第 13 章　Phase-field 法３（結晶変態、ドメイン形成）

本章では立方晶から正方晶への結晶変態に関する Phase-field シミュレーション[1,2]について説明する。これは、マルテンサイト変態によって双晶ドメイン組織が形成される場合の計算に対応している。

13-1 全自由エネルギー式の評価

(1) 化学的自由エネルギー

単相（立方晶）から単相（正方晶）への結晶変態を考えるので、ここではランダウ理論式のマルテンサイト変態の化学的自由エネルギー関数を用いる。立方晶から正方晶が形成される場合、バリアント数は３であるので、自由エネルギー関数は、たとえば、

$$f(s_1,s_2,s_3)=\Delta f\left\{\frac{a}{2}(s_1^2+s_2^2+s_3^2)-\frac{b}{3}(s_1^3+s_2^3+s_3^3)+\frac{c}{4}\left(s_1^2+s_2^2+s_3^2\right)^2\right\} \tag{1}$$

と表現される。ただし s_i の変域は $0\le s_i\le 1$ である。$s_1=s_2=s_3=0$ で $f(0)=0$、および $s_1=1,s_2=s_3=0$ で $f(1)=-\Delta f$ である。これより、

$$\begin{aligned}
\frac{df}{ds_1}&=\Delta f\left\{as_1-bs_1^2+c(s_1^2+s_2^2+s_3^2)s_1\right\}=\Delta f\,s_1\left\{a-bs_1+c(s_1^2+s_2^2+s_3^2)\right\}\\
\frac{df}{ds_2}&=\Delta f\,s_2\left\{a-bs_2+c(s_1^2+s_2^2+s_3^2)\right\}\\
\frac{df}{ds_3}&=\Delta f\,s_3\left\{a-bs_3+c(s_1^2+s_2^2+s_3^2)\right\}
\end{aligned} \tag{2}$$

となる。

(2) 弾性歪エネルギー

弾性歪エネルギーは、非等方弾性体として立方対称を仮定し、マイクロメカニクスに基づき算出する。弾性歪エネルギーの計算に必要な定数は、弾性定数と格子定数であり、これらを、C_{11},C_{12},C_{44} a^c,a^t,c^t と置く。２次元計算では、バリアント p の格子ミスマッチ $\varepsilon_{ij}^{00}(p)$ は、

$$\begin{aligned}
&\varepsilon_{11}^{00}(1)=\frac{c^t-a^t}{a^c},\quad \varepsilon_{22}^{00}(1)=-\varepsilon_{11}^{00}(1)\\
&\varepsilon_{11}^{00}(2)=-\varepsilon_{11}^{00}(1),\quad \varepsilon_{22}^{00}(2)=-\varepsilon_{11}^{00}(2)
\end{aligned} \tag{3}$$

と定義される。これより規則度 s_i に起因する eigen 歪は、

$$\varepsilon_{ij}^{0}(\mathbf{r},t)=\varepsilon_{ij}^{00}(1)s_1(\mathbf{r},t)+\varepsilon_{ij}^{00}(2)s_2(\mathbf{r},t) \tag{4}$$

と表現される（s_i の変域が $0\le s_i\le 1$ であるので、s_i が２乗されていない点に注意）。また物体表面は自由表面で外部応力 σ_{ij}^{A} 以外の外部拘束は存在しないと仮定する。

さて一定外力 σ_{ij}^{A} の作用下における弾性歪エネルギーの基礎式は、通常の弾性歪エネルギーに外力のポテンシャルエネルギーを加えて、

$$
\begin{aligned}
E_{str} &= \frac{1}{2V}\int_{\mathbf{r}} C_{ijkl}\varepsilon_{ij}^{el}(\mathbf{r})\varepsilon_{kl}^{el}(\mathbf{r})d\mathbf{r} - \sigma_{ij}^{A}\frac{1}{V}\int_{\mathbf{r}}\varepsilon_{ij}^{c}(\mathbf{r})d\mathbf{r} \\
&= \frac{1}{2V}\int_{\mathbf{r}} C_{ijkl}\{\overline{\varepsilon}_{ij}^{c} + \delta\varepsilon_{ij}^{c}(\mathbf{r}) - \varepsilon_{ij}^{0}(\mathbf{r})\}\{\overline{\varepsilon}_{kl}^{c} + \delta\varepsilon_{kl}^{c}(\mathbf{r}) - \varepsilon_{kl}^{0}(\mathbf{r})\}d\mathbf{r} - \sigma_{ij}^{A}\frac{1}{V}\int_{\mathbf{r}}\varepsilon_{ij}^{c}(\mathbf{r})d\mathbf{r} \\
&= \frac{1}{2V}\int_{\mathbf{r}} C_{ijkl}\{\overline{\varepsilon}_{ij}^{c} + \delta\varepsilon_{ij}^{c}(\mathbf{r}) - \varepsilon_{ij}^{0}(\mathbf{r})\}\{\overline{\varepsilon}_{kl}^{c} + \delta\varepsilon_{kl}^{c}(\mathbf{r}) - \varepsilon_{kl}^{0}(\mathbf{r})\}d\mathbf{r} - \sigma_{ij}^{A}\overline{\varepsilon}_{ij}^{c}
\end{aligned}
\tag{5}
$$

と表現できる（この式には外力に起因するポテンシャルエネルギーも含まれているので、正確には弾性歪エネルギーではなく、Gibbs エネルギーを記すべきであるが、別項の化学的自由エネルギーとの混乱を避けるために、ここでは弾性歪エネルギーと記す）。

物体の境界条件として、物体表面が拘束されていない場合を想定すると、力学的定常状態では、均一拘束歪は、

$$
\frac{\partial E_{str}}{\partial \overline{\varepsilon}_{ij}^{c}} = 0 \tag{6}
$$

が成立するように決定される。

$$
\begin{aligned}
&\frac{\partial E_{str}}{\partial \overline{\varepsilon}_{ij}^{c}} = \frac{1}{V}\int_{\mathbf{r}} C_{ijkl}\{\overline{\varepsilon}_{kl}^{c} + \delta\varepsilon_{kl}^{c}(\mathbf{r}) - \varepsilon_{kl}^{0}(\mathbf{r})\}d\mathbf{r} - \sigma_{ij}^{A} = 0 \\
&\qquad C_{ijkl}\overline{\varepsilon}_{kl}^{c} - \int_{\mathbf{r}} C_{ijkl}\varepsilon_{kl}^{0}(\mathbf{r})d\mathbf{r} = \sigma_{ij}^{A} \\
&\therefore\ \overline{\varepsilon}_{kl}^{c} = C_{ijkl}^{-1}\sigma_{ij}^{A} + C_{ijkl}^{-1}\int_{\mathbf{r}} C_{ijkl}\varepsilon_{kl}^{0}(\mathbf{r})d\mathbf{r} = C_{ijkl}^{-1}\sigma_{ij}^{A} + \overline{\varepsilon}_{kl}^{0}
\end{aligned}
\tag{7}
$$

