，金属系の場合は少ない k 点では問題がある．計算に用いる k 点のセットを如何に選ぶかということは，計算精度と計算時間のバランスを取る上で重要になってくる．区切り数は整数なので普通のプログラムは a,b,c 軸にそって逆空間を分割するように要求してくる．このときメッシュ間隔が均等になるようにしたほうがよい．時々単に k 点の総数を聞いてくる場合があるが，おそらくプログラム内部で自動的にメッシュを区切っていると思われる．立方晶で 30 と k 点を指定しても立方根の整数をとるから $3 \times 3 \times 3 = 27$ のメッシュで実際には区切られることになる．次のメッシュは $4 \times 4 \times 4 = 64$ だから，30，40，50，60 と k を変化させても収束を調べても，すべて 27 で計算している可能性があるのだから，見た目は全く同じ値を返すことに注意しないといけない．六方晶である hcp ではこれと異なる．また，k 点の値は，リラクゼーションさせる方向と関連性がある．これという一般的な値はないし，k 点の変化によって時間や精度が変化してくるので，できる限りその系での計算の都度検証していくことが望ましい．

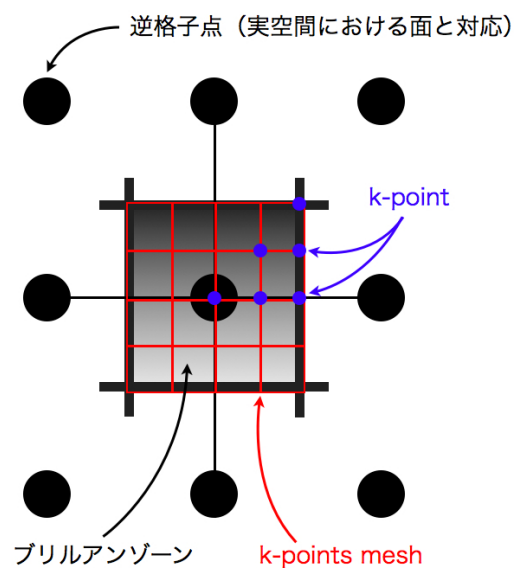


図 2.8: 正方格子における k-points mesh

第3章 ドーパントによる Si 中の積層欠陥への影響

3.1 緒言

小々節??で述べたように，最近，東北大金研の Y. Ohno らによって，ドーパントの影響で積層欠陥エネルギーが変化すると報告された．本小節では，Y. Ohno らの報告について簡単に紹介する．

Y. Ohno らは，Czochralski 法 (CZ 法) で生成したシリコン (Si) バルクに，リン (P) やヒ素 (As) 等の n-type ドーパントをドーピングさせた n 型 Si の試料，ボロン (B) やガリウム (Ga) 等の p-type ドーパントをドーピングさせた p 型 Si の試料と何もドーピングしていない純粋な Si の試料を用意し，約 1173[K] の温度でアニーリング処理を，10 時間ほど施した．そして試料中に見られた積層欠陥 (SF : Stacking Fault) の拡張幅を weak-beam 法を用いて測定した．アニーリング処理を行う前の積層欠陥エネルギーは，試料中の積層欠陥の拡張幅から約 50-70[mJ/m²] と見積もられた．P をドーピングした Si においては，アニーリング処理の時間の増加に応じて積層欠陥の拡張幅も増加する傾向を見せ，積層欠陥エネルギーも 40-60[mJ/m²] に減少した．一方で，p 型 Si とドーピングしていない Si の場合は，アニーリング処理中も積層欠陥エネルギーは変化しなかった．これらの結果は，暗に n-type ドーパントが，熱移動によって積層欠陥付近に集まっていることを示唆している．図 1.1 に，実験的に生成された Si の転位の様子が見える TEM 画像を示す．

また Y. Ohno らは，ドーパントとその濃度ごとに，Si 中の積層欠陥エネルギーを見積もった．その結果を図 1.2 に示す．図 1.2 より，Si 中に P, As 等の n-type ドーパントがドーピングされると，濃度の増加に応じて，Si 中の積層欠陥エネルギーは減少する傾向をみせている．一方で Si 中に B, Ga 等の p-type ドーパントが，大量にドーピングされても，Si 中の積層欠陥エネルギーは変化しない．この結果は，n-type ドーパントは Si 中の積層欠陥エネルギーを低下させ，p-type ドーパントは積層欠陥に影響しないことを示唆する．

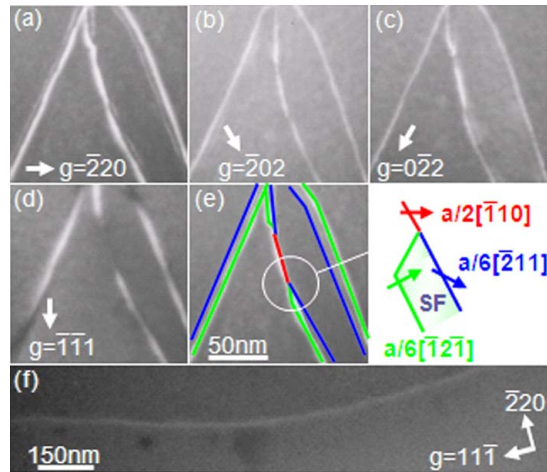


図 3.1: P がドーパされた Si に約 3 時間のアニーリング処理を行った TEM 画像 . 各画像は g 面を映した画像 . (a) $g=[\bar{2}20]$. (b) $g=[\bar{2}02]$. (c) $g=[0\bar{2}2]$. (d) $g=[\bar{1}\bar{1}1]$ (4 g 回折条件 , パラメータ差は $1.4 \times 10^{-2} A^{-1}$) , (e)(a) 中の転位の構造 . (f) ピュアな Si に約 10 時間のアニーリング処理を行った画像 $g=[11\bar{1}]$.

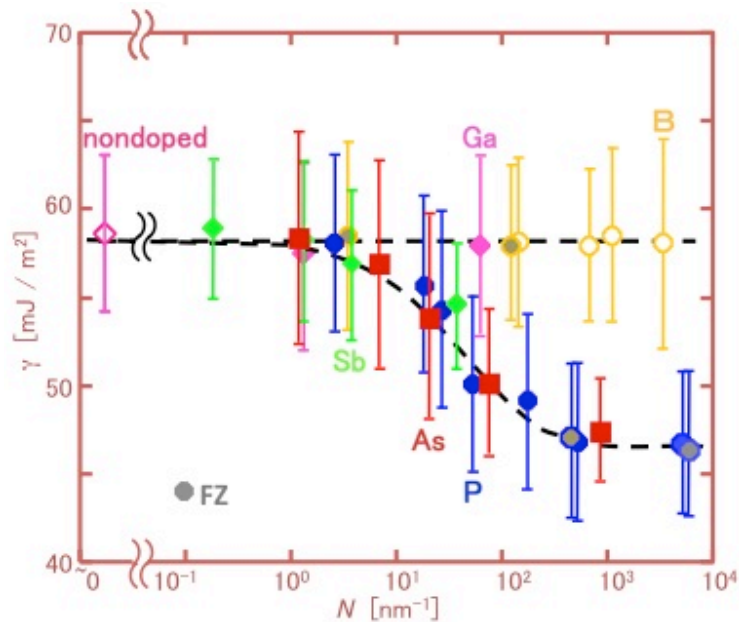


図 3.2: ドーパントごとの Si 中の積層欠陥エネルギー . 縦軸は積層欠陥エネルギー (γ) , 横軸はドーパント濃度を示し , 横軸については右側が高濃度を示す . また FZ は Floating Zone 法 (FZ 法) で生成された Si を使用した結果を示す .

3.2 2H 構造と 3C 構造の構造エネルギー差

3.2.1 計算手法

小節??, ??で述べたように, 積層欠陥は積層順序が乱れた欠陥であり, 全てが cubic 構造から成る 3C 中において, 積層欠陥が生じると, そこには 2H の特徴である hexagonal 構造が現れる. 半導体において, 積層欠陥エネルギーと 2H, 3C の構造エネルギー差は, 相関があることが報告されている [?].

本研究では, ドーパントを置換させた 2H と 3C の構造エネルギー差を計算し, 暫定的に積層欠陥エネルギーを見積もった. また本計算で扱ったドーパントは, I II 族元素, B, Ga, V 族元素 P, As である. また図 1.3 に計算モデルを示し, 計算に用いた格子定数を表 1.1 に示す. 本研究では図 1.3(a) に示すように, c 軸方向に向かって 12 層積まれるように $1 \times 1 \times 3$ のスーパーセルの 2H モデルを作成し, エネルギー (E_{2H}) を計算した. そして, 2H 構造の六方系の格子空間と合わせるため, 図 1.3(b) に示すような 3C モデルを作成した. 3C 構造は本来立方格子であるが, 格子定数から c 軸方向に向かって 6 層積まれる六方格子モデルの格子定数を計算し, $1 \times 1 \times 2$ のセルサイズに拡張した. そうすることで, 図 1.3(b) に示すようなモデルが出来る. そしてエネルギー (E_{3C}) を計算し, 2H と 3C の構造エネルギー差 ($\Delta E = E_{2H} - E_{3C}$) を計算した. そして図 1.3(c), (d) に示すように, モデル中の一層をドーパントに置換したモデル (Si(dopant)) での, 2H と 3C の構造エネルギー差を計算した.

全ての計算モデルの原子数は 12 原子であり, nondoped-Si においては, Si が 12 原子 ($n_{Si} = 12$) となり, ドーパントを含むモデルは, $n_{Si} : n_{dopant} = 11 : 1$ となる. 全ての計算において, Cutoff-Energy は 1000[eV], k-mesh は $12 \times 12 \times 3$ で行った. なお, ドーパントを含むモデルは内部・外部緩和を考慮している.

表 3.1: 計算モデルの格子定数と 2H 構造の内部パラメータ: $u = 0.3742$.

2H	a : b : c	3.85 : 3.85 : 6.363 []
	$\alpha : \beta : \gamma$	90 : 90 : 120 []
3C	a : b : c	5.4687 : 5.4687 : 5.4687 []
	$\alpha : \beta : \gamma$	90 : 90 : 90 []

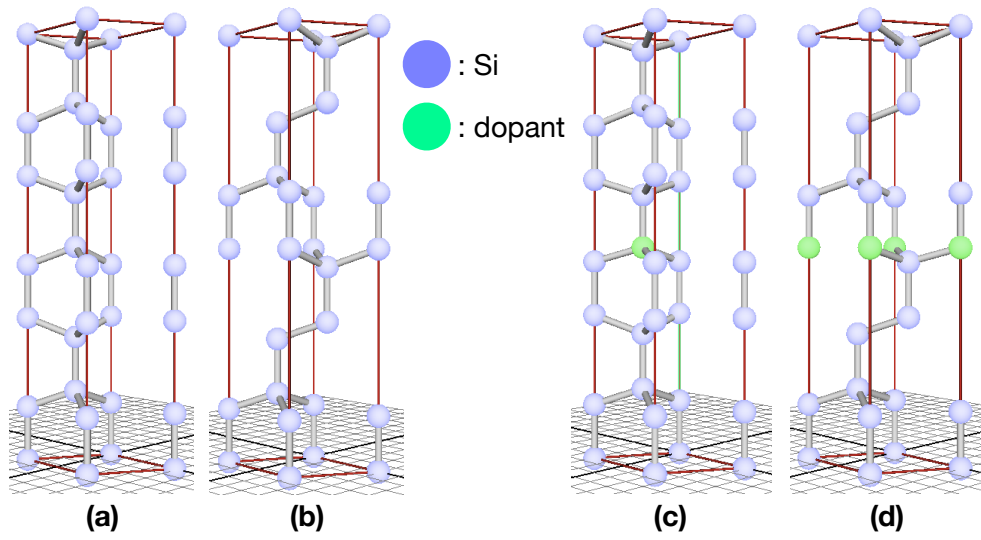


図 3.3: 計算モデル．モデル中の原子数は12原子．nondoped-Si モデルは $n_{\text{Si}} = 12$, doped-Si モデルは $n_{\text{Si}} : n_{\text{dopant}} = 11 : 1$, (a)2H-Si. (b)3C-Si. (c)2H-Si(dopant). (d)3C-Si(dopant).

3.2.2 計算結果

構造エネルギーの計算結果を表 1.2 に，構造エネルギー差を表 1.3 にまとめた．表 1.3 より，nondoped-Si の構造エネルギー差 0.1331 [eV] に対し，P がドーピングされたモデル (Si(P)) の構造エネルギー差が 0.0632 [eV] と低い値を示した．また P と同じく V 族元素である As の場合も 0.0832 [eV] と，0.1331 [eV] に対し低い値を示した．III 族元素である B, Ga の場合も，0.0871 [eV], 0.0679 [eV] と nondoped-Si の構造エネルギー差に比べ，低い値を示している．

表 3.2: 2H, 3C, dopant ごとの全ての計算モデルの構造エネルギー．計算モデルの原子数次の通り，nondoped-Si($n_{Si} = 12$), doped-Si($n_{Si} : n_{dopant} = 11 : 1$)

dopant	E_{2H} [eV]	E_{3C} [eV]
nondoped	-.64966742E+02	-.65099786E+02
P	-.64690383E+02	-.64753624E+02
B	-.64857009E+02	-.64944124E+02
As	-.63549794E+02	-.63632963E+02
Ga	-.62027516E+02	-.62095385E+02

表 3.3: 2H と 3C の構造エネルギー差． $\Delta E = E_{2H} - E_{3C}$

dopant	ΔE [eV]
undoped	0.1331
P	0.0632
B	0.0871
As	0.0832
Ga	0.0679

3.2.3 考察

小節 1.1 で、述べたように、n-type のドーパントがドーピングされた Si の積層欠陥エネルギーは低くなるが、nondoped-Si, p-type のドーパントがドーピングされた Si の積層欠陥エネルギーは変化しない。

本計算では、n-type のドーパントの構造エネルギー差が nondoped-Si の構造エネルギー差より低い値を示したので、n-type ドーパントは Si 中の積層欠陥エネルギーを低下させると思われたが、p-type のドーパントの構造エネルギー差も n-type 同様に低い値を示したので、Si 中の積層欠陥エネルギーは、ドーピングされる元素の種類に依存しないことが示唆された。この計算は、大野らの報告と整合しない。

この結果は、Si において積層欠陥エネルギーと 2H, 3C の構造エネルギー差の間に顕著な比例関係が無いことを示唆している。図 1.4 は、横軸を構造の hexagonal 構造と cubic 構造の積層周期比を、縦軸を Si 一原子あたりのエネルギーとして、積層周期比ごとの Si のエネルギーを示している。本来積層欠陥は、巨大な Si バルク中において、一部分の積層順序が乱れた欠陥であるので (多くの cubic 構造の層中に、一層だけ hexagonal 構造の層があるため)、積層欠陥エネルギーを見積もる場合、図 1.4 の赤で示す曲線において、hexagonal 構造の割合の少ない点の接線の傾きで見積もる必要がある。本計算は、青の直線の傾きを示す 2H と 3C のエネルギー差で見積もっているため、大雑把な計算である。

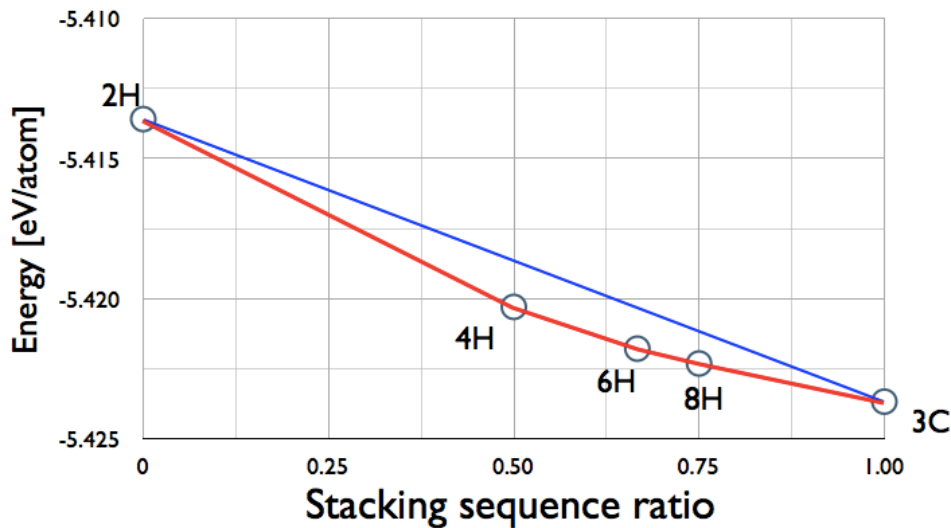


図 3.4: 積層周期比ごとの Si のエネルギー。縦軸：エネルギー [eV]，横軸：積層周期比。2H は hexagonal 構造 100%なので、左端。3C は cubic 構造 100%なので右端に位置する。4H は hexagonal 構造と cubic 構造の比が 1 : 1 なので中心に位置。6H は 1 : 3, 8H は 1 : 4 なので図の通りの位置。

3.3 積層欠陥 Si におけるドーパントの安定な位置

3.3.1 計算手法

本計算において、積層欠陥を含む Si バルクで、ドーパントが安定する位置を探索する。まず積層欠陥を含む Si のモデルを作成するにあたって、glide-set 転位と shuffle-set 転位について考察した。小節??で述べたように、Diamond 構造には、(111) 面において、面間隔の狭い glide-set 転位と面間隔の広い shuffle-set 転位がある。本計算で、着目しているのは積層欠陥であるため、glide-set 転位、shuffle-set 転位それぞれにおいて部分的に積層をズラすと、図??、図??に示すようになる。shuffle-set 転位は、部分的に積層をズラすと Diamond 構造の特徴である正四面体構造が崩れるため、部分転位でなく、完全転位として導入されることが知られている。一方、glide-set 転位は正四面体構造を崩さないため、直感的にも安定な積層欠陥が導入され易いことが示唆され、一般的にもそのように認知されている。そこで本計算では、glide-set 転位の積層欠陥モデル (SF-Si) を作成した。(1 $\bar{1}$ 0) 面からみた計算モデル (SF-Si) の模式図を図 1.5 に示す。表 1.4 に計算モデル (SF-Si) の格子定数を示す。Aa 等のペアを A と見なすと ABCABABC と積層している。赤の破線が積層欠陥面を示す。図右の数字は、[111] 方向に向かって、層に番号 (site-index) を付けた。図の下から第 1 層、第 2 層 .. 第 16 層 (1-site ~ 16-site) とする。このモデルは、16 層で一周期となっているので、site index は 16 までしかない。

表 3.4: 計算モデル (SF-Si) の格子定数と内部パラメータ:u。本計算は、nondoped-Si も doped-Si も全て緩和前の格子定数を用いて計算モデルを作成した後、内部・外部緩和を考慮した構造最適化を行っている。

	a : b : c	3.8671 : 3.8671 : 25.3096 []
緩和前	$\alpha : \beta : \gamma$	90 : 90 : 120 []
	u	0.0935741773872365
	a : b : c	3.862183898 : 3.862183898 : 25.303134745 []
緩和後	$\alpha : \beta : \gamma$	90 : 90 : 120 []
	u	0.09366784094

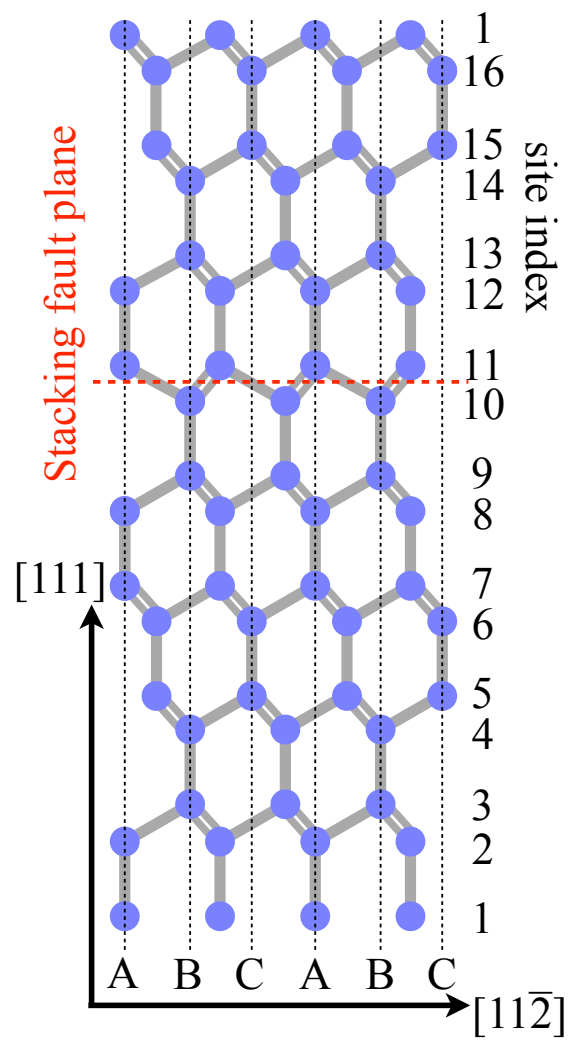


図 3.5: 積層欠陥を含む Si モデルの模式図．Aa 等のペアを A と見なすと AB-CABABC と積層している．赤の破線が積層欠陥面を示す．図右の数字は， $[111]$ 方向に向かって，層に番号を付けた（site index）．図の下から 1-site，2-site...16-site とする．このモデルは，16 層で一周期となっているので，16-site までしかない．

