

クォークソルバーについて

石川 健一

広島大学先進理工系科学研究科



**Graduate School of
Advanced Science and Engineering**
Hiroshima University



広島大学

目的

- 格子上の場の理論の理論的背景はとても大事ですが、実際にそれを数値計算するとなると、プログラムや計算アルゴリズム、計算機の確保など技術的な面で困ることが多々あります。
- 格子上の場の理論のフェルミオン部分の取り扱いには理論的にも数値的にも厄介。
- ウィルソン・クローバー型フェルミオンの数値計算法を題目として、
 - 数値計算に必要なアルゴリズム
 - プログラム作成への取りかかり
- を伝えたいと思います。

もくじ

1. ウィルソンクロールバーフェルミオン作用と
フェルミオン伝搬関数の説明 格子QCDを
元に説明
2. 伝搬関数を数値的に求める方法の説明

1. ウィルソンクロールバーフェルミオン作用とフェルミオン伝搬関数の説明

- 1 - 1. ウィルソンフェルミオンの導入 (菊川先生の講義: 格子フェルミオン)
 - ディラックフェルミオン作用 (ユークリッド時空)

$$S_Q = \int d^4x \bar{\psi}(x) \left[\gamma_\mu (\partial_\mu - igG_\mu(x)) + m \right] \psi(x) = \int d^4x \bar{\psi}(x) [\gamma_\mu D_\mu + m] \psi(x)$$

$\psi(x) = (\psi_{a,\alpha}(x))$: フェルミオン場、 m は質量行列

ゲージ群SU(N)基本表現で変換される($a = 1, 2, \dots, N$)スピノール($\alpha = 1, 2, 3, 4$)

QCDの場合はクォークに該当。

$G_\mu(x)$: ゲージ場、ゲージ群SU(N)の代数の随伴表現に属する $N \times N$ のエルミート行列のベクトル場 QCDの場合はグルーオンに該当。

1. ウィルソンクロールバーフェルミオン作用とフェルミオン伝搬関数の説明

- 1 - 1. ウィルソンフェルミオンの導入 (菊川先生の講義: 格子フェルミオン)
 - 時空座標を格子座標で近似して共変微分を共編差分に置き換えると

$$\begin{aligned} S_Q &\simeq \sum_n a^4 \bar{\psi}(an) \left[\gamma_\mu \frac{1}{2a} \left(U_\mu(n) \psi(a(n + \hat{\mu})) - U_\mu^\dagger(n - \hat{\mu}) \psi(a(n - \hat{\mu})) \right) + m \psi(an) \right] \\ &= \sum_n \hat{\psi}(n) \left[\gamma_\mu \frac{1}{2} \left(U_\mu(n) \hat{\psi}(n + \hat{\mu}) - U_\mu^\dagger(n - \hat{\mu}) \hat{\psi}(n - \hat{\mu}) \right) + \hat{m} \hat{\psi}(n) \right] \end{aligned}$$

$\hat{\psi}(n) = a^{\frac{3}{2}} \psi(an)$: 格子上の無次元化されたフェルミオン場、 $\hat{m} = am$ は無次元質量行列

$U_\mu(n)$: 格子上のゲージ場、リンク変数。ゲージ群SU(N)基本表現に属する $N \times N$ の特殊ユニタリ行列のベクトル場

この作用 (ナイーブフェルミオン) は1粒子ではなく16個の粒子を含むのでだめなので、
改変した作用が使われる。

1. ウィルソンクローバーフェルミオン作用とフェルミオン伝搬関数の説明

- 1 - 1. ウィルソンフェルミオンの導入 (菊川先生の講義: 格子フェルミオン)
 - ウィルソンフェルミオン (Wilson 1975)

$$S_W = \sum_n \hat{\psi}(n) \left[\begin{array}{c} \gamma_\mu \frac{1}{2} (U_\mu(n) \hat{\psi}(n + \hat{\mu}) - U_\mu^\dagger(n - \hat{\mu}) \hat{\psi}(n - \hat{\mu})) \\ \underline{-\frac{r}{2} (U_\mu(n) \hat{\psi}(n + \hat{\mu}) - 2\hat{\psi}(n) + U_\mu^\dagger(n - \hat{\mu}) \hat{\psi}(n - \hat{\mu}))} \\ + \hat{m} \hat{\psi}(n) \end{array} \right]$$
$$\simeq \int d^4x \bar{\psi}(x) \left[\gamma_\mu D_\mu - \underline{\frac{ar}{2} (D_\mu)^2} + m \right] \psi(x)$$

