修士論文

SiC単結晶成長の
原子レベルシミュレーション

関西学院大学　理工学研究科　情報科学専攻
M6433　坂本　憲

2008年　3月

指導教員　　西谷　滋人　教授

概 要

炭化珪素（SiC）は優れた物性的特徴から次世代パワーエレクトロニクス半導体材料として注目されている．現在，SiC のバルク成長には昇華法，いわゆる気相成長法が主に使用されている．液相成長は，Si 中の炭素溶解度が極めて低いため成長速度が遅く現実的に困難と考えられてきた．

ところが，関学の金子教授らにより開発された，準安定溶媒エピタキシー (Metastable Solvent Epitaxy) と呼ばれる新プロセスによって，SiC の液相成長を実現した．MSE では 4H-SiC を基板 (seed)，3C-SiC を原料板 (feed) として使用し，その間に溶媒として液体 Si の薄膜を挟み込んだ装置構成をとる．このプロセスは traveling solvent と同じ構造をとるが温度勾配が存在しない．この系における結晶成長の駆動力は SiC の結晶構造の違いからくる化学ポテンシャル差である．

MSE は全く新しい液相成長法であり，その成長機構に関しては不明な点が多い．化学ポテンシャル差により生じる溶媒中の炭素濃度勾配は液体 Si の炭素溶解度が極めて低いため，WDX（波長分散 X 線）による濃度プロファイル解析でも測定できない．また MSE の SiC 単結晶成長時に，結晶表面が原子レベルでどのようになっているのかも定かではない．

本研究では結晶成長の素過程である「放出」「拡散」「吸着」から溶媒中の炭素濃度プロファイルを再現した．これにより結晶成長速度シミュレーションが可能となり，成長速度の溶媒幅依存性は直線を示すという結果を得た．実験結果でも成長速度の溶媒幅依存性は直線を示していることから，MSE プロセスの律速過程は溶媒中の「拡散」であるという知見を得た．

次に結晶表面における原子位置の第一原理計算を行った．対象を Si 面,Si 修飾をした C 面で行った結果，C 面よりも Si 面に C 原子が付着する方が安定であり，表面拡散は Si 修飾をした C 面のほうが速いという仮説を得るに至った．

またコンピュータグラフィックソフト MAYA により SiC 結晶構造，結晶表面を表示した．これより結晶表面，結晶構造を原子レベルでより視覚的に直接理解する事が可能になった．

目 次

第1章　序論

1.1　パワーエレクトロニクス半導体

1.1.1　SiとSiC

現在，最も多く用いられているパワーエレクトロニクス半導体材料はSi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，耐熱性にすぐれたSiCに大きな期待がかかっている．Siの理論限界とは1つは耐熱とオン抵抗とのトレードオフであり，もう1つは耐熱(接触温度)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1.1にSiCとSiとの物理定数の比較を示した．SiCはワイ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倍必要である．このバンドギャップによりSiにくらべSiCの絶縁破壊電解強度は10倍にもなる[2]．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[2]．また耐圧とオン電圧とのトレードオフでは，絶縁破壊電解強度が影響している．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1.1に示すようにMOSFET構造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持っている．　またSiCは熱伝導特性もSiに比べ良好で銅並みである．このようにSiCで実現できる低オン抵抗，高温動作，高熱伝導は，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．

表 1.1: SiC と Si の物理定数比較 [2].

材料	バンドギャップ (eV)	破壊電解強度 (MV/cm)	熱伝導率 (W/cmK)
SiC(4H)	3.25	3	4.9
Si	1.1	0.3	1.5

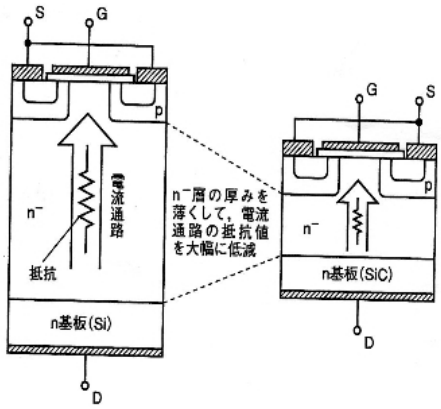

図 1.1: Si-MOSFET と SiC-MOSFET の違い [2].

1.2　液相による代表的な結晶成長法

以下に代表的な成長法を SiC に適用されていないが SiC を例にとって示す．ただし，実際には液体 Si 中の炭素溶解度が極めて低いため，従来の結晶成長手法には液層成長法は非常に困難である．

1.2.1　Traveling Solvent

図 1.2 に Traveling Solvent 法の模式図を示した．Traveling Solvent 法では反応が進むにつれ，溶媒が自動的に移動して行くようにみえるためこのように呼ばれる．基板側，原料板側に Δ T の温度差を付けこれを駆動力として結晶成長が進む．温度の高い界面では結晶は溶け出し，温度の低い界面では結晶が析出する．この際溶媒が温度の高い所に向かって移動して行くように見える．

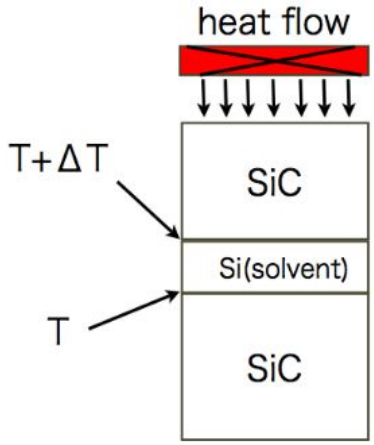

図 1.2: traveling solvent 法.

1.2.2　Floating zone

図 1.3 に Floating Zone 法の模式図を示した．Floating Zone 法では溶媒を横から加熱し，図 1.3 に示したように矢印方向に全体を少しずつ上に移動させる．すると heat flow から抜けた部分は温度が下がり，その分下の

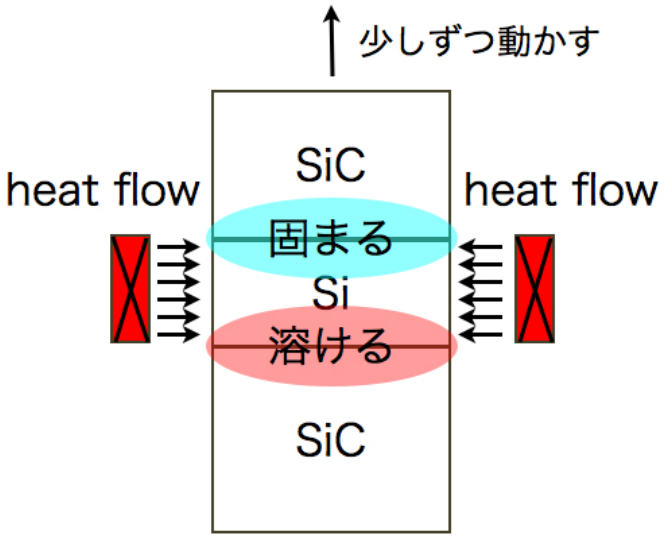

図 1.3: floating zone 法.

SiC が暖められる．冷やされたところでは溶解度が下がるため結晶が析出し，また暖められたところでは，溶解度が上がるため溶け出してくる．Floating Zone 法ではこのようにして結晶成長が進む．しかし全体を矢印の方向に移動させる速度が遅いため結晶成長速度は非常に遅い．また溶媒を暖めているため温度にどうしてもムラが生じ対流が起こる．また溶質濃度が一定に保たれず安定した単結晶成長が得られない可能性がある．

1.3 従来の SiC の結晶成長法

1.3.1 昇華法（昇華再結晶法）

昇華法（昇華再結晶法）は SiC 単結晶作成に用いられている最も一般的な手法であり，非熱平衡状態でプロセスは進行する．図 1.4 にその概略図を示した．昇華再結晶法での SiC 単結晶の成長は閉ざされた黒煙るつぼの中で 2500 ℃程度の環境のもとで進行するため，その成長過程は明らかにされていない．しかし成長過程は大きく 3 段階に分けられる．まず原料である SiC が昇華し，Si,SiC_2,Si_2C を主に含んだ昇華ガスとなり，次に昇華ガスが種結晶へと拡散によって輸送され，最後に昇華ガスが種結晶上で再結晶化するものと考えることができる．そのため　SiC 単結晶

の高品質化には昇華ガスの高純度化，また 2000 ℃を超える環境に黒煙るつぼや断熱材がさらされるため，含有している不純物等の低減が必要であり，わずかな不純物が種結晶上で取り込まれただけで欠陥発生の原因となる．また，多種類のガスを多く含んだ昇華ガスを種結晶上に輸送しなければならないため，種結晶上でガス組成にムラが生じやすく，マイクロパイプ欠陥が生じやすい．

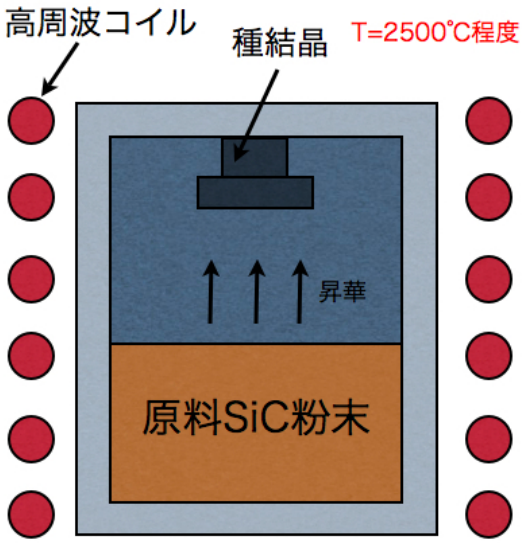

図 1.4: 昇華再結晶法の概略図.

1.3.2 dip 法

dip 法とは液相から SiC の単結晶を成長させる手法である．液相成長法では気相成長法（昇華法）に比べ熱平衡状態で進行するためマイクロパイプ欠陥のような格子欠陥も少なく高品質な SiC 単結晶が得られると考えられる．しかし液体シリコン中の炭素溶解度（二元系溶液を用いた場合）が極めて低いため気相成長法に比べ，SiC 単結晶の成長速度が非常に遅く，単結晶成長には向かないと考えられてきた．図 1.5 に dip 法の概略図を示した．ヒータ（高周波コイル）種結晶付近とるつぼのそこ付近に温度差をつけ，温度勾配を駆動力として結晶成長は進行する．dip 法ではこの液体 Si 中の炭素溶解度を二元溶液に第三元素（遷移金属）を加えた

Si-C-X 系溶液を用いる事で液体 Si 中の炭素溶解度を飛躍的に上げ，成長速度を早めている．

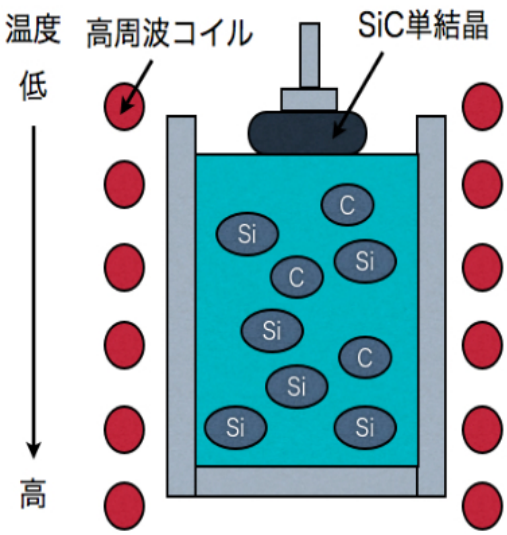

図 1.5: dip 法の概略図.

1.4 準安定溶媒エピタキシー (MSE)

準安定溶媒エピタキシー (MSE) とは，金子教授 (関西学院大学　理工学部) によって提案された温度一定で駆動力を濃度差とする新たな結晶成長法である．従来の結晶成長では温度差を駆動力としてきた．ミョウバンの結晶成長がその代表的な例である．まず高温の水にミョウバンを溶かす．次にその混合水を冷却する．すると溶解度の違いからミョウバンの結晶が析出する．これが駆動力を温度差とする結晶成長プロセスである．しかし準安定溶媒エピタキシーには温度勾配はない，3C,4H の化学ポテンシャルの違いから等温で基板側と原料板側に化学ポテンシャル差が生じ，結晶成長が進む．次章で MSE についてより詳しく述べる．

第2章　MSE(Metastable Solvent Epitaxy)の原理

図 2.1 に MSE プロセスの概略図を示した．MSE では 3C,4H のそれぞれの固液界面での固溶限差すなわち化学ポテンシャル差を駆動力としてプロセスは進行する．feed（原料板）から seed（基板）へ液体 Si により炭素が輸送され seed（基板）上に 4H-SiC の単結晶が成長する．以下で MSE プロセスの原理について解説する．

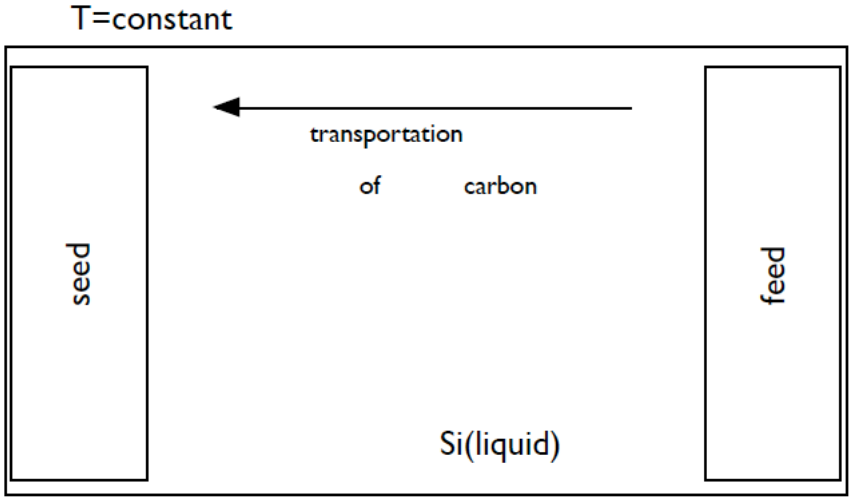

図 2.1: MSE プロセスの概略図.

2.1　ミョウバンの結晶成長

駆動力を温度差とする，結晶成長の代表的な例であるミョウバンの結晶成長を状態図を使って示す．図 2.2 に水とミョウバンの状態図を示した．

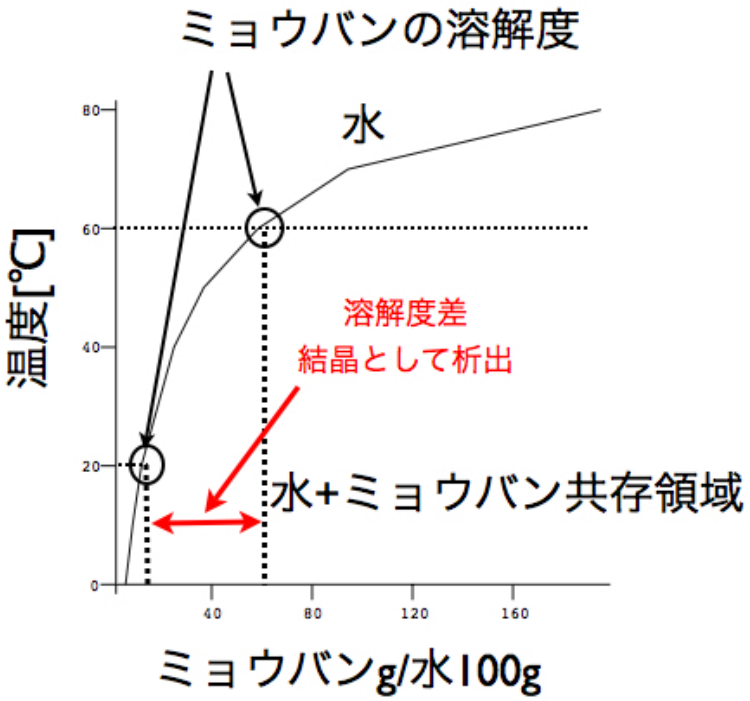

図 2.2: ミョウバンの結晶成長の状態図.

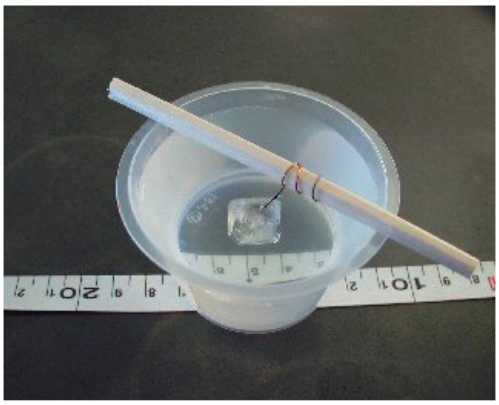

図 2.3: ミョウバンの単結晶.

ミョウバンの結晶成長は図 2.2 を使って説明できる．この混合水を 60 ℃から 20 ℃に下げると図 2.2 に示したように溶解度に差が生じる．この溶解度差分だけ図 2.3 のような結晶が生じる．

2.2 化学ポテンシャル

化学ポテンシャル:μ とは粒子 1 つあたりのギブスの自由エネルギーの事である．N を粒子数，温度を T, 圧力を P，ギブスの自由エネルギーを G として以下の式で定義される．

$$\mu = \left(\frac{\partial G}{\partial N}\right)_{\mathrm{TP}} \tag{2.1}$$

$$G = \mu N \tag{2.2}$$

x_i を組成とすると

$$G = \sum_{i=1} \mu_i x_i \tag{2.3}$$

となり，組成から自由エネルギーが求まる．

また A 原子，B 原子が混合して 1mol の溶体をつくる時，x_{A},x_{B} をそれぞれ組成として，ある組成 x_{A},x_{B} での自由エネルギーは，

$$G = x_{\mathrm{A}}\mu_{\mathrm{A}} + x_{\mathrm{B}}\mu_{\mathrm{B}} \tag{2.4}$$

で求まる．また A 原子と B 原子の平衡状態（つまり A 原子への B 原子の固溶限，また B 原子への A 原子の固溶限が定まる）では，

$$\mu_{\mathrm{A}} = \mu_{\mathrm{B}} \tag{2.5}$$

が成り立つ．

2.3 状態図と組成自由エネルギー曲線

図 2.4(a) に 3C,4H の準安定平衡状態図を示した．図 2.4(b) に, ある温度 T における 3C,4H,$\mathrm{Si_{liquid}}$ のそれぞれの組成自由エネルギー曲線を示した．式 2.1 より μ はギブスの自由エネルギーの粒子微分であり，すなわち組成微分である．したがって μ は図 2.4(b) の組成自由エネルギーの接線

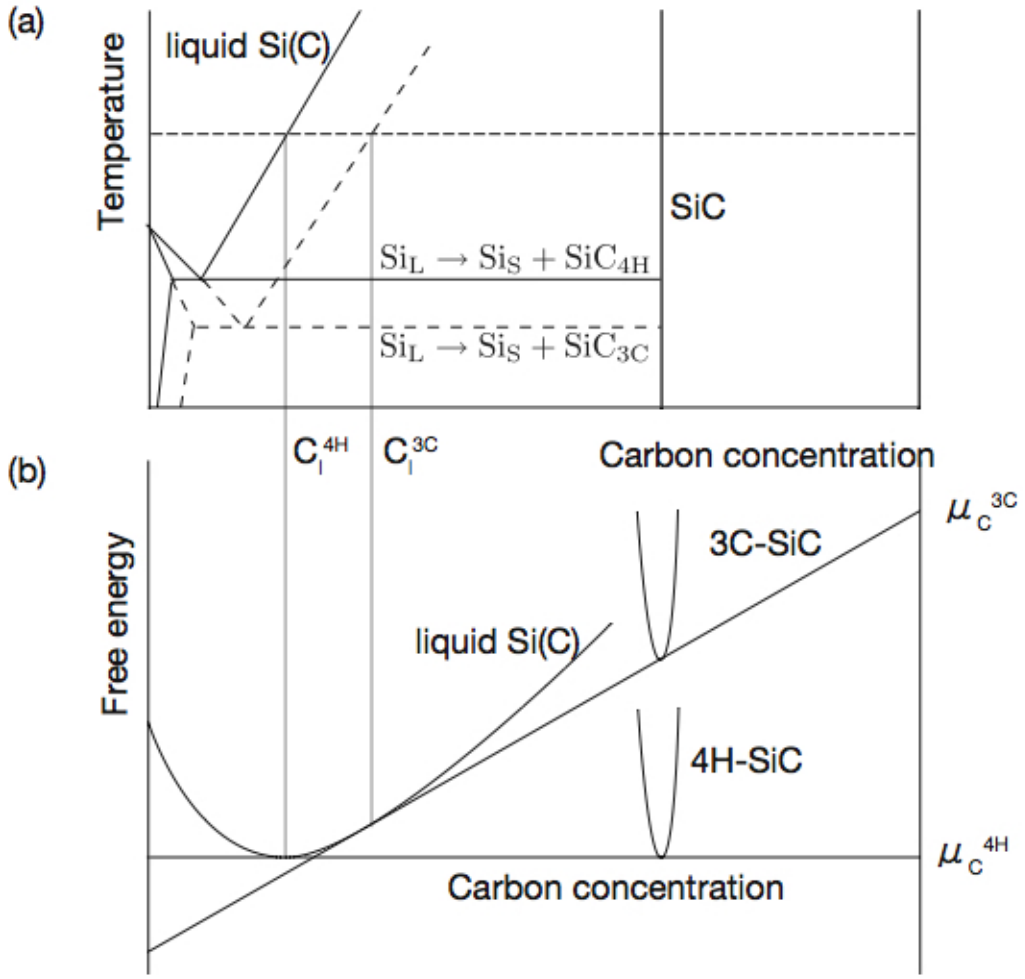

図 2.4: 状態図と組成自由エネルギー曲線.