本計算は，主な条件として，Cutoff-Energy は 1000eV, k-mesh は $25 \times 25 \times 3$ で行った．さらに内部・外部緩和を考慮した．そして以下の手順で計算を行った．以下の計算で扱ったドーパントは P, B, As, Ga である．

1. SF-Si のエネルギー ($E_{\text{SF-Si}}$) を計算．
2. SF-Si モデルの内，1-site を全てドーパントに置換したモデル (SF-Si(dopant)) を作成，エネルギー ($E_{\text{SF-Si(dopant)}}$) を計算．
3. SF-Si(dopant) と SF-Si のエネルギー差を計算 ($\Delta E = E_{\text{SF-Si(dopant)}} - E_{\text{SF-Si}}$) ．
4. 2, 3 の手順を 1-site ~ 16-site，全てのパターンで行う．

次に，表 1.4 で示す緩和前の格子定数の SF-Si を $2 \times 2 \times 1$ に拡張したスーパーセルを作成した．図 1.6 に，拡張前と拡張後のモデルにおいて，それぞれドーパントを置換した一層を (0001) 面からみた様子を示す．図 1.6(a) で示すように，拡張前の格子空間内において，[0001] 方向に積まれている層の原子数が 1 原子であったが，拡張することで，図 1.6(b) に示すように，一層が 4 原子で構成される．その 4 原子の内，1 原子のみドーパントに置換，つまり層中のドーパント濃度を下げても同様の計算を行った．なお，この計算では，ドーパントを置換する層の対象は，1-site と 11-site のみである．また，この計算で扱ったドーパントは，P, B, As, Ga である．

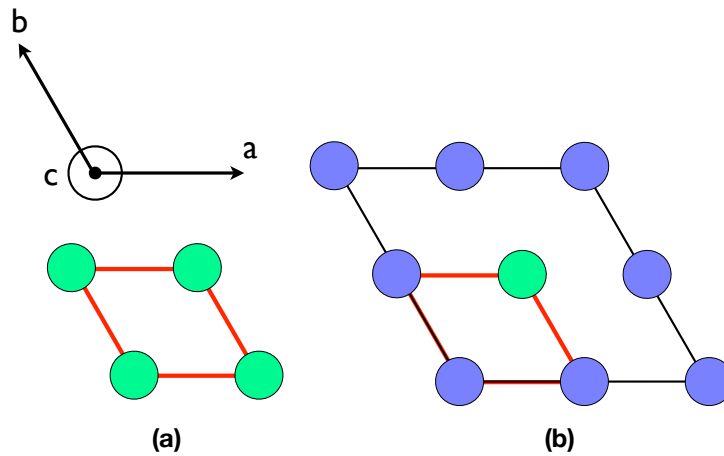


図 3.6: SF-Si モデルにおいて，[0001] 方向に積まれている層の内，ドーパントが含まれる一層模式図．赤菱形が拡張前の格子空間を示し，黒菱形が拡張後の格子空間を示す．緑の原子がドーパントを示し，青の原子が Si を示す．(a) 拡張前の SF-Si モデルにおいて，ドーパントと置換された一層を (0001) 面から見た様子．(b) $2 \times 2 \times 1$ スーパーセルの SF-Si モデルにおいて，ドーパントと置換された一層を (0001) 面から見た様子．

3.3.2 計算結果

計算した結果，SF-Si モデルのエネルギー ($E_{\text{SF-Si}}$) は， $-.86764259\text{E}+02$ [eV] であった．SF-Si(dopant) モデルのエネルギー ($E_{\text{SF-Si(dopant)}}$) 値は，表 1.5 にドーパントの元素，ドーパントが置換された層を示す site index ごとにまとめた．そして，表 1.6 に，SF-Si(dopant) と SF-Si のエネルギー差をまとめた．全てのモデルの原子数は 16 原子であり，SF-Si(dopant) モデルの原子数は， $n_{\text{Si}} : n_{\text{dopant}} = 15 : 1$ である．

表 3.5: ドーパントの元素，置換した層ごとの SF-Si(dopant) の構造エネルギー．例として，SF-Si モデルにおいて，16-site に P を置換したモデルのエネルギー ($E_{\text{SF-Si(P)}}$) は， $-.86410646\text{E}+02$ [eV] である．SF-Si モデルのエネルギー $E_{\text{SF-Si}} = -.86764259\text{E}+02$ [eV]．全てのモデルの原子数は 12 原子であり，SF-Si(dopant) モデルの原子数は， $n_{\text{Si}} : n_{\text{dopant}} = 15 : 1$ である．

site index	$E_{\text{SF-Si(P)}}[\text{eV}]$	$E_{\text{SF-Si(B)}}[\text{eV}]$	$E_{\text{SF-Si(As)}}[\text{eV}]$	$E_{\text{SF-Si(Ga)}}[\text{eV}]$
16	$-.86410646\text{E}+02$	$-.86583907\text{E}+02$	$-.85276134\text{E}+02$	$-.83762825\text{E}+02$
15	$-.86410689\text{E}+02$	$-.86582027\text{E}+02$	$-.85265240\text{E}+02$	$-.83760201\text{E}+02$
14	$-.86407680\text{E}+02$	$-.86585626\text{E}+02$	$-.85260843\text{E}+02$	$-.83764683\text{E}+02$
13	$-.86460564\text{E}+02$	$-.86572962\text{E}+02$	$-.85321459\text{E}+02$	$-.83780315\text{E}+02$
12	$-.86459036\text{E}+02$	$-.86544527\text{E}+02$	$-.85298750\text{E}+02$	$-.83791129\text{E}+02$
11	$-.86518958\text{E}+02$	$-.86540504\text{E}+02$	$-.85395307\text{E}+02$	$-.83804290\text{E}+02$
10	$-.86518963\text{E}+02$	$-.86540504\text{E}+02$	$-.85395308\text{E}+02$	$-.83804295\text{E}+02$
9	$-.86459036\text{E}+02$	$-.86544527\text{E}+02$	$-.85298750\text{E}+02$	$-.83791131\text{E}+02$
8	$-.86460564\text{E}+02$	$-.86572951\text{E}+02$	$-.85321458\text{E}+02$	$-.83780312\text{E}+02$
7	$-.86407680\text{E}+02$	$-.86585626\text{E}+02$	$-.85260843\text{E}+02$	$-.83764677\text{E}+02$
6	$-.86410689\text{E}+02$	$-.86582027\text{E}+02$	$-.85265240\text{E}+02$	$-.83760201\text{E}+02$
5	$-.86410646\text{E}+02$	$-.86583907\text{E}+02$	$-.85276140\text{E}+02$	$-.83760018\text{E}+02$
4	$-.86411483\text{E}+02$	$-.86583968\text{E}+02$	$-.85277885\text{E}+02$	$-.83760018\text{E}+02$
3	$-.86411130\text{E}+02$	$-.86584693\text{E}+02$	$-.85280440\text{E}+02$	$-.83760926\text{E}+02$
2	$-.86411130\text{E}+02$	$-.86584693\text{E}+02$	$-.85280439\text{E}+02$	$-.83760923\text{E}+02$
1	$-.86411483\text{E}+02$	$-.86583968\text{E}+02$	$-.85277851\text{E}+02$	$-.83760013\text{E}+02$

表 3.6: SF-Si(dopant) のエネルギーと SF-Si のエネルギー差 .

$$\Delta E_{\text{dopant}} = E_{\text{SF-Si(dopant)}} - E_{\text{SF-Si}}$$

site index	$\Delta E_{\text{P}}[\text{eV}]$	$\Delta E_{\text{B}}[\text{eV}]$	$\Delta E_{\text{As}}[\text{eV}]$	$\Delta E_{\text{Ga}}[\text{eV}]$
16	0.35361	0.18035	1.48812	3.00143
15	0.35357	0.18223	1.49902	3.00406
14	0.35658	0.17863	1.50342	2.99958
13	0.30370	0.19130	1.44280	2.98394
12	0.30522	0.21973	1.46551	2.97313
11	0.24530	0.22376	1.36895	2.95997
10	0.24530	0.22376	1.36895	2.95996
9	0.30522	0.21973	1.46551	2.97313
8	0.30370	0.19131	1.44280	2.98395
7	0.35658	0.17863	1.50342	2.9995
6	0.35357	0.18223	1.49902	3.0040
5	0.35361	0.18035	1.48812	3.00144
4	0.35278	0.18029	1.48637	3.00424
3	0.35313	0.17957	1.48382	3.00333
2	0.35313	0.17957	1.48382	3.00334
1	0.35278	0.18029	1.48641	3.00425

表 1.7 に, $2 \times 2 \times 1$ に拡張させて計算した, ドーパントの元素, 置換した層ごとの SF-Si(dopant) の構造エネルギーをまとめた. そして, 表 1.8 に, SF-Si のエネルギーと SF-Si(dopant) の構造エネルギー差をまとめた. 本計算のスーパーセルモデルは 64 原子で構成され, SF-Si(dopant) モデルの原子数の内訳は, $n_{\text{Si}} : n_{\text{dopant}} = 63 : 1$ である.

表 3.7: $2 \times 2 \times 1$ に拡張させて計算した, ドーパントの元素, 置換した層ごとの SF-Si(dopant) の構造エネルギー. $2 \times 2 \times 1$ の SF-Si モデルのエネルギー $E_{\text{SF-Si}} = -3.4705773\text{E}+03$ [eV]. 全てのモデルの原子数は 64 原子であり, SF-Si(dopant) モデルの原子数は, $n_{\text{Si}} : n_{\text{dopant}} = 63 : 1$ である.

site index	$E_{\text{SF-Si(P)}}[\text{eV}]$	$E_{\text{SF-Si(B)}}[\text{eV}]$	$E_{\text{SF-Si(As)}}[\text{eV}]$	$E_{\text{SF-Si(Ga)}}[\text{eV}]$
11	-.34679552E+03	-.34736793E+03	-.34588883E+03	-.34403801E+03
1	-.34673664E+03	-.34737311E+03	-.34582199E+03	-.34394702E+03

表 3.8: 層中のドーパント濃度 25[%] の時の SF-Si と SF-Si(dopant) のエネルギー差とドーパント濃度 100[%] の時の SF-Si と SF-Si(dopant) のエネルギー差

$\Delta E = E_{\text{SF-Si(dopant)}} - E_{\text{SF-Si}}$			
dopant	site index	dopant concentration 25[%]	dopant concentration 100[%]
P	11-site	$\Delta E = 0.2622$ [eV]	$\Delta E = 0.2453$ [eV]
	1-site	$\Delta E = 0.3211$ [eV]	$\Delta E = 0.3528$ [eV]
B	11-site	$\Delta E = -0.3102$ [eV]	$\Delta E = 0.2238$ [eV]
	1-site	$\Delta E = -0.3154$ [eV]	$\Delta E = 0.1803$ [eV]
As	11-site	$\Delta E = 1.1689$ [eV]	$\Delta E = 1.3690$ [eV]
	1-site	$\Delta E = 1.2357$ [eV]	$\Delta E = 1.4864$ [eV]
Ga	11-site	$\Delta E = 3.0197$ [eV]	$\Delta E = 2.9600$ [eV]
	1-site	$\Delta E = 3.1101$ [eV]	$\Delta E = 3.0043$ [eV]

3.3.3 考察

表 1.6 に示す計算結果にである SF-Si(dopant) と SF-Si のエネルギー差 (ΔE) について、縦軸を ΔE 、横軸を SF-Si(dopant) モデル中でドーパントが置換されている層 (site index) とし、図 1.7 にまとめた。(a) が B、(b) が Ga、(c) が P、(d) が As を置換させた場合の計算である。これらの図は、例えば、図 1.7(a) の site index=10 の場合、10-site を B を置換したモデル SF-Si(B) と SF-Si のエネルギー差 (ΔE) をとり、その値が縦軸のエネルギー点を示す。

図 1.5 より積層欠陥面が 10-site と 11-site の間にあるので、それをふまえて図 1.7 を用いて計算結果について考察する。(a) の ΔE 点を結んだエネルギー線は、10-site, 11-site を頂上とする山を形成している。山の高さは約 0.04 ~ 0.05 eV である。これは、積層欠陥を含む Si 中において、B は積層欠陥に近い 9-site ~ 12-site に B は集まらず、他の無欠陥部の層に寄り易いことを示している。B と同じく III 族元素である Ga をドーブさせた SF-Si(Ga) モデルの場合を示す (b) では、B とは逆に、10-site, 11-site で谷を形成しており、その深さは約 0.04 ~ 0.05 eV である。これは Ga は積層欠陥部に集まり易いことを示唆している。同じ III 族元素でも、B と Ga で異なる結果となった原因について、著者の考察を次より述べる。その原因は B, Ga の元素の大きさが異なるためと考える。B は Si に比べ小さい元素であるのに対し、Ga は Si より大きい元素であることが知られている。一般的に積層欠陥部は歪んでいるため、Si より小さい B が積層欠陥部にあれば歪みがより大きくなり、それに起因してエネルギーが大きくなったと示唆される。逆に、Si より大きな Ga の場合、欠陥部の歪みを緩和することで、エネルギーを低くしていると示唆される。続いて、V 族元素 P をドーブさせた場合の (c) では、10-site, 11-site 付近で谷を形成している。またその深さは約 0.1 ~ 0.12 eV であり、P は Ga に比べて、より積層欠陥部に集まり易いことを示唆している。P と同じく V 族元素である As の場合を示した (d) においても、P 同様に谷を形成しており、その深さは約 0.1 ~ 0.12 eV であり、P と同様に As は積層欠陥部に集まり易いことを示唆している。

ドーパントを含む層のドーパント濃度を下げするため、 $2 \times 2 \times 1$ に拡張したスーパーセルを用いて計算について、表??、表??を参照して考察する。表??より、層の P の濃度を 25% を下げても、欠陥のない 1-site でのエネルギー差が 0.3211 eV に対し、積層欠陥部である 11-site でのエネルギー差が 0.2622 eV と低い値を示している。これは P 濃度が低くても、P は積層欠陥部に集まり易いことを示唆している。続いて、濃度別に表??を見ると、11-site において、P 濃度 25% の場合は 0.2622 eV であるのに対し、P 濃度 100% の場合は 0.2453 eV と、P 濃度 100% の場合の方が約 0.012 eV 低い値を示している。一方、1-site においては、P 濃度 25% の場合の 0.3211 eV に対し、P 濃度 100% の場合は 0.3528 eV と、約 0.0317 eV 程高い値を示している。これは、積層欠陥部は P の濃度が高い方が安定し、無欠陥部 (1-site) は P の濃度が低い方が安定することを示唆している。つまり、積層欠陥を含む Si 中において、P は積層欠陥に集まり易いことを示唆している。

次に、図 1.8 より、ドーパントの濃度変化に応じた積層欠陥エネルギーについて考察する．図 1.8(a) に示すように、P の場合、層中の濃度を 25[%] に下げると、完全結晶部 (1-site) と積層欠陥部 (11-site) のエネルギー幅が小さくなったが、濃度を 100[%] の場合と同様の傾向をみせた．また完全結晶部 (1-site) では P の濃度は低い方が低いエネルギーを示し、積層欠陥部 (11-site) では P の濃度が高い方が低いエネルギーを示したことから、P は積層欠陥部に集まり易く、積層欠陥エネルギーを低下させることが示唆された．続いて、B の場合について考察する．B の場合、図 1.8(b) に示すように、濃度を下げると、完全結晶部と積層欠陥部とでほとんどエネルギー差がない．また黒点より赤点の方が低いため、B は、結晶中では集まりにくく、バラバラに存在することが示唆される．図 1.8(c) より、As の場合、P と同様に、濃度を下げても完全結晶部より積層欠陥部の方が低いエネルギーを示している．しかし、P の場合と違い、完全結晶部でも積層欠陥部でも As の濃度が低い方が安定化することを示唆しており、As は、Si 中において一カ所に集まりにくいことを示唆している．図 1.8(d) より、Ga の場合、濃度を下げると、積層欠陥部に集まり易いという傾向はそのまま完全結晶部と積層欠陥部のエネルギー幅が大きくなった．また Ga は濃度が高い方が低いエネルギーを示すことから、結晶中の一カ所に集まり易いことが示唆される．

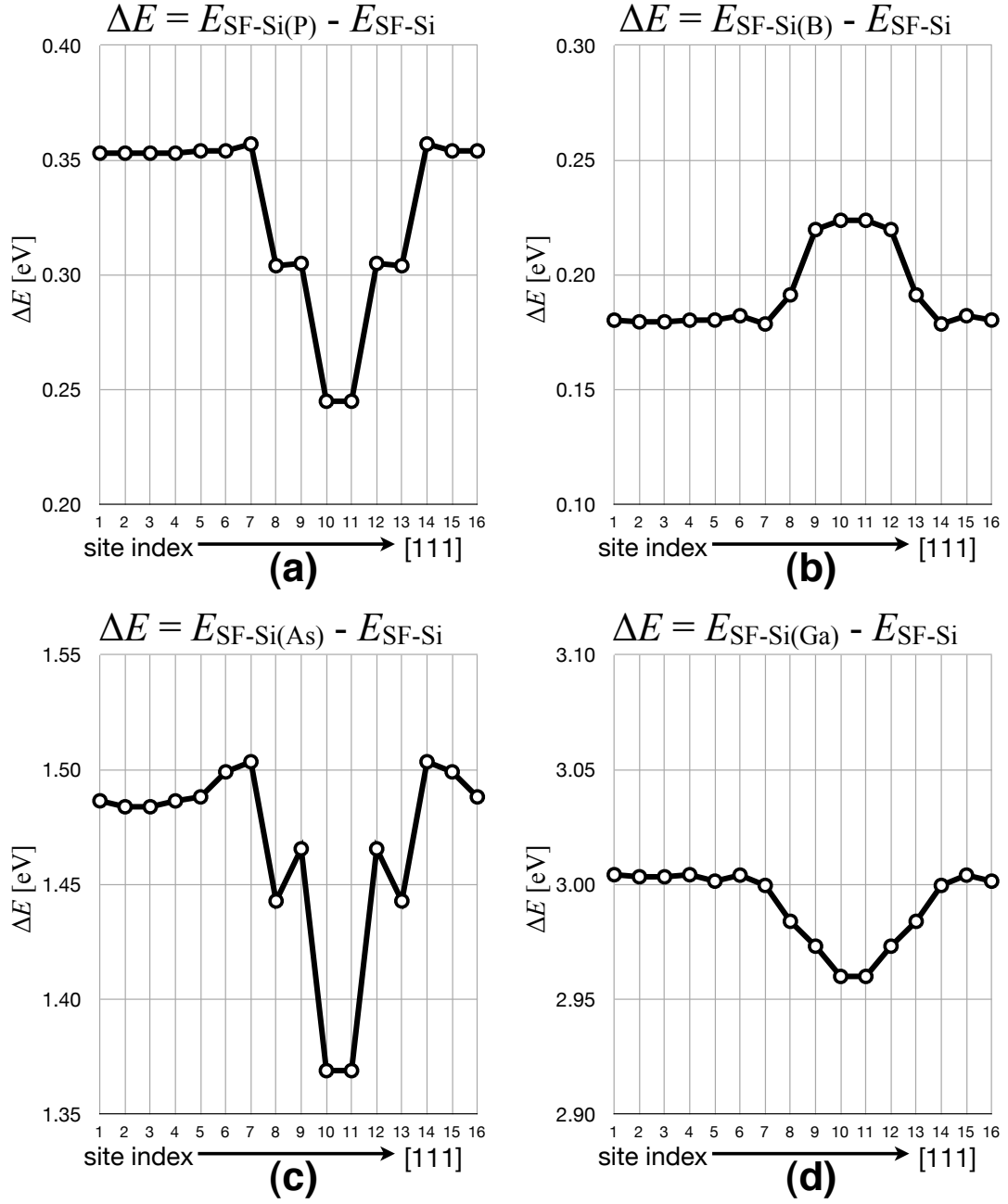


図 3.7: 積層欠陥を含む Si のエネルギーを基準としたドーパント位置に占との構造エネルギー差．縦軸： $\Delta E = E_{\text{SF-Si(dopant)}} - E_{\text{SF-Si}}$ [eV]，横軸：dopant がドーピングされている層，site index (図 1.5 の右側に示す層番号と対応)．(a)dopant = B, (b)dopant = Ga, (c)dopant = P, (d)dopant = As．

