Numerical Analyses of Vortex Structures for Superconductors

東京理科大学　澤渡研究室　6210047 小林慈英

平成 26 年 3 月 20 日

概 要

トポロジカルソリトン模型は素粒子物理学および物性物理学において非常に有用な模型として知られている。私は超伝導体に外部から磁場をかけることで誘起される磁束格子の生成機構の解明を目指し研究を遂行した。その過程で Sine-Gordon 模型、Abelian-Higgs 模型、Baby-Skyrme 模型の数値解を求めた。そして、既存の Baby-Skyrme 模型に外部磁場による相互作用項とサイズ効果を付加した。この新たな模型で外部磁場により磁束格子が生成される現象を再現した。本論文では私が研究してきたソリトン模型の解説及び数値解を求めるための実践的な数値計算の手法を述べる。多少のプログラミング経験がある学生ならば本論文を読んで独学でソリトン模型の解を数値計算により求めることが可能となる。素粒子論だけでなく物性論に興味がある読者にとっても得るものがあるだろう。

目 次

1 はじめに

トポロジカルソリトン模型は素粒子物理から物性物理にいたるまで実に様々な現象を説明する。私が実際に数値計算により解を求め、研究してきた模型は Sine-Gordon 模型、Abelian-Higgs 模型、そして Baby-skyrme 模型である。私自身の興味対象は超伝導体における磁束格子構造とその生成機構である。元々はトポロジカルソリトン模型は素粒子の模型として研究されてきたが物性分野のモデルに応用する研究も近年盛んに行われている。超伝導現象を記述する代表的な理論として Ginzburg-Landau 理論が知られている。Abelian-Higgs 模型はこの Ginzburg-Landau 理論の相対論的アナロジーの模型として考えることができる。そのためまず私は Abelian-Higgs 模型について研究した。次に、異方的超伝導体や強磁性体の模型として知られている Baby-Skyrme 模型について研究を行い、異方的超伝導体に生成される磁束格子の構造および生成機構を Baby-Skyrme 模型に付加した。

数値解析ではSuccesive Over Relaxation method(SOR 法)、Simulated Annealing method(SA 法) の 2 つの数値計算法を用いて解を求めた。このとき用いた数値計算コードの一部は本論文に掲載してある。用いた言語はC++言語であるがクラス機能などのオブジェクト指向寄りの記述はしていない。並列化などの技法も用いていない。そのため、C 言語のプログラムが読める読者であれば解読にさほど困難は生じないだろう。

本論文の構成としては、まずモデル研究のための最低限かつ最重要な数値計算の基礎を解説する。その後、1 次元の最も簡単なソリトン解である kink 解を解とする Sine-Gordon 模型を数値計算法の練習及びソリトン模型の理解を助けるものとして取りあげる。ここで、私が研究に用いてきた代表的な 2 つの数値計算手法である SOR 法と SA 法の基本的な使い方を習得することができる。続く章からは本格的に超伝導現象の研究について述べる。

2　数値計算の基礎

この章ではトポロジカルソリトン模型を解析するための手段である数値計算法の基礎について述べていく。学部の教科書レベルの話なので数値計算に習熟している読者は必要箇所のみ適宜読んでいただきたい。また、数値計算法を習得する際には**習うより慣れろ**の精神を大切にしていただきたい。もしこの章で扱う数値計算法の理論がわからなくても実際にコードを書いているうちに理解できるようになるだろう。臆することなく自分でソースコードを書き数値計算を始めていただきたい。

2.1　数値微分

関数 $f(x)$ の微分は以下のように定義される。

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

数値計算にでは、連続的な値の変化を厳密に求めることは不可能であるため、微分の定義を差分式で作り変えて使用する。h を十分微小な量とする。$f(x+h)$,$f(x-h)$ のテイラー展開は以下の式のようになる。

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2!}f''(x)h^2 + \frac{1}{3!}f'''(x)h^3 + O(h^4) \tag{2}$$

$$f(x-h) = f(x) - f'(x)h + \frac{1}{2!}f''(x)h^2 - \frac{1}{3!}f'''(x)h^3 + O(h^4) \tag{3}$$

これらの式の差をとると $f'(x)$ が求められる。和をとると $f''(x)$ が求められる。

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \tag{4}$$

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - O(h^2) \tag{5}$$

となる。ここで、h の 2 乗以上の項を切り捨てている。これらは h についての 2 次精度の 1 階微分の式と 2 階微分の式である。これらの表式はそのまま格子空間における微分の表現に書き換えることができる。ここでは、1 次元の格子空間を考える。この格子上に一定間隔 Δx で n 個の格子点が設定されているとする。このとき、i 番目の格子点上の関数 f を f_i と表す。ここで、先程求めた差分による微分式で表現すると、

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2\Delta x} + O(\Delta x^2) \tag{6}$$

$$f_i'' = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} - O(\Delta x^2) \tag{7}$$

以上のようにして、数値計算で用いる格子空間における微分を定義することができた。微分方程式を数値計算で解くためには微分方程式をまずこれらの差分式で置き換え、差分方程式を作って数値解を求めていく。

2.2 連立 1 次方程式の反復解法

我々の研究で用いる種々の微分方程式系は 2.1 章で述べた差分化を行うことで連立 1 次方程式になる。そのため、以下に述べる連立 1 次方程式の反復解法を用いることで微分方程式の数値解を求めることが可能となる。本研究の微分方程式の数値計算は SOR 法を用いて行われている。微分方程式系で記述されるソリトン解の数値解を求めるために連立 1 次方程式の反復解法を理解することは必須事項である。

未知数 x_i が n 個存在し、方程式が n 本存在するときの連立 1 次方程式の解法を考える。この方程式は $n \times n$ 行列 A と要素数が n である列ベクトル $\mathbf{x}, \mathbf{b}$ を用いて次のように表せる。

$$A\mathbf{x} = \mathbf{b} \tag{8}$$

ここで、

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \tag{9}$$

行列 A は係数行列であり、以下では行列 A と右辺 $\mathbf{b}$ の要素は実数とし、A は正則行列であるとする。また、自明な解 $\mathbf{x} = 0$ を除外するため $\mathbf{b} \neq 0$ とする。反復解法は「ある演算を繰り返し行い、与えられた初期解を目的とする数値解に近づけていく計算方法である。任意の初期解ベクトル $\mathbf{x_0}$ を初期値として、反復計算を行って収束解を求める。係数行列が後に述べる条件を満たす場合反復計算で収束解を得ることができる。以下に代表的な反復解法の手法を紹介する。

・ヤコビ法 k 回目の反復計算で得られる解ベクトル $\mathbf{x_k}$ の第 i 要素を $x_i^{(k)}$ とする。$(0 \leq k, 1 \leq i \leq n)$。ヤコビ法では、$k$ 回目の演算で $A\mathbf{x}$ を計算する時に、A の第 i 行の対角要素に $x_i^{(k)}$、非対角要素に $x_i^{(k-1)}$ を用いる。よって方程式 $A\mathbf{x} = \mathbf{b}$ の第 i 行の関係は次のように表される。

$$\sum_{j=1}^{i-1} a_{i,j} x_j^{(k-1)} + a_{i,i} x_i^{(k)} + \sum_{j=i+1}^{n} a_{i,j} x_j^{(k-1)} = b_i \tag{10}$$

これより、$x_i^{(k)}$ は値が既知である要素 $x_j^{(k-1)}$ を右辺に移項した次式から計算される。

$$x_i^{(k)} = \frac{1}{a_{i,i}} \Big(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k-1)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(k-1)} \Big) \tag{11}$$

ヤコビ法は上式で得られた $x_i^{(k)}$ を再び右辺にもちいる計算を繰り返し、残差 $\boldsymbol{\omega} = \mathbf{b} - A\mathbf{x_k}$ を十分小さくする収束解 $\mathbf{x_k}$ を求める方法である。

・ガウスザイデル法 ヤコビ法の演算式である (11) 式の右辺では、前の計算ステップで得られた解 $x_j^{(k)}$ が用いられている。同じステップの反復計算で既に値が求められている要素 $x_j^{(k)}$ を (11) 式の右辺で利用する手法がガウスザイデル法である。反復計算の k ステップにおいて、数値解 $x_j^{(k)}$ が $j = 1, 2, \ldots, n$ と順番に求められるとすると、$x_1^{(k)}, x_2^{(k)}, \ldots, x_{i-1}^{(k)}$ が得られているので、ガウスザイデル法の計算式は以下のような式となる。

$$x_i^{(k)} = \frac{1}{a_{i,i}} \Big(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(k-1)} \Big) \tag{12}$$

ガウスザイデル法では収束解に近い $x_j^{(k)}$ がすぐに次の計算の右辺に用いられるのでヤコビ法よりも一般的に少ない計算回数で収束解を得ることができる。

ここで、(12) 式の関係を行列とベクトルで表現する。そのために、係数行列 A を対角行列 D, 対角要素を除くした下三角行列 E, 同じく対角要素を除く上三角行列 F を用いて $A = D - E - F$ と分解する。対角行列 D の逆行列 D^{-1} は D の要素の逆数を要素とする対角行列であることを考慮すると、(12) 式は以下のように書き換えられる。

$$\mathbf{x_k} = (D - E)^{-1}(\mathbf{b} + F\mathbf{x_{k-1}}) \tag{13}$$

これより、

$$T = (D - E)^{-1}F, \qquad C = (D - E)^{-1} \tag{14}$$

とおくと、

$$\mathbf{x_k} = T\mathbf{x_{k-1}} + C\mathbf{b} \tag{15}$$

と表される。この行列 T を反復行列という。以下に述べる条件を満たすときに収束解が得られる。

まず、(15) 式の収束解が得られたときに方程式 $A\mathbf{x} = \mathbf{b}$ になることを確認する。収束解を x^* とすると、

$$\mathbf{x}^* = T\mathbf{x}^* + C\mathbf{b} \tag{16}$$

となるので、(14) 式より、

$$A\mathbf{x}^* = \mathbf{b} \tag{17}$$

となることがわかる。すなわち、収束解は方程式の解である。

また、$n \times n$ の実行列 A の成分が以下の条件式

$$|a_{i,i}| > \sum_{j=1,i\neq i}^{n} |a_{i,j}| \qquad (1 \leq i \leq n) \tag{18}$$

を満たす場合について、(1):A が正則である、(2):ヤコビ法、およびガウスザイデル法で収束解が得られるという 2 つの性質を有する。

・SOR 法 SOR 法は、ガウス・ザイデル法における修正量 $\Delta x_i^{(k)} (= x_i^{(k)} - x_i^{(k-1)})$ に加速パラメータ ω を乗じて行われる反復解法である。ω の値を適切に設定すれば、収束解が得られるまでの反復回数をガウス・ザイデル法よりも減少させることができる。

SOR 法では以下の式から $x_i^{(k)}$ を求める。

$$\begin{aligned} x_i^{(k)} &= x_i^{(k-1)} + \omega\Delta x_i^{(k)} \\ &= x_i^{(k-1)} + \omega\Big[\frac{1}{a_{i,i}}\big(b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{(k)} - \sum_{j=i+1}^{n} a_{i,j}x_j^{k-1}\big) - x_i^{(k-1)}\Big] \end{aligned} \tag{19}$$

SOR 法における加速パラメータ ω の適切な値は事前に知ることはできず、経験則に基づくものである。

2.3 数値積分

数値積分法はエネルギーやトポロジカルチャージの計算など種々の物理量を得るために必要不可欠な計算手法である。本研究では SOR 法により求めた解の妥当性の確認や後に述べる SA 法のコスト計算などに数値積分を用いている。実際の計算としてここでは台形積分を用いた。ここではニュートンコーツ公式から台形積分公式の導出を行う。

・ニュートンコーツ公式 定積分 $I = \int_a^b f(c)dx$ を求めるために、分点 $a = x_0 < x_1 < x_2 < \cdots < x_n = b$ をとり、$f_k = f(x_k)$ を通るラグランジュ補間多項式 $P_n(x)$ を考える。ここで、それぞれの分点は等間隔 $h = \frac{b-a}{n}$ で並んでいるものとする。このとき I を、

$$I_n = \int_a^b P_n(x)dx$$

で近似すると $P_n(x)$ は多項式なので I_n を容易に求めることができる。

具体的に、

$$P_n(x) = \sum_{k=0}^{n} f_k l_k(x)$$

と書くと、

$$\int_a^b f(x)dx \sim \int_a^b P_n(x)dx = \sum_{k=0}^{n} f_k \int_a^b l_k(x)dx = \sum_{k=0}^{n} \alpha_k f_k \tag{20}$$

と書ける。ただし、$\alpha_k = \int_a^b l_k(x)dx$ である。このようにして得られた近似積分公式を $n+1$ 点の**ニュートンコーツ公式**と呼ぶ。

・台形公式 ニュートンコーツ公式において、$n = 1$ とおくと、$h = b - a$,$x_0 = a$,$x_1 = b$ なので、$x = a + sh$ とおくと、

$$\alpha_0 = \int_a^b l_0(x)dx = \int_a^b \frac{x - x_1}{x_0 - x_1}dx = -\frac{1}{h}\int_0^1 h(s-1)hds = \frac{h}{2},$$
$$\alpha_1 = \int_a^b l_1(x)dx = \int_a^b \frac{x - x_0}{x_1 - x_0}dx = \frac{1}{h}\int_0^1 sh \cdot hds = \frac{h}{2}$$

となり、

$$\int_a^b f(x)dx \sim \frac{h}{2}(f_0 + f_1) \tag{21}$$

を得ることができる。この式を**台形公式**という。

実際の計算では、区間 [a,b] を n 等分して、その分点を x_k とし、各小区間 $[x_k,x_{k+1}]$ で台形公式 (21) 式を適用すると、

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{k-0}^{n-1}\int_{x_k}^{x_{k+1}} f(x)dx \sim \sum_{k=0}^{n-1}\frac{h}{2}(f_k + f_{k+1}) \\ &= \frac{h}{2}\{f_0 + f_n + 2(f_1 + f_2 + \ldots + f_{n-1})\}, h = \frac{b-a}{n}\end{aligned} \tag{22}$$

が得られる。他の積分方法として (20) 式において $n = 2$ とすることで得られる**シンプソン公式**なども知られている

・重積分 ここでは、重積分

$$I = \int_a^b \Big\{\int_{\phi(x)}^{\psi(x)} f(x,y)dy\Big\}dx$$

を考える。ここで、

$$F(x) = \int_{\phi(x)}^{\psi(x)} f(x,y)dy$$

とおけば、

$$I = \int_a^b F(x)dx$$

なので、結果的に重積分 I は 1 変数の積分とみなすことができる。

区間 [a,b] を n 等分する台形公式 T_n の場合は、$F_i = f(x_i)$ とすると (22) 式より、

$$I \sim T_n = \frac{h}{2}\{F_0 + F_n + 2(F_1 + F_2 + \cdots + F_{n-1})\}, h = \frac{b-a}{n}$$

となる。あとは F_i を求められれば、I の近似値を求めることができるが、

$$F(x_i) = \int_{\phi(x_i)}^{\psi(x_i)} f(x_i, y)dy$$

なので、$F(x_i)$ は 1 変数関数 $f(x_i, y)$ の積分として求められる。

したがって、台形公式の場合は、$[\phi(x_i),\psi(x_i)]$ を $\phi(x_i) = y_0 < y_1 \cdots < y_m = \psi(x_i)$ と m 等分し、$k = \frac{\psi(x_i)-\phi(x_i)}{m}$ とおくと、

$$F(x_i) = \frac{k}{2}\{g_0 + g_m + 2(g_1 + g_2 + \cdots + g_{m-1})\}$$

と求められる。ただし、$g_j = f(x_i, y_j)$ である。

2.4 Simulated Annealing 法

Simulated Annealing 法 (SA 法) は**「乱数を用いて配位関数の値を変化させ、その配位関数で記述されるコスト関数の値が最小になるように計算を繰り返す手法」**である。この手法は巡回セールスマン問題や数値モデルにおける結晶構造の解析などの ” 最適化問題 ” の解法として使用される。この章で SA 法の基礎について解説を行う。

本研究では種々のラグランジアンで記述される物理系の解を得るために SA 法を用いる。この手法を利用して物理系の解を求める方針は以下のようになる。

1. ラグランジアンからエネルギー汎関数を求める。
2. エネルギー汎関数の従属変数である配位関数 (スカラー場、ゲージ場など) を乱数を用いて変化させる。
3. 2. を繰り返し、エネルギー汎関数が十分下がるようにする。
4. 得られた配位関数が数値計算で得られる解である。

この手法で解を求める際には、どのようにエネルギーを下げていくかを定める**アニーリングスケジュール**が重要となる。最も単純なスケジュールはひたすらエネルギーが下がるように計算を進行させることである。この局所探索手法を**山登り法**という。山登り法は解の収束性が最も速い手法だが、初期配位からエネルギーを下げる変化しか許さないので解が局所安定解に陥る可能性が非常に高い。我々は大域的安定解を得られるようなスケジュールを設定し、物理系の最適解を得るように数値計算を行わなければならない。では、どのようなスケジュールを組めばいいのだろうか？　実は、あらゆる場合に確実なスケジュールというものは存在しない。そのため、我々は try&error によって大域的安定解を得られるようなスケジュールを作らなければならない。

以上で述べたことを概念図にすると以下のようになる。

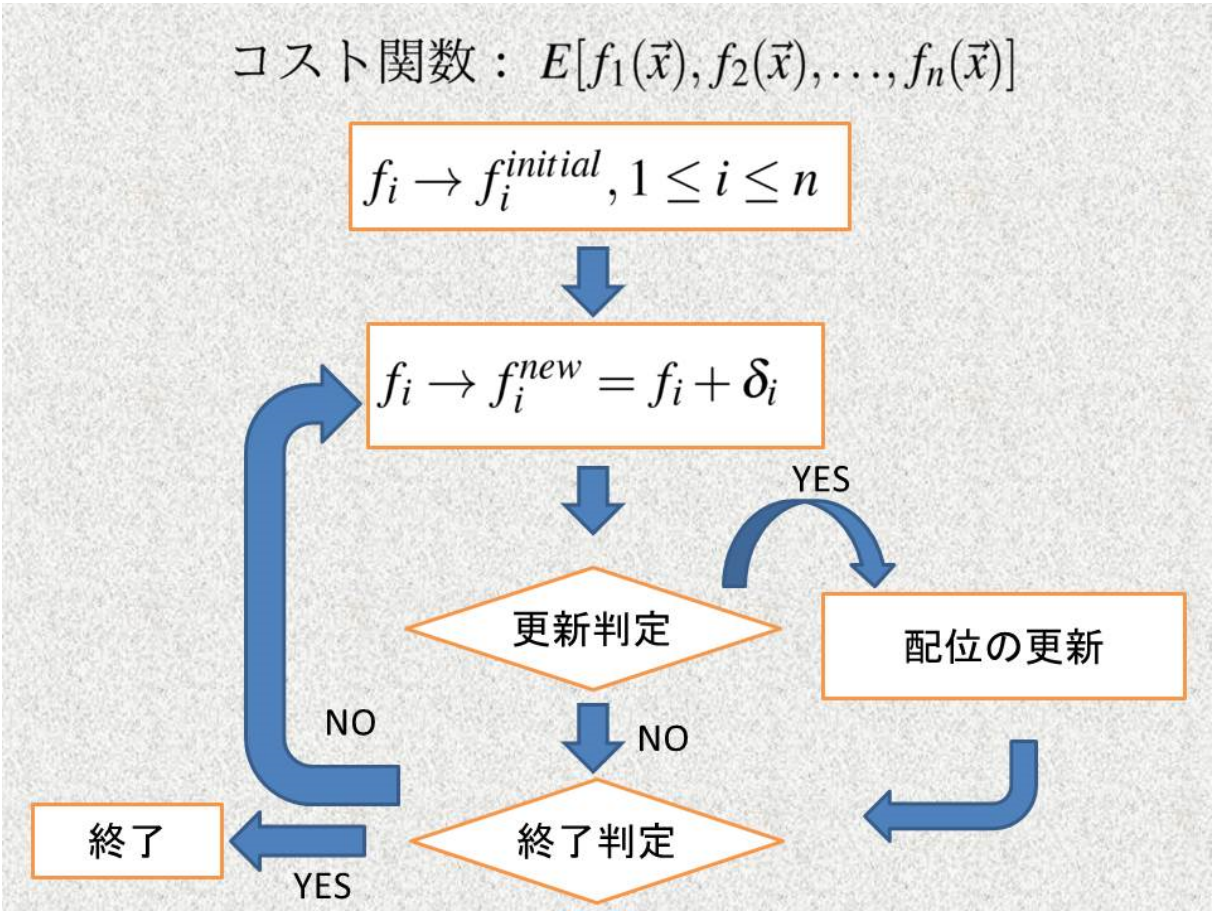


図 1: Simulated Annealing 法の模式図

この概念図を元に SA 法のアルゴリズムについて主要部分毎に解説していく。以下に述べる処理を実装し、読者が計算したいコスト関数についての計算を行えば適切な解の配位を得るこ

とができるはずである。

2.4.1 メッシュ、計算領域の設定

どのような数値計算においても基本的なことだが、計算領域とメッシュの設定を最初に行わなければならない。SA 法の場合、SOR 法やシューティング法などに代表される微分方程式の差分解法に比べて計算にかかる時間が膨大なものとなる傾向がある。そのため、まずはメッシュ量を少なめに設定し、少しずつメッシュ量を増やしていくとよい。解析したい物理系を十分表現できるメッシュ量が確保されていることを確認し数値計算を行うことを勧める。もちろん、メッシュ量が多ければ多いほどより多くの情報を内包することができるため計算精度のよい解を得ることができる。だが、安易にメッシュ量を増やしすぎるのは愚かな行為である。

次に計算領域の設定について。計算したい物理的状況により計算領域の設定は異なるのだが、主に次の 2 つの場合がトポロジカルソリトン模型を用いて行う計算であろう。

1. 無限系におけるスケール変換に不変なソリトン解の計算

2. 現象論への応用のため有限領域でのソリトン解の計算

まず、1. の場合について。この場合は計算領域についてはあまり気にする必要はない。本来無限系のソリトン解は後の章で述べるトポロジカルチャージと呼ばれる保存量を持ち、境界でエネルギーを持たないという要請により定義される。このようなソリトン解はスケール変換に対して不変な解構造を持つ。つまり、計算領域の大きさによる物理的影響は存在しない。そのため、有限領域であることにより生じる境界の効果が効いてこないように領域の設定をしてやればよい。後の章で述べるデリックの定理を満たすように領域を変更しながら計算をしてやれば数値計算に必要な時間を大幅に短縮するとともに、有限領域の効果を無視した計算を行うことができる。

次に、2. の場合について。この場合は計算領域の大きさは注意深く設定しなくてはならない。なぜなら、有限系での物理には領域の端の効果が作用するからである。ディリクレ境界条件、ノイマン境界条件、周期境界条件などの境界条件の設定及びソリトン解の大きさなどを支配するパラメータの設定が物理系に効いてくる。これらの効果はサイズ効果と呼ばれるものである。これにより、無限系のソリトン解では起こらなかったような解の変形などが起こる。現実の物質では格子構造や端の効果、非等方性などにより無限系では描写できない様々な物理現象が生じる。そのため、サイズ効果を取り入れた物理系を解析したい場合は (2.) の有限系で数値計算を行わなければならない。

2.4.2 初期配位の設定

n 個の配位関数によりコスト関数が記述される、つまり $E = E[f_1(\vec{x}), f_2(\vec{x}), \ldots, f_n(\vec{x})]$ と表せるとき、我々は関数 $f_1(\vec{x}), \ldots, f_n(\vec{x})$ のそれぞれの初期配位を設定してやる必要がある。本来 SA 法による計算結果は初期配位には依存しない。なぜなら、この手法は乱数により配位関数を

更新していくため、初期配位がどのような値であろうとも適切な大域解にたどり着くことができるからである。ここでは、我々はトポロジカルソリトン模型の解析にこの手法を用いていくことになる。実はこの場合には初期配位がどのような値であってもよいわけではない。トポロジカルチャージは場の配位関数により定義され、保存されるため初期配位は解析したいチャージ数を持つように設定しなければならない。

2.4.3 遷移幅の設定

SA 法では、定められた遷移幅で関数 $f_1(\vec{x}), \ldots, f_n(\vec{x})$ の値を変化させる。この遷移幅の設定に特別な決まりはない。解を表現するために必要な精度が保証される遷移幅を設定すればよい。ここでは、実際に私が用いた遷移幅の設定方法を具体例を用いて記載する。

まず、コスト関数 E が $E = E[f_1(\vec{x})]$ で表現できるとする。このとき、変化させる配位関数は f_1 のみである。f_1 の定義域を $[0 : \pi]$ とする。このとき、遷移前の配位関数を f_1^{old}, 遷移後の配位関数を f_1^{new} とし、遷移幅 δ_1 を用いて、

$$f_1^{new} = f_1^{old} \pm \delta_1 \tag{23}$$

となる。次に、この計算と配位の更新処理を繰り返すことで関数 $f_1(\vec{x})$ を適切な解にしていくことを考える。この場合遷移幅 δ_1 そのものが解を表現する解空間の精度になる。つまり、$\delta_1 = \frac{\pi}{128}$ と設定した場合は関数 $f(\vec{x})$ の解空間は 128 分割される。この分割数が多ければ多いほど解の精度が上がる。しかし、分割数の増加に伴い、計算終了に要する時間が増えてしまう。分割数が少なければ少ないほど計算終了に必要な時間が減少するが、解を正しく表現できなくなっていく。そのため、遷移幅の正しい決定法は読者が行いたい数値計算に依存する。解の精度を増加させることと計算時間を短縮することは相反するので数値計算屋のバランス感覚が必要となる。配位関数が複数個ある場合は同様のことをそれぞれの配位関数に行えばよい。

2.4.4 配位更新処理の設定

SA 法において最も重要で難しいのがこの配位更新処理の設定である。元々Annealing とは、金属工学における焼きなましのことである。焼きなましは、金属材料を熱した後で徐々に冷やし、結晶を成長させてその欠陥を減らす作業である。熱によって原子は初期の位置（内部エネルギーがローカルな極小状態）から離され、よりエネルギーの高い状態をうろつく。ゆっくり冷却することで、原子は初期状態よりも内部エネルギーがさらに極小な状態を得る可能性が多くなる。　この焼きなましを数値計算であるコスト関数に対して行うという考えで Simulated Annealing 法が生まれた。焼きなましのアナロジーとして考えると我々は数値計算における**温度**を定義する必要がある。この温度が高いときはコスト関数が上がる遷移を起こし、温度が低いときはコスト関数が下がる遷移を起こすように設定する。これにより、配位関数 $f_i(\vec{x})$ が局所的な解に留まることを防ぎ、大域的最適解を得ることが可能となる。

まず、最も一般的な温度と更新判定処理の定め方を述べる。温度を T, 更新前のコスト関数を E_{old}, 更新後のコスト関数を E_{new}, 定数を k_B とすると、確率関数 q を以下のように表現することができる。

$$q = \exp(-\frac{E_{new} - E_{old}}{k_B T}). \tag{24}$$

このとき、$0 \leq q \leq 1$ となる。ここで乱数 r を $0 \leq r \leq 1$ となるように設定する。$E_{new} - E_{old}$ の値が負の場合は、$q = 1$ を代入する。そして、$r \leq q$ のとき以下の処理を行う。

$$E_{old} = E_{new} \tag{25}$$

つまり、E_{new} が E_{old} よりも小さい場合は必ず E_{new} を E_{old} とすることで、コスト関数を下げる。もし、E_{new} が E_{old} よりも大きく、かつ確率 q が乱数 r より大きい場合は $E_{new} = E_{old}$ とする。この処理により配位関数が大域的最適解に行き着くことができる。この場合温度 T が高いほど、$E_{new} - E_{old}$ の差が小さいほど E_{new} を新たな E_{old} として採用する確率が上がる。したがって、我々は温度を高くすることで配位関数を擾乱させ、様々な値をとらせることができる。温度を徐々に下げていくことでコスト関数が下がるように配位関数を変化させ、局所安定解に導くことができる。以上のような仕組みにより、SA 法を構成する。だが、この手法では温度の下げ方の設定が難しい。真の最適解にたどりつく前に温度が下がりきってしまう可能性があるからだ。そうならないようにアニーリングスケジュールを組むためには相当の試行錯誤が必要となる。そのため、我々はより確実に大域的最適解を得るためにこのようなアルゴリズムとは異なる更新判定処理を採用した。ここからはその手法について解説していく。

先の手法では、温度 T を定義し、確率関数を温度の指数関数として定義した。この方法では解が局所解にいるのか、それとも最適解にいるのかの判断が難しく、計算結果が温度の下げ方に依存してしまう。そこで、我々は「大域的最適解に解がたどりついたのを確認してから温度を下げる」というスケジュールを採用する。ここでは先の手法で定義した温度や確率などを用いない。そのかわり、我々はアニーリングスケジュールに用いるための変数を以下のように定義する。

1. $count$: 全計算回数
2. $tcount$: アニーリング処理を終了させるために必要な計算回数
3. N_{up}: N_{up} 回に 1 回コストが上がる方向への遷移を許す。
4. tmp: 1 回の遷移処理において tmp 個の配位関数 (種類と位置) を選択して δ_i 増加もしくは減少させる。
5. $limit$: $E_{new} \leq E_{old} \times limit$ の場合、遷移を許可する。例えば $limit = 1.1$ の場合、更新前のコストの 1.1 倍までの増加なら遷移を許可する。

我々の新たなアニーリングスケジュールでは、「tmp 個の配位関数を遷移させ、N_{up} 回に 1 回コストの判定処理を行い、遷移を許可するかどうかを決める。この処理を $tcount$ 回繰り返したあとで $limit = 1.0$ とおく。そして $count$ 回の計算を繰り返したところで計算を終了する」

以上のようなスケジュールを用いることで、一定量のコスト増加を許可して配位関数を遷移させ、大域的最適解にいたることができる。また、上記の変数を複数個用意することでより細かいスケジュールを組むことが可能である。

2.5 疑似乱数生成器

ここまでの節でSA法の基本的なアルゴリズムの解説を行った。アルゴリズム中に乱数を用いる部分があったが、SA法を扱う際にはどのような乱数を用いるかが非常に重要である。そもそも、コンピュータでは本物の乱数を作ることができない。コンピュータ上で作成される乱数は疑似乱数である。擬似乱数とは、乱数列のように見えるが、実際には確定的な計算によって求めている数列に含まれる数を指す。本物の乱数であれば次にどんな値が出るかを予測することは不可能である。しかし、疑似乱数の場合は計算により作成される。つまり、生成法と内部状態が既知であれば、予測可能となる。SA法では幾度となく乱数を生成し、その乱数により配位を遷移させる。つまり、計算結果は乱数に完全に依存する。そのため、精度の低い乱数列(パターンが浮き彫りになるような乱数列)を使用してしまうと正しい数値解を得ることができない。

C言語で乱数を用いる際、最も手軽に乱数を生成できるものとしてrand関数が挙げられる。しかし、rand関数は非常に精度の低い乱数列を元にして乱数を生成する。この関数を用いて乱数を生成し、SA法に使用しても多くの場合正しい解が得られない。

そのため、我々は精度の高い疑似乱数を生成できる乱数生成器を採用する必要がある。本研究では乱数生成器として**メルセンヌツイスタ**を採用した。メルセンヌツイスタは松本眞氏と西村拓士氏によって開発された高精度擬似乱数生成器である。この乱数生成器を用いて計算を行うためには以下に示すヘッダファイル"mersenne.h"をインクルードしなければならない。以下のヘッダファイルの元々のソースコードは公式ホームページ http://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/SFMT/index-jp.html でダウンロードできる。

```
//C++でメルセンヌツイスタをヘッダファイルにしてみる

/*
   A C-program for MT19937, with initialization improved 2002/1/26.
   Coded by Takuji Nishimura and Makoto Matsumoto.

   Before using, initialize the state by using init_genrand(seed)
   or init_by_array(init_key, key_length).

   Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
```

```
   modification, are permitted provided that the following conditions
   are met:

     1. Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.

     2. Redistributions in binary form must reproduce the above copyright
        notice, this list of conditions and the following disclaimer in the
        documentation and/or other materials provided with the distribution.

     3. The names of its contributors may not be used to endorse or promote
        products derived from this software without specific prior written
        permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
   A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
   CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
   PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
   PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
   LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
   NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
   SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


   Any feedback is very welcome.
   http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html
   email: m-mat @ math.sci.hiroshima-u.ac.jp (remove space)
*/

#include <iostream>

/* Period parameters */
#define N 624
#define M 397
#define MATRIX_A 0x9908b0dfUL   /* constant vector a */
```

```
#define UPPER_MASK 0x80000000UL /* most significant w-r bits */
#define LOWER_MASK 0x7fffffffUL /* least significant r bits */

static unsigned long mt[N]; /* the array for the state vector  */
static int mti=N+1; /* mti==N+1 means mt[N] is not initialized */

/* initializes mt[N] with a seed */
void init_genrand(unsigned long s)
{
    mt[0]= s & 0xffffffffUL;
    for (mti=1; mti<N; mti++) {
        mt[mti] =
    (1812433253UL * (mt[mti-1] ^ (mt[mti-1] >> 30)) + mti);
        /* See Knuth TAOCP Vol2. 3rd Ed. P.106 for multiplier. */
        /* In the previous versions, MSBs of the seed affect   */
        /* only MSBs of the array mt[].                        */
        /* 2002/01/09 modified by Makoto Matsumoto             */
        mt[mti] &= 0xffffffffUL;
        /* for >32 bit machines */
    }
}

/* initialize by an array with array-length */
/* init_key is the array for initializing keys */
/* key_length is its length */
/* slight change for C++, 2004/2/26 */
void init_by_array(unsigned long init_key[], int key_length)
{
    int i, j, k;
    init_genrand(19650218UL);
    i=1; j=0;
    k = (N>key_length ? N : key_length);
    for (; k; k--) {
        mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >> 30)) * 1664525UL))
          + init_key[j] + j; /* non linear */
        mt[i] &= 0xffffffffUL; /* for WORDSIZE > 32 machines */
        i++; j++;
        if (i>=N) { mt[0] = mt[N-1]; i=1; }
```

```
        if (j>=key_length) j=0;
    }
    for (k=N-1; k; k--) {
        mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >> 30)) * 1566083941UL))
          - i; /* non linear */
        mt[i] &= 0xffffffffUL; /* for WORDSIZE > 32 machines */
        i++;
        if (i>=N) { mt[0] = mt[N-1]; i=1; }
    }

    mt[0] = 0x80000000UL; /* MSB is 1; assuring non-zero initial array */
}

/* generates a random number on [0,0xffffffff]-interval */
unsigned long genrand_int32(void)
{
    unsigned long y;
    static unsigned long mag01[2]={0x0UL, MATRIX_A};
    /* mag01[x] = x * MATRIX_A  for x=0,1 */

    if (mti >= N) { /* generate N words at one time */
        int kk;

        if (mti == N+1)   /* if init_genrand() has not been called, */
            init_genrand(5489UL); /* a default initial seed is used */

        for (kk=0;kk<N-M;kk++) {
            y = (mt[kk]&UPPER_MASK)|(mt[kk+1]&LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (y >> 1) ^ mag01[y & 0x1UL];
        }
        for (;kk<N-1;kk++) {
            y = (mt[kk]&UPPER_MASK)|(mt[kk+1]&LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (y >> 1) ^ mag01[y & 0x1UL];
        }
        y = (mt[N-1]&UPPER_MASK)|(mt[0]&LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >> 1) ^ mag01[y & 0x1UL];

        mti = 0;
```

```
    }

    y = mt[mti++];

    /* Tempering */
    y ^= (y >> 11);
    y ^= (y << 7) & 0x9d2c5680UL;
    y ^= (y << 15) & 0xefc60000UL;
    y ^= (y >> 18);

    return y;
}

/* generates a random number on [0,0x7fffffff]-interval */
long genrand_int31(void)
{
    return (long)(genrand_int32()>>1);
}

/* generates a random number on [0,1]-real-interval */
double genrand_real1(void)
{
    return genrand_int32()*(1.0/4294967295.0);
    /* divided by 2^32-1 */
}

/* generates a random number on [0,1)-real-interval */
double genrand_real2(void)
{
    return genrand_int32()*(1.0/4294967296.0);
    /* divided by 2^32 */
}

/* generates a random number on (0,1)-real-interval */
double genrand_real3(void)
{
    return (((double)genrand_int32()) + 0.5)*(1.0/4294967296.0);
    /* divided by 2^32 */
```

```
}

/* generates a random number on [0,1) with 53-bit resolution*/
double genrand_res53(void)
{
    unsigned long a=genrand_int32()>>5, b=genrand_int32()>>6;
    return(a*67108864.0+b)*(1.0/9007199254740992.0);
}
```

このメルセンヌツイスタは非常に強力な乱数生成器であり、本研究において必要となる疑似乱数として十分な精度を発揮してくれる。

2.6 まとめ

この章では 3 章以降で導入するトポロジカルソリトン模型の解を得るために必要な数値計算手法を解説した。まず、導関数の定義とテイラー展開を用いて、差分化を行うことで数値微分の公式を得た。次に、連立 1 次方程式の解法を紹介した。ここで紹介した SOR 法を差分化した微分方程式系に用いることで我々は数値解を得ることができる。次に紹介した数値積分法はエネルギーなどの物理量を見積もるのに必須の計算法である。最後に、本論文の要でもある SA 法について解説した。SA 法は多くの数値計算処理を必要とするが非常に強力な解析手法である。「乱数を用いて配位関数を更新し、コスト関数を下げることで局所解に陥らずに大域的最適解を得る」 全ての計算処理はこの目的のために行われる。(2.4.1) 章から (2.4.4) 章までで必要な処理の解説を行った。