の傾きとも考えられる. 図 2.4(b) の組成自由エネルギー曲線の接線は傾きがそれぞれ

$$\mu_{\mathrm{C}}^{\mathrm{3C}} = \mu_{\mathrm{Si_{liquid}}} \tag{2.6}$$

$$\mu_{\mathrm{C}}^{\mathrm{4H}} = \mu_{\mathrm{Si_{liquid}}} \tag{2.7}$$

となり，式 2.5 で示される平衡状態を示している．これにより 3C,4H での固溶限が決定され，また状態図も決定される．

2.4 準安定平衡状態図

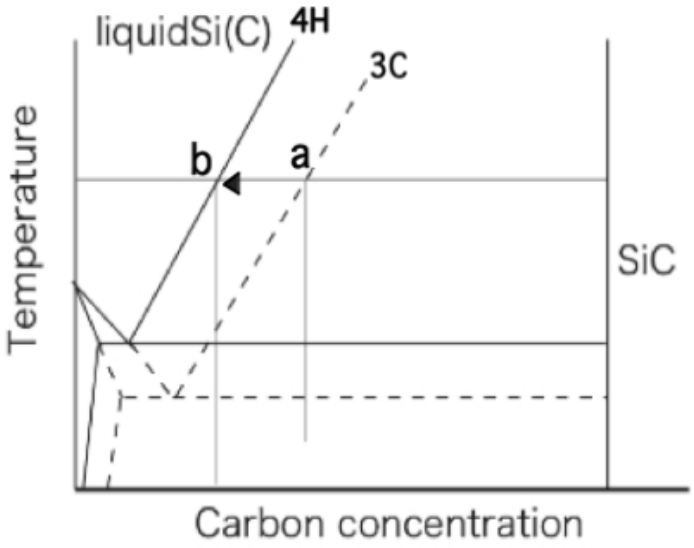

図 2.5: 準安定平衡状態図.

4H-SiC は安定，3C-SiC は 4H-SiC に比べ準安定であるので，共晶温度は必ず低くなる．それに伴って，等温で図 2.5 の a,b に示したように準安定の 3C-SiC より 4H-SiC のほうが炭素濃度が小さくなる．したがって，3C,4H それぞれの Si-C 二元系の状態図は図 2.5 のようになる．3C-SiC と 4H-SiC の共晶反応はそれぞれ破線と実線で示した．また矢印は MSE プロセスにおける炭素の移動方向を示している．

2.5 実験装置の構成

図 2.6: 実験装置の構成図 [5].

図 2.6 に 4H-SiC 単結晶成長の結晶成長構成図を示した．3C-SiC と 4H-SiC とで薄い Si を挟んだものを TaC の坩堝中に入れる．これをある一定の温度 (1700 ℃) まで加熱し放置する．すると Si が融点を超え溶け出し，溶媒となり熱的に安定な 4H-SiC のウエハーが基板 (seed)，不安定な 3C-SiC のウエハーが原料板 (feed) となり基板側に 4H-SiC の単結晶が析出すると考えられる．実験では等温過程において feed 側 (3C-SiC) は溶け出し，seed 側 (4H-SiC) に 4H-SiC 単結晶が析出した．

第3章　炭素濃度プロファイルシミュレーション

3.1　4H-SiCの単結晶成長のSEM像

図3.1に4H-SiCの単結晶成長のSEM像を示した．図3.1は成長温度で10分間保持した試料の断面のSEM像である．4H-SiCが約25 μ mエピタキシャル成長しているのが分かる．基板の4H-SiCは窒素ドープされているため元々の基板と新たに成長した層は明確に区別される．また溶媒（液体Si中）での炭素濃度に勾配がない状態が続いている．これは溶媒中の炭素濃度が低すぎるためWDX（波長分散X線）による測定限界を超えており精確に測定できていないと考えられる．そこで我々はコンピュータシミュレーションによりこの溶媒中の炭素濃度プロファイルを推測する事を試みた．

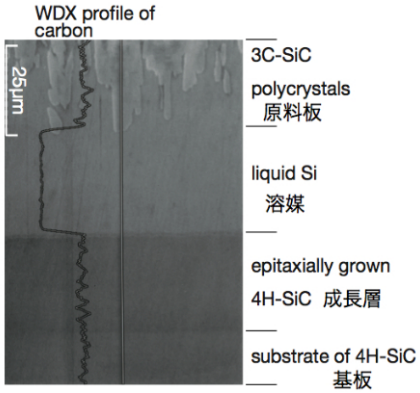

図 3.1: 4H-SiCの結晶成長のSEM像と波長分散X線（WDX）による炭素濃度プロファイル[5].

3.2　シミュレーション手法 (マクロ)

3.2.1　拡散方程式

駆動力を濃度勾配とした時，熱拡散方程式は拡散速度を D とすると，

$$\frac{\partial C}{\partial t} = \frac{1}{D}\frac{\partial^2 C}{\partial x^2} \tag{3.1}$$

とおける．

この偏微分方程式を差分方程式と Crank-Nicolson 法を用いて数値的に解いた．また，結晶の析出・溶解の実現には質量保存を用いた．

3.2.2　差分方程式

差分方程式による数値解法とは，偏微分方程式が与えられた時，その偏導関数の部分を差分でおきかえ，関数値に関する連立方程式に帰着させて，格子点における関数値を求める解法である．ここで，偏導関数の差分により近似を見る．

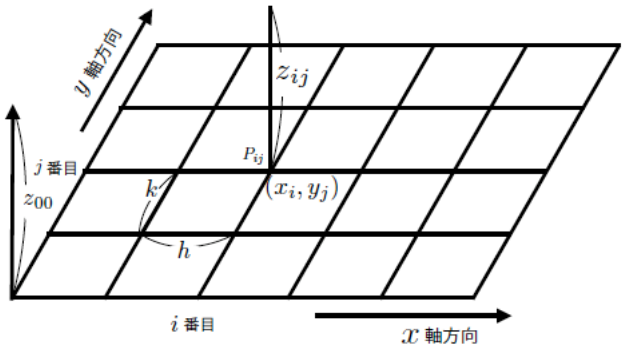

図 3.2: 格子点の模式図.

関数 $z(x, y)$，平面上の一定点 (a, b) および正数 h, k を指定して，

$x_i = a + ih, \quad y_j = b + jk \quad (i, j$ は整数$)$

点 $P_{ij} = (x_i, y_j) \quad$ (格子点と呼ぶ)

$z_{ij} = z(x_i, y_j) \quad (z_{i,j}$ と書くこともある$)$

とおく．

これらの格子点における関数値 z_{ij} によって，偏導関数の値を近似する事を考えるが，偏導関数の値といっても本質的には 1 変数関数 $f(x)$ の導関数の値を近似するのと変わりない．

h を十分小さく取ってあるとしよう．前進差分，後退差分を考えれば，

$$f'(x_i) \fallingdotseq \frac{f_{i+1} - f_i}{h}, \quad f'(x_i) \fallingdotseq \frac{f_i - f_{i-1}}{h}$$

ただし，$f_i = f(x_i)$ とする．

また，２次微分係数は前進差分で，

$$f''(x_i) \fallingdotseq \frac{f'(x_{i+1}) - f'(x_i)}{h}$$

上式の右辺に上の後退差分の近似式を代入すれば，

$$f''(x_i) \fallingdotseq \frac{\frac{f_{i+1}-f_i}{h} - \frac{f_i-f_{i-1}}{h}}{h} = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}$$

となる．これは中心差分と呼ばれる．

したがって，h, k が微小な時，格子点 P_{ij} における $z(x, y)$ の偏導関数の値は次の式で近似される．

$$\frac{\partial z}{\partial x} \fallingdotseq \frac{z_{i+1,j} - z_{ij}}{h}, \quad \frac{\partial z}{\partial y} \fallingdotseq \frac{z_{i,j+1} - z_{ij}}{k} \tag{3.2}$$

$$\frac{\partial^2 z}{\partial x^2} \fallingdotseq \frac{z_{i+1,j} - 2z_{ij} + z_{i-1,j}}{h^2} \tag{3.3}$$

$$\frac{\partial^2 z}{\partial y^2} \fallingdotseq \frac{z_{i,j+1} - 2z_{ij} + z_{i,j-1}}{k^2} \tag{3.4}$$

3.2.3 Crank-Nicolson 法の導出

(3.1) 式の熱拡散方程式 : $\frac{\partial C}{\partial t} = \frac{1}{D}\frac{\partial^2 C}{\partial x^2}$

初期条件　　$C(x, 0) = f(x)$

$$\text{境界条件} \qquad C(0,t) = g_1(t), \quad z(1,t) = g_2(t)$$

を考える．

一般に，$\frac{\partial^2 C}{\partial x^2}$ の点 P_{ij} における中心差分近似は，(3.3) 式より，

$$\frac{\partial^2 C}{\partial x^2} \fallingdotseq \frac{C_{i+1,j} - 2C_{ij} + C_{i-1,j}}{h^2}$$

である．今，点 P_{ij} および $P_{i,j+1}$ における $\frac{\partial^2 C}{\partial x^2}$ の中心差分近似の加重平均で $\frac{\partial^2 C}{\partial x^2}$ を近似してみる．重みを $1-\theta : \theta$ とすれば，

$$\frac{\partial^2 C}{\partial x^2} \fallingdotseq (1-\theta)\frac{C_{i+1,j} - 2C_{ij} + C_{i-1,j}}{h^2} + \theta\frac{C_{i+1,j} - 2C_{ij} + C_{i-1,j}}{h^2}$$

これと (3.2) 式の第 2 式を偏微分方程式に代入すれば，

$$\frac{C_{i,j+1} - Cij}{k} \fallingdotseq \frac{1}{D}\left\{(1-\theta)\frac{C_{i+1,j} - 2C_{ij} + C_{i-1,j}}{h^2} + \theta\frac{C_{i+1,j} - 2C_{ij} + C_{i-1,j}}{h^2}\right\}$$

$r = \frac{ck}{h^2}$ とおき，添え字が $j+1$ のものを左辺に，j のものを右辺にまとめると，

$$\begin{aligned} &-r\theta C_{i-1,j+1} + (1+2r\theta)C_{i,j+1} - r\theta C_{i+1,j+1} \\ &\qquad = r(1-\theta)C_{i-1,j} + \{1-2r(1-\theta)\}C_{ij} + r(1-\theta)C_{i+1,j} \end{aligned}$$

ここで，$\theta = \frac{1}{2}$ とすれば，次の Crank-Nicolson の公式が得られる．

$$C_{i-1,j+1} - 2\left(1+\frac{1}{r}\right)C_{i,j+1} + C_{i+1,j+1} = -C_{i-1,j} + 2\left(1-\frac{1}{r}\right)C_{i,j} - C_{i+1,j}$$

$$(i = 1, 2, \ldots, n-1 : j = 0, 1, \ldots, m) \tag{3.5}$$

ここに，$C_{i,0}(i = 1, 2, \ldots, n)$ は初期条件より，$C_{0,j}, C_{n,j}(j = 1, 2, \ldots, m)$ は境界条件より定まる既知数である．

3.2.4 Crank-Nicolson 法の実践

以下，Crank-Nicolson 法について考えてみる．初期条件及び (3.5) 式で $j = 0$ とおくことにより，C_{ij} $(i = 0, 1, 2, \ldots, n)$ に関する連立方程式ができる．これを解いて，$C_{i,1}$ $(i = 1, 2, \ldots, n-1)$ が得られる．

一般に，$j=k$ での値 $C_{1,k}, C_{2,k}, \cdots, C_{n-1,k}$ が求まれば，$j=k+1$ のときの値 $C_{1,k+1}, C_{2,k+1}, \cdots, C_{n-1,k+1}$ が求められる．実際，

$$p = 2\left(1+\frac{1}{r}\right), \quad q = 2\left(1-\frac{1}{r}\right) \tag{3.6}$$

とおき，$j=k$ として，$i=1,2,\cdots,n-1$ を順次与えて得られる次の連立方程式を解けばよい．

$$\begin{cases} C_{0,k+1} & = g_1(t_{k+1}) \\ C_{0,k+1} - pC_{1,k+1} + C_{2,k+1} & = -C_{0,k} + qC_{1,k} - C_{2,k} \\ \quad C_{1,k+1} - pC_{2,k+1} + C_{3,k+1} & = -C_{1,k} + qC_{2,k} - C_{3,k} \\ \cdots\cdots\cdots\cdots\cdots\cdots & \cdots\cdots\cdots\cdots\cdots\cdots \\ \quad C_{n-2,k+1} - pC_{n-1,k+1} + C_{n,k+1} & = -C_{n-2,k} + qC_{n-1,k} - C_{n,k} \\ C_{n,k+1} & = g_2(t_{k+1}) \end{cases}$$

これを行列を用いて書き表すと次のようになる．

$$\begin{bmatrix} -p & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -p & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -p & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -p & \cdots & 0 & 0 \\ & & & \cdots\cdots\cdots\cdots & & & \\ 0 & 0 & 0 & 0 & \cdots & -p & 1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & -p \end{bmatrix} \begin{bmatrix} C_{1,k+1} \\ C_{2,k+1} \\ C_{3,k+1} \\ C_{4,k+1} \\ \cdots\cdots \\ C_{n-2,k+1} \\ C_{n-1,k+1} \end{bmatrix}$$

$$= \begin{bmatrix} -C_{0,k} + qC_{1,k} - C_{2,k} - g_1(t_{k+1}) \\ -C_{1,k} + qC_{2,k} - C_{3,k} \\ -C_{2,k} + qC_{3,k} - C_{4,k} \\ -C_{3,k} + qC_{4,k} - C_{5,k} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ -C_{n-3,k} + qC_{n-2,k} - C_{n-1,k} \\ -C_{n-2,k} + qC_{n-1,k} - C_{n,k} - g_2(t_{k+1}) \end{bmatrix}$$

この連立方程式を解けば，$C_{i,k+1}$ $(i=1,2,\ldots,n-1)$ が得られる．

3.2.5 質量保存

MSE における溶質の析出・溶解を視覚化した．単純に計算される空間を閉じた系として，溶質をやりとりさせた．溶質のプロファイル変化に

伴い，溶質の供給，消滅が起こる．ここでは，seed 及び feed 界面の移動により，物質保存が成り立つとして求めた．

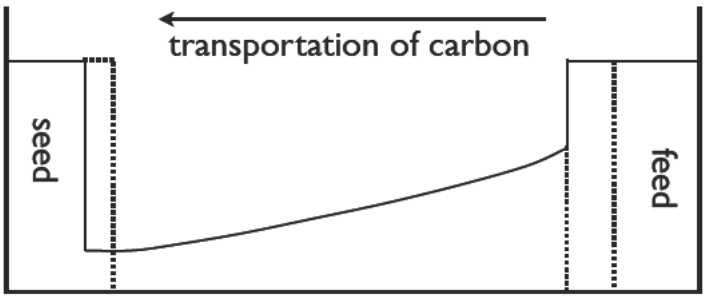

図 3.3: 質量保存の模式図.

3.3　シミュレーション結果（マクロ）

3.3.1　拡散方程式からの炭素濃度プロファイル

図 3.4 に拡散方程式からの溶媒中の炭素濃度プロファイルを示した．図 3.4 は時系列で t=0,t=400,t=600 で濃度プロファイルを記述している．t=0 を初期状態とし，t=400 を遷移状態，t=6000 を定常状態としている．マクロ（拡散方程式）からの溶媒中の炭素濃度プロファイルが導けた．

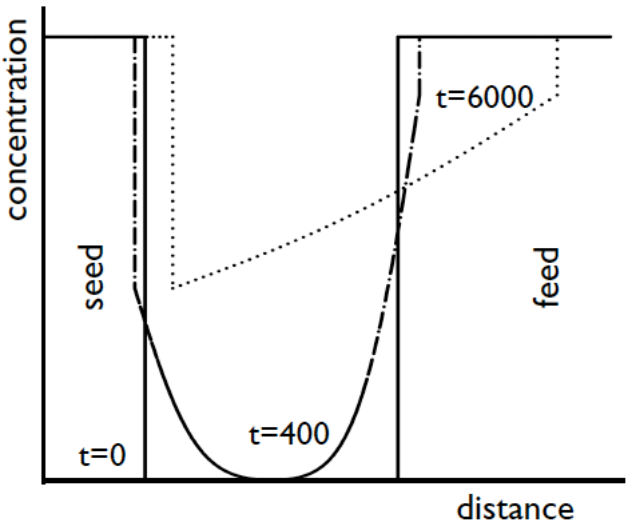

図 3.4: 濃度プロファイルの計算結果.

3.4　シミュレーション手法（原子レベル）

3.4.1　準安定平衡状態図

図 3.5 に 3C-SiC と 4H-SiC の準安定平衡状態図を示した．点 a は 3C-SiC の溶解度を示してあり，点 b は 4H-SiC の溶解度を示している．この準安定平衡状態図（図 3.5）から原子レベルでの MSE を考えていく．a 点は b 点に比べ等温で溶解度が高い．a 点，b 点それぞれが固液界面においてそれぞれの溶解度で局所平衡状態となり，定常状態において炭素濃度は 4H 界面では常に低く，3C 界面では常に低い．したがって溶媒中では a 点側

で高く，b 点側で低くなるように濃度勾配ができる．この濃度勾配を保ったまま結晶成長が進む．MSE における駆動力は 3C の固液界面と 4H の固液界面での化学ポテンシャル差である．

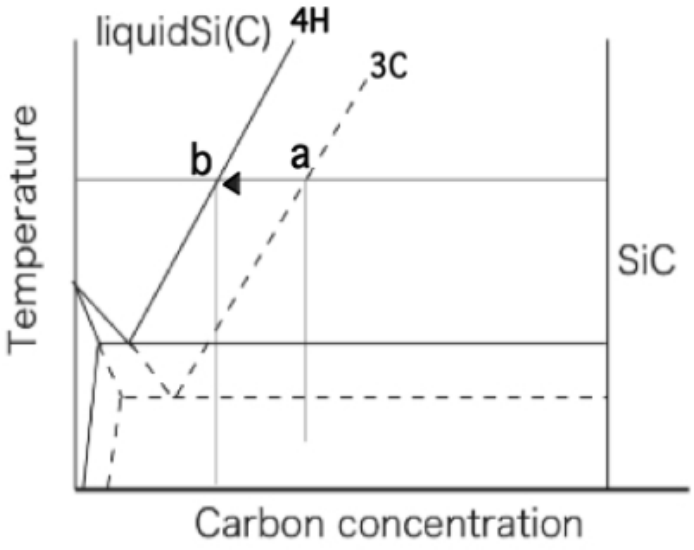

図 3.5: 準安定平衡状態図.

3.4.2 MSE プロセスにおける素過程

素過程とは素構造（elementary structure; それ以上分解できない基本単位）によって構成される構造の挙動のことである．図 3.6 に MSE における素過程を示した．MSE プロセスにおいて素過程となるのは「拡散」,「吸着」,「放出」である．

3.4.3 放出と拡散

図 3.7 に固液界面における熱振動と活性化エネルギーの図を示した．固体からの原子の放出の際，原子は図 3.7 に示された活性化エネルギーの山を越えてなければいけない．これは原子の熱振動によって生じる．拡散により，液体側にエネルギーのくぼみが頻繁に生じても，原子は活性化エ

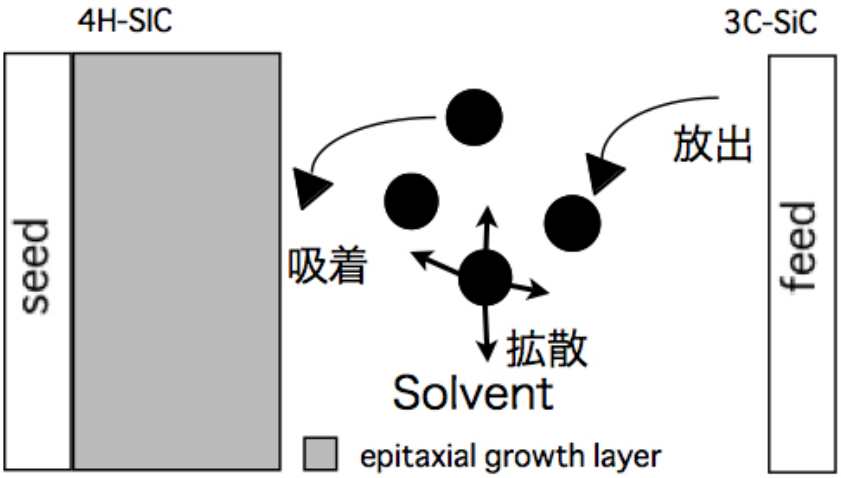

図 3.6: 素過程.