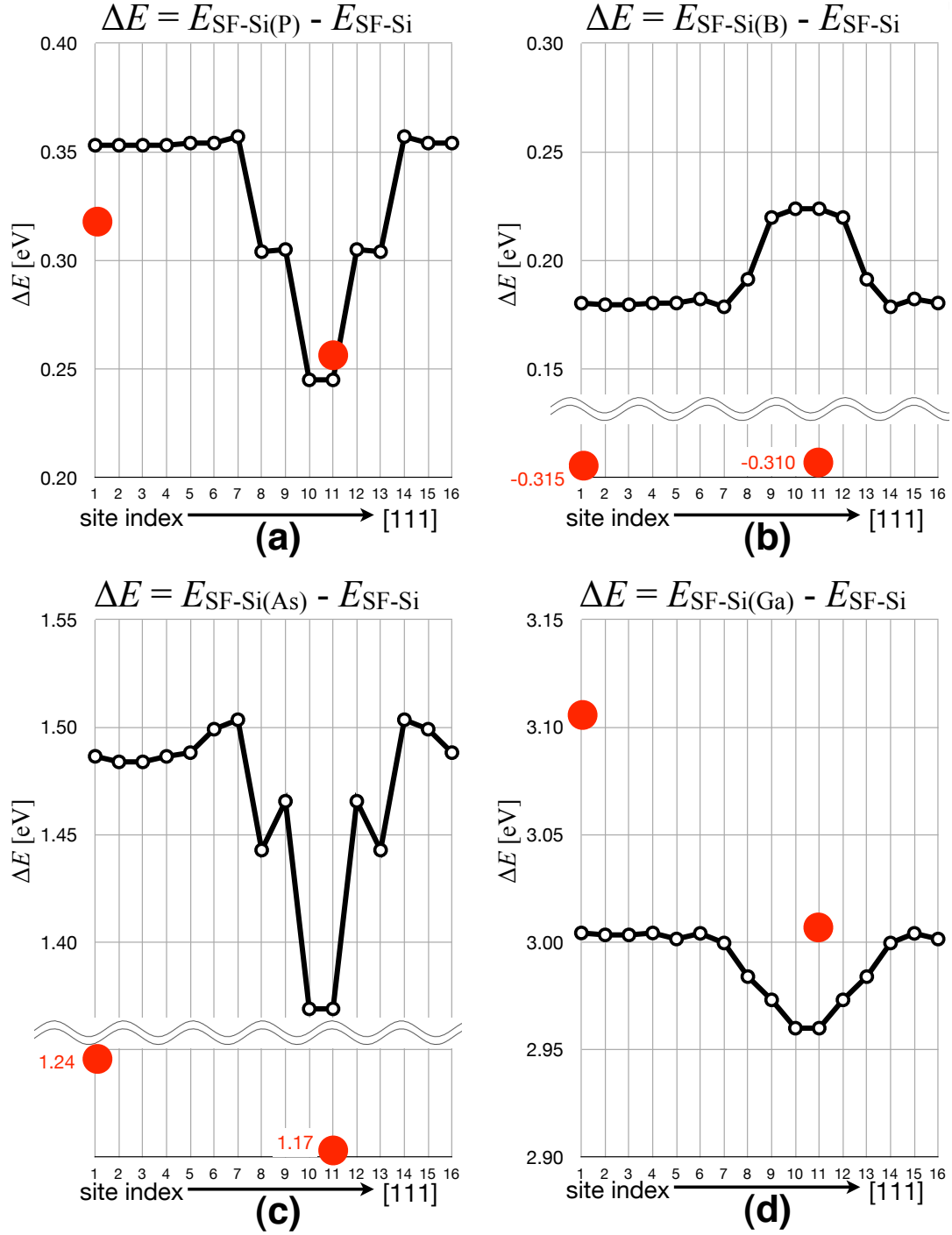


図 3.8: 層中のドーパント濃度を 25[%] に下げたモデルにおいて、積層欠陥を含む Si のエネルギーを基準としたドーパント位置ごとの構造エネルギー差．図 1.7 に示す計算結果上に、赤点で示すように新たに計算したドーパント濃度を 25[%] の場合のデータ点を加えた．縦軸： $\Delta E = E_{\text{SF-Si(dopant)}} - E_{\text{SF-Si}}$ [eV]，横軸：dopant がドーピングされている層，site index（図 1.5 の右側に示す層番号と対応）．(a)dopant = B, (b)dopant = Ga, (c)dopant = P, (d)dopant = As．

第4章 SiC マイクロパイプ欠陥の環境依存性

4.1 緒言

SiC 半導体の結晶成長中に生成するマイクロパイプ欠陥は，デバイス動作時のリーク電流の原因となることが知られている．Lely 法と呼ばれる気相成長法が主に用いられているが，この手法で成長させた SiC 単結晶には，リーク電流の原因となるマイクロパイプ欠陥が 0001 面上に多数確認されている．最近，関西学院大・金子らが，準安定溶媒エピタキシー（MSE：Metastable Solvent Epitaxy）と呼ばれる新奇な SiC 単結晶成長法を開発し，この手法で成長させた SiC 単結晶の {0001} 面には，マイクロパイプ欠陥は無く，平坦に成長している．両手法で成長させた SiC 単結晶の様子を図??に示す．

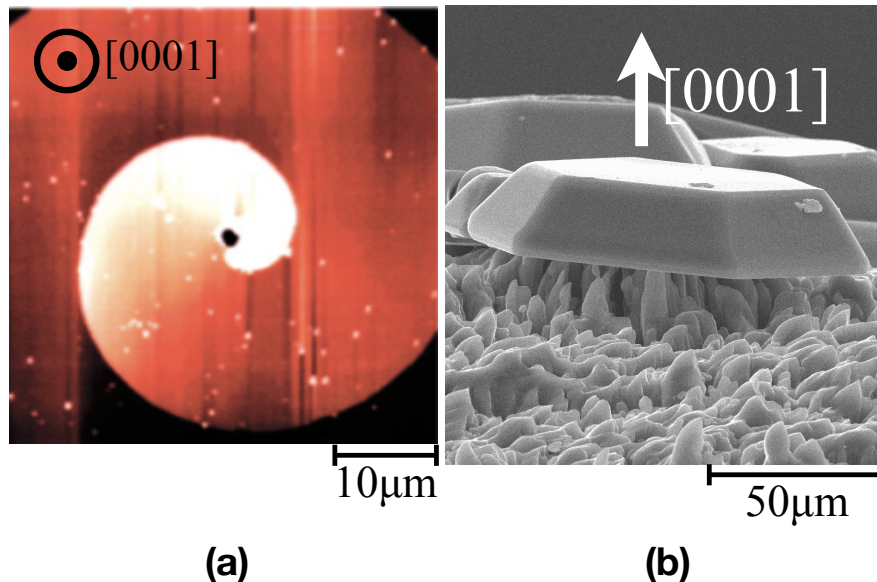


図 4.1: (a)Lely 法で成長させた SiC 単結晶を (0001) 面からみた写真．原子間顕微鏡（AFM）観察．中心の黒い孔がマイクロパイプ欠陥．(b)MSE 法で成長させた SiC 単結晶の写真．走査型電子顕微鏡（SEM）観察．{0001} 面が平坦である様子がわかる．

両手法で成長させた SiC 単結晶中のマイクロパイプ欠陥濃度が異なる原因として、Frank の理論とは別に、両手法の成長プロセスの違いにあるとする説がある。Lely 法では、原料を気化させ不活性ガスを介して、種結晶へ運び、気体から固体へと反応させる。物質の三体は、気体 液体 固体と変化させるのが自然であり、気固反応させる Lely 法のプロセスは非平衡がきついため、マイクロパイプ欠陥が生じやすいといわれている。一方、液体 Si を溶媒に用いて単結晶を成長させる MSE 法の成長プロセスは平衡に近いと、マイクロパイプ欠陥が閉塞していくと考えられている。また、マイクロパイプ欠陥濃度が異なる原因として、結晶表面を拡散する原子の拡散距離に依存している説もある。結晶と溶媒の界面が気固界面の場合、拡散原子の拡散距離が短いこと、液固界面の場合、拡散距離が長いことが知られている。そのため、成長プロセスが気固界面となる Lely 法では、マイクロパイプ欠陥を維持したまま結晶成長が進み、成長プロセスが液固界面となる MSE 法では、マイクロパイプ欠陥が閉塞していくと考えられている。

ところが、両手法には成長環境にも大きな違いがある。Lely 法では黒鉛坩堝を用いているので、気体溶媒中に大量の炭素 (C) が溶け出し、成長環境は C-rich と考えられる。一方、MSE 法の成長環境は種結晶が液体 Si に覆われているため、Si-rich である。環境によってその値が変わる物性として表面エネルギーが知られており、これは静的な要因として結晶成長を支配している。本研究では、精密な第一原理計算によって、SiC の環境に依存した表面エネルギーを計算した。SiC 単結晶成長において、成長プロセス、成長環境、マイクロパイプ欠陥の関連を一覧として、表 4.1 にまとめた。

表 4.1: SiC 単結晶成長において、成長プロセス、成長環境、マイクロパイプ欠陥の関連

	Lely 法	MSE 法
マイクロパイプ欠陥	有	無
成長プロセス	非平衡 (気固)	平衡 (液固)
環境	C-rich	Si-rich

4.2 表面エネルギー計算

本研究では、SiC 多形の内、3C、4H、6H-SiC のバルクモデルを作成した。続いて、六方構造の 4H、6H-SiC では、 $\{0001\}$ 面とそれに直交する $\{11\bar{2}0\}$ 面と $\{1\bar{1}00\}$ 面のスラブモデルを作成し、立方構造の 3C-SiC については、六方構造のそれらと

等価となる面である $\{111\}$ 面, $\{1\bar{1}0\}$ 面, $\{11\bar{2}\}$ 面のスラブモデルを作成し, VASP を用いて各々のモデルのエネルギー ($E_{\text{bulk}}, E_{\text{slab}}$) を計算した. 本計算の注意点として, 界面の環境について述べる. 本研究では真空-固体界面のスラブモデルを作成しており, 計算した表面エネルギーも真空-固体界面の表面エネルギーである. しかし, Lely 法では気体-固体界面であり, MSE 法では液体-固体界面であるため, 界面環境を精確に再現した計算ではない.

表面エネルギーは, バルクモデルから面を切り出すのに必要なエネルギーとして定義されるので, 式??のようにバルクモデルとスラブモデルのエネルギー差から見積もれる.

$$\Delta E = E_{\text{slab}} - E_{\text{bulk}} \quad (4.1)$$

続いて式??のように, エネルギー差 (ΔE) を表面積 ($S [\text{m}^2]$) で割ることで, 単位面積あたりの表面エネルギー (E_{surface}) を計算した.

$$E_{\text{surface}} = \frac{\Delta E}{S} \quad (4.2)$$

本計算で作成した計算モデルの格子定数を表??に示す. そして, 計算対象とした六方構造の3面と立方構造の3面を図??に示す. 本計算で使用した全てのスラブモデルの真空領域は図??に示すように, 約 10×2 である.

表 4.2: 計算に使用した SiC 多形の格子定数.

3C-SiC	a : b : c	4.37744 : 4.37744 : 4.37744 []
	$\alpha : \beta : \gamma$	90 : 90 : 90 []
4H-SiC	a : b : c	3.09541 : 3.09541 : 10.13044 []
	$\alpha : \beta : \gamma$	90 : 90 : 120 []
6H-SiC	a : b : c	3.09541 : 3.09541 : 15.18043 []
	$\alpha : \beta : \gamma$	90 : 90 : 120 []

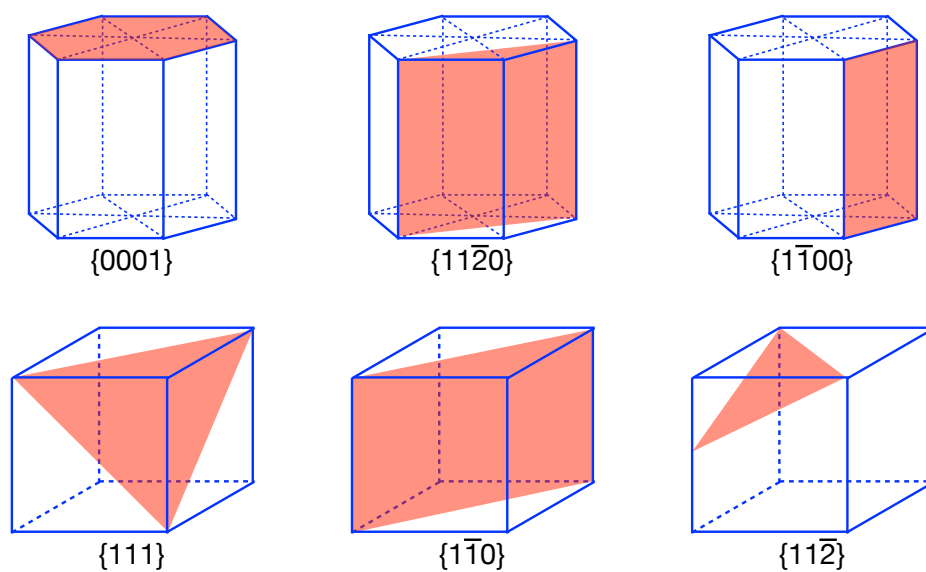


図 4.2: 六方構造の $\{0001\}$ 面とそれに直交する $\{11\bar{2}0\}$ 面と $\{1\bar{1}00\}$ 面．立方構造の $\{111\}$ 面， $\{1\bar{1}0\}$ 面， $\{11\bar{2}\}$ 面．

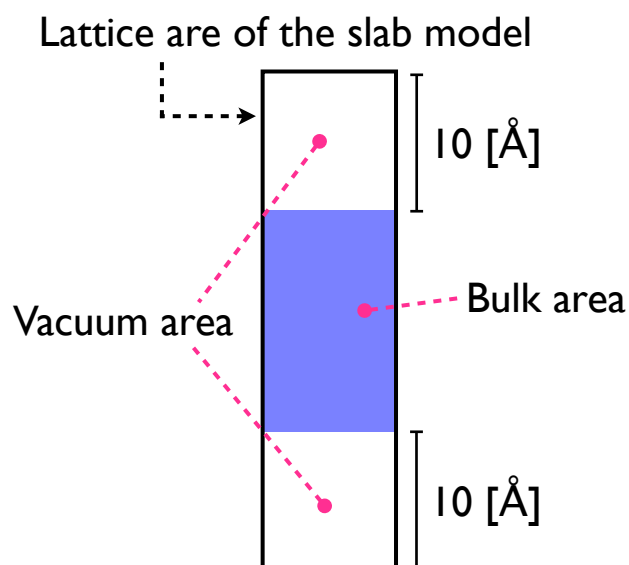


図 4.3: スラブモデルの真空領域

4.3 極性面の表面エネルギー計算

4.3.1 極性面

SiC の $\{0001\}$ 面は，面間に極性が生じるため，極性面と呼ばれる．その理由を図??を用いて説明する．図??より， $\{11\bar{2}0\}$ 面， $\{1\bar{1}00\}$ 面は，青で示す Si 原子と黒で示す C 原子両方が表面に現れている．一方， $\{0001\}$ 面の表面に現れている原子は Si だけである．図の下から $[0001]$ 方向に向かって面中の原子に着目すると，Si だけの面，C だけの面，Si 面，C 面が交互に配列している様子がわかる． $\{0001\}$ 面のように元素の異なる層で構成されている面では，電気的な勾配，極性が生じる．この原因は Si と C は電気陰性度が異なるためである．このように極性を持つ面を極性面という．

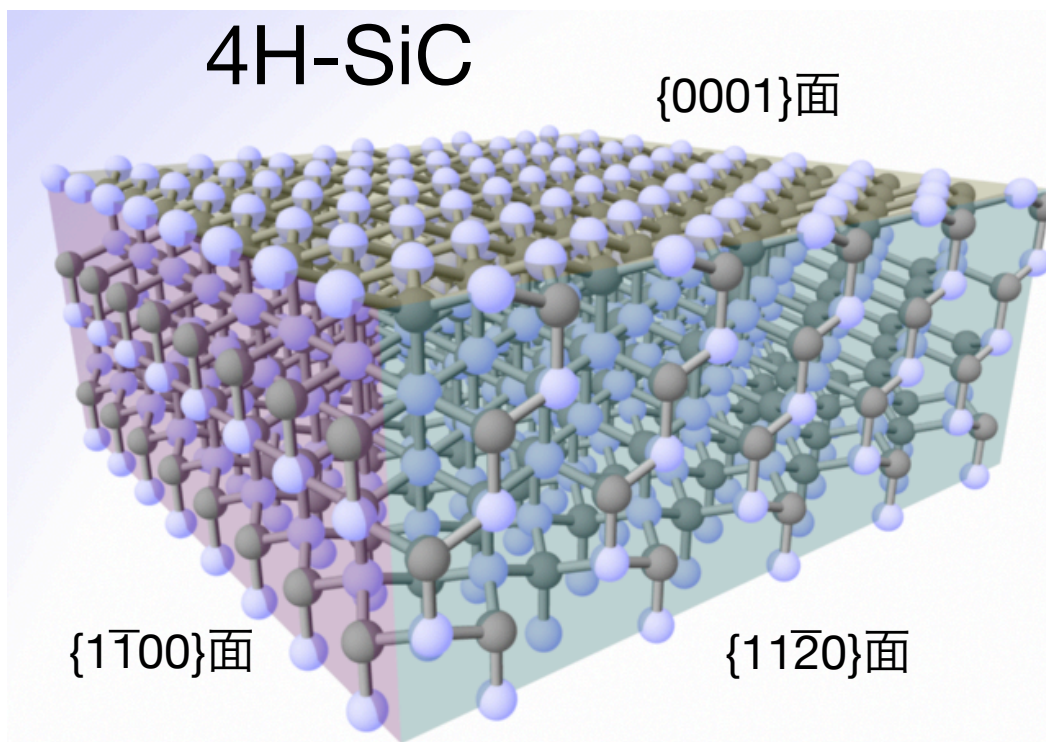


図 4.4: 4H-SiC の $\{0001\}$ 面， $\{11\bar{2}0\}$ 面， $\{1\bar{1}00\}$ 面の原子配列の様子．青は Si 原子，黒は C 原子を示す．4H-SiC は $[0001]$ 方向に向かって，Si だけの面，C だけの面，Si 面，C 面が交互に配列している様子がわかる．Si と C は電気陰性度が異なるため，面間に極性が生じる．このような面を極性面という．

4.3.2 被覆率を考慮した計算モデル

3C-SiC の極性面できったスラブモデルの模式図を図??に示す．図??のように，極性面できったスラブモデルは $[111]$ 方向と $[\bar{1}\bar{1}\bar{1}]$ 方向で，終端面の構成元素が異なる．本研究では，Si-rich な環境と C-rich な環境を区別するため，表面を覆う Si 原子の割合を θ_{Si} とし，図??のようなスラブモデルを作成し，それぞれの表面エネルギーを計算した．図??の $\theta_{\text{Si}} = 0$, $\theta_{\text{Si}} = 1$ のモデルでは，モデル中の Si 原子数と C 原子数の比が 1 : 1 でなくなる．そのため，小節??で述べたように式??から表面エネルギーを計算できない．そのため，化学ポテンシャルの概念を利用した．詳しくは次小々節で述べる．なお，本研究では，極性面のスラブモデルについてのみ，被覆率を考慮した．

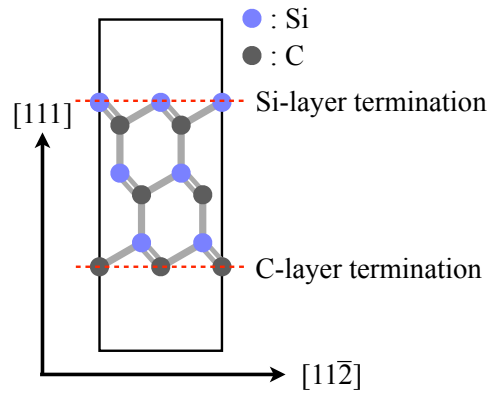


図 4.5: 3C-SiC の極性面できったスラブモデルを $\{1\bar{1}0\}$ 面からみた模式図． $[111]$ 方向は Si 面終端であるのに対し， $[\bar{1}\bar{1}\bar{1}]$ 方向では，C 面終端となる．

4.3.3 化学ポテンシャル

本研究では，Guo-Xin Qian らの論文[?]を参考に，SiC の化学ポテンシャル (chemical potential) を計算した．化学ポテンシャルは，モル [mol] あたり (1 原子，1 分子) の自由エネルギーを意味し，一般に μ で表記される．基底状態においてバルクのエネルギーを $E_{\text{i(bulk)}}$ ，バルク中の原子数を n_{i} とすると，化学ポテンシャル μ は式??で表すことができる．

$$\mu_{\text{i}} = \frac{E_{\text{i(bulk)}}}{n_{\text{i}}} \quad (4.3)$$

そして，式??に示すように SiC 中の化学ポテンシャル $\mu_{\text{SiC(bulk)}}$ は，Si の化学ポテンシャル μ_{Si} と C の化学ポテンシャル μ_{C} の和で表すことができる．また二種類の

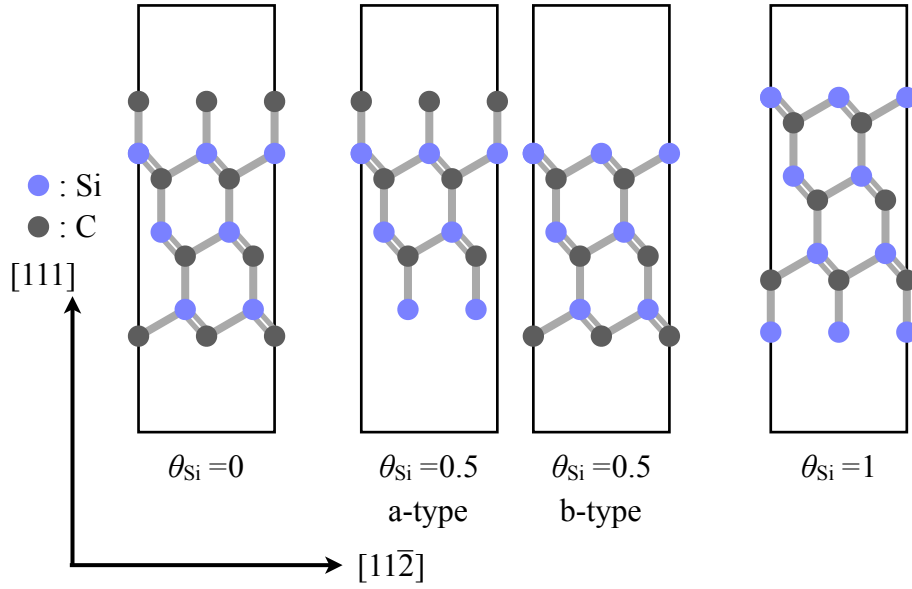


図 4.6: 極性面において，表面を覆う Si 原子の割合を θ_{Si} とし，Si-rich, C-rich の環境を考慮したスラブモデルの模式図．なお， $\theta_{\text{Si}} = 0.5$ の場合，図のように a-type と b-type のモデルを作成した．