3 章以降では、2 章で解説した数値計算手法を用いてトポロジカルソリトン模型の解析を行っていく。

3　トポロジカルソリトンと kink 解

まずこの章では**ソリトン**、**トポロジカルソリトン**及び**ビリアルの定理**について解説する。そのあとで最も簡単なトポロジカルソリトン模型である Sine-Gorden 模型の解である kink 解を SOR 法と Simulated Annealing 法により求める。この章は次の章以降を読む際の基礎になるので実際の数値解析も含めて大いに参考にしていただきたい。

3.1　ソリトン

まず、ソリトンとはなんだろうか？ 通常、小石などを川や海に投げこんで発生した浅水波は徐々に減衰していき、最後には消失してしまう。ところが、1834 年に Russell はエディンバラの運河で船首が作った浅水波が振幅を維持したまま数マイルの距離を進む現象を目撃した。この波こそがソリトンである。この浅水波の解を記述するモデルを KDV 方程式と呼び、以下の式で表現される。

$$\frac{\partial u}{\partial t} + 6u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0, \tag{26}$$

ここで、時間変数 t, 空間変数 x, 波の振幅 $u(t,x)$ である。このような方程式で記述されるソリトン解は波の振幅や速度を変えない。また、ソリトン同士の衝突に対しても安定である。

3.2　トポロジカルソリトン

素粒子物理学における研究で我々は上記のソリトンが**トポロジカルチャージ**と呼ばれる保存量を有する模型を扱う。トポロジカルチャージとはホモトピー群と呼ばれる群構造によって定義される無限遠方の位相の性質から決まる保存量であり、特別な場合を除き整数の値をとる。無限の大きさを持つ空間において、無限大のエネルギーを与えない限りトポロジカルチャージの値が変わることはない。この性質と次の節で述べるビリアルの定理から、トポロジカルソリトン模型は有限のエネルギーを持つ局在した安定な解構造を持つことができる。この性質によりトポロジカルソリトン解を粒子と考えることができるので、この類の模型の研究が盛んに行われるようになった。

3.3　ビリアルの定理

ソリトン解 $\vec{\phi}_{cl}$ は安定解でなければならない。そのため、微小変化 $\vec{\phi}_{cl} + \delta\vec{\phi}$ に対して、エネルギー $E[\vec{\phi}]$ は $E[\vec{\phi}_{cl}] \leq E[\vec{\phi}_{cl} + \delta\vec{\phi}]$ を満たさなければならない。このことから、解のスケール変換に対してソリトン解は安定でなければならない。「トポロジカルソリトン模型のエネルギー汎関数 $E[\vec{\phi}]$ がスケール変換に対して極小を持たなければソリトン解は存在しない」ということがここから導かれる。この定理を**ビリアルの定理**という。ここで、時間 1 次元、空間 D 次元の模型を記述するエネルギー密度を以下のように定義する。

$$\mathscr{H}[t,x,\phi(t,x)] = \frac{1}{2}\partial_\mu\vec{\phi}\cdot\partial^\mu\vec{\phi} + U(\vec{\phi}(t,x)) \tag{27}$$

まず、2 階微分を含む項と含まない項をそれぞれ、ϵ_2, ϵ_0 と定義する。このとき、$\epsilon_2 = \frac{1}{2}\partial_\mu\vec{\phi}\cdot\partial^\mu\vec{\phi}$,$\epsilon_0 = U(\vec{\phi}(t,x))$ である。このとき、スケール変換前のエネルギー汎関数は、

$$\begin{aligned} E[\vec{\phi}] &= \int d^D x\Big[\frac{1}{2}\partial_\mu\vec{\phi}\cdot\partial^\mu\vec{\phi} + U(\vec{\phi}(t,x))\Big] \\ &= \int d^D x\Big[\epsilon_2 + \epsilon_0\Big] \\ &= E_2 + E_0 \end{aligned} \tag{28}$$

ここで、$E_2 = \int \epsilon_2 d^D x, E_0 = \int \epsilon_0 d^D x$ となる。$\vec{\phi}$ を静的な解とするとき、空間スケールを λ 倍するスケール変換を行う。これにより、$x \to \lambda x$ となる。このとき、x の 2 階微分を含む項 ϵ_2 は $\epsilon_2 \to \lambda^{-2}\epsilon_2$ となり、積分は $\int d^D x \to \lambda^D \int d^D x$ となる。以上から、スケール変換したあとのエネルギー汎関数は、

$$E[\phi(\lambda x)] = \int d^D x\Big[\lambda^{D-2}\epsilon_2 + \lambda^D\epsilon_0\Big] \tag{29}$$

となる。(29) 式において、$D = 1$ のとき、エネルギー汎関数 E は $E[\vec{\phi}] = \lambda^{-1}E_2 + \lambda E_0$ となり、

$$\lambda = \sqrt{\frac{E_2}{E_0}} \tag{30}$$

となり、極値を持つ。したがって $D = 1$ の場合、(27) 式のエネルギー密度で記述される模型は静的な安定解を有する可能性がある。(安定解を持つと断言はできない。)$D = 2, 3$ などの場合は極値を持つことがないので静的な安定解を持つことはできない。安定解を得るためには、より高次の微分を有する項が模型に必要となる。そのような観点から構築された模型として後の章で述べる Abelian-Higgs 模型や Baby-Skyrme 模型などが知られている。この次の節で述べる Sine-Gordon 模型は $D = 1$ の場合の模型である。

3.4 Sine-Gordon 模型

この章では、1 次元空間の古典的ソリトン解である kink 解を有する Sine-Gordon 模型について解説する。1 変数スカラー場 $\phi(t,x)$ を用いてラグランジアン密度 $\mathscr{L}$ を

$$\begin{aligned} \mathscr{L} &= \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - V(\phi) \\ &= \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - \frac{\lambda}{4}(\phi^2 - v^2)^2 \end{aligned} \tag{31}$$

ここで、$v = \sqrt{\frac{m^2}{\lambda}}$ であり、m^2, λ は正の定数である。このラグランジアン密度は $\phi \to -\phi$ に対して不変である。しかし、この対称性はポテンシャル V により自発的に破れており、$V(\phi)$ は $\phi = \pm v$ で縮退した最小値を持つ。また、前節のビリアルの定理から、このラグランジアン密度で記述される模型はソリトン解を有する可能性がある。

ここで、古典場の理論におけるオイラーラグランジュ方程式は

$$\frac{\partial \mathscr{L}}{\partial \phi} = \partial_\mu \Big(\frac{\partial \mathscr{L}}{\partial(\partial_\mu \phi)} \Big) \tag{32}$$

である。(32) 式に Sine-Gordon 模型のラグランジアン密度 (31) 式を代入すると、

$$\frac{d^2\phi}{d^2t} - \frac{d^2\phi}{d^2x} = -\lambda(\phi^2 - v^2)\phi. \tag{33}$$

となる。ここで静的な解を仮定すると、

$$\frac{d^2\phi}{d^2x} = \lambda(\phi^2 - v^2)\phi. \tag{34}$$

となる。また、静的なエネルギー汎関数 $E[\phi]$ は、

$$E[\phi] = \int dx \Big[\frac{1}{2}\partial_\mu\phi\partial^\mu\phi + \frac{\lambda}{4}(\phi^2 - v^2)^2 \Big] \tag{35}$$

この模型がソリトン解を持つためには、ソリトン解の性質から、無限遠方でエネルギー密度が 0 にならなければならない。(35) 式の形から、無限遠方において $\phi = \pm\sqrt{\frac{m^2}{\lambda}} = \pm v$ であることがわかる。$x \to -\infty$ で $\phi = -v$、$x \to \infty$ で $\phi = v$ となる解を **kink 解**といい、$x \to -\infty$ で $\phi = v$、$x \to \infty$ で $\phi = -v$ となる解を**反 kink 解**という。ソリトン解は局在した有限のエネルギー密度を持つという性質から Sine-Gordon 模型における無限遠方の境界条件を知ることができた。この境界条件の下で、(33) 式を解くと以下の解が得られる。

$$\phi(x) = \pm v \tanh\Big[\frac{m}{\sqrt{2}}(x - x_0) \Big], \tag{36}$$

ここで、x_0 は定数である。この式の形から、kink 解と反 kink 解は並進対称性があることがわかる。後の節では数値計算によってこの解析解と同じ形状の解を得ることができるかどうか確かめる。

また、このトポロジカルソリトン模型におけるトポロジカルチャージ Q は以下の式で定義される。

$$\begin{aligned} Q &= \frac{1}{2v}\int_{-\infty}^{\infty} \partial_1\phi \\ &= \frac{1}{2v}[\phi(\infty) - \phi(-\infty)]. \end{aligned} \tag{37}$$

kink 解の場合は $Q = 1$, 反キンク解の場合は $Q = -1$ となる。

このソリトン解は物理的にも非常に有用な解である。ブレーン模型などの素粒子物理への応用のみならず、ジョゼフソン結合している超伝導体を貫通する平行な 2 つの磁束の動力学や量子ホール効果を起こす 2 層モデルなどの物性物理への応用にも用いられている。非常に興味深い模型だが、本論文ではこれ以上の内容には立ち入らない。

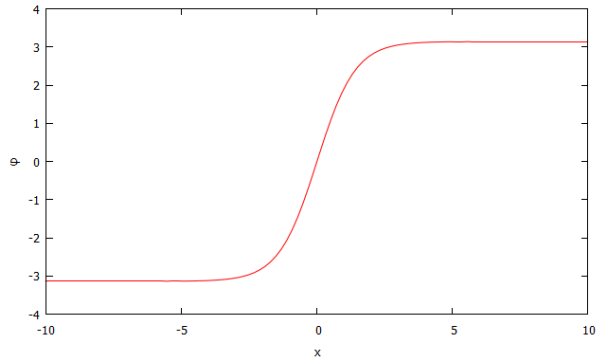


図 2: $v = \pi, m = 1, x_0 = 0$ のときの kink 解

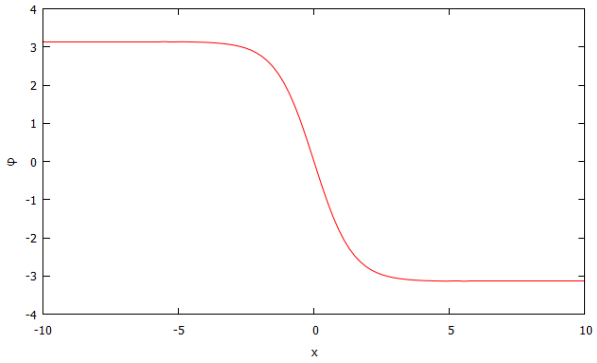


図 3: $v = \pi, m = 1, x_0 = 0$ のときの反 kink 解

3.5 SOR 法で kink 解を求める

この節では、前節で解析的に求めた kink 解を (2.3) 章で解説した SOR 法を用いて数値的に求める。解くべき方程式は (34) 式を差分化した

$$\frac{\phi_{i+1} - 2\phi_i - \phi_{i-1}}{\Delta x^2} = \lambda(\phi_i^2 - v^2)\phi_i. \tag{38}$$

である。

kink 解を得るため、計算領域の左端で $\phi = -v$, 右端で $\phi = v$ とおく。また、並進対称性による解の自由度をなくすために、$x = x_0$ で $\phi = 0$ とおく。以上から、座標空間を n+1 分割された差分方程式 $(i = 0, 1 \ldots, n-1, n)$ における境界条件は、

$$\begin{aligned} \phi_0 &= -v \\ \phi_{\frac{n}{2}} &= 0 \\ \phi_n &= v \end{aligned} \tag{39}$$

となる。

この方程式を $v = \pi, m = 1, x_0 = 0$ のパラメータで SOR 法で解くと次ページの図が得られる。

数値的に得られた解と解析的に得られた解は一致していることがわかる。また、得られた数値解からエネルギー密度を計算した。エネルギーが有限領域で局在していることがわかる。

以下がソースコードである。

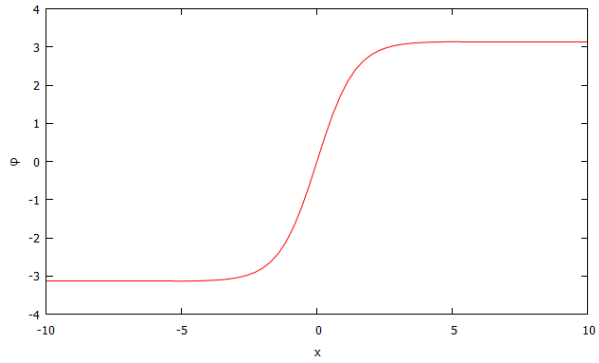


図 4: SOR 数値解

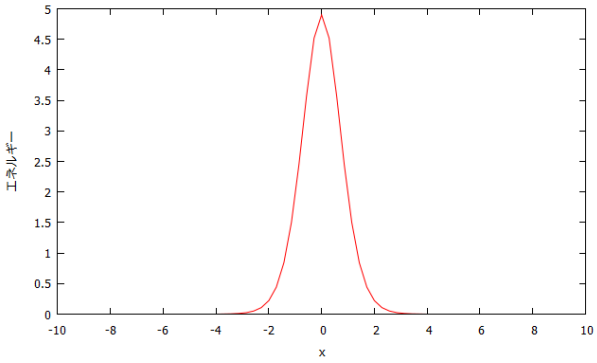


図 5: エネルギー密度

```
//SOR 法で kink 解を出す

#include <cstdio>
#include <cmath>
#include <ctime>
#include <cstdlib>

#define PI 3.1415926535
#define MESH 70 //メッシュの分割数
#define MAX 10 //領域の右端
#define MIN -10 //領域の左端
#define START 40000 //合計計算回数
#define KAKUNIN 2000 //残差とエネルギーなどをこの回数毎に出力

using namespace std;

//初期配位の定義
double func_phi(double x,double v);

int main()
{
  int i,j,k;//ループ用処理変数
  double h,rh;
  double mmax=MAX;
  double mmin=MIN;
```

```
int count;//計算回数
FILE *data;
double phi[MESH+1];//配位Φ'
double tmp[3];//(1+従属変数の数) ×扱いたい場の数分用意する。
double cphi;//計算用Φ
double cdphi;//計算用Φ'
double cddphi;//計算用Φ''
double zd[MESH+1];//残差密度
double omega=0.001;//加速パラメータ
double Ed[MESH+1];//エネルギー密度
double E;//エネルギー
double zan;//残差
double charge;//トポロジカルチャージ

double lam=1.0/PI/PI;//λ
double v=PI;

E=0.0;
zan=0.0;
h=(mmax-mmin)/MESH;
rh=1.0/h;

//初期配位を定義
for(i=MESH;i>=0;i--){
  phi[i]=func_phi(mmin+i*h, v);
  zd[i]=0.0;
}
//境界条件の設定
phi[0]=-v;
phi[MESH/2]=0;
phi[MESH]=v;

//実際の SOR 計算処理
for(count=START; count>0; count--){
  //値確認用
  if(count%KAKUNIN==0){
    data=fopen("kink.txt","w"); //計算結果保存用
    for(i=MESH;i>=0;i--){
```

```
      fprintf(data,"%lf\t%lf\t%lf\n", mmin+i*h, phi[i], zd[i]);
    }
    fclose(data);
    data=fopen("kink_energy.txt","w"); //計算結果保存用
    for(i=MESH-1;i>=0;i--){
      fprintf(data,"%lf\t%lf\n", mmin+i*h, Ed[i]);
    }
    fclose(data);
    printf("%d/%d\n%lf\t%lf\t%lf\n", count, START, E, zan, charge);
  }

  //残差を用いて配位の再設定
  for(i=MESH;i>=0;i--){
    phi[i]=phi[i]+omega*zd[i];
  }
  //境界条件の設定
  phi[0]=-v;
  phi[MESH/2]=0;
  phi[MESH]=v;

  charge=0.0;
  E=0.0;
  zan=0.0;
  //このループの中には両端の計算は入っていない
  for(i=MESH-1;i>=1;i--){
    tmp[0]=phi[i];//現在位置のΦ
    tmp[1]=phi[i-1];//右のΦ
    tmp[2]=phi[i+1];//左のΦ
    cphi=tmp[0];
    cdphi=0.5*(tmp[2]-tmp[1])*rh;
    cddphi=(tmp[1]-2*tmp[0]+tmp[2])*rh*rh;
    zd[i]=cddphi-lam*(cphi*cphi-v*v)*cphi;
    Ed[i]=(0.5*cdphi*cdphi+lam/4.0*(cphi*cphi-v*v)*(cphi*cphi-v*v));
    charge+=cdphi/(2*v)*h;
    E+=Ed[i]*h;
    zan+=zd[i];
  }
}
```

```
  return 0;
}

//初期配位の定義
double func_phi(double x,double v)
{
  return v*x;
}
```

3.6 Simulated Annealing 法で kink 解を求める

前章では SOR 法で kink 解を数値的に求めた。ここでは、SA 法により kink 解を求める。境界条件、メッシュ数、パラメータなどは SOR 法のときと同じにする。このような設定で SA 法で計算された数値解は以下の図で示される。

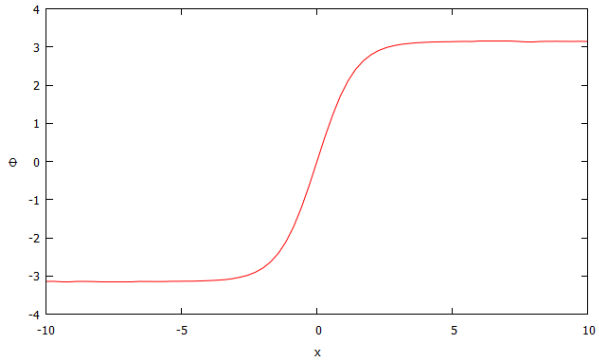


図 6: Simulated Annealing 数値解

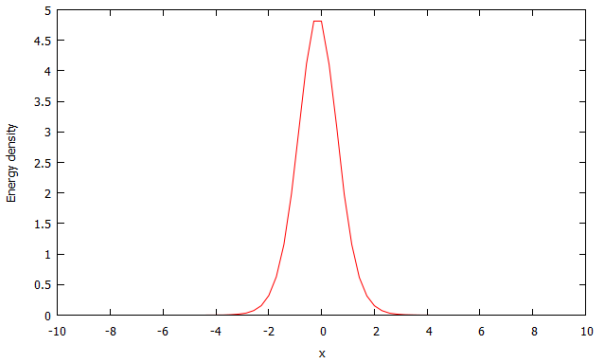


図 7: エネルギー密度

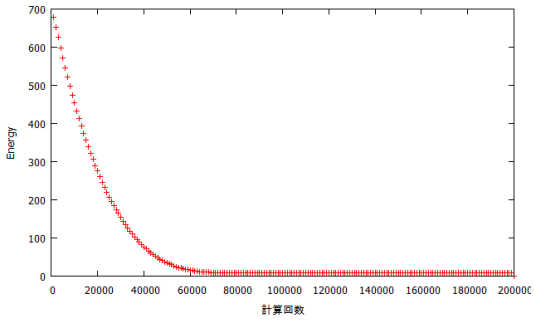


図 8: エネルギーの推移

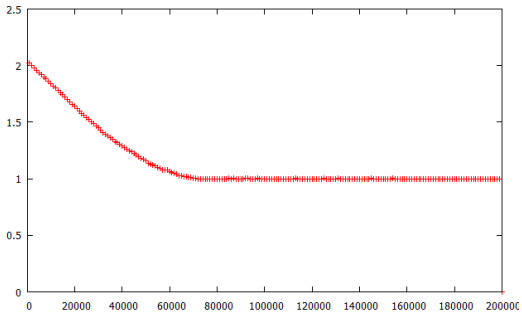


図 9: チャージの推移

SA 法により SOR 法と同じ解が得られていることがわかる。初期配位はかなり悪いが数値計算の過程でエネルギー、チャージ共に変化していき、最終的にはチャージ 1 の kink 解が得られている。局所的な安定解に陥ることなく数値計算が行われていることがわかる。数値計算に用いたソースコードを以下に掲載する。このソースコードを見ていただけるとわかるように、変数の定義をグローバル変数として行っている箇所が多々見られる。グローバル変数による変数定義はソースコードの拡張性、汎用性の観点から本来なら避けねばならない。しかし、一般的には関数を呼び出す際に多くの引数をとると数値計算速度が落ちてしまう。SA 法では数値解の計算にかかる時間が膨大である。並列化などを用いてもっとよいコーディングを目指すべきではあったが自分の稚拙な技量のためこのような見にくいコードになってしまったことをお詫びしておく。

```
//Simulated Annealing 法で kink 解を出す

#include <cstdio>
#include <cmath>
#include <ctime>
#include <cstdlib>
#include <cstring>
#include "mersenne.h"

#define PI 3.1443926584
//メッシュは奇数 !!
#define MESH 70 //メッシュの分割数
#define MIN -10.0 //x,y の最少値
#define MAX 10.0 //x,y の最大値
#define START 200001 //合計計算回数
#define TSTART 5 //TSTART 毎に 1 回コストが上がる方向への遷移を許す
#define KAKUNIN 1000 //kakunin 回毎にエネルギーとチャージの値をファイルに出力

using namespace std;

int M1_2=(MESH+1)*(MESH+1);

double h;//刻み幅の指定
double rh;//h の逆数

double tF[MESH],dx_F[MESH];//Φ場の平均値と微分
double v=PI;
double mass=1.0;
```

```
double lam=mass/v/v;//λ

double Ed[MESH];//エネルギー密度
double cd[MESH];//チャージ密度
double memo_n[START/KAKUNIN+1]; //チャージの推移を記録
double memo_E[START/KAKUNIN+1]; //エネルギーの推移を記録

struct{
  double F[MESH+1]; //配位Φ

} od,nd;

//初期配位の定義
double func_F(double x);

//重積分用の台形公式
double trapezoidal2(void);

//トポロジカルチャージ計算用積分
double trapezoidal_N(void);

//乱数処理部分呼び出し
 void ransu(void);

int main(void)
{
  double oE,nE;//古いエネルギー、新しいエネルギー
  double on,nn;//古いチャージ、新しいチャージ
  int count=START;//計算回数
  int tcount=TSTART;//この回数毎に 1 回コストが上がる方向への遷移を許す
  int dcount=0;//何回配位を取り換えたか
  int i;//ループ用
  FILE *data,*fp;
  char name[80];//場の配位のファイル名
  char name2[80];//エネルギー密度, チャージ密度などの情報
  int number=0;//保存回数 (ファイル名に保存回数の情報を入れる)

   h=(MAX-MIN)/MESH;
```

```
   rh=1.0/h;

//初期配位設定
  for(i=MESH;i>=0;i--){
      od.F[i]=func_F(MIN+i*h);
    }
//境界条件
  od.F[0]=-v;
  od.F[MESH/2]=0;
  od.F[MESH]=v;

  nd=od;

//Φ場の偏微分の定義
  for(i=MESH-1;i>=0;i--){
    dx_F[i]=(nd.F[i+1]-nd.F[i])*rh;
  }
//Φ場の平均値の定義
  for(i=MESH-1;i>=0;i--){
    tF[i]=(nd.F[i+1]+nd.F[i])*0.5;
  }

//再びエネルギー計算
  nn=trapezoidal_N();
  nE=trapezoidal2();
  oE=nE;

  on=nn;

  //計算の途中経過を記録するための名前付け
  sprintf(name,"kink_%d.txt" ,number);
  sprintf(name2,"kink_%d_Bz.txt" ,number);
  printf(name);

//計算結果保存用
  data=fopen(name,"w");
  for(i=MESH;i>=0;i--){
      fprintf(data,"%lf\t%lf\n", MIN+i*h, nd.F[i]);
```

```
}
fclose(data);
data=fopen(name2,"w");
for(i=MESH-1;i>=0;i--){
    fprintf(data,"%lf\t%lf\n", MIN+i*h,  Ed[i]);
}
fclose(data);

//ここからスカラー場を乱数で変化させていく
  init_genrand((unsigned)time(NULL));
  do{

    //確認用
    if(count%KAKUNIN==0){
      memo_E[(START-count)/KAKUNIN]=oE;
      memo_n[(START-count)/KAKUNIN]=on;
      printf("%lf %lf %d\n",oE,on,dcount);
      dcount=0;
    }
    if(count%10000==0) printf("\n 残り%d\n",count);

    //配位を乱数でゆらしてやる
    if(count>(START*0.4)){
      double tmp=M1_2*1.2;
      for(i=tmp;i>=0;i--){
         ransu();
      }
    }
    else if((count>(START*0.1))&&((count<(START*0.4)))) {
       double tmp=M1_2*1.0;
      for(i=tmp;i>=0;i--){
        ransu();
      }
    }
    else {
      double tmp=M1_2*0.8;
      for(i=tmp;i>=0;i--){
        ransu();
```

```
        }
      }

//境界条件
  nd.F[0]=-v;
  nd.F[MESH/2]=0;
  nd.F[MESH]=v;

//Φ場の偏微分の定義
  for(i=MESH-1;i>=0;i--){
    dx_F[i]=(nd.F[i+1]-nd.F[i])*rh;
  }
//Φ場の平均値の定義
  for(i=MESH-1;i>=0;i--){
    tF[i]=(nd.F[i+1]+nd.F[i])*0.5;
  }

    //トポロジカルチャージ変化
      nn=trapezoidal_N();//b/4 π のこと
      //変化した配位でエネルギー計算
      nE=trapezoidal2();

      //エネルギー比較条件緩和処理
      if((tcount==0)&&(count>(START*0.1))){
        tcount=TSTART;
        oE*=1.1;
      }
      if(nE<oE){
        oE=nE;
        on=nn;
        tcount=TSTART;
        dcount++;
        od=nd;//ここで配位全部更新
      }

      if(oE!=nE)  tcount--;

        if(count%5000==0){
```

```
        number++;
        sprintf(name,"kink_%d.txt" ,number);
        sprintf(name2,"kink_%d_Bz.txt" ,number);

        //計算結果保存用
          data=fopen(name,"w");
          for(i=MESH;i>=0;i--){
              fprintf(data,"%lf\t%lf\n", MIN+i*h, nd.F[i]);
          }
          fclose(data);
          data=fopen(name2,"w");
          for(i=MESH-1;i>=0;i--){
              fprintf(data,"%lf\t%lf\t%lf\n", MIN+i*h, Ed[i], cd[i]);
          }
          fclose(data);
       }

    count--;
    }while(count>0);

//横軸に count, 縦軸に,oE,on, のグラフを描く
  data=fopen("kink_c-E.txt","w"); //計算結果保存用
  for(i=START/KAKUNIN;i>0;i--){
    fprintf(data,"%d\t%lf\t%lf\n", i*KAKUNIN, memo_E[i], memo_n[i]);
  }
  fclose(data);

  return 0;
}

//重積分用の台形公式
double trapezoidal2(void)
{
  double S=0.0;
  double s;//台形積分の端っこの点を計算するための係数

  int i;//ループ用処理変数
```

```
  for(i=MESH-1;i>=0;i--){
    if((i==0)||(i==MESH-1)) s=0.5;
    else s=1.0;
      Ed[i]=(0.5*dx_F[i]*dx_F[i]+lam/4.0*(tF[i]*tF[i]-v*v)*(tF[i]*tF[i]-v*v))*s;
      S+=Ed[i]*h;
  }

  return S;
}
//トポロジカルチャージ求めるための積分
double trapezoidal_N(void)
{
  double S=0.0;
  double s;//台形積分の端っこの点を計算するための係数
  int i;
  for(i=MESH-1;i>=0;i--){
    if((i==0)||(i==MESH-1)) s=0.5;
    else s=1.0;
      cd[i]=(dx_F[i]/(2*v))*s*h;

      S+=cd[i];

  }
  return S;
}

 void ransu(void)
{
  int rk,ri,rj;//新乱数処理用変数
  int random;//乱数保存用

  double leng=0.001;//初期振幅パラメータ

  //離散的に乱数ゆらす
  random=genrand_int31()%((MESH-1)*(MESH-1)*2);
  rk=random%2;
  random=(random-rk)*0.5;
  ri=random%(MESH-1);
```

```
  ri+=1;
  switch(rk){
    case 0: nd.F[ri]=od.F[ri]+leng; break;
    case 1: nd.F[ri]=od.F[ri]-leng;
  }
}

//初期配位の定義
double func_F(double x)
{
  return x;
}
```

この節で行ったように、方程式を直接解くことなく SA 法によりエネルギーを下げることで数値解を求めることができる。SOR 法と同じ解が得られるのだからわざわざこの手法を使う必要はないと思う読者がいるかもしれない。しかし、この手法は SOR 法よりも格段に強力な数値解析手法となりうる。なぜなら、複雑な方程式を解くために必要となる解の仮定、Ansatz を用いる必要がないからである。Ansatz を用いることなく解析することができるため、軸対称、球対称などではない変形した解を得ることが可能である。

この章では 1 次元のトポロジカルソリトン解である kink 解を求めるためにこの手法を用いた。次の章からは 2 次元の模型にこの手法を用いて模型の解析を行っていく。2 次元以上の模型に対して Ansatz を用いずに数値解析を行うためには SA 法は必須の技術であることが次章でわかるだろう。

4　超伝導現象と Abelian-Higgs 模型

4.1　超伝導現象について

前章までででトポロジカルソリトン模型に関する基礎知識及びソリトン解の数値計算法の基礎を習得することができた。この節では**超伝導現象**について解説する。

超伝導とはなんだろうか？超伝導状態は様々な物質においてある**臨界温度** T_c を下回り相転移をすることで出現する。その主な性質として以下のものが挙げられる。

1. **熱力学的秩序相** : 超伝導状態は温度 $T > T_c$ での正常状態とは異なる熱力学的な相である。正常状態から超伝導状態への相転移では以下の図のように比熱の変化に異常を伴う。

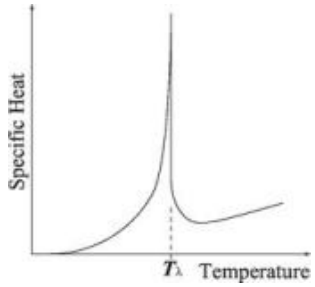


図 10: 超伝導転移における比熱の異常

上図の比熱の温度変化を見ると、T_c 付近で比熱が大きくなり、相転移後で減少する。低温にしていく過程でエントロピーが急激に放出される。すなわち、超伝導状態は正常状態に比べエントロピーが小さくなるので秩序を持つ相となる。この秩序相の形成により以下に記述する超伝導体に特徴的な現象が発現するようになる。

2. **完全導電性** : 超伝導体は超伝導状態になると下図[1]のように電気抵抗 ρ が 0 になる。

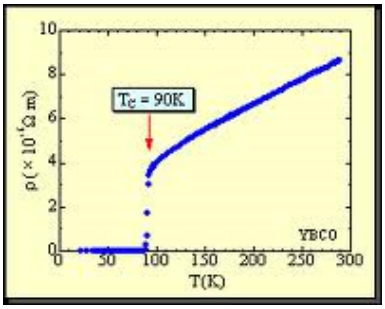


図 11: YBCO 超伝導体の電気抵抗の温度変化

[1] http://home.hiroshima-u.ac.jp/asugi/reserchj.html での掲載図

この現象は 1911 年に Onnes が水銀の抵抗の温度変化を測定しているときに発見された。この出来事が最初の超伝導体の発見である。この性質があるために超伝導体は現在に至るまで重要な研究対象となっている。

3. **完全反磁性**：超伝導体を平行な弱い磁場 H の中におくと、超伝導体表面に反磁性電流が流れ、超伝導体内部に磁場が侵入しなくなる。この性質が完全反磁性である。この現象を**マイスナー効果**と呼ぶ。また、温度 T に依存する値 $H_c(T)$ 以上に外部磁場 H を大きくすると、超伝導状態は壊れて常伝導状態になる。この値 H_c を**臨界磁場**という。

   また、超伝導体には**第一種超伝導体**、**第二種超伝導体**の 2 種類があり磁化 M と H の関係図が異なる。第一種超伝導体、第二種超伝導体それぞれの概念図および磁化過程を以下の図[2]に示す。

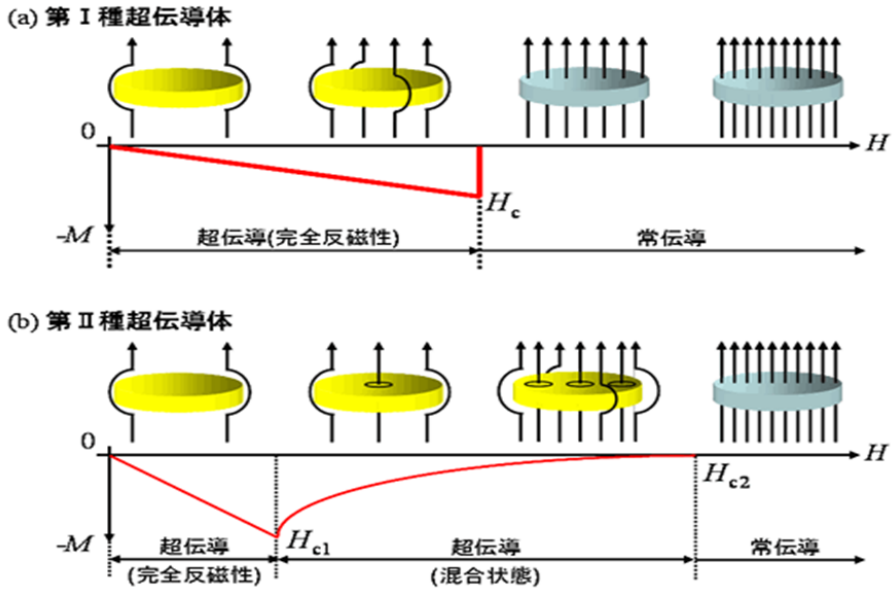


図 12: 第一種超伝導体と第二種超伝導体の磁化過程

図を見ればわかるように、第一種超伝導体と第二種超伝導体では外部磁場をかけたときの挙動が大きく異なる。第一種超伝導体の場合は外部磁場が弱い間は完全反磁性を示すが臨界磁場 H_c を超えると常伝導状態に移行する。これに対して第二種超伝導体の場合は外部磁場の大きさが**下部臨界磁場** H_{c1} になったところから磁束が超伝導内部に侵入し始める混合状態となる。そして、**上部臨界磁場** H_{c2} を超えると常伝導状態となる。

[2]http://www.phys.aoyama.ac.jp/ w3-jun/achievements/study_sc_chara.html にて掲載

4. **磁束の量子化**：超伝導になる金属の中空円筒を外部磁場中で T_c 以下に冷やすと、磁束は超伝導体内部に入らないが、このとき中空部分を貫く磁束の大きさは、**磁束量子** $\Phi_0 = 2\pi\hbar c/2q = 2.07 \times 10^{-15}$ weber$(=10^{-7}\text{Gauss}\cdot\text{cm}^2)$ の整数倍となることが知られている。ここで、$2\pi\hbar$ はプランク定数、c は光速度、q は素電荷である。このとき、$H = 0$ としても、超伝導状態が保たれている限り、磁束は捉えられたままであり、当然それに見合う永久電流が円筒の内側表面に流れている。(これが超伝導磁石) また、第二種超伝導体において $H_{c1} < H < H_{c2}$ の外部磁場 H を加えると、磁束は下図のように格子状に入っていき、個々の磁束のまわりに渦電流が流れる。その１つ１つの格子に相当する磁束の大きさはちょうど ϕ_0 に等しい。

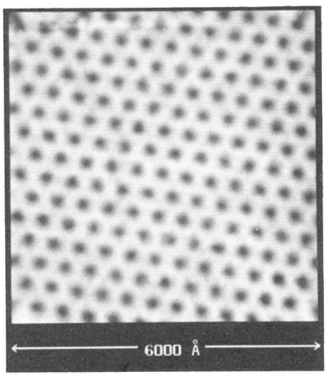


FIG. 2. Abrikosov flux lattice produced by a 1-T magnetic field in $NbSe_2$ at 1.8 K. The scan range is about 6000 Å. The gray scale corresponds to dI/dV ranging from approximately 1×10^{-8} mho (black) to 1.5×10^{-9} mho (white).