ネルギーを超えなければ液体状態に行く事は出来ない．したがって「放出」と「拡散」は独立して考える事が出来る．

3.4.4 結晶表面の構造

図 3.8 に結晶表面の構造を示した．結晶表面は図 3.8 に示されているようにテラス，ステップ，キンク，空孔に大別する事が出来る．結晶表面のテラスについた原子はテラスで拡散しながら，徐々にステップの方へと近づいて行く．その後ステップからキンクへたどり着き，結合し析出する．

3.4.5 格子モデル

図 3.9 にシミュレーションに採用した格子モデルを示した．結晶成長の素過程である「拡散」「吸着」「放出」の３つの過程をプログラムに組み込み溶媒中の炭素濃度プロファイル変化を観察した．プログラムでは溶媒原子を０とし，溶質原子を１としている．また基板，原料板を全て１とした理想化した状態でシミュレーションを行った．

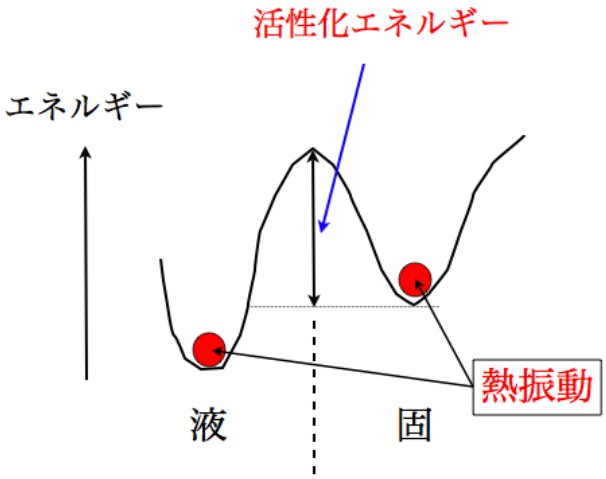

図 3.7: 熱振動と活性化エネルギー

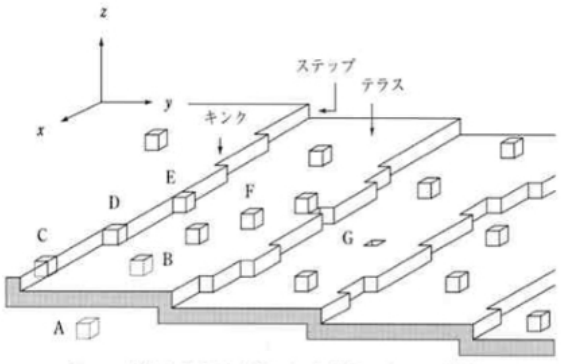

図 3.8: 結晶表面の構造 [3].

図 3.9: シミュレーションに採用した格子モデル.

3.5 シミュレーション結果（原子レベル）

3.5.1 素過程からの溶媒中の炭素濃度プロファイル

図 3.10, 図 3.11, 図 3.12, 図 3.13 に濃度勾配が定常状態になる過程を時系列で示した．図 3.11 に示されているように原子は feed,seed の両サイドから溶け出す．溶け出した原子は図 3.12 のように溶媒中に増え続ける．その後図 3.13 のように濃度プロファイルは直線となり，定常状態になる．

図 3.10, 図 3.11, 図 3.12, 図 3.13 に示されているプロセスの固溶限は feed:200,seed:60 である．

このように素過程からの溶媒中の炭素濃度プロファイル変化のシミュレーションより基板と原料側に溶解度差をつけると溶媒中に濃度勾配が生じた．

3.5.2 溶媒中の炭素濃度勾配

図 3.14 に溶媒中の炭素濃度勾配の拡大画像を示した．図 3.14 を見ると勾配が直線になっているのが分かる．これよりプロセスが定常状態になると濃度勾配は直線に近づくことが分かった．

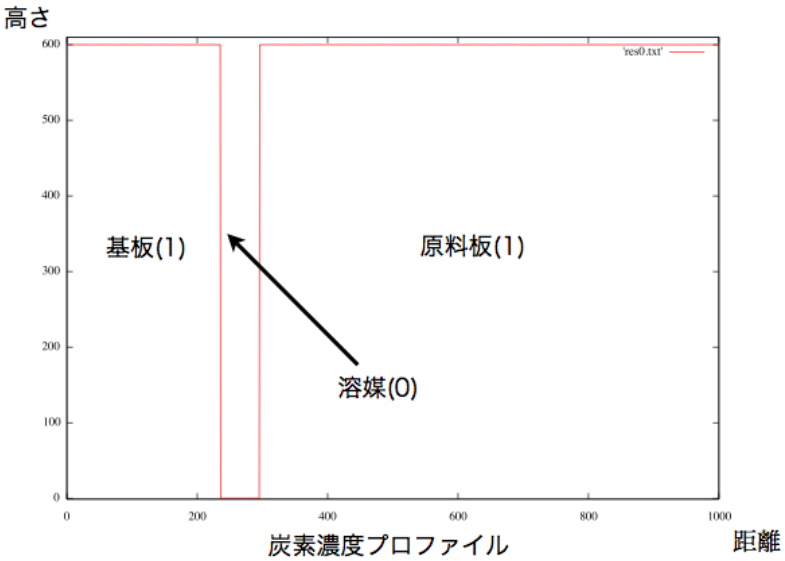

図 3.10: 炭素濃度プロファイル（初期）.

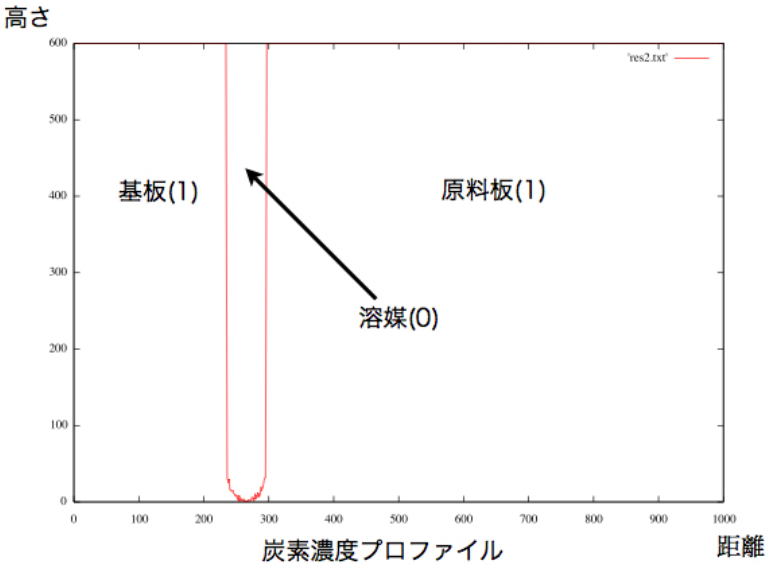

図 3.11: 炭素濃度プロファイル（遷移１）.

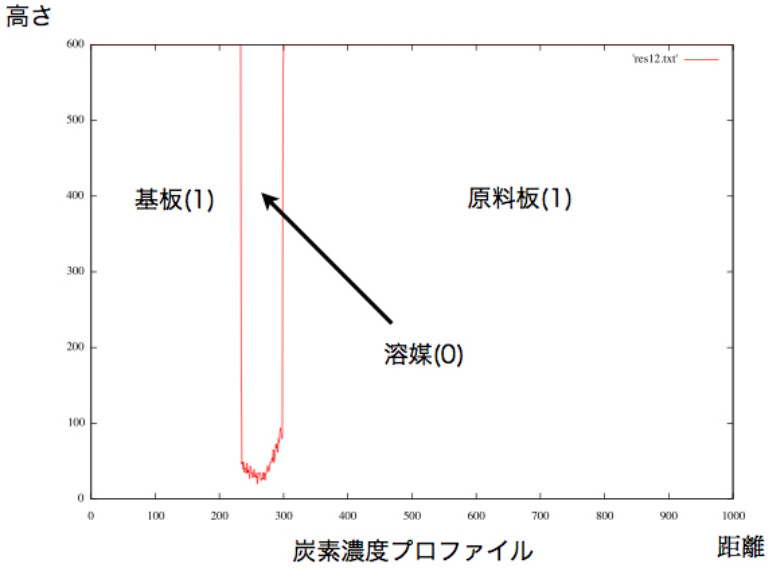

図 3.12: 炭素濃度プロファイル（遷移２）.

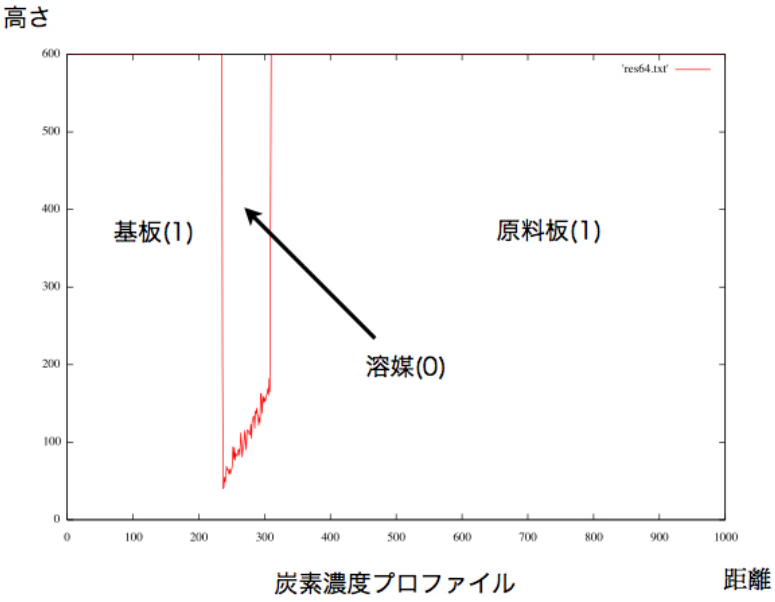

図 3.13: 炭素濃度プロファイル（定常状態）.

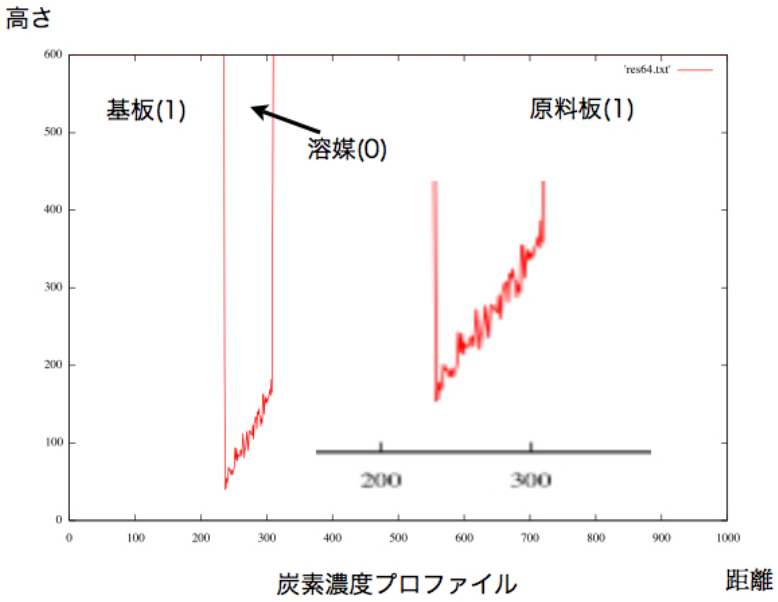

図 3.14: 炭素濃度プロファイルとその拡大画像.

3.5.3 濃度勾配と固溶限

図 3.15 に濃度勾配と固溶限を示した．原料板側の固液界面で炭素濃度が固溶限より低い所で定常状態になっていた．当初の予想では固溶限同士を結んだ形に濃度勾配になると予想していたが, 濃度勾配は基板，原料板での固溶限によるとは限らないという結果になった．

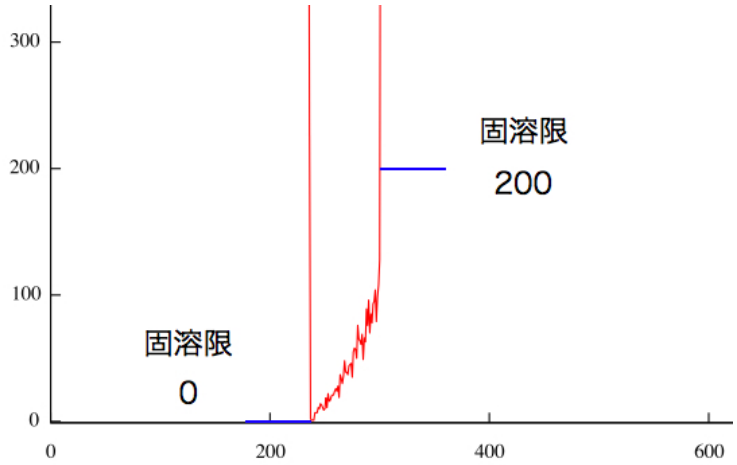

図 3.15: 濃度勾配と固溶限．

3.5.4 異なる溶媒幅での成長速度

図 3.16 に実験で得られた異なる溶媒幅での成長速度の実験結果 (per 1/2 hour) を示した．実験結果では溶媒幅が小さくなると，直線的に成長速度も増す事が示されている．また温度が上がると成長速度が増すことも，この図 3.16 より読み取れる．

図 3.17 に拡散が律速した場合の成長速度のシミュレーション結果を示した．シミュレーション結果も実験結果同様，溶媒幅が小さくなると直線的に成長速度も増す．また基板側での炭素原子の放出頻度を上げると成長速度は増加した．成長速度は直線的な増加関数を示しており，実験結果と近い物となった．

次に図 3.18 に界面反応が律速した時の成長速度のシミュレーション結果を示した．界面反応律速の場合，溶媒幅が広い時は直線的な成長を示

すが溶媒幅が狭くなるにつれ頭打ちしたような成長を示し，全体的に少し，パラボラな増加関数を示した．

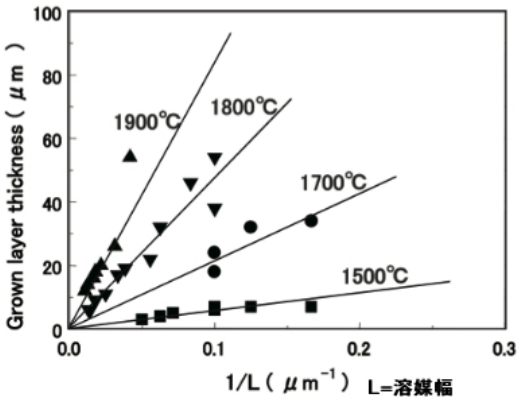

図 3.16: 異なる溶媒幅での成長速度の実験結果 (per 1/2 hour)．

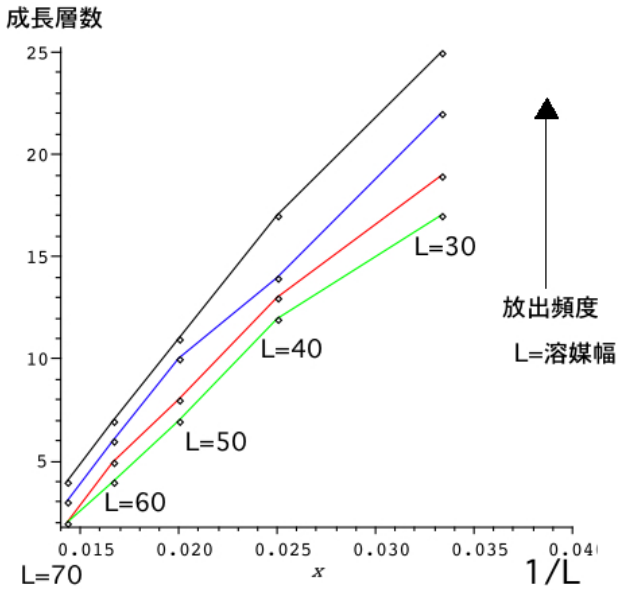

図 3.17: 異なる溶媒幅での成長速度のシミュレーション結果 (拡散律速).

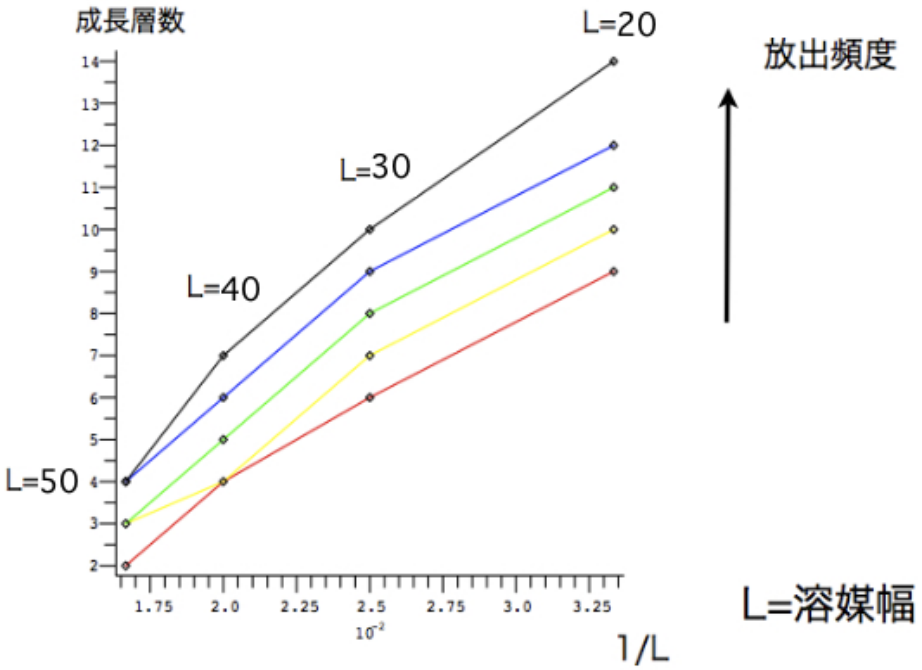

図 3.18: 異なる溶媒幅での成長速度のシミュレーション結果 (界面反応（原料板側での放出）律速).

3.6　シミュレーション結果からの考察

3.6.1　原子レベルシミュレーションでマクロでの結果を再現

原子レベルで求めた濃度プロファイルとマクロなスケールで求めた濃度プロファイルとが一致を示した．遷移過程，定常状態とほぼ同じような濃度プロファイルを辿り，基板側で成長が確認出来た．

3.6.2　定常状態での溶媒中の濃度勾配

定常状態での溶媒中の濃度勾配は直線になる．溶媒中の拡散にはベクトルを持たせていない．全ての方向に等確率で原子は進む．その結果，溶媒中の炭素濃度勾配は定常状態においてすべて直線を示した．

3.6.3　濃度勾配と固溶限

濃度勾配は固溶限差によるとは限らないという結果が出た．これは溶媒量が常に一定に保たれているためと考えられる．実験においても温度一定であるため，液体 Si の溶解度は変わらない．そのため実際の結晶成長でも溶媒量は一定に保たれているはずである．これより，理論上は実際の系でも濃度勾配は固溶限差によるとは限らないといえる．しかし実際は基板側の固溶限は 1

3.6.4　原子レベルでの MSE の律速過程

図 3.19 に MSE における律速過程を示した．図 3.16 の成長速度の実験結果では成長速度は直線的な増加関数を示している．図 3.17 より溶媒中の「拡散」が結晶成長過程を律速している場合，結晶成長の成長速度は直線的な増加関数を示す．このことより MSE における律速過程は図 3.19 に示されているように溶媒中の「拡散」であるという知見が得られた．

図 3.19: MSE における律速過程

3.6.5 表面拡散，ステップ拡散，テラスへの吸着，溶媒拡散と律速過程

溶媒中の拡散が全体のプロセスを律速していることがシミュレーションより明確になったが，では何故テラス上の拡散（表面拡散），ステップでの拡散は速いのであろう．図 3.20 に固体結晶表面の KosselModel を示した．結晶表面において，溶質原子は，図 3.20 のテラス上に溶媒から付着し，溶質はテラス状の拡散によりステップに向かい，スッテプでの拡散によりキンクに向かうと考えられる．溶媒中の拡散が律速してくるという事は，このテラス，スッテプでの拡散が非常に速い事を意味する．結晶表面の構造をより詳しく見る必要がある．

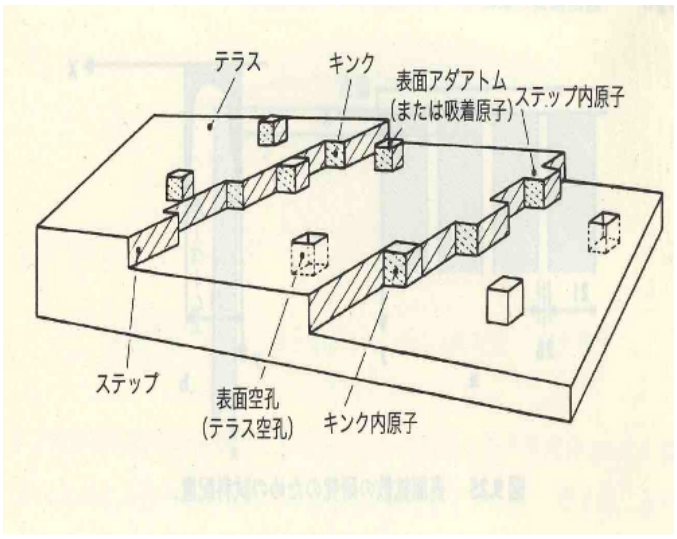

図 3.20: 固体結晶表面の KosselModel[7].