元素 A, B からなる化合物 AB は，熱的に $AB = A + B + \Delta H_f$ が成り立ち，それは SiC においても同様であり式??が成り立つ．ここで ΔH_f は生成熱を示す．

$$\mu_{\text{Si}} + \mu_{\text{C}} = \mu_{\text{SiC(bulk)}} \quad (4.4)$$

$$= \mu_{\text{Si(bulk)}} + \mu_{\text{C(bulk)}} + \Delta H_f \quad (4.5)$$

そして，式??より，SiC 中の化学ポテンシャル (μ_{Si} と μ_{C}) が示し得るエネルギー値の範囲は以下のように計算できる．

$$\Delta H_f \leq 0 \text{ のとき} \quad (4.6)$$

$$\mu_{\text{Si(bulk)}} + \Delta H_f \leq \mu_{\text{Si}} \leq \mu_{\text{Si(bulk)}} \quad (4.7)$$

$$\mu_{\text{C(bulk)}} + \Delta H_f \leq \mu_{\text{C}} \leq \mu_{\text{C(bulk)}} \quad (4.8)$$

式??,??において， μ_{Si} と μ_{C} の値は式??を満足するように，相対的に決まる．例えば $\mu_{\text{Si}} = \mu_{\text{Si(bulk)}}$ とすると， $\mu_{\text{C}} = \mu_{\text{C(bulk)}} + \Delta H_f$ となる．化学ポテンシャルの概念を図??を用いて説明する．横軸において，左端が Si100%，右端が C100%を示し，SiC は組成比 1 : 1 なので中心に位置する．縦軸はエネルギーを示す．図中の曲線は，それぞれ Si バルク，C バルク，SiC バルクの模式的な自由エネルギー曲線を示す．SiC と Si が準安定平衡な状態では，Si と SiC のエネルギーがほぼ等しくなる．

これは $\mu_{\text{Si(bulk)}}$ と SiC 中の μ_{Si} は同じ値をとることを意味し、そのときの SiC 中の μ_{C} の値は、SiC と Si の自由エネルギー曲線の共通接線上にくる。つまり SiC と Si が準安定平衡な状態では、SiC 中の μ_{Si} と μ_{C} は、図??に示す SiC と Si の自由エネルギー曲線の共通接線の両端の値をとる。このとき、SiC の化学ポテンシャルの数値は、 $\mu_{\text{Si}} = \mu_{\text{Si(bulk)}}$ 、 $\mu_{\text{C}} = \mu_{\text{C(bulk)}} + \Delta H_f$ として計算される。C-rich な環境でも同様に、 $\mu_{\text{C}} = \mu_{\text{C(bulk)}}$ 、 $\mu_{\text{Si}} = \mu_{\text{Si(bulk)}} + \Delta H_f$ となることがわかる。

本研究では、この極端な化学ポテンシャルを用いて、極性面の表面エネルギーを計算した。

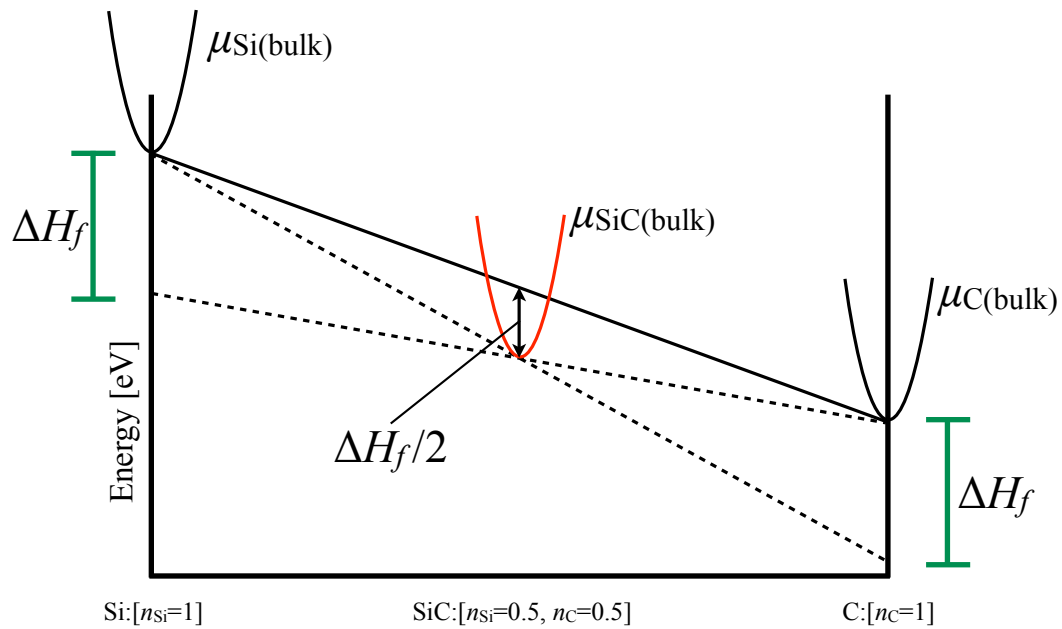


図 4.7: Si-C の系の組成・自由エネルギー図。横軸において、左端が Si100%，右端が C100%を示し、SiC は組成比 1 : 1 なので、中心に位置する。縦軸はエネルギーを示す。

4.4 計算結果

本計算において, SiC バルク, Si バルク, C バルクの計算結果を表??にまとめた. SiC の格子定数は表に示す通り. Si バルクの格子定数は $a = 5.4687[\text{Å}]$, C バルクの格子定数は $a = 3.57368[\text{Å}]$ である.

表 4.3: SiC および Si, C バルクのエネルギー. モデルの原子数 $[n_i]$. Cutoff-E, k-mesh.

polytype	Energy [eV]	n_i	Cutoff-E [eV]	k-mesh
3C-SiC	-.48187544E+03	$n_{\text{Si}} : n_{\text{C}} = 32 : 32$	600	2 2 2
4H-SiC	-.48178775E+03	$n_{\text{Si}} : n_{\text{C}} = 32 : 32$	600	3 3 1
6H-SiC	-.36131817E+03	$n_{\text{Si}} : n_{\text{C}} = 24 : 24$	600	3 3 1
3C-Si	-.10849937E+02	$n_{\text{Si}} : n_{\text{C}} = 2 : 0$	1000	25 25 25
3C-C	-.18195132E+02	$n_{\text{Si}} : n_{\text{C}} = 0 : 2$	1000	31 31 31

続いて, $\{0001\}$ 面に直交する $\{11\bar{2}0\}$ 面, $\{1\bar{1}00\}$ 面できったスラブモデルの計算結果を表??にまとめた.

表 4.4: $\{0001\}$ 面に直交する $\{11\bar{2}0\}$ 面, $\{1\bar{1}00\}$ 面できったスラブモデルのエネルギー. モデルの原子数 $[n_i]$. Cutoff-E, k-mesh.

polytype	Surface	Energy [eV]	n_i	Cutoff-E [eV]	k-mesh
3C-SiC	$\{1\bar{1}0\}$	-.23463915E+03	$n_{\text{Si}} : n_{\text{C}} = 16 : 16$	600	5 1 3
	$\{11\bar{2}\}$	-.34807293E+03	$n_{\text{Si}} : n_{\text{C}} = 24 : 24$	600	2 5 1
4H-SiC	$\{11\bar{2}0\}$	-.45832480E+03	$n_{\text{Si}} : n_{\text{C}} = 32 : 32$	600	3 1 2
	$\{1\bar{1}00\}$	-.46809785E+03	$n_{\text{Si}} : n_{\text{C}} = 32 : 32$	600	5 1 2
6H-SiC	$\{11\bar{2}0\}$	-.68670807E+03	$n_{\text{Si}} : n_{\text{C}} = 48 : 48$	600	3 1 1
	$\{1\bar{1}00\}$	-.69891968E+03	$n_{\text{Si}} : n_{\text{C}} = 48 : 48$	600	5 1 1

続いて， $\{0001\}$ 面できったスラブモデルの計算結果を表??にまとめた．polytype 中にある a-type，b-type は，図??に示すように， $\theta_{\text{Si}} = 1/2$ の時のモデルを示す．

表 4.5: $\{0001\}$ 面できったスラブモデルにおいて，表面の被覆率 (θ_{Si}) ごとのエネルギー．モデルの原子数 $[n_i]$ ．Cutoff-E, k-mesh．a-type，b-type は，図??の $\theta_{\text{Si}} = 1/2$ のモデルを参照．

polytype	coverage (θ_{Si})	Energy [eV]	n_i	Cutoff-E [eV]	k-mesh
3C-SiC	$\theta_{\text{Si}} = 0$	-.73257423E+03	$n_{\text{Si}} : n_{\text{C}} = 48 : 52$	600	3 3 1
a-type	$\theta_{\text{Si}} = 1/2$	-.69360584E+03	$n_{\text{Si}} : n_{\text{C}} = 48 : 48$	600	3 3 1
b-type	$\theta_{\text{Si}} = 1/2$	-.70535186E+03	$n_{\text{Si}} : n_{\text{C}} = 48 : 48$	600	3 3 1
	$\theta_{\text{Si}} = 1$	-.73257423E+03	$n_{\text{Si}} : n_{\text{C}} = 52 : 48$	600	2 5 1
4H-SiC	$\theta_{\text{Si}} = 0$	-.96747961E+03	$n_{\text{Si}} : n_{\text{C}} = 64 : 68$	600	3 3 1
a-type	$\theta_{\text{Si}} = 1/2$	-.93444837E+03	$n_{\text{Si}} : n_{\text{C}} = 64 : 64$	600	3 3 1
b-type	$\theta_{\text{Si}} = 1/2$	-.94625973E+03	$n_{\text{Si}} : n_{\text{C}} = 64 : 64$	600	3 3 1
	$\theta_{\text{Si}} = 1$	-.97345442E+03;	$n_{\text{Si}} : n_{\text{C}} = 68 : 64$	600	2 5 1
6H-SiC	$\theta_{\text{Si}} = 0$	-.72659499E+03	$n_{\text{Si}} : n_{\text{C}} = 48 : 52$	600	3 3 1
a-type	$\theta_{\text{Si}} = 1/2$	-.69362152E+03	$n_{\text{Si}} : n_{\text{C}} = 48 : 48$	600	3 3 1
b-type	$\theta_{\text{Si}} = 1/2$	-.70522105E+03	$n_{\text{Si}} : n_{\text{C}} = 48 : 48$	600	3 3 1
	$\theta_{\text{Si}} = 1$	-.73244362E+03	$n_{\text{Si}} : n_{\text{C}} = 52 : 48$	600	2 5 1

最後に，極性面の表面エネルギーと極性面に直交する表面の表面エネルギーを表??と表??にそれぞれまとめた．

表 4.6: 極性面の表面エネルギー $[\text{J}/\text{m}^2]$. $1[\text{eV}]=1.60218 \times 10^{-19} [\text{J}]$

environment	coverage (θ_{Si})	3C-SiC	4H-SiC	6H-SiC
Si-rich	$\theta_{\text{Si}} = 0$	8.14	8.17	8.16
a-type	$\theta_{\text{Si}} = 1/2$	7.05	7.03	7.00
b-type	$\theta_{\text{Si}} = 1/2$	4.92	4.94	4.94
	$\theta_{\text{Si}} = 1$	2.80	2.82	2.83
C-rich	$\theta_{\text{Si}} = 0$	7.60	7.63	7.62
a-type	$\theta_{\text{Si}} = 1/2$	7.05	7.03	7.00
b-type	$\theta_{\text{Si}} = 1/2$	4.92	4.94	4.94
	$\theta_{\text{Si}} = 1$	3.33	3.36	3.38

表 4.7: 直交面の表面エネルギー $[\text{J}/\text{m}^2]$. $1[\text{eV}]=1.60218 \times 10^{-19} [\text{J}]$

Surface	3C-SiC	4H-SiC	6H-SiC
$\{11\bar{2}0\}$	3.53	3.31	3.39
$\{1\bar{1}00\}$	4.45	4.40	4.27

4.5 考察

計算結果より Si-rich において，SiC の多形ごと表面ごとの表面エネルギーを図??にまとめた．全ての面において，3C, 4H, 6H といった多形に応じた大きな変化が見られない．また全ての多形において， $\{0001\}$ 面が最も安定していることがわかる．一方で，C-rich において，SiC の多形ごと表面ごとの表面エネルギーを図??にまとめた．Si-rich 同様，全ての面において，3C, 4H, 6H といった多形に応じた大きな変化が見られないが，C-rich では， $\{0001\}$ 面が最も不安定であることを示している．

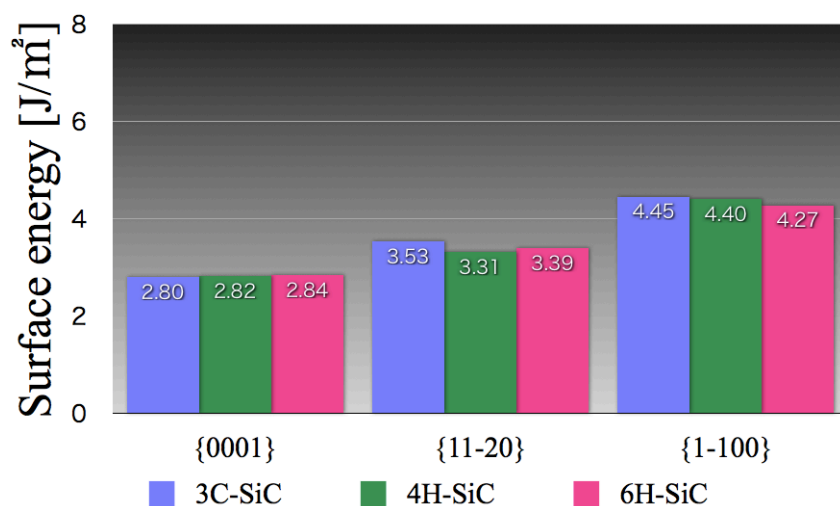


図 4.8: Si-rich 環境における SiC の表面エネルギー．

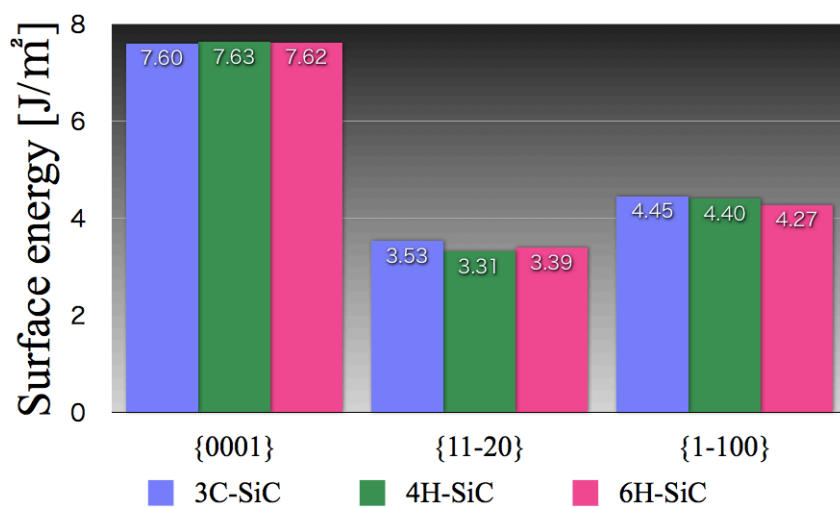


図 4.9: C-rich 環境における SiC の表面エネルギー．

SiC の $\{0001\}$ 面は，Si-rich では直交面より安定しており，C-rich では直交面より不安定であった．この結果より，図??に示すような，マイクロパイプ欠陥の生成機構を提案した．以下より，マイクロパイプ欠陥の生成モデルについて説明する．図 (a) は， $\{0001\}$ 面上にマイクロパイプ欠陥が生じた様子の断面図である．図 (b) のようにマイクロパイプ欠陥周辺に拡散原子がきたとき，環境に応じて結晶成長の仕方が異なる．表面エネルギーの低い面の方が面積を大きくするように結晶成長するため，Si-rich では，図 (c) のように， $\{0001\}$ 面の面積を大きくするため，マイクロパイプ欠陥を閉塞させる．一方 C-rich の場合， $\{0001\}$ 面が不安定なため，図 (d) のように，直交面の面積を大きくする．そのため，マイクロパイプ欠陥を維持したまま結晶成長する．

実験では，Si-rich 環境である MSE 法で成長させた SiC 単結晶のマイクロパイプ欠陥濃度は低く，C-rich 環境である Lely 法で成長させた SiC 単結晶のマイクロパイプ欠陥濃度が高い．本計算は実験結果と整合しており，成長環境と表面エネルギーが，マイクロパイプ欠陥の発生を支配していることを示唆している．

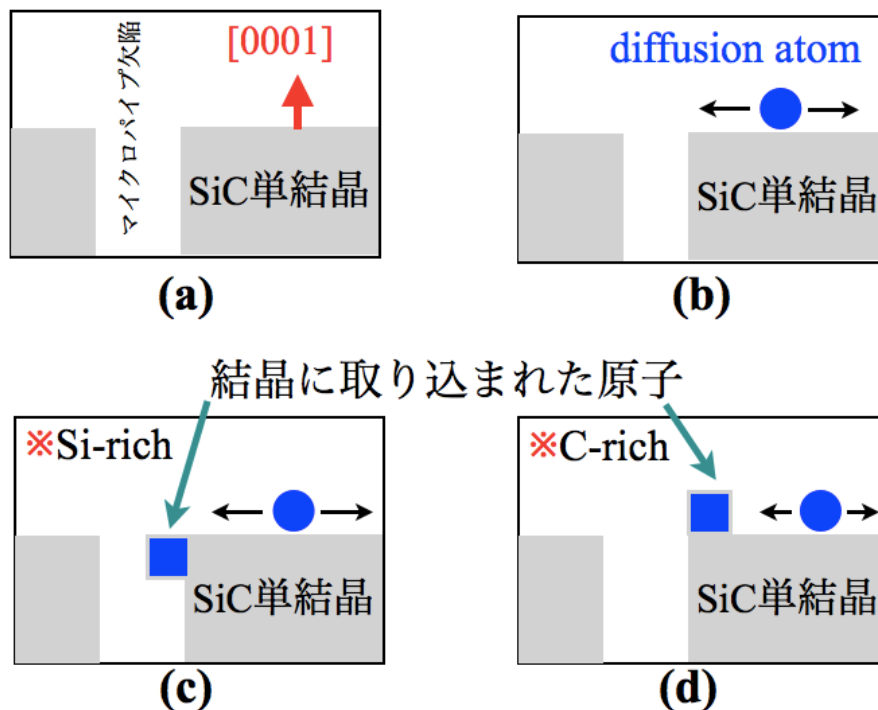


図 4.10: 表面エネルギーに依存したマイクロパイプ欠陥の生成モデル．

第5章 総括

本研究では、半導体の結晶表面や積層欠陥等の二次元欠陥を対象に、第一原理計算ソフト VASP を用いてエネルギー計算を行った。具体的には、シリコン (Si) 中に含まれる積層欠陥エネルギーを見積もり、さらにドーパされたリン (P) が Si 中の積層欠陥に与える影響を考察した。一方で、シリコンカーバイド (SiC) において、環境依存を考慮した精密な表面エネルギー計算からマイクロパイプ欠陥の生成起源を調べた。

ドーパントが Si 中の積層欠陥に与える影響について。

東北大金研の Y. Ohno らによって、Si 中の積層欠陥が、P が欠陥部に集まることで安定すると報告された。また Y. Ohno らの実験によって、P, As 等の n-type ドーパントは Si 中に濃化することで積層欠陥エネルギーを減少させることが、B, Ga 等の p-type ドーパントは Si 中の積層欠陥エネルギーに影響しないことがそれぞれ示唆された。

