平成 29 年度

卒業論文

時間依存ギンツブルグ-ランダウ方程式を用いた超伝導体内磁束運動解析における GPU による計算高速化

木内研究室

（学籍番号: 12232091）

吉原 敬貴

九州工業大学情報工学部

電子情報工学科

指導教員: 木内 勝 准教授

平成 30 年 2 月 16 日

目次

1. 序論

1.1. はじめに

オランダの Kamerlingh-Onnes は 1908 年, 世界で初めて液体ヘリウムの生成に成功した．後の 1911 年には，液体ヘリウムを用いて水銀を冷却すると 4.2 K 近傍で電気抵抗が消失する現象を発見し，これを超伝導と称した．損失なく電気を輸送できるこの現象の発見に，工学界は大きな期待を寄せ, 研究が進むにしたがって様々な物質において超伝導の発現が観測されることとなった．また，1933 年にはドイツの物理学者 W.Meißner と R.Ochsenfeld によって超伝導体内部に侵入する正味の磁束がゼロになる完全反磁性（Meißner-Ochsenfeld 効果）が発見されている．このような電気的，磁気的特性を持つ超伝導体だが，超伝導現象を発現するためには温度的，磁気的に制約がかかる．物質が超伝導状態に遷移する温度を臨界温度T_c，磁界を臨界磁界B_cと呼ぶ.

1957 年，J.Bardeen，L.N.Cooper，J.R.Schrieffer らによって BCS 理論が提唱され，超伝導現象発現機構の基本的な理解が得られた．BCS 理論はT_cが 30 K を超えない（BCS 理論の壁）と主張している．

しかしながら，1986 年にドイツの物理学者 J.G.Bednorz とスイスの物理学者 K.A.Müller らによって，T_cが 35 K となる La-Ba-Cu-O 系の超伝導体が発見されたことを皮切りに，世界中で高いT_cを持つ物質の探索が行われ，翌 1987 年には液体窒素の沸点 77 K を超えるT_cを持つ超伝導体が発見されるに至った．1986 年以降に発見された高いT_cを持つ超伝導物質群を一般に高温超伝導体, またその中で銅酸化物の高温超伝導体を特に銅酸化物高温超伝導体と称する．高温超伝導体の発見により，これまで高コストな上，将来的な枯渇が予想されていた液体ヘリウムに代わり，冷却媒体に液体窒素を用いることが可能となった．

1.2. 磁束ピンニング

超伝導体は，超伝導状態にあるときに電流を流した場合や外部磁場をかけた場合，その磁気的な性質，振る舞いの違いにより，第一種超伝導体と第二種超伝導体に区別される．第一種超伝導体は，電流および外部磁界を与えていない場合，その超伝導体のT_c以下の温度において超伝導状態となり完全反磁性を示す．しかし，これに外部磁界を与えていくと，ある外

部磁界の大きさにおいてその超伝導状態が破壊されてしまう．この磁界は前述したB_cである．一方で，第二種超伝導体は，第一種超伝導体と同じようにある磁界までは完全反磁性を示すが，その磁界を超えると第一種超伝導体とは異なり，超伝導体内部に一定の磁束（磁束線）を侵入させ，超伝導状態を維持することができる．磁束線を侵入させた領域は常伝導状態となるが，全体としては超伝導状態を維持している．この状態を混合状態と呼ぶ．さらに，この第二種超伝導体に外部磁界を与えていくと超伝導状態が破壊される．第二種超伝導体の完全反磁性を示さなくなる転移磁界をB_{c1}，超伝導状態が破壊される転移磁界をB_{c2}とする．

現在発見されている超伝導体では，第一種超伝導体のB_cと比較すると，第二種超伝導体のB_{c2}は非常に大きいことが知られている．このため，工学的な応用には第二種超伝導体が用いられていることが一般的である．第二種超伝導体は，前述したとおり混合状態においては超伝導体内部に磁束線が侵入している（この磁束線の磁束密度を$\boldsymbol{B}$とする）．そのため，超伝導体に流す輸送電流（この電流密度を$\boldsymbol{J}$とする）により，その磁束線（正確にはその磁束線を留める渦糸電流）に Lorentz 力$\boldsymbol{F}_{\mathrm{L}}$が与えられる(図 1.1)．この$\boldsymbol{F}_{\mathrm{L}}$は，

$$\boldsymbol{F}_{\mathrm{L}} = \boldsymbol{J} \times \boldsymbol{B} \tag{1.1}$$

と表すことができる．また，この$\boldsymbol{F}_{\mathrm{L}}$により磁束線が速度$\boldsymbol{v}$で運動した場合，Josephson の式より，誘導起電力

$$\boldsymbol{E} = \boldsymbol{B} \times \boldsymbol{v} \tag{1.2}$$

が生じる．この$\boldsymbol{E}$は$\boldsymbol{J}$と同じ向きに生ずるので，

$$\boldsymbol{J} \cdot \boldsymbol{E} > 0 \tag{1.3}$$

こうした状態が定常的に続くためには，この誘導起電力に見合った損失が発生しなければならない．即ちこの$\boldsymbol{E}$は超伝導体に対して Ohmic な損失をもたらすこととなり，超伝導体の超伝導状態を破壊する原因となる．しかしながら，実際の第二種超伝導体には磁束の運動を止める（$\boldsymbol{v} = 0$）作用があり，第二種超伝導体に含まれる常伝導析出物，空隙，結晶粒界面など，あらゆる欠陥や不均質物質がその作用をする．こうした欠陥などをピンニング・センターと呼び，それらの作用を磁束ピンニングと呼ぶ．磁束ピンニングは，$\boldsymbol{F}_{\mathrm{L}}$がある臨界値を超えるまで磁束線の動きを止めるため，$\boldsymbol{E}$による損失に対する防御機能を果たす．単位体積当たりのピンニング・センターが磁束線に及ぼす力をピン力密度$\boldsymbol{F}_{\mathrm{p}}$とすると，超伝導体に$\boldsymbol{E}$が生じ始める電流密度（これを$\boldsymbol{J}_{\mathrm{c}}$とする）の下では，磁束線に単位体積当たりに大きさ

$$F_{\mathrm{L}} = J_{\mathrm{c}} B \tag{1.4}$$

の Lorentz 力が働いており，これが$\boldsymbol{F}_{\mathrm{p}}$と釣り合っていることから，

$$J_{\mathrm{c}} = \frac{F_{\mathrm{p}}}{B} \tag{1.5}$$

の関係がある．(1.4)式のJ_{c}を臨界電流密度という．第二種超伝導体ではT_{c}，B_{c2}，J_{c}それぞれのパラメータが工学的な応用において重要視される．

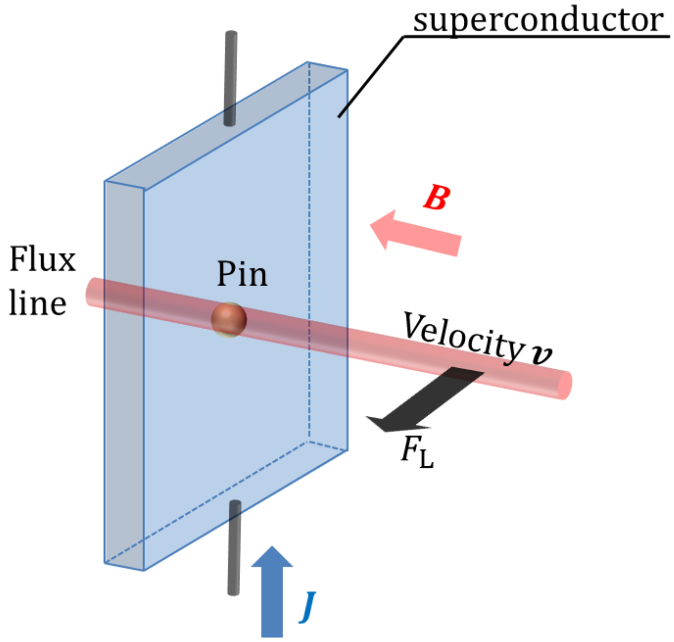

図 1.1. 超伝導体に対して電流密度と磁束密度を垂直に与えた場合の各物理量の模式図．ピンにより発生するピン力は Lorentz 力$\boldsymbol{F}_{\mathrm{L}}$に対する抵抗力として働く

1.3. 超伝導理論

1.3.1. GL（Ginzburg-Landau）方程式

Ginzburg-Landau (以降 G-L と記す)理論は，1950 年に V.L. Ginzburg と L.D. Landau によってロシアで提唱された超伝導を説明する現象論である．G-L 理論は磁界と超伝導が共存する場合の相転移を取り扱ったもので，とくに第二種超伝導体の磁気特性をよく記述することが知られている．[1]

この G-L 理論では，まず超伝導状態の秩序を表す量として，複素数のオーダーパラメータ$\Psi = |\Psi|\exp(i\varphi)$を定義し，$n_{\mathrm{s}}$を超伝導電子密度として以下の関係を満たすものとする．

$$|\Psi|^2 \propto n_{\mathrm{s}} \tag{1.6}$$

超伝導状態の自由エネルギーE_{s}はn_{s}に依存しているため，(1.6)式より$|\Psi|^2$の関数となる．ここで，転移点近傍において$|\Psi|^2$は十分小さいと期待できるため，E_{s}は以下の式のように$|\Psi|^2$の冪展開ができる．

$$E_{\mathrm{s}} = E_{\mathrm{n}} + \alpha|\Psi|^2 + \frac{\beta}{2}|\Psi|^4 \tag{1.7}$$