3.6.6　一次元，二次元，三次元の核生成

液体シリコンに溶解する炭素原子は非常に少ない．そのため溶媒中（三次元）で炭素原子が核生成する可能性は極めて低い．また二次元（テラス上）も同様である．しかしステップ（一次元）では炭素原子が集中的に集まる．そのためステップでの核生成の可能性はあるといえる．しかし（実験事実として）実際には核生成は起こっていない．

3.6.7　実験データ，シミュレーションデータと図 3.21 との比較

実験で観測されている結晶表面構造は図 3.23 のような stepped 構造を示している．本来結晶表面が stepped 構造を示すならば成長速度は図 3.21 に示されるスパイラル成長を示すはずである．しかし図 3.16 で示されたデータは直線を示している．これらの矛盾の説明として以下が考えられる．

1. **表面拡散が異常に速いため付着成長と同じ成長速度を示す．**
2. **図 3.16（実験データ）は図 3.21 に示されたスパイラル成長の局所的な所だけ見ている．**
3. **結晶表面構造と成長速度の理論に新たな解釈が生まれる可能性がある．**

仮説 2,3 に対しては，これを検証するシミュレーションあるいは実験計画を立てるには至っていない．律速過程と拡散，吸着，一次元の核生成，それらを熱統計力学的手法だけで解明するの現時点では非常に難しい．今後の研究の進展が期待される．本研究では仮説 1 に対して結晶表面での原子位置の第一原理計算により 0001 面の Si 面，Si 修飾した C 面について表面拡散の検証を行った．また MAYA で原子レベルに結晶構造を表示しすることにより幾何学的にこれらの疑問にアプローチした．

3.6.8　付着成長

図 3.21 の付着成長では図 3.22 のような kinked 構造をとると考えられている．これにより結晶表面での拡散が生じずキンクサイトへ直接，溶媒から原子が付着するため，付着成長では溶媒中の拡散が律速し，成長速度は直線的になる．

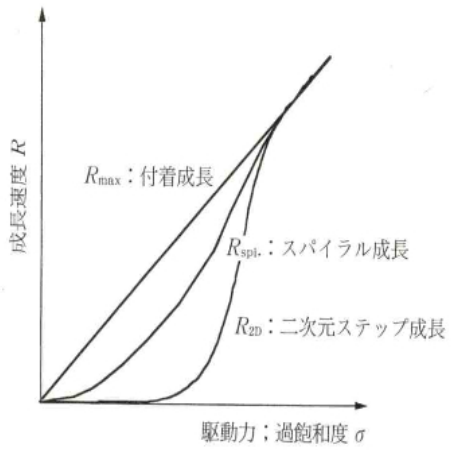

図 3.21: 成長速度の飽和度依存性 [6].

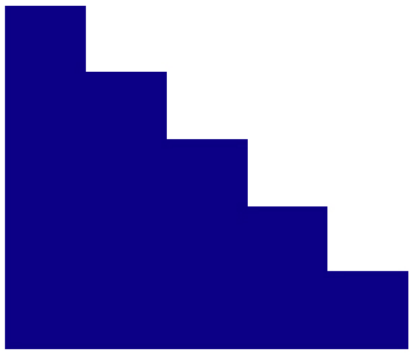

図 3.22: 結晶表面の kinked 構造.

3.6.9　スパイラル成長

図 3.21 のスパイラル成長で結晶表面は図 3.23 のような構造をとると考えられる.

図 3.23: 結晶表面の stepped 構造.

第4章　結晶表面での原子位置の第一原理計算

4.1　第一原理計算

4.1.1　平面波基底擬ポテンシャル法

密度汎関数に基づく第一原理擬ポテンシャル法によるバンド計算VASP code を用いて計算を行う．この手法では，3 次元周期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．交換相関ポテンシャルはGGA(generalized gradient approximation) を用いた．そして，擬ポテンシャルとしてPAW(projector augmented wave) を用いた．

4.1.2　VASP(Vienna Ab-initio Simulation Package)

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．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．

4.1.3 PAW(Projector Augmented Wave)法

本論では，VASP を使用するに至り，擬ポテンシャル法として PAW 法を用いた.. 擬ポテンシャル法には，フルポテンシャル・PAW ポテンシャル・(ウルトラソフト型) 擬ポテンシャルの 3 つに分類される．PAW ポテンシャルは Blochl が考案した全電子計算の方法で，フルポテンシャルの精度と擬ポテンシャルの高速性の両者を兼ね備えた方法で，それぞれの特徴を表 4.1 に示す.

表 4.1: 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 法	○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)	○計算時間を軽減 ×アルカリ金属，アルカリ土類，希土類に難

4.2 計算手法

4.2.1 真空-固体，液体-固体の第一原理計算

今回 MedeA で計算した結果は真空-固体の第一原理計算である．しかし実際は液体（溶媒シリコン）-固体（4H-SiC）である．そのため実際の系とは少しずれている.

4.2.2 手法

図 4.1 に表面エネルギー計算を行った結晶構造を示した．黄色のボールを Si 原子, 黒のボールを C 原子としている．図 4.2 は図 4.1 の上部である Si 面を表し，図 4.3 は図 4.1 の下部である Si を修飾した C 面を表してい

る．これらの面での表面拡散，付着原子位置および原子の付着する面の安定性を解明するため図 4.2 の 1〜3, 図 4.3 の 1〜3 の計 6 点について MedeA でエネルギー計算を行った．

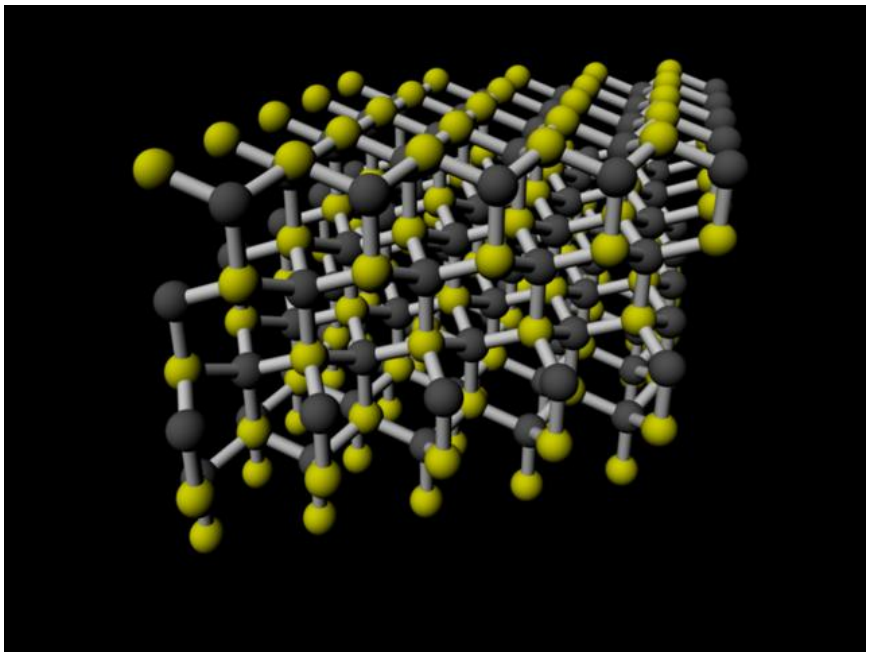

図 4.1: 表面計算をした結晶の結晶構造．

図 4.2 の面から三点取り計算を行った．点 1 ではすぐ下に炭素原子がなく空洞になっている．点 2 ではすぐ下に炭素原子が存在している．点 3 は炭素原子が結合をする場所である．

図 4.3 の面から三点取り計算を行った．点 1 ではすぐ下に C 原子がなく空洞になっている．点 2 ではすぐ下に炭素原子が存在している．点 2 は炭素原子が結合をする場所である．

4.3 計算結果

表 4.2, 表 4.3 の最安定位置は図 4.2，図 4.3 での Si 面および Si 修飾した C 面をそれぞれ基準（0 として）としている．また黄色のボールを Si 原子, 黒のボールを C 原子としている．

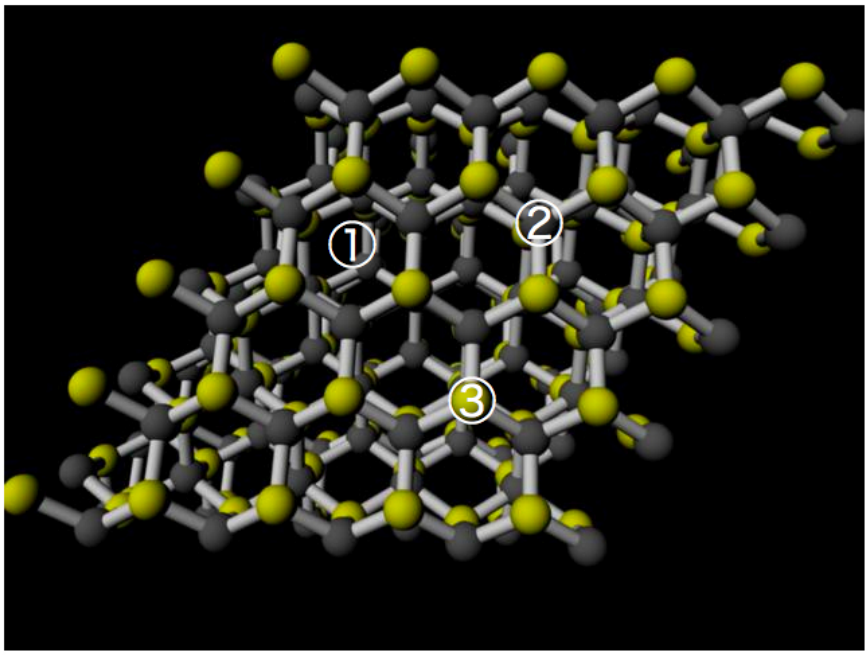

図 4.2: 図 4.1 を上部から見た結晶構造.

表 4.2: 図 4.2 の 1,2,3 で計算した最安定エネルギーと最安定位置.

場所	最安定エネルギー (eV)	最安定位置（Å）
1	−7.821877	0.540187500
2	−8.747969	0.540187500
3	−5.792461	0.9421875

表 4.3: 図 4.3 の 1,2,3 で計算した最安定エネルギーと最安定位置.

場所	最安定エネルギー (eV)	最安定位置（Å）
1	−8.008408	0.212683125
2	−8.019229	0.313183125
3	−3.944570	0.916183125

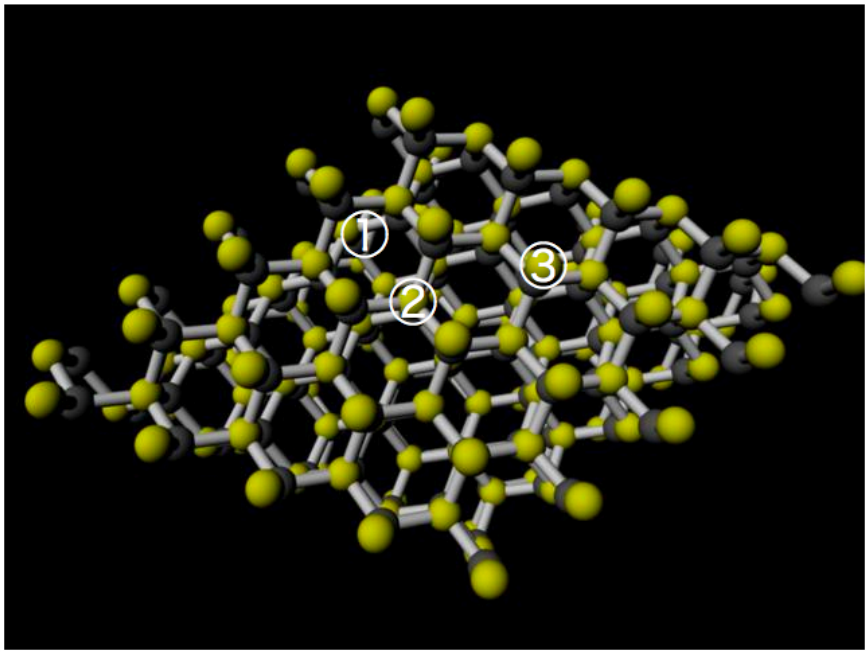

図 4.3: 図 4.1 を下部から見た結晶構造.

4.4　計算結果の考察

4.4.1　Si 面と C 面

図 4.4 に Si 面と C 面の定義を示した．図 4.4 に示されているように Si 面，C 面とは結合ボンドが 1 つの Si-C 結合を切断した時に注目する方向の違いである．Si 面，C 面とは結合ボンドを切断した時，Si 原子面が表面となる方向を向いた面が Si 面であり，逆に C 原子が表面となる方向を向いた面が C 面となる．

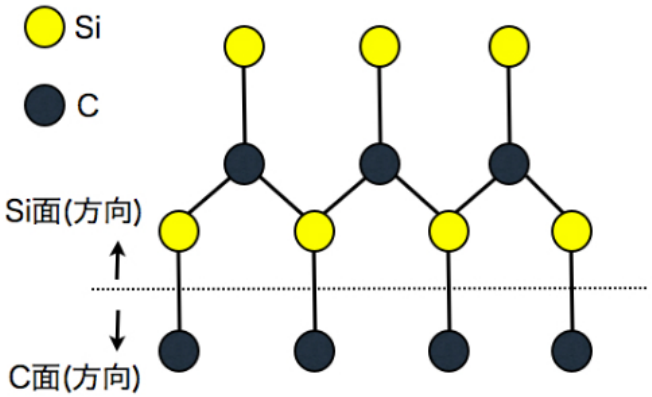

図 4.4: Si 面と C 面の模式図.

4.4.2　考察

図 4.5，図 4.6 に Si 面,Si 修飾をした C 面に C 原子を吸着させた時のエネルギー準位の模式図を示した．1,2,3 は図 4.2 に示した吸着サイトをそれぞれ示している．図 4.5，図 4.6 より C 原子の拡散には経路がいくつかある事が分かる．それら C 原子の拡散経路の模式図を図 4.7 に示した．図 4.7 より拡散には赤色の矢印と青色の矢印の 2 つの経路が考えられる．このうち，実際に原子がとる経路はもっとも活性化エネルギー，つまりエネルギーの山が低い経路をとると考えられる．SiC の表面においては，図 4.5，図 4.6 より，C 面を Si 修飾した面での経路つまり図 4.7 の赤色の矢印で示された経路がもっとも低い活性化エネルギーを示している．従って表面拡散は，Si 面よりも C 面上の方が速いと考えられる．またエネル

ギー差が 0.01ev と非常に小さいことからその表面拡散は異常に速いと考えられる．

また Si 面である図 4.2 の場所 2 が最安定位置である事から，C 面よりも Si 面に C 原子が付着する方が安定であることが分かった．

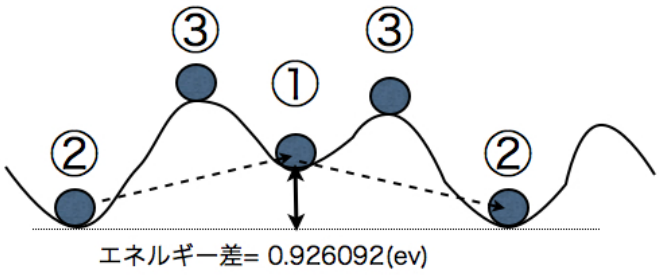

図 4.5: Si 面に C 原子を吸着させた時のエネルギー準位を示した模式図．1,2,3 は 4.2 に示した吸着サイトをそれぞれ示している．

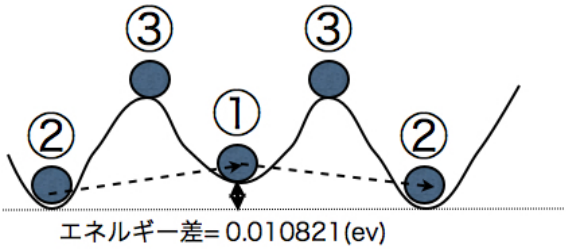

図 4.6: Si 修飾をした C 面に C 原子を吸着させた時のエネルギー準位を示した模式図．1,2,3 は 4.3 に示した吸着サイトをそれぞれ示している．

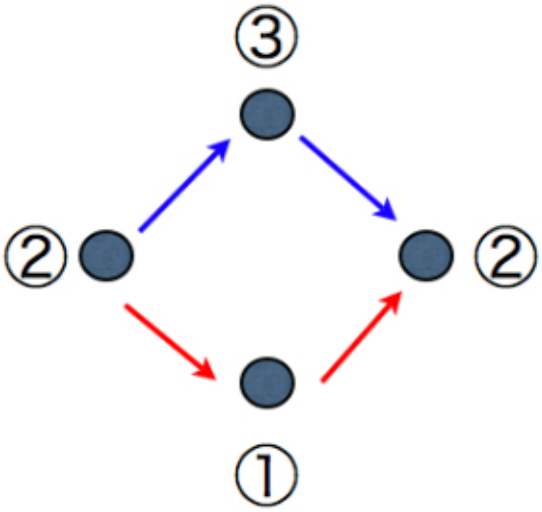

図 4.7: C 原子の表面拡散の模式図．1,2,3 はそれぞれ計算した原子のサイトを表している．

第5章　MAYAによるSiC結晶の表示

5.1　SiC結晶構造，結晶表面

5.1.1　3C,4H,6H

SiC にはその積層周期の違いから 200 種類もの結晶多形が存在する．その中でも主なものは六方晶の 4H,6H 及び立方晶の 3C の 3 種類である．4H は主にパワーデバイス用として，6H は主に青色発光素子などの GaN 用基板として使われている．4H,6H の製造には現在，主に昇華法が用いられている．3C は Si 基板上に成長させることができること，製造プロセスが Si 並みに低温化できる可能性があり，材料コストの劇的な低減と既存の Si パワーデバイス製造ラインの活用という点で非常におおきな可能性を秘めている．図 5.1 に 3C,4H,6H の積層周期を示した.

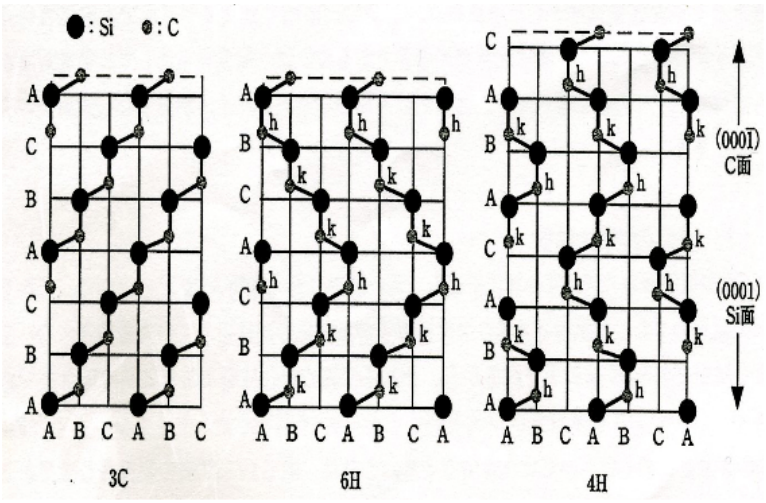

図 5.1: 3c,4h,6h の結晶構造 ((11-20) 面から見た) 模式図.

5.1.2　結晶表面（KosselModel）

図 5.2 に結晶表面の KosselModel を示した.

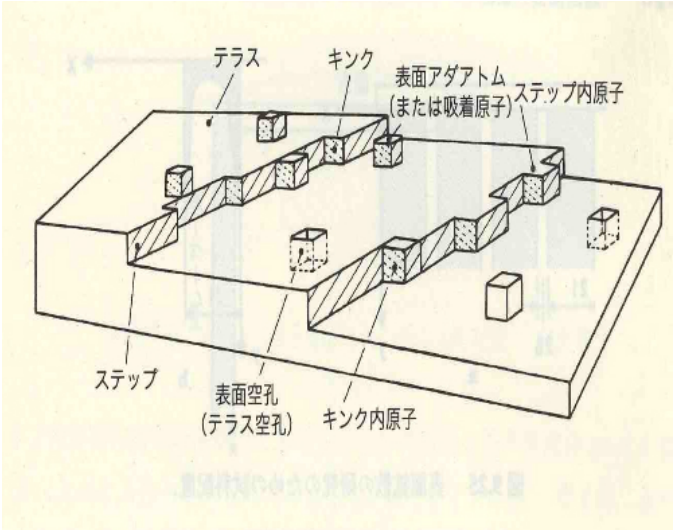

図 5.2: 固体結晶の表面結晶の KosselModel[7].