図 13: 走査型トンネル顕微鏡による磁束格子の像 (参考文献 [12])

4.2 Abelian-Higgs 模型とは

超伝導体の巨視的理論として Ginzburg-Landau 理論が知られている。Abelian-Higgs 模型はこの Ginzburg-Landau 理論を相対論的にした模型とされる。Abelian-Higgs 模型では温度変化によるポテンシャル構造の変化の概念が存在せず、常に二重底ポテンシャルになっている。本研究では第二種超伝導体の渦糸構造の解析のためこの模型を用いた解析を行った。また、注意点として、この節以降では解析の利便性のためプランク単位系を採用している。現象論と対応付けたい場合は単位系を適宜直して読み進めてもらいたい。(光速度 $c=1$, 万有引力定数 $G=1$, 換算プランク定数 $\bar{h}=1$, クーロン力定数 $\frac{1}{4\pi\epsilon}=1$, ボルツマン定数 $k=1$ となる。)

Abelian Higgs 模型のラグランジアン密度は、

$$\mathscr{L} = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu} + \frac{1}{2}|D_\mu\phi|^2 - \frac{\lambda}{4}\Big(|\phi|^2 - \frac{\mu^2}{\lambda}\Big)^2 \tag{40}$$

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu,\ \ D_\mu = \partial_\mu - iqA_\mu$$

で与えられる。ここで $F^{\mu\nu}$ はゲージ群 U(1) のゲージ場 A^μ に対する電磁場の強さであり、ϕ は複素スカラー場、D_μ は共変微分である。また、q は ϕ 粒子の持つ電荷である。このスカラー場 ϕ がこの模型の秩序パラメータとなっている。

Abelian Higgs 模型は、ゲージ群として $U(1)$ を持ち、複素スカラー場 ϕ とゲージ場 $A_\mu(x)$ の相互作用を記述する局所的にゲージ不変な理論である。ここで、$\mathscr{L}$ は以下の $U(1)$ ゲージ変換、

$$A_\mu(x) \to A'_\mu(x) + \frac{1}{q}\partial_\mu\chi(x),\ \phi(x) \to \phi'(x) = e^{i\chi(x)}\phi(x)$$

で不変である。また、真空の安定性より $\lambda > 0$ となる。場の方程式は、

$$D^\mu D_\nu\phi = -\lambda\phi\big(|\phi|^2 - \frac{\mu^2}{\lambda}\big) \tag{41}$$

$$\partial^\nu F_{\mu\nu} = j_\mu \equiv \frac{1}{2}iq\big(\phi^*\partial_\mu\phi - \phi\partial_\mu\phi^*\big) \tag{42}$$

と求められる。(41),(42) 式は $6(=2+4)$ 個の連立非線形偏微分方程式であり、一般解を厳密に求めることは極めて困難である。そこで、方程式の自由度を減らし、目的に合わせた対称性を持った解の仮定、Ansatz を課していくことになる。

まず、渦糸型の解における解のトポロジー的な性質を見てみる。z 軸方向を向いている磁場の渦に対して、z 軸から十分離れた、閉じた経路 C を境界とする面を貫く磁束 Φ は

$$\Phi = \oint_C d\sigma^{\mu\nu}F_{\mu\nu}(x) = \oint_{C=\partial S} dx^\mu A_\mu(x) \tag{43}$$

で与えられる。ここでは、ストークスの定理を用いた。運動方程式は (3) より、$\phi(x) = |\phi(x)|e^{i\chi(\phi(x))}$ とおくと、

$$A_\mu(x) = \frac{1}{q^2}\frac{j_\mu(x)}{|\phi|^2} + \frac{1}{q}\partial_\mu\chi(x) \tag{44}$$

が得られる。これを、$j_\mu(x)=0$ になる十分遠方で C に沿って線積分すると、

$$\Phi=\frac{1}{q}\oint_C dx^\mu\partial_\mu\chi=\frac{1}{q}[\chi(2\pi)-\chi(0)] \tag{45}$$

場 $\phi(x)$ が 1 価関数になるという要請を課すと、位相 $\chi(x)$ は 2π の整数倍だけ変化する。これにより、渦糸の磁束は量子化される。ここで、n は渦糸の巻き数 (winding number) を意味している。そしてこの巻き数 n こそが超伝導体における磁束格子の数に対応する。したがって磁束 Φ は前節の磁束量子 Φ_0 を用いて、

$$\Phi=\frac{2\pi}{q}n=\Phi_0 n \tag{46}$$

となる。(ここではプランク単位系を用いているので見た目の表式が前節と異なる) このように、Abelian-Higgs 模型の位相的性質から磁束量子の整数倍になる保存量を定義することができた。この模型のトポロジカルチャージはこの n となる。

次に、第二種超伝導体において存在する渦糸解に対応する解を求めるため、以下のような、静的 (時間に非依存) で軸対称な Ansatz を採用する。

$$\begin{aligned}&A_0(x)=0,\ \ \mathbf{A}(x)=\mathbf{e}_\phi A(r)\ \ (A_r(x)=A(r),\ A_r(x)=0=A_z(x))\\&\phi(x)=f(r)e^{in\phi}\ \ (n=0,\pm1,\pm2,\dots)\end{aligned} \tag{47}$$

ここで、円柱座標 (r,ϕ,z) を導入した。場の方程式に Ansatz を代入すると、以下のような一変数 r の関数 f と A に対する連立非線形微分方程式が求められる。

$$-\frac{1}{r}\frac{d}{dr}\left[r\frac{d}{dr}f(r)\right]+\left[\left(\frac{n}{r}-qA(r)\right)^2+\lambda\left(f^2(r)-\frac{\mu^2}{\lambda}\right)\right]f(r)=0 \tag{48}$$

$$-\frac{d}{dr}\left[\frac{1}{r}\frac{d}{dr}(rA(r))\right]+\left[q^2A(r)-\frac{nq}{r}\right]f^2(r)=0 \tag{49}$$

これらの方程式 (48),(49) 式が Ansatz を課した相対論的 Ginzburg-Landau 方程式である。方程式の形から、$f(r)$ と $A(r)$ の一番簡単な解は、

$$A(r)=\frac{n}{qr},\ \ f(r)=\frac{\mu}{\sqrt{\lambda}} \tag{50}$$

である。これは最低エネルギーの解であり、真空解と呼ばれる。しかし、我々が求める解は真空解ではなく渦糸解である。なんとかして真空解以外の解を得たい。渦糸解がトポロジカルソリトン解であると期待して、模型にトポロジカルソリトン解として持たねばならない性質を付加していく。ソリトン解は局在した有限エネルギーの解を持つのでこの模型においても有限エネルギーの解を得ることを目指す。

さて、この模型のエネルギー汎関数は、

$$E=\int d^3x\left[\frac{1}{2}(|\vec{B}|^2+|\vec{E}|^2)+\frac{1}{2}|D_k\phi|^2+\frac{1}{2}|D_0\phi|^2+\frac{\lambda}{4}\left(|\phi|^2-\frac{\mu^2}{\lambda}\right)^2\right] \tag{51}$$

である。Ansatz を代入すると、

$$E=\int_{-\infty}^{\infty}dz\int_{0}^{\infty}2\pi rdr\Big[\frac{1}{2r^2}\Big(\frac{d}{dr}\big(rA(r)\big)\Big)^2+\Big(\frac{d}{dr}f(r)\Big)^2+\Big(qA(r)-\frac{n}{r}\Big)^2f(r)^2+\frac{\lambda}{4}\Big(f(r)^2-\frac{\mu^2}{\lambda}\Big)\Big] \tag{52}$$

となる。エネルギーの表式から、被積分関数はすべて正の項からなるので、有限エネルギーを得るためには $r\to\infty$ で各項はそれぞれ 0 にならなければならない。したがって、$f(r), A(r)$ は遠方においては真空解 (50) 式に近づかなければならない。(48), (49) 式において、$r\to\infty$ として漸近解を求めると、

$$f(r)\sim\frac{\mu}{\sqrt{\lambda}}+\frac{C_1}{\sqrt{r}}\exp\Big(-\frac{r}{\xi}\Big), \tag{53}$$

$$A(r)\sim\frac{n}{qr}+\frac{C_2}{\sqrt{r}}\exp\Big(-\frac{r}{\delta}\Big). \tag{54}$$

ここで、C_1, C_2 は積分定数であり、$\xi=\frac{1}{\mu}, \delta=\frac{\sqrt{\lambda}}{q\mu}$ である。ξ が**コヒーレンス長**、δ が**磁場侵入長**である。この 2 つのパラメータが Abelian-Higgs 模型の解の挙動を支配する。コヒーレンス長と磁場侵入長の比は無次元で **Ginzburg-Landau パラメータ (GL パラメータ)** と呼ばれている。

$$\kappa\equiv\frac{\delta}{\xi}=\frac{\sqrt{\lambda}}{q}. \tag{55}$$

磁場 B_z は $\vec{B}=\nabla\times\vec{A}$ より、

$$B_z=\frac{1}{r}\frac{d}{dr}(rA(r))\sim-\frac{C'}{\delta\sqrt{r}}\exp\Big(-\frac{r}{\delta}\Big). \tag{56}$$

以上で $r\to\infty$ における解の振る舞いを知ることができた。関数形を見ると、$f(r)$ は渦糸の中心から距離 ξ 以上離れると真空解の振る舞いが支配的になる。$A(r)$ の場合は中心から距離 δ 離れると真空解の振る舞いが支配的になる。それに伴い、B_z も中心から距離 δ 離れると 0 に近づく。数値解析により、この解はひものような解を許すことが知られており、**Nielsen-Olesen-Vortex** と呼ばれている。[8] このような解が実際に得られるかどうかを解析的に確認することは不可能なので数値計算により確かめることになる。次節では実際に数値解を求めていく。

磁束の量子化はこの解を用いると、以下の式でも表すことができるようになる。

$$\Phi=\int_S dxdyB_z=\oint_C dx^\mu A_\mu=\lim_{r\to\infty}r\int_0^{2\pi}d\phi A(r)=\frac{2\pi}{q}n. \tag{57}$$

境界では真空解のみが寄与することから、たしかにこの解で磁束の量子化を表現できている。すなわち、Nielsen-Olesen-vortex 解は我々が求めていた渦糸解であることがわかる。

また、コヒーレンス長と磁場侵入長の大きさが渦糸間に与える相互作用を支配する。コヒーレンス長が大きいときは関数 $f(r)$ の寄与が強く、個々の渦糸間には引力が支配的に働く。磁場

侵入長が大きいときは関数 $A(r)$ の寄与が強く、個々の渦糸間には斥力が支配的に働くようになる。通常の超伝導体においては、κ の値によって

$$\kappa < \frac{1}{\sqrt{2}} \cdots \text{第一種超伝導体}, \qquad \kappa > \frac{1}{\sqrt{2}} \cdots \text{第二種超伝導体} \tag{58}$$

の 2 種類に分けられる。非相対論的な超伝導模型である Ginzburg-Landau 模型の場合には渦糸解は第二種超伝導体にしか存在しない。しかし、相対論的な超伝導模型である Abelian-Higgs 模型では κ の値に依らず渦糸解が存在する。Nielsen-Olesen-Vortex は $\kappa < \frac{1}{\sqrt{2}}$ では互いに引き合い、$\kappa = \frac{1}{\sqrt{2}}$ では相互作用しない。また、$\kappa > \frac{1}{\sqrt{2}}$ では n 単位の磁束量子を持つ 1 つの vortex は、1 単位の磁束量子を持つ vortex の n 倍より大きいエネルギーを持つため、単位 vortex に解離することが知られている。ここで疑問が生じる。$\kappa < \frac{1}{\sqrt{2}}$ の場合一般的に第一種超伝導体においては磁束量子が存在しない。しかし、この模型では複数の磁束量子が 1 箇所に集まった大きな磁束として存在する。この模型は現象と一致しないのであろうか?実は、近年この模型で得られる解に対応すると考えられる超伝導体が発見された。それが**メゾスコピック超伝導体**である。メゾスコピック超伝導体とはコヒーレンス長 ξ が試料面積と同程度の超伝導体のことで、ξ が大きければ磁束量子が解離して存在し、ξ が小さければ磁束量子が集まって存在することが知られている。解離した状態の渦糸解は multi vortex state, 集まった状態の渦糸解は gigantic vortex state と呼ばれている。このようなメゾスコピック超伝導現象を解析するためには、Abelian-Higgs 模

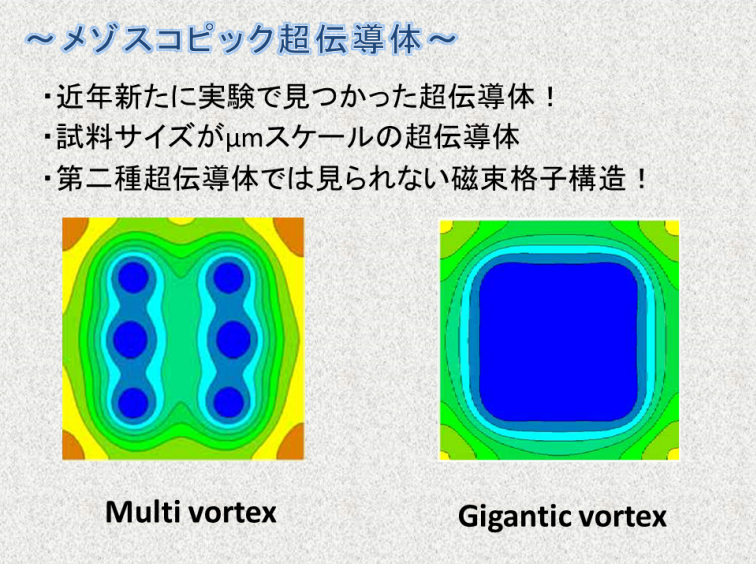


図 14: メゾスコピック超伝導体の概念図 (参考文献 [13])

型にサイズ効果を含めて計算しなければならない。また、代表的なメゾスコピック超伝導体である MgB_2 などは 2 バンドの超伝導体として知られているため、スカラー場をもう 1 つ追加した Abelian-Higgs 模型なども近年盛んに研究されている。

このように、Abelian-Higgs 模型は通常の第二種超伝導体はもちろん、メゾスコピック超伝導体における磁束格子の構造解析にも非常に有用なトポロジカルソリトン模型である。次節以降では数値計算で実際にこの模型を解析していく。

4.3 SOR 法で 1 次元 Abelian-Higgs 模型を解析

ここでは、前章で解説した Abelian-Higgs 模型の数値解を求める。まずは Ansatz を課して 1 次元の方程式となった (48),(49) 式の数値解を SOR 法により求めていく。これらの式を差分化すると、

$$\begin{aligned}
&\frac{f_{i+1}-2f_i-f_{i-1}}{\Delta r^2}+\frac{1}{r}\frac{f_{i+1}-f_{i-1}}{2\Delta r}-\Big(\frac{n}{r}-A_i\Big)^2+\Big(f_i^2-\frac{\mu^2}{\lambda}\Big)f_i=0,\\
&\frac{A_{i+1}-2A_i-A_{i-1}}{\Delta r^2}+\frac{1}{r}\frac{A_{i+1}-A_{i-1}}{2\Delta r}-\frac{A_i}{r^2}+\Big(q^2A_i-\frac{nq}{r}\Big)f_i^2=0
\end{aligned} \tag{59}$$

となる。座標空間を $n+1$ 分割された差分方程式 $(i=0,1\ldots,n-1,n)$ における境界条件は、

$$\begin{aligned}
f_0&=0\\
f_n&=\frac{\mu}{\sqrt{\lambda}}\\
A_0&=0\\
A_n&=\frac{n}{qr}
\end{aligned} \tag{60}$$

となる。

いくつかのパラメータについて、(59) 式を境界条件 (60) 式の下で SOR 法により数値計算を行った。得られた数値解 $f(r), A(r), B_z(r)$ それぞれを以下の図にプロットした。

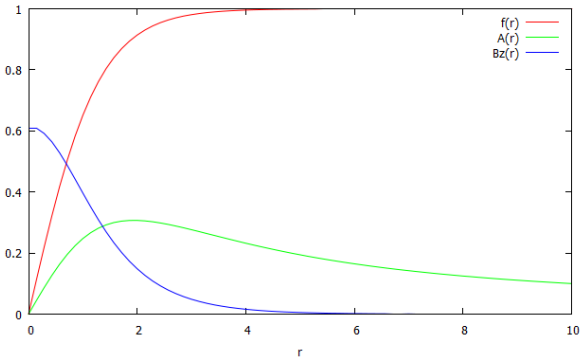


図 15: $\mu = q = \lambda = n = 1.0$

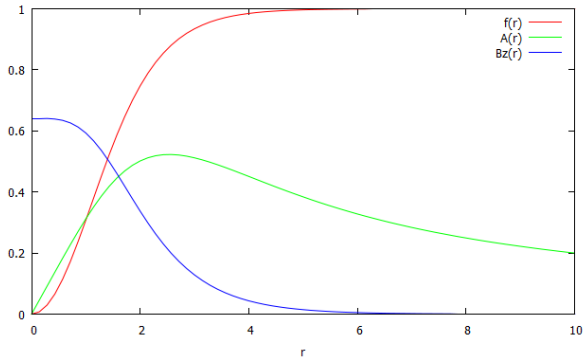


図 16: $\mu = q = \lambda = 1.0, n = 2.0$

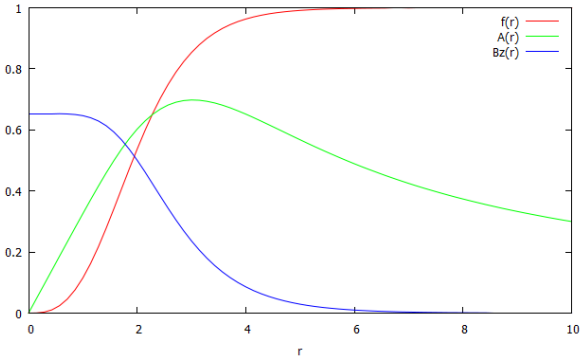


図 17: $\mu = q = \lambda = 1.0, n = 3.0$

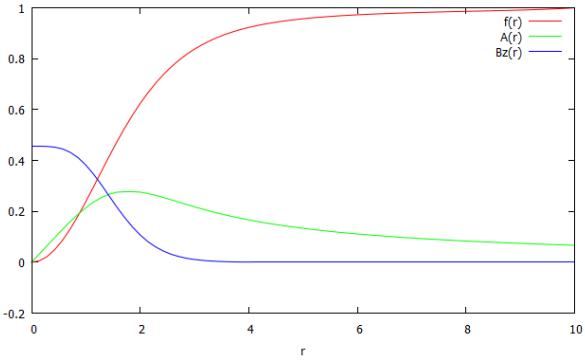


図 18: $\mu = \lambda = 1.0, q = 3.0, n = 2.0$

以上の図で着目すべき点がいくつかある。まず、SOR 法によって得られた解が遠方において解析的に導出された漸近解の挙動と一致している。次に、トポロジカルチャージ n を増やしていくと磁束格子の塊が大きくなり、磁束格子が集まって存在していることがわかる。最後に、これらの数値解では Ginzburg-Landau パラメータ κ の値によらず、磁束格子が原点に集まって存在している。これはどういうことだろうか?本来 Abelian-Higgs 理論によると κ の値によって磁束格子間に働く引力と斥力の強さが変化し、解の形状が multi vortex もしくは gigantic vortex のどちらかに分かれるはずである。しかし、数値解ではそのような挙動は見られない。我々の数値計算が間違っているのだろうか?そうではない。これは我々が課した Ansatz によるものである。Ansatz により軸対称解しか許されない状況となっているため、multi-center の数値解を得ることができなくなっているのである。これが Ansatz を課して運動方程式を解くことにより生じる問題点である。

したがって、multi vortex state の渦糸解を得るためには Ansatz を課さずに数値計算を行わなければならない。しかし、Ansatz を課さずに SOR 法により 2 次元の Abelian-Higgs 模型の数値解を求めることは非常に困難である。そのため、次節では SA 法を用いて運動方程式を解かずに直接エネルギーを最小化することで数値解を求めていくことになる。

この節の最後に、1 次元の Abelian-Higgs 模型の数値解を得るために用いたソースコードを以下に示す。

```
//SOR 法で 1 次元 Abelian-Higgs 模型の数値解を求める。

#include <cstdio>
#include <cmath>
#include <ctime>
#include <cstdlib>

#define PI 3.1415926535
#define MESH 70 //メッシュの分割数
#define MAX 10
#define MIN 0
#define START 500000
#define KAKUNIN 2000

using namespace std;

double q=3.0;//パラメータ q
double lam=1.0;//パラメータ λ
double mu=1.0;//パラメータ μ
double n=2.0;//チャージ数 n

//初期配位の定義
double func_F(double x);
double func_A(double x);

int main()
{
  int i,j,k;//ループ用処理変数
  double h,rh;//微小距離とその逆数
  double x;//運動方程式を解くときに使う
  double mmax=MAX;//計算領域右端
  double mmin=MIN;//計算領域左端
```

```
int count;//計算回数
FILE *data;

double F[MESH+1];//配位 f
double A[MESH+1];//配位 A
double Bz[MESH+1];//A から計算した Bz
double Ed[MESH+1];//エネルギー密度
double tmpF[3];//(1+従属変数の数) ×扱いたい場の数分用意する。
double tmpA[3];//
double cF;//計算用 f
double cdF;//計算用 f'
double cddF;//計算用 f''
double cA;//計算用 A
double cdA;//計算用 A'
double cddA;//計算用 A''
double Fzd[MESH+1];//f の残差密度
double Azd[MESH+1];//A の残差密度
double E;//エネルギー
double Fzan;//F の残差
double Azan;//A の残差

E=0.0;
Fzan=0.0;
Azan=0.0;
h=(mmax-mmin)/MESH;
rh=1.0/h;

//初期配位を定義
for(i=MESH;i>=0;i--){
  F[i]=func_F(mmin+i*h);
  A[i]=func_A(mmin+i*h);
  Fzd[i]=0.0;
  Azd[i]=0.0;
}
//計算前の準備としての境界条件の設定
F[0]=0.0;
A[0]=0;
F[MESH]=mu/sqrt(lam);
```

```
A[MESH]=n/q/mmax;
Bz[0]=Bz[1];//微分の関係上端の磁場の計算はできないので原点ではとなりと同じ.
Bz[MESH]=Bz[MESH-1];

//実際の SOR 計算処理
for(count=START; count>0; count--){
  //値確認用
  if(count%KAKUNIN==0){
    data=fopen("SOR_abelian.txt","w"); //計算結果保存用
    for(i=MESH;i>=0;i--){
      fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n",
        mmin+i*h, F[i], Fzd[i], A[i], Azd[i], Bz[i]);
    }
    fclose(data);
    printf("%d/%d\n%lf\t%lf\t%lf\n", count, START, E, Fzan, Azan);
  }
  //残差を用いて配位の再設定
  for(i=MESH;i>=0;i--){
    F[i]=F[i]+0.0001*Fzd[i];
    A[i]=A[i]+0.0001*Azd[i];
  }
  //境界条件の設定
  F[0]=0;
  A[0]=0.0;
  F[MESH]=mu/sqrt(lam);
  A[MESH]=n/q/mmax;
  Bz[0]=Bz[1];//微分の関係上端の磁場の計算はできないので原点ではとなりと同じ.
  Bz[MESH]=Bz[MESH-1];

  Fzan=0.0;
  Azan=0.0;
  E=0.0;

  //このループの中には両端の計算は入っていない
  for(i=MESH-1;i>=1;i--){
    tmpF[0]=F[i];//現在位置の f
    tmpF[1]=F[i-1];//右の f
    tmpF[2]=F[i+1];//左の f
```

```
      tmpA[0]=A[i];//現在位置の A
      tmpA[1]=A[i-1];//右の A
      tmpA[2]=A[i+1];//左の A
      cF=tmpF[0];
      cdF=0.5*(tmpF[2]-tmpF[1])*rh;
      cddF=(tmpF[1]-2*tmpF[0]+tmpF[2])*rh*rh;
      cA=tmpA[0];
      cdA=0.5*(tmpA[2]-tmpA[1])*rh;
      cddA=(tmpA[1]-2*tmpA[0]+tmpA[2])*rh*rh;

      x=mmin+i*h;
      Fzd[i]=cddF+cdF/x-((n/x-cA)*(n/x-cA)+(cF*cF-mu*mu/lam))*cF;
      Azd[i]=cddA+cdA/x-cA/x/x-(q*q*cA-n*q/x)*cF*cF;
      Bz[i]=cdA+cA/x;
      Ed[i]=( (cA+x*cdA)*(cA+x*cdA)*0.5/x/x+(cdF*cdF)+(q*cA-n/x)*(q*cA-n/x)*cF*cF
               +0.25*lam*(cF*cF-mu*mu/lam)*(cF*cF-mu*mu/lam) )*2*PI*x*h;
      Fzan+=Fzd[i];
      Azan+=Azd[i];
      E+=Ed[i];
    }
  }
  return 0;
}

//初期配位の定義
double func_F(double x)
{
  return 1.0;
}

double func_A(double x)
{
  return 1.0;
}
```

4.4 Simulated Annealing 法で 2 次元 Abelian-Higgs 模型を解析

この節では 2 次元直交座標系の Abelian-Higgs 模型の数値解を SA 法により得ることを目的とする。SA 法によりエネルギーを下げて数値解を得るのでまずは 2 次元直交座標のエネルギー

汎関数の式が必要となる。(51) 式より、

$$E = \iint_D dxdy \Big[\frac{1}{2}\Big(\frac{\partial A_y}{\partial x} - \frac{\partial A_x}{\partial y}\Big)^2 + \frac{1}{2}\Big\{\Big(\frac{\partial \phi_R}{\partial x}\Big)^2 + \Big(\frac{\partial \phi_R}{\partial y}\Big)^2 + \Big(\frac{\partial \phi_I}{\partial x}\Big)^2 + \Big(\frac{\partial \phi_I}{\partial y}\Big)^2\Big\} + qA_x\Big(\phi_I \frac{\partial \phi_R}{\partial x} - \phi_R \frac{\partial \phi_I}{\partial x}\Big) + qA_y\Big(\phi_I \frac{\partial \phi_R}{\partial y} - \phi_R \frac{\partial \phi_I}{\partial y}\Big) + \frac{q^2}{2}(\phi_R^2 + \phi_I^2)(A_x^2 + A_y^2) + \frac{\lambda}{4}\Big(\phi_R^2 + \phi_I^2 - \frac{\mu^2}{\lambda}\Big)\Big] \tag{61}$$

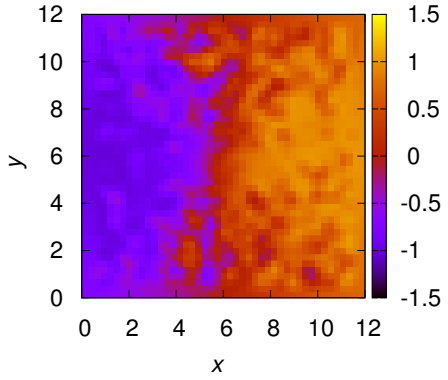
ここで、ϕ_R, ϕ_I は複素スカラー場 ϕ の実部と虚部であり、A_x, A_y はゲージ場 A_μ の x, y 成分である。D は超伝導体の領域を表す。

数値計算に入る前に、GL パラメータ κ がなぜ解の挙動を支配するのかを数理的に確かめる。トポロジカルソリトン模型はスケール変換に対して不変な理論である。そのため、模型に対して無次元化を行えばあらゆるスケールの現象に用いることができる模型となる。この無次元化を行った際に全てのパラメータの情報が集約される物理量が GL パラメータなのである。$\tilde{x} = qvx, \tilde{y} = qvy, \tilde{\phi_R} = \frac{\phi_R}{qv}, \tilde{\phi_I} = \frac{\phi_I}{qv}, \tilde{A_x} = \frac{A_x}{qv}, \tilde{A_y} = \frac{A_y}{qv}$ と置くことで無次元化を行うことができる。ここで、$v = \frac{\mu}{\sqrt{\lambda}}$ であり、チルダがついた物理量はすべて無次元である。無次元化を終えたあとのエネルギーは、

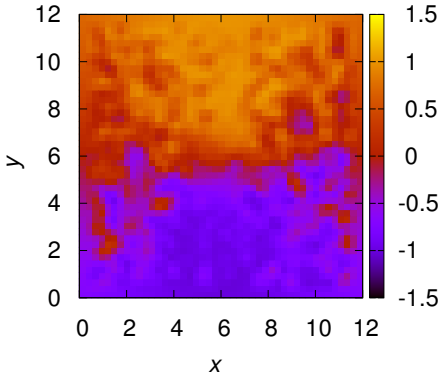
$$E = q^2v^2 \iint_D d\tilde{x}d\tilde{y} \Big[\frac{1}{2}\Big(\frac{\partial \tilde{A}_y}{\partial \tilde{x}} - \frac{\partial \tilde{A}_x}{\partial \tilde{y}}\Big)^2 + \frac{1}{2}\Big\{\Big(\frac{\partial \tilde{\phi}_R}{\partial \tilde{x}}\Big)^2 + \Big(\frac{\partial \tilde{\phi}_R}{\partial \tilde{y}}\Big)^2 + \Big(\frac{\partial \tilde{\phi}_I}{\partial \tilde{x}}\Big)^2 + \Big(\frac{\partial \tilde{\phi}_I}{\partial \tilde{y}}\Big)^2\Big\} + \tilde{A}_x\Big(\tilde{\phi}_I \frac{\partial \tilde{\phi}_R}{\partial \tilde{x}} - \tilde{\phi}_R \frac{\partial \tilde{\phi}_I}{\partial \tilde{x}}\Big) + \tilde{A}_y\Big(\tilde{\phi}_I \frac{\partial \tilde{\phi}_R}{\partial \tilde{y}} - \tilde{\phi}_R \frac{\partial \tilde{\phi}_I}{\partial \tilde{y}}\Big) + \frac{1}{2}(\tilde{\phi}_R^2 + \tilde{\phi}_I^2)(\tilde{A}_x^2 + \tilde{A}_y^2) + \frac{\kappa^2}{4}\Big(\tilde{\phi}_R^2 + \tilde{\phi}_I^2 - 1\Big)\Big] \tag{62}$$

となる。模型の作用は無次元でなければならず、Abelian-Higgs 模型は (3+1) 次元の模型なので、エネルギーの次元は $[\mathrm{L}^{-2}]$ でなければならない。無次元化の処方により、たしかにエネルギーの次元は $[\mathrm{L}^{-2}]$ となっていることがわかる。(62) 式の形を見ればわかるように、この模型では GL パラメータ κ が支配的に働いていることがわかる。無次元化を行い、後に次元戻しを行うことで様々なスケールの現象を扱うことができるようになる。また、無次元化を行って解析を行うことでパラメータ細部の情報にとらわれることなく模型の解析を行うことができる。しかし、本研究ではサイズ効果や境界条件が重要な超伝導現象を取り扱うので数値計算に用いた式には無次元化は行っていない。

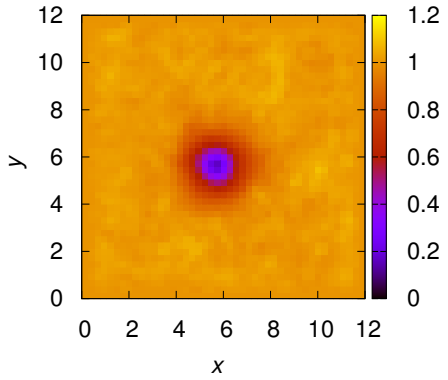
(61) 式に対してディリクレ&ノイマン境界条件を課し、メッシュ数 46×46 で SA 法を行った。課す境界条件によって得られる数値解は変化する。ここでは、GL パラメータ κ の違いにより multi vortex state、gigantic vortex state の 2 種類の解が存在することを示すためにこの境界条件を採用した。チャージ $N = 1, 2, 3, 4$,$\kappa = 1.0, 0.3$ について $\phi_R, \phi_I, \sqrt{\phi_R^2 + \phi_I^2}, B_z$, エネルギー密度、そしてエネルギーの推移をそれぞれ以下に図示した。



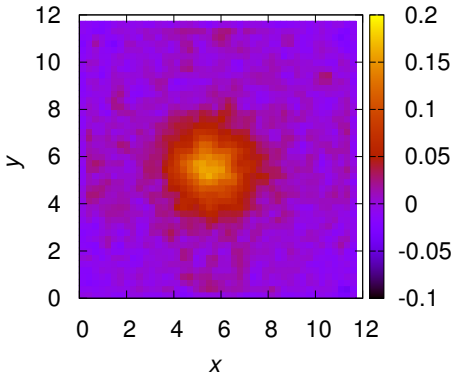
ϕ_R



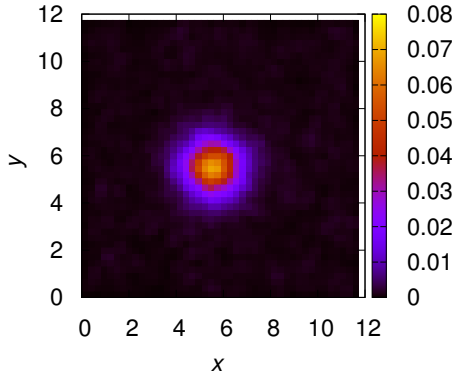
ϕ_I



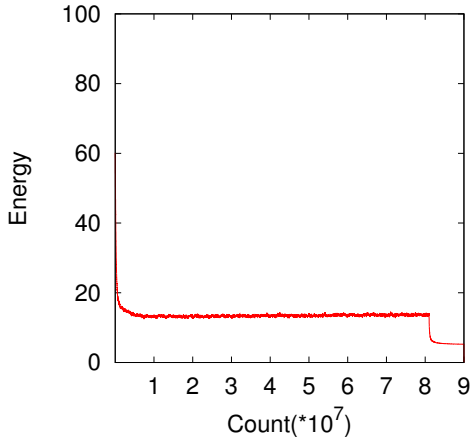
$\sqrt{\phi_R^2 + \phi_I^2}$



B_z

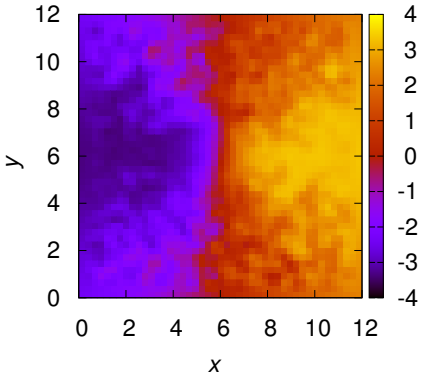


エネルギー密度

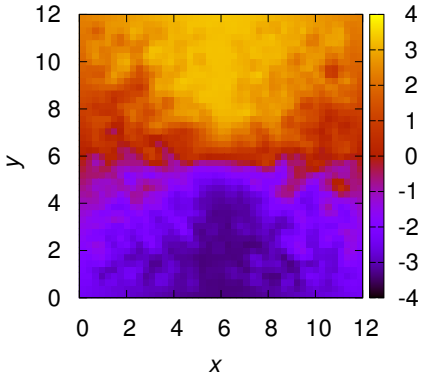


エネルギーの推移

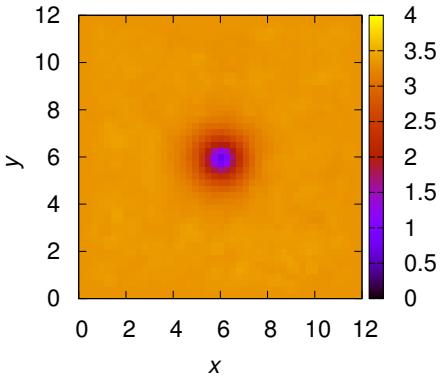
図 19: チャージ $N = 1$, GL パラメータ $\kappa = 1.0$



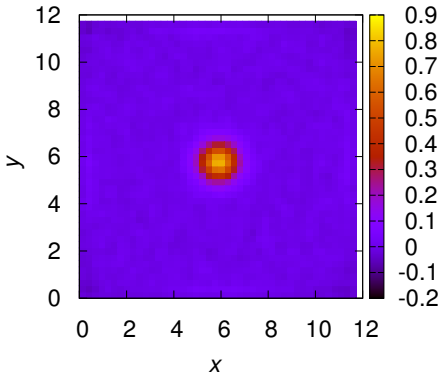
ϕ_R



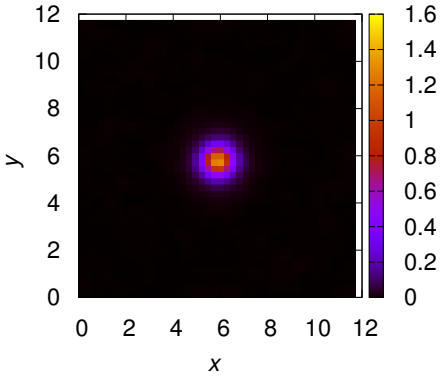
ϕ_I



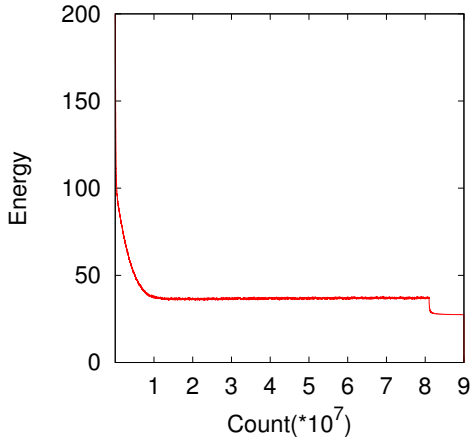
$\sqrt{\phi_R^2 + \phi_I^2}$



B_z

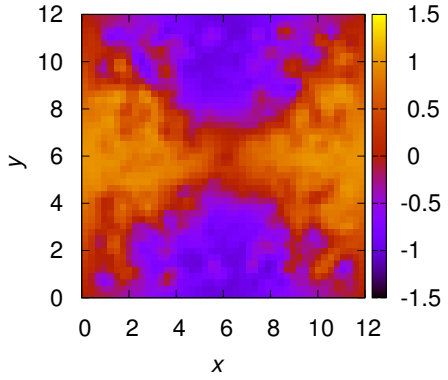


エネルギー密度

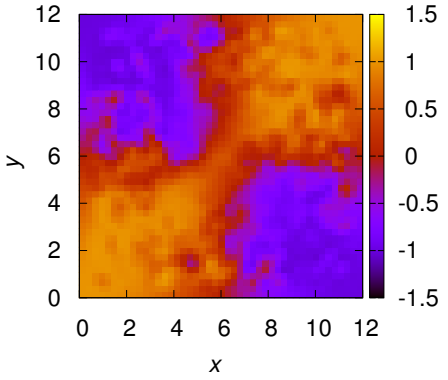


エネルギーの推移

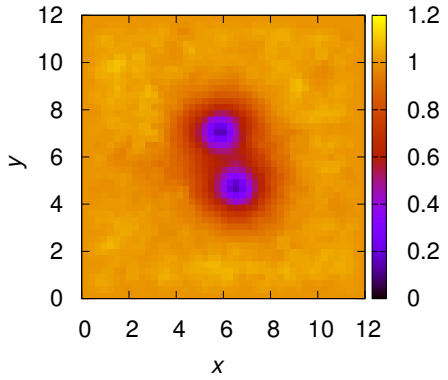
図 20: チャージ $N = 1$, GL パラメータ $\kappa = 0.3$



