非線形方程式の数値解法：自己無撞着な方程式を解く

永井佑紀

平成 19 年 4 月 24 日

バルクの s 波のギャップ方程式を自己無撞着に（self-consistent に）数値的に解くことで、超伝導ギャップの温度依存性を求める。このノートは、その際の数値計算手法についてまとめたものである。方法としては、単純な逐次代入法、Steffensen の反復法、Aitken の Δ^2 法、Wyne の ϵ アルゴリズム、Extended Steffensen 法について述べ、それらを用いてギャップ方程式を自己無撞着に解き、計算速度を見てみる。

1 自己無撞着方程式

バルクの s 波のギャップ方程式：

$$\Delta = N(0)V \int_0^{\hbar\omega_c} \frac{\Delta \tanh \frac{1}{2}\beta(\xi^2+\Delta^2)^{1/2}}{(\xi^2+\Delta^2)^{1/2}} \tag{1}$$

は自己無撞着な方程式の例である。つまり、絶対零度近傍や転移点近傍等でのみ近似的に解けるが、それ以外の温度領域ではこの方程式は解析的には解けない。この方程式は

$$G(x) = x \tag{2}$$

という非線形方程式の形をしており、このような方程式を数値的に解く方法がわかればよい。

2 様々な数値計算手法

2.1 逐次代入法

一番単純でわかりやすい数値計算手法の一つは、逐次代入法である。これは、初期値を x_0 とすると、

$$\begin{aligned} x_1 &= G(x_0) \\ x_2 &= G(x_1) \\ x_3 &= G(x_2) \\ &\cdot \quad\quad \cdot \\ &\cdot \quad\quad \cdot \\ &\cdot \quad\quad \cdot \end{aligned}$$

というように計算を行っていく（図. 1）。この方法は、解 x における傾き $f'(x)$ が $|f'(x)| > 1$ を満たす場合には収束しない（図. 2）[1]。また、一般的には収束は遅い。プログラムとして実装するのは簡単だが、実用性の面から言うともっと他の収束の早いアルゴリズムを使ったほうがよいだろう。

[1] 参考文献には収束条件の導出が論じられているが、ここでは述べない。

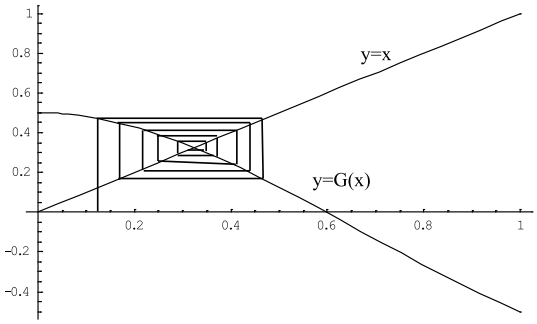

図 1: 逐次近似法における解への収束の流れ。

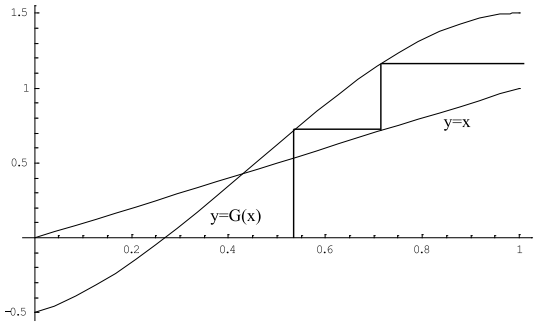

図 2: 逐次近似法における解への収束の流れ。収束しない場合。

2.2 Steffensen の反復法と Aitken の Δ^2 法

2.2.1 Steffensen の反復法

逐次近似法の収束を加速させることを考える。Steffensen の反復法のアルゴリズムは、二点の情報を用いて線形補間を行い、三点目を求めるところに特徴がある。まず、二点 $(x_0, G(x_0) \equiv G_0)$、$(x_1, G(x_1) \equiv G_1)$ が与えられたとする。このとき、この二点を通る一次関数は

$$y = \frac{G_1 - G_0}{x_1 - x_0}(x - x_0) + G_0 \tag{3}$$

と書ける。この一次関数と $y = x$ との交点は

$$x = \frac{(x_1 - x_0)G_0}{x_1 - x_0 - G_1 + G_0} - \frac{(G_1 - G_0)x_0}{x_1 - x_0 - G_1 + G_0} \tag{4}$$