この作用は1粒子を記述するが、質量ゼロ($m = 0$)にしても、カイラル対称性を破っている ar に比例する項 (ウィルソン項) を含む。有限の格子間隔 ($a \neq 0$) での計算では a に比例する誤差を含む。この誤差を消しつつ1粒子を記述するように a に比例する次元5の項を入れた作用がよく使われている。クローバー (ウィルソン・クローバー、Sheikholeslami-Wohlert) 作用と呼ばれる

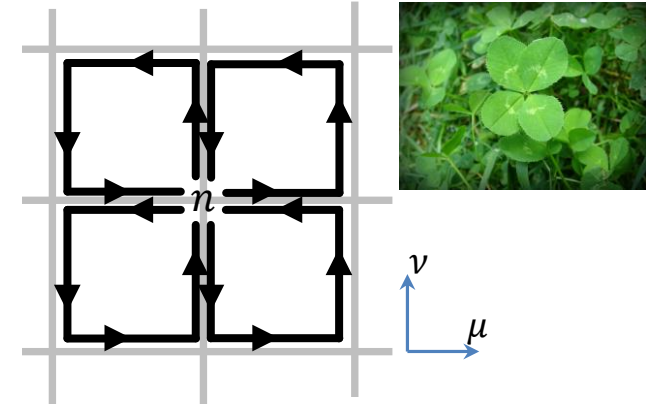
1. ウィルソクローバーフェルミオン作用とフェルミオン伝搬関数の説明

• 1-1. ウィルソクローバーフェルミオンの導入

- ウィルソクローバーフェルミオン (Sheikholeslami and Wohlert 1985)

$$\begin{aligned}
 S_{CL} &= \sum_n \hat{\psi}(n) \left[\begin{aligned} &\gamma_\mu \frac{1}{2} (U_\mu(n) \hat{\psi}(n + \hat{\mu}) - U_\mu^\dagger(n - \hat{\mu}) \hat{\psi}(n - \hat{\mu})) \\ & - \frac{r}{2} (U_\mu(n) \hat{\psi}(n + \hat{\mu}) - 2\hat{\psi}(n) + U_\mu^\dagger(n - \hat{\mu}) \hat{\psi}(n - \hat{\mu})) \\ & + \hat{m} \hat{\psi}(n) - \frac{c_{SW}}{4} \sigma_{\mu\nu} \hat{G}_{\mu\nu}^{CL}(n) \hat{\psi}(n) \end{aligned} \right] \\
 &\simeq \int d^4x \bar{\psi}(x) \left[\gamma_\mu D_\mu - \frac{ar}{2} (D_\mu)^2 - \frac{ac_{SW}}{4} \sigma_{\mu\nu} G_{\mu\nu}(x) + m \right] \psi(x) + O(a^2) \\
 &= \int d^4x \bar{\psi}(x) [\gamma_\mu D_\mu + m] \psi(x) + O(a^2)
 \end{aligned}$$

Wikipedia:四つ葉のクローバー



$\hat{G}_{\mu\nu}^{CL}(n)$: ゲージ場の強さテンソルの格子版

$G_{\mu\nu}(x)$: ゲージ場の強さテンソル (電磁気学の電磁場に対応)