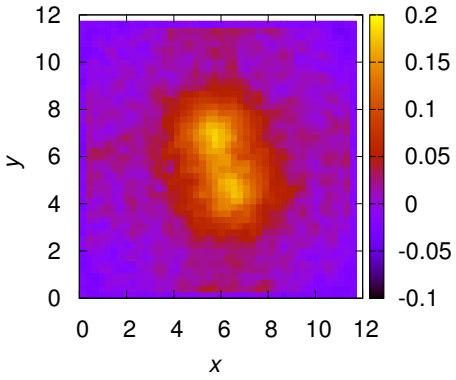
ϕ_R



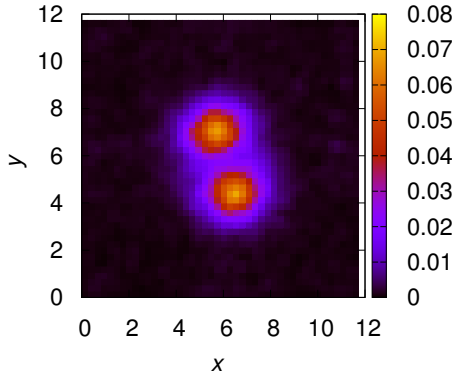
ϕ_I



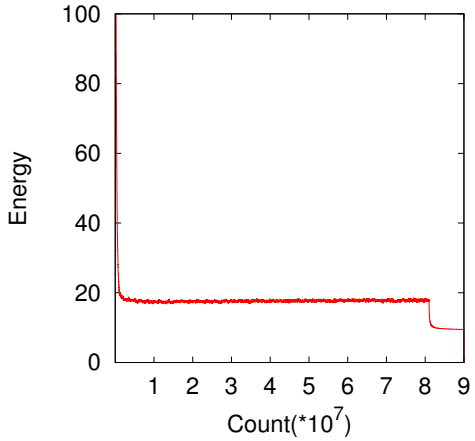
$\sqrt{\phi_R^2 + \phi_I^2}$



B_z

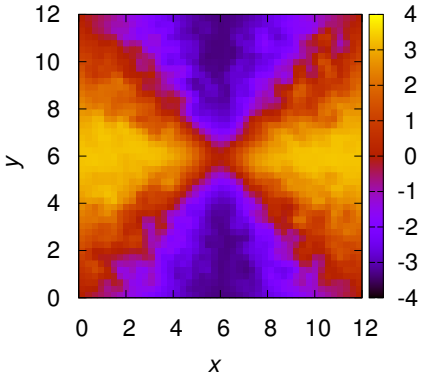


エネルギー密度

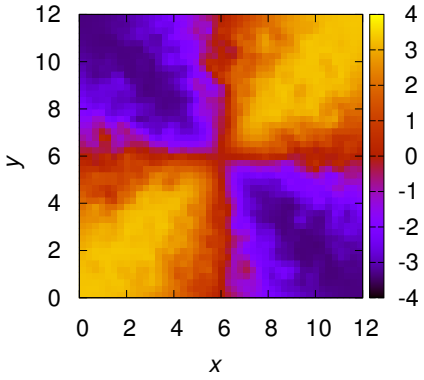


エネルギーの推移

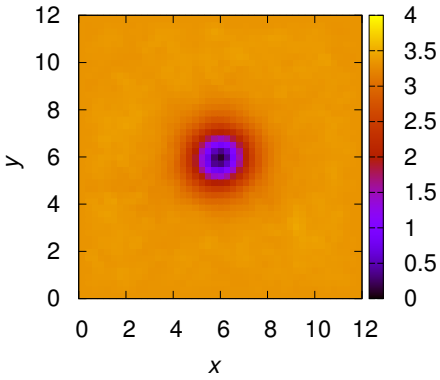
図 21: チャージ $N = 2$, GL パラメータ $\kappa = 1.0$



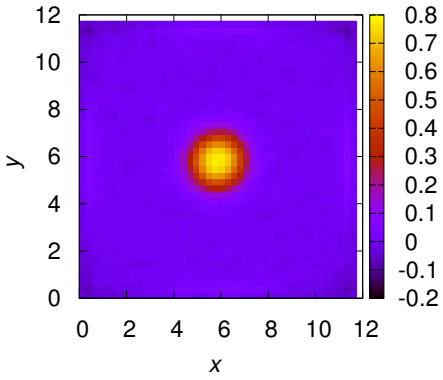
ϕ_R



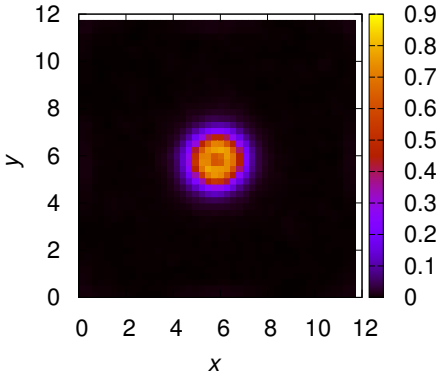
ϕ_I



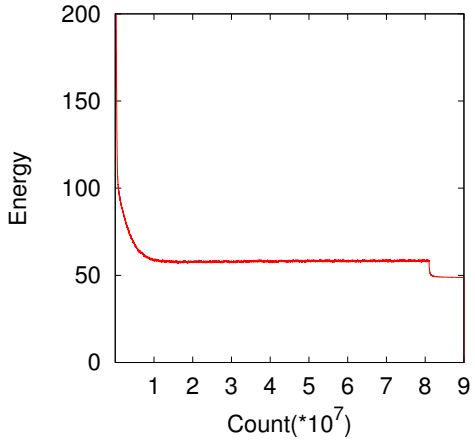
$\sqrt{\phi_R^2 + \phi_I^2}$



B_z

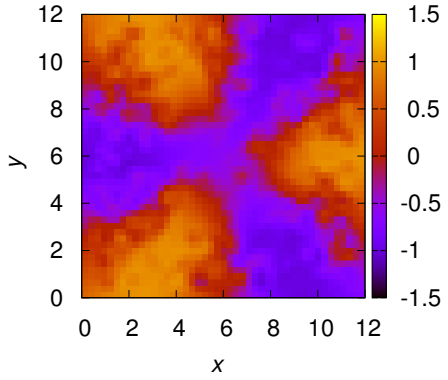


エネルギー密度

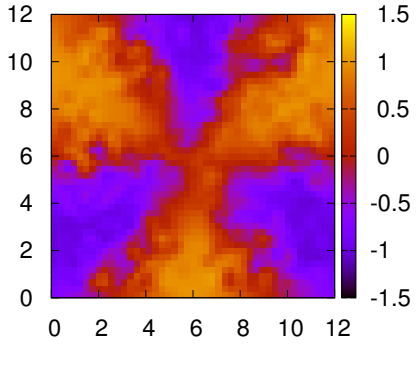


エネルギーの推移

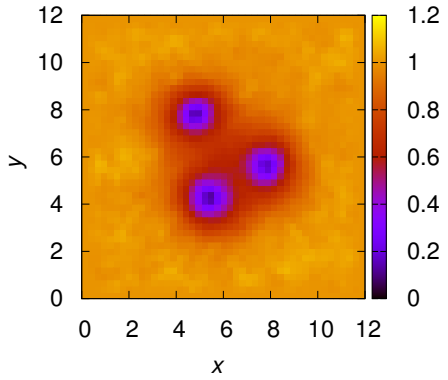
図 22: チャージ $N = 2$, GL パラメータ $\kappa = 0.3$



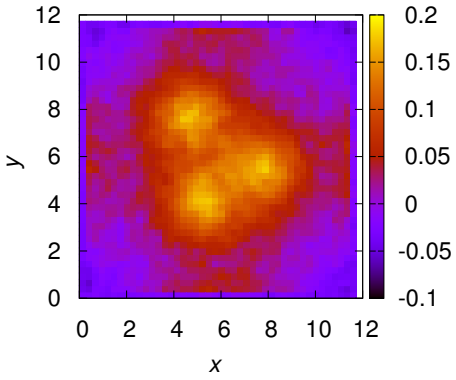
ϕ_R



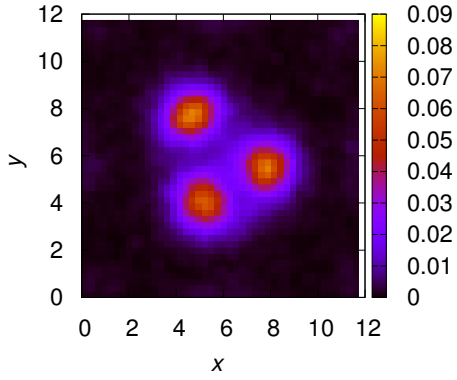
ϕ_I



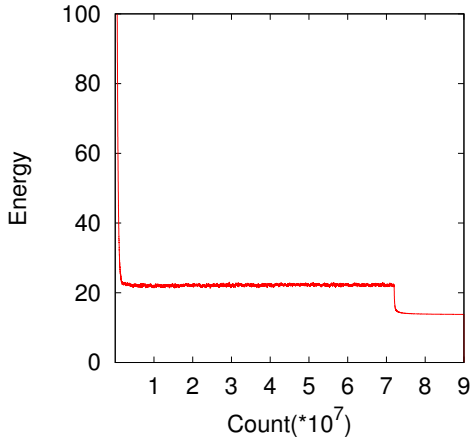
$\sqrt{\phi_R^2 + \phi_I^2}$



B_z

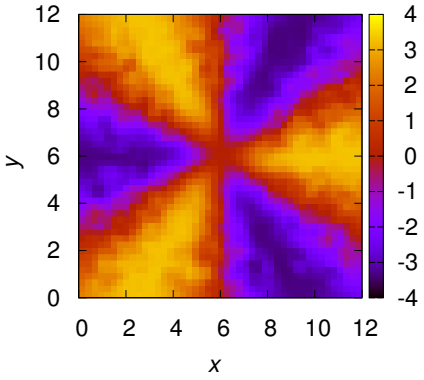


エネルギー密度

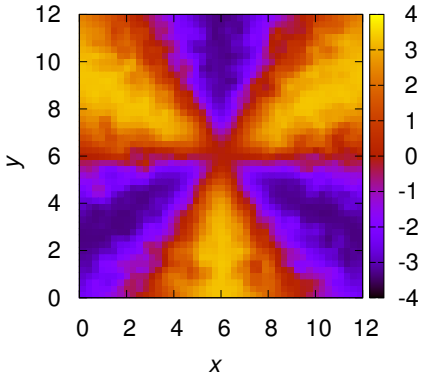


エネルギーの推移

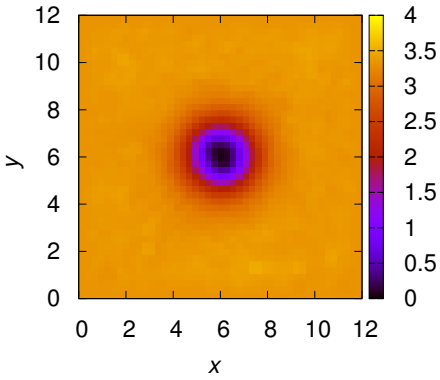
図 23: チャージ $N = 3$, GL パラメータ $\kappa = 1.0$



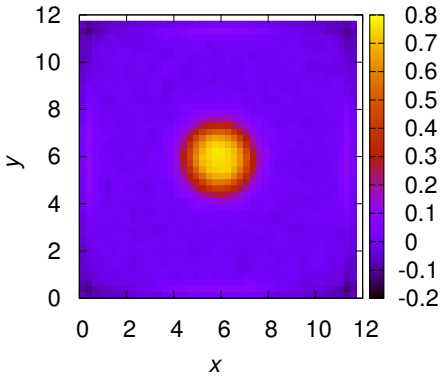
ϕ_R



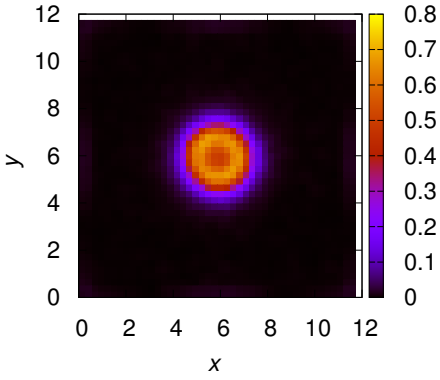
ϕ_I



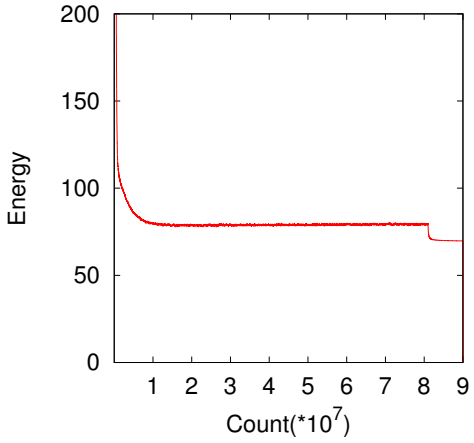
$\sqrt{\phi_R^2 + \phi_I^2}$



B_z

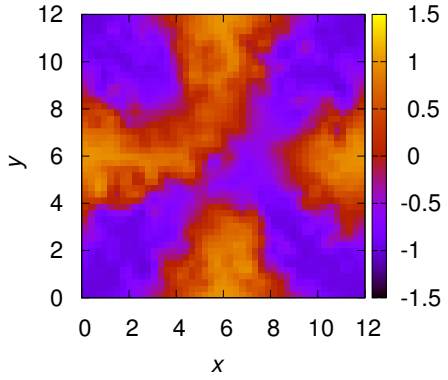


エネルギー密度

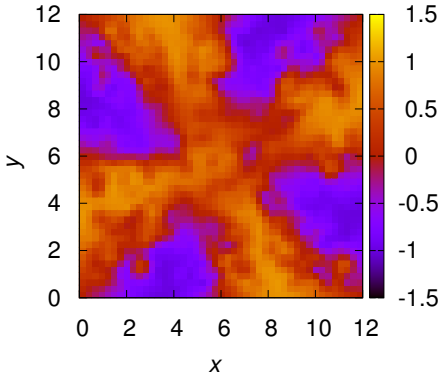


エネルギーの推移

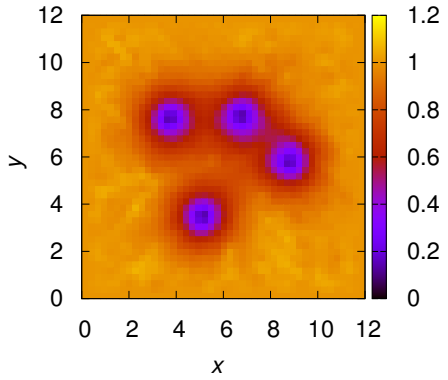
図 24: チャージ $N = 3$, GL パラメータ $\kappa = 0.3$



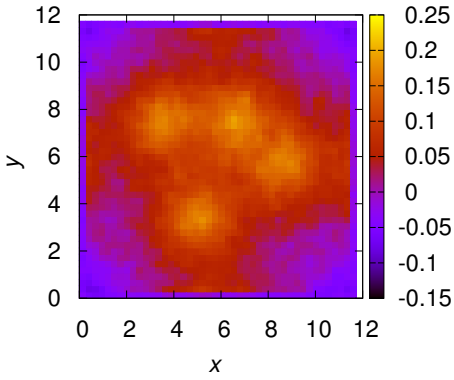
ϕ_R



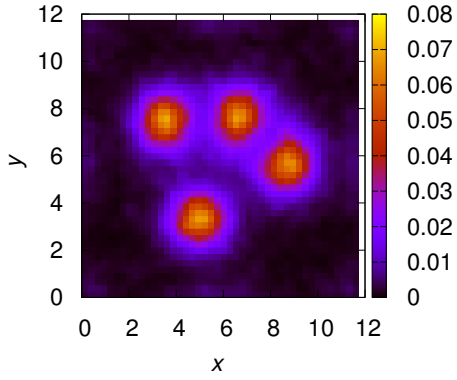
ϕ_I



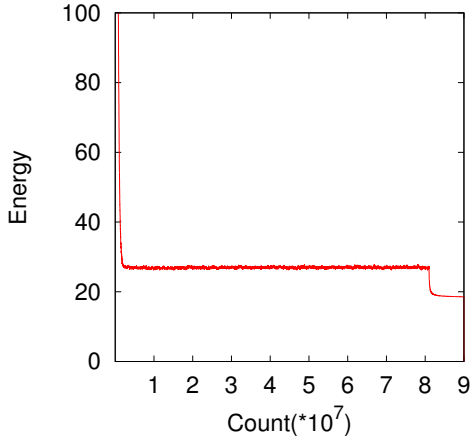
$\sqrt{\phi_R^2 + \phi_I^2}$



B_z

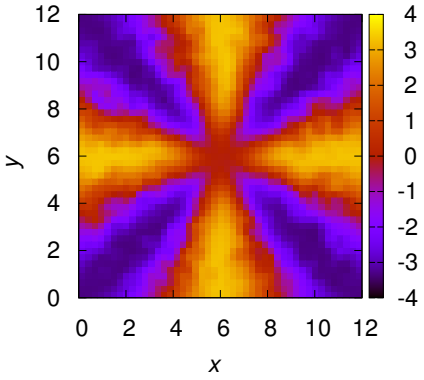


エネルギー密度

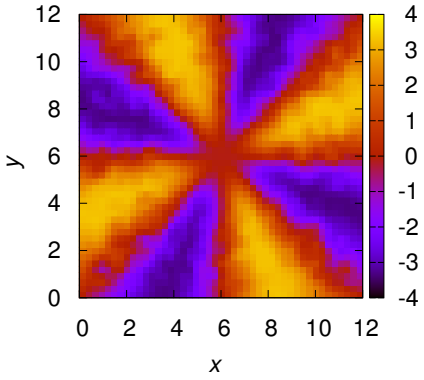


エネルギーの推移

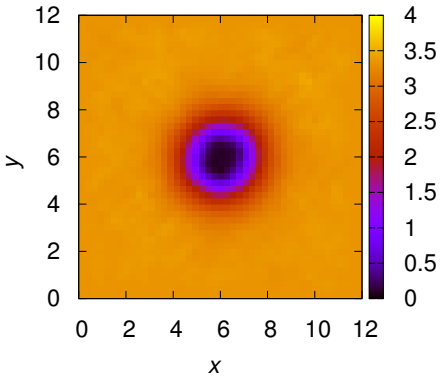
図 25: チャージ $N = 4$, GL パラメータ $\kappa = 1.0$



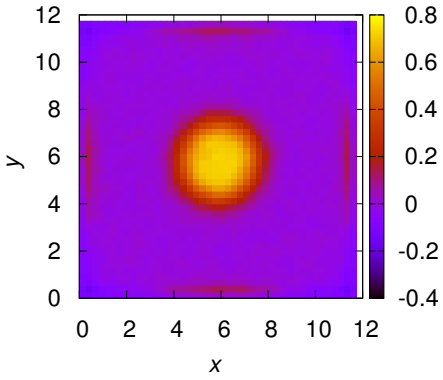
ϕ_R



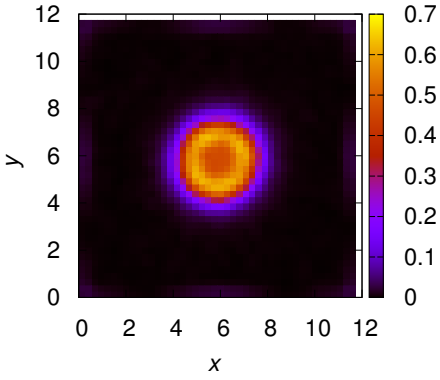
ϕ_I



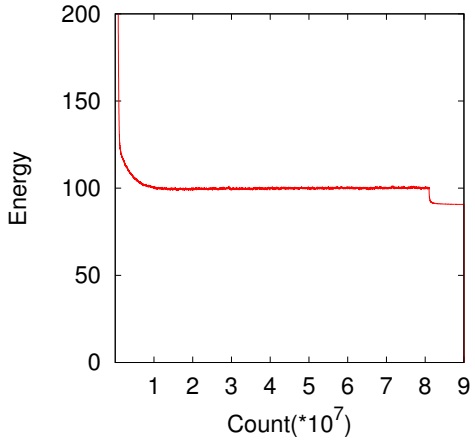
$\sqrt{\phi_R^2 + \phi_I^2}$



B_z



エネルギー密度



エネルギーの推移

図 26: チャージ $N = 4$, GL パラメータ $\kappa = 0.3$

チャージが 2 以上のときは κ の値によって数値解の挙動が変化している。Ansatz を課して 1 次元の模型で数値計算をしたときは κ の値の変化が解に反映されていなかった。しかし、2 次元で Ansatz を課さずに数値計算を行えば κ の値の変化が解に反映される。このように、Simulated Annealing 法を用いることで Ansatz なしで数値解を得ることができる。$\kappa > \frac{1}{\sqrt{2}}$ のときは Multi vortex state、$\kappa < \frac{1}{\sqrt{2}}$ のときは Gigantic vortex state となることを確かめることができた。

2 次元の Abelian-Higgs 模型を用いることで超伝導体の磁束格子構造の解析を行うことができることを示した。この模型に対して異なる境界条件を用いたり、複素スカラー場の数を増やしたり、ポテンシャルとして新たな項を導入するなど様々な研究が行われている。[25, 26, 27] 次章では強磁性体や超伝導体の解析に用いられる Baby-Skyrme 模型の解説を行う。

この節の最後に、2 次元 Abelian-Higgs 模型の数値計算のソースコードを掲載する。

```
//Abelian-Higgs 模型の Ansatz なしの解を Annealing で探す！

#include <cstdio>
#include <cmath>
#include <ctime>
#include <cstdlib>
#include "mersenne.h"

#define PI 3.1415926535
//メッシュは奇数 !!
#define MESH 45 //メッシュの分割数 (xy 平面で共通)
#define MIN 0.0 //x,y の最少値
#define MAX 12.0 //x,y の最大値
#define START 90000000 //合計計算回数
#define TSTART 20 //TSTART 毎に 1 回コストが上がる方向への遷移を許す
#define KAKUNIN 2000 //kakunin 回毎にエネルギーとチャージの値をファイルに出力

using namespace std;

int M1=MESH+1;
int M_2=MESH*MESH;
int M1_2=(MESH+1)*(MESH+1);
double Ep2,E0,Em2;
double mu=1.0;//パラメータ μ
double q=1.0;//パラメータ q
double lam=0.09;//パラメータ λ

double on,nn;//古い&新しいトポロジカルチャージ
```

```
double mmin=MIN;
double mmax=MAX;
double h;//刻み幅の指定 (x,y 方向)
double rh;//h の逆数

double Ed[MESH][MESH];//エネルギー密度
double cd[MESH][MESH];//チャージ密度
//エネルギーやチャージの推移を保存
double memo_En[START/KAKUNIN+1];
double memo_n[START/KAKUNIN+1];
double memo_E[START/KAKUNIN+1];
double memo_deri[START/KAKUNIN+1];

  double tF[MESH][MESH],tiF[MESH][MESH],tAx[MESH][MESH],tAy[MESH][MESH];//Φ、A
の平均値

struct{
  double F[(MESH+1)][(MESH+1)];//Φの実部成分
  double iF[(MESH+1)][(MESH+1)];//Φの虚部成分
  double Ax[(MESH+1)][(MESH+1)];//A の x 成分
  double Ay[(MESH+1)][(MESH+1)];//A の y 成分
  double dx_F[MESH][MESH];//Φの実部の x 微分
  double dy_F[MESH][MESH];//Φの実部の y 微分
  double dx_iF[MESH][MESH];//Φの虚部の x 微分
  double dy_iF[MESH][MESH];//Φの虚部の y 微分
  double dx_Ax[MESH][MESH];//Ax の x 微分
  double dy_Ax[MESH][MESH];//Ax の y 微分
  double dx_Ay[MESH][MESH];//Ay の x 成分
  double dy_Ay[MESH][MESH];//Ay の y 微分
} od,nd;

//初期配位の定義
double func_F(double x, double y);
double func_iF(double x, double y);
double func_Ax(double x, double y);
double func_Ay(double x, double y);

//重積分用の台形公式
```

```
double trapezoidal2(void);

//atan を計算する関数
double myatan(double y, double x);

//トポロジカルチャージ計算用の磁場積分求める
double trapezoidal_Bz(void);

//乱数処理部分呼び出し
 void ransu(void);

int main(void)
{
  double oE,nE;//古いエネルギー、新しいエネルギー
  double oEm2,nEm2;//トポロジカルチャージ用のエネルギー部分
  double oE0,nE0;
  double oEp2,nEp2;
  int count=START;//計算回数
  int tcount=TSTART;//この回数だけ配位の取り換え起きなかったら条件緩める。
  int dcount=0;//何回配位を取り換えたか

  int i,j;//ループ用
  //ファイル処理用
  FILE *data;

  h=(mmax-mmin)/MESH;
  rh=1.0/h;

//配位を初期関数で定める
  for(i=MESH;i>=0;i--){
    for(j=MESH;j>=0;j--){
      od.F[i][j]=func_F(-(MESH)*0.5+1.0*i, -(MESH)*0.5+1.0*j);
      od.iF[i][j]=func_iF(-(MESH)*0.5+1.0*i, -(MESH)*0.5+1.0*j);
      od.Ax[i][j]=func_Ax(-(MESH)*0.5+1.0*i, -(MESH)*0.5+1.0*j);
      od.Ay[i][j]=func_Ay(-(MESH)*0.5+1.0*i, -(MESH)*0.5+1.0*j);
    }
  }
//配位の微分を定義
```

```
  for(i=MESH-1;i>=0;i--){
    for(j=MESH-1;j>=0;j--){
      od.dx_F[i][j]=(od.F[i+1][j+1]+od.F[i+1][j]-od.F[i][j+1]-od.F[i][j])*0.5;
      od.dy_F[i][j]=(od.F[i+1][j+1]+od.F[i][j+1]-od.F[i+1][j]-od.F[i][j])*0.5;
      od.dx_iF[i][j]=(od.iF[i+1][j+1]+od.iF[i+1][j]-od.iF[i][j+1]-od.iF[i][j])*0.5;
      od.dy_iF[i][j]=(od.iF[i+1][j+1]+od.iF[i][j+1]-od.iF[i+1][j]-od.iF[i][j])*0.5;
      od.dx_Ax[i][j]=(od.Ax[i+1][j+1]+od.Ax[i+1][j]-od.Ax[i][j+1]-od.Ax[i][j])*0.5;
      od.dy_Ax[i][j]=(od.Ax[i+1][j+1]+od.Ax[i][j+1]-od.Ax[i+1][j]-od.Ax[i][j])*0.5;
      od.dx_Ay[i][j]=(od.Ay[i+1][j+1]+od.Ay[i+1][j]-od.Ay[i][j+1]-od.Ay[i][j])*0.5;
      od.dy_Ay[i][j]=(od.Ay[i+1][j+1]+od.Ay[i][j+1]-od.Ay[i+1][j]-od.Ay[i][j])*0.5;
    }
  }

  nd=od;

//初期配位でのエネルギー計算
  nE=trapezoidal2();
  nEm2=Em2;
  nE0=E0;
  nEp2=Ep2;
  oE=nE;
  oEm2=nEm2;
  oE0=nE0;
  oEp2=nEp2;

  nn=trapezoidal_Bz()/(2*PI);
  on=nn;

  data=fopen("SA_Abelian.txt","w"); //計算結果保存用
  for(i=MESH;i>=0;i--){
    for(j=MESH;j>=0;j--){
      fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h,
                od.F[i][j], od.iF[i][j], od.Ax[i][j], od.Ay[i][j],
                 sqrt(od.F[i][j]*od.F[i][j]+od.iF[i][j]*od.iF[i][j]));
    }
    fprintf(data,"\n");
  }
    fclose(data);
```

```
  data=fopen("SA_Abelian_Bz.txt","w"); //計算結果保存用
  for(i=MESH-1;i>=0;i--){
    for(j=MESH-1;j>=0;j--){
      fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h,
                rh*(od.dx_Ay[i][j]-od.dy_Ax[i][j]), Ed[i][j], cd[i][j]);
    }
    fprintf(data,"\n");
  }
  fclose(data);

//ここからスカラー場とゲージ場を乱数で変化させていく
  init_genrand((unsigned)time(NULL));
  do{
    //確認用
    if(count%KAKUNIN==0){
      memo_E[(START-count)/KAKUNIN]=oE;
      memo_n[(START-count)/KAKUNIN]=on;
      memo_En[(START-count)/KAKUNIN]=oE/on;
      memo_deri[(START-count)/KAKUNIN]=Ep2/Em2;
      printf("%lf %lf %lf %lf %lf %d %lf \n",oE,Ep2,E0,Em2,on,dcount,Ep2/Em2);
      dcount=0;
    }
    if(count%1000000==0) printf("\n 残り%d\n",count);

    //配位、ゲージ場を乱数でゆらしてやる
    if(count>(START*0.4)){
      double tmp=M1_2*0.8;
      for(i=tmp;i>=0;i--){
        ransu();
      }
    }
    else if((count>(START*0.1))&&((count<(START*0.4)))) {
       double tmp=M1_2*0.8;
      for(i=tmp;i>=0;i--){
        ransu();
      }
    }
    else {
```

```
      double tmp=M1_2*0.5;
      for(i=tmp;i>=0;i--){
        ransu();
      }
    }

//配位、ゲージ場の中点偏微分を計算する
    for(int i=0;i<=MESH-1;i++){
      for(int j=0;j<=MESH-1;j++){
        nd.dx_F[i][j]=(nd.F[i+1][j+1]+nd.F[i+1][j]-nd.F[i][j+1]-nd.F[i][j])*0.5;
        nd.dy_F[i][j]=(nd.F[i+1][j+1]+nd.F[i][j+1]-nd.F[i+1][j]-nd.F[i][j])*0.5;
        nd.dx_iF[i][j]=(nd.iF[i+1][j+1]+nd.iF[i+1][j]-nd.iF[i][j+1]-nd.iF[i][j])*0.5;
        nd.dy_iF[i][j]=(nd.iF[i+1][j+1]+nd.iF[i][j+1]-nd.iF[i+1][j]-nd.iF[i][j])*0.5;
        nd.dx_Ax[i][j]=(nd.Ax[i+1][j+1]+nd.Ax[i+1][j]-nd.Ax[i][j+1]-nd.Ax[i][j])*0.5;
        nd.dy_Ax[i][j]=(nd.Ax[i+1][j+1]+nd.Ax[i][j+1]-nd.Ax[i+1][j]-nd.Ax[i][j])*0.5;
        nd.dx_Ay[i][j]=(nd.Ay[i+1][j+1]+nd.Ay[i+1][j]-nd.Ay[i][j+1]-nd.Ay[i][j])*0.5;
        nd.dy_Ay[i][j]=(nd.Ay[i+1][j+1]+nd.Ay[i][j+1]-nd.Ay[i+1][j]-nd.Ay[i][j])*0.5;
      }
    }

//トポロジカルチャージ変化
    nn=trapezoidal_Bz()*0.15915494;//Bz/2 π
    //変化した配位でエネルギー計算
    nE=trapezoidal2();
    nEm2=Em2;
    nE0=E0;
    nEp2=Ep2;

    //エネルギー比較条件緩和処理
    if((tcount==0)&&(count>(START*0.1))){
//    if((tcount==0)){
      tcount=TSTART;
      oE*=1.05;
    }
    //エネルギー比較して配位とゲージ場の値を変化させる。

    if(nE<oE){
      oE=nE;
```

```
    oEm2=nEm2;
    oE0=nE0;
    oEp2=nEp2;

    on=nn;
    tcount=TSTART;
    dcount++;
    od=nd;//ここで配位全部更新
  }

  if(oE!=nE)  tcount--;

  if(count%100000==0){
    data=fopen("SA_Abelian.txt","w"); //計算結果保存用
    for(i=MESH;i>=0;i--){
      for(j=MESH;j>=0;j--){
        fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h,
              od.F[i][j], od.iF[i][j], od.Ax[i][j], od.Ay[i][j],
               sqrt(od.F[i][j]*od.F[i][j]+od.iF[i][j]*od.iF[i][j]));
      }
      fprintf(data,"\n");
    }

    fclose(data);
    //磁場。
    data=fopen("SA_Abelian_Bz.txt","w"); //計算結果保存用
    for(i=MESH-1;i>=0;i--){
      for(j=MESH-1;j>=0;j--){
        fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h,
              rh*(od.dx_Ay[i][j]-od.dy_Ax[i][j]), Ed[i][j], cd[i][j]);
      }
      fprintf(data,"\n");
    }
    fclose(data);
  }

  count--;
  }while(count>0);
```

```
  data=fopen("SA_Abelian.txt","w"); //計算結果保存用
  for(i=MESH;i>=0;i--){
    for(j=MESH;j>=0;j--){
      fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h,
                od.F[i][j], od.iF[i][j], od.Ax[i][j], od.Ay[i][j],
                 sqrt(od.F[i][j]*od.F[i][j]+od.iF[i][j]*od.iF[i][j]));
    }
    fprintf(data,"\n");
  }

  fclose(data);
  //磁場。
  data=fopen("SA_Abelian_Bz.txt","w"); //計算結果保存用
  for(i=MESH-1;i>=0;i--){
    for(j=MESH-1;j>=0;j--){
      fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h,
                rh*(od.dx_Ay[i][j]-od.dy_Ax[i][j]), Ed[i][j], cd[i][j]);
    }
    fprintf(data,"\n");
  }
  fclose(data);

//横軸に count, 縦軸に,oE,on,oE/on のグラフを描く
  data=fopen("SA_Abelian_c-E.txt","w"); //計算結果保存用
  for(i=START/KAKUNIN;i>0;i--){
    fprintf(data,"%d\t%lf\t%lf\t%lf\t%lf\n", i*KAKUNIN,
              memo_E[i], memo_n[i], memo_En[i],memo_deri[i]);
  }
  fclose(data);

  printf("\n SA_Abelian");
  return 0;
}

//重積分用の台形公式
double trapezoidal2(void)
{
```

```
double S=0.0;
double s,t;//台形積分の端っこの点を計算するための係数
int i,j;//ループ用処理変数
double Ep2d,E0d,Em2d;
double h2=h*h;
double rh2=1/h/h;
double tF2,tA2,dx_F2,dy_F2,dx_iF2,dy_iF2,stA;//一時処理配列用変数

double para=mu*mu/lam;//計算用パラメータ μ^2/λ

for(i=MESH-1;i>=0;i--){
  for(j=MESH-1;j>=0;j--){
    tF[i][j]=(nd.F[i][j]+nd.F[i+1][j]+nd.F[i][j+1]+nd.F[i+1][j+1])*0.25;
    tiF[i][j]=(nd.iF[i+1][j+1]+nd.iF[i+1][j]+nd.iF[i][j+1]+nd.iF[i][j])*0.25;
    tAx[i][j]=(nd.Ax[i][j]+nd.Ax[i+1][j]+nd.Ax[i][j+1]+nd.Ax[i+1][j+1])*0.25;
    tAy[i][j]=(nd.Ay[i][j]+nd.Ay[i+1][j]+nd.Ay[i][j+1]+nd.Ay[i+1][j+1])*0.25;
  }
}

Ep2=0.0;
E0=0.0;
Em2=0.0;

for(i=MESH-1;i>=0;i--){
  if((i==0)||(i==MESH-1)) s=0.5;
  else s=1.0;
  for(j=MESH-1;j>=0;j--){
    if((j==0) || (j==MESH-1)) t=0.5;
    else t=1.0;
    tF2=tF[i][j]*tF[i][j]+tiF[i][j]*tiF[i][j];
    tA2=tAx[i][j]*tAx[i][j]+tAy[i][j]*tAy[i][j];
    dx_F2=nd.dx_F[i][j]*nd.dx_F[i][j];
    dy_F2=nd.dy_F[i][j]*nd.dy_F[i][j];
    dx_iF2=nd.dx_iF[i][j]*nd.dx_iF[i][j];
    dy_iF2=nd.dy_iF[i][j]*nd.dy_iF[i][j];
    stA=(nd.dx_Ay[i][j]-nd.dy_Ax[i][j]);

    Ep2d=h2*( 0.25*lam*(tF2-para)*(tF2-para))*s*t;
```

```
      E0d=( 0.5*q*q*(tA2*tF2) + 0.5*(dx_F2+dx_iF2+dy_F2+dy_iF2)
              +q*( tAx[i][j]*(-tF[i][j]*nd.dx_iF[i][j]+tiF[i][j]*nd.dx_F[i][j])
              +tAy[i][j]*(-tF[i][j]*nd.dy_iF[i][j]+tiF[i][j]*nd.dy_F[i][j]) ) )*s*t;
      Em2d=0.5*stA*stA*rh2*s*t;
      Ep2+=Ep2d;
      E0+=E0d;
      Em2+=Em2d;
      Ed[i][j]=Ep2d+E0d+Em2d;
    }
  }
  S=Em2+Ep2+E0;
  return S;
}

//トポロジカルチャージ求めるための Bz 積分
double trapezoidal_Bz(void)
{
  double S=0.0;
  double s,t;//台形積分の端っこの点を計算するための係数
  int i, j;
  for(i=MESH-1;i>=0;i--){
    if((i==0)||(i==MESH-1)) s=0.5;
    else s=1.0;
    for(j=MESH-1;j>=0;j--){
      if((j==0) || (j==MESH-1)) t=0.5;
      else t=1.0;
      cd[i][j]=( nd.dx_Ay[i][j]-nd.dy_Ax[i][j])*s*t;
      S+=cd[i][j];
    }
  }

  return S;
}

 void ransu(void)
{
  int rk,ri,rj;//新乱数処理用変数
  int random;//乱数保存用 (整数だよ！)
```

```
  int m;//ループ用
  double leng=0.0005;//Φの振幅パラメータ
  double aleng=0.0005;//A の振幅パラメータ

  //離散的に乱数ゆらす (ri,rj を利用して端から 2 点固定してる。ディリクレ&ノイマン境界
条件の役割)
  random=genrand_int31()%((MESH-3)*(MESH-3)*8);
  rk=random%8;
  random=(random-rk)*0.125;
  ri=random%(MESH-3);
  random=(random-ri)/(MESH-3);
  rj=random%(MESH-3);
  ri+=2;
  rj+=2;
  if(rk==0) nd.F[ri][rj]=od.F[ri][rj]+leng;
  switch(rk){
    case 1: nd.F[ri][rj]=od.F[ri][rj]-leng; break;
    case 2: nd.iF[ri][rj]=od.iF[ri][rj]+leng; break;
    case 3: nd.iF[ri][rj]=od.iF[ri][rj]-leng; break;
    case 4: nd.Ax[ri][rj]=od.Ax[ri][rj]+aleng; break;
    case 5: nd.Ax[ri][rj]=od.Ax[ri][rj]-aleng; break;
    case 6: nd.Ay[ri][rj]=od.Ay[ri][rj]+aleng; break;
    case 7: nd.Ay[ri][rj]=od.Ay[ri][rj]-aleng;
  }
}

//初期配位の定義
double func_F(double x,double y)
{
  double a=0.0001;
  double n=1.0;//初期チャージ
  return cos(n*myatan(y,x))*mu/sqrt(lam)*(1-exp(-a*(x*x+y*y)*(x*x+y*y)));
}

double func_iF(double x,double y)
{
  double a=0.0001;
  double n=1.0;//初期チャージ
```

```
  return sin(n*myatan(y,x))*mu/sqrt(lam)*(1-exp(-a*(x*x+y*y)*(x*x+y*y)));
}

double func_Ax(double x,double y)
{
  //初期チャージに合うように値を設定する
  double a=0.22;
  double b=0.355;
  return -b*y*exp(-a*sqrt(x*x+y*y));
}

double func_Ay(double x,double y)
{
  //初期チャージに合うように値を設定する
  double a=0.22;
  double b=0.355;
  return b*x*exp(-a*sqrt(x*x+y*y));
}

double myatan(double y, double x)
{
  double ret;
  if((x>=0)&&(y>=0)) ret=atan2(y,x);
  else if((x<=0)&&(y>=0)) ret=atan2(-x,y)+0.5*PI;
  else if((x<=0)&&(y<=0)) ret=atan2(-y,-x)+PI;
  else ret=atan2(x,-y)+1.5*PI;
  return ret;
}
```