となる。ここで、

$$x_1 = G(x_0) = G_0 \tag{5}$$

$$x_2 = G(x_1) = G_1 \tag{6}$$

であるとすれば、一次関数と $y = x$ との交点は

$$x = x_0 - \frac{(x_1 - x_0)^2}{x_0 - 2x_1 + x_2} \tag{7}$$

と書ける。この点を三点目:

$$x_3 = x_0 - \frac{(x_1 - x_0)^2}{x_0 - 2x_1 + x_2} \tag{8}$$

とするのが Steffensen の反復法の重要なポイントである。つまり、$n+1$ 番目の点 x_{n+1} は x_n を用いて

$$x_{n+1} = x_n - \frac{(G(x_n) - x_n)^2}{x_n - 2G(x_n) + G(G(x_n))} \tag{9}$$

とする。ここでの $n+1$ 番目とは、実際のプログラムにおける反復計算においては 3 の倍数番目であることに注意しなければならない。つまり、計算の手順としては

$$\begin{aligned}
x_1 &= G(x_0) && (10)\\
x_2 &= G(x_1) && (11)\\
x_3 &= x_0 - \frac{(x_1 - x_0)^2}{x_0 - 2x_1 + x_2} && (12)\\
x_4 &= G(x_3) && (13)\\
x_5 &= G(x_5) && (14)\\
x_6 &= x_3 - \frac{(x_4 - x_3)^2}{x_3 - 2x_4 + x_5} && (15)\\
\vdots &= \vdots && (16)\\
& && (17)
\end{aligned}$$

となる。この $x_3, x_6, \cdots, x_{i+2}$ において計算が収束したかどうかを判定する。このアルゴリズムは、逐次代入法で収束しない場合にも収束することが知られている。

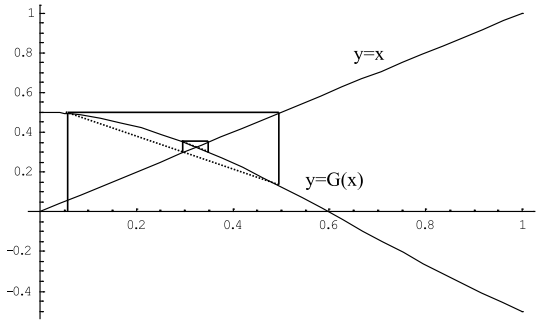

図 3: Steffesen の反復法における解への収束の流れ。

2.2.2 Aitken の Δ^2 法

$\Delta x_n \equiv x_{n+1} - x_n$、$\Delta^2 x_n \equiv \Delta(\Delta x_n) = x_n - 2x_{n+1} + x_{n+2}$ と定義し、新しい数列 $\bar{x}_n$ を

$$\begin{aligned}
\bar{x}_n &\equiv x_n - \frac{(x_{n+1} - x_n)^2}{x_n - 2x_{n+1} + x_{n+2}} && (18)\\
&= x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n} && (19)
\end{aligned}$$

と定義する。このような変換を Aitken 変換と呼ぶ。Aitken の Δ^2 法とは、既知の数列 x_n から、より早く収束する数列 $\bar{x}_n$ を作り出す方法である。この場合は、数列 x_n を作りながら数列 $\bar{x}_n$ を作ることになる。つまり、

$$
\begin{aligned}
x_1 &= G(x_0) && (20)\\
x_2 &= G(x_1) && (21)\\
\bar{x}_1 &= x_0 - \frac{(x_1 - x_0)^2}{x_0 - 2x_1 + x_2} && (22)\\
x_3 &= G(x_2) && (23)\\
\bar{x}_2 &= x_1 - \frac{(x_2 - x_1)^2}{x_1 - 2x_2 + x_3} && (24)\\
x_4 &= G(x_3) && (25)\\
\bar{x}_3 &= x_2 - \frac{(x_3 - x_2)^2}{x_2 - 2x_3 + x_4} && (26)\\
\vdots &= \vdots && (27)\\
& && (28)
\end{aligned}
$$

のような計算手順となる。Steffensen の反復法の手順と比べると、手順がやや異なっていることがわかる。Steffensen の反復法では、x_3 を生成する際、x_2 を用いるのではなく、$\bar{x}_1$ を用いている。これが、Steffensen の反復法が Aitken の Δ^2 法を少し改良したものであるといわれる所以である。