5.1.3　螺旋転位

転位とは線状の欠陥のことであり，材料の変形，特性をつかさどる重要な概念である.

図 5.3, 図 5.4, 図 5.5 に螺旋転位の概略図を示した．螺旋転位では，図 5.3 のように層がずれる．それらずれの部分を詳しく見ているのが図 5.4 である．図 5.4 の F 点の部分に screw がある．図 5.5 より層が少しずつずれていき，最終的に 1 層ずれている事が分かる．螺旋転位はこのようにして連続体として少しずつ層がずれていき，最終的に層が 1 つずれて生じる．また図 5.6 にスパイラル成長のしくみを示した．図 5.6（a）のようにずれた層の部分に原子が付着する．このとき常に C 点の screw の近くに原子が吸い込まれて行く．このため図 5.6(a) → (b) → (c) → (d) のようになる．このようにして螺旋転位，スパイラル成長が生じる.

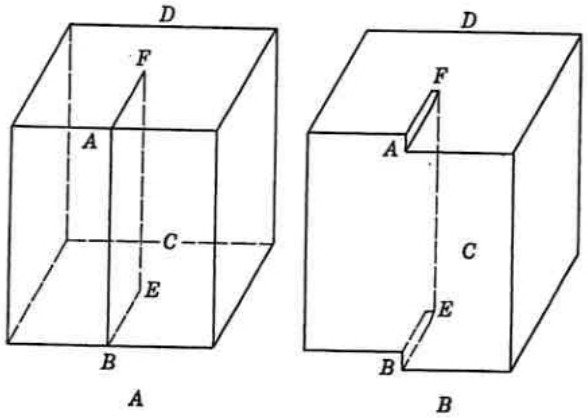

図 5.3: 螺旋転位の概略図 (a)[8].

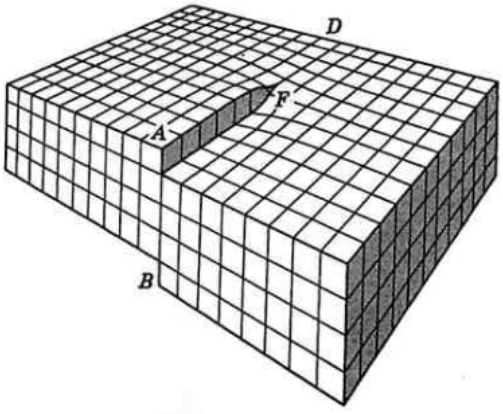

図 5.4: 螺旋転位の概略図 (b)[8].

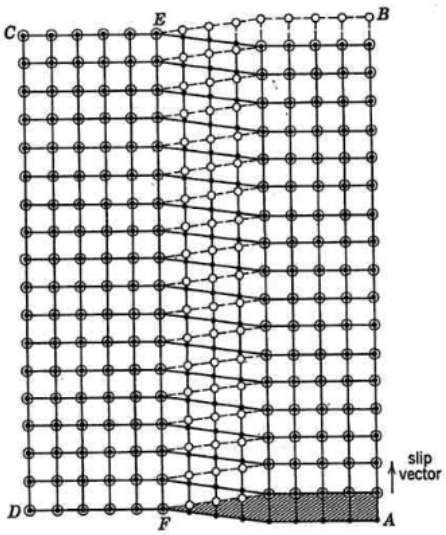

図 5.5: 螺旋転位の概略図 (c)[8].

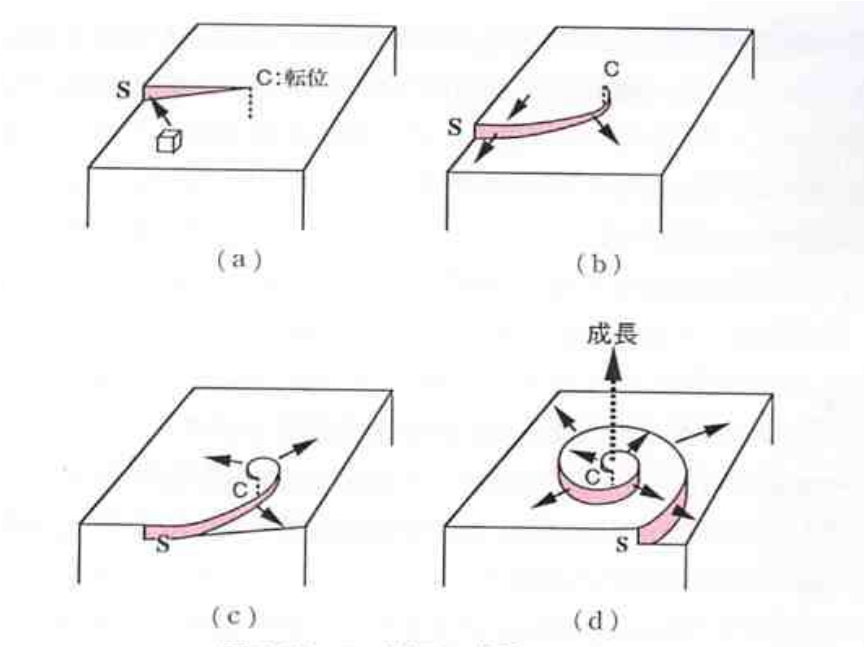

図 5.6: スパイラル成長のモデル [6].

5.2 MAYA による SiC 結晶の表示

以下では黄色のボールを Si 原子とし，黒色のボールを C 原子としている．

MSE における結晶成長では，その結晶構造が非常に重要であり，SiC の結晶構造を詳細に表示し，観察する事は非常に有意義である．下に重要な結晶多形である 3C,4H,6H を MAYA で表示した．

5.2.1 3C-SiC

図 5.7 に炭素原子が 3 周期で立方晶をなす 3C-SiC の結晶構造を示す．赤線で囲まれているのが 3C-SiC のユニットセルである．

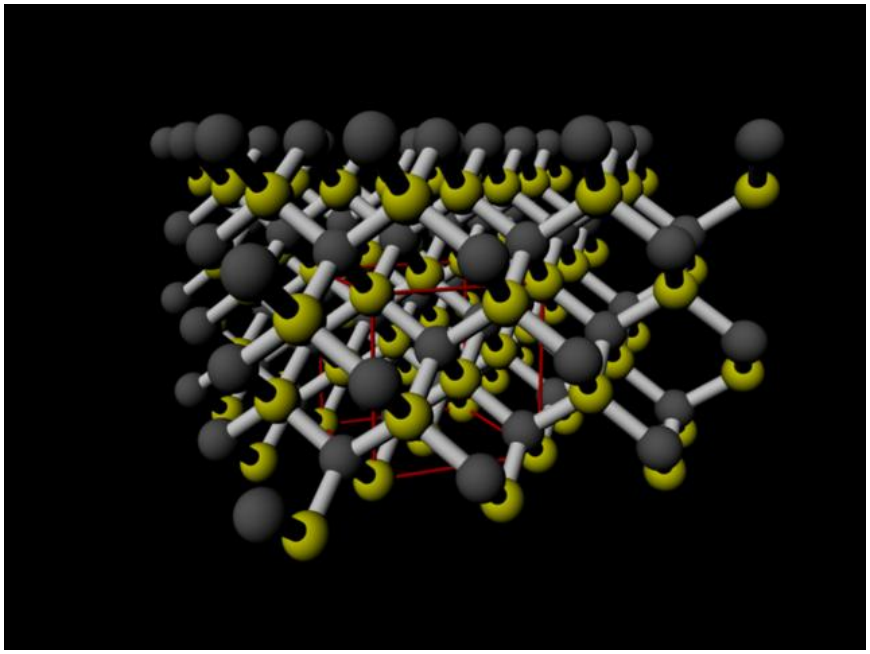

図 5.7: 3C の結晶構造．

5.2.2 4H-SiC

図 5.8 に炭素原子が 4 周期で六方晶をなす 4H-SiC の結晶構造を示す．赤線で囲まれているのが 4H-SiC のユニットセルである．

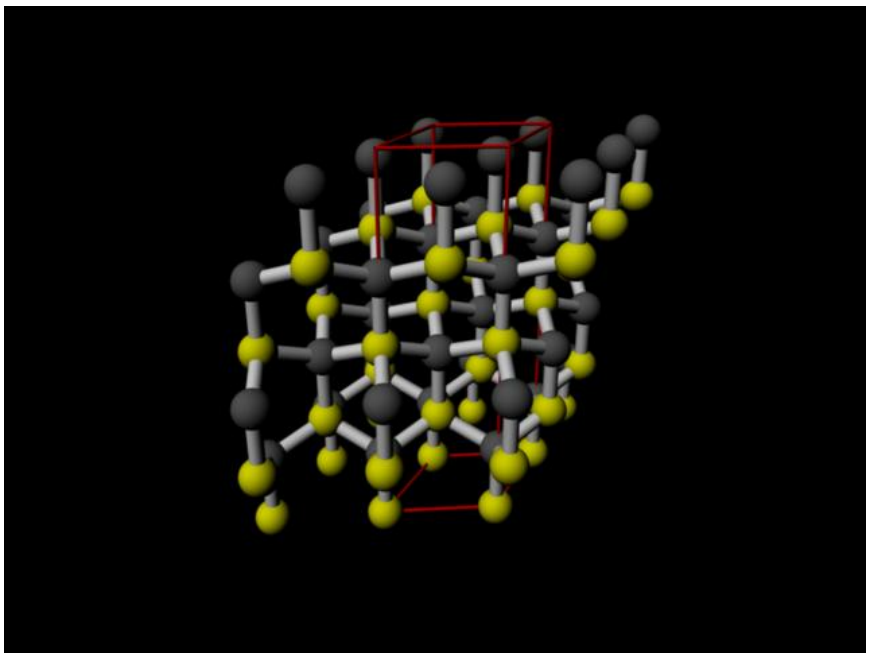

図 5.8: 4H の結晶構造.

5.2.3 6H-SiC

図 5.9 に炭素原子が 6 周期で六方晶をなす 6H-SiC の結晶構造を示す. 赤線で囲まれているのがユニットセルである.

5.2.4 結晶表面構造 (ステップ，キンク，テラス)

結晶成長の成長速度の律速過程を議論する際，結晶表面の構造，結晶表面上の原子の挙動は重要になってくる．図 5.10, 図 5.11, 図 5.12 に結晶の表面構造を連続体から原子レベルにかけて maya で表示した．図 5.10 のテラス状に付着した原子は，テラス状の表面拡散により，ステップの方へと進む．その後ステップでの一次元拡散によりキンクへと付着する (図 5.11, 図 5.12) .

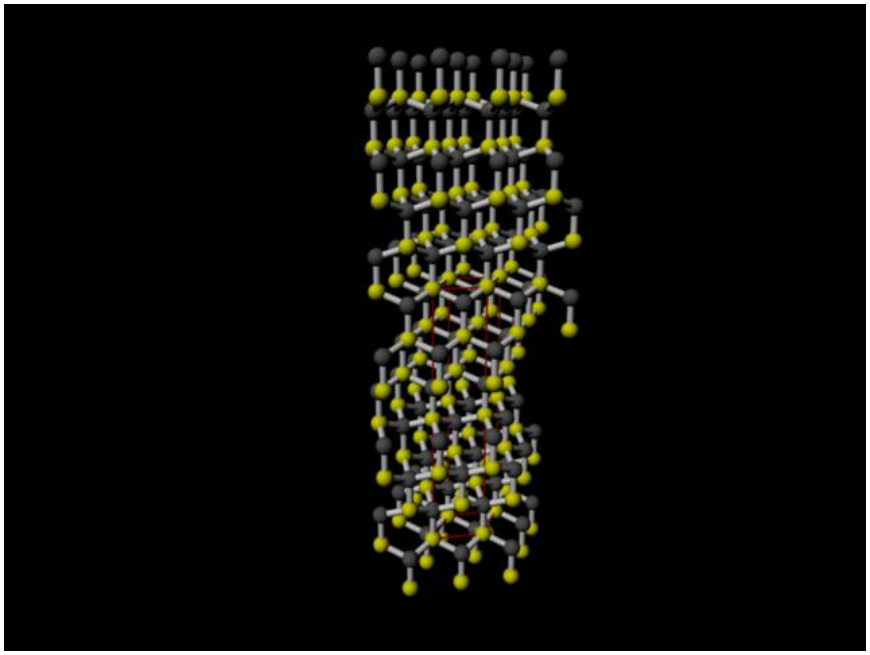

図 5.9: 6H の結晶構造.

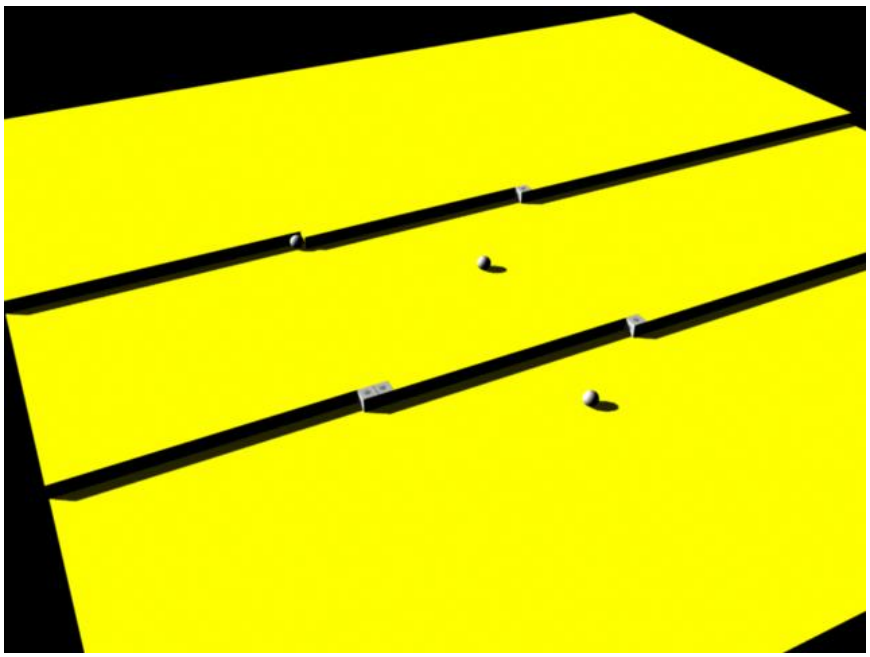

図 5.10: テラス，ステップ，キンク（連続体）.

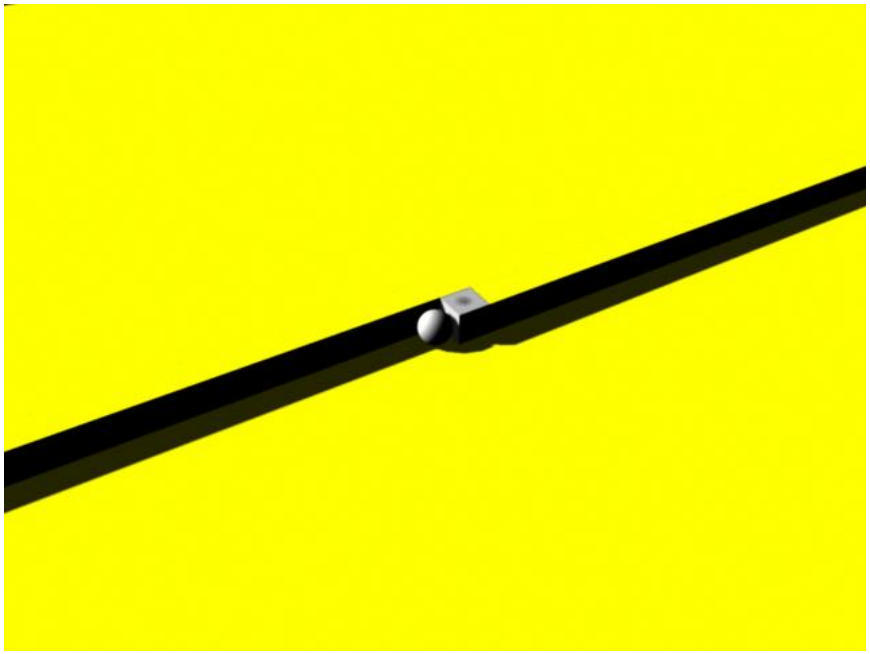

図 5.11: キンクに付着する原子（連続体）.

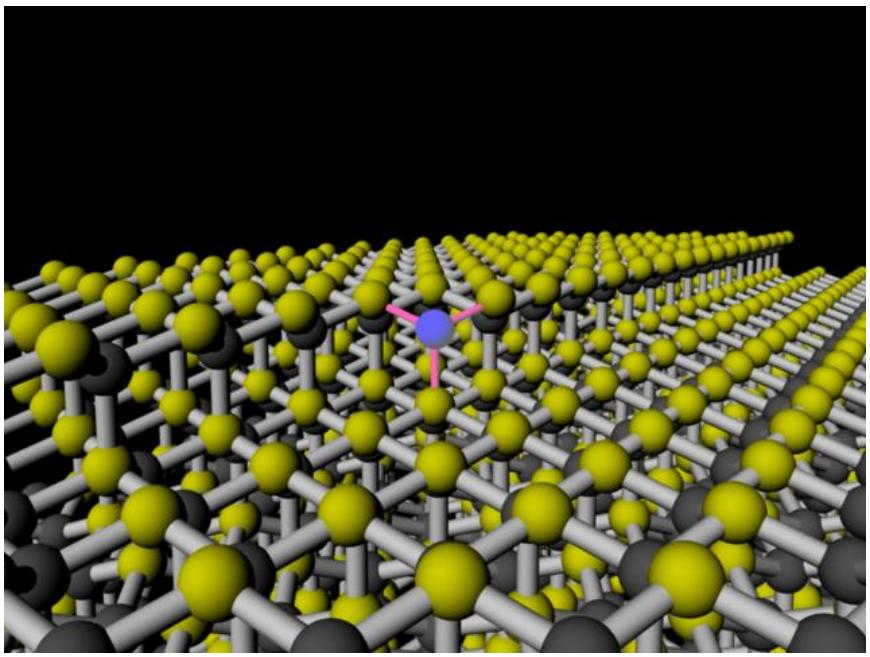

図 5.12: キンクに付着する原子（原子レベル）.

5.2.5 maya による 4H-SiC 螺旋転位

MSE における 4H-SiC の単結晶成長では螺旋転位 (screw dislocation) が発生する．この 4H-SiC で螺旋転位を原子レベルで観察するため，MAYA で再現した．これよりスッテプ以外に異常は見られない事が分かる．積層欠陥や成長時のらせん転位の芯において，原子がどのようなエネルギーを取っているのか今後，第一原理計算等で確認していく必要がある．

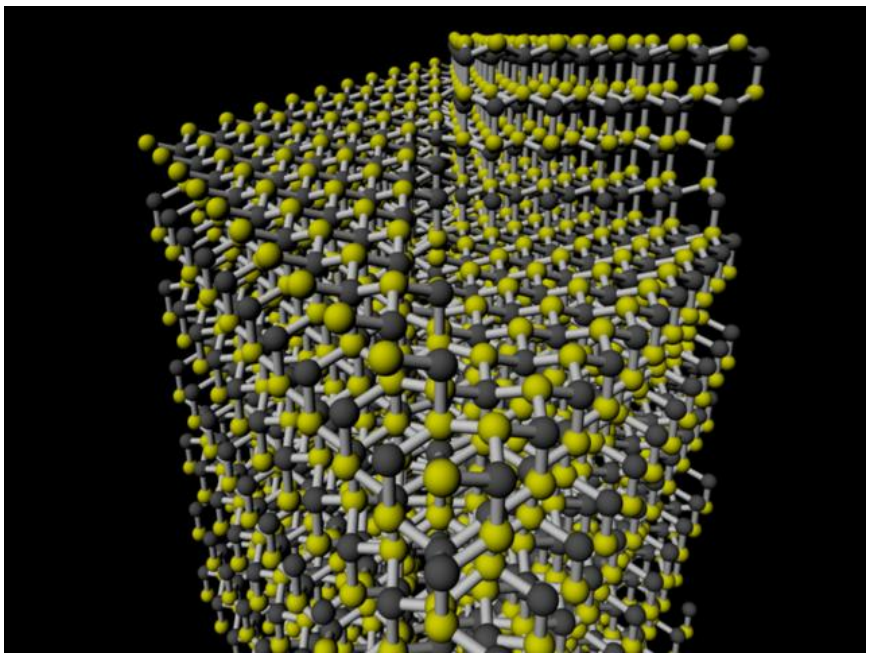

図 5.13: MAYA による 4H-SiC の螺旋転位.