なお弾性コンプライアンス C_{ijkl}^{-1} は、立方晶の場合、

$$
C_{11}^{-1} = \frac{C_{11}+C_{12}}{(C_{11}-C_{12})(C_{11}+2C_{12})},\quad C_{12}^{-1} = \frac{-C_{12}}{(C_{11}-C_{12})(C_{11}+2C_{12})},\quad C_{44}^{-1} = \frac{1}{C_{44}}
$$

となる（以下の計算では等方弾性体を仮定しているので注意）。また eigen 歪の空間平均値 $\overline{\varepsilon}_{kl}^{0}$ は、

$$
\overline{\varepsilon}_{kl}^{0} = \int_{\mathbf{r}}\varepsilon_{kl}^{0}(\mathbf{r})d\mathbf{r} \tag{8}
$$

である。

さてポテンシャルを算出しよう。ポテンシャルは、式(5)より、

$$
\chi_{str}^{(p)}(\mathbf{r}) = \frac{\partial E_{str}}{\partial s_p(,\mathbf{r})} = \frac{\partial E_{str}}{\partial \varepsilon_{ij}^{0}(\mathbf{r})}\frac{\partial \varepsilon_{ij}^{0}(\mathbf{r})}{\partial s_p(\mathbf{r})} = -C_{ijkl}(\mathbf{r})\{\overline{\varepsilon}_{kl}^{c} + \delta\varepsilon_{kl}^{c}(\mathbf{r}) - \varepsilon_{kl}^{0}(\mathbf{r})\}\frac{\partial \varepsilon_{ij}^{0}(\mathbf{r})}{\partial s_p(\mathbf{r})} \tag{9}
$$

にて与えられる。実空間表示の eigen 歪より、

$$
\frac{\partial \varepsilon_{ij}^{0}(\mathbf{r})}{\partial s_p(\mathbf{r})} = \varepsilon_{ij}^{00}(p) \tag{10}
$$

である。したがってポテンシャルは式(7)も考慮して、２次元計算（$\varepsilon_{33}^{00} = 0$）では、

$$
\begin{aligned}
\chi_{str}^{(p)}(\mathbf{r}) &= -C_{ijkl}\{\bar{\varepsilon}_{kl}^{c} + \delta\varepsilon_{kl}^{c}(\mathbf{r}) - \varepsilon_{kl}^{0}(\mathbf{r})\}\varepsilon_{ij}^{00}(p) \\
&= -C_{ijkl}\{\bar{C}_{ijkl}^{-1}\sigma_{ij}^{A} + \bar{\varepsilon}_{kl}^{0} + \delta\varepsilon_{kl}^{c}(\mathbf{r}) - \varepsilon_{kl}^{0}(\mathbf{r})\}\varepsilon_{ij}^{00}(p) \\
&= \left[C_{ijkl}\left\{\varepsilon_{kl}^{0}(\mathbf{r}) - \bar{\varepsilon}_{kl}^{0}\right\} - C_{ijkl}\delta\varepsilon_{kl}^{c}(\mathbf{r}) - C_{ijkl}\bar{C}_{ijkl}^{-1}\sigma_{ij}^{A} \right]\varepsilon_{ij}^{00}(p) \\
&= C_{ijkl}\left[\varepsilon_{kl}^{0}(\mathbf{r}) - \bar{\varepsilon}_{kl}^{0} - \delta\varepsilon_{kl}^{c}(\mathbf{r}) - \varepsilon_{kl}^{A} \right]\varepsilon_{ij}^{00}(p)
\end{aligned}
\tag{11}
$$

となり（$\varepsilon_{kl}^{A} \equiv \bar{C}_{ijkl}^{-1}\sigma_{ij}^{A}$ と定義)、等方体では、

$$
C_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})
$$

であるので、

$$
\chi_{str}^{(p)}(\mathbf{r}) = \left\{\lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\right\}\left[\varepsilon_{kl}^{0}(\mathbf{r}) - \bar{\varepsilon}_{kl}^{0} - \delta\varepsilon_{kl}^{c}(\mathbf{r}) - \varepsilon_{kl}^{A} \right]\varepsilon_{ij}^{00}(p)
$$

$$
\begin{aligned}
\chi_{str}^{(p)}(\mathbf{r}) &= \left\{\lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\right\}\left[\varepsilon_{kl}^{0}(\mathbf{r}) - \bar{\varepsilon}_{kl}^{0} - \delta\varepsilon_{kl}^{c}(\mathbf{r}) - \varepsilon_{kl}^{A} \right]\varepsilon_{ij}^{00}(p) \\
&= (\lambda + 2\mu)\left\{\left[\varepsilon_{11}^{0}(\mathbf{r}) - \bar{\varepsilon}_{11}^{0} - \delta\varepsilon_{11}^{c}(\mathbf{r}) - \varepsilon_{11}^{A} \right]\varepsilon_{11}^{00}(p) + \left[\varepsilon_{22}^{0}(\mathbf{r}) - \bar{\varepsilon}_{22}^{0} - \delta\varepsilon_{22}^{c}(\mathbf{r}) - \varepsilon_{22}^{A} \right]\varepsilon_{22}^{00}(p)\right\} \\
&\quad + \lambda\left[\varepsilon_{22}^{0}(\mathbf{r}) - \bar{\varepsilon}_{22}^{0} - \delta\varepsilon_{22}^{c}(\mathbf{r}) - \varepsilon_{22}^{A} \right]\varepsilon_{11}^{00}(p) + \lambda\left[\varepsilon_{11}^{0}(\mathbf{r}) - \bar{\varepsilon}_{11}^{0} - \delta\varepsilon_{11}^{c}(\mathbf{r}) - \varepsilon_{11}^{A} \right]\varepsilon_{22}^{00}(p) \\
&= \lambda\left[\{\varepsilon_{11}^{0}(\mathbf{r}) - \bar{\varepsilon}_{11}^{0} - \delta\varepsilon_{11}^{c}(\mathbf{r}) - \varepsilon_{11}^{A}\} + \{\varepsilon_{22}^{0}(\mathbf{r}) - \bar{\varepsilon}_{22}^{0} - \delta\varepsilon_{22}^{c}(\mathbf{r}) - \varepsilon_{22}^{A}\} \right]\{\varepsilon_{11}^{00}(p) + \varepsilon_{22}^{00}(p)\} \\
&\quad + 2\mu\left\{\left[\varepsilon_{11}^{0}(\mathbf{r}) - \bar{\varepsilon}_{11}^{0} - \delta\varepsilon_{11}^{c}(\mathbf{r}) - \varepsilon_{11}^{A} \right]\varepsilon_{11}^{00}(p) + \left[\varepsilon_{22}^{0}(\mathbf{r}) - \bar{\varepsilon}_{22}^{0} - \delta\varepsilon_{22}^{c}(\mathbf{r}) - \varepsilon_{22}^{A} \right]\varepsilon_{22}^{00}(p)\right\}
\end{aligned}
$$