2.3 Shanks 変換と Wyne の ε アルゴリズム、Extended Steffensen 法

2.3.1 Shanks 変換

実は、式 (19) で表わされる Aitken 変換は、k 次の Shanks 変換の $k = 1$ の場合であるということが言える。k 次の Shanks 変換とは、

$$
\begin{aligned}
A_k^{(j)} &\equiv \begin{vmatrix} y_j & y_{j+1} & \cdots & y_{j+k} \\ y_{j+1} & y_{j+2} & \cdots & y_{j+k+1} \\ \vdots & \vdots & \ddots & y_{j+k+1} \\ y_{j+k} & y_{j+k+1} & \cdots & y_{j+2k} \end{vmatrix} && (29)\\
B_k^{(j)} &\equiv \begin{vmatrix} \Delta^2 y_j & \Delta^2 y_{j+1} & \cdots & \Delta^2 y_{j+k-1} \\ \Delta^2 y_{j+1} & \Delta^2 y_{j+2} & \cdots & \Delta^2 y_{j+k} \\ \vdots & \vdots & \ddots & y_{j+k+1} \\ \Delta^2 y_{j+k-1} & \Delta^2 y_{j+k} & \cdots & \Delta^2 y_{j+2k-2} \end{vmatrix} && (30)\\
\Delta^2 y_j &\equiv y_{j+2} - 2y_{j+1} + y_j && (31)
\end{aligned}
$$

という Hankel determinant と呼ばれる行列式を用いて

$$
e_k(y_j) \equiv \frac{A_k^{(j)}}{B_k^{(j)}} \tag{32}
$$

と定義される。ここで、$k = 1$ と置くと、

$$
\begin{aligned}
A_1^{(j)} &= \begin{vmatrix} y_j & y_{j+1} \\ y_{j+1} & y_{j+2} \end{vmatrix} && (33)\\
&= y_j y_{j+2} - y_{j+1}^2 && (34)\\
B_1^{(j)} &= \Delta^2 y_j && (35)
\end{aligned}
$$

であるから、

$$
\begin{aligned}
e_1(y_j) &= \frac{y_j y_{j+2} - y_{j+1}^2}{\Delta^2 y_j} && (36)\\
&= \frac{y_{j+2}y_j - 2y_{j+1}y_j + y_j^2 + 2y_{j+1}y_j - y_{j+1}^2 - y_j^2}{y_{j+2} - 2y_{j+1} + y_j} && (37)\\
&= y_j - \frac{(y_{j+1} - y_j)^2}{y_{j+2} - 2y_{j+1} + y_j} && (38)
\end{aligned}
$$

となり、式 (19) と等しくなる。

Shanks 変換においては $2k+1$ 個の $y_j, \cdots, y_{j+2k}$ を用いる。Aitken の Δ^2 法では $k=1$ なので 3 個である。つまり、Shanks 変換ではより多くの点を用いて次の点を求めていると言える。

2.3.2 Wyne の ε アルゴリズム

Shanks 変換においては、行列式を計算しなければならない。行列式の計算をより効率的に行うためのアルゴリズムが Wyne の ε アルゴリズムである。

さて、$e_k(y_j)$ を得たい。このとき、

$$
\begin{aligned}
\epsilon_{-1}^{(j)} &= 0 && (39)\\
\epsilon_{0}^{(j)} &= y_j && (40)\\
\epsilon_{i+1}^{(j)} &= \epsilon_{i-1}^{(j+1)} + \frac{1}{\epsilon_i^{(j+1)} - \epsilon_i^{(j)}} && (41)
\end{aligned}
$$

という数列を作ると、

$$
e_k(y_j) = \epsilon_{2k}^{(j)} \tag{42}
$$

となる。

次に具体的に $k=1$ のときに計算を行ってみる。

$$
\begin{aligned}
\epsilon_{-1}^{(j)} &= 0 && (43)\\
\epsilon_{-1}^{(j+1)} &= 0 && (44)\\
\epsilon_{-1}^{(j+2)} &= 0 && (45)\\
\epsilon_{0}^{(j)} &= y_j && (46)\\
\epsilon_{0}^{(j+1)} &= y_{j+1} && (47)\\
\epsilon_{0}^{(j+2)} &= y_{j+2} && (48)\\
\epsilon_{1}^{(j)} &= \epsilon_{-1}^{(j+1)} + \frac{1}{\epsilon_0^{(j+1)} - \epsilon_0^{(j)}} = \frac{1}{y_{j+1} - y_j} && (49)\\
\epsilon_{1}^{(j+1)} &= \epsilon_{-1}^{(j+2)} + \frac{1}{\epsilon_0^{(j+2)} - \epsilon_0^{(j+1)}} = \frac{1}{y_{j+2} - y_{j+1}} && (50)\\
\epsilon_{2}^{(j)} &= \epsilon_{0}^{(j+1)} + \frac{1}{\epsilon_1^{(j+1)} - \epsilon_1^{(j)}} && (51)\\
&= y_{j+1} + \frac{1}{\frac{1}{y_{j+2} - y_{j+1}} - \frac{1}{y_{j+1} - y_j}} && (52)\\
&= y_j - \frac{(y_{j+1} - y_j)^2}{y_{j+2} - 2y_{j+1} + y_j} && (53)
\end{aligned}
$$