第6章　総括

1. 結晶成長を原子レベルでシミュレートすることによりいくつかの重要な知見が得られた．得られた知見を以下に示す．

   (a) **素過程（放出，拡散，吸着）から溶媒中の炭素濃度プロファイルを予測できる．**
   本研究ではシミュレーションにより溶媒中の炭素濃度を再現した．炭素濃度プロファイルはMSEにおける素過程である「放出」「拡散」「吸着」の３つの物理現象からシミュレートしたものである．またこのシミュレーションにより，MSEでの成長速度もシミュレートできるようになった．[3章3.5.1][3章3.6.1]

   (b) **素過程から得られる炭素濃度勾配は定常状態において必ず直線になる．**
   シミュレーションより得られた濃度プロファイルは定常状態において必ず直線を示していた．実際の系でも濃度プロファイルは直線になっていると考えられる．[3章3.5.2][3章3.6.2]

   (c) **溶媒中の炭素濃度勾配は原料板，基板での固溶限によるとは限らない．**
   炭素濃度勾配は基板，原料板での固溶限によるとは限らないことが分かった．これはシミュレーションにおいて溶媒量は一定であるためと考えられる．実際の系においても温度が一定である事から，固溶限が定まり，溶媒量は一定に保たれるはずである．したがって理論上は実際の系炭素濃度勾配は基板，原料板での固溶限によるとは限らないと言える．しかし実際は基板側の固溶限は1

   (d) **成長速度は溶媒（液体Si）中での拡散が律速している．**
   またMSEにおける単結晶成長において成長速度は溶媒中の拡散が律速していることがシミュレーション結果より導かれた．今後は結晶表面（テラス，ステップ）での拡散が何故「速い」

のかを検証して行く必要性がある．[3 章 3.5.4][3 章 3.6.4]

2. この検証のための１つの試みとして，結晶表面での炭素の挙動の第一原理計算を行った．さらに結晶成長を原子レベルでシミュレートすることによりいくつかの仮説を立てた．

   (a) **Si 面に C 原子が付着する方が安定である．また表面拡散は C 面の方が速い**
   Si 面，Si 修飾をした C 面で合計 6 点を取り第一原理計算を行った．その結果エネルギー差が均衡しているため断定は出来ないが Si 面側に最小なエネルギーが得られた．C 面のほうが最安定エネルギーと準最安定エネルギーの差が小さかった．これより Si 面に C 原子が付着する方が安定であり，また表面拡散は C 面の方が速いという仮説を得た．[4 章 4.3][4 章 4.4.2]

   (b) **表面拡散が異常に速いため付着成長と同じ成長速度を示す．**
   Si 面，Si 修飾をした C 面でそれぞれの最安定エネルギー，準最安定エネルギーの差が Si 面で 0.1ev,Si 修飾した C 面で 0.93ev であった．これより Si 面での表面拡散が異常に速いという結果を得た．しかし第一原理計算は真空と固体での計算である．実際の系は液体と固体であるためその分のずれを今後考えていく必要がある．[4 章 4.3][4 章 4.4.2]

   (c) **成長速度の実験データ（図 3.16）はスパイラル成長（図 3.21）の局所的な所だけ見ている可能性がある．**
   実験データでは成長速度は線形に増加している．線形に増加を示すのは付着成長であり，結晶表面はキンクサイトで構成される．しかし実際の系ではステップとテラスで構成された結晶表面構造になっている．これはスパイラル成長での結晶表面構造である．あくまで１つの仮説としてこの事実より実験データはスパイラル成長の局所的な所だけ見ている可能性があると言える．[3 章 3.6.7][3 章 3.6.8][3 章 3.6.9]

3. MAYA により SiC 単結晶の結晶構造，表面構造が原子レベルで表示可能となった．それを用いて原子レベルでの結晶成長のシミュレーションモデルの視覚化を行った．これにより結晶成長の素過程および律速過程についての知見を得る事が出来た．

謝辞

研究室配属から修士課程修了時まで研究の進め方，学会発表，および論文作成等，終始多大なる有益なご指導，及び丁寧な助言を頂いた西谷滋人教授に深い感謝の意を表します.

また，本研究を進めるにつれ，西谷研究室員の皆様にも様々な知識の供給，ご協力を頂き，本研究を大成する事ができました．最後になりましたが，この場を借りて心から深く御礼申し上げます.

付 録A　プログラム

A.1　溶媒中の炭素濃度プロファイルをシミュレートするプログラム

A.1.1　autocalc

シミュレートする条件を決め，自動で条件を変化させ計算を行うためのスクリプトである．この場合，select を自動で変化させている．

```
require 'mse1d'

autocalc = 10 #<---calc times
path = Dir.pwd

puts 'START'
for i in 0..autocalc-1

  $length = 1000
  $height = 600
  $solventwidth = 20
  $feedlimit = $height/3
  $seedlimit = $height/10
  $feedfront = []
  $seedfront = []
  center = $length/2
  width = $solventwidth/2
  front_move = (center-width-1)/2
  $feedfront = [0,center+width-front_move]
  $seedfront = [0,center-width-1-front_move]
  $time = 10000
```

```
  $select = 1000+100*i

  dir_name = "len"+$length.to_s+"_hei"+$height.to_s+"_sol"
+$solventwidth.to_s+"_select"+$select.to_s+"_feedlimit"
+$feedlimit.to_s+"_seedlimit"+$seedlimit.to_s
  #dir_name = 'sample'

  Dir.chdir(path)
  Dir.mkdir(dir_name)
  ARGV[0]=dir_name

  puts
  puts 'mse ---> '+ARGV[0]
  mse()

end
puts 'END'
```

A.1.2 mse

プロゴラムの中でメインになるスクリプトである．このプログラムでシミュレートが行われ，またシミュレーションの条件がテキストファイルに書き込まれる．

```
def mse

t0 = Time.now
tms0 = Process.times

Dir.chdir("all_def")
require 'make_cell'
require 'checkConcentration'
require 'atomswap'
require 'trymove'
require 'Detach'
```

```
require 'Attach'
Dir.chdir("..")

$array2 = make_cell($height,$length,$seedfront[1]+1,$feedfront[1]-1)
$concentration = []
$lconcentration = make_nm_array($time+2,$length)
$solute = []
$solvent = []
$time2 = 0
$path1 = Dir.pwd
$flag_Attach = 0
first_front = make_nm_array(2,2)
solute_ave = 0
first_front[0][0] = $seedfront[0]
first_front[0][1] = $seedfront[1]
first_front[1][0] = $feedfront[0]
first_front[1][1] = $feedfront[1]

#-----------------------------------------------------main

checkConcentration(0)

#print_front()
#print_matrix()
count0()

kk = 0
Dir.chdir(ARGV[0])  #今いるpathをカレントディレクトリから
(ARGV[0])に移す
for mse in 0..$time
  if($solute.length<$select)
    feedDetach()
    seedDetach()

    solutehead = 0
```

```
  while(solutehead<$solute.length)
    solutehead += trymove(solutehead)
  end
else

  solute_array = []
  for solutehead in 0..$solute.length-1
    solute_array.push solutehead
  end

  flag = 0
  while(flag!=1)
    feedDetach()
    seedDetach()

    $select.times do
      if(solute_array.length>0)
        n = rand(solute_array.length)
        trymove(solute_array[n])

        #checkConcentration(0)

        if($flag_Attach==0)
          solute_array.delete_at(n)
        else
          solute_array.pop
          $flag_Attach = 0
        end
      else
        flag = 1
      end
    end
  end
end
#checkConcentration(mse+1)
```

```
  solute_ave += $solute.length


  if ((mse % 100) == 0)
    checkConcentration(mse+1)
    puts 'check'+mse.to_s
    if(kk<10)
      number = '00'+kk.to_s
    elsif(kk>=10 && kk<100)
      number = '0'+kk.to_s
    else
      number = kk.to_s
    end
    filename1 = []
    filename1 = 'res' + number + '.txt'
    File.open filename1,'w' do |f|
      f.write 0.to_s + ' ' + $concentration[0].to_s + "\n"
    end

    for i in 1..$length-1
      File.open filename1,'a' do |f|
        f.write i.to_s + ' ' + $concentration[i].to_s + "\n"
      end
    end
    kk += 1
  end
  $time2 = kk - 1

end

#puts
#print_front()
#print_matrix()
count0()
puts 'solute_average = '+(solute_ave/$time).to_s
```

```
batch_name = 'res'
filename = 'anime.gpl'
anime_range = 'set xrange [0:' + $length.to_s + ']' + "\n" +
              'set yrange [-1:' + ($height + 10 ).to_s + ']' + "\n"
#Dir.chdir("data")

File.open filename,'w' do |f|
  f.write anime_range
end
for i in 0..$time2
  if(i<10)
      number = '00'+i.to_s
    elsif(i>=10 && i<100)
      number = '0'+i.to_s
    else
      number = i.to_s
    end
  File.open filename,'a' do |f|
    f.write 'plot \'' + batch_name  + number + '.txt\' w l' + "\n"
    + 'pause 1' + "\n"
  end
end

t1 = Time.now
tms1 = Process.times
puts '実時間 = '+(t1-t0).to_s
puts 'CPU時間 = '+(tms1.utime-tms0.utime).to_s

filename_contents = 'contents.txt'
contents_array = []

contents_array.push 'Detach'+"\n"
contents_array.push Time.new
contents_array.push "\n"
```

```
contents_array.push "\n"
contents_array.push 'length = '+$length.to_s+"\n"
contents_array.push 'height = '+$height.to_s+"\n"
contents_array.push 'solventwidth = '+$solventwidth.to_s+"\n"
contents_array.push 'time = '+$time.to_s+"\n"
contents_array.push 'select = '+$select.to_s+"\n"
contents_array.push "\n"
contents_array.push 'seedlimit = '+$seedlimit.to_s+"\n"
contents_array.push 'feedlimit = '+$feedlimit.to_s+"\n"
contents_array.push 'feedlimit/seedlimit = '
+($feedlimit/$seedlimit).to_s+"\n"
contents_array.push 'seedfront初期値 = ['+first_front[0].join(',')+']'
+"\n"
contents_array.push 'feedfront初期値 = ['+first_front[1].join(',')+']'
+"\n"
contents_array.push 'seedfront最終値 = ['+$seedfront.join(',')+']'
+"\n"
contents_array.push 'feedfront最終値 = ['+$feedfront.join(',')+']
'+"\n"
contents_array.push "\n"
contents_array.push 'seed成長度 = '
+($seedfront[1]-first_front[0][1]).to_s+"\n"
contents_array.push 'feed減少度 = '
+($feedfront[1]-first_front[1][1]).to_s+"\n"
contents_array.push "\n"
contents_array.push 'after_solventwidth = '
+($feedfront[1]-$seedfront[1]).to_s+"\n"
contents_array.push 'solute_average = '+(solute_ave/$time).to_s+"\n"
contents_array.push 'solute最終値 = '+($solute.length).to_s+"\n"
contents_array.push "\n"
contents_array.push '実時間 = '+(t1-t0).to_s
contents_array.push "\n"
contents_array.push 'CPU時間 = '+(tms1.utime-tms0.utime).to_s
contents_array.push "\n"
```

```
File.open filename_contents,'w' do |f|
  f.write ''
end

File.open filename_contents,'a' do |f|
  f.write contents_array
end


end
```

A.1.3 makecell

計算をおこなうため，n × m のマトリクスをつくるプログラム．原料板 (feed)，基板 (seed) を 1 とし，溶媒を 0 としている．

```
def make_nm_array(n,m)
  (0...n).map {Array.new(m)}
end

def make_cell(height,length,seedfront,feedfront)
  array = make_nm_array(height,length)
  for i in 0..height-1
    for j in 0..length-1
      if(j<seedfront || j>feedfront)
        array[i][j] = 1
      else
        array[i][j] = 0
      end
    end
  end
  return array
end

def print_matrix
```

```
  for i in 0..$height-1
    print $array2[i]
    puts
  end
end

def count0
  count = 0
  for i in 0..$height-1
    for j in 0..$length-1
      if($array2[i][j]==0)
        count += 1
      end
    end
  end
  puts '0 = '+count.to_s
end

def print_front
  puts 'feedfront = '+$feedfront.join(',')
  puts 'seedfront = '+$seedfront.join(',')
end

def make_solvent
  $solvent = []
  for i in 0..$height-1
    for j in $seedfront[1]+2..$feedfront[1]-2
      if($array2[i][j]==0)
        $solvent.push [i,j]
      end
    end
  end
end
```

A.1.4 Detach

素過程の１つである「放出」をするスクリプトである．界面（原料板）で律速する場合としない場合のふた通りを作成した．

feedDetach（Detach 界面反応律速）

「放出」が界面（原料板側）で律速するスクリプトである．

```
def feedDetach
  ix = $feedfront[0]
  iy = $feedfront[1]
  jx = ix
  jy = iy - 1
  if($array2[jx][jy]==0 && $concentration[$feedfront[1]-1]<$feedlimit)
    $solute.push [jx,jy]
    atomswap($solute.length-1,ix,iy,jx,jy)
    $feedfront[0] += 1

    if($feedfront[0]>$height-1)
      $feedfront[0] = 0
      $feedfront[1] += 1
    end
  end
end
```

feedDerach（溶媒中の拡散律速）

「放出」が界面（原料板側）で律速せずスムーズに行われるスクリプトである．

```
def interface_check(y)
  $interface = []
  for i in 0..$height-1
    if($array2[i][y]==0)
      $interface.push i
    end
```

```
  end
end

def detach_check(jx,jy,limit)
  interface_check(jy)

  if(($height-$interface.length)>limit)
    return 0
  else
    return detach_serch(jx,jy,$interface)
  end
end

def detach_serch(jx,jy,interface)
  temp = ($height*$height)
  for i in 0..interface.length-1
    a=( (interface[i]-jx)*(interface[i]-jx) )
    if(temp>a)
      temp = a
      $new_jx = interface[i]
    end
  end
  return 1
end

def feedDetach
  ix = $feedfront[0]
  iy = $feedfront[1]
  jx = ix
  jy = iy - 1
  if($array2[jx][jy]==0 && $concentration[$feedfront[1]-1]<$feedlimit)
    $solute.push [jx,jy]
    atomswap($solute.length-1,ix,iy,jx,jy)
    $feedfront[0] += 1

    if($feedfront[0]>$height-1)
```

```
        $feedfront[0] = 0
        $feedfront[1] += 1
      end
    end
  end
```

seedDetach

seed (基板側）で「放出するスクリプトである.

```
  def seedDetach
    ix = $seedfront[0]
    iy = $seedfront[1]
    jx = ix
    jy = iy + 1
    if($array2[jx][jy]==0 && $concentration[$seedfront[1]+1]<$seedlimit)
      $solute.push [jx,jy]
      atomswap($solute.length-1,ix,iy,jx,jy)
      $seedfront[0] += 1

      if($seedfront[0]>$height-1)
        $seedfront[0] = 0
        $seedfront[1] -= 1
      end
    end
  end
```

A.1.5 Attach

素過程の１つである「吸着」を行うスクリプトである.

feedAttach

feed （原料板）で「吸着」をするスクリプトである.

```
def feedAttach(iatom,ix,iy,jx,jy)
  if($array2[jx][jy]==0 && $concentration[$feedfront[1]-1]>$feedlimit)
    wardenX = $feedfront[0]-1
    wardenY = $feedfront[1]
    if(wardenX<0)
      wardenX = $height-1
      wardenY += 1
    end

    if($array2[wardenX][wardenY]==1)
      escape_serch(wardenX,wardenY)
      puts 'feed_escape'
      puts
    end

    $solute.delete_at(iatom)
    $flag_Attach = 1
    $array2[ix][iy] = 0
    $feedfront[0] -= 1
    if($feedfront[0]<0)
      $feedfront[0] = $height-1
      $feedfront[1] -= 1
    end
    $array2[$feedfront[0]][$feedfront[1]] = 1
    return 0
  end
  return 1
end
```

seedAttach

seed（基板）で「吸着」をするスクリプトである.

```
def seedAttach(iatom,ix,iy,jx,jy)
  if($concentration[$seedfront[1]+1]>$seedlimit)
```

```
      wardenX = $seedfront[0]-1
      wardenY = $seedfront[1]
      if(wardenX<0)
        wardenX = $height-1
        wardenY += 1
      end

      if($array2[wardenX][wardenY]==1)
        escape_serch(wardenX,wardenY)
        puts 'seed_escape'
        puts
      end

      $solute.delete_at(iatom)
      $flag_Attach = 1
      $array2[ix][iy] = 0
      $seedfront[0] -= 1
      if($seedfront[0]<0)
        $seedfront[0] = $height-1
        $seedfront[1] += 1
      end
      $array2[$seedfront[0]][$seedfront[1]] = 1
      return 0
    end
    return 1
  end
```

A.1.6 escape

```
  def escape(solutehead,wardenX,wardenY)
    temp = ($height*$height)+($solventwidth*$solventwidth)
    for serch in 0..$solvent.length-1
      a=( ($solvent[serch][0]-wardenX)*($solvent[serch][0]-wardenX) )
      b=( ($solvent[serch][1]-wardenY)*($solvent[serch][1]-wardenY) )
      c = a+b
```

```
    if(temp>c)
      temp = c
      x = $solvent[serch][0]
      y = $solvent[serch][1]
    end
  end

  puts 'from = '+$solute[solutehead].join(',')
  #atomswap(solutehead,wardenX,wardenY,x,y)
  $solute[solutehead][0] = x
  $solute[solutehead][1] = y
  puts ' to  = '+$solute[solutehead].join(',')
end

def escape_serch(wardenX,wardenY)
  for serch in 0..$solute.length-1
    if($solute[serch][0]==wardenX && $solute[serch][1]==wardenY)
      make_solvent()
      escape(serch,wardenX,wardenY)
    end
  end
end
```

A.1.7 Atomswap

隣同士の原子を交換するスクリプトである．これにより素過程の１つである「拡散」が生じる．

```
def atomswap(iatom,ix,iy,jx,jy)
  $array2[ix][iy] = 0
  $array2[jx][jy] = 1
  $solute[iatom][0] = jx
  $solute[iatom][1] = jy
end
```

A.1.8 checkConcentration

レイヤーごとの炭素濃度（1）を求めるスクリプトである.

```
def checkConcentration(num)
  $concentration = []
  for iy in 0..$length-1
    concent_sum = 0
    for ix in 0..$height-1
      if($array2[ix][iy]==1)
        concent_sum += 1
      end
    end
    $concentration.push concent_sum
    $lconcentration[num].push concent_sum.to_s + ' '
  end
end
```

A.2 SiCの結晶構造

A.2.1 vectordate3C

3C-SiC の結晶構造を作るために必要なベクトルデータである.

```
Primitive vectors
a(1) =   4.36 0.00000000  0.00000000
a(2) =   0.00000000  0.00000000  4.36
a(3) =   0.00000000  4.36 0.00000000


Volume =  82.87874525


Reciprocal vectors
b(1) =   0.32462157 -0.18742035  0.00000000
b(2) =   0.32462157  0.18742035  0.00000000
```

```
b(3) =   0.00000000  0.00000000  0.09915913


Basis Vectors:8
Atom    Lattice Coordinates

Si  0.25  0.25  0.25
Si  0.75  0.25  0.75
C   0.00000000  0.00000000  0.00000000
C   0.50000000  0.00000000  0.50000000
Si  0.75  0.75  0.25
Si  0.25  0.75  0.75
C   0.50000000  0.50000000  0.00000000
C   0.00000000  0.50000000  0.50000000

Cartesian Coordinates
Si  0.00000000  0.00000000  0.00000000
Si  0.00000000  0.00000000  5.04240000
C  0.00000000  0.00000000  1.89090000
C  0.00000000  0.00000000  6.93330000
Si  1.54025500  0.88926664  2.51943516
Si  1.54025500 -0.88926664  7.56183516
C  1.54025500  0.88926664  4.41033516
C  1.54025500 -0.88926664  9.45273516
```

A.2.2 vectordate4H

```
Primitive vectors
a(1) =   1.54025500 -2.66779992  0.00000000
a(2) =   1.54025500  2.66779992  0.00000000
a(3) =   0.00000000  0.00000000 10.08480000


Volume =  82.87874525
```

```
Reciprocal vectors
b(1) =   0.32462157 -0.18742035  0.00000000
b(2) =   0.32462157  0.18742035  0.00000000
b(3) =   0.00000000  0.00000000  0.09915913


Basis Vectors:
Atom    Lattice Coordinates

Si  0.00000000  0.00000000  0.00000000
Si  0.00000000  0.00000000  0.50000000
C   0.00000000  0.00000000  0.18750000
C   0.00000000  0.00000000  0.68750000
Si  0.33333333  0.66666667  0.24982500
Si  0.66666667  0.33333333  0.74982500
C   0.33333333  0.66666667  0.43732500
C   0.66666667  0.33333333  0.93732500

Cartesian Coordinates

Si  0.00000000  0.00000000  0.00000000
Si  0.00000000  0.00000000  5.04240000
C   0.00000000  0.00000000  1.89090000
C   0.00000000  0.00000000  6.93330000
Si  1.54025500  0.88926664  2.51943516
Si  1.54025500 -0.88926664  7.56183516
C   1.54025500  0.88926664  4.41033516
C   1.54025500 -0.88926664  9.45273516
```