E_{n}は常伝導状態の自由エネルギーである．また，超伝導-常伝導転移(以降 S-N 転移と記す)

を記述するためには$|\Psi|^4$の項までの展開で十分である．αおよびβはそれぞれ冪展開した際の 1 次と 2 次の係数である．$T < T_c$では$\alpha < 0$，$\beta > 0$である．

次に，磁界の存在がΨの空間的変化に寄与することを考慮して，(1.7)式に磁界のエネルギー密度と運動エネルギー密度を加算する．

$$E_s = E_n + \alpha|\Psi|^2 + \frac{\beta}{2}|\Psi|^4 + \frac{1}{\mu_0}(\boldsymbol{\nabla} \times \boldsymbol{A})^2 + \frac{1}{2m^*}|(-i\hbar\boldsymbol{\nabla} + e^*\boldsymbol{A})\Psi|^2 \tag{1.8}$$

μ_0は真空中の透磁率，Aはベクトルポテンシャル，m^*は超伝導電子の質量，e^*は超伝導電子の電荷量，$\hbar$はプランク定数，iは虚数単位である．

E_sを最小とするように，Ψの共役複素数Ψ^*と$\boldsymbol{A}$について変分法を適用すると，

$$\frac{\delta E_s}{\delta \Psi^*} = \frac{\partial E_s}{\partial \Psi^*} - \left[\boldsymbol{\nabla} \cdot \frac{\partial E_s}{\partial \nabla \Psi^*}\right] = 0 \tag{1.9}$$

$$\frac{\delta E_s}{\delta \boldsymbol{A}} = \frac{\partial E_s}{\partial \boldsymbol{A}} - \left[\boldsymbol{\nabla} \cdot \frac{\partial E_s}{\partial \nabla \boldsymbol{A}}\right] = 0 \tag{1.10}$$

となり，(1.9)式と(1.10)式をそれぞれ解くと，以下の 2 式が得られる．

$$\frac{1}{2m^*}(-i\hbar\boldsymbol{\nabla} + e^*\boldsymbol{A})^2\Psi + \alpha\Psi + \beta|\Psi|^2\Psi = 0 \tag{1.11}$$

$$\frac{1}{\mu_0}\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times \boldsymbol{A} = \frac{i\hbar e^*}{2m^*}(\Psi^*\boldsymbol{\nabla}\Psi - \Psi\boldsymbol{\nabla}\Psi^*) - \frac{e^{*\boldsymbol{2}}}{m^*}|\Psi|^{\boldsymbol{2}}\boldsymbol{A} \tag{1.12}$$

ここで，Coulomb ゲージ$\boldsymbol{\nabla} \times \boldsymbol{A} = 0$を用いた．また，条件として超伝導体表面を横切る電流は流れないことを仮定した．この(1.11)，(1.12)式を G-L 方程式という．

1.3.2. TDGL 方程式

Time-Dependent G-L(以降 TDGL と記す)方程式は，G-L 方程式に時間依存性を付与したものである． (1.9)及び(1.10)式に対して時間発展する場合を考えると，以下の 2 式を得る．

$$\frac{\delta E_s}{\delta \Psi^*} = \frac{\partial E_s}{\partial \Psi^*} - \left[\boldsymbol{\nabla} \cdot \frac{\partial E_s}{\partial \nabla \Psi^*}\right] = -\gamma\frac{\partial \Psi}{\partial t} \tag{1.13}$$

$$\frac{\delta E_s}{\delta \boldsymbol{A}} = \frac{\partial E_s}{\partial \boldsymbol{A}} - \left[\boldsymbol{\nabla} \cdot \frac{\partial E_s}{\partial \nabla \boldsymbol{A}}\right] = -\nu\frac{\partial \boldsymbol{A}}{\partial t} \tag{1.14}$$

γとνはそれぞれΨと$\boldsymbol{A}$の時定数である．さらにゲージ変換を与えると，

$$\hbar\frac{\partial\Psi}{\partial t} \to \left(\hbar\frac{\partial}{\partial t} + \mathrm{i}e^{*}V\right)\Psi \tag{1.15}$$

$$\frac{\partial\boldsymbol{A}}{\partial t} \to \frac{\partial A}{\partial t} + \boldsymbol{\nabla}V \tag{1.16}$$

となる．ここで，Vはスカラーポテンシャルである．(1.15)，(1.16)式を(1.13)，(1.14)式それぞれに代入すると，

$$\gamma\left(\frac{\partial\Psi}{\partial t} + \mathrm{i}e^{*}V\Psi\right) + \frac{1}{2m^{*}}(-i\hbar\boldsymbol{\nabla} + e^{*}\boldsymbol{A})^{2}\Psi + \alpha\Psi + \beta|\Psi|^{2}\Psi = 0 \tag{1.17}$$

$$\begin{aligned} &\nu\left(\frac{\partial\boldsymbol{A}}{\partial t} + \boldsymbol{\nabla}V\right) + \frac{1}{\mu_0}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\boldsymbol{A} + \frac{\mathrm{i}\hbar e^{*}}{2m^{*}}(\Psi^{*}\boldsymbol{\nabla}\Psi - \Psi\boldsymbol{\nabla}\Psi^{*}) \\ &-\frac{e^{*\boldsymbol{2}}}{m^{*}}|\Psi|^{\boldsymbol{2}}\boldsymbol{A} = 0 \end{aligned} \tag{1.18}$$

となる．

本研究では TDGL 方程式を数値解析により解くが，(1.17)，(1.18)式をそのまま解くのは困難であるため，細線近似と規格化による 2 つの簡易化を行う．

細線近似では，非常に細い超伝導線(以降 SC ナノワイヤと記す)に外部磁界$\boldsymbol{B}_{\mathrm{ext}}$を印加した時，SC ナノワイヤ全体に$\boldsymbol{B}_{\mathrm{ext}}$が侵入すると仮定する．これにより，$\boldsymbol{A}$は$\boldsymbol{B}_{\mathrm{ext}}$のみに依存する変数となる．本研究で$\boldsymbol{B}_{\mathrm{ext}}$は一定とするため，(1.18)式は，左辺第一項の時間偏微分が 0 となり一定となる．

次に(1.17)，(1.18)に対して規格化を行う．超伝導体のコヒーレンス長ξと磁界侵入長λは以下の 2 式のように表す．

$$\xi = \frac{\hbar}{\sqrt{2m^{*}|\alpha|}} \tag{1.19}$$

$$\lambda = \frac{e^{*}\mu_0 H_{\mathrm{c}}}{\sqrt{m^{*}|\alpha|}} \tag{1.20}$$

そして，(1.21)~(1.26)式に記す規格化を行う．

$$\xi\boldsymbol{\nabla} \to \boldsymbol{\nabla} \tag{1.21}$$

$$\frac{|\alpha|}{\gamma}t \to t \tag{1.22}$$

$$\frac{e^{*}\gamma}{|\alpha|}V \to V \tag{1.23}$$

$$\frac{\lambda}{\sqrt{2}\mu_0 H_{\mathrm{c}}}\boldsymbol{A} \to \boldsymbol{A} \tag{1.24}$$

$$\left(\frac{\beta}{|\alpha|}\right)^{\frac{1}{2}}\Psi \to \Psi \tag{1.25}$$

すると，(1.17)式左辺第 1 項は,

$$\begin{aligned}
\gamma\left(\frac{\partial\Psi}{\partial t}+\mathrm{i}e^*V\Psi\right) &\to \gamma\left[\frac{\partial}{\partial\left(\frac{\gamma}{|\alpha|}t\right)}\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\Psi+\mathrm{i}e^*\left(\frac{|\alpha|}{e^*\gamma}V\right)\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\Psi\right]\\
&=\gamma\left[\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\frac{\partial\Psi}{\partial\left(\frac{\gamma}{|\alpha|}t\right)}+\mathrm{i}\frac{|\alpha|}{\gamma}\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}V\Psi\right]\\
&=\gamma\left[\frac{1}{\gamma}|\alpha|\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\frac{\partial\Psi}{\partial t}+\frac{1}{\gamma}|\alpha|\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\mathrm{i}V\Psi\right]\\
&=|\alpha|\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\left[\frac{\partial\Psi}{\partial t}+\mathrm{i}V\Psi\right]
\end{aligned} \tag{1.26}$$

同第 2 項は,

$$\begin{aligned}
&\frac{1}{2m^*}(-i\hbar\boldsymbol{\nabla}+e^*\boldsymbol{A})^2\Psi\\
&\quad\to\frac{1}{2m^*}\left(-\mathrm{i}\hbar\frac{\boldsymbol{\nabla}}{\xi}-e^*\frac{\sqrt{2}\mu_0H_\mathrm{c}}{\lambda}\boldsymbol{A}\right)^2\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\Psi\\
&\quad=\left(-\mathrm{i}\hbar\frac{1}{\sqrt{2m^*}\xi}\boldsymbol{\nabla}-e^*\frac{\sqrt{2}\mu_0H_\mathrm{c}}{\sqrt{2m^*}\lambda}\boldsymbol{A}\right)^2\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\Psi\\
&\quad=\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\left(-\mathrm{i}\hbar\frac{\sqrt{2m^*|\alpha|}}{\sqrt{2m^*}\hbar}\boldsymbol{\nabla}\right.\\
&\qquad\qquad\left.-e^*\frac{\sqrt{2}\mu_0H_\mathrm{c}\sqrt{m^*|\alpha|}}{\sqrt{2m^*}e^*\mu_0H_\mathrm{c}}\boldsymbol{A}\right)^2\Psi\\
&\quad=|\alpha|\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}(-\mathrm{i}\boldsymbol{\nabla}-\boldsymbol{A})^2\Psi
\end{aligned} \tag{1.27}$$