$$
\begin{aligned}
\therefore\ \chi_{str}^{(1)}(\mathbf{r}) &= \lambda\left[\{\varepsilon_{11}^{0}(\mathbf{r}) - \bar{\varepsilon}_{11}^{0} - \delta\varepsilon_{11}^{c}(\mathbf{r}) - \varepsilon_{11}^{A}\} + \{\varepsilon_{22}^{0}(\mathbf{r}) - \bar{\varepsilon}_{22}^{0} - \delta\varepsilon_{22}^{c}(\mathbf{r}) - \varepsilon_{22}^{A}\} \right]\{\varepsilon_{11}^{00}(1) + \varepsilon_{22}^{00}(1)\} \\
&\quad + 2\mu\left\{\left[\varepsilon_{11}^{0}(\mathbf{r}) - \bar{\varepsilon}_{11}^{0} - \delta\varepsilon_{11}^{c}(\mathbf{r}) - \varepsilon_{11}^{A} \right]\varepsilon_{11}^{00}(1) + \left[\varepsilon_{22}^{0}(\mathbf{r}) - \bar{\varepsilon}_{22}^{0} - \delta\varepsilon_{22}^{c}(\mathbf{r}) - \varepsilon_{22}^{A} \right]\varepsilon_{22}^{00}(1)\right\} \\
\chi_{str}^{(2)}(\mathbf{r}) &= \lambda\left[\{\varepsilon_{11}^{0}(\mathbf{r}) - \bar{\varepsilon}_{11}^{0} - \delta\varepsilon_{11}^{c}(\mathbf{r}) - \varepsilon_{11}^{A}\} + \{\varepsilon_{22}^{0}(\mathbf{r}) - \bar{\varepsilon}_{22}^{0} - \delta\varepsilon_{22}^{c}(\mathbf{r}) - \varepsilon_{22}^{A}\} \right]\{\varepsilon_{11}^{00}(2) + \varepsilon_{22}^{00}(2)\} \\
&\quad + 2\mu\left\{\left[\varepsilon_{11}^{0}(\mathbf{r}) - \bar{\varepsilon}_{11}^{0} - \delta\varepsilon_{11}^{c}(\mathbf{r}) - \varepsilon_{11}^{A} \right]\varepsilon_{11}^{00}(2) + \left[\varepsilon_{22}^{0}(\mathbf{r}) - \bar{\varepsilon}_{22}^{0} - \delta\varepsilon_{22}^{c}(\mathbf{r}) - \varepsilon_{22}^{A} \right]\varepsilon_{22}^{00}(2)\right\}
\end{aligned}
$$

を得る。

次に $\delta\varepsilon_{ij}^{c}(\mathbf{r},t)$ の計算について説明する。まず秩序変数 $s_p(\mathbf{r})$ のフーリエ変換と逆変換を、

$$
\tilde{s}_p(\mathbf{k},t) = \frac{1}{V}\int_{\mathbf{r}} s_p(\mathbf{r},t)\exp(-i\mathbf{kr})d\mathbf{r}
$$

$$
s_p(\mathbf{r},t) = \int_{\mathbf{k}} \tilde{s}_p(\mathbf{k},t)\exp(i\mathbf{kr})\frac{d\mathbf{k}}{(2\pi)^3}
$$

と置き、eigen 応力 $\sigma_{ij}^{0}(\mathbf{r})$ を、

$$
\begin{aligned}
&\sigma_{ij}^0(\mathbf{r}) = C_{ijkl}\varepsilon_{kl}^0(\mathbf{r}) = \left\{\lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\right\}\varepsilon_{kl}^0(\mathbf{r}) \\
&\sigma_{ij}^0(\mathbf{r}) = \left\{\lambda\delta_{ij}\delta_{11} + \mu(\delta_{i1}\delta_{j1} + \delta_{i1}\delta_{j1})\right\}\varepsilon_{11}^0(\mathbf{r}) + \left\{\lambda\delta_{ij}\delta_{22} + \mu(\delta_{i2}\delta_{j2} + \delta_{i2}\delta_{j2})\right\}\varepsilon_{22}^0(\mathbf{r}) \\
&\therefore\ \sigma_{11}^0(\mathbf{r}) = (\lambda+2\mu)\varepsilon_{11}^0(\mathbf{r}) + \lambda\varepsilon_{22}^0(\mathbf{r}) \\
&\quad \sigma_{22}^0(\mathbf{r}) = \lambda\varepsilon_{11}^0(\mathbf{r}) + (\lambda+2\mu)\varepsilon_{22}^0(\mathbf{r}) \\
&\quad \sigma_{33}^0(\mathbf{r}) = \lambda\{\varepsilon_{11}^0(\mathbf{r}) + \varepsilon_{22}^0(\mathbf{r})\} \\
&\quad \sigma_{12}^0(\mathbf{r}) = \sigma_{21}^0(\mathbf{r}) = \sigma_{13}^0(\mathbf{r}) = \sigma_{31}^0(\mathbf{r}) = \sigma_{23}^0(\mathbf{r}) = \sigma_{32}^0(\mathbf{r}) = 0
\end{aligned}
\tag{12}
$$

にて定義する。λ と μ はラーメの定数である（等方弾性体を仮定）。$\sigma_{ij}^0(\mathbf{r})$ と $\varepsilon_{kl}^0(\mathbf{r})$ は線形結合であるから、$\sigma_{ij}^0(\mathbf{r})$ のフーリエ変換は、

$$
\tilde{\sigma}_{ij}^0(\mathbf{k}) = \frac{1}{V}\int_{\mathbf{r}}\sigma_{ij}^0(\mathbf{r})\exp(-i\mathbf{kr})d\mathbf{r} = \frac{1}{V}\int_{\mathbf{r}}C_{ijkl}\varepsilon_{kl}^0(\mathbf{r})\exp(-i\mathbf{kr})d\mathbf{r} \tag{13}
$$

$$
\begin{aligned}
&\tilde{\sigma}_{11}^0(\mathbf{k}) = (\lambda+2\mu)\tilde{\varepsilon}_{11}^0(\mathbf{k}) + \lambda\tilde{\varepsilon}_{22}^0(\mathbf{k}) \\
&\tilde{\sigma}_{22}^0(\mathbf{k}) = \lambda\tilde{\varepsilon}_{11}^0(\mathbf{k}) + (\lambda+2\mu)\tilde{\varepsilon}_{22}^0(\mathbf{k}) \\
&\tilde{\sigma}_{33}^0(\mathbf{k}) = \lambda\{\tilde{\varepsilon}_{11}^0(\mathbf{k}) + \tilde{\varepsilon}_{22}^0(\mathbf{k})\} \\
&\tilde{\sigma}_{12}^0(\mathbf{k}) = \tilde{\sigma}_{21}^0(\mathbf{k}) = \tilde{\sigma}_{12}^0(\mathbf{k}) = \tilde{\sigma}_{21}^0(\mathbf{k}) = \tilde{\sigma}_{12}^0(\mathbf{k}) = \tilde{\sigma}_{21}^0(\mathbf{k}) = 0
\end{aligned}
$$