A.2.3 vectordate6H

6H-SiC の結晶構造を作るために必要なベクトルデータである.

```
Primitive vectors
a(1) =   1.54035000 -2.66796446   .00000000
a(2) =   1.54035000  2.66796446   .00000000
a(3) =    .00000000   .00000000 15.11740000


Volume = 124.25290558


Reciprocal vectors
b(1) =    .32460155  -.18740879   .00000000
b(2) =    .32460155   .18740879  -.00000000
b(3) =   -.00000000   .00000000   .06614894


Basis Vectors:12
Atom    Lattice Coordinates

C    .00000000   .00000000   .12540000
C    .00000000   .00000000   .62540000
Si   .00000000   .00000000   .00000000
Si    .00000000   .00000000   .50000000
C    .33333333   .66666667   .79190000
C    .66666667   .33333333   .29190000
C    .33333333   .66666667   .45840000
C    .66666667   .33333333  -.04160000
Si    .33333333   .66666667   .66670000
Si    .66666667   .33333333   .16670000
Si    .33333333   .66666667   .33320000
Si    .66666667   .33333333  -.16680000

 Cartesian Coordinates
C   .00000000   .00000000  1.89572196
C   .00000000   .00000000  9.45442196
Si  .00000000   .00000000   .00000000
```

```
Si  .00000000   .00000000  7.55870000
C   1.54035000   .88932149 11.97146906
C   1.54035000  -.88932149  4.41276906
C   1.54035000   .88932149  6.92981616
C   1.54035000  -.88932149  -.62888384
Si  1.54035000   .88932149 10.07877058
Si  1.54035000  -.88932149  2.52007058
Si  1.54035000   .88932149  5.03711768
Si 1.54035000  -.88932149 -2.52158232
```

A.2.4 input vector data,make primitive vector

必要となる結晶構造のベクトルデータを読み込み，primitive vector に変換するスクリプトである.

```
include Math
require 'yaml'
require 'pp'
require "matrix"

def input(vec_data)
  $y_vec_data = vec_data
  if(vec_data=='3c')
    input_3c(vec_data)
  else
    input_4h_6h(vec_data)
  end
end
def input_3c(vec_data)
Dir.chdir("data"){

    infile = File.read("vector_data_3c")

p_vec=[]
for i in 0..2 do
  p1 = infile.grep(/a.*$/)[i].split(/=/)[1].chomp.split(/ +/)
```

```
  p_vec.push Vector[p1[1].to_f,p1[2].to_f,p1[3].to_f]
end

num1=infile.grep(/Basis Vectors.*$/)[0].split(/:/)[1].to_i/2

si_ball=[]
for i in 0..num1-1 do
  p1=infile.grep(/Si  /)[i].split(/ +/)
  si_ball.push Vector[p1[1].to_f,p1[3].to_f,p1[2].to_f]
end

c_ball=[]
for i in 0..num1-1 do
  p1=infile.grep(/C  /)[i].split(/ +/)
  c_ball.push Vector[p1[1].to_f,p1[3].to_f,p1[2].to_f]
end

ball0=[]
element=[]

for i in 0..num1-1 do
  ball0.push si_ball[i]
  element.push 0
  ball0.push c_ball[i]
  element.push 1
end

ballE=element
unitcellE=element
ball1=[]
unitcell=[]
for i in 0..ball0.size-1 do
  ball1.push p_vec[0]*ball0[i][0]
+p_vec[2]*ball0[i][1]+p_vec[1]*ball0[i][2]
  unitcell.push p_vec[0]*ball0[i][0]
```

```
+p_vec[2]*ball0[i][1]+p_vec[1]*ball0[i][2]
end

return ball1,ballE,unitcell,unitcellE,p_vec
}
end
def input_4h_6h(vec_data)
Dir.chdir("data"){

    if(vec_data=='4h')
      infile = File.read("vector_data_4h")
    elsif(vec_data=='6h')
      infile = File.read("vector_data_6h")
    end

p_vec=[]
for i in 0..2 do
  p1 = infile.grep(/a.*$/)[i].split(/=/)[1].chomp.split(/ +/)
  p_vec.push Vector[p1[1].to_f,p1[3].to_f,p1[2].to_f]
end

num1=infile.grep(/Basis Vectors.*$/)[0].split(/:/)[1].to_i/2

si_ball=[]
for i in 0..num1-1 do
  p1=infile.grep(/Si  /)[i].split(/ +/)
  si_ball.push Vector[p1[1].to_f,p1[3].to_f,p1[2].to_f]
end

c_ball=[]
for i in 0..num1-1 do
  p1=infile.grep(/C  /)[i].split(/ +/)
  c_ball.push Vector[p1[1].to_f,p1[3].to_f,p1[2].to_f]
end
```

```
ball0=[]
element=[]

for i in 0..num1-1 do
  ball0.push si_ball[i]
  element.push 0
  ball0.push c_ball[i]
  element.push 1
end

ballE=element
unitcellE=element
ball1=[]
unitcell=[]
for i in 0..ball0.size-1 do
  ball1.push p_vec[0]*ball0[i][0]
+p_vec[2]*ball0[i][1]+p_vec[1]*ball0[i][2]
  unitcell.push p_vec[0]*ball0[i][0]
+p_vec[2]*ball0[i][1]+p_vec[1]*ball0[i][2]
end

return ball1,ballE,unitcell,unitcellE,p_vec
}
end
def input_nucleus(vec_data)
Dir.chdir("data"){
    if(vec_data=='bcc')
      infile = File.read("vector_data_nucleus_bcc")
    end
p_vec=[]
for i in 0..2 do
  p1 = infile.grep(/a.*$/)[i].split(/=/)[1].chomp.split(/ +/)
  p_vec.push Vector[p1[1].to_f,p1[3].to_f,p1[2].to_f]
end
```

```
num1=infile.grep(/Basis Vectors.*$/)[0].split(/:/)[1].to_i/2

atom1=[]
for i in 0..num1-1 do
  p1=infile.grep(/atom1  /)[i].split(/ +/)
  atom1.push Vector[p1[1].to_f,p1[3].to_f,p1[2].to_f]
end

atom2=[]
for i in 0..num1-1 do
  p1=infile.grep(/atom2  /)[i].split(/ +/)
  atom2.push Vector[p1[1].to_f,p1[3].to_f,p1[2].to_f]
end

ball0=[]
element=[]

for i in 0..num1-1 do
  ball0.push atom1[i]
  ball0.push atom2[i]
end

atom1_ball=[]
atom2_ball=[]
unitcell=[]
for i in 0..num1-1 do
  atom1_ball.push p_vec[0]*atom1[i][0]+p_vec[2]*atom1[i][1]
+p_vec[1]*atom1[i][2]
  atom2_ball.push p_vec[0]*atom2[i][0]+p_vec[2]*atom2[i][1]
+p_vec[1]*atom2[i][2]
  unitcell.push p_vec[0]*atom1[i][0]+p_vec[2]*atom1[i][1]
+p_vec[1]*atom1[i][2]
  unitcell.push p_vec[0]*atom2[i][0]+p_vec[2]*atom2[i][1]
+p_vec[1]*atom2[i][2]
end
```

```
return atom1_ball,atom2_ball,unitcell,p_vec
}
end
```

A.2.5 makecell

MAYA で表示させるボール（原子）を作り，また原子と原子を繋ぐシリンダー（結合）を作るスクリプトである．

```
include Math

$cell_ball_count = 0
$cell_cylinder_count = 0


def make_cell(file1,ball1,ballE,color1,color2,color3,name1,name2)

  file1 = make_sphere_proc(file1,color1,name1,[0.5,0.5,0.5])
  file1 = make_sphere_proc(file1,color2,name2,[0.5,0.5,0.5])

  for i in 0..ball1.size-1 do
    if(ballE[i]==0)
      file1.printf("make_%s(%7.5f,%7.5f,%7.5f);n",
name1,ball1[i][0],ball1[i][1],ball1[i][2])
    else
      file1.printf("make_%s(%7.5f,%7.5f,%7.5f);n",
name2,ball1[i][0],ball1[i][1],ball1[i][2])
    end
  end

  nmax=ball1.size
  comb_tmp=[]

  for i in 0..nmax-1 do
```

```
    for j in 0..nmax-1 do
      tmp_vector=ball1[i]-ball1[j]

      if tmp_vector.r < 2.0 && tmp_vector.r!=0 then
        comb_tmp.push([i,j])
      end
    end
  end
  $cell_ball_count = ball1.size
  file1 = make_cylinder_proc(file1,color3,[1,0.2,0.2])
  file1 = make_cylinder(file1,ball1,comb_tmp)
  return file1
end
def make_sphere_proc(file1,color,name,scale)
  file1.printf("global proc make_%s(float $x,
float $y,float $z)n",name)
  file1.printf("{n")
  file1.printf("sphere;n")
  file1.printf("scale %7.5f %7.5f %7.5f;n",
scale[0],scale[1],scale[2])
  file1.printf("move $x $y $z;n")
  if(color!=1)
    file1.printf("sets -e -forceElement lambert%dSG;n",color)
  end
  file1.printf("}n")
  return file1
end
$cylinder_proc_count = 0
def make_cylinder_proc(file1,color,scale)
  $cylinder_proc_count += 1
  d = $cylinder_proc_count
  file1.printf("global proc make_cylinder%d(float $x,
float $y,float $z,float $b2,float $a2)n",d)
  file1.printf("{n")
  file1.printf("cylinder;n")
```

```
  file1.printf("scale %7.5f %7.5f %7.5f;n",scale[0],scale[1],scale[2])
  file1.printf("move $x $y $z;n")
  file1.printf("rotate 0 $b2 $a2;n")
  file1.printf("sets -e -forceElement lambert%dSG;n",color)
  file1.printf("}n")
  return file1
end
def make_cylinder(file1,ball1,comb_tmp)
  comb_tmp.each{|ij|
    half_v=(ball1[ij[0]]+ball1[ij[1]])*0.5
    tmp=ball1[ij[0]]-ball1[ij[1]]
    if tmp[1] < 0  then
      dir_vec=ball1[ij[1]]-ball1[ij[0]]
    else
      dir_vec=ball1[ij[0]]-ball1[ij[1]]
    end
    x=dir_vec[0]
    y=dir_vec[1]
    z=dir_vec[2]
    aa=1/sqrt(x**2+y**2+z**2)
    x1=x*aa
    y1=y*aa
    z1=z*aa
    b1=-asin(z1)
    a1=acos(x1/cos(b1))
    a2=a1*180/PI
    b2=b1*180/PI
    file1.printf("make_cylinder%d(%7.5f,%7.5f,%7.5f,%7.5f,%7.5f);n",
$cylinder_proc_count,half_v[0],half_v[1],half_v[2],b2,a2)
    $cell_cylinder_count += 1
  }
  return file1
end

def make_cell_nucleus(file1,ball1,color_ball,color_cylinder)
```

```
  file1 = make_sphere_proc(file1,color_ball,'atom',[0.8,0.8,0.8])

  for i in 0..ball1.size-1 do
    file1.printf("make_atom(%7.5f,%7.5f,%7.5f);n",
ball1[i][0],ball1[i][1],ball1[i][2])
  end

  nmax=ball1.size
  comb_tmp=[]

  for i in 0..nmax-1 do
    for j in 0..nmax-1 do
      tmp_vector=ball1[i]-ball1[j]

      if tmp_vector.r < 4.5 && tmp_vector.r!=0 then
        comb_tmp.push([i,j])
      end
    end
  end
  $cell_ball_count = ball1.size
  #file1 = make_cylinder_proc(file1,color_cylinder,[2,0.3,0.3])
  #file1 = make_cylinder(file1,ball1,comb_tmp)
  return file1
end
```

A.2.6 color

MAYA で表示させる色を決めるスクリプトである.

```
require 'pp'

$color_count=1
def color_add(file1,rgb)
  cc = $color_count
  #if(cc==1)
```

```
  #  file1.printf("setAttr "lambert%d.color"
  # -type double3 %1.5f %1.5f %1.5f;n",cc,rgb[0],rgb[1],rgb[2])
  #else
    file1.printf("shadingNode -asShader lambert;n")
    file1.printf("sets -renderable true
-noSurfaceShader true -empty -name lambert%dSG;n",cc)
    file1.printf("connectAttr -f lambert%d.outColor
lambert%dSG.surfaceShader;n",cc,cc)
    file1.printf("select -r lambert%d;n",cc)
    file1.printf("setAttr "lambert%d.color"
 -type double3 %1.5f %1.5f %1.5f;n",cc,rgb[0],rgb[1],rgb[2])
  #end
  $color_count += 1
  return file1,cc
end
def color_trans_add(file1,rgb,trgb)
  cc = $color_count
  if(cc==1)
    file1.printf("setAttr "lambert%d.color"
-type double3 %1.5f %1.5f %1.5f;n",cc,rgb[0],rgb[1],rgb[2])
    file1.printf("setAttr "lambert%d.transparency"
-type double3 %1.5f %1.5f %1.5f;n",cc,trgb[0],trgb[1],trgb[2])
  else
    file1.printf("shadingNode -asShader lambert;n")
    file1.printf("sets -renderable true -noSurfaceShader
true -empty -name lambert%dSG;n",cc)
    file1.printf("connectAttr -f lambert%d.outColor
 lambert%dSG.surfaceShader;n",cc,cc)
    file1.printf("select -r lambert%d;n",cc)
    file1.printf("setAttr "lambert%d.color"
-type double3 %1.5f %1.5f %1.5f;n",cc,rgb[0],rgb[1],rgb[2])
    file1.printf("setAttr "lambert%d.transparency"
-type double3 %1.5f %1.5f %1.5f;n",cc,trgb[0],trgb[1],trgb[2])
  end
  $color_count += 1
```

```
  return file1,cc
end
def color_incan_add(file1,rgb,trgb)
  cc = $color_count
  if(cc==1)
    file1.printf("setAttr "lambert%d.color"
-type double3 %1.5f %1.5f %1.5f;n",cc,rgb[0],rgb[1],rgb[2])
    file1.printf("setAttr "lambert%d.transparency"
 -type double3 %1.5f %1.5f %1.5f;n",cc,trgb[0],trgb[1],trgb[2])
  else
    file1.printf("shadingNode -asShader lambert;n")
    file1.printf("sets -renderable true -noSurfaceShader
 true -empty -name lambert%dSG;n",cc)
    file1.printf("connectAttr -f lambert%d.outColor
lambert%dSG.surfaceShader;n",cc,cc)
    file1.printf("select -r lambert%d;n",cc)
    file1.printf("setAttr "lambert%d.color"
-type double3 %1.5f %1.5f %1.5f;n",cc,rgb[0],rgb[1],rgb[2])
    file1.printf("setAttr "lambert%d.incandescence"
-type double3 %1.5f %1.5f %1.5f;n",cc,trgb[0],trgb[1],trgb[2])
  end
  $color_count += 1
  return file1,cc
end
```

A.2.7 unitcell

ユニットセルベクトルを決定し，ユニットセルごとに三次元的にセル
を拡張させるスクリプトである.

```
include Math
require 'yaml'
require 'pp'
require "matrix"
def point_unitcell(ball1,ballE,unitcell,unitcellE,p_vec,x,y,z,i)
```

```
  if($y_vec_data=='3c')
    y_vec=Vector[0,4.36,0]
  elsif($y_vec_data=='4h')
    y_vec=Vector[0,10.08506484,0]
  elsif($y_vec_data=='6h')
    y_vec=Vector[0,15.11740000,0]
  end
  ball1.push unitcell[i]+p_vec[0]*x+p_vec[1]*z+y_vec*y
  ballE.push unitcellE[i]
  return ball1,ballE
end
def unitcell_basis(ball1,ballE,unitcell,unitcellE,p_vec,xyz)
  unimax = unitcell.size
  n1 = xyz[0]
  n2 = xyz[1]
  n3 = xyz[2]
  for x in -n1..n1 do
    for y in -n2..n2 do
      for z in -n3..n3 do
        for i in 0..unimax-1 do
          if x==0 && z==0 && y==0 then
          else
            ball1,ballE=point_unitcell(ball1,ballE,
unitcell,unitcellE,p_vec,x,y,z,i)
          end
          #if(i==0)
            #ball1,ballE=point_unitcell(ball1,ballE,
unitcell,unitcellE,p_vec,x,y+1,z,i)
          #end
        end
      end
    end
  end
  return ball1
end
```

A.3　螺旋転移

A.3.1　main スクリプト

require で必要な関数を呼び出し，実行するメインスクリプトである．

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

Dir.chdir("lib"){
require 'input'
require 'unitcell'
require 'color_add'
require 'make_cell'
require 'cell_support'
require 'currentTime'
require 'object'
require 'light'
require 'camera'
}
filename1="maya_dislocation.mel"


ball1,ballE,unitcell,unitcellE,p_vec = input('4h')


ball1 = unitcell_dislocation(ball1,ballE,unitcell,
unitcellE,p_vec,[5,1,5])
ball1 = make_spiral(ball1)

Dir.chdir("mel"){
  file1 = File.open(filename1,"w")
  file1,color1 = color_add(file1,[1,1,0]) #color1 = color_label
```

```
  file1,color2 = color_add(file1,[0.4,0.4,0.4])
  file1,color3 = color_add(file1,[1,1,1])
  file1,color4 = color_add(file1,[1,0,0])

  file1 = make_cell(file1,ball1,ballE,color1,color2,color3,'Si','C')

  file1.close

  puts 'output >>> '+filename1
}
```

A.3.2 makespiral

primitive vector から必要な結晶構造のユニットセル，螺旋転位に必要なユニットセルを作成し，スパイラル構造を作るスクリプトである．

```
include Math
require 'yaml'
require 'pp'
require "matrix"

def unitcell_dislocation(ball1,ballE,unitcell,unitcellE,p_vec,xyz)
  unimax = unitcell.size
  n1 = xyz[0]
  n2 = xyz[1]
  n3 = xyz[2]
  for x in -n1..n1 do
    for y in -n2..n2 do
      for z in -n3..n3 do
        for i in 0..unimax-1 do
          if x==0 && z==0 && y==0 then
          else
            ball1,ballE=point_unitcell(ball1,ballE,
unitcell,unitcellE,p_vec,x,y,z,i)
          end
```

```
          if(i==0)
            ball1,ballE=point_unitcell(ball1,ballE,
unitcell,unitcellE,p_vec,x,y+1,z,i)
          end
        end
      end
    end
  end
  return ball1
end
def make_spiral(ball1)
  for i in 0..ball1.length-1 do
    len = ball1[i].r
    if(len!=0)
      theta = asin(ball1[i][2]/len)
      theta = theta*(180/PI)
      if(ball1[i][0]<0 && ball1[i][2]>=0)
        theta = 180-theta
      elsif(ball1[i][0]<=0 && ball1[i][2]<0)
        theta = 180-theta
      elsif(ball1[i][0]>0 && ball1[i][2]<=0)
        theta = 360+theta
      end
      #add = (1-(theta/360)) #var continuous
      #add = (1-(theta/360))*10.08506484 #var ball and stick
      add = (theta/360)*10.08506484 #var ball and stick
      ball1[i] += Vector[0,add,0]
    end
  end
  return ball1
end
def make_spiral_var_test(ball1)
  for i in 0..ball1.length-1 do
    len = ball1[i].r
    if(len!=0)
```

```
      theta = asin(ball1[i][2]/len)
      theta = theta*(180/PI)
      if(ball1[i][0]>0 && ball1[i][2]<0)
        theta = 360+theta
        add = (1-(theta/360))*5
        ball1[i] += Vector[0,add,0]
      end
    end
  end
  return ball1
end
def make_spiral_6face(ball1)
  #pp ball1
  for i in 0..ball1.length-1 do
    if(ball1[i][2]>=sqrt(3)*ball1[i][0] && ball1[i][2]<0)
      ball1[i] += Vector[0,1,0]
    elsif(ball1[i][2]<sqrt(3)*ball1[i][0] && ball1[i][2]<=
-sqrt(3)*ball1[i][0])
      ball1[i] += Vector[0,0.8,0]
    elsif(ball1[i][2]>-sqrt(3)*ball1[i][0] && ball1[i][2]<=0)
      ball1[i] += Vector[0,0.6,0]
    elsif(ball1[i][2]>0 && ball1[i][2]<sqrt(3)*ball1[i][0])
      ball1[i] += Vector[0,0.4,0]
    elsif(ball1[i][2]>sqrt(3)*ball1[i][0] && ball1[i][2]>=
-sqrt(3)*ball1[i][0])
      ball1[i] += Vector[0,0.2,0]
    end
  end
  return ball1
end
def nucleus_cluster(atom1_ball,atom2_ball,p_vec)
ball1 = []
  for i in 0..atom1_ball.size-1
  ball1.push atom1_ball[i]
end
```

```
for i in 0..atom1_ball.size-1
  ball1.push atom1_ball[i]+p_vec[1]*1
end
for i in 0..atom2_ball.size-1
  ball1.push atom2_ball[i]
end
for i in 0..atom2_ball.size-1
  ball1.push atom2_ball[i]+p_vec[0]*(-1)
end
for i in 0..atom2_ball.size-1
  ball1.push atom2_ball[i]+p_vec[0]*(-1)+p_vec[2]*(-1)
end
  return ball1
end
def unitcell_nucleus(unitcell,p_vec,xyz)
  ball1 = []
  #ball1.push unitcell
  unimax = unitcell.size
  n1 = xyz[0]
  n2 = xyz[1]
  n3 = xyz[2]
  for x in -n1..n1 do
    for y in -n2..n2 do
      for z in -n3..n3 do
        for i in 0..unimax-1 do
          #if x==0 && z==0 && y==0 then
          #else
            ball1.push unitcell[i]+p_vec[0]*x+p_vec[1]*y+p_vec[2]*z
          #end
        end
      end
    end
  end
  return ball1
end
```

A.4 Kossel model

A.4.1 terrace,step,kink

include Math require 'yaml' require 'pp' require "matrix"