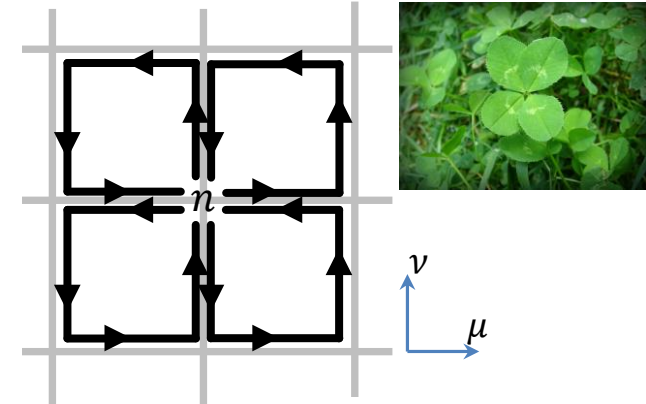
ツリーレベルで ($r = c_{SW}$) のとき運動方程式を使うと、有限の格子間隔 ($a \neq 0$) でも a に比例する誤差がない。量子化される場合は、観測量 (期待値) に $O(a)$ の誤差が出ないように c_{SW} を微調整する必要がある。

1. ウィルソクローバーフェルミオン作用とフェルミオン伝搬関数の説明

• 1 - 1. ウィルソクローバーフェルミオンの導入

- ウィルソクローバーフェルミオン (Sheikholeslami and Wohlert 1985)

Wikipedia:四つ葉のクローバー



$\hat{G}_{\mu\nu}^{CL}(n)$: ゲージ場の強さテンソルの格子版

$$S_{CL} = \sum_{n,m} \hat{\psi}(n) D_{CL}(n, m) \hat{\psi}(m)$$

$$D_{CL}(n, m) \equiv -\frac{r}{2} \left(U_{\mu}(n) \delta_{n+\hat{\mu},m} - 2\delta_{n,m} + U_{\mu}^{\dagger}(n - \hat{\mu}) \delta_{n-\hat{\mu},m} \right) + \hat{m} \delta_{n,m} - \frac{c_{SW}}{4} \sigma_{\mu\nu} \hat{G}_{\mu\nu}^{CL}(n) \delta_{n,m}$$

ウィルソクローバーフェルミオン演算子 (行列) $D_{CL}(n, m)$ の形で表した。行列の大きさ (次元) は 4次元格子点数 x スピン x SU(N) のランク = 4次元格子点数 x 4 x N。

QCDで4次元格子が 128^4 の格子なら、次元は $128^4 \times 4 \times 3 = 3,221,225,472 = 32$ 億次元。とても大きな行列

1. ウィルソクローバーフェルミオン作用とフェルミオン伝搬関数の説明

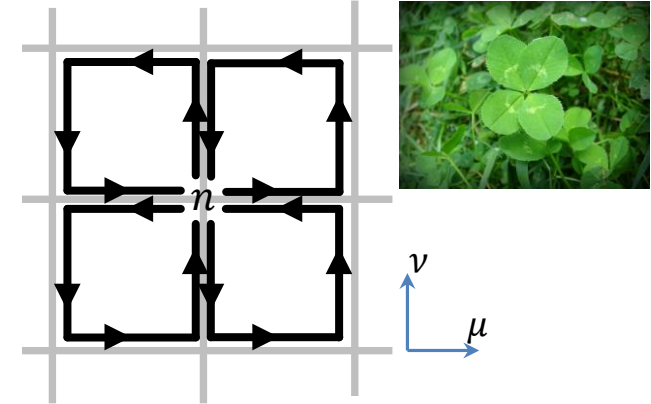
- 1 - 1. ウィルソクローバーフェルミオンの導入
 - ウィルソクローバーフェルミオン (Sheikholeslami and Wohlert 1985)

$$S_{CL} = \sum_{n,m} \hat{\psi}(n) D_{CL}(n,m) \hat{\psi}(m)$$

$$D_{CL}(n,m) \equiv -\frac{r}{2} \sum_{\mu=1}^4 \left[U_{\mu}(n) \delta_{n+\hat{\mu},m} - \underline{2\delta_{n,m}} + U_{\mu}^{\dagger}(n - \hat{\mu}) \delta_{n-\hat{\mu},m} \right]$$

$$+ \underline{\hat{m}\delta_{n,m}} - \frac{c_{SW}}{4} \sigma_{\mu\nu} \hat{G}_{\mu\nu}^{CL}(n) \delta_{n,m}$$