同第 3 項は,

$$\mathrm{\alpha}\Psi \rightarrow \alpha \left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}} \Psi \tag{1.28}$$

同第 4 項は,

$$\begin{aligned} \beta|\Psi|^2\Psi &\rightarrow \beta \left|\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}} \Psi\right|^2 \left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}} \Psi \\ &= |\alpha| \left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}} |\Psi|^2\Psi \end{aligned} \tag{1.29}$$

(1.26)~(1.29)式をまとめると，(1.28)式右辺のαは負で,

$$\alpha \left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}} \left[\frac{\partial \Psi}{\partial t} + \mathrm{i}V\Psi + (-\mathrm{i}\boldsymbol{\nabla} - \boldsymbol{A})^2\Psi - \Psi + |\Psi|^2\Psi\right] = 0 \tag{1.30}$$

(1.30)式の両辺を

$$\alpha \left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}$$

で割ると,

$$\frac{\partial \Psi}{\partial t} + \mathrm{i}V\Psi + (-\mathrm{i}\boldsymbol{\nabla} - \boldsymbol{A})^2\Psi - \Psi + |\Psi|^2\Psi = 0 \tag{1.31}$$

(1.18)式も同様に簡易化する．(1.18)式左辺第一項は，$\boldsymbol{A}$が一定として,

$$\begin{aligned} \nu\boldsymbol{\nabla}V &\rightarrow \nu\frac{1}{\xi}\boldsymbol{\nabla}\frac{|\alpha|}{e^*\gamma}V \\ &= \frac{|\alpha|}{\xi e^*\gamma}\nu\boldsymbol{\nabla}V \\ &= \frac{\sqrt{2m^*|\alpha|}}{\hbar}\cdot\frac{|\alpha|}{e^*\gamma}\nu\boldsymbol{\nabla}V \end{aligned} \tag{1.32}$$

(1.18)式左辺第 2 項は,

$$
\begin{aligned}
&\frac{1}{\mu_0}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\boldsymbol{A}\to\frac{1}{\mu_0}\cdot\frac{1}{\xi}\boldsymbol{\nabla}\times\frac{1}{\xi}\boldsymbol{\nabla}\times\frac{\sqrt{2}\mu_0H_\mathrm{c}}{\lambda}\boldsymbol{A}\\
&=\frac{\sqrt{2}H_\mathrm{c}}{\xi^2\lambda}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\boldsymbol{A}\\
&=\frac{2m^*|\alpha|}{\hbar^2}\cdot\frac{\sqrt{m^*|\alpha|}}{e^*\mu_0H_\mathrm{c}}\cdot\sqrt{2}H_\mathrm{c}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\boldsymbol{A}\\
&=\frac{2\sqrt{2}m^*|\alpha|\sqrt{m^*|\alpha|}}{\hbar^2e^*\mu_0}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\boldsymbol{A}
\end{aligned}
\tag{1.33}
$$

(1.18)式左辺第 3 項は,

$$
\begin{aligned}
&\frac{\mathrm{i}\hbar\mathrm{e}^*}{2m^*}(\Psi^*\boldsymbol{\nabla}\Psi-\Psi\boldsymbol{\nabla}\Psi^*)\\
&\to\frac{\mathrm{i}\hbar\mathrm{e}^*}{2m^*}\left\{\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\Psi^*\frac{1}{\xi}\boldsymbol{\nabla}\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\Psi\right.\\
&\qquad\left.-\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\Psi\frac{1}{\xi}\boldsymbol{\nabla}\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\Psi^*\right\}\\
&=\frac{\mathrm{i}\hbar\mathrm{e}^*}{2m^*}\cdot\frac{|\alpha|}{\beta}\cdot\frac{1}{\xi}(\Psi^*\boldsymbol{\nabla}\Psi-\Psi\boldsymbol{\nabla}\Psi^*)\\
&=\frac{\mathrm{i}\hbar\mathrm{e}^*}{2m^*}\cdot\frac{|\alpha|}{\beta}\cdot\frac{\sqrt{2m^*|\alpha|}}{\hbar}(\Psi^*\boldsymbol{\nabla}\Psi-\Psi\boldsymbol{\nabla}\Psi^*)\\
&=\frac{\mathrm{i}e^*|\alpha|\sqrt{2m^*|\alpha|}}{2m^*\beta}(\Psi^*\boldsymbol{\nabla}\Psi-\Psi\boldsymbol{\nabla}\Psi^*)
\end{aligned}
\tag{1.34}
$$

同第 4 項は,

$$
\begin{aligned}
&\frac{e^{*2}}{m^*}|\Psi|^2\boldsymbol{A}\to\frac{e^{*2}}{m^*}\left|\left(\frac{|\alpha|}{\beta}\right)^{\frac{1}{2}}\Psi\right|^2\frac{\sqrt{2}\mu_0H_\mathrm{c}}{\lambda}\boldsymbol{A}\\
&=\frac{e^{*2}}{m^*}\cdot\frac{|\alpha|}{\beta}\cdot\frac{\sqrt{2}\mu_0H_\mathrm{c}}{\lambda}|\Psi|^2\boldsymbol{A}\\
&=\frac{e^{*2}}{m^*}\cdot\frac{|\alpha|}{\beta}\cdot\sqrt{2}\mu_0H_\mathrm{c}\cdot\frac{\sqrt{m^*|\alpha|}}{e^*\mu_0H_\mathrm{c}}|\Psi|^2\boldsymbol{A}\\
&=\frac{e^*|\alpha|\sqrt{2m^*|\alpha|}}{m^*\beta}|\Psi|^2\boldsymbol{A}
\end{aligned}
\tag{1.35}
$$

(1.32)~(1.35)式をまとめると,

$$\frac{2\sqrt{2}m^*|\alpha|\sqrt{m^*|\alpha|}}{\hbar^2 e^* \mu_0}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\boldsymbol{A}$$
$$=\frac{e^*|\alpha|\sqrt{2m^*|\alpha|}}{m^*\beta}\left\{|\Psi|^2\boldsymbol{A}-\frac{\mathrm{i}}{2}(\Psi^*\boldsymbol{\nabla}\Psi-\Psi\boldsymbol{\nabla}\Psi^*)\right\}-\frac{\sqrt{2m^*|\alpha|}}{\hbar}\cdot\frac{|\alpha|}{e^*\gamma}\nu\boldsymbol{\nabla}V \tag{1.36}$$

(1.36)式の両辺を$|\alpha|\sqrt{2m^*|\alpha|}$で割ると，

$$\frac{2m^*}{\hbar^2 e^*\mu_0}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\boldsymbol{A}=\frac{e^*}{m^*\beta}\left\{|\Psi|^2\boldsymbol{A}-\frac{\mathrm{i}}{2}(\Psi^*\boldsymbol{\nabla}\Psi-\Psi\boldsymbol{\nabla}\Psi^*)\right\}-\frac{1}{\hbar e^*\gamma}\nu\boldsymbol{\nabla}V \tag{1.37}$$

ここで，本研究ではTDGL方程式をΨとVについて解くが，変数が2つに対して方程式が(1.31)式のみであるため，電流の発散の式

$$\boldsymbol{\nabla}\cdot\boldsymbol{J}=0 \tag{1.38}$$

を第二の方程式として解く．ここで，

$$\boldsymbol{J}=\frac{1}{\mu_0}\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\boldsymbol{A} \tag{1.39}$$

である．(1.31)式の両辺で$\boldsymbol{\nabla}$との内積を取ると，

$$\begin{aligned}
&0=\boldsymbol{\nabla}\cdot\left[\frac{e^*}{m^*\beta}\left\{|\Psi|^2\boldsymbol{A}-\frac{\mathrm{i}}{2}(\Psi^*\boldsymbol{\nabla}\Psi-\Psi\boldsymbol{\nabla}\Psi^*)\right\}-\frac{1}{\hbar e^*\gamma}\nu\boldsymbol{\nabla}V\right]\\
&\leftrightarrow\frac{e^*}{m^*\beta}\left\{\frac{\mathrm{i}}{2}(\Psi^*\boldsymbol{\nabla}^2\Psi-\Psi\boldsymbol{\nabla}^2\Psi^*)-|\Psi|^2\boldsymbol{A}\right\}=-\frac{1}{\hbar e^*\gamma}\nu\boldsymbol{\nabla}^2V\\
&\leftrightarrow\frac{\mathrm{i}}{2}(\Psi^*\boldsymbol{\nabla}^2\Psi-\Psi\boldsymbol{\nabla}^2\Psi^*)-|\Psi|^2\boldsymbol{A}=-\frac{m^*\beta}{\hbar {e^*}^2\gamma}\nu\boldsymbol{\nabla}^2V\\
&\leftrightarrow\sigma\boldsymbol{\nabla}^2V=\frac{\mathrm{i}}{2}(\Psi^*\boldsymbol{\nabla}^2\Psi-\Psi\boldsymbol{\nabla}^2\Psi^*)-|\Psi|^2\boldsymbol{A}
\end{aligned} \tag{1.40}$$