にて与えられる。また $\delta\varepsilon_{kl}^c(\mathbf{r})$ は平衡方程式から、

$$
\begin{aligned}
\delta\varepsilon_{ij}^c(\mathbf{r}) &= \int_{\mathbf{k}}\delta\varepsilon_{ij}^c(\mathbf{k})\exp(i\mathbf{k}\cdot\mathbf{r})\frac{d\mathbf{k}}{(2\pi)^3} = \int_{\mathbf{k}}\left[G_{ik}(\mathbf{k})k_jk_l\tilde{\sigma}_{kl}^0(\mathbf{k})\right]\exp(i\mathbf{k}\cdot\mathbf{r})\frac{d\mathbf{k}}{(2\pi)^3} \\
&= \int_{\mathbf{k}}\left[\Omega_{ik}(\mathbf{n})n_jn_l\tilde{\sigma}_{kl}^0(\mathbf{k})\right]\exp(i\mathbf{k}\cdot\mathbf{r})\frac{d\mathbf{k}}{(2\pi)^3}
\end{aligned}
\tag{14}
$$

であり、等方体では、

$$
\begin{aligned}
\delta\varepsilon_{ij}^c(\mathbf{k}) &= \Omega_{ik}(\mathbf{n})n_jn_l\tilde{\sigma}_{kl}^0(\mathbf{k}) = \left\{\frac{\delta_{ik}}{\mu} - \frac{n_in_k}{2\mu(1-\nu)}\right\}n_jn_l\tilde{\sigma}_{kl}^0(\mathbf{k}) \\
&= \left\{\frac{\delta_{ik}}{\mu}n_jn_l - \frac{n_in_jn_kn_l}{2\mu(1-\nu)}\right\}\tilde{\sigma}_{kl}^0(\mathbf{k}) \\
&= \left\{\frac{\delta_{i1}}{\mu}n_jn_1 - \frac{n_in_jn_1n_1}{2\mu(1-\nu)}\right\}\tilde{\sigma}_{11}^0(\mathbf{k}) + \left\{\frac{\delta_{i2}}{\mu}n_jn_2 - \frac{n_in_jn_2n_2}{2\mu(1-\nu)}\right\}\tilde{\sigma}_{22}^0(\mathbf{k})
\end{aligned}
\tag{15}
$$

$$
\begin{aligned}
&\delta\varepsilon_{11}^c(\mathbf{k}) = \left\{\frac{n_1^2}{\mu} - \frac{n_1^4}{2\mu(1-\nu)}\right\}\tilde{\sigma}_{11}^0(\mathbf{k}) - \frac{n_1^2n_2^2}{2\mu(1-\nu)}\tilde{\sigma}_{22}^0(\mathbf{k}) \\
&\delta\varepsilon_{22}^c(\mathbf{k}) = -\frac{n_1^2n_2^2}{2\mu(1-\nu)}\tilde{\sigma}_{11}^0(\mathbf{k}) + \left\{\frac{n_2^2}{\mu} - \frac{n_2^4}{2\mu(1-\nu)}\right\}\tilde{\sigma}_{22}^0(\mathbf{k}) \\
&\delta\varepsilon_{12}^c(\mathbf{k}) = \left\{\frac{n_1n_2}{\mu} - \frac{n_1^3n_2}{2\mu(1-\nu)}\right\}\tilde{\sigma}_{11}^0(\mathbf{k}) - \frac{n_1n_2^3}{2\mu(1-\nu)}\tilde{\sigma}_{22}^0(\mathbf{k}) \\
&\delta\varepsilon_{21}^c(\mathbf{k}) = -\frac{n_1^3n_2}{2\mu(1-\nu)}\tilde{\sigma}_{11}^0(\mathbf{k}) + \left\{\frac{n_1n_2}{\mu} - \frac{n_1n_2^3}{2\mu(1-\nu)}\right\}\tilde{\sigma}_{22}^0(\mathbf{k})
\end{aligned}
$$

と書き下すことができる。ここで、$\mathbf{n} \equiv \mathbf{k}/|\mathbf{k}|$ および $\Omega_{ik}^{-1}(\mathbf{n}) \equiv C_{ijkl} n_j n_l$ である（８章参照）。

(3) 勾配エネルギー

勾配エネルギーは、通常の Phase-field 法の取り扱いに従い、

$$E_{surf} = \frac{1}{2}\kappa_s \int_{\mathbf{r}} [(\nabla s_1)^2 + (\nabla s_2)^2] d\mathbf{r} \tag{16}$$

を用いて計算する。

13-2 発展方程式と計算条件

この場合の発展方程式は、非保存場の発展方程式になるので、

$$\frac{\partial s_1}{\partial t} = -M_s \frac{\delta G_{sys}}{\delta s_1}, \quad \frac{\partial s_2}{\partial t} = -M_s \frac{\delta G_{sys}}{\delta s_2} \tag{17}$$

を使用すればよい。なお無次元化した方程式は、

$$\frac{\partial s_1}{\partial (t M_s RT)} = -\frac{\delta (G_{sys}/RT)}{\delta s_1}, \qquad \frac{\partial s_2}{\partial (t M_s RT)} = -\frac{\delta (G_{sys}/RT)}{\delta s_2} \tag{18}$$

となる（M_s の次元が $[\mathrm{J \cdot s}]^{-1}$ である点に注意）。エネルギーは RT にて、時間は $M_s RT$ にて無次元化している。

計算条件としては、たとえば、以下を仮定する。

$$T = 1000(\mathrm{K}), \quad \Delta f = 500.0(\mathrm{J/mol}), \quad L = 250(\mathrm{nm})$$
$$a = 0.15, \; b = 3a + 12, \; c = 2a + 12$$
$$\kappa_s = 2.0 \times 10^{-15} (\mathrm{J \cdot m^2/mol})$$
$$C_{11} = 2.508 \times 10^{11} (\mathrm{Pa}), \quad C_{44} = 1.235 \times 10^{11} (\mathrm{Pa}), \quad C_{12} = C_{11} - 2C_{44}$$
$$\varepsilon_{11}^{00}(\mathrm{l}) = 0.02, \quad \sigma_{11}^{A} = \pm 30(\mathrm{MPa})$$

13-3 計算結果

図 13-1 は 1000K における等温結晶変態の時間発展過程の２次元シミュレーションである。図中の数値が無次元化した時間で、黒は立方晶を、濃い灰色と薄い灰色がそれぞれ正方晶場 s_1 と s_2 を表している。正方晶の c 軸は薄い灰色が横方向で、濃い灰色が縦方向である。また正方晶比は１よりも大きい（c 軸の方向に伸びた単位胞）。図は(001)面で、横方向が[100]および縦が[010]方向である。相分解の初期状態(a)は立方晶で、結晶変態の進行に伴い細かな正方晶ドメインが至る所に形成さ