5　異方的超伝導と Baby-Skyrme 模型

5.1　新たな超伝導体：異方的超伝導体

BCS 理論は低温状態における超伝導現象の解析には非常に有用であるが、高温超伝導体に適用するのは一般的には不可能とされている。BCS 理論では超伝導ギャップが等方的に開く S 波超伝導体を扱う。S 波超伝導体はスピン 1 重項のクーパー対が媒介となる超伝導体である。軌道部分が偶パリティならスピン部分は 1 重項となる。近年、BCS 理論の枠組みでは扱えない高温超伝導体が数多く発見されている。その中で非 S 波の超伝導体が現実の物質で見つかっている。それが異方的超伝導体である。この超伝導体は超伝導ギャップが等方的に開かない。軌道部分が奇パリティとなり、スピン 3 重項を形成する。代表的なスピン 3 重項の P 波超伝導体として Sr_2RuO_4 が知られていて、実験的に様々な面白い現象が発見されている。電子が強相関の系ではクーパー対を形成する電子の波動関数が中心にノードを持つ方がエネルギー的に得なので非 s 波の対形成が起こりやすい。強磁性的なスピン相関が強い系でクーパー対ができる場合、スピン 3 重項の形をとる方がエネルギー的に得をするので、奇パリティの超伝導が出現しやすい。(p 波、f 波) 一方、反強磁性的なスピン相関が強い系でクーパー対ができるとすれば、スピン 1 重項の d 波超伝導が出現するのが自然である。強相関電子系として扱われる酸化物系、重い電子系、有機超伝導体などの諸物質に関して異方的超伝導の研究が行われている。

異方的超伝導体である Sr_2RuO_4 では新たな現象が発見されている。そのうちの一つが磁束量子が半整数で入るということ。BCS 理論、通常の Ginzburg-Landau 理論、そして Abelian-Higgs 模型などでは磁束量子は整数となる。ところが、Sr_2RuO_4 では磁束量子が整数ではなく半整数単位で侵入する。参考文献 [14] に掲載されている実験の図が以下のもの。

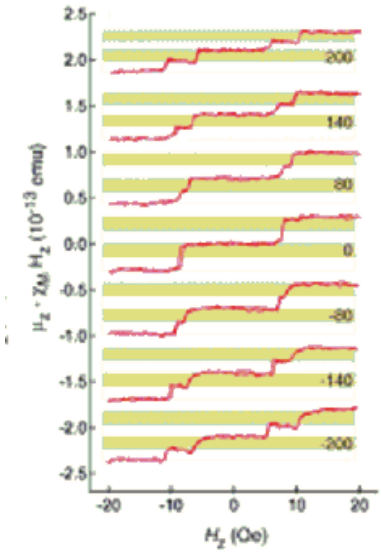


図 27: 外部磁場による半整数磁束量子の誘起

この図は Sr_2RuO_4 に外部磁場をかけていくと磁束量子が増えていくことを示したグラフである。図を見るとわかるように、外部磁場に誘起されて磁束量子が半整数単位で侵入していることが実験で示されている。本研究ではこのような現象を担う模型は Baby-Skyrme 模型ではないかと考え研究を行ってきた。根拠は主に 2 つ。まず、Baby-Skyrme 模型はスピン 3 重項が担う現象を記述する模型であること、次に、この模型が先ほど説明した磁束量子の半整数化に対応する Half-Skyrmion 解を持つこと。通常のトポロジカルソリトン模型では 1 つのソリトンが有することのできるチャージは整数のみであった。ところが、Baby-Skyrme 模型では様々な場合に Half-Skyrmion 解と呼ばれる半整数のチャージを持つ解が発見されている。本論文でも実際に半整数単位で存在する解を得ることに成功した。これは Baby-Skyrme 模型が近年発見された Sr_2RuO_4 の現象を記述できることを示唆するものではないだろうか？

半整数の磁束量子状態が Baby-Skyrme 模型で再現することが可能だとする。しかし、超伝導現象をこの模型で説明するにはまだ重要な機構が足りない。実験では外部磁場を増加させていくと磁束量子の数が増え続ける。この機構が Baby-Skyrme 模型には欠けている。外部磁場との相互作用により磁束量子が侵入するような模型に変化させなくてはならない。そのように考え私は卒業研究を行ってきた。次節以降では Baby-Skyrme 模型の解説を行い、代表的なポテンシャルで SA 法により数値解を得る。その後周期境界条件を付加し、超伝導体の格子構造を再現し、Half-Skyrmion 解が得られることを示す。そして続く章では実際に外部磁場によりチャージが増加する機構を Baby-Skyrme 模型に組み込み、数値解析を行う。

5.2 Baby-Skyrme 模型とは

Baby-Skyrme 模型は非線形 O(3) シグマ模型に安定項である Skyrme 項とポテンシャル項を付け加えた (2+1) 次元の模型 [16] であり、安定なトポロジカルソリトン解を有する。以下のラグランジアン密度によって記述される。

$$\mathcal{L} = V(\vec{n}) + \frac{M^2}{2}\partial_\mu \vec{n}\cdot\partial^\mu \vec{n} - \frac{\kappa^2}{2}(\partial_\mu \vec{n}\times\partial_\nu \vec{n})^2, \tag{63}$$

ここで、$\vec{n} = (n_1, n_2, n_3)$ は単位球面 S^2 上に任意の点を取る実スカラー場の三重項であり、拘束条件 $\vec{n}\cdot\vec{n} = 1$ を満たす。また、この模型はポテンシャル $V(\vec{n})$、運動項 $\frac{M^2}{2}\partial_\mu\vec{n}\cdot\partial^\mu\vec{n}$、そして Skyrme 項 $\frac{\kappa^2}{2}(\partial_\mu\vec{n}\times\partial_\nu\vec{n})^2$ から成り、ポテンシャルに 0 階微分の項を与えることでデリックの定理を満たすようになる。我々は静的な解に興味があるので時間微分についてはこの先考えない。

トポロジカルチャージは以下の式で定義される。

$$N = \frac{1}{4\pi}\iint d^2x\vec{n}\cdot\left(\partial_1\vec{n}\times\partial_2\vec{n}\right). \tag{64}$$

ここで、$V(\vec{n})$ に old-Baby ポテンシャル $\mu^2(1-n_3)$ を代入して解析を行ってみる。軸対称解を得るための Ansatz として、$n_1 = \sin F\cos\Theta, n_2 = \sin F\sin\Theta, n_3 = \cos F$ を課し、このとき、$\Theta = n\theta$,$F(r,\theta) = F(r)$ とすると、作用 S は

$$S = -n^2\sin^2 F(M^2 + \kappa^2 F'^2) + r^2\left\{2\mu^2(-1+\cos F) - M^2F'^2\right\} \tag{65}$$

となる。この作用から得られる微分方程式は、

$$\Big(M^2 r + \frac{\kappa^2 n^2 \sin^2 F}{r}\Big)F'' + \Big(M^2 - \frac{\kappa^2 n^2 \sin^2 F}{r^2} + \frac{\kappa^2 n^2 F' \cos F \sin F}{r}\Big)F' - \frac{M^2 n^2 \sin F \cos F}{r} - r\mu^2 \sin F = 0 \tag{66}$$

ここで、$r \to \infty$ において、F が微小であるとすると、(66) 式は

$$F'' + \frac{1}{r}F' - (\frac{n^2}{r^2} + \frac{\mu^2}{M^2})F = 0 \tag{67}$$

となる。トポロジカルチャージ $n = 1$ としたとき、(67) 式の解は変形ベッセル関数で記述できる。漸近解は、

$$F = \sqrt{\frac{M\pi}{2\mu r}} \exp(-\frac{\mu}{M} r) \tag{68}$$

となる。この表式は第 4 章の Abelian-Higgs 模型における軸対称解 (53) 式と同様の形をとることがわかる。つまり、Baby-Skyrme 模型における磁束の半径としての長さスケール ξ は式の形から、$\xi = \frac{M}{\mu}$ となることがわかり、Abelian-Higgs 模型におけるコヒーレンス長と対応すると考えられる。軸対称の漸近解の存在が解析的に明らかになったので次に数値計算により解の存在を確かめる。

2 次元直交座標系において、ラグランジアン密度 (63) 式から得られるエネルギー汎関数は、

$$E = \frac{1}{2}\iint_S dxdy\Big[2\mu^2(1 - n_3) + M^2\big\{(\partial_x \vec{n})^2 + (\partial_y \vec{n})^2\big\} + V(\vec{n}) + \kappa^2(\partial_x \vec{n} \times \partial_y \vec{n})^2\Big], \tag{69}$$

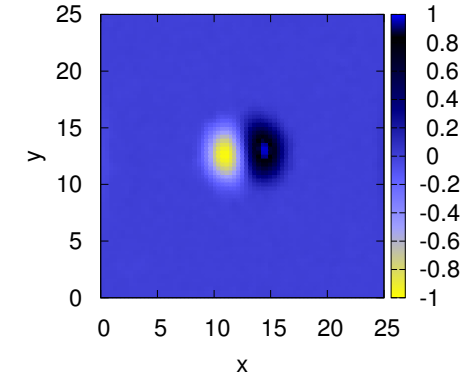
ここで、S は超伝導体の面積を表す。境界では以下の式を真空値とする。

$$\vec{n}_{edge} = (0, 0, 1) \tag{70}$$

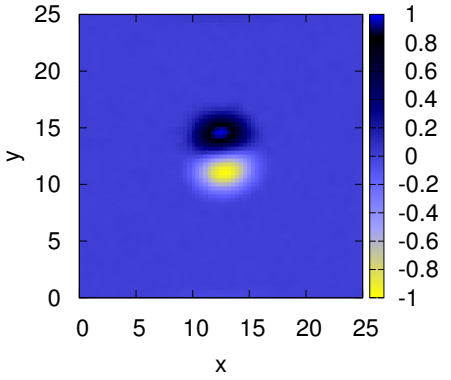
また、トポロジカルチャージ N は以下の式で表される。

$$N = \frac{1}{4\pi}\iint_S dxdy\Big[\vec{n} \cdot \big(\partial_x \vec{n} \times \partial_y \vec{n}\big)\Big]. \tag{71}$$

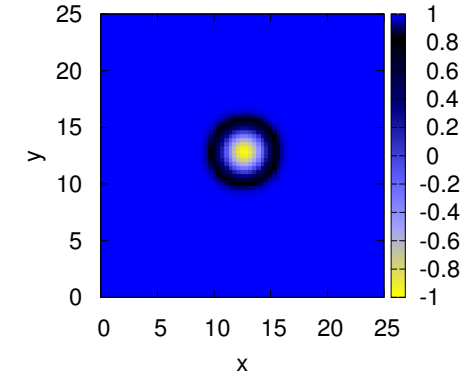
メッシュ 72×72、$M^2 = 0.1, \kappa = 1.0$、$\mu = \frac{1}{3}$ として E/N を最小化することで解を得る。このとき、単に E を最小化した場合チャージがすぐさま 0 に減少してしまい、真空解に陥ってしまう。そのため、チャージ N で E を割ることにより、チャージが減少するのを防ぐ。チャージ 1 から 5 までの解を以下の図に示す。



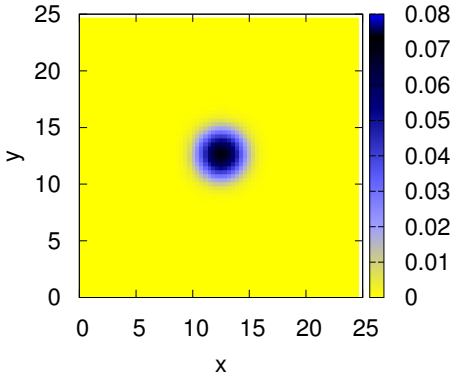
n_1



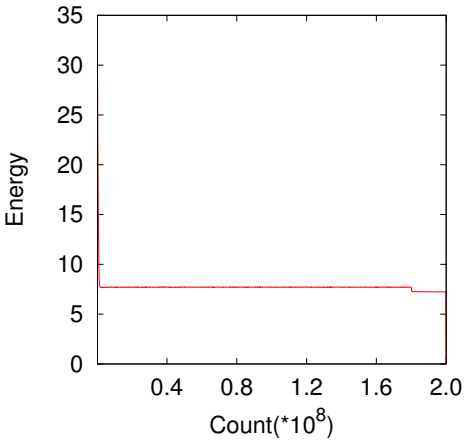
n_2



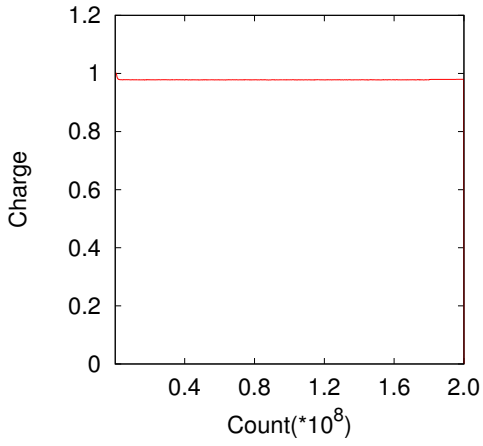
n_3



エネルギー密度

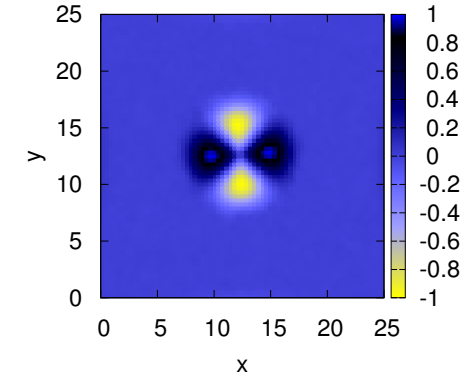


エネルギーの推移

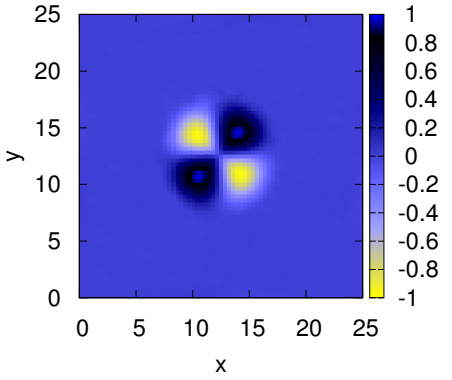


チャージの推移

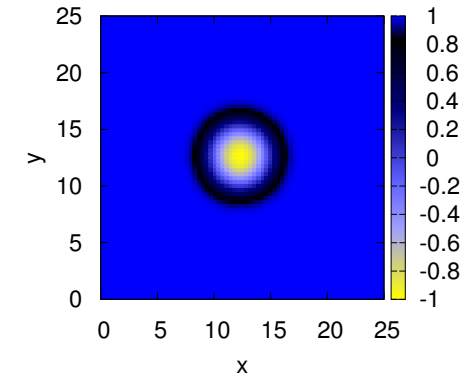
図 28: チャージ $N = 1$



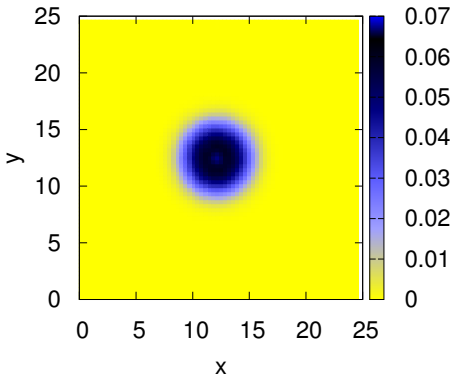
n_1



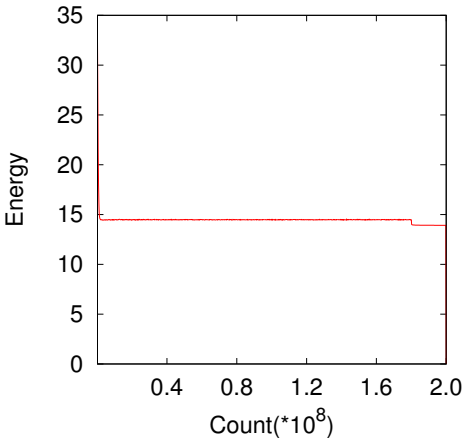
n_2



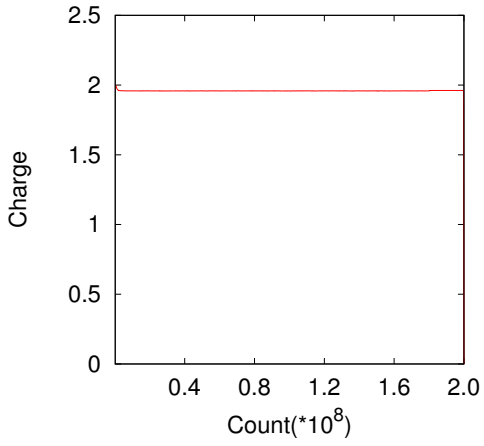
n_3



エネルギー密度

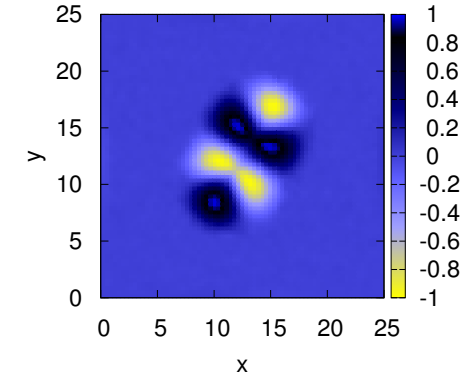


エネルギーの推移

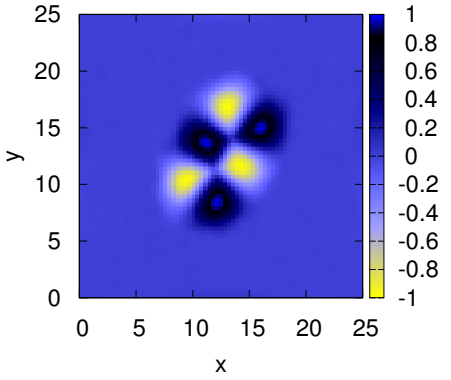


チャージの推移

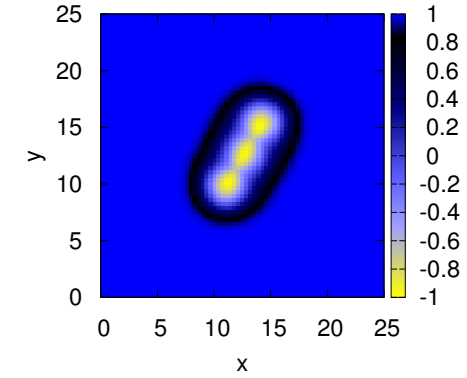
図 29: チャージ $N = 2$



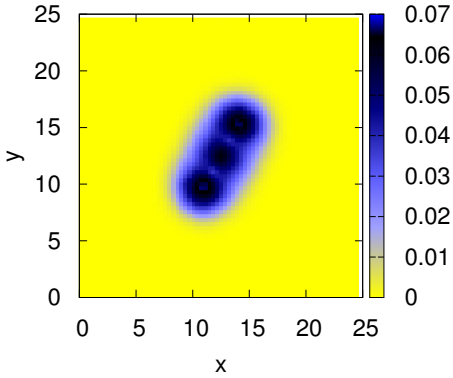
n_1



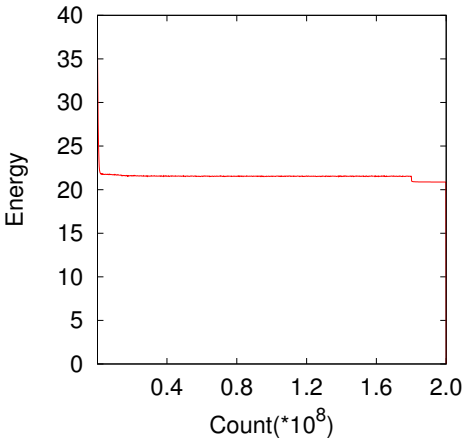
n_2



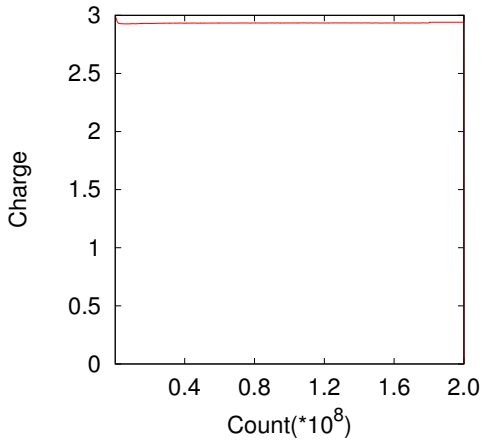
n_3



エネルギー密度

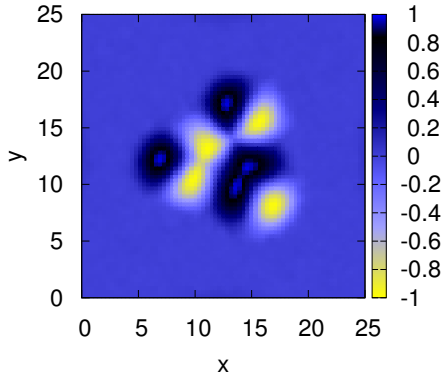


エネルギーの推移

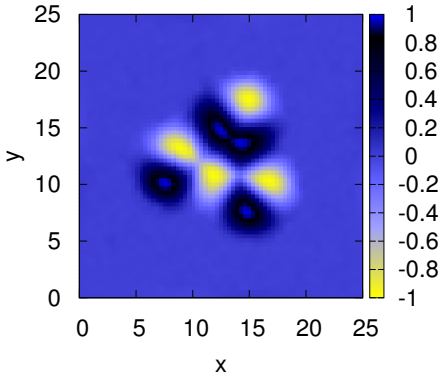


チャージの推移

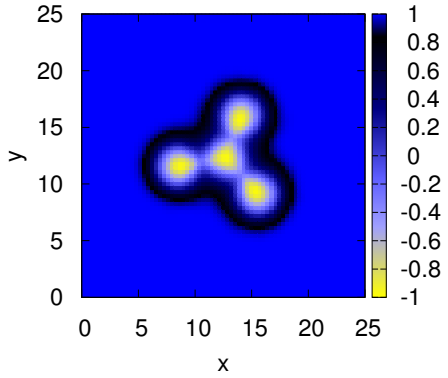
図 30: チャージ $N = 3$



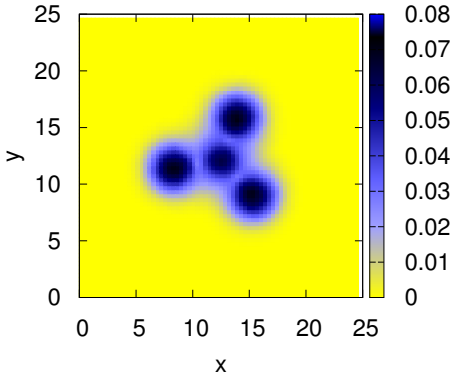
n_1



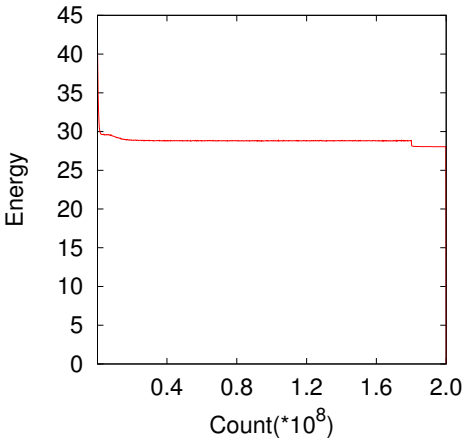
n_2



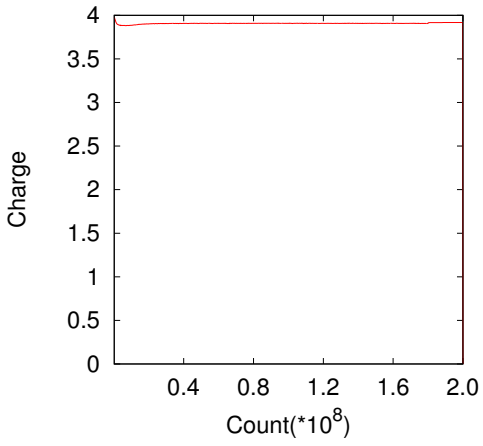
n_3



エネルギー密度

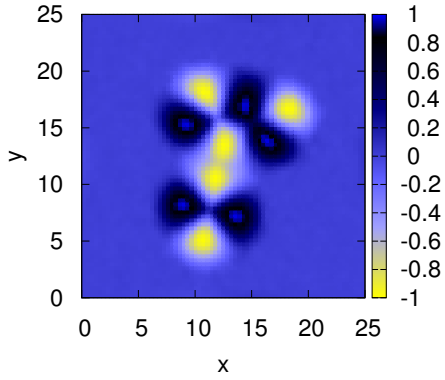


エネルギーの推移

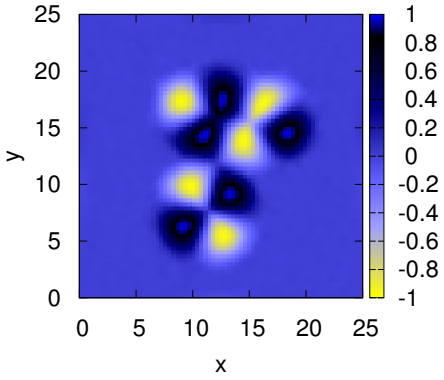


チャージの推移

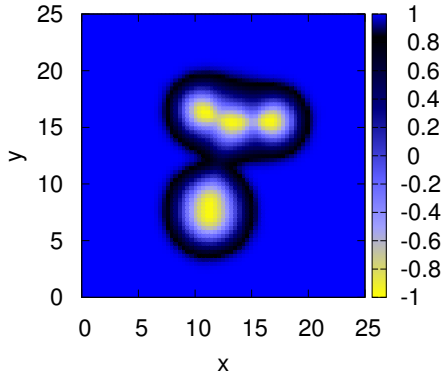
図 31: チャージ $N = 4$



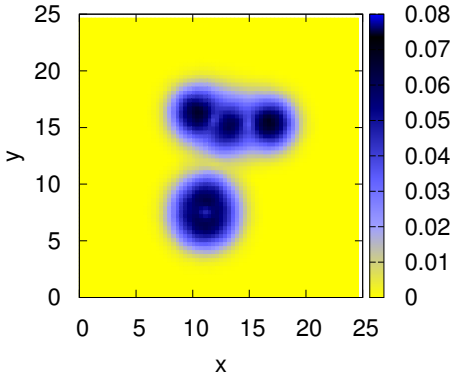
n_1



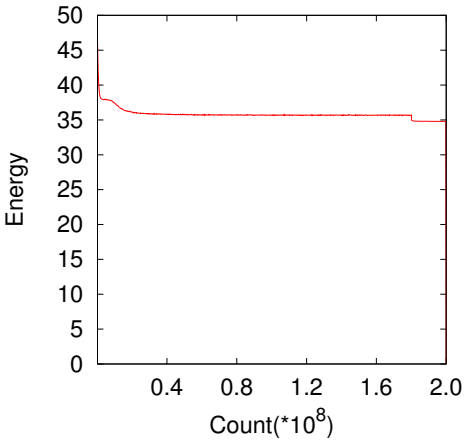
n_2



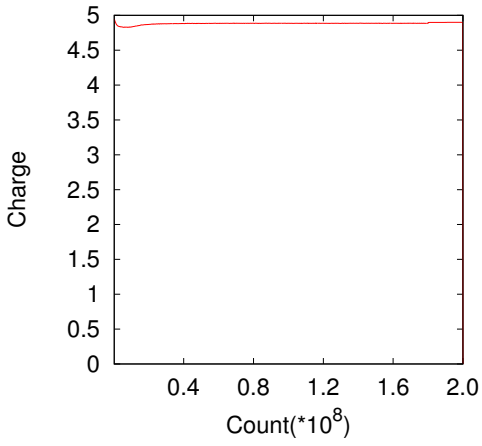
n_3



エネルギー密度



エネルギーの推移



チャージの推移

図 32: チャージ $N = 5$

チャージ N が 1〜5 の場合の数値解の挙動について述べる。まず、チャージ 1 の数値解は軸対称な Ansatz を課したことで得られた方程式の解析解 (68) 式と一致することがわかる。次にチャージ 2 の数値解は形状変化するが解の軸対称性は壊れておらず Ansatz を課した解析解と対応していることがわかる。しかし、チャージ 3 以降の解は明らかに軸対称性を保持していない。チャージ 3 では縦に 3 つのセンターができている multi-vortex 構造となっている。Abelian-Higgs 模型の数値解と形状などは異なるがソリトンセンターが分かれて配置されている。チャージ 4 の場合もソリトンセンターが 4 つ分かれた構造で安定化している。チャージ 5 の場合はチャージ 2 の軸対称解とチャージ 3 で 3 つのセンターからなる解に分かれている。チャージ 2 の軸対称解は安定であるが、チャージ 3 の 3 センターからなる解の安定性がより高いことがわかる。もしチャージ 2 の軸対称解が最安定であるならチャージ 4 の解は 2 つの軸対称解に分離しているはずだ。

以上から old-Baby ポテンシャルを用いることで multi vortex を得ることができる。ただし、このときの解の挙動は Abelian-Higgs 模型の場合とは異なることがわかる。