Wikipedia:四つ葉のクローバー



$\hat{G}_{\mu\nu}^{CL}(n)$: ゲージ場の強さテンソルの格子版

数値計算し易いように、単位行列に比例するところの係数が1になるように場の規格化を変えたものをシミュレーションでは使う。 物理量の期待値を計算するときには最後に規格化を標準的なものに戻す。

1. ウィルソクローバーフェルミオン作用とフェルミオン伝搬関数の説明

Wikipedia:四つ葉のクローバー



- 1 - 1. ウィルソクローバーフェルミオンの導入
 - ウィルソクローバーフェルミオン (Sheikholeslami and Wohlert 1985)

$$S_{CL} = \sum_{n,m} \hat{\psi}'(n) D_{CL}'(n,m) \hat{\psi}'(m)$$

$$D_{CL}'(n,m) \equiv \left[1 - \frac{c_{SW}\kappa}{2} \sigma_{\mu\nu} \hat{G}_{\mu\nu}^{CL}(n) \right] \delta_{n,m} - \kappa \sum_{\mu=1}^4 [(r - \gamma_{\mu}) U_{\mu}(n) \delta_{n+\hat{\mu},m} + (r + \gamma_{\mu}) U_{\mu}^{\dagger}(n - \hat{\mu}) \delta_{n-\hat{\mu},m}]$$

$$\kappa \equiv \frac{1}{2(4r + \hat{m})}$$

ホッピングパラメター
フェルミオンが隣接する格子点を
移動する (ホップする) 強さを表す = 質量に反比例

数値計算し易いように、単位行列に比例するところの係数が1になるように場の規格化を変えたものをシミュレーションでは使う。物理量の期待値を計算するときには最後に規格化を標準的なものに戻す。数値計算ではウィルソン項の係数 r は $r = 1$ と置くのが標準的であり、また数値計算の計算量の観点からも推奨される。4次元格子対角部分の $\left[1 - \frac{c_{SW}\kappa}{2} \sigma_{\mu\nu} \hat{G}_{\mu\nu}^{CL}(n) \right]$ 項は 12×12 エルミート行列が各格子点にある。

1. ウィルソクローバーフェルミオン作用とフェルミオン伝搬関数の説明

• 1 – 2. フェルミオン伝搬関数

- フェルミオン演算子の期待値は $\hat{\psi}(n)$ と $\hat{\psi}(m)$ が Wick 縮約されてフェルミオン伝搬関数に置き換わる。格子ゲージ理論でも同じ。経路積分でフェルミオン場のグラスマン数積分を先に行い、ゲージ場（リンク変数）の積分はせずついておくので、フェルミオン伝搬関数はリンク変数の関数として求める必要がある。解析的にはこれは不可能なので、数値的に与えられた背景リンク変数のもとで伝搬関数を数値的に求める必要がある。

- 渡辺先生講義：経路積分量子化

- 山崎先生講義：格子ゲージ理論

- 金森先生講義：マルコフ連鎖モンテカルロとHMC

$$\langle O[U, q, \bar{q}] \rangle_{U, q, \bar{q}} = \langle O[U, S_q[U]] \rangle_U$$

$$S_q[U] = D_q[U]^{-1} : \text{格子上のフェルミオン伝搬関数}$$

$$D_q[U] : \text{格子上のフェルミオンのディラック演算子}$$

ウィルソクローバーフェルミオン、クローバーフェルミオン、オーバーラップフェルミオン、ドメインウォールフェルミオン

1. ウィルソンクローバーフェルミオン作用とフェルミオン伝搬関数の説明

• 1 – 2. フェルミオン伝搬関数

- フェルミオン演算子の期待値は $\hat{\psi}(n)$ と $\hat{\psi}(m)$ が Wick 縮約されてフェルミオン伝搬関数に置き換わる。格子ゲージ理論でも同じ。経路積分でフェルミオン場のグラスマン数積分を先に行い、ゲージ場（リンク変数）の積分はせずついておくので、フェルミオン伝搬関数はリンク変数の関数として求める必要がある。解析的にはこれは不可能なので、数値的に与えられた背景リンク変数のもとで伝搬関数を数値的に求める必要がある。