が得られる．ここで，

$$-\frac{m^*\beta}{\hbar {e^*}^2\gamma}\nu\rightarrow\sigma \tag{1.41}$$

とする．本研究では(1.31)式と(1.40)式を数値計算によって解析する．

一方，SCナノワイヤの対破壊電流密度J_dを考えると，1次元のG-L方程式は$\Psi(x)=f\exp(\mathrm{i}qx)$に対して$\boldsymbol{A}=0$として，(31)式より，

$$
\begin{aligned}
0 &= \nabla^2\Psi + (1-|\Psi|^2)\Psi \\
&= \frac{d^2}{dx^2}\left(fe^{\mathrm{i}qx}\right) + (1-f^2)fe^{\mathrm{i}qx} \\
&= (-q^2+1-f^2)fe^{\mathrm{i}qx}
\end{aligned} \tag{1.42}
$$

故に,

$$
f = 1 - q^2 \tag{1.43}
$$

が成り立つ．このパラメータfとqはJ_sから決定される．

$$
\begin{aligned}
J_\mathrm{s} &= |\Psi|^2\frac{\partial\varphi}{\partial x} \\
&= f^2 q \\
&= q(1-q^2)
\end{aligned} \tag{1.44}
$$

$q(1-q^2)$は

$$
q = \frac{1}{\sqrt{3}} \approx 0.577
$$

のとき最大値

$$
\frac{2}{3\sqrt{3}} \approx 0.385
$$

を取るため,

$$
\begin{aligned}
J_\mathrm{s} \le J_\mathrm{d} &= \frac{2}{3\sqrt{3}} \\
&\approx 0.385
\end{aligned} \tag{1.45}
$$

(1.40)式より，規格化された電流密度Jは

$$
\frac{2}{3\sqrt{3}} \approx 0.385
$$

を超えて印加してはならない.

1.4.　ガウス＝ザイデル法

連立 1 次方程式の解法には様々な手段があるが, その中に反復法の名で総称される 1 つの方法群があり，この反復法の 1 つにガウス＝ザイデル法がある．[2][3][4]

このガウス＝ザイデル法は同じく反復法の 1 つであるヤコビ法といわれる方法を改良したもので，連立 1 次方程式$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$を仮定するとき，ヤコビ法では,

$$
x_i^{k+1} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^k - \sum_{j=i+1}^{n} a_{ij}x_j^k\right) \tag{1.46}
$$

ガウス＝ザイデル法では,

$$a_{ii}x_i^{k+1} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^{k}\right) \tag{1.47}$$

の定義を与える．[2]変化値$|x_i^{k+1} - x_i^k|$が任意の値以下になるまで繰り返しこの計算を行う．ガウス＝ザイデル法はヤコビ法と比べて実装した際の使用容量と計算速度の両方の点で優れる．

1.5. オイラー法

常微分方程式の数値解法の一つにオイラー法がある．[3][5]これは導関数の定義式

$$\frac{dx}{dt} = f(t,x) = \lim_{\Delta t}\frac{f(t+\Delta t) - f(t)}{\Delta t} \tag{1.48}$$

が元となっている．ここで，Δtが微小であると仮定し，(1.51)式を

$$f(t,x) = \frac{f(t+\Delta t) - f(t)}{\Delta t} \tag{1.49}$$

のように差分商で置き換える．これより，$t+\Delta t$における変数の値$x(t+\Delta t)$は，

$$x(t+\Delta t) = x(t) + \Delta t \cdot f(t,x) \tag{1.50}$$

と表現できる．

オイラー法は数学的にシンプルであり，プログラムに実装することも容易であるが，1 階段数常微分方程式の数値解法としては，tの発展に伴って誤差が蓄積されるため精度が悪い．この計算誤差を低減するためにいくつかの手法が考えられている．[3]

1.6. GPU と GPGPU

1.6.1. GPU

GPU とは graphics processing unit の略称で，画像処理を目的に開発されたプロセッサである．高負荷な画像処理を目的としているため，定型かつ大量の演算を並列に処理する能力に重点が置かれ開発される．その起源は 1970 年代から使用され始めたグラフィックコントローラと呼ばれる，コンピュータがディスプレイに出力する映像を処理するための IC（Integrated Circuit：集積回路）にまで遡る．当時のグラフィックコントローラは安価に作れば機能に乏しく，性能を求めれば非常に高価となってしまったため広く普及することはなかったが，技術の進歩に伴って次第に機能面と価格面での釣り合いが取れるようになり，パーソナルコンピュータの普及によってその需要が高まったことも重なって広く知られるよ

うになった．DirectX 第 8 世代以降では陰影処理をユーザ側で自由に記述できるプログラマブルシェーダが導入され，DirectX 第 9 世代になるとこのプログラマブルシェーダがさらに進化し，専用の高級言語の登場などの助けも借りて，物理演算など画像処理以外の目的で使われ始めることとなった．DirectX 第 10 世代に入るといくつかに分割されていたシェーダの機能を統合する統合型シェーダ（Unified Shader）が登場し，プログラムの自由度が向上，GPU の汎用計算への利用が進んでいくこととなった．現在では HPC（high-performance computing）分野向けに，GPU でありながら映像出力端子を持たない汎用計算専用製品も開発されている．CPU と比較すると，搭載された多数のプロセッサを活用した並列処理による，高負荷で密な大量演算に優れ，また最大バンド幅が大きく，メモリへのアクセス速度に秀でる．以下に，NVIDIA 社が製造する GPU を例にとってその構造を示す．

1.6.1.1. 構造

NVIDIA 社製の GPU を構成する部品は主に 6 つある．基板，GPU チップ，DRAM（Dynamic Random Access Memory），画面出力端子，電源入力部，冷却装置である．このうちメモリが実装されているのは基板と GPU チップであるが，GPU にはメモリが数種類存在し，大別すると GPU チップ上に実装されたオンチップメモリと，基板上に実装されたオフチップメモリとなる．下に Pascal 世代（GP104）チップの概略図を図 1.2 として示す．[6]Pascal 世代では図中緑色の四角形で示された演算ユニット（CUDA コア：後述）32 個をまとめ，Warp スケジューラ（効率よく CUDA コアを動かすためのスケジューラ），2 命令発行の命令バッファ，及び超越関数ユニット（Super Function Unit：SFU）8 個と合わせて PB（Processing Block）と呼ばれる実行単位を設けている．PB は 4 つを 1 組として SM（ストリーミングマルチプロセッサ：後述）を構成する．各 SM にはテクスチャ・ユニットが設置され，SM に PolyMorph エンジンを併設したものを TPC（Texture/Processing Cluster）と呼ぶ．そしてこの TPC を 5 つまとめたものに Raster エンジンを加えたものを GPC（Graphic Processing Cluster）とする．GP104 コアにはこの GPC が 4 基実装されており，合計で 2,560 個の CUDA コアを擁する．

図 1.2.　Pascal 世代（GP104）GPU チップ概略図．緑色の四角形で示された演算ユニットの CUDA コア 32 個，Warp スケジューラ，2 命令発行の命令バッファ，SFU8 個をまとめて PB と呼ばれる実行単位として設けている．PB は 4 つを 1 組として SM を組織し，各 SM にはテクスチャ・ユニットが置かれる．SM に PolyMorph エンジン（ジオメトリエンジン）をつないだものを TPC（Texture/Processor Cluster）とし，5 基の TPC に Raster エンジンを併設したものを GPC（Graphics Processing Cluster）と呼ぶ．GP104 コアにはこの GPC が 4 基実装されており，合計で 2,560 個の CUDA コアを擁する．

1.6.1.1.1.　ストリーミングマルチプロセッサ（SM）

ストリーミングマルチプロセッサ（以下 SM）とは GPU チップ上に実装された演算部であり，製品によって SM がいくつ搭載されているかは異なる．CUDA コアと呼ばれる演算装置とメモリ（レジスタとシェアードメモリ）及びその他いくつかのユニットで構成されている．CUDA コアは GPU の持つ演算装置の最小単位であり，各 SM 内に搭載されている CUDA コアの数は世代によって異なる（Pascal 世代で 128 個）．

レジスタは CUDA コアに最も近い場所に置かれ，アクセスも最速である．48 byte を単位として構成され，各 CUDA コアが独立に保持する領域で，共有はできない．

シェアードメモリは L1 キャッシュと呼ばれるメモリと合計で 96 Kbyte の容量があるメモリで，Pascal アーキテクチャでは 4 つの PB が 1 つのシェアードメモリを共有する（24 Kbyte/PB）．レジスタに次いで高速にアクセス可能である．

下に SM のブロックダイアグラムを図 1.3 として示す．[6]

図 1.3. SM ブロックダイアグラム．命令キャッシュ，テクスチャ/L1 キャッシュ 2 個（テクスチャキャッシュと L1 キャッシュは領域を共有），テクスチャ・ユニット 8 個，96 Kbyte シェアードメモリ及び 4 基の PB で構成される．PB は命令バッファ，Warp スケジューラ，2 個のディスパッチユニット，レジスタファイル，CUDA コア 32 個，ロード/ストアユニッ

ト 8 個，SFU8 個で構成される GPU の実行単位である．

1.6.1.1.2.　DRAM

DRAM 上にはグローバルメモリやローカルメモリといった，比較的低速大容量のメモリが実装されている．

グローバルメモリは CUDA コアから離れた位置に設置されるためアクセスは遅い．容量はほかのメモリより大きく，すべての CUDA コアが読み書きを行える領域である．

ローカルメモリはレジスタに格納しきれなかったデータを一時的に保持する領域である．

1.6.1.1.3.　L2 キャッシュ

L2 キャッシュ（Level 2 cache）は L1 キャッシュの下位にあるキャッシュで，GPU チップ上に搭載されている．格納されるデータは各 SM 間で共有が可能である．