この節で用いた数値計算のソースコードを以下に掲載する。

```
//Baby-Skyrme 模型の数値解を SA 法で求める

#include <cstdio>
#include <cmath>
#include <ctime>
#include <cstdlib>
#include <cstring>
#include "mersenne.h"

#define PI 3.1443926584
//メッシュは奇数 !!
#define MESH 71 //メッシュの分割数 (xy 平面で共通)
#define MIN 0 //x,y の最少値
#define MAX 25 //x,y の最大値
#define START 200000000 //合計計算回数
#define TSTART 20 //TSTART 毎に 1 解コストが上がる方向への遷移を許す
#define KAKUNIN 200000 //kakunin 回毎にエネルギーやチャージの値をファイル出力

using namespace std;

int M1_2=(MESH+1)*(MESH+1);
double Ep2,E0,Em2,E0Bz; //各項のエネルギーの値を保存
double ka=1.0; //κ^2(4 次項)
double u=0.3333; //パラメータ μ (ポテンシャル 1 項目)
```

```
double M2=0.1; //2 次項の強度 M
double on,nn; //古い&新しいトポロジカルチャージ
double mmin=MIN;
double mmax=MAX;
double h;//刻み幅の指定 (x,y 方向)
double rh;//割り算をなくすための h の逆数

double Ed[MESH][MESH];//エネルギー密度
double cd[MESH][MESH];//チャージ密度
double memo_En[START/KAKUNIN+1]; //チャージ/エネルギーの推移を記録
double memo_n[START/KAKUNIN+1]; //チャージの推移を記録
double memo_E[START/KAKUNIN+1]; //エネルギーの推移を記録
double memo_Em2[START/KAKUNIN+1]; //Skyrme 項由来のエネルギーの推移を記録
double memo_E0[START/KAKUNIN+1]; //運動項由来のエネルギーの推移を記録
double memo_Ep2[START/KAKUNIN+1]; //ポテンシャル項由来のエネルギーの推移を記録

//エネルギー&チャージ計算用の変数 (場の平均値や計算の途中式格納用)
double sinF,cosF,sinT,cosT;
double dxn2,dyn2,dxndyn,dxn4,dyn4;//Dxn^2,Dyn^2.Dxn*Dyn,Dxn^4,Dyn^4
double n1[MESH+1][MESH+1],n2[MESH+1][MESH+1],n3[MESH+1][MESH+1];
double tn1[MESH][MESH],tn2[MESH][MESH],tn3[MESH][MESH],
           dx_n1[MESH][MESH],dy_n1[MESH][MESH],dx_n2[MESH][MESH],dy_n2[MESH][MESH],
           dx_n3[MESH][MESH],dy_n3[MESH][MESH];
double bT[(MESH+1)][(MESH+1)];//base とする角度 θ =bT+T として扱う

struct{
  double F[(MESH+1)][(MESH+1)];
  double T[(MESH+1)][(MESH+1)];
} od,nd;

//初期配位の定義
double func_F(double x, double y);
double func_T(double x, double y);
double func_bT(double x, double y);

//重積分用の台形公式
double trapezoidal2(void);
```

```
//トポロジカルチャージ計算用の積分
double trapezoidal_Bz(void);

//乱数処理部分呼び出し
void ransu(void);

//atan を計算する関数
double myatan(double y, double x);

int main(void)
{
  double oE,nE;//古いエネルギー、新しいエネルギー
  double oEm2,nEm2;//トポロジカルチャージ用のエネルギー部分
  double oE0,nE0;
  double oEp2,nEp2;
  double oE0Bz, nE0Bz;
  int count=START;//計算回数
  int tcount=TSTART;//この回数だけ配位の取り換え起きなかったら条件緩める。
  int dcount=0;//何回配位を取り換えたか
  int i,j;//ループ用
  FILE *data,*fp;
  char name[80];//場の配位のファイル名
  char name2[80];//エネルギー、チャージ密度などの情報
  char bs[3];//磁場あり=bb, 磁場なし=aa(拡張用変数。old-Baby のときは不要)
  int number=0;//保存回数 (ファイル名に保存回数の情報を入れる)
  double dammy;

  h=(mmax-mmin)/MESH;
  rh=1.0/h;

  //配位を初期関数で定める
  for(i=MESH;i>=0;i--){
    for(j=MESH;j>=0;j--){
      od.F[i][j]=func_F(-(MESH)*0.5+1.0*i, -(MESH)*0.5+1.0*j);
      od.T[i][j]=func_T(-(MESH)*0.5+1.0*i, -(MESH)*0.5+1.0*j);
      bT[i][j]=func_bT(-(MESH)*0.5+1.0*i, -(MESH)*0.5+1.0*j);
    }
  }
```

```
nd=od;

//計算用の n1,n2,n3 の定義
for(i=MESH;i>=0;i--){
  for(j=MESH;j>=0;j--){
    n1[i][j]=sin(nd.F[i][j])*cos(nd.T[i][j]+bT[i][j]);
    n2[i][j]=sin(nd.F[i][j])*sin(nd.T[i][j]+bT[i][j]);
    n3[i][j]=cos(nd.F[i][j]);
  }
}
//n 場の偏微分の定義
for(i=MESH-1;i>=0;i--){
  for(j=MESH-1;j>=0;j--){
    dx_n1[i][j]=(n1[i+1][j+1]+n1[i+1][j]-n1[i][j+1]-n1[i][j])*0.5;
    dy_n1[i][j]=(n1[i+1][j+1]+n1[i][j+1]-n1[i+1][j]-n1[i][j])*0.5;
    dx_n2[i][j]=(n2[i+1][j+1]+n2[i+1][j]-n2[i][j+1]-n2[i][j])*0.5;
    dy_n2[i][j]=(n2[i+1][j+1]+n2[i][j+1]-n2[i+1][j]-n2[i][j])*0.5;
    dx_n3[i][j]=(n3[i+1][j+1]+n3[i+1][j]-n3[i][j+1]-n3[i][j])*0.5;
    dy_n3[i][j]=(n3[i+1][j+1]+n3[i][j+1]-n3[i+1][j]-n3[i][j])*0.5;
  }
}
//場の平均値の定義
for(i=MESH-1;i>=0;i--){
  for(j=MESH-1;j>=0;j--){
    tn1[i][j]=(n1[i+1][j+1]+n1[i+1][j]+n1[i][j+1]+n1[i][j])*0.25;
    tn2[i][j]=(n2[i+1][j+1]+n2[i+1][j]+n2[i][j+1]+n2[i][j])*0.25;
    tn3[i][j]=(n3[i+1][j+1]+n3[i+1][j]+n3[i][j+1]+n3[i][j])*0.25;

  }
}

//再びエネルギー計算
nn=trapezoidal_Bz();
nE=trapezoidal2();
nEm2=Em2;
nE0Bz=E0Bz;
nE0=E0;
```

```
nEp2=Ep2;
oE=nE;
oEm2=nEm2;
oE0Bz=nE0Bz;
oE0=nE0;
oEp2=nEp2;

on=nn;

strcpy(bs,"bb");
sprintf(name,"oldbaby%s_%d.txt" ,bs ,number);
sprintf(name2,"oldbaby%s_%d_Bz.txt" ,bs ,number);

data=fopen(name,"w"); //計算結果保存用
for(i=MESH;i>=0;i--){
  for(j=MESH;j>=0;j--){
    fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h,
                nd.F[i][j], nd.T[i][j], n1[i][j], n2[i][j],n3[i][j]);
  }
  fprintf(data,"\n");
}
fclose(data);
//磁場。
data=fopen(name2,"w"); //計算結果保存用
for(i=MESH-1;i>=0;i--){
  for(j=MESH-1;j>=0;j--){
    fprintf(data,"%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h, Ed[i][j], cd[i][j]);
  }
  fprintf(data,"\n");
}
fclose(data);

//ここからスカラー場とゲージ場を乱数で変化させていく
  init_genrand((unsigned)time(NULL));
  do{

    //確認用
    if(count%KAKUNIN==0){
```

```
    memo_E[(START-count)/KAKUNIN]=oE;
    memo_n[(START-count)/KAKUNIN]=on;
    memo_En[(START-count)/KAKUNIN]=oE/on;
    memo_Em2[(START-count)/KAKUNIN]=oEm2;
    memo_E0[(START-count)/KAKUNIN]=oE0;
    memo_Ep2[(START-count)/KAKUNIN]=oEp2;
    printf("%lf %lf %lf %lf %lf %lf %d %lf \n",oE,Ep2,E0,E0Bz,Em2,on,dcount,oE/on);
    dcount=0;
  }
  if(count%1000000==0) printf("\n 残り%d\n",count);

  //配位、ゲージ場を乱数でゆらしてやる
  if(count>(START*0.4)){
    double tmp=M1_2*0.8;
    for(i=tmp;i>=0;i--){
      ransu();
    }
  }
  else if((count>(START*0.1))&&((count<(START*0.4)))) {
     double tmp=M1_2*0.8;
    for(i=tmp;i>=0;i--){
      ransu();
    }
  }
  else {
    double tmp=M1_2*0.8;
    for(i=tmp;i>=0;i--){
      ransu();
    }
  }

//計算用の n1,n2,n3 の定義
for(i=MESH;i>=0;i--){
  for(j=MESH;j>=0;j--){
    n1[i][j]=sin(nd.F[i][j])*cos(nd.T[i][j]+bT[i][j]);
    n2[i][j]=sin(nd.F[i][j])*sin(nd.T[i][j]+bT[i][j]);
    n3[i][j]=cos(nd.F[i][j]);
  }
```

```
}
//n 場の偏微分の定義
for(i=MESH-1;i>=0;i--){
  for(j=MESH-1;j>=0;j--){
    dx_n1[i][j]=(n1[i+1][j+1]+n1[i+1][j]-n1[i][j+1]-n1[i][j])*0.5;
    dy_n1[i][j]=(n1[i+1][j+1]+n1[i][j+1]-n1[i+1][j]-n1[i][j])*0.5;
    dx_n2[i][j]=(n2[i+1][j+1]+n2[i+1][j]-n2[i][j+1]-n2[i][j])*0.5;
    dy_n2[i][j]=(n2[i+1][j+1]+n2[i][j+1]-n2[i+1][j]-n2[i][j])*0.5;
    dx_n3[i][j]=(n3[i+1][j+1]+n3[i+1][j]-n3[i][j+1]-n3[i][j])*0.5;
    dy_n3[i][j]=(n3[i+1][j+1]+n3[i][j+1]-n3[i+1][j]-n3[i][j])*0.5;
  }
}
//場の平均値の定義
for(i=MESH-1;i>=0;i--){
  for(j=MESH-1;j>=0;j--){
    tn1[i][j]=(n1[i+1][j+1]+n1[i+1][j]+n1[i][j+1]+n1[i][j])*0.25;
    tn2[i][j]=(n2[i+1][j+1]+n2[i+1][j]+n2[i][j+1]+n2[i][j])*0.25;
    tn3[i][j]=(n3[i+1][j+1]+n3[i+1][j]+n3[i][j+1]+n3[i][j])*0.25;
  }
}

//変化した配位でトポロジカルチャージ計算
nn=trapezoidal_Bz();
//変化した配位でエネルギー計算
nE=trapezoidal2();
nEm2=Em2;
nE0Bz=E0Bz;
nE0=E0;
nEp2=Ep2;

//エネルギー比較条件緩和処理
if((tcount==0)&&(count>(START*0.1))){
  tcount=TSTART;
  oE*=1.1;
}
//エネルギー比較して配位の値を変化させる。
if((nE/fabs(nn))<(oE/fabs(on))){
  oE=nE;
```

```
  oEm2=nEm2;
  oE0Bz=nE0Bz;
  oE0=nE0;
  oEp2=nEp2;
  on=nn;
  tcount=TSTART;
  dcount++;

  od=nd;//ここで配位全部更新
}

if(oE!=nE)  tcount--;

//状況確認用↓
if(count%100000==0){
  number++;
  sprintf(name,"oldbaby%s_%d.txt" ,bs ,number);
  sprintf(name2,"oldbaby%s_%d_Bz.txt" ,bs ,number);
  data=fopen(name,"w"); //計算結果保存用
  for(i=MESH;i>=0;i--){
    for(j=MESH;j>=0;j--){
      fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h,
           nd.F[i][j], nd.T[i][j], n1[i][j], n2[i][j],n3[i][j]);
    }
    fprintf(data,"\n");
  }
  fclose(data);
  data=fopen(name2,"w"); //計算結果保存用
  for(i=MESH-1;i>=0;i--){
    for(j=MESH-1;j>=0;j--){
      fprintf(data,"%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h, Ed[i][j], cd[i][j]);
    }
    fprintf(data,"\n");
  }
  fclose(data);
}

count--;
```

```
  }while(count>0);



  //最終計算結果
  data=fopen("oldbabya.txt","w"); //計算結果保存用
  for(i=MESH;i>=0;i--){
    for(j=MESH;j>=0;j--){
      fprintf(data,"%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h,
                nd.F[i][j], nd.T[i][j], n1[i][j], n2[i][j],n3[i][j]);
    }
    fprintf(data,"\n");
  }
  fclose(data);

  data=fopen("oldbabya_Bz.txt","w"); //計算結果保存用
  for(i=MESH-1;i>=0;i--){
    for(j=MESH-1;j>=0;j--){
      fprintf(data,"%lf\t%lf\t%lf\t%lf\n", mmin+i*h, mmin+j*h, Ed[i][j], cd[i][j]);
    }
    fprintf(data,"\n");
  }
  fclose(data);

//横軸に count, 縦軸に,oE,on,oE/on などのグラフを描く
  data=fopen("oldbabya_c-E.txt","w"); //計算結果保存用
  for(i=START/KAKUNIN;i>0;i--){
    fprintf(data,"%d\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", i*KAKUNIN,
          memo_E[i],memo_Ep2[i],memo_E0[i],memo_Em2[i] , memo_n[i], memo_En[i],memo_En[i]);
  }
  fclose(data);

  return 0;
}

//重積分用の台形公式
double trapezoidal2(void)
{
```

```
  double S=0.0;
  double s,t;//台形積分の端っこの点を計算するための係数
  int i,j;//ループ用処理変数

  double Ep2d,E0d,Em2d,E0Bzd;
  double h2=h*h;
  double rh2=1/h/h;

  Ep2=0.0;
  E0=0.0;
  Em2=0.0;
  E0Bz=0.0;
  for(i=MESH-1;i>=0;i--){
    if((i==0)||(i==MESH-1)) s=0.5;
    else s=1.0;
    for(j=MESH-1;j>=0;j--){
      if((j==0) || (j==MESH-1)) t=0.5;
      else t=1.0;

      dxn2=(dx_n1[i][j])*(dx_n1[i][j])+(dx_n2[i][j])*(dx_n2[i][j])+dx_n3[i][j]*dx_n3[i][j];
      dyn2=(dy_n1[i][j])*(dy_n1[i][j])+(dy_n2[i][j])*(dy_n2[i][j])+dy_n3[i][j]*dy_n3[i][j];
      dxndyn=(dx_n1[i][j])*(dy_n1[i][j])+(dx_n2[i][j])*(dy_n2[i][j])+dx_n3[i][j]*dy_n3[i][j];

      Ep2d=h2*( u*u*(1-tn3[i][j]) )*s*t;
      E0d=( 0.5*M2*(dxn2+dyn2) )*s*t;
      Em2d=0.5*ka*(dxn2*dyn2-dxndyn*dxndyn)*rh2*s*t;
      E0Bzd=0.0;

      Ep2+=Ep2d;
      E0+=E0d;
      Em2+=Em2d;
      E0Bz+=E0Bzd;
      Ed[i][j]=Ep2d+E0d+Em2d+E0Bzd;
    }
  }
  S=Em2+Ep2+E0+E0Bz;
  return S;
}
```

```
//トポロジカルチャージ求めるための積分
double trapezoidal_Bz(void)
{
  double S=0.0;
  double s,t;//台形積分の端っこの点を計算するための係数
  int i, j,m;
  double cdx_n1,cdx_n2,cdx_n3,cdy_n1,cdy_n2,cdy_n3;
  for(i=MESH-1;i>=0;i--){
    if((i==0)||(i==MESH-1)) s=0.5;
    else s=1.0;
    for(j=MESH-1;j>=0;j--){
      if((j==0) || (j==MESH-1)) t=0.5;
      else t=1.0;
      cdx_n1=dx_n1[i][j];
      cdx_n2=dx_n2[i][j];
      cdx_n3=dx_n3[i][j];
      cdy_n1=dy_n1[i][j];
      cdy_n2=dy_n2[i][j];
      cdy_n3=dy_n3[i][j];

      cd[i][j]=-( tn1[i][j]*(cdx_n2*cdy_n3-cdx_n3*cdy_n2)
                  +tn2[i][j]*(cdx_n3*cdy_n1-cdx_n1*cdy_n3)
                  +tn3[i][j]*(cdx_n1*cdy_n2-cdx_n2*cdy_n1))*s*t;
      S+=cd[i][j];
    }
  }
  return S*0.080577;
}

 void ransu(void)
{
  int rk,ri,rj;//新乱数処理用変数
  int random;//乱数保存用 (整数！)

  double leng=0.001;//初期振幅パラメータ
  double tleng=0.001;
  double aleng=0.001;
```

```
  //離散的に乱数ゆらす
  random=genrand_int31()%((MESH-3)*(MESH-3)*4);
  rk=random%4;
  random=(random-rk)*0.25;
  ri=random%(MESH-3);
  random=(random-ri)/(MESH-3);
  rj=random%(MESH-3);
  ri+=2;
  rj+=2;
  if(rk==0) nd.F[ri][rj]=od.F[ri][rj]+leng;
  switch(rk){
    case 1: nd.F[ri][rj]=od.F[ri][rj]-leng; break;
    case 2: nd.T[ri][rj]=od.T[ri][rj]+tleng; break;
    case 3: nd.T[ri][rj]=od.T[ri][rj]-tleng;
  }
}

//初期配位の定義
double func_F(double x,double y)
{
  double a=0.0001;
  return PI*(exp(-a*(x*x+y*y)*sqrt(x*x+y*y)) );
}

double func_T(double x,double y)
{
  return 0.0;
}

double func_bT(double x,double y)
{
  return 1.0*myatan(y,x);
}

double myatan(double y, double x)
{
  double ret;
  if((x>=0)&&(y>=0)) ret=atan2(y,x);
```

```
  else if((x<=0)&&(y>=0)) ret=atan2(-x,y)+0.5*PI;
  else if((x<=0)&&(y<=0)) ret=atan2(-y,-x)+PI;
  else ret=atan2(x,-y)+1.5*PI;
  return ret;
}
```

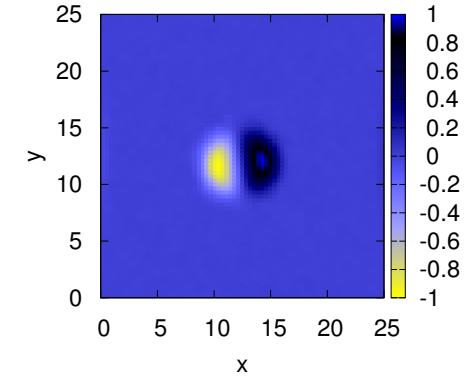
5.3 様々なポテンシャルにおける数値解

前節では old-Baby ポテンシャル $V(\vec{n}) = \mu^2(1 - n_3)$ の場合について、Baby-Skyrme 模型の数値解を求めた。ポテンシャルは物理的な解構造を決定するものである。我々は目的にかなったポテンシャルを用いることで多様な解構造を得ることができる。超伝導現象にこの模型を適用する場合、解構造は磁束構造に対応するのでポテンシャルが超伝導体の物質構造を担う。前節とは異なる興味深いポテンシャルを 2 種類用い、実際に数値解を求めていく。

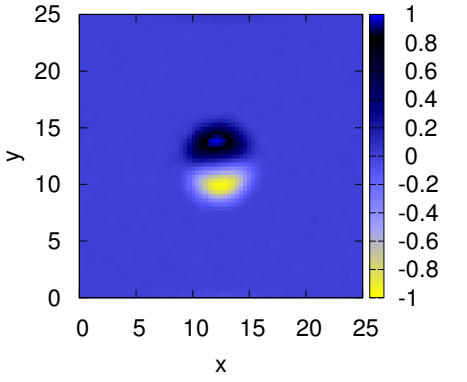
まず、old-Baby ポテンシャルの代わりに以下の new-Baby ポテンシャル [32] を用いる。

$$V(\vec{n}) = \mu^2(1 - n_3^2) \tag{72}$$

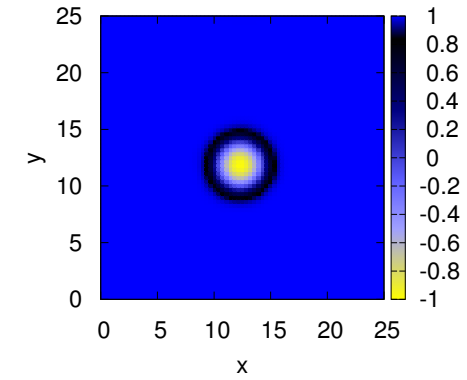
old-Baby ポテンシャルのときは $n_3 = 1$ のときのみ $V(\vec{n}) = 0$ となったが、new-Baby ポテンシャルの場合は $n_3 = \pm 1$ のときに $V(\vec{n}) = 0$ となる。この真空構造の違いが解構造を変化させる。数値計算の条件は前節の old-Baby ポテンシャルのときと同様である。SA 法で得られた図を以下に示す。



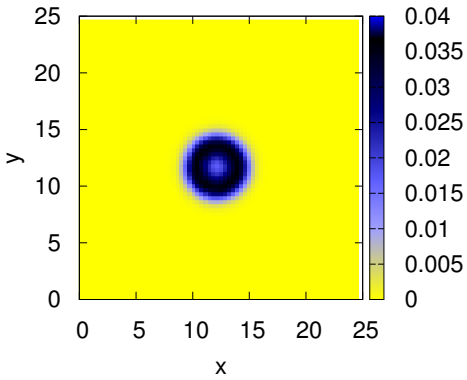
n_1



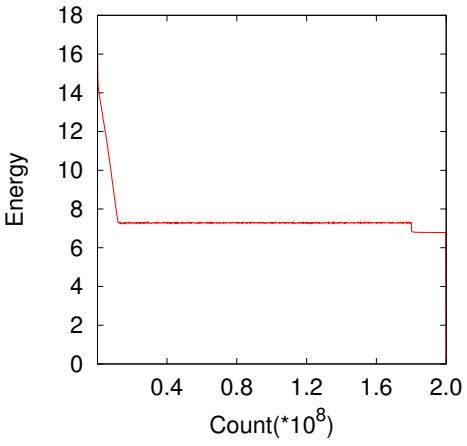
n_2



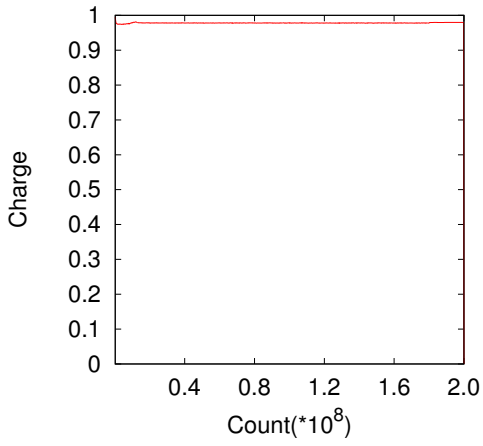
n_3



エネルギー密度

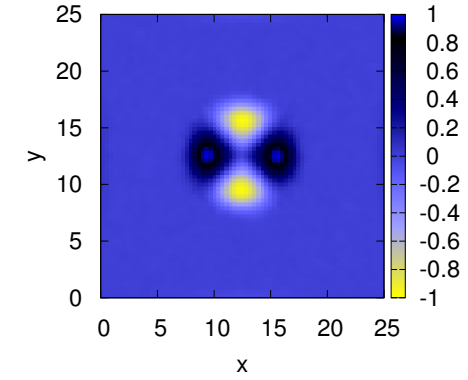


エネルギーの推移

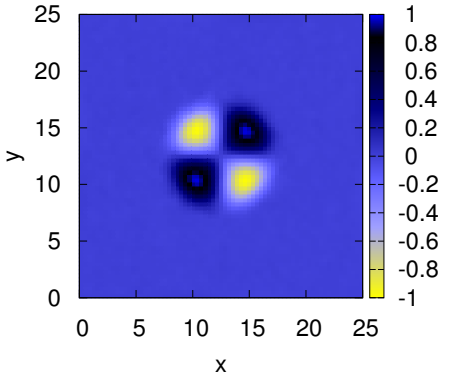


チャージの推移

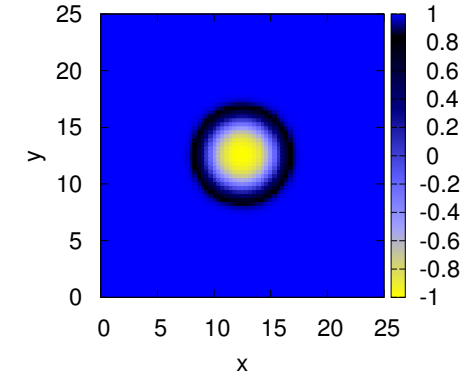
図 33: チャージ $N = 1$



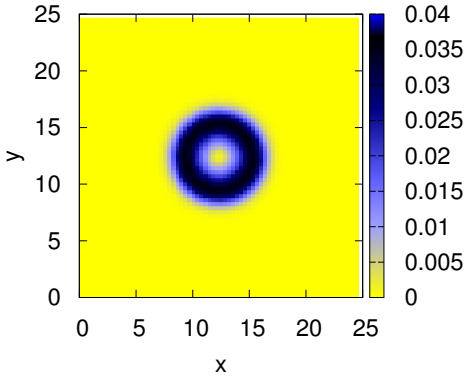
n_1



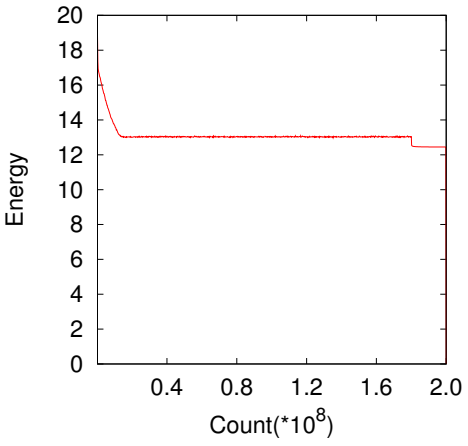
n_2



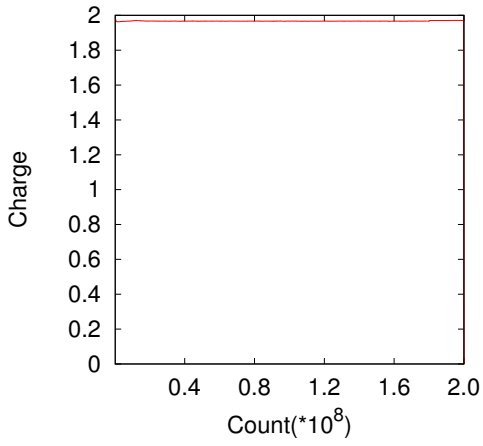
n_3



エネルギー密度

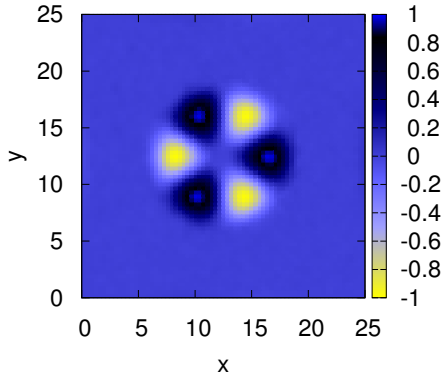


エネルギーの推移

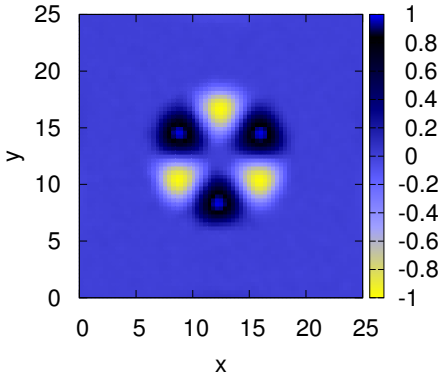


チャージの推移

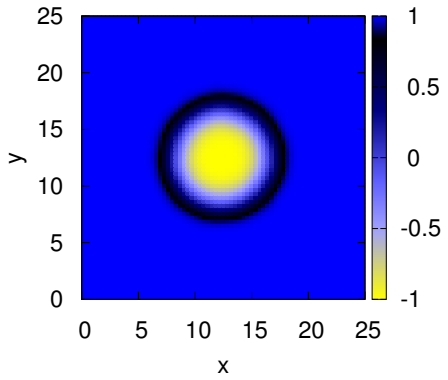
図 34: チャージ $N = 2$



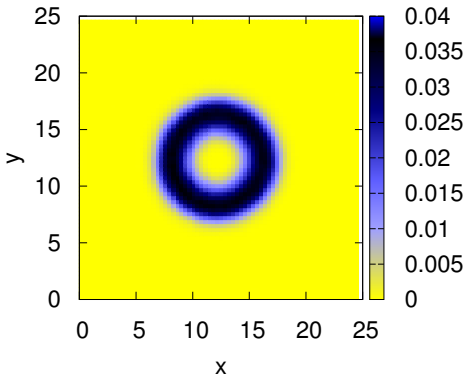
n_1



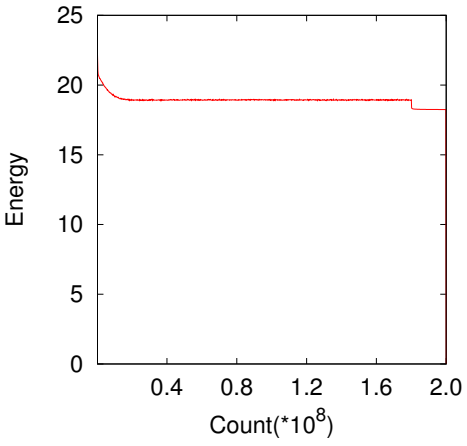
n_2



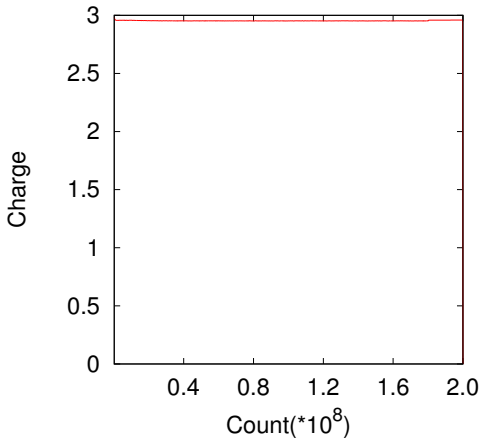
n_3



エネルギー密度

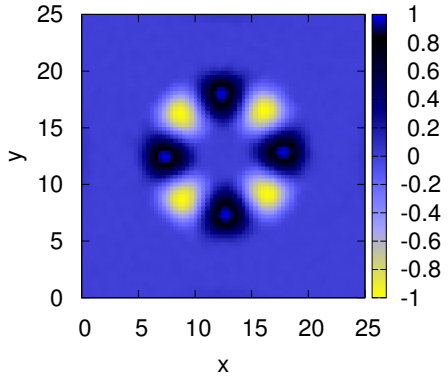


エネルギーの推移

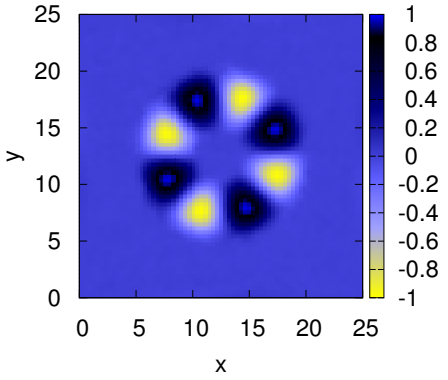


チャージの推移

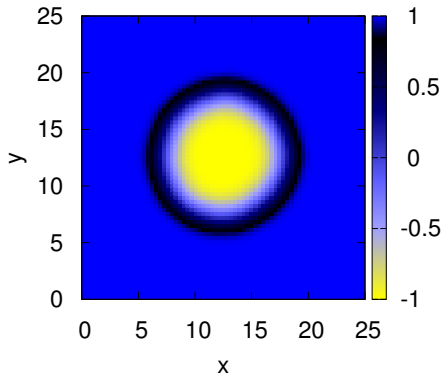
図 35: チャージ $N = 3$



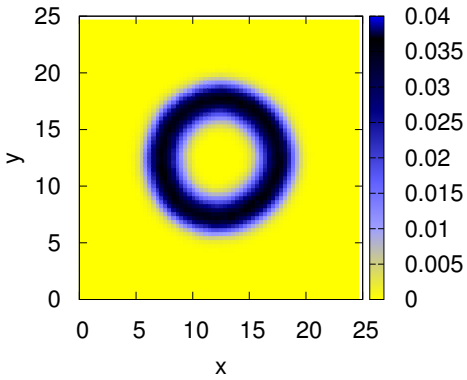
n_1



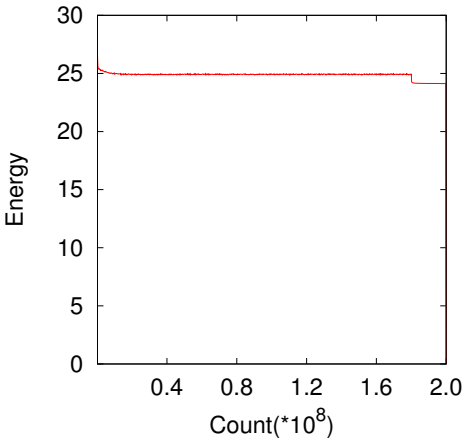
n_2



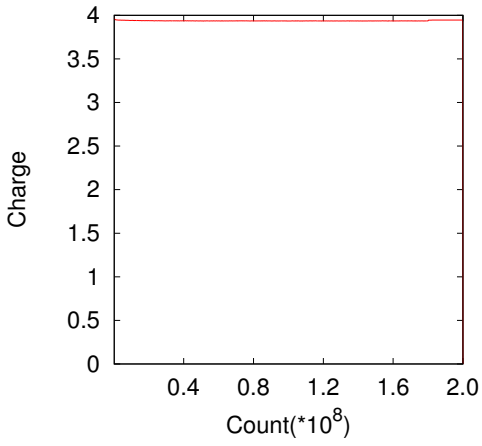
n_3



エネルギー密度

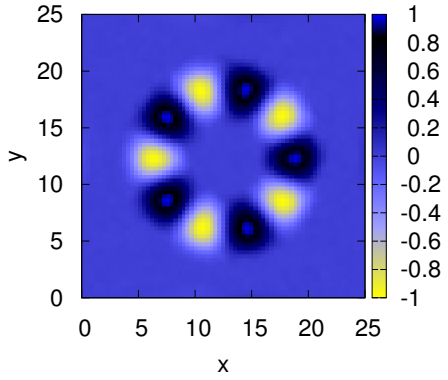


エネルギーの推移

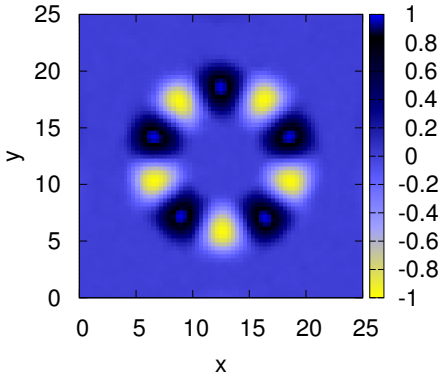


チャージの推移

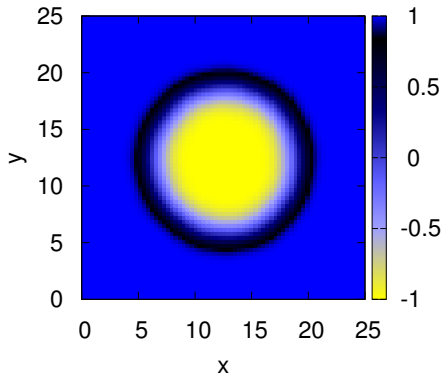
図 36: チャージ $N = 4$



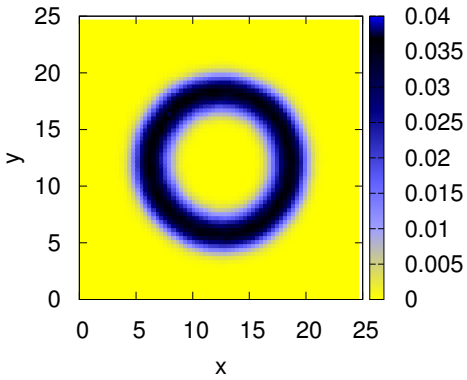
n_1



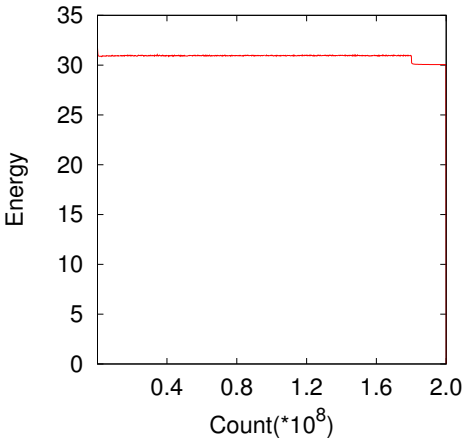
n_2



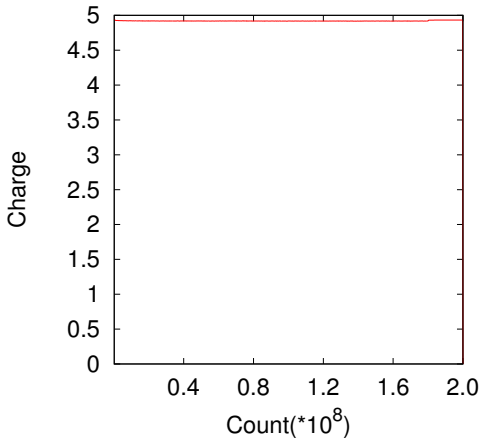
n_3



エネルギー密度



エネルギーの推移



チャージの推移

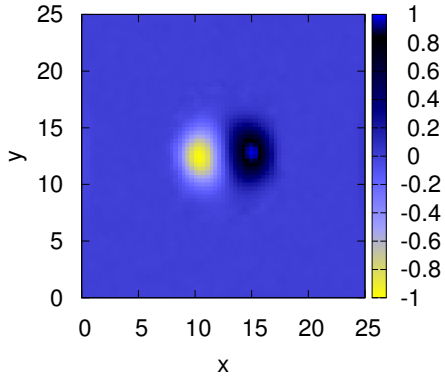
図 37: チャージ $N = 5$

チャージ 1〜5 についての数値解を掲載した。old-Baby ポテンシャルとの違いは一目瞭然で、new-Baby ポテンシャルではチャージ $N \geq 3$ の場合も軸対称解のままである。この解の形状は定性的に超伝導体における Gigantic Vortex に対応する解となる。Abelian-Higgs 模型においてはパラメータの大きさにより引力効果と斥力効果の寄与を調節することができた。Baby-Skyrme 模型においてはパラメータによる引力斥力効果の概念などが存在しないためポテンシャルを変化させることで現象に対応する模型を構築することになる。

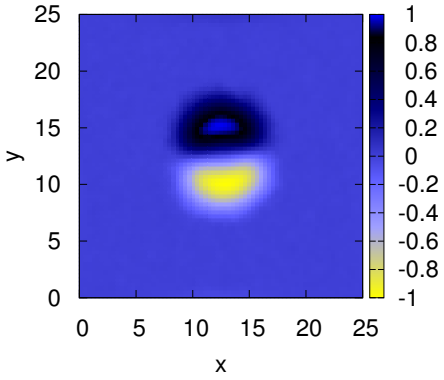
次に、もう一つの非常に興味深いポテンシャルを紹介する。ポテンシャルの形は以下の式で表される。

$$V(\vec{n}) = \mu^2 n_1^2 \tag{73}$$

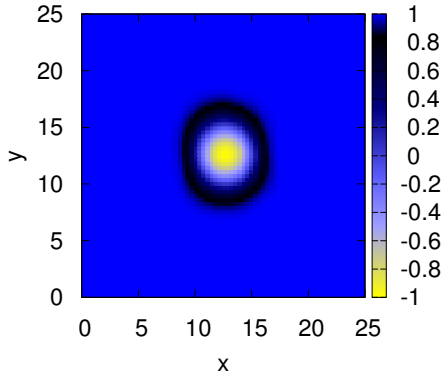
このポテンシャルは近年発見されたものでポテンシャルの値を 0 にする値が境界での真空を支配する n_3 ではなく n_1 に依存する。(参考文献 [33] では境界での真空値を $n_1 = 1$ にとっているためポテンシャルは n_3 によって記述されている。) 本論文ではこのポテンシャルを n_1 ポテンシャルと仮称する。n_1 ポテンシャルを用いた場合の Baby-Skyrme 模型の数値解は以下の通りである。今回の計算もチャージ N が 1〜5 まで変化する場合について行い、パラメータやメッシュなどは old-Baby,new-Baby ポテンシャルの場合と同様である。