- 渡辺先生講義：経路積分量子化

- 山崎先生講義：格子ゲージ理論

- 金森先生講義：マルコフ連鎖モンテカルロとHMC

$$\langle O[U, q, \bar{q}] \rangle_{U, q, \bar{q}} = \langle O[U, S_q[U]] \rangle_U$$

$$\simeq \frac{1}{N_{\text{sample}}} \sum_{s=1}^{N_{\text{sample}}} O[U^{(s)}, S_q[U^{(s)}]]$$

$N_{\text{sample}} \gg 1$

$U^{(s)} = \{U_\mu(n)\}^{(s)}$: MCMCで生成されたリンク変数のs番目のサンプル。

$S_q[U^{(s)}] = D_q[U^{(s)}]^{-1}$: サンプル上での格子上のフェルミオン伝搬関数

1. ウィルソンクロールバーフェルミオン作用とフェルミオン伝搬関数の説明

• 1 – 2. フェルミオン伝搬関数

- 例： π^\pm 中間子の2点関数 $O(n) = \bar{u}(n)\gamma_5 d(n)$, $u(n), d(n)$: アップ/ダウクォークの $q_{u/d}(n)$

$$\begin{aligned} \langle O(n)^\dagger O(m) \rangle_{U,u,d,\bar{u},\bar{d}} &= \langle \bar{d}(n)\gamma_5 u(n)\bar{u}(m)\gamma_5 d(m) \rangle_{U,u,d,\bar{u},\bar{d}} \\ &= -\langle \text{Tr}[\gamma_5 u(n)\bar{u}(m)\gamma_5 d(m)\bar{d}(n)] \rangle_{U,u,d,\bar{u},\bar{d}} \\ &= -\langle \text{Tr}[\gamma_5 S_u[U](n,m)\gamma_5 S_d[U](m,n)] \rangle_U \\ &\simeq \frac{-1}{N_{\text{sample}}} \sum_{s=1}^{N_{\text{sample}}} \text{Tr}[\gamma_5 S_u[U^{(s)}](n,m)\gamma_5 S_d[U^{(s)}](m,n)] \end{aligned}$$

$N_{\text{sample}} \gg 1$

- 一般に伝搬関数の四次元格子座標の添字の全列成分を求めるのは不可能： $(S_u[U^{(s)}](n,m))$ の m の添字
- m を一つか数個ほどで計算して並進対称性を課して平均。時間格子座標依存性からエネルギー = 質量が求まる。

$$C(n_t) = \frac{1}{V} \sum_{\vec{n}} \langle O(n)^\dagger O(0) \rangle_{U,u,d,\bar{u},\bar{d}} = \sum_{\text{state}} A_{\text{state}} e^{-\hat{E}_{\text{state}} n_t}$$

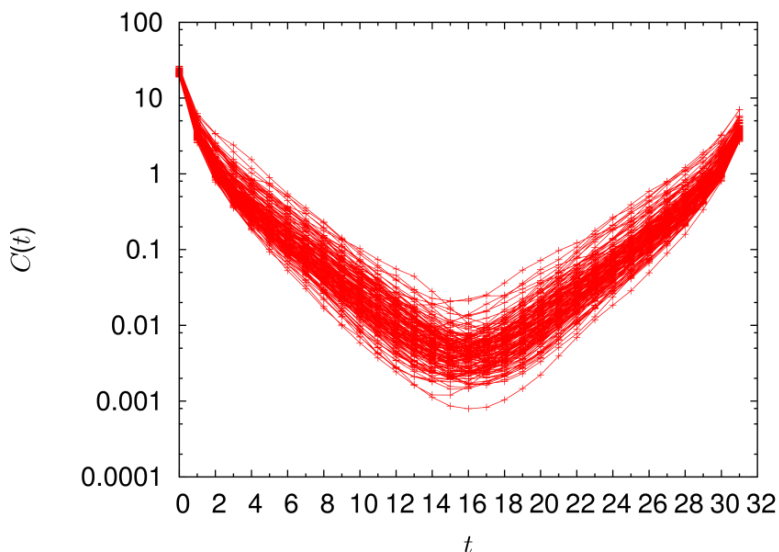
$S_{u/d}[U^{(s)}](n, 0)$: をサンプルの数だけ求める必要がある。