```
def unitcell_terrace_step_kink(ball1,ballE,unitcell,
unitcellE,p_vec,xyz)
  y_vec=Vector[0,10.08506484,0]
  unimax = unitcell.size
  n1 = xyz[0]
  n2 = xyz[1]
  n3 = xyz[2]
  for x in -n1..n1 do
    for y in -n2..n2 do
      for z in -n3..n3 do
        for i in 0..unimax-1 do
          if x==0 && y==0 && z==0 then
          else
            ball1,ballE=point_unitcell(ball1,ballE,
unitcell,unitcellE,p_vec,x,y,z,i)
          end
          if(i==0)
            ball1,ballE=point_unitcell(ball1,ballE,
unitcell,unitcellE,p_vec,x,y+1,z,i)
          end
        end
      end
    end
  end

  kk = 1
  for p in -n1-4..n1-4 do
    for q in -n3..n3 do
```

```
        if(q>5)
          i=1
          ball1.push unitcell[i]+p_vec[0]*(p-1)+p_vec[1]*q+y_vec*kk
          ballE.push unitcellE[i]
          i=4
          ball1.push unitcell[i]+p_vec[0]*(p-1)+p_vec[1]*q+y_vec*kk
          ballE.push unitcellE[i]
        else
          i=1
          ball1.push unitcell[i]+p_vec[0]*p+p_vec[1]*q+y_vec*kk
          ballE.push unitcellE[i]
          i=4
          ball1.push unitcell[i]+p_vec[0]*p+p_vec[1]*q+y_vec*kk
          ballE.push unitcellE[i]
        end
    end
  end

  return ball1
end
```

A.4.2 nucleus

```
include Math
require 'yaml'
require 'pp'
require "matrix"
def nucleus_cluster(atom1_ball,atom2_ball,p_vec)
ball1 = []
  for i in 0..atom1_ball.size-1
  ball1.push atom1_ball[i]
end
for i in 0..atom1_ball.size-1
  ball1.push atom1_ball[i]+p_vec[1]*1
end
```

```
for i in 0..atom2_ball.size-1
  ball1.push atom2_ball[i]
end
for i in 0..atom2_ball.size-1
  ball1.push atom2_ball[i]+p_vec[0]*(-1)
end
for i in 0..atom2_ball.size-1
  ball1.push atom2_ball[i]+p_vec[0]*(-1)+p_vec[2]*(-1)
end
  return ball1
end
def unitcell_nucleus(unitcell,p_vec,xyz)
  ball1 = []
  #ball1.push unitcell
  unimax = unitcell.size
  n1 = xyz[0]
  n2 = xyz[1]
  n3 = xyz[2]
  for x in -n1..n1 do
    for y in -n2..n2 do
      for z in -n3..n3 do
        for i in 0..unimax-1 do
          #if x==0 && z==0 && y==0 then
          #else
            ball1.push unitcell[i]+p_vec[0]*x+p_vec[1]*y+p_vec[2]*z
          #end
        end
      end
    end
  end
  return ball1
end
```

A.5 MAYA でのカメラ位置，アニメーション，ライト等を設定

A.5.1 camera

```
$camera_count = 1

def camera_aim(file1,camera,aim,xyz,aim_xyz)
  c = $camera_count
  file1.printf("camera -centerOfInterest 5 -focalLength 35
 -lensSqueezeRatio 1 -cameraScale 1 -horizontalFilmAperture
 1.4173 -horizontalFilmOffset 0 -verticalFilmAperture 0.9449
-verticalFilmOffset 0 -filmFit Fill -overscan 1 -motionBlur
 0 -shutterAngle 144 -nearClipPlane 0.01 -farClipPlane 1000
-orthographic 0 -orthographicWidth 30;n")
  file1.printf("objectMoveCommand;n")
  file1.printf("cameraMakeNode 2 "";n")
  file1.printf("setAttr "camera%d.translateX" %7.5f;n",c,xyz[0])
  file1.printf("setAttr "camera%d.translateY" %7.5f;n",c,xyz[1])
  file1.printf("setAttr "camera%d.translateZ" %7.5f;n",c,xyz[2])
  file1.printf("setAttr "camera%d_aim.translateX" %7.5f;n"
,c,aim_xyz[0])
  file1.printf("setAttr "camera%d_aim.translateY" %7.5f;n"
,c,aim_xyz[1])
  file1.printf("setAttr "camera%d_aim.translateZ" %7.5f;n"
,c,aim_xyz[2])
  file1.printf("select -r  camera%d;n",c)
  file1.printf("string $%s[] = `ls -sl`;n",camera)
  file1.printf("select -r  camera%d_aim;n",c)
  file1.printf("string $%s[] = `ls -sl`;n",aim)
  $camera_count += 1
  return file1
end
```

A.5.2 light

```
$light_count = 1

def ambientLight(file1,tmp,xyz,rxyz,sxyz)
  c = $light_count
  file1.printf("defaultAmbientLight(1,1,1,1,"0",0,0,0,0);n")
  file1.printf("setAttr "ambientLight%d.translateX" %7.5f;n",c,xyz[0])
  file1.printf("setAttr "ambientLight%d.translateY" %7.5f;n",c,xyz[1])
  file1.printf("setAttr "ambientLight%d.translateZ" %7.5f;n",c,xyz[2])
  file1.printf("setAttr "ambientLight%d.rotateX" %7.5f;n",c,rxyz[0])
  file1.printf("setAttr "ambientLight%d.rotateY" %7.5f;n",c,rxyz[1])
  file1.printf("setAttr "ambientLight%d.rotateZ" %7.5f;n",c,rxyz[2])
  file1.printf("setAttr "ambientLight%d.scaleX" %7.5f;n",c,sxyz[0])
  file1.printf("setAttr "ambientLight%d.scaleY" %7.5f;n",c,sxyz[1])
  file1.printf("setAttr "ambientLight%d.scaleZ" %7.5f;n",c,sxyz[2])
  file1.printf("setAttr "ambientLightShape%d.intensity" %1.5f;n",c,tmp)
  $light_count += 1
  return file1
end
def derectionalLight(file1,tmp,x,y,z,rx,ry,rz,sx,sy,sz)
  c = $light_count
  file1.printf("defaultDirectionalLight(1,1,1,1,"0",0,0,0,0);n")
  file1.printf("setAttr "directionalLight%d.translateX" %7.5f;n",
c,xyz[0])
  file1.printf("setAttr "directionalLight%d.translateY" %7.5f;n",
c,xyz[1])
  file1.printf("setAttr "directionalLight%d.translateZ" %7.5f;n",
c,xyz[2])
  file1.printf("setAttr "directionalLight%d.rotateX" %7.5f;n",c,
rxyz[0])
  file1.printf("setAttr "directionalLight%d.rotateY" %7.5f;n",
c,rxyz[1])
  file1.printf("setAttr "directionalLight%d.rotateZ" %7.5f;n",
c,rxyz[2])
  file1.printf("setAttr "directionalLight%d.scaleX" %7.5f;",c,sxyz[0])
```

```
  file1.printf("setAttr "directionalLight%d.scaleY" %7.5f;",c,sxyz[1])
  file1.printf("setAttr "directionalLight%d.scaleZ" %7.5f;",c,sxyz[2])
  file1.printf("setAttr "directionalLightShape%d.intensity"
%1.5f;n",c,tmp)
  $light_count += 1
  return file1
end
def pointLight(file1,tmp,x,y,z,rx,ry,rz,sx,sy,sz)
  c = $light_count
  file1.printf("defaultPointLight(1,1,1,1,"0",0,0,0,0);n")
  file1.printf("setAttr "pointLight%d.translateX" %7.5f;n",c,x)
  file1.printf("setAttr "pointLight%d.translateY" %7.5f;n",c,y)
  file1.printf("setAttr "pointLight%d.translateZ" %7.5f;n",c,z)
  file1.printf("setAttr "pointLight%d.rotateX" %7.5f;n",c,rx)
  file1.printf("setAttr "pointLight%d.rotateY" %7.5f;n",c,ry)
  file1.printf("setAttr "pointLight%d.rotateZ" %7.5f;n",c,rz)
  file1.printf("setAttr "pointLight%d.scaleX" %7.5f;n",c,sx)
  file1.printf("setAttr "pointLight%d.scaleY" %7.5f;n",c,sy)
  file1.printf("setAttr "pointLight%d.scaleZ" %7.5f;n",c,sz)
  file1.printf("setAttr "pointLightShape%d.intensity" %1.5f;n",c,tmp)
  $light_count += 1
  return file1
end
def spotLight(file1,tmp,x,y,z,rx,ry,rz,sx,sy,sz)
  c = $light_count
  file1.printf("defaultSpotLight(1,1,1,1,"0",0,0,0,0);n")
  file1.printf("setAttr "spotLight%d.translateX" %7.5f;n",c,xyz[0])
  file1.printf("setAttr "spotLight%d.translateY" %7.5f;n",c,xyz[1])
  file1.printf("setAttr "spotLight%d.translateZ" %7.5f;n",c,xyz[2])
  file1.printf("setAttr "spotLight%d.rotateX" %7.5f;n",c,rxyz[0])
  file1.printf("setAttr "spotLight%d.rotateY" %7.5f;n",c,rxyz[1])
  file1.printf("setAttr "spotLight%d.rotateZ" %7.5f;n",c,rxyz[2])
  file1.printf("setAttr "spotLight%d.scaleX" %7.5f;n",c,sxyz[0])
  file1.printf("setAttr "spotLight%d.scaleY" %7.5f;n",c,sxyz[1])
  file1.printf("setAttr "spotLight%d.scaleZ" %7.5f;n",c,sxyz[2])
```

```
  file1.printf("setAttr "spotLightShape%d.intensity" %1.5f;n",c,tmp)
  $light_count += 1
  return file1
end
def areaLight(file1,tmp,x,y,z,rx,ry,rz,sx,sy,sz)
  c = $light_count
  file1.printf("defaultAreaLight(1,1,1,1,"0",0,0,0,0);n")
  file1.printf("setAttr "areaLight%d.translateX" %7.5f;n",c,xyz[0])
  file1.printf("setAttr "areaLight%d.translateY" %7.5f;n",c,xyz[1])
  file1.printf("setAttr "areaLight%d.translateZ" %7.5f;n",c,xyz[2])
  file1.printf("setAttr "areaLight%d.rotateX" %7.5f;n",c,rxyz[0])
  file1.printf("setAttr "areaLight%d.rotateY" %7.5f;n",c,rxyz[1])
  file1.printf("setAttr "areaLight%d.rotateZ" %7.5f;n",c,rxyz[2])
  file1.printf("setAttr "areaLight%d.scaleX" %7.5f;n",c,sxyz[0])
  file1.printf("setAttr "areaLight%d.scaleY" %7.5f;n",c,sxyz[1])
  file1.printf("setAttr "areaLight%d.scaleZ" %7.5f;n",c,sxyz[2])
  file1.printf("setAttr "areaLightShape%d.intensity" %1.5f;n",c,tmp)
  $light_count += 1
  return file1
end
def volumeLight(file1,tmp,x,y,z,rx,ry,rz,sx,sy,sz)
  c = $light_count
  file1.printf("defaultVolumeLight(1,1,1,1,"0",0,0,0,0);n")
  file1.printf("setAttr "volumeLight%d.translateX" %7.5f;n",c,xyz[0])
  file1.printf("setAttr "volumeLight%d.translateY" %7.5f;n",c,xyz[1])
  file1.printf("setAttr "volumeLight%d.translateZ" %7.5f;n",c,xyz[2])
  file1.printf("setAttr "volumeLight%d.rotateX" %7.5f;n",c,rxyz[0])
  file1.printf("setAttr "volumeLight%d.rotateY" %7.5f;n",c,rxyz[1])
  file1.printf("setAttr "volumeLight%d.rotateZ" %7.5f;n",c,rxyz[2])
  file1.printf("setAttr "volumeLight%d.scaleX" %7.5f;n",c,sxyz[0])
  file1.printf("setAttr "volumeLight%d.scaleY" %7.5f;n",c,sxyz[1])
  file1.printf("setAttr "volumeLight%d.scaleZ" %7.5f;n",c,sxyz[2])
  file1.printf("setAttr "volumeLightShape%d.intensity" %1.5f;n",c,tmp)
  $light_count += 1
  return file1
```

```
end
```

A.5.3 currentTime

```
$currentTime_move_count = 1
def currentTime_move(file1,time,name,xyz)
  if($currentTime_move_count==1)
    file1.printf("global proc make_current_
move(string $names[],int $c_time,float $tx,float $ty,float $tz)n")
    file1.printf("{n")
    file1.printf("currentTime $c_time;n")
    file1.printf("setAttr ($names[0] + ".tx") $tx;n")
    file1.printf("setAttr ($names[0] + ".ty") $ty;n")
    file1.printf("setAttr ($names[0] + ".tz") $tz;n")
    file1.printf("setKeyframe -at "tx" $names[0];n")
    file1.printf("setKeyframe -at "ty" $names[0];n")
    file1.printf("setKeyframe -at "tz" $names[0];n")
    file1.printf("}n")
  end
  file1.printf("make_current_move($%s,%d,%7.5f,%7.5f,%7.5f);n"
,name,time,xyz[0],xyz[1],xyz[2])
  $currentTime_move_count += 1
  return file1
end

$currentTime_rotate_count = 1
def currentTime_rotate(file1,time,name,xyz)
  if($currentTime_rotate_count==1)
    file1.printf("global proc make_current_rotate(string $names[],
int $c_time,float $tx,float $ty,float $tz)n")
    file1.printf("{n")
    file1.printf("currentTime $c_time;n")
    file1.printf("setAttr ($names[0] + ".rx") $tx;n")
    file1.printf("setAttr ($names[0] + ".ry") $ty;n")
```

```
    file1.printf("setAttr ($names[0] + ".rz") $tz;n")
    file1.printf("setKeyframe -at "rx" $names[0];n")
    file1.printf("setKeyframe -at "ry" $names[0];n")
    file1.printf("setKeyframe -at "rz" $names[0];n")
    file1.printf("}n")
  end
  file1.printf("make_current_rotate($%s,%d,%7.5f,%7.5f,%7.5f);n",
name,time,xyz[0],xyz[1],xyz[2])
  $currentTime_rotate_count += 1
  return file1
end

$currentTime_scale_count = 1
def currentTime_scale(file1,time,name,xyz)
  if($currentTime_scale_count==1)
    file1.printf("global proc make_current_scale(string $names[],
int $c_time,float $tx,float $ty,float $tz)n")
    file1.printf("{n")
    file1.printf("currentTime $c_time;n")
    file1.printf("setAttr ($names[0] + ".sx") $tx;n")
    file1.printf("setAttr ($names[0] + ".sy") $ty;n")
    file1.printf("setAttr ($names[0] + ".sz") $tz;n")
    file1.printf("setKeyframe -at "sx" $names[0];n")
    file1.printf("setKeyframe -at "sy" $names[0];n")
    file1.printf("setKeyframe -at "sz" $names[0];n")
    file1.printf("}n")
  end
  file1.printf("make_current_scale($%s,%d,%7.5f,%7.5f,%7.5f);n",
name,time,xyz[0],xyz[1],xyz[2])
  $currentTime_scale_count += 1
  return file1
end

def currentTime_color_trans(file1,time,color,xyz)
  file1.printf("currentTime %d;n",time)
```

```
  file1.printf("select -r lambert%d;n",color)
  file1.printf("setAttr "lambert%d.transparency" -type double3 ",color)
  file1.printf(" %7.5f %7.5f %7.5f;n",xyz[0],xyz[1],xyz[2])
  file1.printf("setKeyframe -breakdown 0 -hierarchy none
-controlPoints 0 -shape 0 {"lambert%d"};n",color)
  return file1
end
```

A.5.4 object

```
$set_object_cube_count = 1
def set_object_cube(file1,color,move,scale,rotate)
  c = $set_object_cube_count
  if(c==1)
    file1.printf("global proc make_cube")
    file1.printf("(float $mx,float $my,float $mz,float $sx,
float $sy,float $sz,float $rx,float $ry,float $rz)n")
    file1.printf("{n")
    file1.printf("nurbsCube;n")
    file1.printf("scale $sx $sy $sz;n")
    file1.printf("move $mx $my $mz;n")
    file1.printf("rotate $rx $ry $rz;n")
    file1.printf("sets -e -forceElement lambert%dSG;n"
,color)
    file1.printf("}n")
  end
  file1.printf("make_cube(%7.5f,%7.5f,%7.5f,%7.5f,
%7.5f,%7.5f,%7.5f,%7.5f,%7.5f);n",move[0],move[1],move[2],
scale[0],scale[1],scale[2],rotate[0],rotate[1],rotate[2])
  $set_object_cube_count += 1
  return file1
end

def object_add(file1,obj,name,color)
```

```
  file1.printf("string $%s[];n",name)
  file1.printf("$%s = `%s`;n",name,obj)
  file1.printf("sets -e -forceElement lambert%dSG;n",color)

  return file1
end
```

A.5.5 cellsupport

ユニットセルを囲むワイヤーを作るスクリプトである.

```
def wire_frame(file1,p_vec,color)
  file1.printf("global proc make_wireframe(float $x,
float $y,float $z,float $b2,float $a2,float $sa)n")
  file1.printf("{n")
  file1.printf("cylinder;n")
  file1.printf("scale $sa 0.05 0.05;n")
  file1.printf("move $x $y $z;n")
  file1.printf("rotate 0 $b2 $a2;n")
  file1.printf("sets -e -forceElement lambert%dSG;n",color)
  file1.printf("}n")

  toga = [ Vector[0.0, 0.0, 0.0],
         Vector[p_vec[0][0],p_vec[0][1],p_vec[0][2]],
         Vector[p_vec[1][0],p_vec[1][1],p_vec[1][2]],
         Vector[p_vec[2][0],p_vec[2][1],p_vec[2][2]],
         Vector[p_vec[0][0]+p_vec[1][0],
p_vec[0][1]+p_vec[1][1],p_vec[0][2]+p_vec[1][2]],
         Vector[p_vec[0][0]+p_vec[2][0],
p_vec[0][1]+p_vec[2][1],p_vec[0][2]+p_vec[2][2]],
         Vector[p_vec[1][0]+p_vec[2][0],
p_vec[1][1]+p_vec[2][1],p_vec[1][2]+p_vec[2][2]],
         Vector[p_vec[0][0]+p_vec[1][0]+p_vec[2][0],
p_vec[0][1]+p_vec[1][1]+p_vec[2][1],
p_vec[0][2]+p_vec[1][2]+p_vec[2][2]]]
```

```
  co = [[0,1],[0,2],[0,3],[1,4],[1,5],[2,4],
        [2,6],[3,5],[3,6],[4,7],[5,7],[6,7]]
  for i in 0..co.length-1 do
    half_v = (toga[co[i][0]]+toga[co[i][1]])*0.5
    tmp = toga[co[i][0]]-toga[co[i][1]]
    if tmp[1] < 0  then
      dir_vec = tmp*(-1)
    else
      dir_vec=tmp
    end
    x=dir_vec[0]
    y=dir_vec[1]
    z=dir_vec[2]
    aa=1/sqrt(x**2+y**2+z**2)
    x1=x*aa
    y1=y*aa
    z1=z*aa
    b1=-asin(z1)
    a1=acos(x1/cos(b1))
    a2=a1*180/PI
    b2=b1*180/PI
    file1.printf("make_wireframe(%7.5f,%7.5f,%7.5f,%7.5f,
%7.5f,%7.5f);n",half_v[0],half_v[1],half_v[2],b2,a2,(1/aa)/2)
    $cell_cylinder_count += 1
  end
  return file1
end
```

参考文献

[1] 西谷滋人著.「固体物理の基礎」, 森北出版，2006 年.

[2] 由宇義珍著.「はじめてのパワーデバイス」，工業調査会，2006 年１.

[3] 上羽牧夫　責任編集.「結晶成長のしくみを探る　その物理的基礎」共立出版株式会社 2002 年.

[4] 浅岡康著．関西学院大学 博士論文,「極薄液層環境を用いた単結晶 SiC 薄膜成長に関する研究」2004 年 1 月.

[5] 西谷滋人，金子忠昭．計算工学講演会論文集　vol.12，Simulations on Metastable Solvent Epitaxy of SiC　2007 年 5 月.

[6] 宮澤信太郎　責任編集.「メルト成長のダイナミクス」共立出版株式会社 2002 年.

[7] H・メーラー協力　Th ホイマン著　藤川　辰一郎訳.「金属における拡散」springer 2005 年.

[8] Ajit Ram Verma,P.Krishna. Polymorphism and polytytypism in crystals 1996 年