以下より本計算の内容を述べる。

1. 2H, 3C の構造エネルギー差を持って、積層欠陥エネルギーを見積った。すると n-type, p-type に関係なく、Si 中にドーパントが含まれると、構造エネルギー差は減少する傾向を見せた。この結果は、積層欠陥エネルギーは、単純な構造エネルギー差で見積もることができないことを示唆している。
2. 周期的境界条件を満たすように 16 層で構成された glide-set 転位における積層欠陥モデルの一層をドーパントに置換し、エネルギーを計算した。すると P, As は、積層欠陥部に集まり易く、積層欠陥エネルギーを減少させることが示唆された。B は積層欠陥部に集まらず、積層欠陥エネルギーに影響しないことが示唆された。Ga は積層欠陥部に集まり易いことが示唆されたが、P や As と比較すると積層欠陥への影響が小さいことが期待された。
3. ドーパントが含まれる積層欠陥モデルにおいて、ドーパントが含まれる層のドーパントの濃度を 100[%] から 25[%] に低下させて、P, B を置換した場合のエネルギー計算を行った。層中の P 濃度 25[%] の場合は、P 濃度 100[%] の場合と同様に P は完全結晶部に配置するより積層欠陥部に配置し易い結果が得られた。また積層欠陥部は、P 濃度が高く、完全結晶部は P 濃度が低くな

り易い傾向がみられた．一方 B を置換した場合は，B 濃度を下げるとわずかに完全結晶部に配置し易い結果が得られた．また B は，積層欠陥部に配置しようが，完全結晶部に配置しようが一カ所に集まらず，バラバラに結晶中に配置することが示唆された．総じて，P は積層欠陥部に集まり易く，積層欠陥エネルギーを低下させることが示唆され，B は積層欠陥部に集まらないため，積層欠陥エネルギーに影響を与えないことが示唆された．

以上が本計算より得られた知見であり，Y. Ohno らの実験結果と整合した．

SiC 表面エネルギー計算より提案するマイクロパイプ生成モデル．

SiC の単結晶成長法として，Lely 法と MSE 法がある．Lely 法で成長させた SiC 単結晶には，マイクロパイプ欠陥が $\{0001\}$ 面上に多数確認されている．一方 MSE 法で成長させた SiC 単結晶の $\{0001\}$ 面には，マイクロパイプ欠陥は無く，平坦に成長している．

以下より本計算の内容を述べる．3C, 4H, 6H の多形を対象に SiC の $\{0001\}$ 面とそれに直交する $\{11\bar{2}0\}$ 面， $\{1\bar{1}00\}$ 面の表面エネルギーをバルクモデルと真空-固体界面を有するスラブモデルのエネルギー差から計算した．極性面の表面エネルギー計算については，まず，スラブモデルの表面の Si の被覆率を θ_{Si} として $\theta_{\text{Si}} = 1$, $\theta_{\text{Si}} = 1/2$, $\theta_{\text{Si}} = 0$ となるスラブモデルを作成した．そして SiC が Si, C バルクと平衡となる極端な場合の化学ポテンシャルを用いて表面エネルギーを計算した．計算結果として，

1. $\{0001\}$ 面， $\{11\bar{2}0\}$ 面， $\{1\bar{1}00\}$ 面において，SiC の多形の違いによる表面エネルギーの大きな変化は見られなかった．この結果より，SiC では，多形は表面エネルギーに影響しないことが示唆された．
2. Si-rich では， $\{0001\}$ 面の表面エネルギーが最も低く， $\{0001\}$ 面に直交する $\{11\bar{2}0\}$ 面， $\{1\bar{1}00\}$ 面より安定であることが示唆された．この結果は，Si-rich では，SiC は， $\{0001\}$ 面に直交する面より $\{0001\}$ 面の表面積を大きくするように結晶成長することが示唆された．
3. C-rich では， $\{0001\}$ 面の表面エネルギーが最も高く， $\{0001\}$ 面に直交する $\{11\bar{2}0\}$ 面， $\{1\bar{1}00\}$ 面より不安定であることが示唆された．この結果は，C-rich では，SiC は， $\{0001\}$ 面よりそれに直交する面の表面積を大きくするように結晶成長することが示唆された．

これらの計算結果より考えられるマイクロパイプ欠陥の生成モデルを次に述べる．

1. Si-rich では $\{0001\}$ 面の表面積が大きくなるように結晶成長するため， $\{0001\}$ 面上に生じたマイクロパイプ欠陥は拡散原子によって埋め立てられ，閉塞していくことを示唆している．

2. C-rich では， $\{0001\}$ 面上で拡散原子が結晶に取り込まれ，マイクロパイプ欠陥を維持したまま結晶成長するため，欠陥濃度が高くなることを示唆している．

本計算より提案されたマイクロパイプ生成モデルは，SiC 単結晶成長の実験と整合した．

参考文献

- [1] 福田 哲生 著,『はじめての半導体シリコン』, 工業調査会, (2006), p8.
- [2] Y. Ohno, T. Shirakawa, T. Taishi and I. Yonenaga, *Applied Physics Letters* **95**, (2009), p091915.
- [3] 由宇 義珍 著,『はじめてのパワーデバイス』, 森北出版, (2006).
- [4] Tairov Yu.M. and Tsvetkov V.F., *Journal of Crystal Growth* **43**, (1978), p209.
- [5] N. Ohtani, M.Katsuno, T. Fjimoto and H. Yashiro, *Silicon Garbide Recent Major Advances*, Springer-Verlag Berlin, (2004), p137-162.
- [6] S. R. Nishitani and T. Kaneko, *Journal of Crystal Growth* **310**, (2008), p1815-1818..
- [7] 黒田 登志雄 著,『結晶は生きている～その成長と形の変化のしくみ～』, サイエンス社, (1984), p8-23 .
- [8] 鈴木 秀次 著,『転位論入門』, アグネ, (1967).
- [9] Hirth and Lothe, " *Theory of Dislocations*", McGraw-Hill, (1968).
- [10] Neudeck P.G., Powell J.A., *IEEE Electron Device Letters* **15**, (1994), p63.
- [11] Frank F.C., *Acta Crystallographica* **4**, (1951), p497.
- [12] Giocondi J., Rohrer G.S., Skowronski M.,Balakrishna V., Augustine G., Hobgood H.M., Hopkins R.H., *Journal Crystal Growth* **181**, (1997), p351.
- [13] Liliental-Weber Z., Chen Y., Ruvimov S., Swider W., Washburn J., *Materials. Research Society Symposium Proceedings* **449**, (1997), p417.
- [14] Pirouz P., *Philosophical Magazine A* **78**, (1998), p727.
- [15] Heindl J., Strunk H.P., Heydemann V.D., Pensl G., *Physica Status Solidi (a)* **162**, (1997), p251.

- [16] 山本 良一 編 ,『MARUZEN Advanced Technology-材料工学編- 材料の物性予測』 , 丸善株式会社 , (1998) .
- [17] David S. Sholl and Janice A. Steckel, ” *DENSITY FUNCTIONAL THEORY -A PRACTICAL INTRODUCTION-*” , WILEY, (2009).
- [18] P. E. Blochl, *Physical Review B* **50**, (1994), p17953.
- [19] VASP the GUIDE, <http://cms.mpi.univie.ac.at/vasp/vasp/vasp.html>, (2011/01/10 現在).
- [20] S. Takeuchi, K. Suzuki, *Physica Status Solidi (a)* **171**, (1999), p99.
- [21] G.-X. Qian, R. M. Martin and D. J. Chadi, *Physical Review B* **38**, (1988), p7649.

謝辞

本研究を遂行するにあたり，終始多大なる有益なご指導及び丁寧なご助言を賜り，また学士過程からの研究活動においても多くのご配慮を賜りました関西学院大学理工学部情報科学科 西谷滋人教授に深く感謝するとともに心より御礼申し上げます．

元・関西学院大学大学院理工学研究科 山本洋佑氏には，本研究の進行に伴い，常に有益なご助言ならびにご協力を賜りました．心より御礼申し上げます．また谷口僚氏，正木佳宏氏をはじめとする西谷研究室の皆様，さらに西谷研究室をご卒業された先輩後輩の方々にも，多大なご協力を賜り，深く感謝の意を表します．心より御礼申し上げます．

東北大学金属材料研究所 米永一郎教授，大野裕准教授，徳本有紀助教には，実験について有益な知識ならびにご助言を賜り，また本研究においてもご助力を賜り，深く感謝の意を表します．心より御礼申し上げます．

最後になりましたが，長かった私の大学生活，および研究活動を暖かく見守り，支援して下さった家族，全ての皆様に心から感謝します．ありがとうございました．

付 録 A Maple による結晶格子データの作成

本付録では，Si を例として，図 A.1(a) に示す立方晶形の Diamond 構造を，図 A.1(b) に示す六方晶形の構造として作成する例を示す．

Maple を用いて POSCAR を作成する方法は本付録を参考にして欲しい．

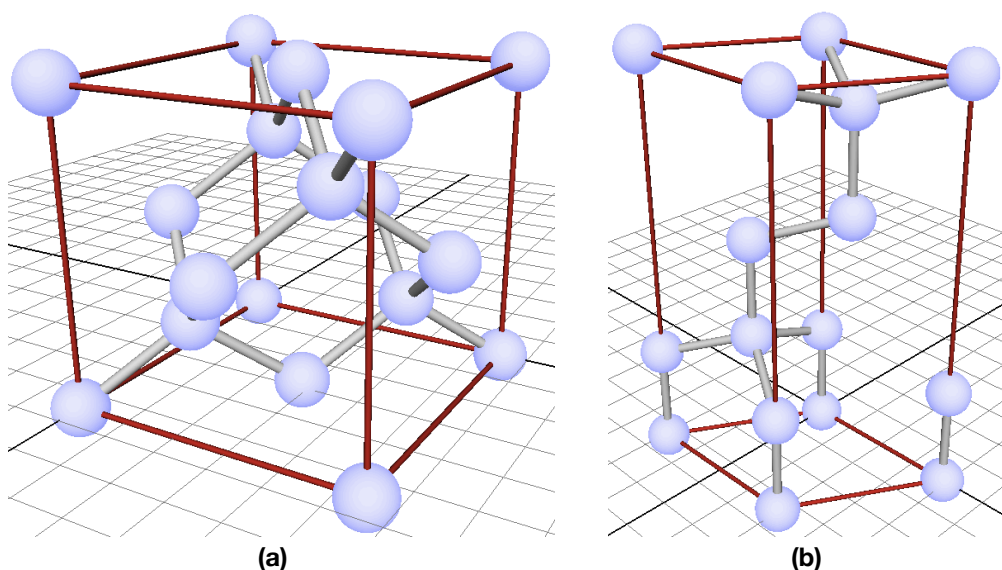


図 A.1: Diamond 構造 (3C) の Si. (a) 立方晶形のユニットセル . (b) 六方晶形のユニットセル . (a)(b) は両方とも Diamond 構造である .

まず POSCAR 形式にプリントする関数を用意する . また `Digits:=50:` は計算する有効桁を設定するパラメータである (Digits の値は小さくても問題ないので , 好みで) . POSCAR 関数の引数として , 原子位置の相対座標を格納した ball 配列と primitive vector を格納した p_vec 行列とする .

```

restart:
with(LinearAlgebra):
Digits:=50:

POSCAR:=proc(ball,p_vec) # for VASP calculation

local v,i;
printf("POSCAR\n");
printf("  1.000000000000000\n");
printf("    %1.16f    %1.16f    %1.16f\n",p_vec[1][1],p_vec[2][1],p_vec[3][1]):
printf("    %1.16f    %1.16f    %1.16f\n",p_vec[1][2],p_vec[2][2],p_vec[3][2]):
printf("    %1.16f    %1.16f    %1.16f\n",p_vec[1][3],p_vec[2][3],p_vec[3][3]):
printf("  %d\n",nops(ball)):
printf("Direct\n"):
for i from 1 to nops(ball) do
  v:=ball[i];
  printf("  %1.16f  %1.16f  %1.16f\n",v[1],v[2],v[3]);
end do:

end proc:

```

次に，立方晶形の Si の格子定数から，六方晶形の格子定数を計算する．立方晶形の Si の格子定数は $a=5.4687$ [Å] である． a_0 を立方晶形の格子定数の値とし，以下のように，六方晶形の格子定数 a : b : c と内部パラメータ u を計算できる． c 軸は立方晶の対角線の長さと同じ．さらに Diamond 構造の原子間距離は $a * \sqrt{(1/4)^2 + (1/4)^2 + (1/4)^2}$ より算出できる．内部パラメータ u は c 軸とボンド長から計算可能である．

Maple スクリプト

```
a0:=5.4687;
a := sqrt((0.5*a0)^2+(0.5*a0)^2+(0.0)^2);
c := sqrt(3.0*((a0)^2));
u := sqrt(3.0*((a0/4)^2))/c;
```

上記より計算した格子定数から primitive vector を計算する．六方晶では図??に示すように格子定数の内 $\alpha : \beta : \gamma = 90 : 90 : 120$ [度] となるため， a 軸を 120 度回転させると b 軸となる．

3 つの primitive vector, a_1 , a_2 , a_3 は以下の通りになり，これらを p_vec 行列にまとめる．

Maple スクリプト

```
a1:=Vector([a,0.0,0.0]):
a2:=Vector([-a*0.5, a*0.5*sqrt(3),0.0]):
a3:=Vector([0.0,0.0,c]):
p_vec:=Matrix([a1,a2,a3]);
```

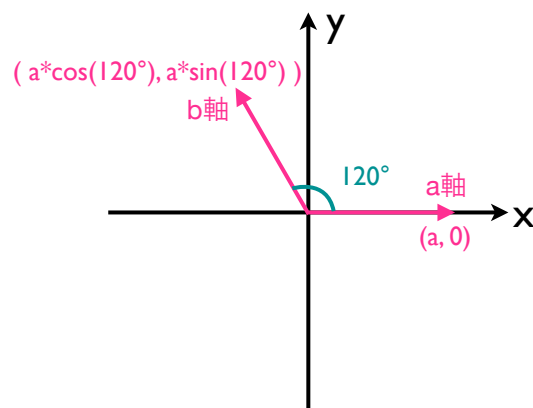


図 A.2: 六方晶形の a 軸と b 軸の関係．六方晶では格子定数の内 $\alpha : \beta : \gamma = 90 : 90 : 120$ [度] となるため， a 軸を 120 度回転させると b 軸となる．

次に原子の相対座標を設定する．六方晶では最密面は A, B, C 面と区別できる．これらの面の位置は，図??(a) に示すように $(0,0)$, $(1/3, 2/3)$, $(2/3, 1/3)$ の 3 点で区別できる．また Diamond 構造 (3C) は，3 層で 1 周期となるので，図??(b) に示すように c 軸方向に， $0.0, 1/3, 2/3$ の位置と図??(c) に示すように，内部パラメータを含んだ $0.0+u, 1/3+u, 2/3+u$ の位置に原子層がある．以上より，basis に相対座標を入力する．

Maple スクリプト

```
basis:=[Vector([0.0, 0.0, 0.0]),
        Vector([1/3, 2/3, 1/3]),
        Vector([2/3, 1/3, 2/3]),
        Vector([0.0, 0.0, 0.0+u]),
        Vector([1/3, 2/3, 1/3+u]),
        Vector([2/3, 1/3, 2/3+u])
      ]:
```

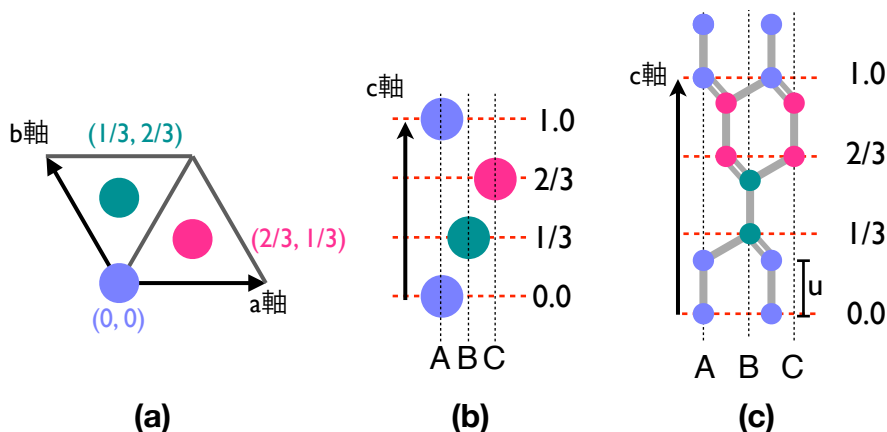


図 A.3: (a) 六方晶における最密面 A, B, C 面の座標． a , b 軸に対し，A 面を緑で示す $(0, 0)$ とすると，B, C 面の位置は青で示す逆正三角形の重心もしくは正三角形の重心となる． a 軸を $(1, 0)$ とすると青の点は a 軸， b 軸に対し， $(1/3, 2/3)$, $(2/3, 1/3)$ と計算できる．(b) 3C の場合，3 層で 1 周期となるので， c 軸方向に向かって， $0.0, 1/3, 2/3$ の位置に A, B, C 面がある．(c) Diamond 構造の場合，AaBbCc と，Aa 等の原子対が ABC と積層している． c 軸方向に対して $0.0, 1/3, 2/3$ の位置の他に，内部パラメータを含んだ $0.0+u, 1/3+u, 2/3+u$ の位置にも原子層がある．

以上で，ユニットセルのデータは整い，Maple で `POSCAR(basis,p-vec);` と入力すれば POSCAR データを出力できる．しかし後々のことも考慮し，セルのサイズも自由に変更できるようにしておく．

セルのサイズを指定し，モデルの実座標を計算する． S にセルのサイズを入力する． $S:=[2, 2, 1]$ とすると， $2 \times 2 \times 1$ に拡張したスーパーセルを作成する．`ball` には，拡張した全ての原子の実座標が格納される．実座標は primitive vector の行列に，右から相対座標をかけると計算できるので，Maple 上で，`p_vec.basis[i]` と記述している．詳しい計算式は小節??の POSCAR に記述している．

Maple スクリプト

```
S:=[1,1,1]: # supercell : SxSxS
ball:=[]:
for x from 0 to S[1]-1 by 1 do
  for y from 0 to S[2]-1 by 1 do
    for z from 0 to S[3]-1 by 1 do
      for i from 1 to nops(basis) do
        v:=p_vec.basis[i] + p_vec.Vector([x,y,z]);
        ball:=[op(ball),v];
      end do:
    end do:
  end do:
end do:
```

上記で計算した座標は実座標であるため，POSCAR に使えない．そこで `ball` に格納された実座標から相対座標を計算する．先程述べたように，実座標は primitive vector の行列に相対座標をかけると計算できたので，相対座標は，primitive vector の逆行列に右から実座標をかけると計算できる．Maple では `with(LinearAlgebra):` でライブラリを呼び出しておくと，逆行列は `MatrixInverse` を使うと簡単に計算できる．まず `p_vec:=Matrix([a1*S[1],a2*S[2],a3*S[3]])`; で格子空間を拡張し，逆行列を計算する．そして，Maple 上で `v:=ip_vec.ball[i]`; と記述し，相対座標を計算，`basis` に格納する．以上で，POSCAR 形式の原子位置のデータを出力する．

Maple スクリプト

```
p_vec:=Matrix([a1*S[1],a2*S[2],a3*S[3]]);
ip_vec:=MatrixInverse(p_vec);
basis:=[]:
for i from 1 to nops(ball) do
  v:=ip_vec.ball[i];
  basis:=[op(basis),v];
end do:

POSCAR(basis,p_vec);

POSCAR
1.0000000000000000
 3.8669548542748724 0.0000000000000000 0.0000000000000000
-1.9334774271374362 3.3488811390895915 0.0000000000000000
0.0000000000000000 0.0000000000000000 9.4720662513519193
6
Direct
0.0000000000000000 0.0000000000000000 0.0000000000000000
0.3333333333333333 0.6666666666666667 0.3333333333333333
0.6666666666666667 0.3333333333333333 0.6666666666666667
0.0000000000000000 0.0000000000000000 0.2500000000000000
0.3333333333333333 0.6666666666666667 0.5833333333333333
0.6666666666666667 0.3333333333333333 0.9166666666666667
```

付 録 B Maple によるスラブモデル の作成

本付録では、3C-SiC を $\{111\}$ 面で切ったスラブモデルを例に、その POSCAR を作成する Maple スクリプトを紹介する。

まず付録??と同様に、POSCAR 関数を用意する。そして六方晶形の格子定数を計算し、p_vec に行列として格納する。3C-SiC の格子定数は $a=4.37754$ [] である。

Maple スクリプト

```
restart:
with(LinearAlgebra):
Digits:=50:
POSCAR:=proc(ball,p_vec) # for VASP calculation
local v,i;
printf("POSCAR\n");
printf("  1.000000000000000\n");
printf("    %1.16f    %1.16f    %1.16f\n",p_vec[1][1],p_vec[2][1],p_vec[3][1]):
printf("    %1.16f    %1.16f    %1.16f\n",p_vec[1][2],p_vec[2][2],p_vec[3][2]):
printf("    %1.16f    %1.16f    %1.16f\n",p_vec[1][3],p_vec[2][3],p_vec[3][3]):
printf("    %d\n",nops(ball)):
printf("Direct\n"):
for i from 1 to nops(ball) do
  v:=ball[i];
  printf("    %1.16f    %1.16f    %1.16f\n",v[1],v[2],v[3]);
end do:
end proc:

a0:=4.37754;
a := sqrt((0.5*a0)^2+(0.5*a0)^2+(0.0)^2);
c := sqrt(3.0*((a0)^2));
u := sqrt(3.0*((a0/4)^2))/c;

a1:=Vector([a,0.0,0.0]):
a2:=Vector([-a*0.5, a*0.5*sqrt(3),0.0]):
a3:=Vector([0.0,0.0,c]):
p_vec:=Matrix([a1,a2,a3]);
```