1. ウィルソンクロールバーフェルミオン作用とフェルミオン伝搬関数の説明

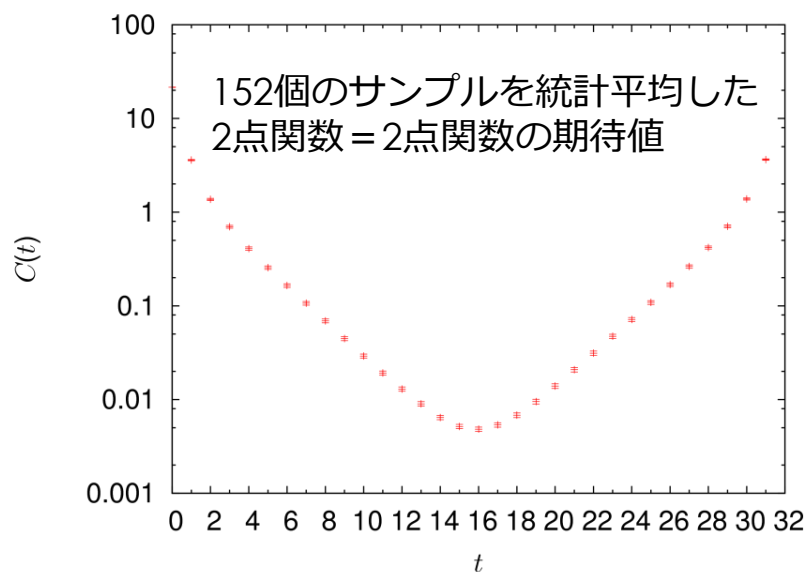
• 1 - 2. フェルミオン伝搬関数

- 例: π^\pm 中間子の2点関数 $O(n) = \bar{u}(n)\gamma_5 d(n)$, $u(n), d(n)$: アップ/ダウクォークの $q_{u/d}(n)$

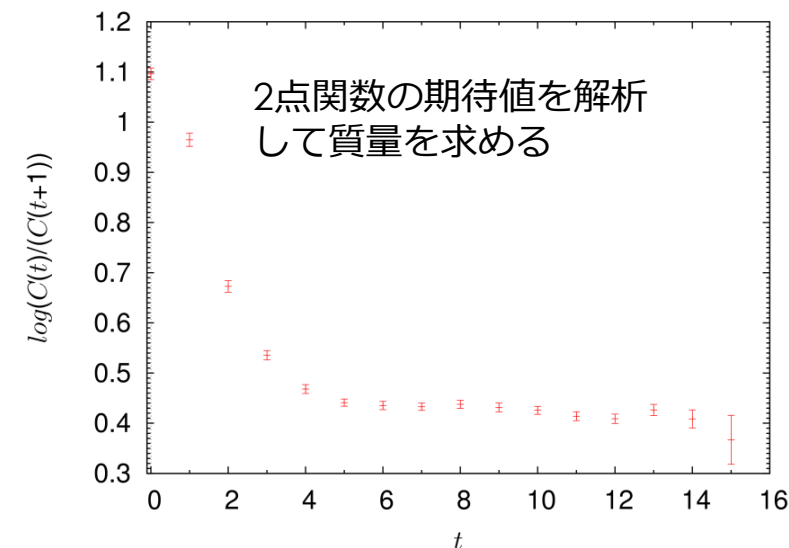
PS meson propagator, 152 samples



PS meson propagator, averaged



PS meson effective mass



152個のサンプルの各
2点関数
($12^3 \times 32$ 格子, $b=1.8$,
Iwasaki+Clover,
 $\kappa=0.1328$)

$$C(n_t) = \frac{1}{V} \sum_{\vec{n}} \langle O(n)^\dagger O(0) \rangle_{U,u,d,\bar{u},\bar{d}} = \sum_{\text{state}} A_{\text{state}} e^{-\hat{E}_{\text{state}} n_t}$$