1.6.1.1.4.　L1 キャッシュ

L1 キャッシュ（Level 1 cache）は L2 キャッシュの上位にあるキャッシュで，シェアードメモリと容量を共有する．同一 SM 内での CUDA コア間で共有ができる点もシェアードメモリと共通している．

1.6.1.2.　本研究で使用する GPU

本研究で使用するのは 1.7.1.1. で示した NVIDIA 社製 Pascal 世代（GP104）の GeForce GTX1080 である．図 1.4 に外観を示す．CUDA コアを 2,560 個搭載しており，複数の GPU を同時に複数利用する SLI テクノロジーに対応している．

図 1.4. GeForce GTX1080 外観

1.6.2.　GPGPU

GPGPU とは General-purpose computing on graphics processing units の略称で，GPU の演算資源を画像処理以外の目的に利用すること，またその技術を指す．1.6.1.で述べたように，定型な大量の演算を並列処理によって高速に行うことができ，プログラマブルシェーダの発展によって柔軟性を獲得した GPU を他の計算に応用することを目的に様々な開発環境や製品が登場してきている．

GPU はメモリに連続的にアクセスし，かつ条件分岐のない密な計算に用いるのに適している．逆に条件分岐の多数存在する処理，木構造やポインタをたどる処理を含む処理は苦手としている．これは，1.6.1.で述べた SM に代表されるように，GPU が複数の演算ユニットをまとめてクラスタとしており, 演算ユニットに命令を出すインストラクションユニットは多くの GPU でクラスタごとに 1 つしか設置されておらず，同一クラスタ内のすべてのプロセッサが異なるデータを受け取り，同じ命令を処理するためである．このような SMID 型のデータ処理は画像処理などの単純で大量な計算処理を得意とする一方で, 条件分岐を含む

体系ではオーバーヘッドがかさみ，効率を著しく落としてしまう．CPU ではこのようなペナルティを避けるためにプリフェッチ，プリデコードや投機実行，レジスタリネーミングといった機能を持たされているが，本来画像処理のみを目的として製造される GPU には一般的に搭載されていない．

また，単精度浮動小数点演算ではハイパフォーマンスな GPU だが，画像処理において必要とされることの少ない倍精度浮動小数点演算では，単精度用に作られた演算器で計算しなければならず，対象を分割し複数回演算を行う必要が生じるため性能の低下に大きく寄与することがある．倍精度専用演算器を搭載した製品も存在するが，倍精度専用演算器では単精度演算が不可能となるため，倍精度性能と単精度性能はトレードオフの関係にあるといえる．

また，GPGPU を用いてプログラムを設計する場合は，極力メモリへのアクセスを連続にする，共有メモリを利用する場合はそれを用いるデータを同一 SM 内に格納する，条件分岐をできるだけ削減する，データ構造は基本的に配列しか使えないなど性能を十分に発揮させるための制約が多く存在する．加えて，デバイスとの通信を行うローレベルの API を使う必要があるなど，プログラミングの難易度が高い．

上記のように問題点も少なからず抱える GPGPU だが，実用ソフトウェアも次第に数を増やしており，開発環境は整いつつある．

1.7.　CUDA と JCuda

1.7.1.　CUDA

CUDA とは Compute Unified Device Architecture の略称で，NVIDIA 社が開発・提供する，GPU 向けの汎用並列コンピューティングプラットフォーム及びプログラミングモデルである．NVIDIA 社製 GPU のみをターゲットに開発される環境であり，また NVIDIA 社製 GPU は GPGPU として利用する場合，すべて CUDA によって動作する（OpenCL や DirectCompute などを用いる場合も API コールは全て CUDA を経由する）ため，ソフトとハードが相互に強く依存している．専用チューニングとなるため，CUDA は NVIDIA 社製 GPU を最も高効率に動作させることができる．

CUDA は C や C++の構文に部分的にではあるが対応している．またデバイス（GPU）を駆動するためのカーネルと呼ばれるデバイス用並列処理プログラムコード片を専用の言語で記述する必要がなく，そのローンチ（デバイスへのカーネル実行命令）も C 言語に近い形式で実行できるなど，他の GPU 向け言語に比べ抽象度が高く記述しやすい．更に上述のように NVIDIA 社製 GPU の性能を最大限引き出すことができるため，NVIDIA 社製品を用いる上では非常に有効なプラットフォームといえる．

反面 NVIDIA 社製でない GPU には用いることができず，プログラムが強く CUDA に依存するときには使用するデバイスに NVIDIA 社製品を用いなければならないという制約が生ずる，いわゆるベンダーロックインに陥りやすくなるという欠点もある．

本研究では Microsoft Visual Studio（以下 VS）を用いて CUDA 用ソースコードである CU ファイル（CUDA Source Code File）を編集する．また，VS ツールキットに格納されている開発者向けのコマンドプロンプトを用いて, 実行ファイルである PTX ファイル（Pro Tools 10 Session File）をコンパイルし，プログラムの開発を進める．

1.7.2. JCuda

JCuda は Java で記述されたプログラムを CUDA で動作させるための言語バインディングである．搭載されたライブラリと API により，Java から CUDA Runtime API と Driver API が使用可能となる．また，C/C++で記述されたプログラムを Java に読み替えることも可能となる．CUDA 用のソースコードである CU ファイルを実行するためには，対象の CU ファイルをもとにコンパイルされた PTX ファイルが必要である．

1.8. 研究の目的

過去の様々な研究において，超伝導体の臨界電流密度J_Cはその超伝導体内のピンの条件によって変化することが知られている．J_Cは電気抵抗なく輸送できる電力量に直結するため，工業的にはJ_Cが大きい超伝導体が望ましい．よって，J_Cの改善は超伝導体の研究において一つの大きな課題である．

無数に存在するピンの濃度，形状，大きさ，配置などの条件の組み合わせに対し，それを持つ超伝導体を実際に作成し測定することで研究を行うと，時間的・資源的・資金的に莫大なコストが必要とされる虞がある．プログラムを用いて臨界電流密度の磁界依存性を示すJ_C-B特性の解析ができれば，これらの費用を削減することができ，合理的である．そして，簡易化された TDGL 方程式をオイラー法により解決することでこの特性を解析するプログラム TDGL_Euler_3D が既に存在する．

しかしながら，このプログラムは 3 次元空間内での磁束の運動を解析するものであり，オーダーパラメータΨやスカラーポテンシャルVを全領域的に求める必要があるため，本質的に大量の計算を行う必要がある．既存のプログラムは解析にかなりの時間を要し，効率的に研究を進めるにあたって重大な支障をきたすため，高速化は必須の課題である．しかし上述の通り本質的に大量の計算は避けられず，プログラムの改良による高速化には限界がある．そこで GPU による高速化を試みる考案がなされた．多数のプロセッサによる並列演算によ

って，定型かつ密なものならば大量の計算を高速に行うことのできる GPU の演算資源をこの計算に用いれば，ソフト面だけでは限界のあった TDGL_Euler_3D の高速化を高い水準で達成できることが予測されたためである．本研究の目的は，先進的な技術である GPGPU を用いて，大量の計算を並列処理によって行うことで TDGL_Euler_3D を高速化することとする．

2. 計算手法

2.1. 実行環境

本研究は以下表 2.1 に示す環境の下で行う.

表 2.1. TDGL 解析プログラム開発の実行環境

CPU	Intel® Core™ i7-7700K（4.6GHz）
GPU	NVIDIA® GeForce® GTX 1080
Visual Studio	Microsoft Visual Studio 2017
CUDA	NVIDIA® CUDA Ver.9.0
OS	Windows 10 Pro

また，本研究で高速化を試みる TDGL_Euler_3D は Processing と呼ばれる Java ベースのプログラミングプラットフォームを用いて記述されている．そこで CUDA と Java の言語バインディングである JCuda を用いて Processing 上でカーネル関数をローンチすることで CUDA を利用し，TDGL_Euler_3D_CUDA を作成する.

2.2. フローチャート

本研究で用いる解析用のプログラム（以降 TDGL_Euler_3D と呼ぶ）について，以下に図 2.1 としてフローチャートを示す．TDGL_Euler_3D ではまず，超伝導体内部のピンの条件（形状，数量，配置など），与える磁束密度と電流密度の初期値を設定する．ただしこの磁束密度と電流密度は時間の発展とともに変化しない値とする．次に，一組の磁束密度と電流密度に対し, 差分近似の計算結果が十分に安定な状態に達するまで時間発展を与えて解析する．そして，この処理を電流密度，磁束密度を変化させて繰り返す.