続いて，相対座標を入力する．以下のように入力しているのは，図??のようなモデルを作成するためである．

Maple スクリプト

```
basis:= [Vector([1/3, 2/3, 1/3-u]),  
         Vector([2/3, 1/3, 2/3-u]),  
         Vector([0.0, 0.0, 3/3-u]),  
         Vector([0.0, 0.0, 0.0]),  
         Vector([1/3, 2/3, 1/3]),  
         Vector([2/3, 1/3, 2/3])  
       ]:
```

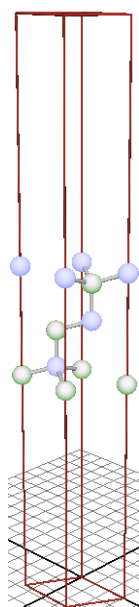


図 B.1: 3C-SiC のスラブモデル

続いて、セルの拡張を行う。付録??では、単元素のモデルであったが、今回は化合物である 3C-SiC を対象としている。後に POSCAR に相対座標を記述する際、上部に Silicon, 下部に Carbon の原子位置を記述するため、ここでデータの格納先を `Silicon:=[]: Carbon:=[]:` と分けておく。そして最後に ball にまとめて座標を格納する。また後に説明するが、原子座標において、c 軸成分の最大の値を `c_max` に格納しておく。今回は、{111} 面のスラブモデルを作成するので、`c_max` しか使わない。`b_max` の if 文のみ、`evalf` を使っているが、Maple では、`if sqrt(2)<2 then` と記述するとエラーが起こるためである。

Maple スクリプト

```
S:=[1,1,1]: # supercell : SxSxS
Silicon:=[]: Carbon:=[]:
a_max:=0: b_max:=0: c_max:=0:
N:=nops(basis):
for x from 0 to S[1]-1 by 1 do
  for y from 0 to S[2]-1 by 1 do
    for z from 0 to S[3]-1 by 1 do

      for i from 1 to N do
        v:=p_vec.basis[i] + p_vec.Vector([x,y,z]);
        if i<N*0.5+1 then
          Silicon:=[op(Silicon),v];
        else
          Carbon:=[op(Carbon),v];
        end if:

        if v[1]>a_max then a_max:=v[1]: end if:
        if evalf(v[2])>evalf(b_max) then b_max:=v[2]: end if:
        if v[3]>c_max then c_max:=v[3]: end if:
      end do:

    end do:
  end do:
end do:
ball:=[op(Silicon), op(Carbon)]:
```

続いて、スラブモデルの真空領域とバルク領域を設定する．ここでは図??に示すようなスラブモデルを作成する．片方の真空領域を `value` で設定すると真空領域全体は `vacuum` となる．今回は， $\{111\}$ 面で切ったスラブモデルを作成するため，`vacuum` の c 成分に値を入力している，また `bulk` の領域は先程計算しておいた `c_max` となる．バルク中の原子座標においての最も低い c の値は $(0.0, 0.0, 0.0)$ なので，`c_max` の値がそのまま `bulk` 領域となる．つまり，スラブモデルの c 軸は図??に示すように，真空領域とバルク領域の和で表される．

Maple スクリプト

```
value:=12:
vacuum:=Vector([0.0, 0.0, value*2]);
bulk:=Vector([0.0, 0.0, c_max]);
p_vec:=Matrix([a1*S[1], a2*S[2], bulk+vacuum]);
#p_vec:=Matrix([a1*S[1], a2*S[2], a3*S[3]]); #バルクモデルの POSCAR を作る場合はこっち．
```

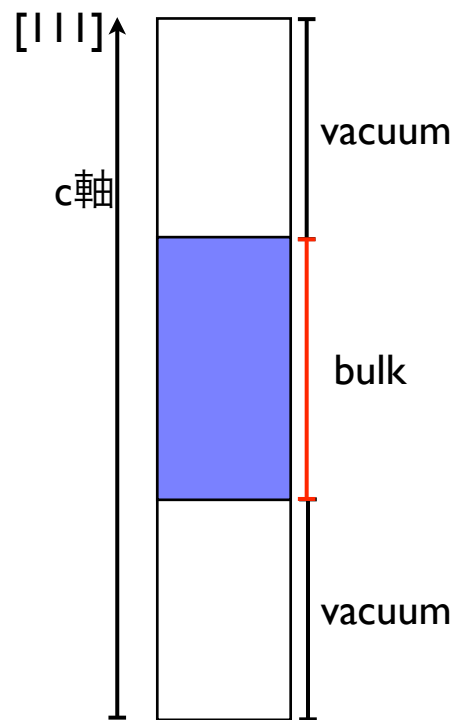


図 B.2: スラブモデルの真空領域とバルク領域

最後に，primitive vector の逆行列から原子位置の相対座標を計算し，POSCAR 関数を用いて出力する．計算のとき，原子座標は真空領域分だけ c 軸方向に移動している．そのため， $v:=ip_vec.(ball[i]+0.5*vacuum);$ のように，真空領域分だけ移動させた実座標から相対座標を計算する．また，出力した際，POSCAR の原子数の箇所が 6 とある．今回は， $1 \times 1 \times 1$ のサイズの 3C-SiC のスラブモデルを作成したので，原子数を 3 3 と書き換える必要がある．

Maple スクリプト

```
ip_vec:=MatrixInverse(p_vec):
basis:=[]:
for i from 1 to nops(ball) do
  v:=ip_vec.(ball[i]+0.5*vacuum);
  basis:=[op(basis),v];
end do:

POSCAR(basis,p_vec);

POSCAR
1.000000000000000
  3.0953882189153592    0.000000000000000    0.000000000000000
 -1.5476941094576796    2.6806848321557684    0.000000000000000
  0.000000000000000    0.000000000000000    29.6865912691237973
6
Direct
0.3333333333333333 0.6666666666666667 0.4255066996353037
0.6666666666666667 0.3333333333333333 0.5106419000520995
0.0000000000000000 0.0000000000000000 0.5957771004688953
0.0000000000000000 0.0000000000000000 0.4042228995311047
0.3333333333333333 0.6666666666666667 0.4893580999479005
0.6666666666666667 0.3333333333333333 0.5744933003646963
```

これまで紹介した Maple スクリプトの全容を以下に示す．

Maple スクリプト

```
restart:
with(LinearAlgebra):
Digits:=50:
POSCAR:=proc(ball,p_vec) # for VASP calculation
local v,i;
printf("POSCAR\n");
printf("  1.000000000000000\n");
printf("    %1.16f    %1.16f    %1.16f\n",p_vec[1][1],p_vec[2][1],p_vec[3][1]):
printf("    %1.16f    %1.16f    %1.16f\n",p_vec[1][2],p_vec[2][2],p_vec[3][2]):
printf("    %1.16f    %1.16f    %1.16f\n",p_vec[1][3],p_vec[2][3],p_vec[3][3]):
printf("    %d\n",nops(ball)):
printf("Direct\n"):
for i from 1 to nops(ball) do
  v:=ball[i];
  printf("    %1.16f    %1.16f    %1.16f\n",v[1],v[2],v[3]):
end do:
end proc:
```

```

a0:=4.37754;
a := sqrt((0.5*a0)^2+(0.5*a0)^2+(0.0)^2);
c := sqrt(3.0*((a0)^2));
u := sqrt(3.0*((a0/4)^2))/c;

a1:=Vector([a,0.0,0.0]):
a2:=Vector([-a*0.5, a*0.5*sqrt(3),0.0]):
a3:=Vector([0.0,0.0,c]):
p_vec:=Matrix([a1,a2,a3]);

basis:=Vector([1/3, 2/3, 1/3-u],
              Vector([2/3, 1/3, 2/3-u]),
              Vector([0.0, 0.0, 3/3-u]),
              Vector([0.0, 0.0, 0.0]),
              Vector([1/3, 2/3, 1/3]),
              Vector([2/3, 1/3, 2/3])
              ]:

S:= [1,1,1]: # supercell : SxSxS
Silicon:=[]: Carbon:=[]:
a_max:=0: b_max:=0: c_max:=0:
N:=nops(basis):
for x from 0 to S[1]-1 by 1 do
  for y from 0 to S[2]-1 by 1 do
    for z from 0 to S[3]-1 by 1 do

      for i from 1 to N do
        v:=p_vec.basis[i] + p_vec.Vector([x,y,z]);
        if i<N*0.5+1 then
          Silicon:=[op(Silicon),v];
        else
          Carbon:=[op(Carbon),v];
        end if:

        if v[1]>a_max then a_max:=v[1]: end if:
        if evalf(v[2])>evalf(b_max) then b_max:=v[2]: end if:
        if v[3]>c_max then c_max:=v[3]: end if:
      end do:

    end do:
  end do:
end do:
ball:=[op(Silicon), op(Carbon)]:

value:=12:
vacuum:=Vector([0.0, 0.0, value*2]);
bulk:=Vector([0.0, 0.0, c_max]);
p_vec:=Matrix([a1*S[1],a2*S[2],bulk+vacuum]);
#p_vec:=Matrix([a1*S[1],a2*S[2],a3*S[3]]); #バルクモデルの POSCAR を作る場合はこっち .

ip_vec:=MatrixInverse(p_vec):
basis:=[]:
for i from 1 to nops(ball) do
  v:=ip_vec.(ball[i]+0.5*vacuum);
  basis:=[op(basis),v];
end do:

```

付 録C Mayaによる視覚化

本付録では，Maya を使用した結晶格子の視覚化について簡単に紹介する．

C.1 視覚化の流れ

以下に視覚化を行うカレントディレクトリを示し，視覚化の流れを図??を示す．

カレントディレクトリ

```
[togase-kensuke-no-macbook-pro-2:~/Desktop/maya_test] toga% ls
anime.rb* lib/ plot.mel
data/ outcar.rb* poscar.rb*
[togase-kensuke-no-macbook-pro-2:~/Desktop/maya_test] toga%
```

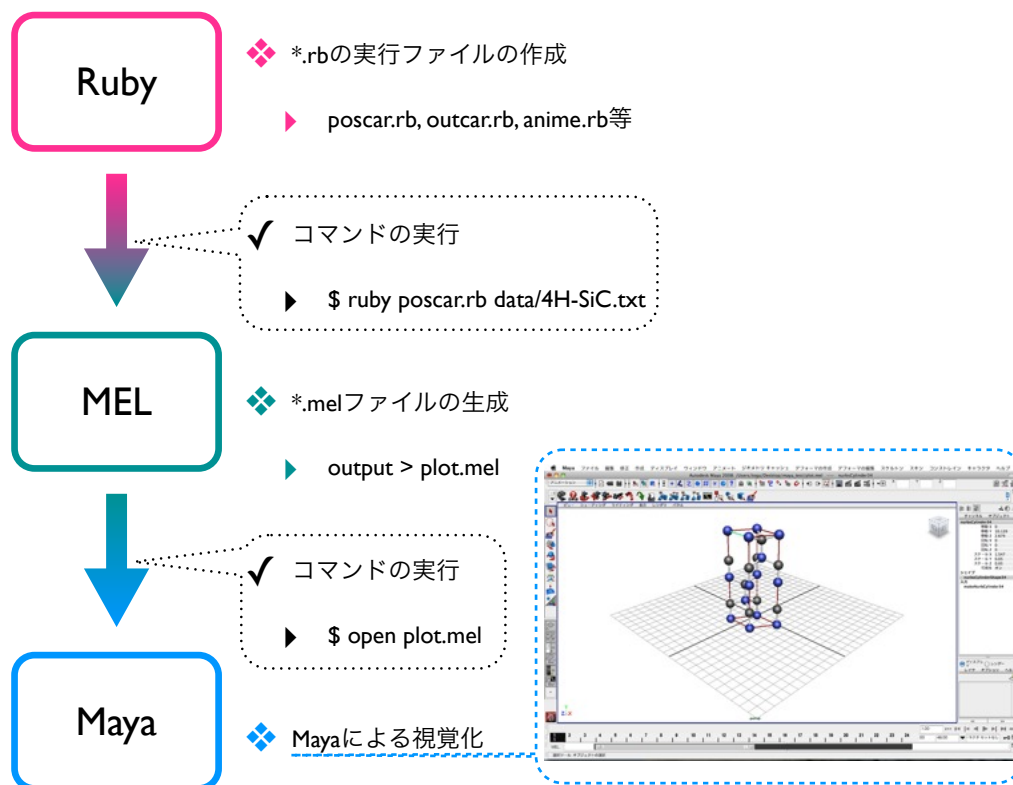


図 C.1: Maya による視覚化の流れ．

カレントディレクトリ中の `data/` は、結晶格子のデータファイルを格納したディレクトリであり、`lib/` は、視覚化を行うための様々な関数を用意したクラスを格納したライブラリである。`poscar.rb`、`outcar.rb`、`anime.rb` は視覚化を行う実行ファイルであり、`plot.mel` は、`*.rb` を実行後に生成される MEL スクリプトである。図??を示すように、`*.rb` を実行し、`*.mel` を開くと Maya 上で視覚化される。なお、`lib/` には以下のファイルがある。

`lib/`

```
[togase-kensuke-no-macbook-pro-2:~/Desktop/maya_test] toga% ls lib/
Calculation.rb* Maya.rb input.rb
Calculation.rb~ Maya.rb~ input.rb~
```

C.2 実行ファイルについて

実行ファイルの例として、`poscar.rb`、`outcar.rb`、`anime.rb` について紹介する。`poscar.rb` は VASP の POSCAR ファイルを視覚化させるスクリプトである。ここでは 4H-SiC の視覚化を行う。まず格子データ `4H-SiC.txt` を用意する。

`4H-SiC.txt`

```
[togase-kensuke-no-macbook-pro-2:~/Desktop/maya_test] toga% cat data/4H-SiC.txt
4H-SiC_fix_k=0.1_toga
1.0000000000000000
 3.0935700000000000 0.0000000000000000 0.0000000000000000
-1.5467850000000000 2.6791102083854259 0.0000000000000000
 0.0000000000000000 0.0000000000000000 10.1287000000000000
 4 4
Direct
0.0000000000000000 0.0000000000000000 0.0000000000000000
0.0000000000000000 0.0000000000000000 0.5000000000000000
0.3333333333333333 0.6666666666666667 0.2500000000000000
0.6666666666666667 0.3333333333333333 0.7500000000000000
0.0000000000000000 0.0000000000000000 0.1874200000000000
0.0000000000000000 0.0000000000000000 0.6874200000000000
0.3333333333333333 0.6666666666666667 0.4374200000000000
0.6666666666666667 0.3333333333333333 0.9374200000000000
```

続いて、`poscar.rb` の中身を順に説明する。まず Ruby の公式ライブラリとして、便利な `pp`、行列計算に必要な `matrix`、三角関数や平方根の計算に必要な `Math` を呼び出す。そして、著者が作成したライブラリ、`input.rb`、`Calculation.rb`、`Maya.rb` を呼び出す。

`poscar.rb`

```
require 'pp'
require 'matrix'
include Math

Dir.chdir("lib")do # lib/に path を通して、ライブラリを呼び出す。
  require 'input'
  require 'Calculation'
  require 'Maya'
end
```

出力する `*.mel` のファイル名の設定と、入力ファイルを引数として、結晶格子のデータを編集する `Calc` クラスを呼び出す。そして POSCAR ブロックを用いて、primitive vector 等の結晶格子データをまとめる。ここでは 4H-SiC を入力しているので、Si と C が存在する。そのため、`model.atom` には、`atom0` と `atom1` がある。入力ファイルの原子数の行が 4 4 となっており、前者は Si の原子数、後者は C の原子数を示す。そのため、ここでの `atom0` は Si を示し、`atom1` は C を示す。すると Si の原子座標は `model.cell[atom0]` に、C の原子座標は `model.cell[atom1]` にそれぞれ格納される。

poscar.rb

```
filename = 'plot.mel'          # *.mel ファイルの名前を設定 .
infile = ARGV[0]              # 入力ファイルをコマンドラインから入力
model = Calc.new(infile)      # Calculation.rb にある Calc クラスを呼び出し, 変数 model に代入する .
model.input(POSCAR)           # Calc クラスの input 関数を呼び出す . その時, POSCAR 形式のファイルを読み込むので, input.rb 中の POSCAR ブロックを引数とする .
# model.p_vec Primitive vector を格納
# model.atom 結晶の元素を atom0, atom1.. として格納 .
# model.cell 元素ごとの原子の実座標を格納 .
```

続いて, 結晶格子のセルのサイズを設定する. サイズの入力形式は以下の通りである. ここで注意して欲しいのが結晶格子と Maya の軸である. 図??(a) に示すように, 結晶格子の軸は c 軸が見た目の '高さ' に対応するが, Maya 上では図??(b) に示すように y 軸が見た目の '高さ' に対応する. このことから, プログラム上では結晶格子空間と Maya の空間を, a 軸 = x 軸, b 軸 = z 軸, c 軸 = y 軸として対応させている. また入力に応じた拡張の仕方を図??に示す. また `model.visualcoordinate_unitcell` は, 周期境界線上の原子を表示させる関数である. この一文があると, 結晶は図??(a) に示すように視覚化される.

poscar.rb

```
s = [[-0,0],[-0,0],[-0,0]]    # セルサイズの設定, それぞれのパラメータは [a 軸, c 軸, b 軸] 方向の拡張サイズを指す .
model.supercell(s)            # セルを拡張する .
model.visualcoordinate_unitcell # 周期境界線上の原子を表示
```

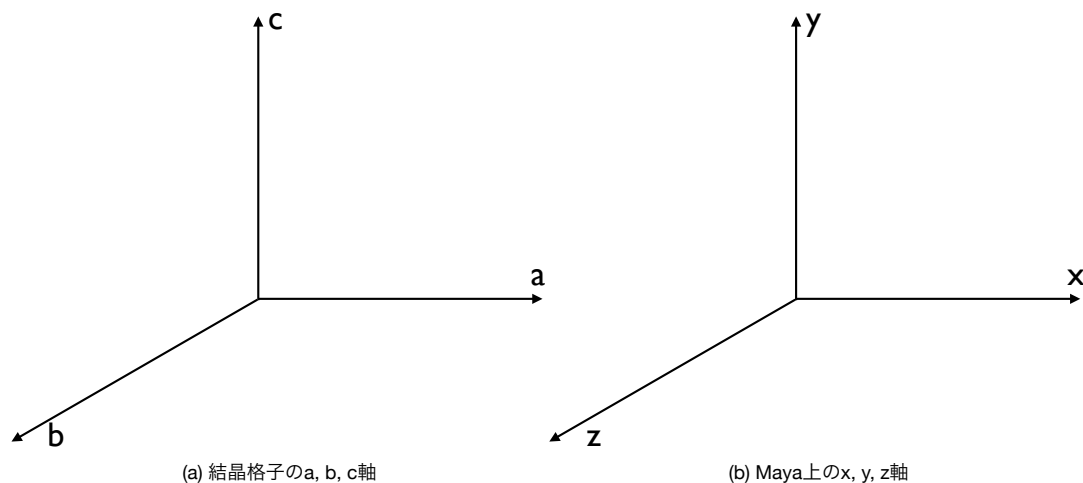


図 C.2: 結晶格子と Maya の軸 . (a) 結晶格子の a, b, c 軸 . (b) Maya の x, y, z 軸 .

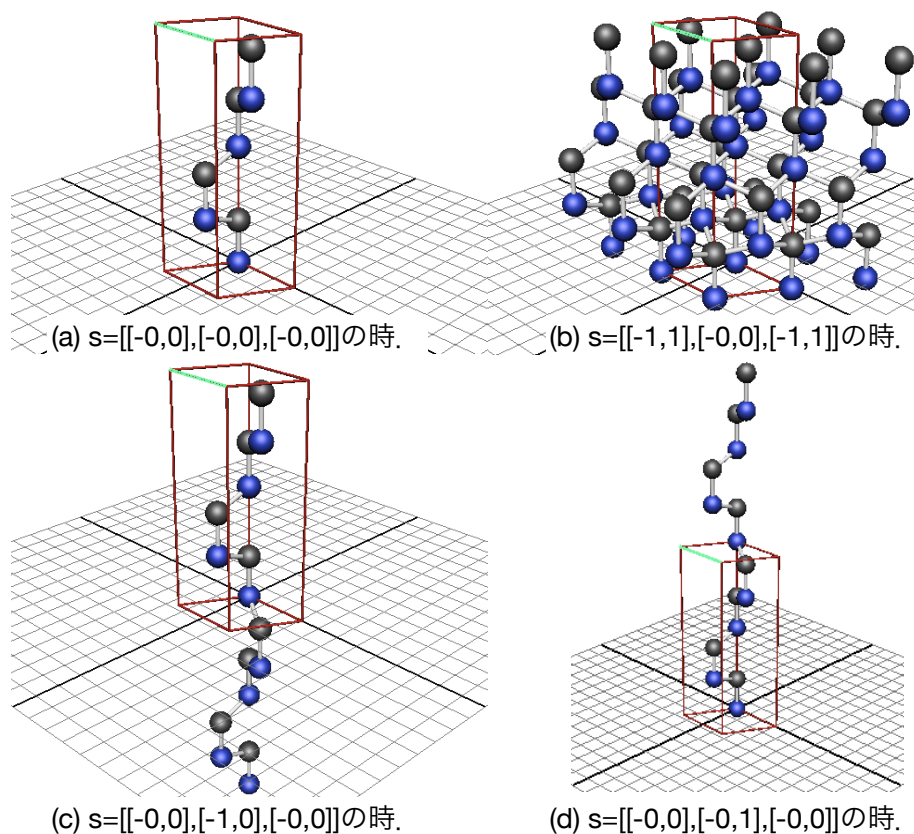


図 C.3: 拡張サイズの入力値's' ごとの結晶の様子 . (a) $s=[[-0,0],[-0,0],[-0,0]]$ の時 . (b) $s=[[-1,1],[-0,0],[-1,1]]$ の時 . (c) $s=[[-0,0],[-1,0],[-0,0]]$ の時 . (d) $s=[[-0,0],[-0,1],[-0,0]]$ の時 .