$S_{u/d}[U^{(s)}](n, 0)$: をサンプルの数だけ求める必要がある。

1. ウィルソクローバフェルミオン作用とフェルミオン伝搬関数の説明

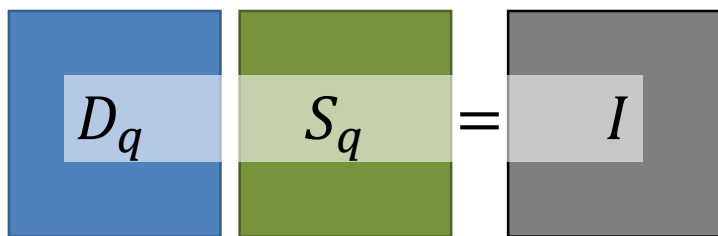
•まとめ

- ✓ウィルソクローバフェルミオンの運動方程式の係数行列を確かめた。
- ✓フェルミオンを含む格子上的物理量の期待値にはフェルミオンの伝搬関数が含まれる。
- ✓伝搬関数はフェルミオンの運動方程式に現れる係数行列の逆行列である。
- ✓係数行列の次元はとても大きい。
- ✓期待値を計算するためにモンテカルロサンプルのゲージ場のサンプルごとに伝搬関数を求めておく必要がある。

2. 伝搬関数を数値的に求める方法の説明

• 2-1. フェルミオン伝搬関数の求め方の方針

- 格子上のディラック演算子は巨大な行列である。その行列の成分をすべてメモリ上に保持することは普通は不可能です。ただし、ウィルソン型であればそのほとんどの成分はゼロなのでゼロでない成分を持つことは可能です。
- しかしながら、その行列の逆行列になると、すべての成分がゼロでない値を持つため、メモリ上に持つことは不可能になります。QCDの 128^4 格子だと次元が 32 億次元でした。複素数の 32 億次元の行列のコンピュータ上での記憶容量は倍精度複素数では 144 エクサバイトのようです。2020 年頃の世界全体のストレージ容量は 6.7 ゼタバイトだったらしいです。1000 エクサ = 1 ゼタなので世界で協力すれば 60 個ほど持てますね。
- ではその成分はどうやって求めるかというと愚直には連立方程式を解きます。

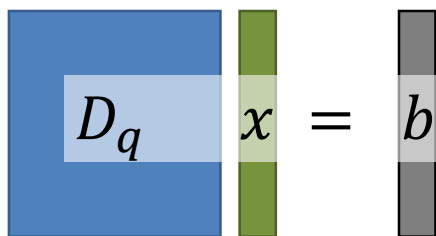

$$D_q [U^{(s)}] S_q = I$$

$$D_q [U^{(s)}] S_q = I : \text{巨大な連立方程式を解いて } S_q \text{ を求める}$$

2. 伝搬関数を数値的に求める方法の説明

• 2-1. フェルミオン伝搬関数の求め方の方針

- 格子上のディラック演算子は巨大な行列である。その行列の成分をすべてメモリ上に保持することは普通は不可能です。ただし、ウィルソン型であればそのほとんどの成分はゼロなのでゼロでない成分を持つことは可能です。
- しかしながら、その行列の逆行列になると、すべての成分がゼロでない値を持つため、メモリ上に持つことは不可能になります。QCDの 128^4 格子だと次元が 32 億次元でした。複素数の 32 億次元の行列のコンピュータ上での記憶容量は倍精度複素数では 144 エクサバイトのようです。2020 年頃の世界全体のストレージ容量は 6.7 ゼタバイトだったらしいです。1000 エクサ = 1 ゼタなので世界で協力すれば 60 個ほど持てますね。無理！
- ではその成分はどうやって求めるかというと愚直には連立方程式を解きます。