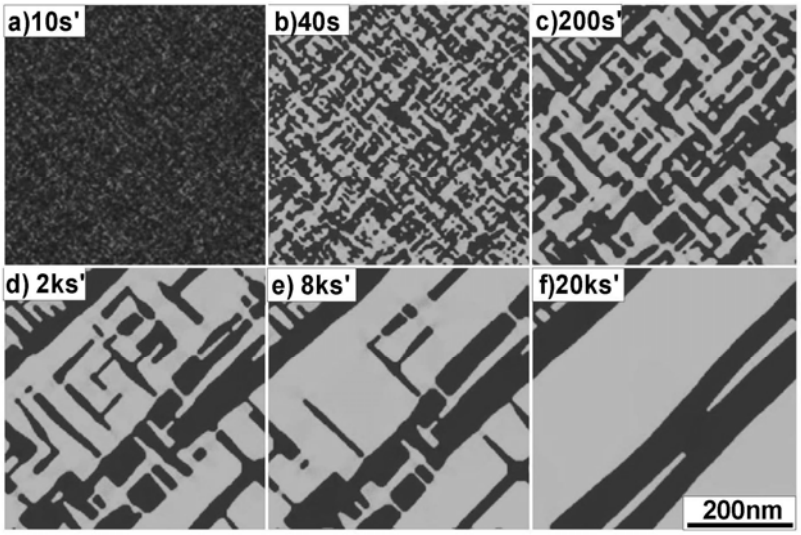

図 13-1　1000K における等温結晶変態の２次元計算結果

れ、45°の方向に周期性を持つツイード組織が形成される(b,c)。その後、正方晶ドメインが合体しながら成長し、後期において 45°方向に傾いたラメラ組織となる（d,e,f)。このラメラ組織の s_1 と s_2 のドメイン境界は、ちょうど双晶界面となる。したがって、初期の細かいツイード組織は、時効に伴い、双晶組織へと変化していくことがわかる。

図 13-2 は、外力によるドメイン成長の違いを計算した結果である。図の上段が外力がかかっていない場合のドメイン成長過程で、図 13-1 の場合に等しい。図の中段は上下方向に引張り応力をかけながらドメインを成長させた場合で、時効の進行に伴い、薄い灰色のドメインが優先的に成長していくことがわかる。これは以下のように理解することが出来る。いま計算条件において正方晶歪は 1 以上の場合を想定しいているので、縦方向に引張り応力を加えた場合、縦方向に c 軸を持つ薄い灰色のドメインの方が濃い灰色のドメインよりも、歪エネルギーは小さく安定である。したがって、この場合、薄い灰色のドメインが成長する。一方、図の下段は縦方向に圧縮応力をかけた場合で、今度は濃い灰色のドメインが優先的に成長していく。圧縮応力では横方向に物体は伸びようとするので、これは、横方向に c 軸を持つ濃い灰色のドメインが安定化した結果生じたと理解することができる。

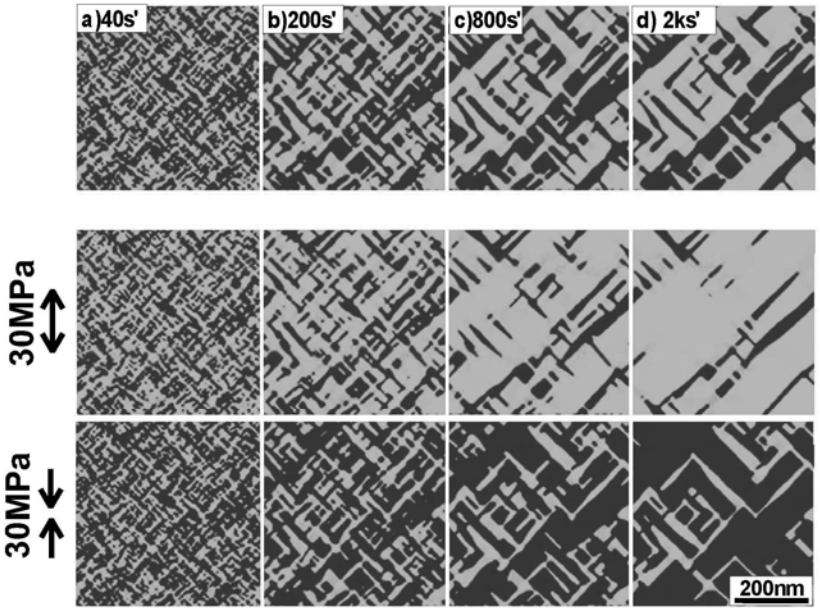

図 13-2　外力によるドメイン成長の相違

以上のような、外部応力におけるドメイン境界の移動現象は、NiTi 合金等の形状記憶効果のメカニズムに対応するので、この計算は、ちょうど形状記憶合金の応力下における内部組織変化過程の計算に対応している。

13-4 まとめ

以上では具体的に計算していないが、計算結果から拘束歪の平均値を求めることが出来るので、周期応力場で、外力に対して拘束歪の平均値をプロットすれば、形状記憶現象における応力-歪曲線を定量的に求めることが出来る。したがって、特にナノおよびメゾスケールにおけるアクチュエータの設計に本計算手法は大いに役立つと考えられる。

参考文献

(1) T.Koyama and H.Onodera: Mater. Trans., **44**(2003), 2503.
(2) Y.M.Jin, A.Artemev and A.G.Khachaturyan: Acta Mater., **49**(2001), 2309.