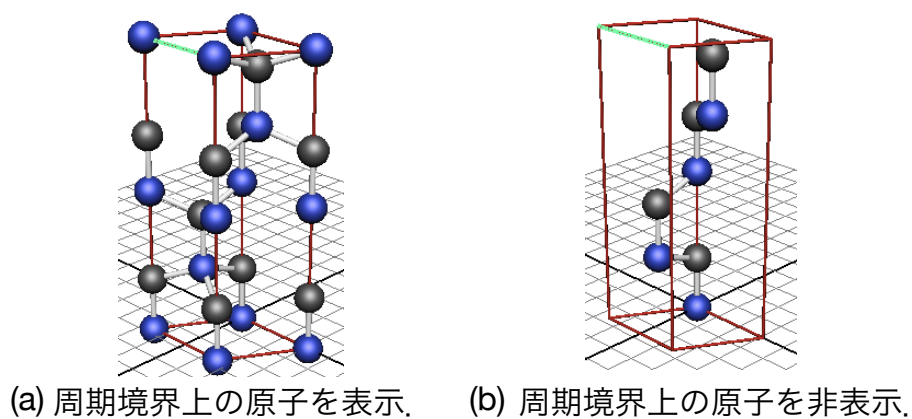


図 C.4: 周期境界上の原子の表示と非表示 (visualcoordinate_unitcell 関数を利用するか否か) . (a) 表示 . (b) 非表示

結晶格子のデータ編集を終えたので、続いて Maya クラスを呼び出し、MEL スクリプトを作成・編集する。まず原子（ボール）やボンド（スティック）などのオブジェクトに着色する。そのため、Maya クラスの関数 `color_lambert` や `color_blinn` を呼び出す。`lambert` は単調な色を作成し、`blinn` は金属質な色を作成する。これらの関数は配列に格納した RGB 値を引数とし、色の名前となる文字列を返す。引数となる配列の要素は [Red, Green, Blue] と対応しており、数値幅は 0.0 1.0 である。つまり、黄色を作成する場合、`color_lambert([1.0, 1.0, 0.0])` と記述する。また色に対して、発光度合いと透明度合いを調整できる。それらの機能を司るのが、`color_incandescence`, `color_transparency` である。これらの関数は色の名前と RGB 値の配列を引数として、その色の発光、透明度を調整する。

poscar.rb

```
main = Maya.new(filename)

color = Array.new
color[0] = main.color_lambert([1.0,1.0,1.0]) # background
main.color_incandescence(color[0],[0.5,0.5,1.0]) # 発光する。
# main.color_transparency(color[0],[0.4,0.4,0.4]) # 透明にする。

color[1] = main.color_blinn([0.2,0.2,1.0]) # Si color
color[2] = main.color_blinn([0.4,0.4,0.4]) # C color
color[3] = main.color_lambert([0.95,0.95,0.95]) # bond
main.color_incandescence(color[3],[0.2,0.2,0.2])

color[4] = main.color_lambert([1.0,0.0,0.0]) # wireframe
```

続いて、ボールやスティック等のオブジェクトを作成する。オブジェクトを作成するには、`object` 関数を呼び出す。この関数は、オブジェクトの種類と色を引数とし、オブジェクトの名前を文字列として返す。そして `translate` 関数、`scale` 関数、`rotate` 関数を呼び出し、オブジェクトの位置、スケール（大きさ）、回転を設定する。これらの関数は、オブジェクト名と [x,y,z] 配列を引数とし、そのオブジェクトを移動させる等の MEL スクリプトを作成する。以下の例では、[0,0,0] の位置に大きなボールを作成している。これは、大きなボールの中に結晶格子を置き、結晶格子の背景とするためである。

poscar.rb

```
background = main.object('sphere',color[0])
main.translate(background,[0,0,0])
main.scale(background,[500,500,500])
```

表 C.1: `object` 関数の引数となる種類と作成するオブジェクトの対応表。

種類の文字列	作成されるオブジェクト
sphere	球
cylinder	円柱
cone	円錐
nurbsCube	長方体

続いて、結晶格子の CG を作成する。原子位置のデータに応じて、`object` 関数を利用しても良いが、ここでは結晶中の原子とボンドを簡単に作成する関数を紹介する。それが、`ball` 関数と `stick` 関数である。`ball` 関数は原子座標、原子の名前となる文字列、原子の色、原子のスケール (大きさ) を引数として、結晶の原子のみを表示させる MEL スクリプトを作成する。また `stick` 関数は、ボンドをつなげる 2 種類の原子座標、ボンドの色とボンドの太さを引数とし、ボンドとなるスティックを作成する。SiC のような 2 元素の化合物の場合は、`stick(Si_position, C_position, color, [1.0,0.1,0.1])` のように記述し、Si のみの場合は、`stick(Si_position, Si_position, color, [1.0,0.1,0.1])` のように記述する。またボンドの太さは `[a, b, c]` で設定するが、`a` は意味が無く、`b=c` としなければ円柱にならない。

poscar.rb

```
c = 0
model.atom.each do |h|
  main.ball(model.cell[h],h,color[c+=1],[0.5,0.5,0.5])
end

#main.stick(model.cell[model.atom[0]],model.cell[model.atom[0]],color[3],[1.0,0.1,0.1])
main.stick(model.cell[model.atom[0]],model.cell[model.atom[1]],color[3],[1.0,0.1,0.1])
```

結晶格子を視覚化する上で、原子とボンドの他に、格子空間がわかるとありがたい。そのため、Primitive vector から成る格子空間を表示させる関数、`wireframe` を用意する。この関数は、Primitive vector が格納された行列と色を引数とし、格子空間をワイヤーフレームで囲む。この時、引数とする Primitive vector は入力した格子データから引用せずとも、自分でカスタマイズしても良い。

poscar.rb

```
main.wireframe(model.p_vec,color[4])
# a = Vector[5.0, 0.0, 0.0]
# b = Vector[0.0, 5.0, 0.0]
# c = Vector[0.0, 0.0, 5.0]
# tmp = Matrix[a, b, c]
# main.wireframe(tmp,color[4])
```

全ての編集が終われば、`end` 関数で締めくくる。プログラムの最後に出力ファイル名をプリントして置くと便利かもしれないと思わなくもなかったりするかもしれない。

poscar.rb

```
main.end
puts 'output > '+filename
```

最後に例とした poscar.rb の全容を以下に記し、このコードで視覚化された結晶の様子を図??に示す。

```

require 'pp'
require 'matrix'
include Math

Dir.chdir("lib")do
  require 'input'
  require 'Calculation'
  require 'Maya'
end

##### main

filename = 'plot.mel'
infile = ARGV[0]
model = Calc.new(infile)
model.input(POSCAR)

s = [[-0,0],[-0,0],[-0,0]]
model.supercell(s)
model.visualcoordinate_unitcell

main = Maya.new(filename)

color = Array.new
color[0] = main.color_lambert([1.0,1.0,1.0])
main.color_incandescence(color[0],[0.5,0.5,1.0])
# main.color_transparency(color[0],[0.4,0.4,0.4])

color[1] = main.color_blinn([0.2,0.2,1.0]) # Si color
color[2] = main.color_blinn([0.4,0.4,0.4]) # C color
color[3] = main.color_lambert([0.95,0.95,0.95]) # bond
main.color_incandescence(color[3],[0.2,0.2,0.2])

color[4] = main.color_lambert([1.0,0.0,0.0]) # wireframe

background = main.object('sphere',color[0])
main.translate(background,[0,0,0])
main.scale(background,[500,500,500])

c = 0
model.atom.each do |h|
  main.ball(model.cell[h],h,color[c+=1],[0.5,0.5,0.5])
end

#main.stick(model.cell[model.atom[0]],model.cell[model.atom[0]],color[3],[1.0,0.1,0.1])
main.stick(model.cell[model.atom[0]],model.cell[model.atom[1]],color[3],[1.0,0.1,0.1])

main.wireframe(model.p_vec,color[4])
# a = Vector[5.0, 0.0, 0.0]
# b = Vector[0.0, 5.0, 0.0]
# c = Vector[0.0, 0.0, 5.0]
# tmp = Matrix[a, b, c]
# main.wireframe(tmp,color[4])

main.end

puts 'output > '+filename

```

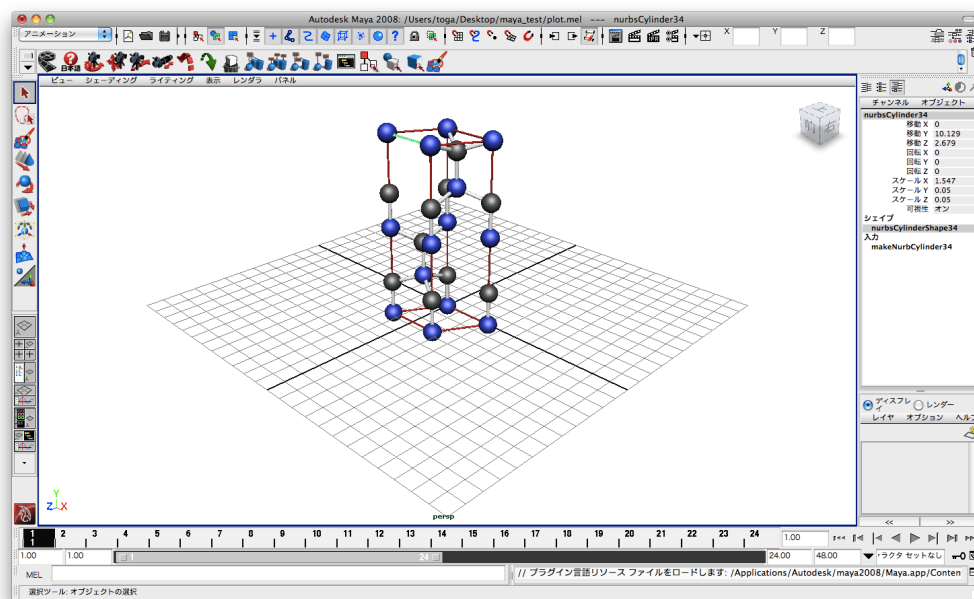


図 C.5: poscar.rb で視覚化させた例 .

実行ファイルの例として、`outcar.rb` について紹介する。`outcar.rb` は VASP の OUTCAR ファイル中の原子座標のデータから結晶格子を視覚化させるスクリプトである。ここでは積層欠陥の入った Si 中に P を置換させたモデル (SF-Si+P) の視覚化を行う。格子データ `SF-Si+P.txt` を図??に示し、`outcar.rb` の全容を以下に示す。`poscar.rb` と `outcar.rb` の大きな違いは `model.input(POSCAR)` と記述するか `model.input(OUTCAR)` と記述するかである。また SF-Si+P は Si が 15 原子、P が 1 原子であるため、全ての原子間にボンドを表示させるため、Si-Si のボンド表示用の `main.stick` と Si-P のボンド表示用 `main.stick` を記述してある。最後にこの `outcar.rb` で視覚化された結晶の様子を図??に示す。

SF-Si+P

VOLUME and BASIS-vectors are now :

```
-----
energy-cutoff :   1000.00
volume of cell :    324.24
  direct lattice vectors      reciprocal lattice vectors
  3.881665793  0.000000000  0.000000000   0.257621355  0.148737758  0.000000000
 -1.940832897  3.361621186  0.000000000   0.000000000  0.297475517  0.000000000
  0.000000000  0.000000000  24.848822609   0.000000000  0.000000000  0.040243355

length of vectors
  3.881665793  3.881665793  24.848822609   0.297475517  0.297475517  0.040243355
```

atoms: 15 1

POSITION		TOTAL-FORCE (eV/Angst)			
0.00000	0.00000	-0.05555	0.000000	0.000000	-0.003272
0.00000	0.00000	2.31160	0.000000	0.000000	0.000889
0.00000	2.24108	3.09264	0.000000	0.000000	0.000482
0.00000	2.24108	5.46004	0.000000	0.000000	0.009053
1.94083	1.12054	6.24255	0.000000	0.000000	0.003448
1.94083	1.12054	8.61103	0.000000	0.000000	-0.002254
0.00000	0.00000	9.39171	0.000000	0.000000	-0.001225
0.00000	0.00000	11.76037	0.000000	0.000000	-0.003990
0.00000	2.24108	12.51938	0.000000	0.000000	0.003835
0.00000	2.24108	14.87940	0.000000	0.000000	0.017192
0.00000	0.00000	17.79137	0.000000	0.000000	-0.007655
0.00000	2.24108	18.49790	0.000000	0.000000	-0.016639
0.00000	2.24108	20.86744	0.000000	0.000000	0.002440
1.94083	1.12054	21.64482	0.000000	0.000000	0.005961
1.94083	1.12054	24.01281	0.000000	0.000000	-0.006121
0.00000	0.00000	15.51591	0.000000	0.000000	-0.002144
total drift:		0.000000	0.000000	-0.000652	

図 C.6: SF-Si+P.txt の中身。赤字は自身で追加したモデルのタイトルとモデルの原子数を示す。黒字は VASP の出力結果である OUTCAR より、コピーしたデータ。OUTCAR の内、最も底部に記述された部分をコピーする。


```

require 'pp'
require 'matrix'
include Math

Dir.chdir("lib")do
  require 'input'
  require 'Calculation'
  require 'Maya'
end

##### main

filename = 'plot.mel'
infile = ARGV[0]
model = Calc.new(infile)
model.input(OUTCAR)

s = [[-0,0],[-0,0],[-0,0]]
model.supercell(s)
model.visualcoordinate_unitcell
model.reciprocal
main = Maya.new(filename)

color = Array.new
color[0] = main.color_lambert([1.0,1.0,1.0])
main.color_incandescence(color[0],[0.5,0.5,1.0])

color[1] = main.color_blinn([0.2,0.2,1.0]) # Si color
color[2] = main.color_blinn([0.2,1.0,0.2]) # P color
color[3] = main.color_lambert([0.95,0.95,0.95]) # bond
main.color_incandescence(color[3],[0.2,0.2,0.2])

color[4] = main.color_lambert([1.0,0.0,0.0]) # wireframe

background = main.object('sphere',color[0])
main.translate(background,[0,0,0])
main.scale(background,[500,500,500])

c = 0
model.atom.each do |h|
  main.ball(model.cell[h],h,color[c+=1],[0.5,0.5,0.5])
end

main.stick(model.cell[model.atom[0]],model.cell[model.atom[0]],color[3],[1.0,0.1,0.1])
main.stick(model.cell[model.atom[0]],model.cell[model.atom[1]],color[3],[1.0,0.1,0.1])

main.wireframe(model.p_vec,color[4])

main.end

puts 'output > '+filename

```

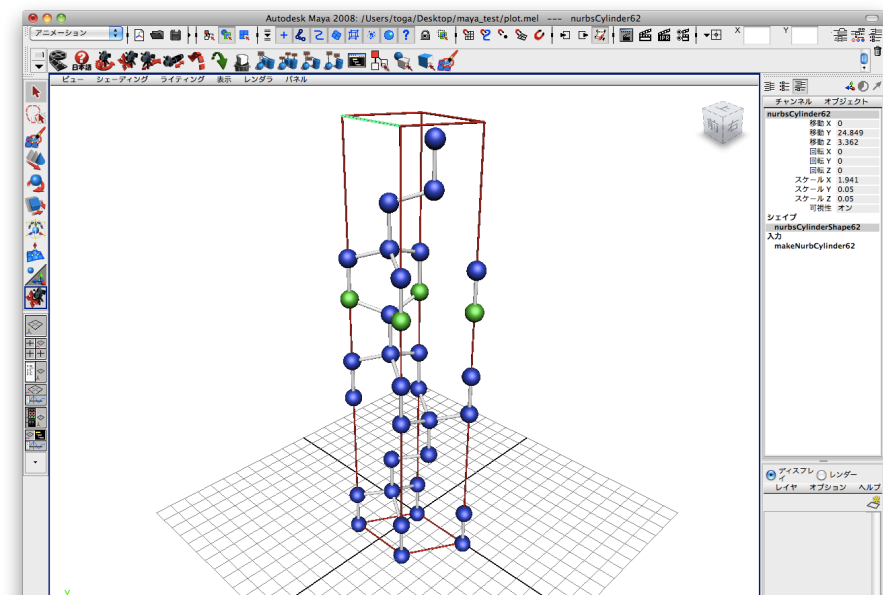


図 C.7: outcar.rb で視覚化させた例 .

最後に `anime.rb` について紹介する。`anime.rb` は簡単なアニメーションを作成するスクリプトである。`anime.rb` の内容を以下に示す。ボールやスティック等のオブジェクトを動かすには、`anime` 関数を利用する。この関数は、フレームナンバー、オブジェクトの名前、`[x,y,z]` パラメータ、タグ (`t, s, r` の3つの内一つを選択) を引数とし、フレームナンバー時のオブジェクトの挙動を `[x,y,z]` パラメータによって設定する。3つのタグは、それぞれ移動 or 座標 (`t=translate`)、スケール (`s=scale`)、回転角度 (`r=rotate`) を示す。`anime.rb` で作成されたアニメーションを図??に示す。

`poscar.rb`

```
require 'pp'
require 'matrix'
include Math

Dir.chdir("lib")do
  require 'input'
  require 'Calculation'
  require 'Maya'
end

##### main

filename = 'plot.mel'
main = Maya.new(filename)

color = Array.new
color[0] = main.color_blinn([1.0,1.0,0.0])
color[1] = main.color_lambert([0.0,1.0,1.0])
main.color_transparency(color[1],[0.4,0.4,0.4])

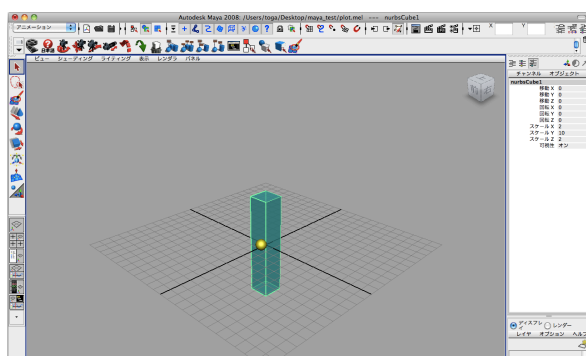
ball = main.object('sphere',color[0])
main.scale(ball,[0.5,0.5,0.5])

cube = main.object('nurbsCube',color[1])
main.scale(cube,[2.0,10.0,2.0])

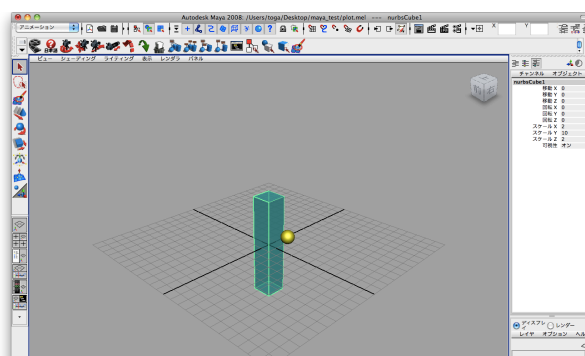
range = 1.0
24.times do |i|
  x = 2*cos(i)
  y = 24.0/(24.0-i.to_f)
  z = 2*sin(i)
  main.anime(i,ball,[x,y,z],'t')
  main.anime(i,ball,[0.5*y,0.5*y,0.5*y],'s')
end

main.end

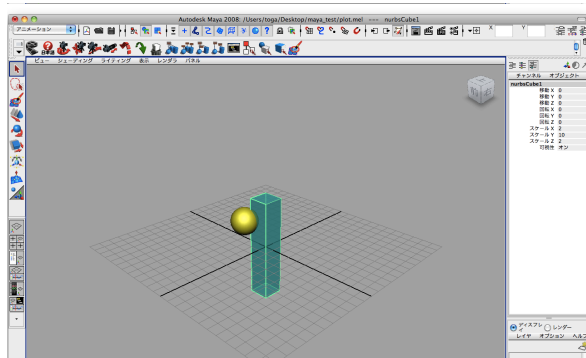
puts 'output > '+filename
```



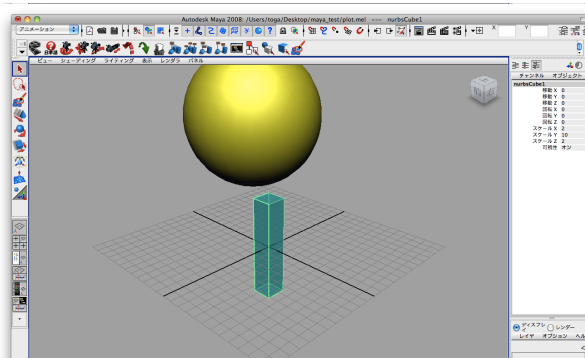
(a) 1 frame



(b) 6 frame



(c) 15 frame



(d) 22 frame

図 C.8: anime.rb で視覚化させた例 .