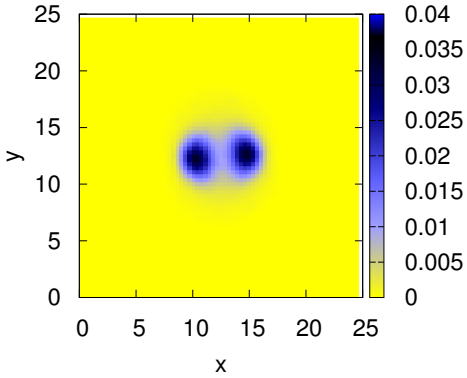
n_1



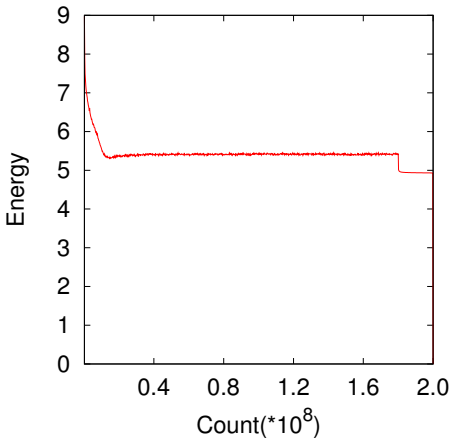
n_2



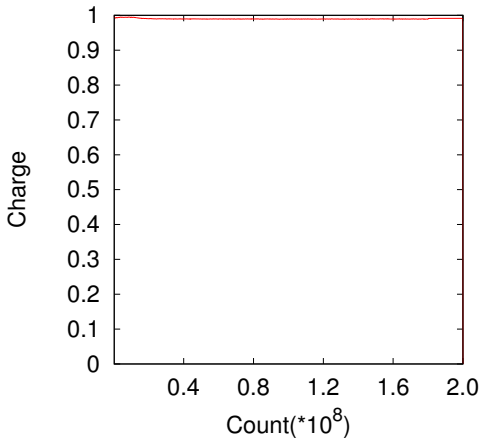
n_3



エネルギー密度

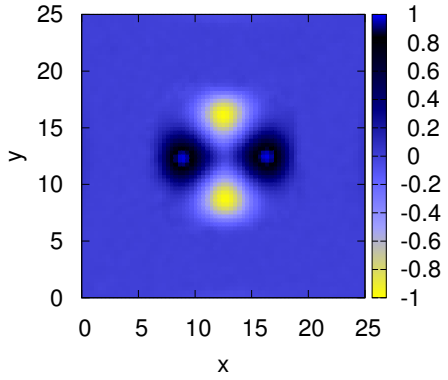


エネルギーの推移

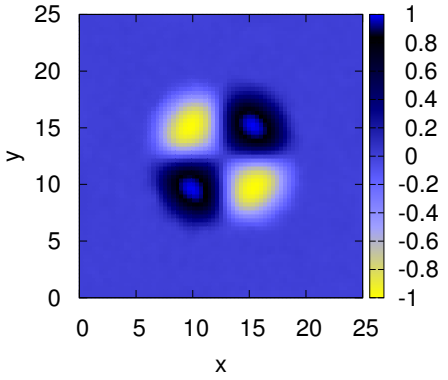


チャージの推移

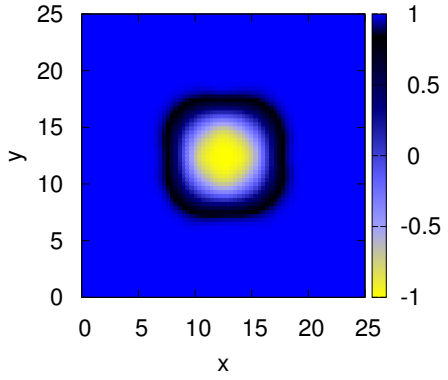
図 38: チャージ $N = 1$



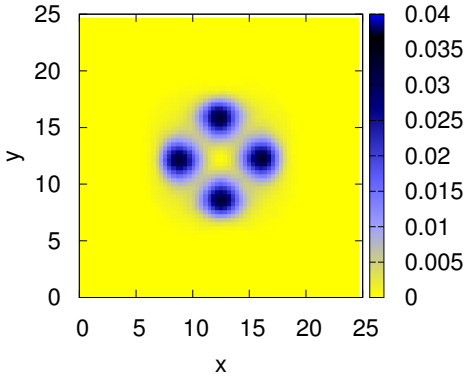
n_1



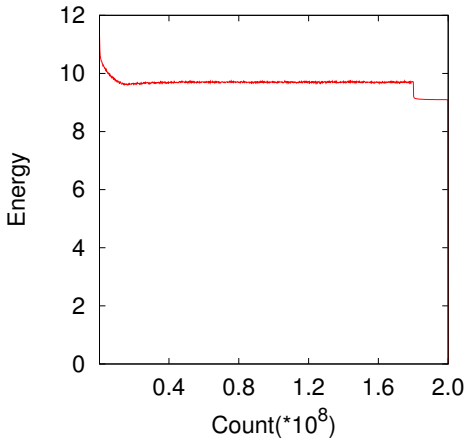
n_2



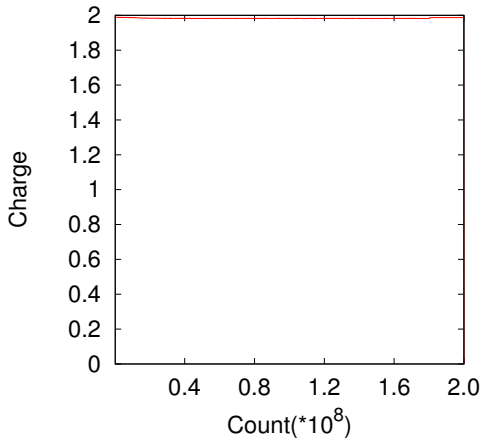
n_3



エネルギー密度

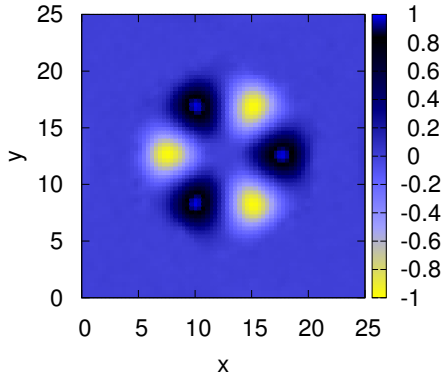


エネルギーの推移

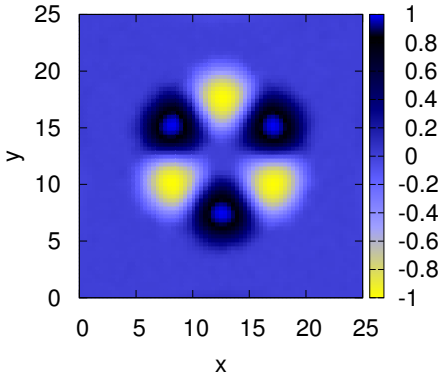


チャージの推移

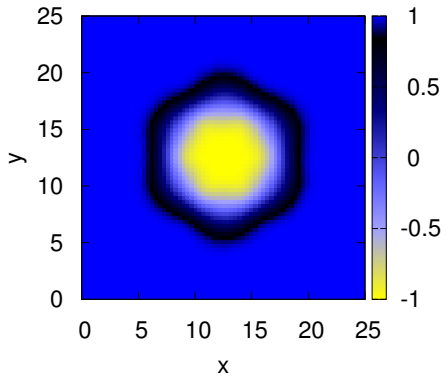
図 39: チャージ $N = 2$



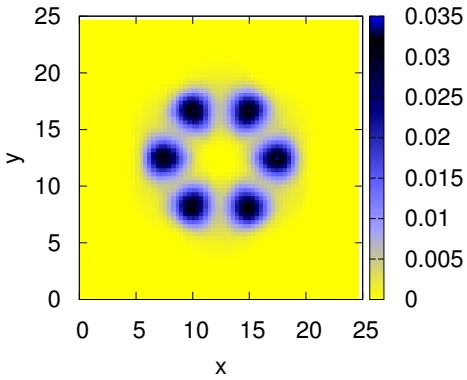
n_1



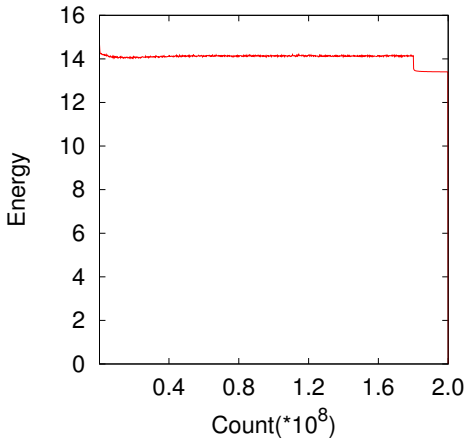
n_2



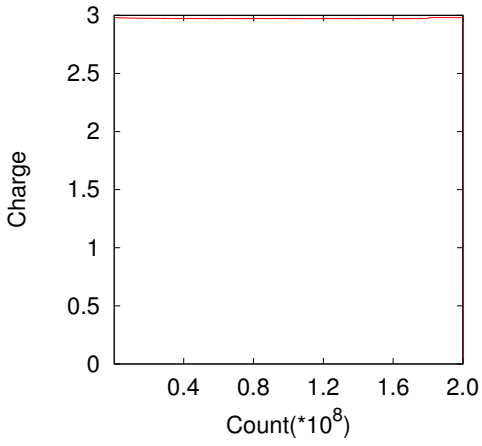
n_3



エネルギー密度

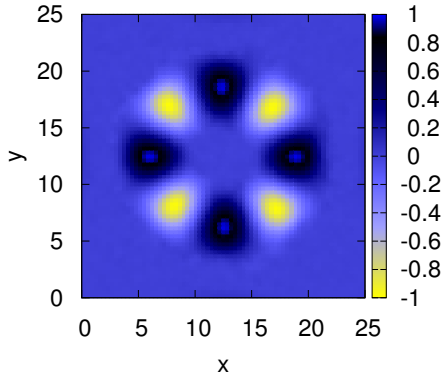


エネルギーの推移

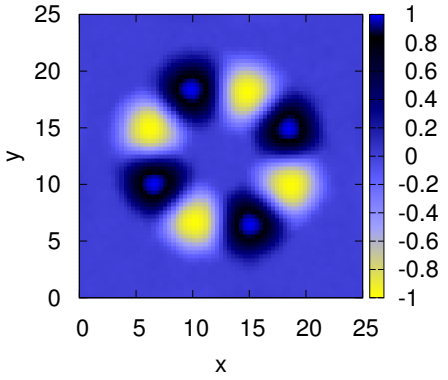


チャージの推移

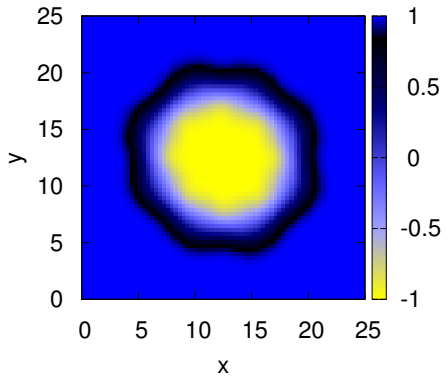
図 40: チャージ $N = 3$



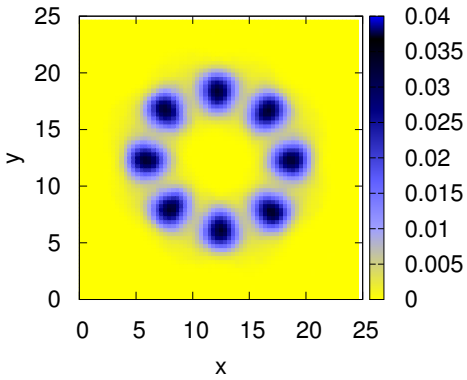
n_1



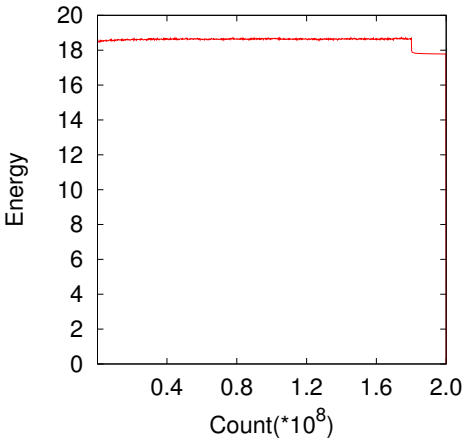
n_2



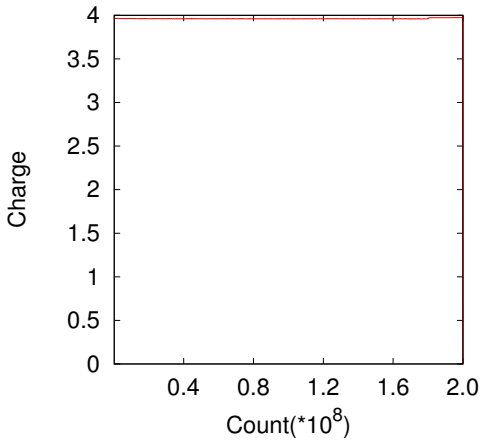
n_3



エネルギー密度

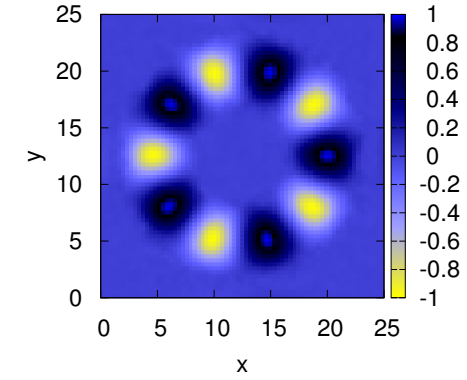


エネルギーの推移

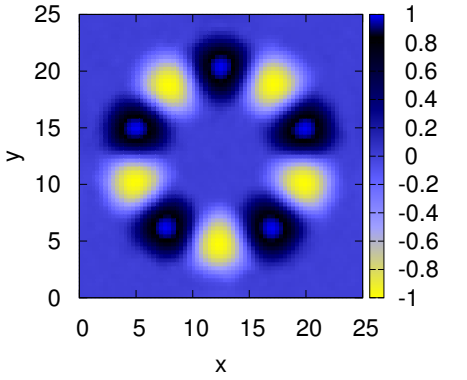


チャージの推移

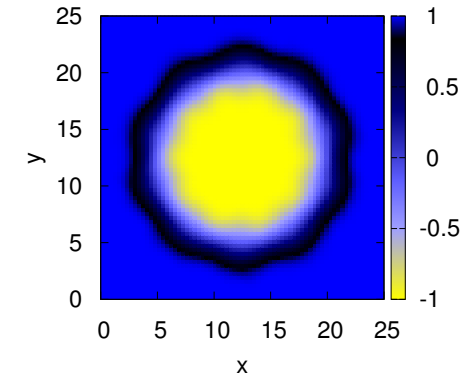
図 41: チャージ $N = 4$



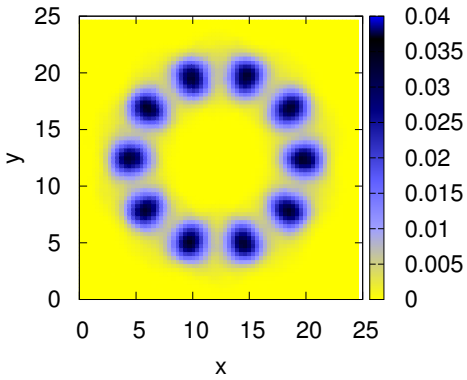
n_1



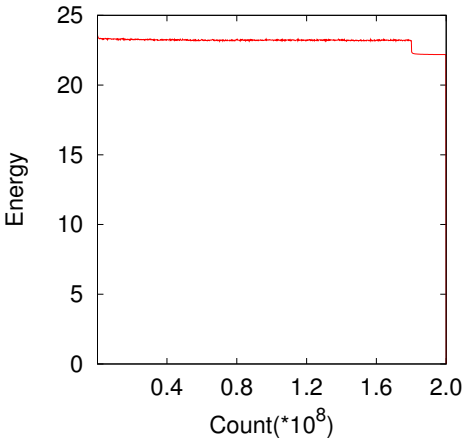
n_2



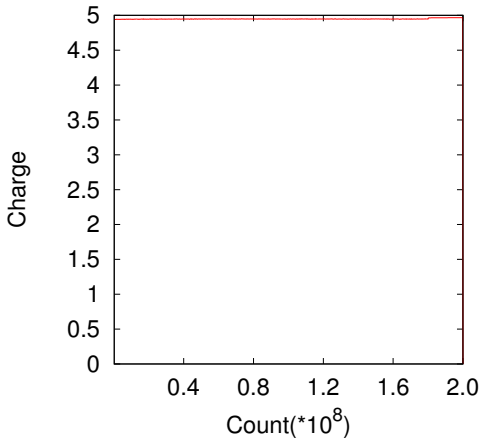
n_3



エネルギー密度



エネルギーの推移



チャージの推移

図 42: チャージ $N = 5$

得られた図では old-Baby ポテンシャルや new-baby ポテンシャルの解析で得られたものとはまた違う非常に興味深い構造をしている。全てのチャージに共通して言えることだが、数値解の形状が Half-Skyrmion となっていることがわかる。トポロジカルチャージ $N = 1$ の場合に着目してみよう。old-Baby,new-Baby ポテンシャルの場合エネルギー密度の図では 1 つのピークしか立っていなかったが、n_1 ポテンシャルを用いるとチャージが 1 でも 2 つのピークを持つ解を得ることができる。これはそれぞれのピークが $\frac{1}{2}$ のチャージを持つことを意味する。このポテンシャルを用いることで Half-Skyrmion を得ることができる。この構造は異方的超伝導体 Sr_2RuO_4 の実験結果と対応する磁束構造となっている。Baby-Skyrme 模型においては様々なポテンシャルを用いることで解構造、すなわち磁束の構造を変えることができる。実験との対応を考えると Half-Skyrmion 構造を得られるこのポテンシャルは現実の物質 Sr_2RuO_4 の機構を説明することができる可能性があることが示唆される。

このような Half-Skyrmion 構造を実現するために n_1 ポテンシャルを用いればよいことがわかった。では Baby-Skyrme 模型の解構造はポテンシャルによってのみ決まるのであろうか？実はポテンシャルだけで解構造が決まるわけではない。ポテンシャルは解構造を決定するのに非常に大きな役割を果たすが本研究では計算領域を超伝導体の表面であると考えるため境界条件も解構造の決定に非常に大きな役割を果たす。ここまではポテンシャルについての考察を行ったが次節では周期境界条件を模型に課した場合の数値解を導出し、その考察を行う。

5.4 周期的境界条件を付加した Baby-Skyrme 模型

この節では Baby-Skyrme 模型を用いて実際の実験で発見されている超伝導の磁束格子構造を再現することを試みる。前節では模型に付加されるポテンシャルが数値解の構造を変えることを確認し、ポテンシャルの違いを超伝導物質の物性に起因するものと考えた。ここでは、境界条件の寄与が格子構造にどのような影響を与えるかを見ていく。[35] エネルギー汎関数は (69) 式で与えられる。

ここで、$M = 1.0, \kappa = 0.03, \mu^2 = 0.01$ とし、old-Baby ポテンシャルを用いた。

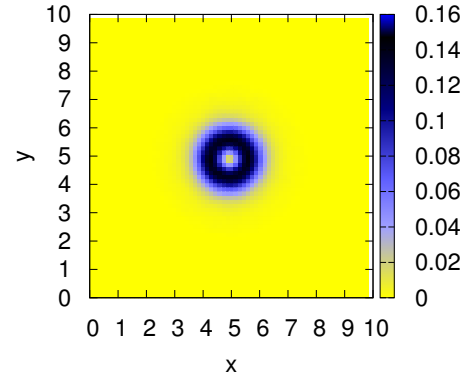
このエネルギー汎関数に以下の図で示されるような周期境界条件を付加する。

周期境界条件を定めると、計算領域は超伝導体の単位格子を意味することになる。ここでは、単位格子の形状は正方形。1 辺の長さを L とすると面積 $S = L^2$ となる。このとき、単位格子内部に”baby-skyrme vortex がどの程度詰まっているか”を表現する指標として”密度”ρ を定義する。トポロジカルチャージを N とすると、ρ は

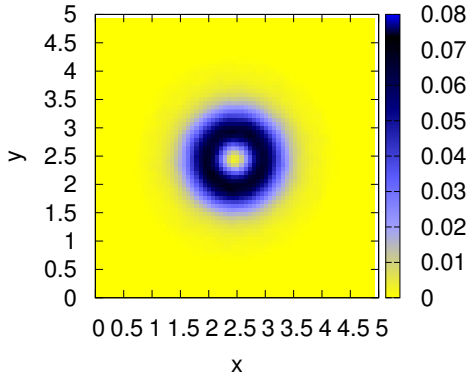
$$\rho = \frac{N}{S} \tag{74}$$

周期境界条件を課した Baby-Skyrme 模型において様々な密度に対して数値解を求めた。メッシュ数は 70×70 である。各密度毎のエネルギー密度の分布図を得た。

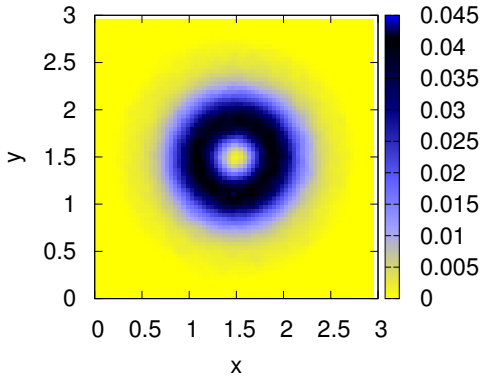
図 43: 周期境界条件の概念図



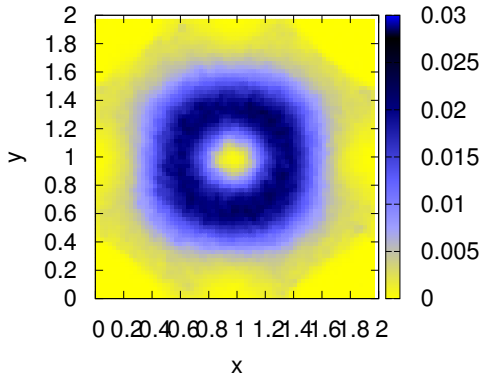
密度 $\rho = 0.02$



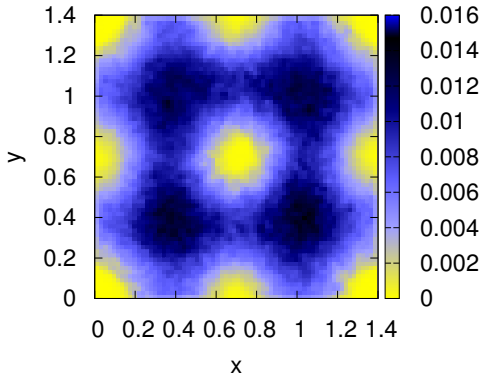
密度 $\rho = 0.08$



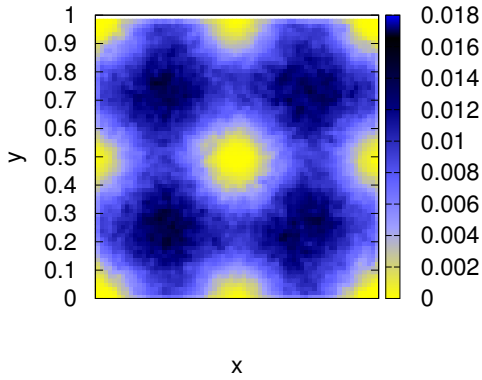
密度 $\rho = 0.22$



密度 $\rho = 0.50$



密度 $\rho = 1.00$



密度 $\rho = 2.00$

図 44: チャージ $N = 2$, 周期境界条件下でのエネルギー密度

上図のように、密度 ρ の値が小さいときは計算領域に対して解のサイズが小さいので境界条件の効果が効いていない。密度 ρ が大きくなっていくと徐々に周期境界条件の寄与が数値解の構造変化に効いてくることがわかる。密度 $\rho = 1.0$ 以上のエネルギー密度は明らかにそれまでの Baby-Skyrme 模型の解構造とは異なり、Half-Skyrmion 構造となっている。前節ではポテンシャルの効果で Half-Skyrmion を得ることに成功した。ここでは、ポテンシャルではなく境界条件の効果により Half-Skyrmion を得ることができた。ここでも Baby-Skyrme 模型で異方的超伝導体 Sr_2RuO_4 の実験で生じた半整数の磁束量子構造を得ることに成功した。また、実際に密度 $\rho = 2.0$ の数値解を周期的につなげた図と実際の実験[3]において観測された磁束量子構造の比較図を以下に示す。

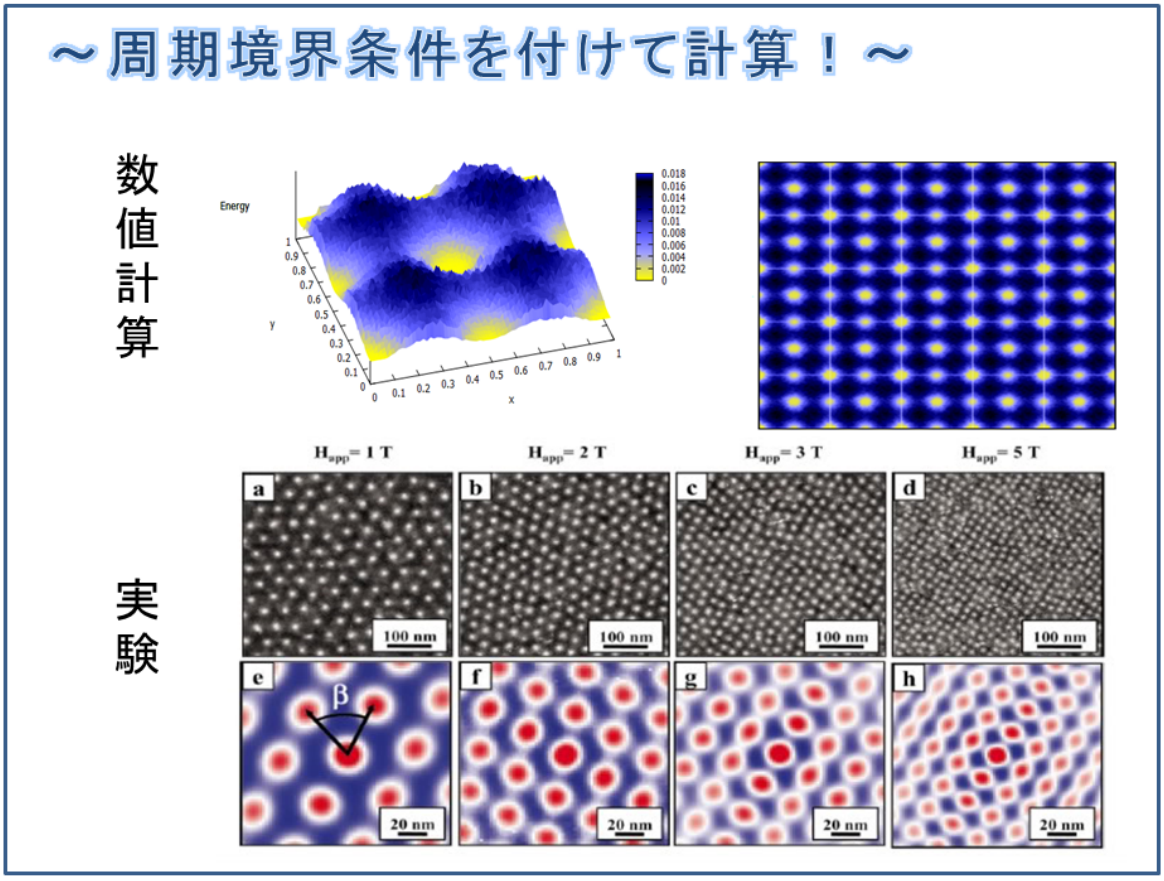


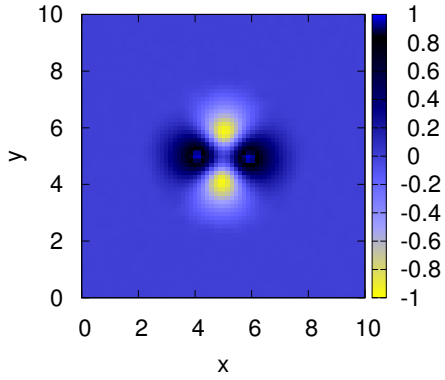
図 45: 数値計算結果と実験結果の比較

[3]http://www.thunderbolts.info/forum/phpBB3/viewtopic.php?p=4783#p4794 にて掲載

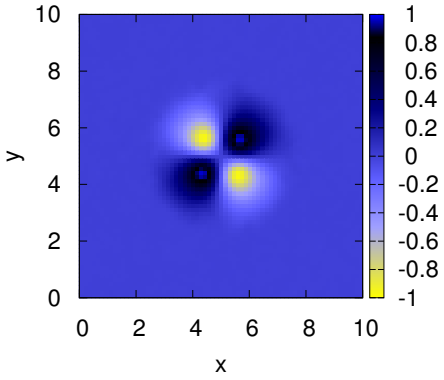
上段のように、チャージ $N = 2$, 密度 $\rho = 2.0$ の場合の数値解をつなげていくとたしかに周期的な構造が再現されていることがわかる。このような数値解に対応する現象の一例として下段に実際に超伝導体に磁場をかけていったときの磁束格子構造を掲載した。左側の図から磁場を 1T(テスラ),2T,3T,5T とかけていったときの図となっている。磁場が弱いときの格子構造は六角格子構造となっており、磁場を強めていくと三角格子、四角格子と格子構造が変化していくことがわかる。Baby-Skyrme 模型における数値計算結果は四角格子の実験結果と対応している。では六角格子や三角格子は作れないのだろうか？参考文献 [30] では単位格子の形状を変化させることで六角格子の再現が行われている。より現象へ近づくためには外部磁場により磁束格子の数が増加するという機構が必要になる。トポロジカルソリトン模型ではトポロジカルチャージ、つまり磁束格子の個数が保存される状況となっている。そのため、外部磁場の応答性が模型には組み込まれていない。