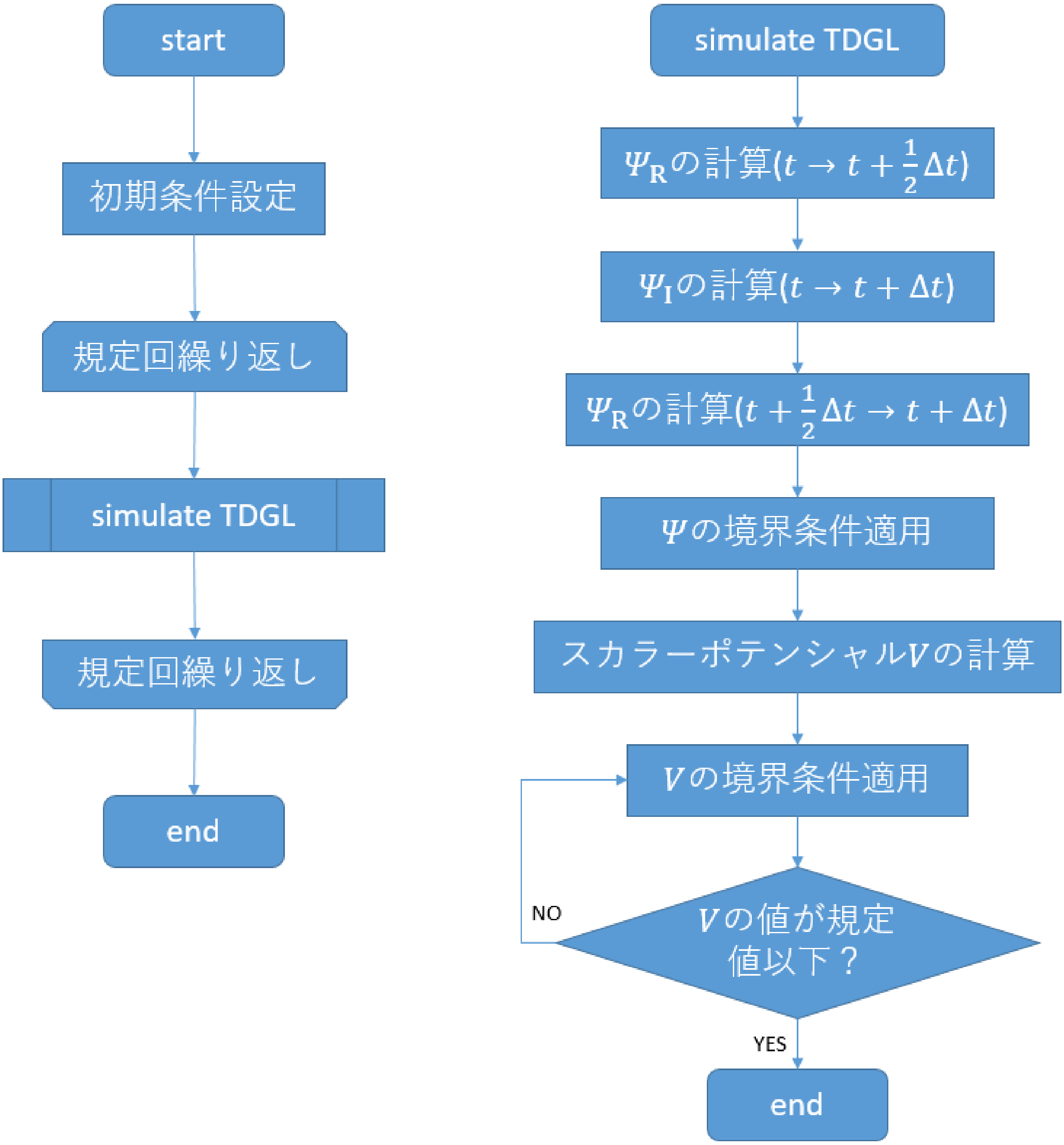

図 2.1. TDGL_Euler_3D フローチャート．ピンに関する条件や時間発展の際の時間離散幅，磁束密度や電流密度の初期値，離散幅，上限値などの設定を最初に行い，これらの設定に応じた回数 TDGL による解析を繰り返す．オーダーパラメータΨの計算は高密度・高負荷となるため，GPGPU による演算に適している．

2.3. GPU 処理部の設定

TDGL_Euler_3D では簡易化した TDGL 方程式をオイラー法により解く．この際超伝導

状態の秩序度を表すオーダーパラメータΨの解析には大量の計算が必要になり，この計算はGPGPU が得意とする定型かつ密な計算である．よって本研究ではこれを GPU により行う．以降，このプログラムを TDGL_Euler_3D_CUDA と呼称する．図 2.2 に TDGL_Euler_3D_CUDA のフローチャートを示す．基本的な流れは TDGL_Euler_3D を踏襲するが，新しく追加された処理及び GPU 資源によって演算を行う処理を赤く着色する．ここで，ポインタやカーネルパラメータ，モジュールなど CUDA の利用にあたって新たに必要となった変数の設定など，一連の CUDA 関係の処理をまとめて「CUDA 準備」と呼ぶ．

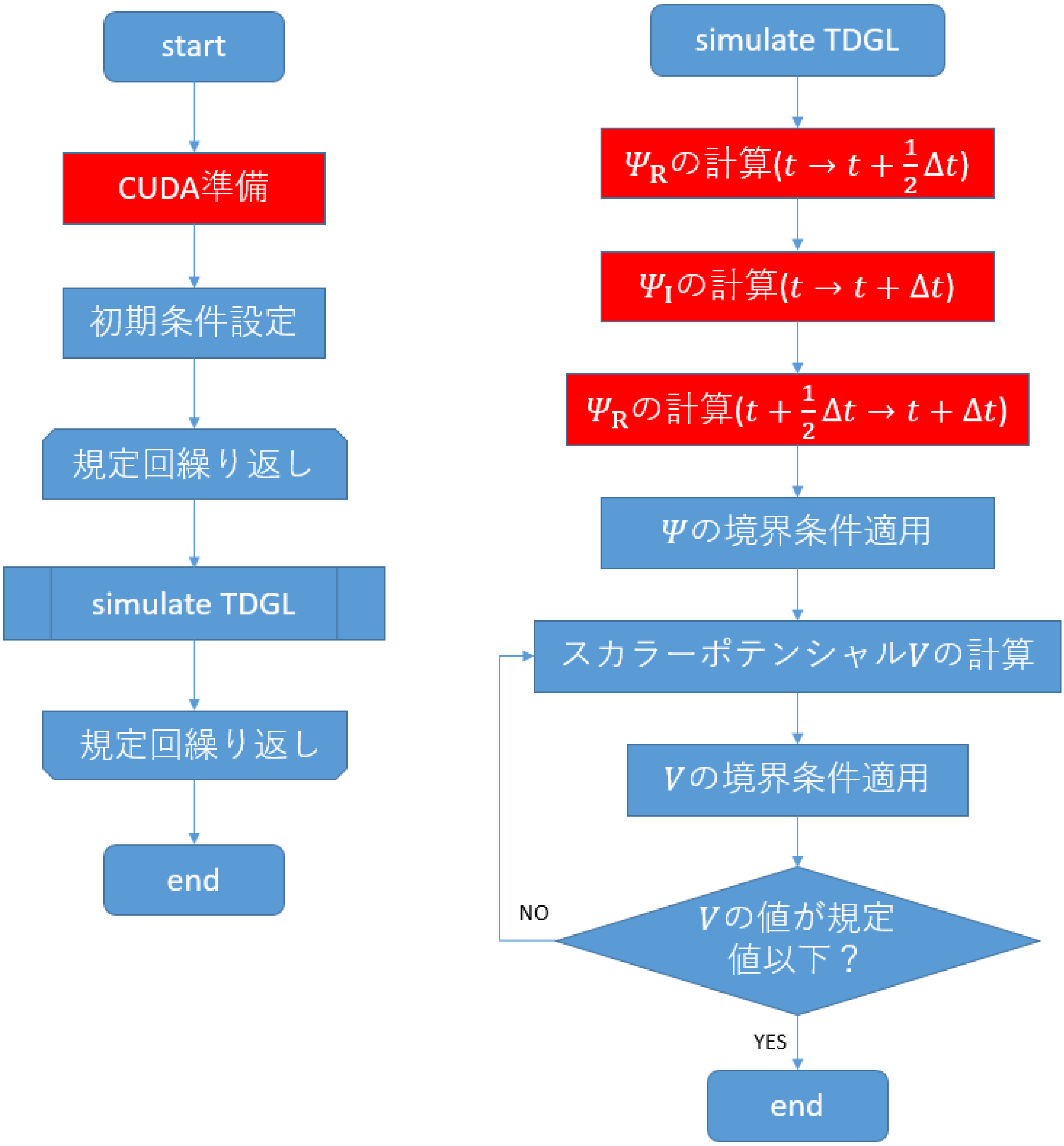

図 2.2. TDGL_Euler_3D_CUDA フローチャート．TDGL_Euler_3D から新たにポインタや

カーネルパラメータ，モジュールなど CUDA の利用にあたって新たに必要となった変数の設定など，一連の CUDA 関係の処理（CUDA 準備）が必要になる．赤く色を付けたのがこの CUDA 準備と，GPU により行う処理の部分である．定型かつ大量な演算であるΨの計算を GPU に担わせることにより，TDGL_Euler_3D と比較して解析を高速に行うことを目的とする．

3. 結果及び考察

3.1. GPGPU による演算

本研究にあたり 2 つ作成した TDGL_Euler_3D_CUDA を, これ以降作成順に Ver.1, ver.2 と呼称することとする．以下に各 TDGL_Euler_3D_CUDA について，得られた結果と考察を記す.

3.1.1. Ver.1

Ver.1 は TDGL_Euler_3D の本体が記述されている Processing と CUDA の連携を確認することを主目的として作成されたプロトタイプである．CUDA との連携に成功した結果，下図 3.1 に示すように実行時間の短縮が確認された．しかし十分に高速とはいえず，速度面での改良は不可欠といえた.