C.3 ライブラリ

ライブラリとして作成した `input.rb`, `Calculation.rb`, `Maya.rb` の中身を簡単に紹介する。
`input.rb` は, POSCAR, OUTCAR の原子座標を読み込むブロックが記述されている。読み込むデータファイルの形式に応じて, POSCAR, OUTCAR ブロックを呼び出せばよい。

`input.rb`

```
POSCAR = lambda do |infile,data| # POSCAR 読み取り
  text = infile.split(/\n+/)
  i = 0; box=[]; numbers=[]
  while i<text.size do
    if text[i]~/Direct/ then
      if text[i-1]~/Selective dynamics/ then
        flag = i-1
      else flag = i
      end
      3.times do |j|
        a = text[flag-4+j].split(/ +/)
        box.push Vector[a[1].to_f,a[3].to_f,a[2].to_f]
      end
      data.p_vec = Matrix[[box[0][0], box[2][0], box[1][0]],
                          [box[0][1], box[2][1], box[1][1]],
                          [box[0][2], box[2][2], box[1][2]]]
      a = text[flag-1].split(/ +/)

      while (tmp=a.pop)!=" " do # 原子数の取り出し
        numbers.unshift tmp.to_i
      end

      head = 0
      numbers.each do |num|
        data.atom.push 'elem'+head.to_s
        data.cell['elem'+head.to_s] = []
        num.times do
          a = text[i+1].split(/ +/)
          vec = Vector[a[1].to_f, a[3].to_f, a[2].to_f]
          data.cell['elem'+head.to_s].push data.p_vec*vec
          i += 1
        end
        head += 1
      end
      break
    end
    i += 1
  end
end
```

```

OUTCAR = lambda do |infile,data|# OUTCAR 読み取り
  text = infile.split(/\n/)
  i = 0; box=[]; numbers=[]
  while i<text.size do
    # Primitive Vectors(direct lattice vectors) の取り出し .
    if text[i]=~/direct lattice vectors/ then
      i.upto(i+2) do |k|
        a = text[k+1].split(/ +/)
        box.push Vector[a[1].to_f,a[3].to_f,a[2].to_f]
      end
      data.p_vec = Matrix[[box[0][0],box[2][0],box[1][0]],
                          [box[0][1],box[2][1],box[1][1]],
                          [box[0][2],box[2][2],box[1][2]]]
    end

    if text[i]=~/atoms:/ then # 原子数の取り出し .
      a = text[i].split(/ +/)
      while (tmp=a.pop)!='atoms:' do
        numbers.unshift tmp.to_i
      end
    end

    if text[i]=~/POSITION/ then # 原子位置を格納
      head = 0
      numbers.each do |num|
        data.atom.push 'elem'+head.to_s
        data.cell['elem'+head.to_s] = []
        num.times do
          a = text[i+2].split(/ +/)
          vec = Vector[a[1].to_f,a[3].to_f,a[2].to_f]
          data.cell[data.atom[head]].push vec
          i += 1
        end
        head += 1
      end
      break
    end
    i += 1
  end
end

```

[Calculation.rb](#) は、結晶格子のデータを読み込む、あるいはカスタマイズする時に呼び出すクラスである。

```

class Calc
  attr_accessor :cell, :p_vec, :r_vec, :atom

  def initialize(filename)
    @infile = File.read(filename)
    @cell = Hash.new
    @atom = Array.new
  end

  def input(input_formatter)
    input_formatter.call(@infile,self)
  end

  def bondlength(a,b)
    len = 999
    a.each do |v1|
      b.each do |v2|
        tmp = v1-v2; l = tmp.r
        if 10**(-3)<l && l<len then len = l end
      end
    end
    return len
  end

  def visualcoordinate_unitcell
    box = {'tff'=>[ Vector[1,0,0] ], 'ftf'=>[ Vector[0,1,0] ], 'fft'=>[ Vector[0,0,1] ],
           'tft'=>[ Vector[1,1,0], Vector[1,0,0], Vector[0,1,0] ],
           'tft'=>[ Vector[1,0,1], Vector[1,0,0], Vector[0,0,1] ],
           'ftt'=>[ Vector[0,1,1], Vector[0,1,0], Vector[0,0,1] ],
           'ttt'=>[ Vector[1,0,0], Vector[0,1,0], Vector[0,0,1],
                    Vector[1,1,0], Vector[1,0,1], Vector[0,1,1], Vector[1,1,1]]}

    mi = @p_vec.inverse
    @atom.each do |n|
      @cell[n].size.times do |i|
        b = mi*@cell[n][i]
        head = ''
        3.times do |j|
          if b[j]>-10**(-4)&&b[j]<10**(-4) then
            head += 't'
          else
            head += 'f'
          end
        end
        if box.key?(head) then
          box[head].each do |v|
            @cell[n].push @cell[n][i]+@p_vec*v
          end
        end
      end
    end
  end

  def supercell(s)
    @atom.each do |h|
      n = @cell[h].size
      for x in s[0][0]..s[0][1] do
        for y in s[1][0]..s[1][1] do
          for z in s[2][0]..s[2][1] do
            if x==0&&y==0&&z==0 then next end
            n.times do |i|
              @cell[h] += [@cell[h][i]+@p_vec*(Vector[x,y,z])]
            end
          end
        end
      end
      # @cell[h].uniq!
    end
  end
end
# 次ページへ続く

```

```

def inner(a,b) #内積
  t = 0.0
  3.times do |i|
    t += a[i]*b[i]
  end
  return t
end
def cross(a,b) #外積
  e1 = a[1]*b[2]-a[2]*b[1]
  e2 = a[2]*b[0]-a[0]*b[2]
  e3 = a[0]*b[1]-a[1]*b[0]
  return Vector[e1,e2,e3]
end
def reciprocal
  a1 = @p_vec.column(0)
  a2 = @p_vec.column(2)
  a3 = @p_vec.column(1)
  a1 = Vector[a1[0],a1[2],a1[1]]
  a2 = Vector[a2[0],a2[2],a2[1]]
  a3 = Vector[a3[0],a3[2],a3[1]]

  # 逆格子ベクトル (reciprocal vector) について .
  # 因子 2*PI を結晶学者は使わないが, 固体物理学では使う方が便利 .
  # VASP で出力される OUTCAR の逆格子ベクトルでは因子 2*PI を用いていない .
  # Ref: 宇野良清, 津屋昇, 新聞駒二郎, 森田章, 山下次郎 共訳,
  # 「Charles Kittel キットル固体物理学入門 ハードカバー版」第 8 版, 丸善, (2005), p32 .

  b1 = ( 1.0 / inner( a1, cross(a2,a3) ) ) * cross(a2,a3)
  b2 = ( 1.0 / inner( a2, cross(a3,a1) ) ) * cross(a3,a1)
  b3 = ( 1.0 / inner( a3, cross(a1,a2) ) ) * cross(a1,a2)
  b1 = Vector[b1[0],b1[2],b1[1]]
  b2 = Vector[b2[0],b2[2],b2[1]]
  b3 = Vector[b3[0],b3[2],b3[1]]

  @r_vec = Matrix[b1,b3,b2]
end
def gui(dataname)
  slider_sphere_scale(dataname)
end
end

```

Maya.rb は, Maya を操作する MEL スクリプトを出力する様々な関数を持つクラスである .


```

class Maya < Calc
  $color_lambert_count = 0
  $color_blinn_count = 0
  $object_count = -1

  $object_flag = {'t'=>true,'s'=>true,'r'=>true}
  $anime_flag = {'t'=>true,'s'=>true,'r'=>true}

  $cylinder_proc_count = 0
  $ball_proc_count = 0
  $wire_frame_count = 0

  $sphere_object_count = 1
  $cylinder_object_count = 1
  $slider_sphere_scale_count = 1
  def initialize(filename)
    @f = File.open(filename,'w')
  end
  def end
    @f.close
  end
  def color_lambert(rgb) # 単調な色
    cc = ($color_lambert_count += 1)
    @f.printf("shadingNode -asShader lambert;\n")
    @f.printf("sets -renderable true -noSurfaceShader true -empty -name lambert%dSG;\n",cc)
    @f.printf("connectAttr -f lambert%d.outColor lambert%dSG.surfaceShader;\n",cc,cc)
    @f.printf("select -r lambert%d;\n",cc)
    @f.printf("setAttr \"lambert%d.color\" -type double3 %1.5f %1.5f %1.5f;\n",
      cc,rgb[0],rgb[1],rgb[2])
    return tmp = 'lambert'+cc.to_s
  end
  def color_blinn(rgb) # 金属光沢のある色
    cc = ($color_blinn_count += 1)
    @f.printf("shadingNode -asShader blinn;\n")
    @f.printf("sets -renderable true -noSurfaceShader true -empty -name blinn%dSG;;\n",cc)
    @f.printf("connectAttr -f blinn%d.outColor blinn%dSG.surfaceShader;\n",cc,cc)
    @f.printf("select -r blinn%d;\n",cc)
    @f.printf("setAttr \"blinn%d.color\" -type double3 %1.5f %1.5f %1.5f ;\n",
      cc,rgb[0],rgb[1],rgb[2])
    return tmp = 'blinn'+cc.to_s
  end
  def color_transparency(color,rgb) # 透明にする
    @f.printf("select -r %s;\n",color)
    @f.printf("setAttr \"%s.transparencyR\" %1.5f;\n",color,rgb[0])
    @f.printf("setAttr \"%s.transparencyG\" %1.5f;\n",color,rgb[1])
    @f.printf("setAttr \"%s.transparencyB\" %1.5f;\n",color,rgb[2])
  end
  def color_incandescence(color,rgb) # 光る
    @f.printf("select -r %s;\n",color)
    @f.printf("setAttr \"%s.incandescenceR\" %1.5f;\n",color,rgb[0])
    @f.printf("setAttr \"%s.incandescenceG\" %1.5f;\n",color,rgb[1])
    @f.printf("setAttr \"%s.incandescenceB\" %1.5f;\n",color,rgb[2])
  end
  def object(obj,color) # オブジェクト作成 . ball, stick, cube, plane, cone 等
    name = 'object'+($object_count+=1).to_s
    @f.printf("string %s[];\n",name)
    @f.printf("%s = '%s';\n",name,obj)
    @f.printf("sets -e -forceElement %sSG;\n",color)
    return name
  end
end
# 次ページへ続く

```

```

def translate(name,xyz) # オブジェクトの移動 (座標)
  position_scale_rotate('t',name,xyz)
end
def scale(name,xyz) # オブジェクトのスケール
  position_scale_rotate('s',name,xyz)
end
def rotate(name,xyz) # オブジェクトの回転
  position_scale_rotate('r',name,xyz)
end
def position_scale_rotate(tag,name,xyz)
  if $object_flag[tag] then
    @f.printf("global proc object_%s(string $names[],float $x,float $y,float $z)\n",tag)
    @f.printf("{\n")
    @f.printf("setAttr ($names[0] + \"%.sx\") $x;\n",tag)
    @f.printf("setAttr ($names[0] + \"%.sy\") $y;\n",tag)
    @f.printf("setAttr ($names[0] + \"%.sz\") $z;\n",tag)
    @f.printf("}\n")
    $object_flag[tag] = false
  end
  @f.printf("object_%s($s,%7.5f,%7.5f,%7.5f);\n",tag,name,xyz[0],xyz[1],xyz[2])
end
def anime(time,name,xyz,tag)
  # time はアニメーションのキーフレーム値
  # name は object で設定した name
  # xyz はキーフレーム時の translate, scale, rotate 値が格納した配列 [x,y,x]
  # tag はアニメーションが translate, scale, rotate のいずれかを特定するタグ. 入力は't, s, r' のいづ
  れか .
  # t=translate, s=scale, r=rotate を示す .
  if $anime_flag[tag] then
    @f.printf("global proc anime_%s(string $names[],int $c_time,float $x,float $y,float $z)\n",tag)
    @f.printf("{\n")
    @f.printf("currentTime $c_time;\n")
    @f.printf("setAttr ($names[0] + \"%.sx\") $x;\n",tag)
    @f.printf("setAttr ($names[0] + \"%.sy\") $y;\n",tag)
    @f.printf("setAttr ($names[0] + \"%.sz\") $z;\n",tag)
    @f.printf("setKeyframe -at \"%sx\" $names[0];\n",tag)
    @f.printf("setKeyframe -at \"%sy\" $names[0];\n",tag)
    @f.printf("setKeyframe -at \"%sz\" $names[0];\n",tag)
    @f.printf("}\n")
    $object_flag[tag] = false
  end
  @f.printf("anime_%s($s,%d,%7.5f,%7.5f,%7.5f);\n",tag,name,time,xyz[0],xyz[1],xyz[2])
end
# 次ページへ続く

```

```

def ball(ball,name,color,scale) #結晶格子の視覚化 , ball のみ
  dd = $ball_proc_count
  $ball_proc_count += 1
  @f.printf("global proc make%d_%s(float $x,float $y,float $z)\n",dd,name)
  @f.printf("{\n")
  @f.printf("sphere;\n")
  @f.printf("scale %7.5f %7.5f %7.5f;\n",scale[0],scale[1],scale[2])
  @f.printf("move $x $y $z;\n")
  @f.printf("sets -e -forceElement %sSG;\n",color)
  @f.printf("}\n")
  first_number = $sphere_object_count
  ball.each do |v|
    @f.printf("make%d_%s(%7.5f,%7.5f,%7.5f);\n",dd,name,v[0],v[1],v[2])
    $sphere_object_count += 1
  end
  end_number = $sphere_object_count
end

def stick(ball1,ball2,color,scale) #結晶格子の視覚化 , stick のみ
  dd = $cylinder_proc_count
  $cylinder_proc_count += 1
  @f.printf("global proc make_cylinder%d(float $x,float $y,float $z,float $len,float $yy,float $zz)\n",dd)
  @f.printf("{\n")
  @f.printf("cylinder;\n")
  @f.printf("scale $len %7.5f %7.5f;\n",scale[1],scale[2])
  @f.printf("move $x $y $z;\n")
  @f.printf("rotate 0 $yy $zz;\n")
  @f.printf("sets -e -forceElement %sSG;\n",color)
  @f.printf("}\n")

  len = bondlength(ball1,ball2)*1.1
  #len = 2.5 # spiral だけの特別
  pair = Array.new
  ball1.each do |v1|
    ball2.each do |v2|
      tmp_vector = v1 - v2
      length = tmp_vector.r
      if(0<length && length<len)then
        if(tmp_vector[0]<0)then
          tmp_vector = tmp_vector*-1
        end
        pair.push [tmp_vector, (v1+v2)*0.5]
      end
    end
  end
  pair.uniq!
  pair.each do |v|
    ro = rotate_angle(v[0])
    @f.printf("make_cylinder%d(%7.5f,%7.5f,%7.5f,%7.5f,%7.5f,%7.5f);\n",
      dd,v[1][0],v[1][1],v[1][2],(v[0].r)*0.5,ro[1],ro[2])
  end
end
# 次ページへ続く

```

```

def rotate_angle(vec)
  # ex=Vector[1,0,0] に対し, vec の回転成分を計算 (ex を y 軸に対し yy 回転, z 軸に対し zz 回転させると vec と一致する.)
  x = vec[0]
  y = vec[1]
  z = vec[2]
  len = vec.r
  theta2 = -asin(z/len)
  tmp = y/(len*cos(theta2))
  if(-1.001<=tmp && tmp<-0.999)then
    theta3 = -1.5707963267949
  elsif(0.999<=tmp && tmp<1.001)then
    theta3 = 1.5707963267949
  else
    theta3 = asin(tmp)
  end
  yy = theta2*180/PI
  zz = theta3*180/PI
  return [0,yy,zz]
end

def wireframe(p_vec,color)
  d = $wire_frame_count
  $wire_frame_count += 1
  @f.printf("global proc make_wireframe%d(float $x,float $y,float $z,float $b2,float $a2,float $sa)\n",d)
  @f.printf("{\n")
  @f.printf("cylinder;\n")
  @f.printf("scale $sa 0.05 0.05;\n")
  @f.printf("move $x $y $z;\n")
  @f.printf("rotate 0 $b2 $a2;\n")
  @f.printf("sets -e -forceElement %sSG;\n",color)
  @f.printf("}\n")

  v = [p_vec.column(0), p_vec.column(1), p_vec.column(2)]
  wire = [ Vector[0.0, 0.0, 0.0],
    Vector[v[0][0],v[0][1],v[0][2]],
    Vector[v[1][0],v[1][1],v[1][2]],
    Vector[v[2][0],v[2][1],v[2][2]],
    Vector[v[0][0]+v[1][0],v[0][1]+v[1][1],v[0][2]+v[1][2]],
    Vector[v[0][0]+v[2][0],v[0][1]+v[2][1],v[0][2]+v[2][2]],
    Vector[v[1][0]+v[2][0],v[1][1]+v[2][1],v[1][2]+v[2][2]],
    Vector[v[0][0]+v[1][0]+v[2][0],v[0][1]+v[1][1]+v[2][1],v[0][2]+v[1][2]+v[2][2]]]

  pair = [[0,1],[0,2],[0,3],[1,4],[1,5],[2,4],
    [2,6],[3,5],[3,6],[4,7],[5,7],[6,7]]

  pair.each do |a|
    v = (wire[a[0]]+wire[a[1]])*0.5
    dir = wire[a[0]]-wire[a[1]]
    if dir[0]<0 then dir = dir*(-1) end
    ro = rotate_angle(dir)
    len = (dir.r)*0.5
    @f.printf("make_wireframe%d(%7.5f,%7.5f,%7.5f,%7.5f,%7.5f,%7.5f);\n",d,v[0],v[1],v[2],ro[1],ro[2],len)
  end
end
# 次ページへ続く

```

```

def slider_sphere_scale(dataname)
  d = $slider_sphere_scale_count
  $slider_sphere_scale_count += 1
  @f.printf("global proc atom_scale_change%d()\n{\n",d)
  @atom.each do |label|
    n = label + d.to_s
    num = @group['sphere'].shift
    @f.printf(" float $val%s;\n",n)
    @f.printf(" $val%s = 'floatSliderGrp -q -value radiusSlider%s';\n",n,n)
    @f.printf("for($i=%d; $i<%d;$i++)\n{\n",num[0],num[1])
    @f.printf("setAttr (\\"nurbsSphere\\"+$i+\\".sx\\") $val%s;\n",n)
    @f.printf("setAttr (\\"nurbsSphere\\"+$i+\\".sy\\") $val%s;\n",n)
    @f.printf("setAttr (\\"nurbsSphere\\"+$i+\\".sz\\") $val%s;\n",n)
    @f.printf("}\n")
  end
  @f.printf("}\n")
  @f.printf("window -title \\"Atom Scale of %s\\";\n",dataname)
  @f.printf("columnLayout;\n")
  @f.printf("text -label \"%s : change the scale of atoms.\\\";\n",dataname)
  @atom.each do |label|
    n = label + d.to_s
    @f.printf("floatSliderGrp -label \"%s scale\" -field true\n",label)
    @f.printf(" -min 0.0 -max 3.0 -step 0.01 -value 0.5 radiusSlider%s;\n",n)
  end
  @f.printf("button -label \\"OK\\" -command \\"atom_scale_change%d()\\\";\n",d)
  @f.printf("showWindow;\n")
end
$group_count = 0
def dislocation_cell(name)
  $group_count += 1
  con = 'select -r '
  name.each do |e|
    con += '$'+e+'[0] '
  end
  con = con[0..con.length-1]+";"
  @f.printf("%s\n",con)
  @f.printf("group;\n")
  @f.printf("string $g%dname[];\n",$group_count)
  @f.printf("select -r group%d;\n",$group_count)
  @f.printf("$g%dname = 'ls -sl';\n",$group_count)
end
def dislocation_gui(n)
  @f.printf("global proc dislocation()\n{\n")
  @f.printf("global string $g1name[];\n")
  @f.printf("global string $g2name[];\n")
  @f.printf("float $val;\nint $i;\n")
  @f.printf("$val = 'floatSliderGrp -q -value radiusSlider1';\n")
  @f.printf("setAttr ($g1name[0]+\\".ry\\") $val;\n")
  @f.printf("$val = $val * -1.0;\n")
  @f.printf("setAttr ($g2name[0]+\\".ry\\") $val;\n")
  @f.printf("for($i=2;$i<%d;$i++)\n{\n",n+1)
  @f.printf("if($val>30)\n{\n")
  @f.printf("setAttr (\\"nurbsSphere\\"+$i+\\".sx\\") 0;\n")
  @f.printf("setAttr (\\"nurbsSphere\\"+$i+\\".sy\\") 0;\n")
  @f.printf("setAttr (\\"nurbsSphere\\"+$i+\\".sz\\") 0;\n")
  @f.printf("}\n}\n")
  @f.printf("window -title \\"Dislocation\\";\n")
  @f.printf("columnLayout;\n")
  @f.printf("text -label \\"rotate\\";\n")
  @f.printf("floatSliderGrp -label \\"rotate\" -field true\n")
  @f.printf(" -min 0.0 -max 360.0 -step 0.1 -value 0 radiusSlider1;\n")
  @f.printf("button -label \\"OK\\" -command \\"dislocation()\\\";\n")
  @f.printf("showWindow;\n")
end
end
end

```