そこで、本研究では Baby-Skyrme 模型に外部磁場に対する応答を組み込むことに挑戦した。詳細は次章で述べていく。

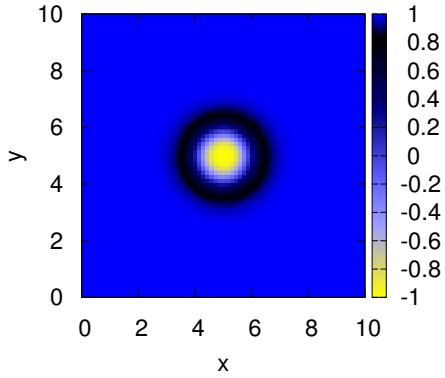
その前に周期境界条件を課したそれぞれの密度の数値解を以下に掲載する。



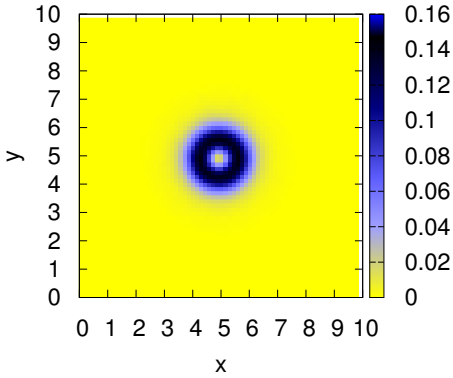
n_1



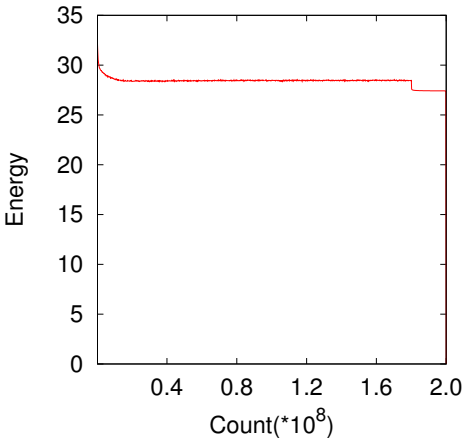
n_2



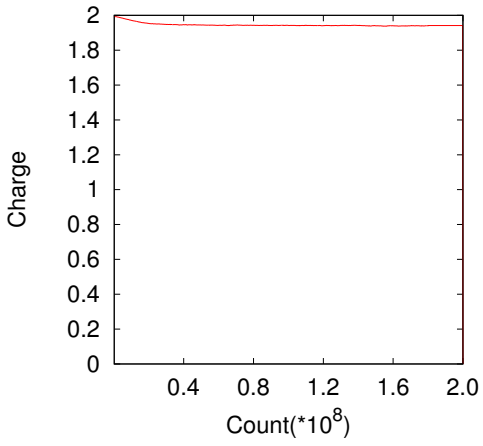
n_3



エネルギー密度

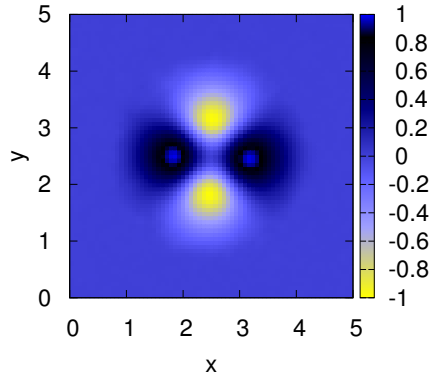


エネルギーの推移

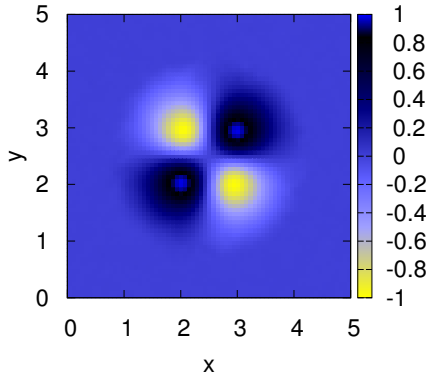


チャージの推移

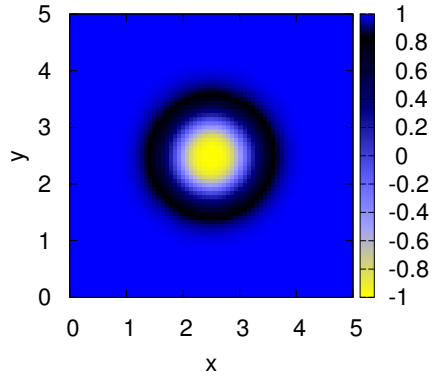
図 46: 密度 $\rho = 0.02$



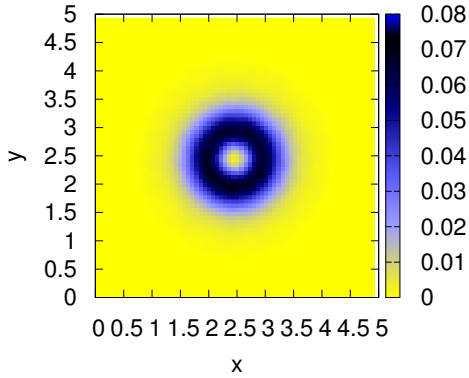
n_1



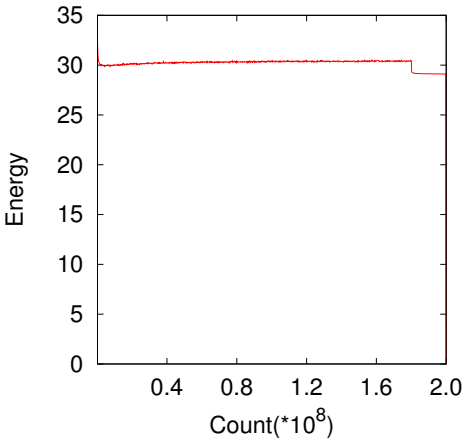
n_2



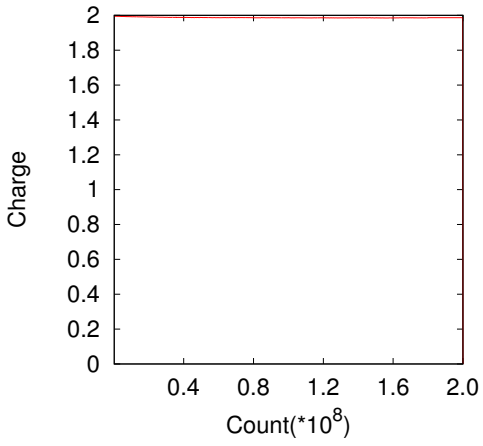
n_3



エネルギー密度

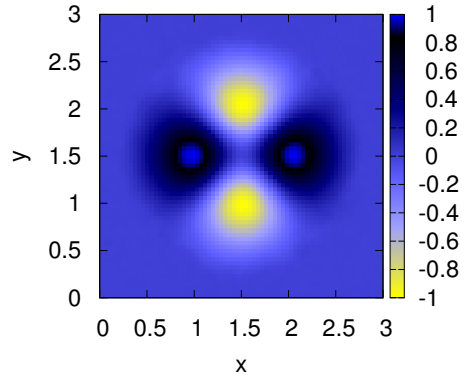


エネルギーの推移

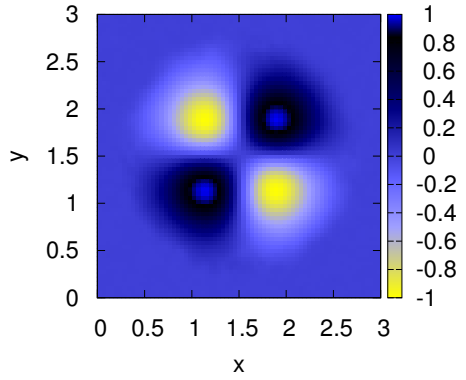


チャージの推移

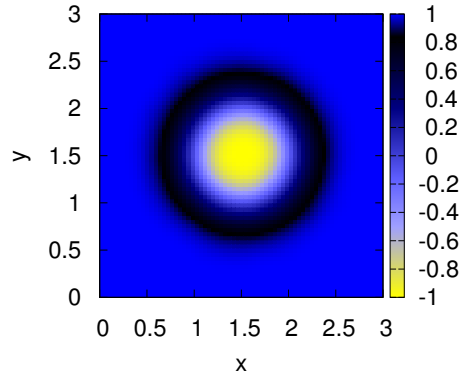
図 47: 密度 $\rho = 0.08$



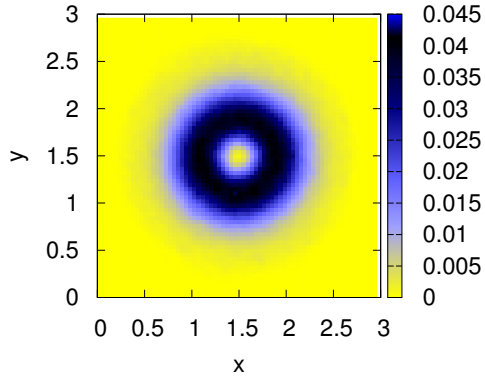
n_1



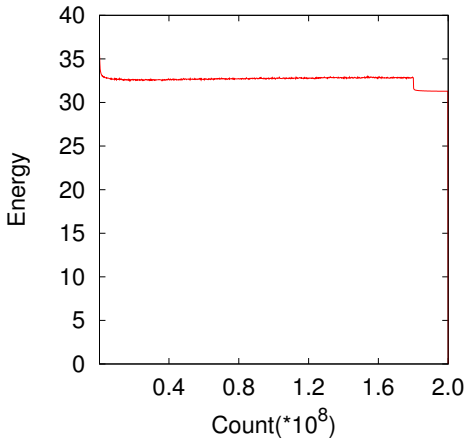
n_2



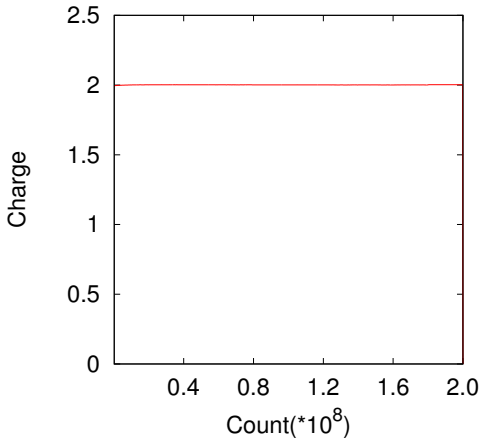
n_3



エネルギー密度

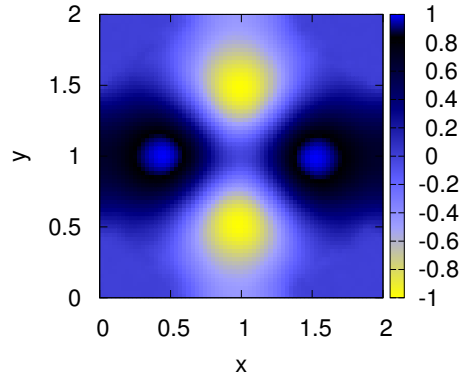


エネルギーの推移

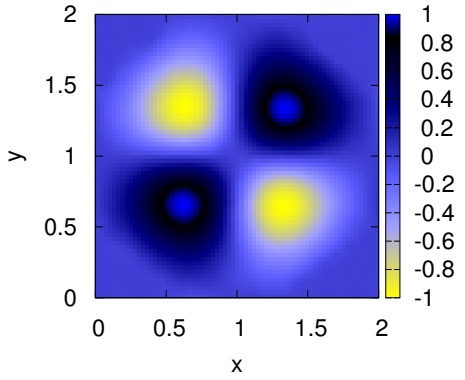


チャージの推移

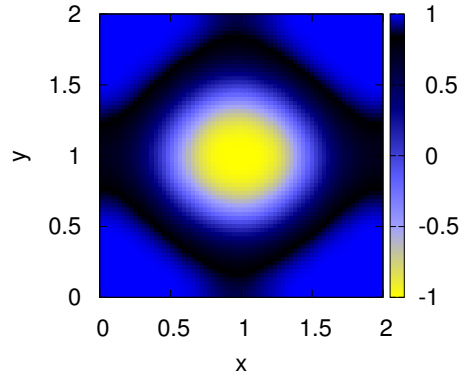
図 48: 密度 $\rho = 0.22$



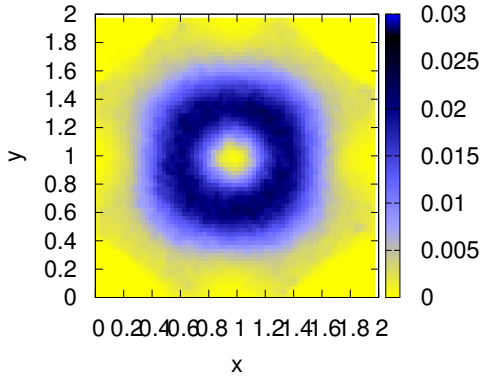
n_1



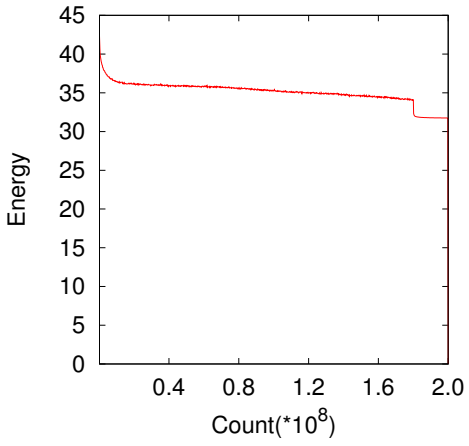
n_2



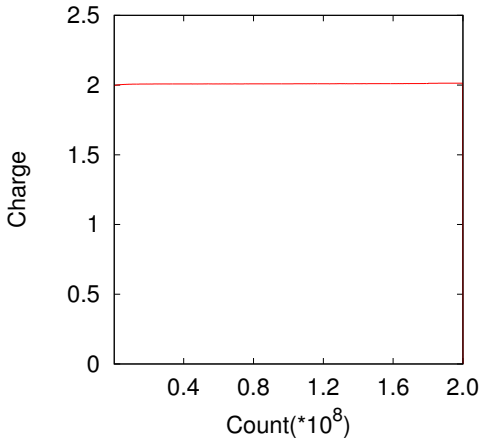
n_3



エネルギー密度

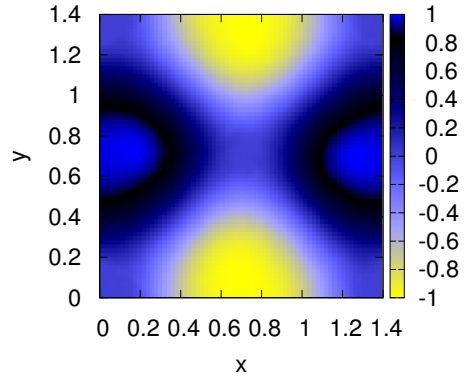


エネルギーの推移

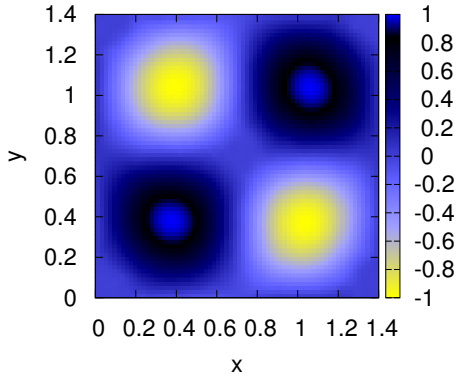


チャージの推移

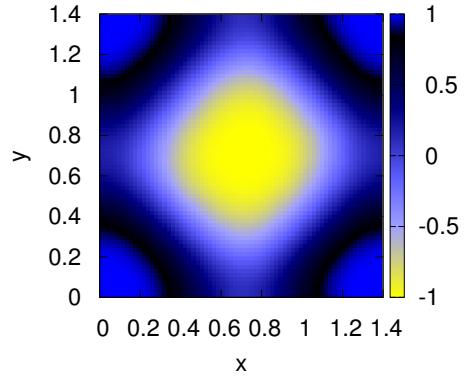
図 49: 密度 $\rho = 0.50$



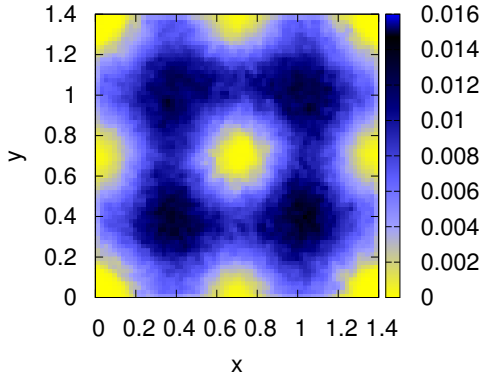
n_1



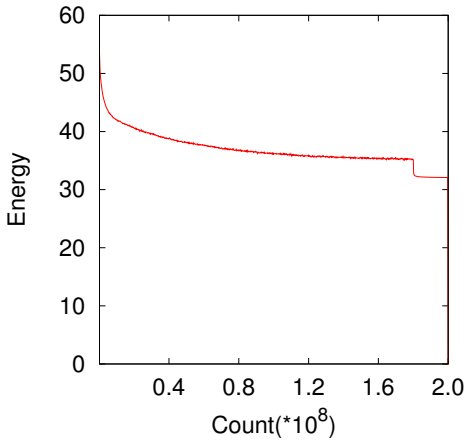
n_2



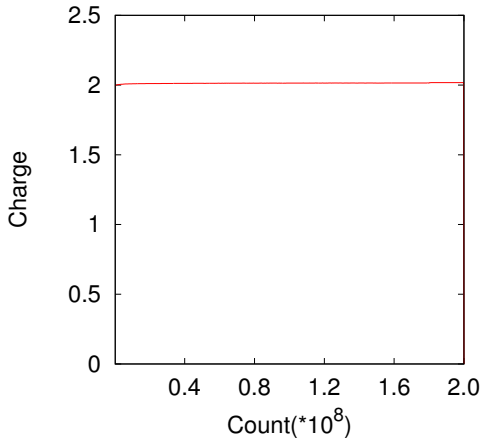
n_3



エネルギー密度

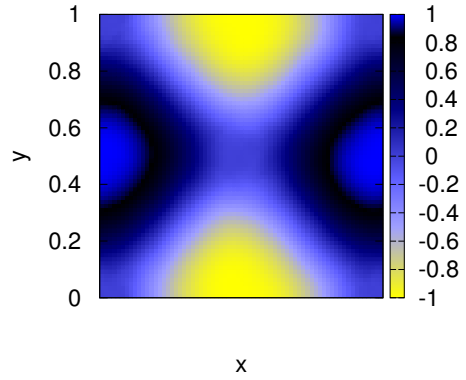


エネルギーの推移

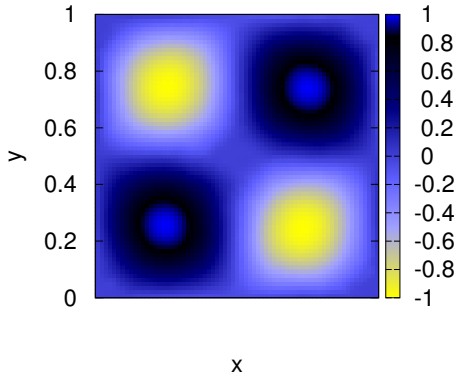


チャージの推移

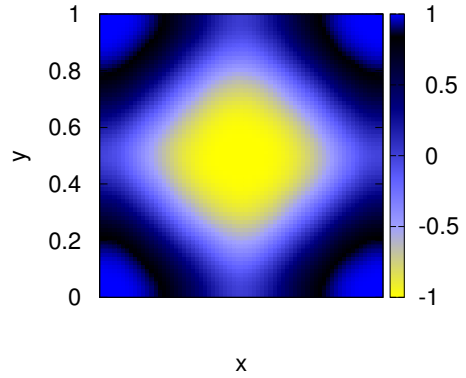
図 50: 密度 $\rho = 1.00$



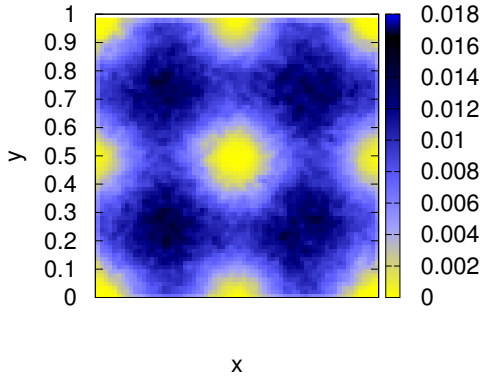
n_1



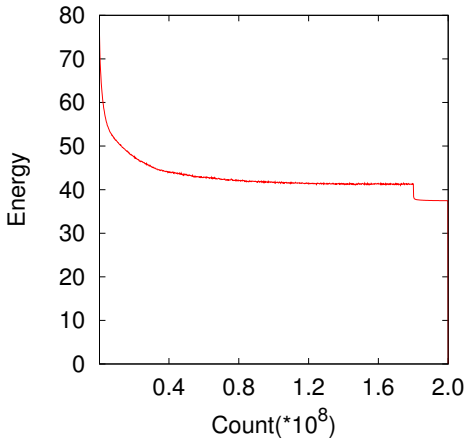
n_2



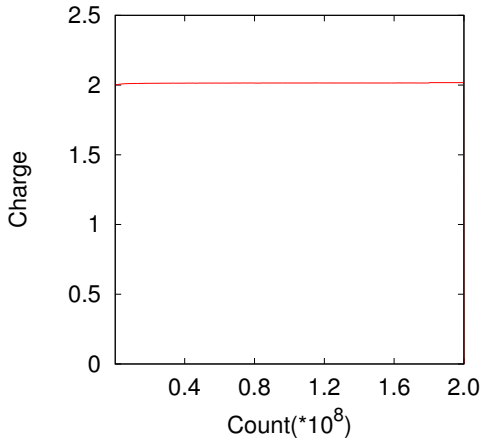
n_3



エネルギー密度



エネルギーの推移



チャージの推移

図 51: 密度 $\rho = 2.00$

6 新たな Baby-Skyrme 模型の構築と数値解析

現象として、超伝導体に外部磁場をかけることで磁束格子の数が増えていくことを第 4 章で解説した。また、Baby-Skyrme 模型では磁束格子の数はトポロジカルチャージに対応している。ソリトン解はチャージが保存することで安定に存在することが可能なため、通常の模型の解析ではチャージの数が一定に保たれなければならない。ところが、我々は外部磁場をかけることで磁束格子の生成機構を再現し、チャージが増加するという従来の模型にはない性質を有する新たな超伝導模型の構築に成功した。

この章では Baby-Skyrme 模型に外部磁場との相互作用と超伝導体のサイズ効果を付加し、新たな模型を得る。その模型の構築法の物理的説明を述べ、いくつかの状況設定についての数値計算を行う。その数値計算結果と現象との対応関係、課題点、そして応用法を述べる。

6.1 模型の構築

Baby-Skyrme 模型のラグランジアン密度は (63) 式、エネルギー汎関数は (69) 式で記述されていた。この模型においてトポロジカルチャージが増える機構を再現するような項をどのようにして作ればよいのだろうか？

ここで、"E/N を最小化する" ことと "n_3 の構造とエネルギー密度の構造は対応する" ことに着目する。まず、"E/N を最小化する" ことについて述べていく。前章で我々は SA 法により E/N を最小化してきた。もし E のみをコスト関数にとった場合、チャージがすぐさま減少し、得られる解が真空解になってしまうからだ。物理的には E を最小化させるものが自然界が選ぶ解構造であると考えてきた。にも関わらず、E/N をコストとして数値計算を行うことでソリトン解を得ることができた。単なる数値計算上の都合として考えることもできるだろう。しかし、これは実は本質的なことを示しているのではないだろうか？この模型において自然界が選ぶ解は "E を最小化する解" ではなく "E/N を最小化させる解" なのではないだろうか？もし、E を最小化させる解が自然界に存在する解であるとするならば、チャージ N が増加することなど自然界に起こるはずはない。なぜなら、E は N によって下限が抑えられ、N が増加すれば E も増加するからである。このことは 4 章と 5 章で計算した解のエネルギーの推移を確認すれば一目瞭然の事実である。しかし、自然界において、実際の超伝導体において、**磁束格子の増加は起こる**のである。我々はこのような事実から、E/N すなわち単位磁束あたりのエネルギーこそが最小化させるべき物理量であると考える。このとき、我々は 2 種類の方法によって E/N を減少させることができる。一つ目が E を下げること、そして 2 つ目が N **を上げる**ことである。(63) 式の模型は、前者の方法である E を下げることにより E/N を最小化させるという構造になっている。この場合 N は保存され、通常のトポロジカルソリトン模型と考えてよいことがわかる。これに対し、我々は 2 つ目の方法である N を上げるという方向性から E/N を最小化させる模型を作ることで磁束格子の増加を再現する模型を構築することができるのではないかと考えた。

"n_3 の構造とエネルギー密度の構造は対応する" ことと超伝導体に外部磁場をかけた場合の現象論との対応関係を用いて Baby-Skyrme 模型に新たな項を付加する。5 章の様々な数値計算結果から、高エネルギー密度で囲まれた領域に $n_3 = -1$ の領域が存在していることがわかる。

Baby-Skyrme 模型では、その領域に渦糸及び外部磁場がその領域を貫通するとされている。また、外部磁場を強くしていくほど磁束格子の数が増える。したがって、我々は以下のような性質を模型に付与させる項を欲する。

1. 外部磁場がかかっている領域で $n_3 = -1$
2. 外部磁場を強めていくと N が増加する

このような性質を満たす項を try&error により発見した。それが以下の項である。

$$\gamma H(1 + n_3), \tag{75}$$

ここで、γ は次元を持ったパラメータ、H は外部磁場を表している。この項の意味するところを考えていく。このような項が E の中に含まれているときに E を下げようとすると外部磁場 H の値が大きい場所では $n_3 \to -1$ となることがわかる。$n_3 = -1$ の領域が増加することで n_1, n_2 も変化していくことになり、結果的にチャージが増加するのだ。もう少し細かく状況を説明する。まず、境界では真空条件により $n_3 = 1$ となっている。新たに加えた外部磁場の項の効果により、各点で $n_3 = -1$ に変化していき、解の形状が境界付近に広がっていく。$n_3 = 1$ と $n_3 = -1$ が非常に近い領域に共存することになる。この $\vec{n}$ 場の急激な変化はドメインウォールを生み出す。ドメインウォールはトポロジカル的欠損となり、Skyrmion を生成する。以上のようなメカニズムで Skyrmion が外部磁場により生成される。

我々は以下のラグランジアン密度で記述される新たな Baby-Skyrme 模型を得る。

$$\begin{aligned}\mathcal{L} = &-\mu^2(1 - n_3) + \frac{M^2}{2}\partial_\mu \vec{n} \cdot \partial^\mu \vec{n} \\ &- \gamma H(1 + n_3) - \frac{\kappa^2}{2}(\partial_\mu \vec{n} \times \partial_\nu \vec{n})^2\end{aligned} \tag{76}$$

したがって、この模型のエネルギー汎関数は以下の式

$$\begin{aligned}E = &\frac{1}{2}\iint_S dxdy\Big[2\mu^2(1 - n_3) + M^2\big\{(\partial_x \vec{n})^2 + (\partial_y \vec{n})^2\big\} \\ &+2\gamma H(1 + n_3) + \kappa^2(\partial_x \vec{n} \times \partial_y \vec{n})^2\Big].\end{aligned} \tag{77}$$

となる。(2 次元直交座標形式)

以下の節では実際に我々が得た新たな模型の解を SA 法により求めていく。我々の作った模型がどのように機能するのか、物理的な解釈は正しかったのかを確認していく。

6.2 SA 法による数値解析と考察

この節では、前節で構築した新たな模型についての数値解析を行い、模型の性質について議論する。用いるエネルギー汎関数は (77) 式でコストは E/N である。パラメータについては、$M = \kappa = 1.0, \mu = \frac{1}{3}, \gamma = 1.0$ と固定して計算を行った。メッシュ数は 72×72 である。外部磁

場として一様磁場を選択した。H の値を 0.05 刻みで 1.00 まで増加させていき、外部磁場に対してチャージやエネルギーなどがどのように応答するかを確かめた。また、サイズ効果を検証するために 1 辺の長さ L を 3 つの場合、$L = 4, L = 7, L = 10$ の場合について数値計算を行った。その数値計算結果をまとめた図を以下に示す。

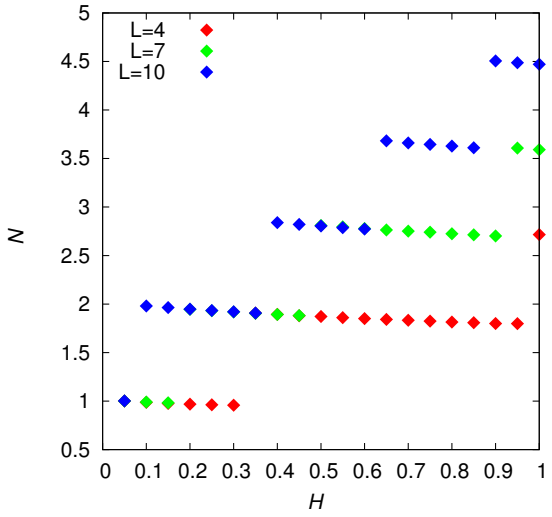


図 52: 外部磁場の増加に対するチャージの推移

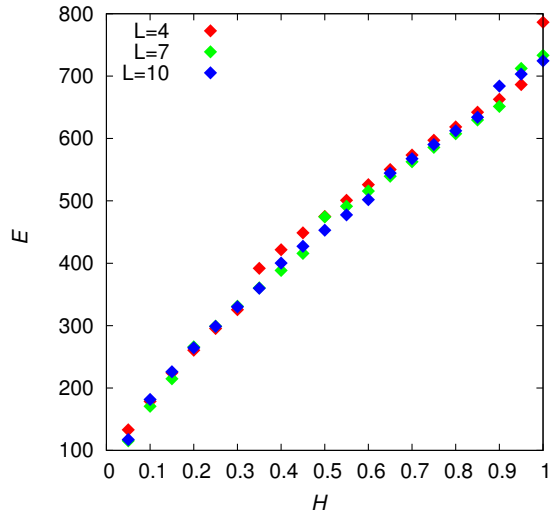
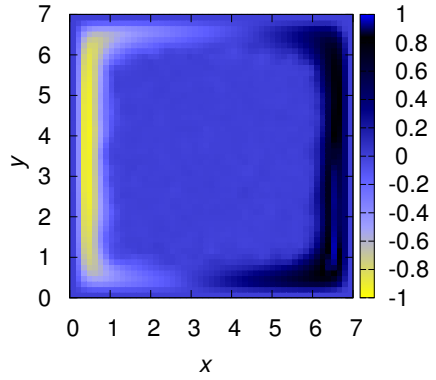


図 53: 外部磁場の増加に対するエネルギーの推移

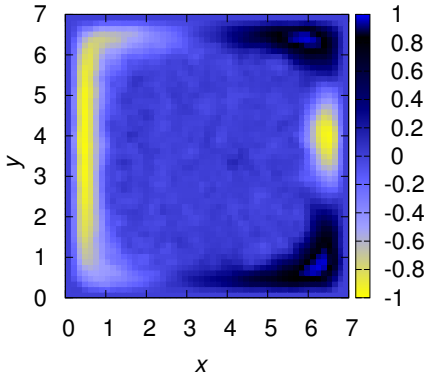
まず、チャージの推移図について解説する。外部磁場 H の値を増加させていくと 1 つずつ磁束格子が超伝導体中に発生していることがわかる。また、L の値が大きければ大きいほど N が増加しやすいことがわかる。この 2 点から現象論における磁束格子の増加が数値計算において再現されていることが示唆される。$L = 4$ の場合は N が 1 から 3 まで増加し、$L = 7$ の場合は N が 1 から 4 まで増加し、そして $L = 10$ の場合は N が 1 から 5 まで増加しているが、数値計算の精度がやや低いためチャージ N の欠損が起こってしまっている。メッシュの数をもっと増やして計算すればこの欠損は取り除かれるはずである。また、本来 L を大きくして行う計算ではより多くのメッシュを必要とすることにも留意しておくべきである。$L = 4$ の領域でとるメッシュ数 72 と $L = 10$ の領域でとるメッシュ数 72 では情報量に違いがあるからである。$L = 4$ の場合では長さ 1 の線分に対して $72/4 = 18$ 個分の格子を配置することができるが $L = 10$ の場合では長さ 1 の線分に対して $72/10 = 7.2$ 個分の格子しか配置できず明らかに L が大きいほうが情報量に乏しくなってしまう。だが、我々が作り出した模型の本質を確認するという目的においてはこの精度でも十分である。

次に、エネルギーの推移図について解説する。外部磁場をかけていくと全体としてのエネルギーは増加していくことがわかる。エネルギーの下限はチャージで抑えられるのでチャージが増加するときにもエネルギーが増加する。また、L が大きいほどチャージの増加前と増加後のギャップは小さい。これが超伝導体のサイズ効果となる。あるセクターから異なるセクターへ移るために必要なエネルギーは超伝導体のサイズにより異なる。ギャップが大きければ大きいほどチャージを増加させるために強い外部磁場を印加する必要があることがわかる。新たに作った模型では超伝導体のサイズ効果が含まれており、超伝導体の磁束格子構造を記述するのによりふさわしい模型となっていることがわかる。

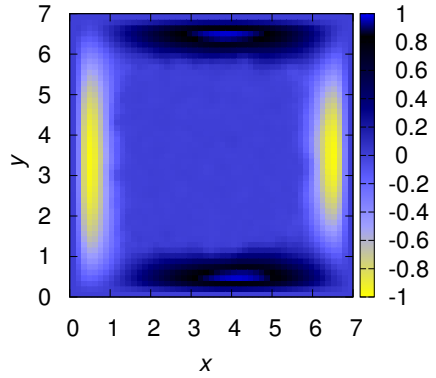
さて、次に $L = 7$ についての n_1 の変化と (n_1, n_2) のベクトル場を以下に示す。



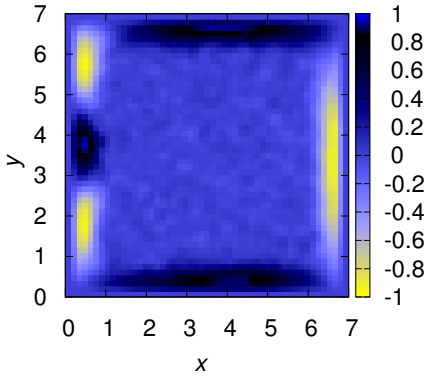
$N = 1$



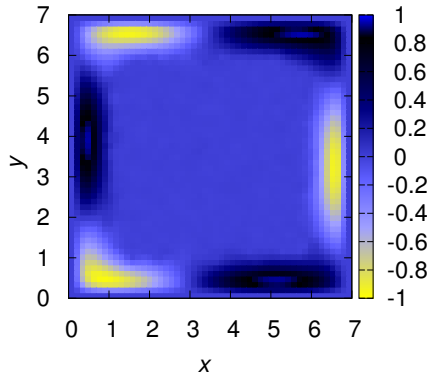
$N = 1 \to 2$



$N = 2$



$N = 2 \to 3$



$N = 3$

図 54: $L = 7$ における外部磁場印加による n_1 の変化

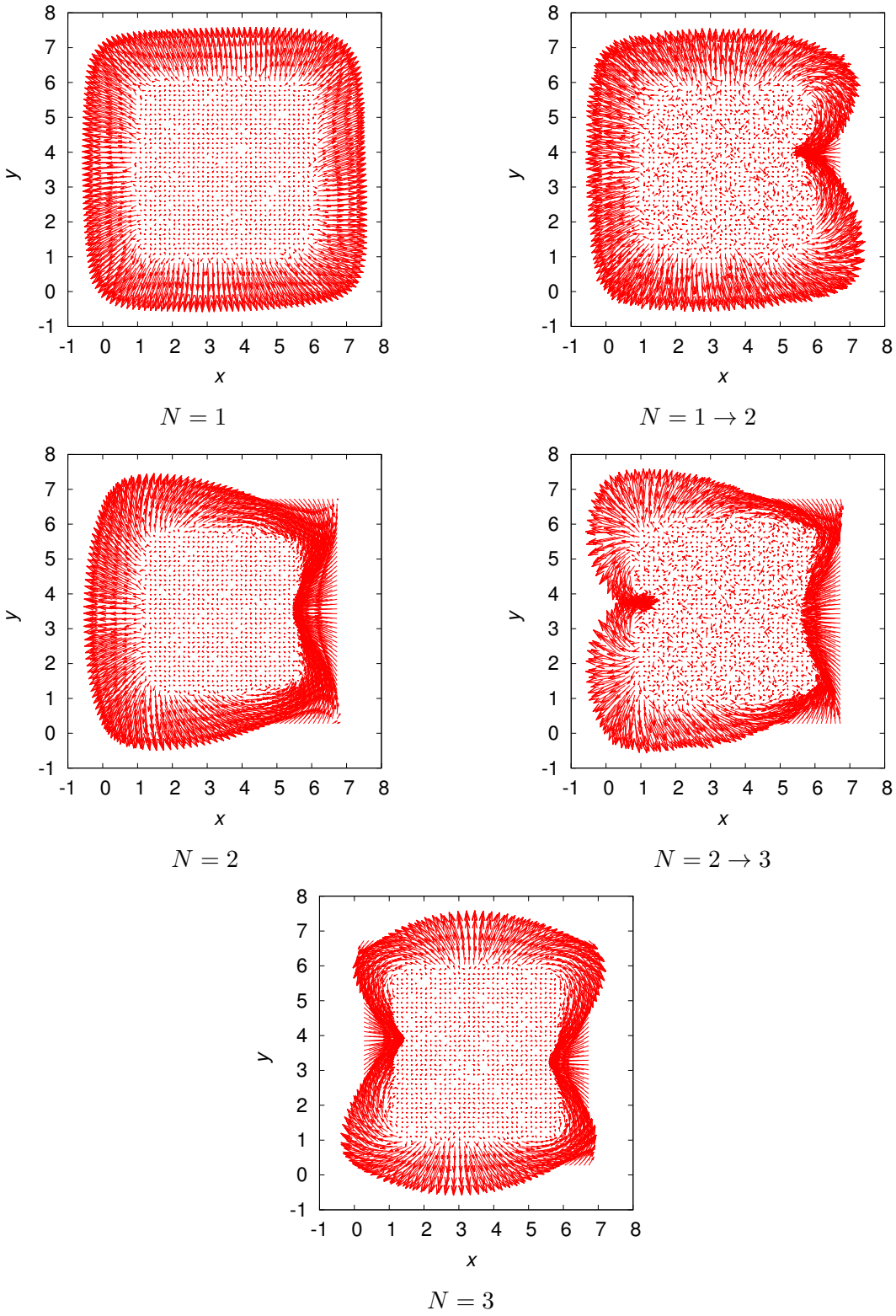


図 55: $L = 7$ における外部磁場印加による (n_1, n_2) ベクトル場の変化.

上図より、外部磁場を印加することで n_1 は境界付近に追いやられ、ドメインウォールを形成する。外部磁場の強度の増加に従って超伝導体の端に発生したドメインウォールからチャージが発生していることがわかる。ベクトル場からも巻きが増加してチャージが増えていることが確認できる。従来の Baby-Skyrme 模型ではポテンシャルにより n 場の形状が変化し、解構造が決まる。しかし、この新たな模型では一様磁場を印加している間の解構造はドメインウォールとなり、一様磁場をなくすと再び従来の Baby-Skyrme 模型の解として振る舞う。一様磁場を印加している間は超伝導体の領域全域を占める Gigantic Skyrmion 構造をとる。Half Skyrmion 構造などはこの模型のままでは得られていない。しかし、外部磁場の印加によりチャージが増加する機構をサイズ効果を含んだ形で Baby-Skyrme 模型に組み込むことに成功した。

7 まとめ、今後の展望

本論文では、数値計算法の基礎を解説することから始まり、種々のトポロジカルソリトン模型の数値解析と解の考察を行った。用いた模型は Sine-Gordon 模型、Abelian-Higgs 模型、Baby-Skyrme 模型の 3 つである。Sine-Gordon 模型では、SOR 法と SA 法の活用法についての解説を主に行った。Abelian-Higgs 模型では超伝導体への理解を深めるとともに、模型の解が超伝導体における磁束構造を再現することを示した。Baby-Skyrme 模型では、異方的超伝導体の存在を念頭に、通常の磁束量子とは挙動が異なる磁束構造を再現、考察した。

Baby-Skyrme 模型では、外部磁場の効果と超伝導体のサイズ効果が取り入れられていなかったため現象論との対応関係を念頭に置き、新たな模型の構築を行った。その結果、外部磁場により生成されるドメインウォールを介して Skyrmion が励起される新たな Baby-Skyrme 模型を構築することに成功した。この模型では超伝導体の面積や磁束格子の大きさに依存して外部磁場により励起される磁束格子の数に制限を与えることができる。このような模型を提唱したのは本研究が始めてである。

ただし、新たな模型では生成される Skyrmion は集まって存在する Gigantic Skyrmion 構造をとる。そのため、Sr_2RuO_4 などで実験的に発見されている Half-Skyrmion 構造を得ることはまだできていない。この構造を得るためにポテンシャルを変更したり、境界条件の効果を付加するなどしてより現象論と一致する模型を作ることが望まれる。また、現段階では外部磁場により生成される磁束そのものが担う磁場がこの模型には含まれていない。そのため、この模型をゲージ化することも非常に興味深い研究対象となるだろう。

筆者は本学を卒業後、他分野での大学院で引き続き研究活動を続けていく所存である。そのため、この研究対象をこれ以上掘り下げていくことは叶わない。本論文を読んで興味を持った読者が本研究をより発展させていってくれれば望外の喜びである。

8 謝辞

私は学部 1 年生から 4 年生までの非常に長い期間にわたり、澤渡先生の下で様々なことを学ばせていただきました。公私共に幾度となくお世話になった先生には心より深く感謝いたします。また、昨年ご卒業された早坂先輩と M1 の吉井先輩には夜遅くまで何度も議論に付き合っていただき研究のアイデアを得るきっかけとなりました。そのほか M1 の先輩方および研究室の同期とは日頃から競い合い切磋琢磨することで非常に有意義な時間を過ごせました。研究室のメンバーの皆様、本当にありがとうございました。

参考文献

[1] R.Rajaraman, Solitons and Instantons, North-Holand personal library

[2] Erick J. Weinberg, Classical Solutions in Quantum Field Theory, Cambridge Monographs on mathematical physics

[3] N.Manton and P Sutcliffe, Topological Solitons, Cambridge Monographs on mathematical physics

[4] 皆本晃弥　C 言語による数値計算入門—解法・アルゴリズム・プログラム　サイエンス社

[5] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery　丹慶勝市・奥村晴彦・佐藤俊朗・小林誠訳　Numerical Recipes in C [日本語版]　技術評論社.

[6] 牛島省　数値計算のための Fortran90/95 入門　森北出版株式会社

[7] 近藤慶一　ゲージ場の量子論入門　サイエンス社

[8] H. B. Nielsen and P. Olesen, Nucl. Phys. B **61**, 45 (1973).

[9] 丹羽雅昭　超伝導の基礎 第 3 版　東京電機大学出版局

[10] 恒藤敏彦　超伝導・超流動 (現代物理学叢書)　岩波書店

[11] 家泰弘　超伝導 (朝倉物性物理シリーズ)　朝倉書店

[12] H. F. Hess, R. B. Robinson, R. C. Dynes, J. M. Valles and J. V. Waszczak, Phys. Rev. Lett. **62**, 214 (1989).

[13] 神田晶申　微小超伝導体における新しい渦の観測　筑波大学物理学セミナー 050601

[14] Jang, J., et al., Science 331.6014 (2011): 186-188.

[15] T. H. R. Skyrme, Nucl. Phys. **31**, 556 (1962).

[16] B. M. A. G. Piette, B. J. Schroers and W. J. Zakrzewski, Nucl. Phys. B **439**, 205 (1995) [hep-ph/9410256].

[17] X. Z. Yu et al. Nature (London) **465**, 901 (2010).

[18] M. Ezawa, Phys. Rev. Lett. **105**, 197202 (2010) [arXiv:1007.4048 [cond-mat.str-el]].

[19] S. Mühlbauer, et al., Science **323**, 915 (2009).

[20] U. K. Rossler, A. A. Leonov, and A. N. Bogdanov, J. Phys. Chem. Solids **200**, 022029 (2010).

[21] U. Rossler, A. A. Leonov and A. N. Bogdanov, arXiv:1009.4849 [cond-mat.str-el].

[22] T. Morinari, arXiv:1007.4047 [cond-mat.str-el].

[23] Y. Maeno., et al., arXiv preprint arXiv:1112.1620 (2011).

[24] Vargas-Paredes, Alfredo A., et al., Journal of Superconductivity and Novel Magnetism (2013): 1-4.

[25] J. Garaud and E. Babaev, "Skyrmionic state and stable half-quantum vortices in chiral p-wave superconductors," Phys. Rev. B **86**, 060514 (2012).

[26] J. Garaud, J. Carlstrom, E. Babaev and M. Speight, Phys. Rev. B **87**, 014507 (2013) [arXiv:1211.4342 [cond-mat.supr-con]].

[27] J. Garaud, K. A. H. Sellin, J. Jykk and E. Babaev, arXiv:1307.3211.

[28] M. Hale, O. Schwindt and T. Weidig, Phys. Rev. E **62**, 4333 (2000) [hep-th/0002058].

[29] J. Gladikowski, B. M. A. G. Piette and B. J. Schroers, Phys. Rev. D **53**, 844 (1996) [hep-th/9506099].

[30] I. Hen and M. Karliner, Nonlinearity **21**, 399 (2008) [arXiv:0710.3939 [hep-th]].

[31] J. Jaykka, M. Speight and P. Sutcliffe, Proc. Roy. Soc. Lond. A **468**, 1085 (2012) [arXiv:1106.1125 [hep-th]].

[32] T. Weidig, Nonlinearity **12**, 1489 (1999) [hep-th/9811238].

[33] M. Kobayashi and M. Nitta, Phys. Rev. D **87**, 125013 (2013) [arXiv:1307.0242 [hep-th]].

[34] P. Jennings and T. Winyard, arXiv:1306.5935 [hep-th].

[35] I. Hen and M. Karliner, Phys. Rev. D **77**, 054009 (2008) [arXiv:0711.2387 [hep-th]].