付録

・Java アプリケーションプログラム（TwinMar2D.java）

```
//*****************************************************************
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class TwinMar2D extends Frame{

   static int ND=128, IG=7;   //組織 1 辺の分割数と log2_(ND)
   static int nd=ND, ndm=ND-1, nd2=ND/2, ig=IG;
```

```
    static int width, height, insetx, insety, xwidth, yheight;
    static double PI=3.14159;      //円周率
    static double RR=8.3145;       //ガス定数
    static double [][] s1h=new double[ND][ND];   //マテリンサイト場のデータ配列
    static double [][] s2h=new double[ND][ND];   //マテリンサイト場のデータ配列

    static double qs;              //フーリエ変換(qs:-1)と逆フーリエ変換(qs:1)の区別
    static double [][] xi=new double[ND][ND];
    static double [][] xr=new double[ND][ND];
    static double [] xif=new double[ND];
    static double [] xrf=new double[ND];   //フーリエ変換の実部・虚部配列
    static double [] s=new double[ND];
    static double [] c=new double[ND];     //sin と cos のテーブル
    static int [] ik=new int[ND];          //ビット反転配列

    //グローバル変数化
    static double time1;       //計算時間
    Image buff;                //組織イメージのバッファ
    static Graphics bg, g;     //組織イメージのグラフィクオブジェクト

    //*************** コンストラクタ ***************************
    public TwinMar2D(){
      xwidth=400; yheight=400; insetx=4; insety=30;
      width=xwidth+insetx*2;   height=yheight+insetx+insety;
      setSize(width, height);
      setBackground(Color.lightGray);
      setVisible(true);
      buff=this.createImage(width, height);
      bg=buff.getGraphics();
      addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent e){ System.exit(0); }
      });
    }

//************** main *****************************
    public static void main(String[] args){

      TwinMar2D prog=new TwinMar2D();

      double s1, s2;         //結晶度
      double [][] ep11h1=new double[ND][ND];
      double [][] ep22h1=new double[ND][ND];
      double [][] ep11qrh1=new double[ND][ND];
      double [][] ep11qih1=new double[ND][ND];
      double [][] ep22qrh1=new double[ND][ND];
      double [][] ep22qih1=new double[ND][ND];
      double [][] s1k_su=new double[ND][ND];
      double [][] s2k_su=new double[ND][ND];
      double s1k_chem, s1k_str;          //ポテンシャル
      double s2k_chem, s2k_str;          //ポテンシャル
      double c11, c22, c33, c44, c55, c66, c12, c13, c23, c21, c31, c32; //弾性定数
      double ram0, mu0, nu0, munu0;          //等方弾性体における弾性定数
      double [][] eta_s1=new double[4][4];
      double [][] eta_s2=new double[4][4];
      double [][] ec11=new double[ND][ND];   //拘束歪配列
```

```
        double [][] ec22=new double[ND][ND];
        double [][] ec12=new double[ND][ND];
        double [][] ec21=new double[ND][ND];
        double ep11T, ep22T;
        double ep11_0, ep22_0;
        double ep11_a, ep22_a, ep12_a, ep21_a;
        double sig11_a, sig22_a;
        double ep11_s1, ep22_s1;
        double ep11_s2, ep22_s2;
        double sig11r, sig22r, sig11i, sig22i;
        double sum11, sum22;
        double s1ddtt, s2ddtt;              //時間変化量
        double el_mag;                      //弾性定数規格化変数

        int     i, j, k, l;                 //整数
        int     ii=0, jj=0, kk=0;           //整数
        int     iii, jjj;                   //整数
        int     p, q, m, n;                 //整数
        int     ip, im, jp, jm, Nstep;      //整数
        double al, temp, delt;              //計算領域(0～2π)、時間、時間きざみ
        double time1max;                    //最大時間（計算を止める際に使用）
        double b1, vm0, atom_n;             //規格化長さ、モル体積、単位胞内の原子数
        double smob1, smob2, smob3;         //結晶変態の易動度
        double nxx, nyy, nxy, alnn;         //逆空間の基本ベクトル、ノルム

        double AA0, AA1, AA2, AA3, AA4, Ac1, Ac2, dG_gp2_gp;      //化学的駆動力定数
        double a1_c, b1_c, c1_c;
        double a1_t, b1_t, c1_t;
        double kapa_s1, kapa_s2;            //結晶度勾配エネルギー定数
        double fact1;

        String s_delt;

//-------------------------------------------------------------------------
        BufferedReader input=new BufferedReader(new InputStreamReader(System.in));

        s_delt="1.0";
        try{ System.out.print("del_time1(1.0) =    ");    s_delt=input.readLine(); }
        catch(IOException e){ System.out.println("Exception : "+e); }
        delt=new Double(s_delt).doubleValue();

//****  各種定数の設定  ***********************************************************
        temp=1000.0;
        al=400.0*1.0E-09;          //計算領域を al[m]にセット
        b1=al/nd;

        time1=0.0;    time1max=10001.0;     //計算時間を 10000 繰返しカウントにセット

        smob1=1.0;
        fact1=1.0;

        AA0=400.0/RR/temp*fact1;
        AA1=0.15;
        AA2=3.0*AA1+12.0;
        AA3=2.0*AA1+12.0;
```

```
      kapa_s1=kapa_s2=5.0e-15/RR/temp/b1/b1;

      a1_c=b1_c=c1_c=3.563E-10;
      //a1_t=b1_t=3.541E-10; c1_t=0.5*7.216E-10;
      atom_n=4.0;    vm0=6.02E23*a1_c*b1_c*c1_c/atom_n;

//*** s1 場の格子ミスマッチの設定 ***
      eta_s1[1][1]=0.02; eta_s1[2][2]=-0.02;
      //eta_s1[1][1]=0.0126; eta_s1[2][2]=-0.0126;
      eta_s1[3][3]=0.0;
      eta_s1[1][2]=eta_s1[2][1]=eta_s1[1][3]=eta_s1[3][1]=eta_s1[2][3]=eta_s1[3][2]=0.0;

//*** s2 場の格子ミスマッチの設定 ***
      eta_s2[1][1]=eta_s1[2][2];
      eta_s2[2][2]=eta_s1[1][1];
      eta_s2[3][3]=0.0;
      eta_s2[1][2]=eta_s2[2][1]=eta_s2[1][3]=eta_s2[3][1]=eta_s2[2][3]=eta_s2[3][2]=0.0;

//***** Ni の弾性定数 ***************************
      el_mag=1.0e+11*vm0/RR/temp;
      c11=c22=c33=2.508*el_mag;
      c44=c55=c66=1.235*el_mag;
      c12=c21=c13=c31=c23=c32=c11-2.0*c44;
      ram0=c12;      mu0=c44;
      nu0=ram0/2.0/(ram0+mu0);
      munu0=2.0*mu0*(1.0-nu0);

//***  外力の設定 ***
      sig22_a=0.0;
      //sig22_a=30.0*1.0e+06*vm0/RR/temp; //30MPa
      ep11_a=-c12/(c11-c12)/(c11+2.*c12)*sig22_a;
      ep22_a=(c11+c12)/(c11-c12)/(c11+2.*c12)*sig22_a;
      ep12_a=ep21_a=0.0;

//*** sin および cos テーブルおよび初期場の設定 ***************