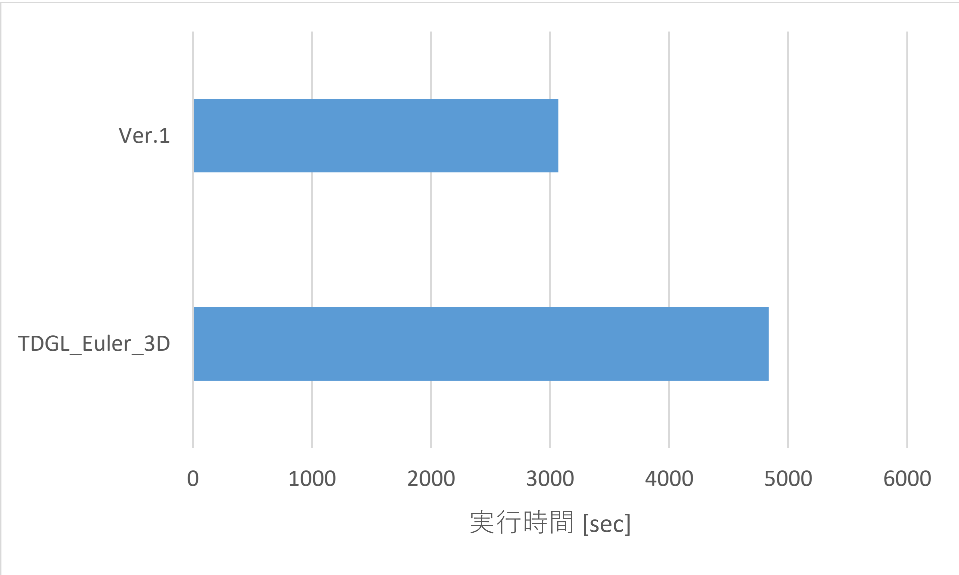

図 3.1. TDGL_Euler_3D と Ver.1 の同一条件下における処理実行時間の比較．Ver.1 での実行時間は 3070 sec，TDGL_Euler_3D では 4836 sec であり，比較すると ver.1 の方が高速に処理できることが確認された．しかし速度は TDGL_Euler_3D の 1.58 倍程度であり，実用的に十分高速とは言えない.

Ψの計算は以下の手順で行われる．まず，ホスト(CPU)からデバイス(GPU)へとホストが持っているΨ及びスカラーポテンシャルVのデータをコピーする．次に GPU で受け取った

データをもとに新たな(離散幅時間経過後の）Ψを計算し，最後に GPU で計算したΨの値を用いてホスト側のΨの値を更新する．図 3.2 にこの処理のフローチャートを示す．GPU は CPU のメモリに干渉できないため，カーネルのローンチに際してこのデータコピー処理が必要となる．これを空間の大きさに応じた回数繰り返すことになるため，Ver.1 ではホストとデバイスの間でデータの交換が頻繁に行われることになり，これが高速化を大きく妨げるボトルネックになっていることが予想される．また，カーネル関数内においても条件分岐が存在し，1.6.2 で述べたように GPU は条件分岐を含む処理では大きくその処理効率を落としてしまう．よってこれも高速化を阻害する原因の一つとして考えられた．

そこで，カーネル関数内の条件分岐の記述を，式の形を変えることで削減したところ，若干スループットが低下した．これは，条件分岐の削減のためにすべてのスレッドが実行する共通の式が冗長化し処理に時間を要するようになったこと，また条件によって処理が分岐する領域が連続であり，SM 単位ではそれほどオーバーヘッドが嵩まないことから，条件分岐によって単純な式を計算した方が却って速かったものと考えられる．

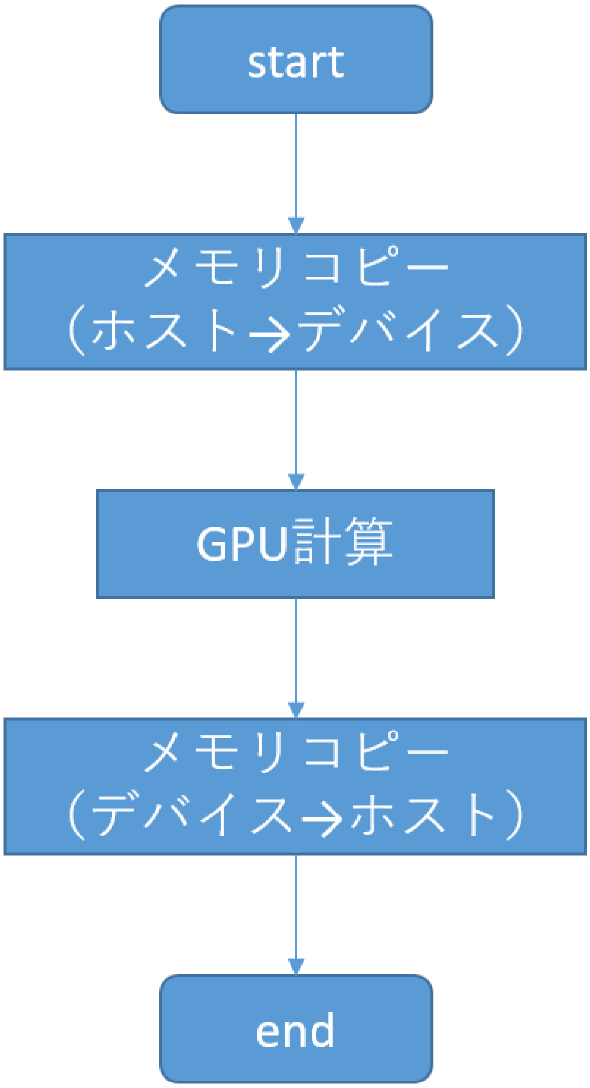

図 3.2. Ver.1 におけるΨのカーネル関数による計算手順．GPU での計算前後にメモリコピーの処理が必要になり，通信速度によってはこれがボトルネックとなる虞がある．

3.1.2. Ver.2

Ver.2 は Ver.1 から得られた知見をもとに改良を加えた完成版である．図 3.3 に示すように，Ver.2 では Ver.1，TDGL_Euler_3D と比較して大きく実行時間を短縮することに成功している．

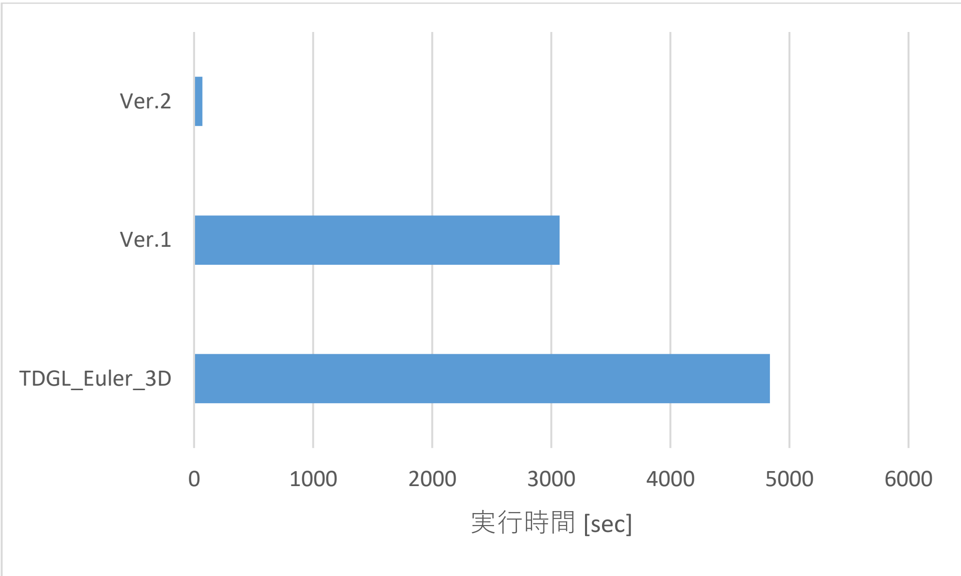

図 3.3. Ver.2，Ver.1，TDGL_Euler_3D の同一条件下における実行時間比較．Ver.2 では他の 2 者と比べ大きく実行時間を短縮することに成功した．TDGL_Euler_3D では 4836 sec を要した計算を，Ver.2 では 69 sec で終了することが可能であり，実行速度は約 70.1 倍になっている．

まず，Ver.1 において条件分岐削減のために統一した式を，再度条件分岐を用いて分割した．また，スループットの低下を招くもう一つの要因であると考えられたホストとデバイスの間で行われるデータ交換の頻度を抑えるため，それまではホスト側で処理を行っていた解析空間の表面（すなわち境界）上のΨの計算をカーネルに統合しΨの計算を完全に GPU に委ねることで，ホストからデバイスへの一部のメモリコピーを不要とした．更に GPU への適性がわかったスカラーポテンシャルVの計算をカーネル化し，Vの計算速度を高めるとともに，それまでホスト側で計算していたが故にΨの計算のたびに GPU へコピーせざるを得なかったVの値を GPU 内部で計算させることで，このメモリコピー処理を削減することを可能とした．また，これにより Ver.2 のフローチャートは Ver.1 から変更されて下図 3.4 に示す通りとなる．