となり、式 (38) と等しくなる。この計算はプログラムに容易に実装できる[2]。

[2] $i+1$ 番目の値 $\epsilon_{i+1}^{(j)}$ を求めるには、$i-1$ 用と i 用の配列を二つ用意し、Do ループを二重にかける。外側を変数 i のループ、内側を変数 j のループにして逐次計算していけばよい。

2.3.3 Extended Steffesen 法

Steffensen の反復法は Aitken 変換を用いたアルゴリズムであった。k 次の Shanks 変換を用いたアルゴリズムは Extended Steffensen 法と呼ばれる。このアルゴリズムの場合、$2k+1$ 個ごとに k 次の Shanks 変換を使い、収束を早くする。ここで、Wyne の ε アルゴリズムによって得られる Shanks 変換を $F(\boldsymbol{x})$ とおく。$\boldsymbol{x}$ は $2k+1$ 次元のベクトル $\boldsymbol{x} = (x_0, x_1, \cdots, x_{2k})$ である。このとき、$k = 2$ のときの具体的な計算手順は

$$
\begin{aligned}
x_1 &= G(x_0) && (54)\\
x_2 &= G(x_1) && (55)\\
x_3 &= G(x_2) && (56)\\
x_4 &= G(x_3) && (57)\\
x_5 &= F(\boldsymbol{x}) && (58)\\
x_6 &= G(x_5) && (59)\\
x_7 &= G(x_6) && (60)\\
\vdots &= \vdots && (61)\\
& && (62)
\end{aligned}
$$

となる。

3 計算時間の比較

基本的には、逐次代入法が最も収束が遅く、加速法のうちどれが一番収束が早いかはどのような計算を行うかによって異なるように思われる。今回はギャップ方程式の温度依存性の計算を行って、計算時間を比較する。今回用

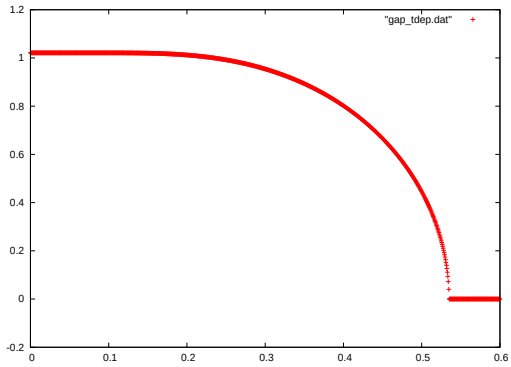

図 4: ギャップ方程式の温度依存性

いた計算機は、理化学研究所古崎物性理論研究室の計算機であり、スペックは「cat /proc/cpuinfo」等によれば

- CPU: Itanium 2 1.6GHz x2
- Physical memory: 16GB

である[3]。状況は

- 図. 4 が得られるようなパラメータ

[3]今回はメモリをほとんど使わない計算である。CPU は 1 つのみを使うプログラムである。

- 計算する点の数は 1000 点
- 収束判定は相対誤差が 10^{-4} となったとき
- 初期値は $\Delta = 1.5$
- エネルギー積分に用いたアルゴリズムは二重指数型数値積分アルゴリズム (DE 公式)
- 逐次代入法、Steffensen の反復法、Extended Steffensen 法の三つのアルゴリズムでの計算

である。

得られた結果は

- 逐次代入法：57.8 秒
- Stefffensen の反復法:10.7 秒
- 2 次の Extended Steffensen 法:14.5 秒

となった。逐次代入法は T_c 近傍において収束しなかった。この問題に限れば、$k = 1$ である Steffensen の反復法が一番収束が早かった。もしかすると、ほかの問題においては Extended Steffensen 法が収束が早くなる場合もあるのかもしれない。2 次の Extended Steffensen 法が早くならなかったのは、途中で Wyne のアルゴリズムによる計算を余計に行っているからかもしれない[4]。

参考文献

Koichi KONDO: Studies on Integrability for Nonlinear Dynamical Systems and its Applications, Doctor Thesis, Osaka University, June 21, 2001.

二宮市三編　”数値計算のわざ”　（共立出版）

[4] この Extended Steffensen 法は参考文献に挙げた近藤氏の博士論文に記述されていた方法である。この論文内では、より詳細な議論がなされている。