      prog.table();          //sin および cos テーブル
      prog.ini_field();      //乱数によって初期場を設定
      //prog.datin();        //ファイルから初期場を読み込み

//***  相分解の時間発展過程の計算スタート ****************************************
   while(time1<=time1max){

      if((((int)(time1) % 10)==0)){
         System.out.println(time1); prog.repaint();          //所定カウント数おきに場を描画
      }

      //if(time1<=100.){Nstep=5;} else{Nstep=200;}
      //if((((int)(time1) % Nstep)==0)) {prog.datsave();}  //場データの保存
      //if(time1==1000.0){ prog.datsave(); }               //所定カウント数の時に場データを保存

//******[界面ポテンシャルの計算]*****************************************************
      for(i=0;i<=ndm;i++){
         for(j=0;j<=ndm;j++){
            ip=i+1; im=i-1; jp=j+1; jm=j-1;
```

```
            if(i==ndm){ip=0;}    if(i==0){im=ndm;}
            if(j==ndm){jp=0;}    if(j==0){jm=ndm;}
            s1k_su[i][j]=-kapa_s1*(s1h[ip][j]+s1h[im][j]+s1h[i][jp]+s1h[i][jm]-4.0*s1h[i][j]);
            s2k_su[i][j]=-kapa_s2*(s2h[ip][j]+s2h[im][j]+s2h[i][jp]+s2h[i][jm]-4.0*s2h[i][j]);
         }
      }

//**** eigen 歪場のフーリエ変換  ep11 ********************************
      for(i=0;i<=ndm;i++){
         for(j=0;j<=ndm;j++){
            xr[i][j]=ep11h1[i][j]=eta_s1[1][1]*s1h[i][j]+eta_s2[1][1]*s2h[i][j];
            xi[i][j]=0.0;
         }
      }
      qs=-1.0; prog.rcfft();
      for(i=0;i<=ndm;i++){
         for(j=0;j<=ndm;j++){
            ep11qrh1[i][j]=xr[i][j];    ep11qih1[i][j]=xi[i][j];
         }
      }
      ep11qrh1[0][0]=ep11qih1[0][0]=0.;

//**** eigen 歪場のフーリエ変換  ep22 ********************************
      for(i=0;i<=ndm;i++){
         for(j=0;j<=ndm;j++){
            xr[i][j]=ep22h1[i][j]=eta_s1[2][2]*s1h[i][j]+eta_s2[2][2]*s2h[i][j];
            xi[i][j]=0.0;
         }
      }
      qs=-1.0; prog.rcfft();
      for(i=0;i<=ndm;i++){
         for(j=0;j<=ndm;j++){
            ep22qrh1[i][j]=xr[i][j];    ep22qih1[i][j]=xi[i][j];
         }
      }
      ep22qrh1[0][0]=ep22qih1[0][0]=0.;

//*** eigen 歪場の平均値の算出  ***
      sum11=sum22=0.0;
      for(i=0;i<=ndm;i++){
         for(j=0;j<=ndm;j++){
            sum11=sum11+ep11h1[i][j];    sum22=sum22+ep22h1[i][j];
         }
      }
      ep11_0=sum11/nd/nd;    ep22_0=sum22/nd/nd;

//*****  拘束歪 ec11 の計算  ************************************
      for(i=0;i<=ndm;i++){
         if(i<=nd2-1){ii=i;}    if(i>=nd2){ii=i-nd;}
         for(j=0;j<=ndm;j++){
            if(j<=nd2-1){jj=j;}    if(j>=nd2){jj=j-nd;}
               alnn=Math.sqrt((double)ii*(double)ii+(double)jj*(double)jj);
               if(alnn==0.0){alnn=1.0;}
               nxx=(double)ii/alnn*(double)ii/alnn;
               nyy=(double)jj/alnn*(double)jj/alnn;
               nxy=(double)ii/alnn*(double)jj/alnn;
```

```
            sig11r=ram0*(ep11qrh1[i][j]+ep22qrh1[i][j])+2.0*mu0*ep11qrh1[i][j];
            sig22r=ram0*(ep11qrh1[i][j]+ep22qrh1[i][j])+2.0*mu0*ep22qrh1[i][j];
            sig11i=ram0*(ep11qih1[i][j]+ep22qih1[i][j])+2.0*mu0*ep11qih1[i][j];
            sig22i=ram0*(ep11qih1[i][j]+ep22qih1[i][j])+2.0*mu0*ep22qih1[i][j];
            xr[i][j]=nxx*(1.0/mu0-nxx/munu0)*sig11r-nxy*nxy/munu0*sig22r;
            xi[i][j]=nxx*(1.0/mu0-nxx/munu0)*sig11i-nxy*nxy/munu0*sig22i;
        }
    }
    qs=1.0; prog.rcfft();
    for(i=0;i<=ndm;i++){
        for(j=0;j<=ndm;j++){
            ec11[i][j]=xr[i][j];
        }
    }

//*****   拘束歪 ec22 の計算  ****************************
    for(i=0;i<=ndm;i++){
        if(i<=nd2-1){ii=i;}    if(i>=nd2){ii=i-nd;}
        for(j=0;j<=ndm;j++){
            if(j<=nd2-1){jj=j;}    if(j>=nd2){jj=j-nd;}
            alnn=Math.sqrt((double)ii*(double)ii+(double)jj*(double)jj);
            if(alnn==0.0){alnn=1.0;}
            nxx=(double)ii/alnn*(double)ii/alnn;
            nyy=(double)jj/alnn*(double)jj/alnn;
            nxy=(double)ii/alnn*(double)jj/alnn;

            sig11r=ram0*(ep11qrh1[i][j]+ep22qrh1[i][j])+2.0*mu0*ep11qrh1[i][j];
            sig22r=ram0*(ep11qrh1[i][j]+ep22qrh1[i][j])+2.0*mu0*ep22qrh1[i][j];
            sig11i=ram0*(ep11qih1[i][j]+ep22qih1[i][j])+2.0*mu0*ep11qih1[i][j];
            sig22i=ram0*(ep11qih1[i][j]+ep22qih1[i][j])+2.0*mu0*ep22qih1[i][j];
            xr[i][j]=-nxy*nxy/munu0*sig11r+nyy*(1.0/mu0-nyy/munu0)*sig22r;
            xi[i][j]=-nxy*nxy/munu0*sig11i+nyy*(1.0/mu0-nyy/munu0)*sig22i;
        }
    }
    qs=1.0; prog.rcfft();
    for(i=0;i<=ndm;i++){
        for(j=0;j<=ndm;j++){
            ec22[i][j]=xr[i][j];
        }
    }

//******   ポテンシャルの計算  ******************************
    for(i=0;i<=ndm;i++){
        for(j=0;j<=ndm;j++){

        s1=s1h[i][j];       s2=s2h[i][j];

//******   化学ポテンシャルの計算  ******************************
        s1k_chem=AA0*s1*(AA1-AA2*s1+AA3*(s1*s1+s2*s2));
        s2k_chem=AA0*s2*(AA1-AA2*s2+AA3*(s1*s1+s2*s2));

//******   弾性ポテンシャルの計算  ******************************
        ep11_s1=eta_s1[1][1];
        ep22_s1=eta_s1[2][2];
        ep11_s2=eta_s2[1][1];
        ep22_s2=eta_s2[2][2];
```

```
        ep11T=ep11h1[i][j]-ep11_0-ec11[i][j]-ep11_a;
        ep22T=ep22h1[i][j]-ep22_0-ec22[i][j]-ep22_a;

        s1k_str=ram0*(ep11T+ep22T)*(ep11_s1+ep22_s1)+2.0*mu0*(ep11T*ep11_s1+ep22T*ep22_s1);
        s2k_str=ram0*(ep11T+ep22T)*(ep11_s2+ep22_s2)+2.0*mu0*(ep11T*ep11_s2+ep22T*ep22_s2);

//******   結晶場の時間発展の計算 ********************************
        //s1ddtt=-smob1*(s1k_chem+s1k_su[i][j]);
        //s2ddtt=-smob1*(s2k_chem+s2k_su[i][j]);
        s1ddtt=-smob1*(s1k_chem+s1k_su[i][j]+s1k_str);
        s2ddtt=-smob1*(s2k_chem+s2k_su[i][j]+s2k_str);
        s1h[i][j]=s1h[i][j]+s1ddtt*delt+0.001*(2.0*Math.random()-1.0);
        s2h[i][j]=s2h[i][j]+s2ddtt*delt+0.001*(2.0*Math.random()-1.0);
        //s1h[i][j]=s1h[i][j]+s1ddtt*delt;
        //s2h[i][j]=s2h[i][j]+s2ddtt*delt;

//*** s1 の変域は(0<=s1<=1) ***
           if(s1h[i][j]>=1.0){s1h[i][j]=1.0;}
           if(s1h[i][j]<=0.0){s1h[i][j]=0.0;}
           if(s2h[i][j]>=1.0){s2h[i][j]=1.0;}