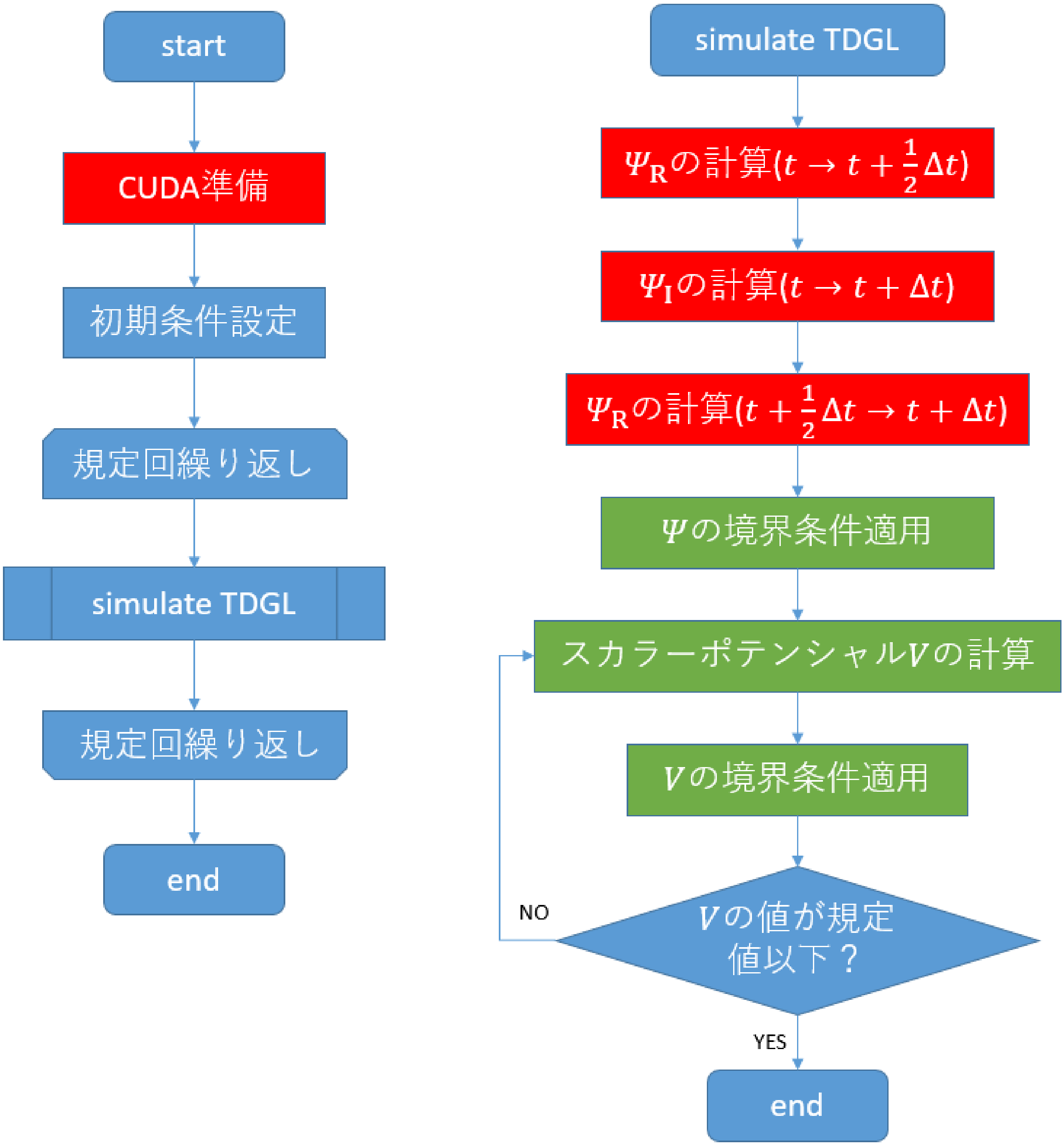

図 3.4. Ver.2 フローチャート．青いブロックはホストの行う処理，赤いブロックは Ver.1 以降で GPU を用いて行われる処理，緑のブロックが Ver.2 において新しく GPU で処理するよう書き換えられた処理である．これによりサブルーチン simulate TDGL は全ての計算を GPU にソーシングすることとなった．

これらの改変によって，Ver.2 は Ver.1 と比較して高速に動作することが確認できた．しかし期待されたほどの高速化は達成されず，ボトルネックが他に存在することが疑われた．そこで Ver.2 内で実行される関数の実行に要する時間（カーネル関数の場合はローンチ後制

御がホスト側に帰ってきて cuCtxSynchronize()関数による同期が終了するまでの時間）を測定したところ，Processing 内標準ライブラリに格納されている draw()関数の反復周期がボトルネックになっていることが判明した．本来 Processing は描画に適したプラットフォームであり, draw()関数の反復実行によってアニメーションを生成する機能が標準で備わっている．アニメーションのフレームレートは draw()関数の反復周期と等しく，デフォルトでは 60 fps に設定されている．これにより draw()関数は 1/60 sec 周期で反復されるが，Ψ やVの計算を draw()関数内部で行う場合，これらの計算が終了した後も draw()関数は自身の反復周期が経過するまでは反復を待機してしまう. これがボトルネックとなって高速化を妨げていたことが判明し，フレームレートの変更によってこれの解消を試みたところ，CUDA との衝突が発生して失敗した．[7]そこで，ΨとVの計算を draw()関数から隔離して計算することで，この問題を解決した．

3.2.　今後の研究

Ver.2 において大幅な性能の向上に成功した TDGL_Euler_3D_CUDA だが, GPU 内部で高速にアクセスできる共有メモリは使っておらず, すべてのデータはグローバルメモリに格納され，計算されている．これは同一 SM 内でのみ共有される共有メモリの性質と，本研究で作成したカーネル関数の相性が良くなかったためであるが, 共有メモリを活用することができれば，更に高速となる可能性がある．

4. まとめ

Ver.1 は CUDA での動作を確認するという目的のために作られたプロトタイプであったが，TDGL_Euler_3D より速く動作していた．但し，期待された範囲には及ばず，目的の達成には改善が不可欠であった．そこで，条件分岐を削減することにより更なる速度の向上を試みたが，条件分岐を削減した Ver.1.1 では動作速度がむしろ低下した．これは，条件分岐を回避するためにすべてのスレッドが実行する式が冗長化したこと, その冗長化した式に倍精度少数の変数が含まれていたこと, そして条件により分岐する式は領域が連続であり, SM 単位ではそれほどオーバーヘッドが嵩まなかったことが主な原因と考えられる．また，CUDA の導入に成功したにもかかわらず速度の向上が期待より伸び悩んだことについては, ホストからデバイス, デバイスからホストへの値の受け渡しを頻繁に行ったことでこの処理がボトルネックとなってしまった可能性がある．

Ver.2 では Ver.1.1 における変更で冗長化した式を条件分岐によって再度分割した．更に Ver.1 における知見から，ホスト関数及びカーネル関数の仕様を大幅に変更し，Ψの値を格納する配列の操作を GPU の専任とする（境界条件の適用をカーネル関数で処理する）ことによりホストとデバイスの間で行われるメモリコピー処理を減らした．また，スカラーポテンシャルVの CUDA への適性から，Vの計算もカーネル関数により行うこととした．これによりスループットの向上を図ったところ，Ver.1 と比較して実行速度は向上したが，やはり期待される範囲には届かなかった．そこで各関数の実行時間（カーネル関数の場合はローンチ後制御がホスト側に帰ってきて cuCtxSynchronize()関数による同期が終了するまで）を調査した結果，60 fps で反復する draw()関数の待機時間がボトルネックになっていたことが判明した．そこで，この調査の結果を受け，Ver.2.2 において，Processing 内の fps 設定の変更により draw()関数の反復周期を調整することを試みたが, CUDA との衝突が発生し, fps の調整によってボトルネックを解消することは不可能であることが判明した．よって，Ψ及びVの計算を draw()関数から隔離し, 一定の回数を外部で演算したのちに draw()関数へ処理を返すことでこの問題を解決した．また，先の各関数の実行時間の調査により，磁束の状態を描画する関数が他の関数と比較して大きく時間を消費することが判明したため, それまでは時間離散幅Δtに対して，Δtごとにキャンバスを更新していたが，$100\Delta t$ごとに更新するよう変更した．その結果実行速度に大きな向上が見られ，実用に堪える実行速度を達成するに至った．

5.　謝辞

本研究を進めるにあたり，松下照男名誉教授には TDGL 方程式の導出について大変なご助力を賜りました．この場を借りまして深謝申し上げます.

また，私の指導を担当してくださいました木内勝准教授には，ゼミ等を通じまして浅学な私にも理解できるよう超伝導の基礎理論を丁寧にご教示いただきました. 不出来な学生に幾度となく教鞭をお取りいただきましたこと，誠に恐縮で申し訳なく存じましたが，深くお礼申し上げます.

小田部荘司教授には解析プログラムの作成にあたり，ソフトウェアやハードウェアについて何度もご相談申し上げました．若輩者の我儘に寛大なお心で接してくださいましたこと，また，研究に行き詰った際に的確なご助言を賜りましたこと，誠にありがとう存じます.

最後に，本研究において礎となるプログラム作成に携わり，不勉強な私に解析プログラムの内容について丁寧にご指導いただきました谷村賢太氏に心より感謝の念をお伝え申し上げたく存じます.

6.　参考文献

[1] T. Matsushita: Flux Pinning in Superconductor, second ed., Springer, 2014

[2] 高橋亮一，棚町芳弘: 計算力学と CAO シリーズ差分法 培風館 東京 1991

[3] 田中敏幸: 数値計算法基礎 コロナ社 東京 2006

[4] 戸川隼人: 数値解析とシミュレーション 共立出版 東京 1987

[5] 新濃清志，船田哲男: 数値解析の基礎-理論と PAD・PASCAL・C 培風館 東京 1991

[6] GeForce GTX 1080 Whitepaper - NVIDIA File Downloads(2018 年 2 月現在)
https://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_1080_Whitepaper_FINAL.pdf

[7] OpenCL Programming Guide for the CUDA Architecture(2018 年 2 月現在)
http://www.nvidia.com/content/cudazone/download/OpenCL/NVIDIA_OpenCL_ProgrammingGuide.pdf