           if(s2h[i][j]<=0.0){s2h[i][j]=0.0;}
        }
     }

//******[時間増加]**************************************************
     time1=time1+1.0;
        //try{ thread1.sleep(100); }         // 0.01 seconds
        //catch( InterruptedException e){ }
  }//while
}//main

// ****  初期場の設定 ********************************************************
  public void ini_field(){
     int i, j;
     double rnd0, fac1;

     fac1=0.1; //最大 1%の初期揺らぎ
     for(i=0;i<=ndm;i++){
        for(j=0;j<=ndm;j++){
           rnd0=2.0*Math.random()-1.0;    s1h[i][j]=fac1*rnd0;
           rnd0=2.0*Math.random()-1.0;    s2h[i][j]=fac1*rnd0;
        }
     }
  }

// ***** update 関数のオーバーライド（再描画時における画面のチラツキ防止）*******
    public void update(Graphics g){paint(g);}

// ****  結晶場の描画 ******************************************************
    public void paint(Graphics g){
        bg.clearRect(0, 0, width, height);
        bg.setColor(Color.lightGray);

        int i, j, ii, jj;
        int icol, icol_R, icol_G, icol_B, icol_RG;
        int ixmin=0, iymin=0, igx, igy, irad0;
```

```
        int ixmax, iymax;
        double c, x, xmax, xmin, y, ymax, ymin, rad0;

        xmin=0.0; xmax=1.0;    ymin=0.0; ymax=1.0;
        ixmax=xwidth;    iymax=yheight;

        //System.out.println(time1);
        rad0=1.0/(double)nd/2.0;
        irad0=1+(int)( ((double)ixmax-(double)ixmin)/(xmax-xmin)*rad0 );

        for(i=0;i<=nd;i++){
          for(j=0;j<=nd;j++){
            x=1.0/(double)nd*(double)i+rad0;
            igx=(int)( ((double)ixmax-(double)ixmin)*(x-xmin)/(xmax-xmin)+(double)ixmin );
            y=1.0/(double)nd*(double)j+rad0;
            igy=(int)( ((double)iymax-(double)iymin)*(y-ymin)/(ymax-ymin)+(double)iymin );
            ii=i; jj=j;
            if(i==nd){ii=0;}    if(j==nd){jj=0;}

            icol_R=(int)(255.0*s1h[ii][jj]);
            icol_G=(int)(255.0*s2h[ii][jj]);
            icol_RG=icol_R+icol_G;
            if(icol_RG>255){icol_RG=255;}
            icol_B=255-icol_RG;
            if(icol_R>=255){icol_R=255;} if(icol_R<=0){icol_R=0;}
            if(icol_G>=255){icol_G=255;} if(icol_G<=0){icol_G=0;}
            if(icol_B>=255){icol_B=255;} if(icol_B<=0){icol_B=0;}
            bg.setColor(new Color(icol_R,icol_G,icol_B));
            bg.fillRect(insetx+igx-irad0,insety+igy-irad0, irad0*2, irad0*2);
            //bg.setColor(Color.blue);
            //bg.setColor(new Color(0,0,icol));
          }
        }
        g.drawImage(buff, 0, 0, this);
      }

//************[場データの保存]************************************
      public void datsave(){
        int   i,j;

        try{
          PrintWriter outfile= new PrintWriter(
            new BufferedWriter(new FileWriter("test.dat", true)) );

          outfile.println(time1);
          //outfile.println(time1+'¥n');
          for(i=0;i<=ndm;i++){
            for(j=0;j<=ndm;j++){
              outfile.println(s1h[i][j]); //場の書き込み
              outfile.println(s2h[i][j]); //場の書き込み
            }
          }
          outfile.close();
        }
        catch(Exception e){System.out.println(e);System.exit(1);}
      }
```

```
//************[結晶場データの読込み]**********************************
    public void datin(){
      int   i, j;
      double sumc;
      String s_data, s_time1;

      try{
        BufferedReader infile=new BufferedReader(new FileReader("ini000.dat"));
        try{
          s_time1=infile.readLine();   time1=new Double(s_time1).doubleValue();
          for(i=0;i<=ndm;i++){
            for(j=0;j<=ndm;j++){
              s_data=infile.readLine();   s1h[i][j]=new Double(s_data).doubleValue();
              s_data=infile.readLine();   s2h[i][j]=new Double(s_data).doubleValue();
            }
          }
        }
        catch(Exception e){System.out.println(e); System.exit(1);}
        finally{infile.close();}
      }
      catch(Exception e){System.out.println(e); System.exit(1);}
    }

//******* Sin, Cos Table ************************************
      public void table(){
        int it, it1, it2, mc, mn;
        double q;

        q=2.0*PI/nd;
        for(it=0;it<=nd2-1;it++){ c[it]=Math.cos(q*it); s[it]=Math.sin(q*it); }
        ik[0]=0;   mn=nd2;   mc=1;
        for(it1=1;it1<=ig;it1++){
          for(it2=0;it2<=mc-1;it2++){ ik[it2+mc]=ik[it2]+mn; }
          mn=mn/2; mc=2*mc;
        }
      }

//********** FFT ***************************************
      public void fft(){
        int ix, ka, kb, l2, lf, mf, n2, nf;
        double tj, tr;

        l2=1;
        for(lf=1;lf<=ig;lf++){
          n2=nd2/l2;
          for(mf=1;mf<=l2;mf++){
            for(nf=0;nf<=n2-1;nf++){
              ix=nf*l2;   ka=nf+2*n2*(mf-1);   kb=ka+n2;
              tr=xrf[ka]-xrf[kb];                tj=xif[ka]-xif[kb];
              xrf[ka]=xrf[ka]+xrf[kb];           xif[ka]=xif[ka]+xif[kb];
              xrf[kb]=tr*c[ix]+tj*qs*s[ix];  xif[kb]=tj*c[ix]-tr*qs*s[ix];
            }
          }
          l2=l2*2;
        }
      }
```

```
//************ RC FFT ***********************************
      public void rcfft(){
         int i, ic, ir, j;

         for(ir=0;ir<=ndm;ir++){
            for(ic=0;ic<=ndm;ic++){ xrf[ic]=xr[ir][ic];   xif[ic]=xi[ir][ic]; }
            fft();
            for(ic=0;ic<=ndm;ic++){ xr[ir][ic]=xrf[ik[ic]]; xi[ir][ic]=xif[ik[ic]]; }
         }
         for(ic=0;ic<=ndm;ic++){
            for(ir=0;ir<=ndm;ir++){ xrf[ir]=xr[ir][ic]; xif[ir]=xi[ir][ic]; }
            fft();
            for(ir=0;ir<=ndm;ir++){ xr[ir][ic]=xrf[ik[ir]];   xi[ir][ic]=xif[ik[ir]]; }
         }
         if(qs>0.){return;}
         for(i=0;i<=ndm;i++){
            for(j=0;j<=ndm;j++){ xr[i][j]=xr[i][j]/nd/nd; xi[i][j]=xi[i][j]/nd/nd; }
         }
      }

//*****************************************************************
}//class TwinMar2D

//**** 以上、プログラム終わり  ****************************************
```