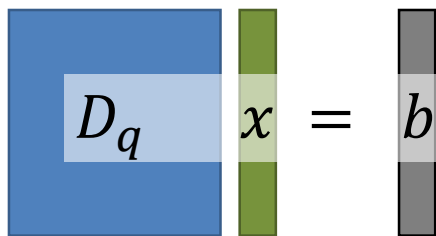

$$D_q x = b$$

$$D_q[U^{(s)}]x = b : \text{巨大な連立方程式を解いて}$$
$$x : S_q \text{の一部の列を求める}$$

2. 伝搬関数を数値的に求める方法の説明

• 2-1. フェルミオン伝搬関数の求め方の方針

- ではその成分はどうやって求めるかという愚直には連立方程式を解きます。

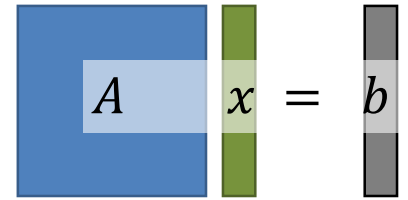

$$D_q x = b$$

$D_q[U^{(s)}]x = b$: 巨大な連立方程式を解いて
 $x : S_q$ の**一部の列**を求める

- 数値計算の線形代数の学部の講義でやったかもしれませんが、「掃き出し法」や「LU分解」を用いて解くのは係数行列 D_q の成分を保持していないとできないのでかなり厄介です。仮に保持できていたとしても計算の量（足し算、引き算、掛け算を何回行うか: number of floating-point number operations, FLOP/FLOPS）が行列の大きさの3乗に比例するため不可能です。
- 「掃き出し法」や「LU分解」のように行列の成分を動かしながら解く方法を「**直接法**」といいます。
- ほしいのは解 x の数値精度内（倍精度実数だと相対精度で 10^{-15} 程度）での近似値で良いので、この近似値を求める方法を説明します。

2. 伝搬関数を数値的に求める方法の説明

$$Ax = b$$



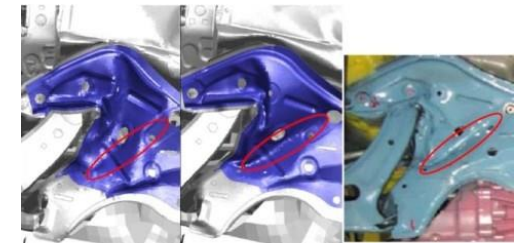
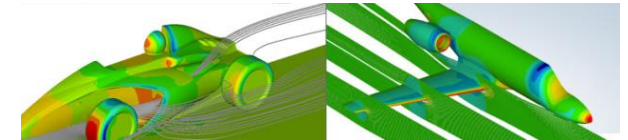
$$A_{ij} = D_{\alpha,\beta}^{a,b}(n, m)$$

• 2-2. 反復法による近似解法

• この手の大規模な連立方程式を近似的に解く方法として大別して以下の2つがある

– 定常反復法

- ヤコビ(Jacobi)反復法
- ガウス・ザイデル(Gauss-Seidel (GS))法
- 逐次過緩和反復法(Successive Over Relaxation (SOR) iteration)
-

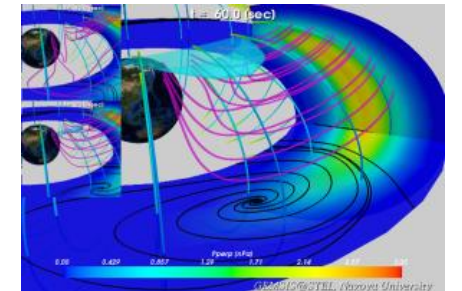


100万メッシュモデル計算結果 1000万メッシュモデル計算結果 実際の衝突実験結果写真
提供：(独)海洋研究開発機構・(社)自動車工業会

– クリロフ(Krylov)部分空間法、非定常反復法

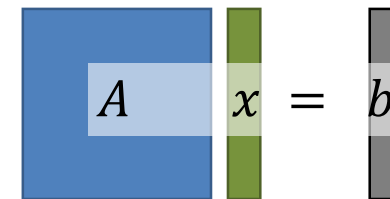
- 共役勾配法 Conjugate Gradient (CG) algorithm
- 双共役勾配法 Bi-Conjugate Gradient (BiCG) algorithm とその仲間たち
- 一般化最小残差法 Generalized Minimal Residual (GMRES) algorithm

構造力学や流体の運動方程式を解くのも同じぐらいの問題



2. 伝搬関数を数値的に求める方法の説明

$$Ax = b$$



• 2-2. 反復法による近似解法

- この手の大規模な連立方程式を近似的に解く方法として大別して以下の2つがある

- 定常反復法
- クリロフ(Krylov)部分空間法、非定常反復法

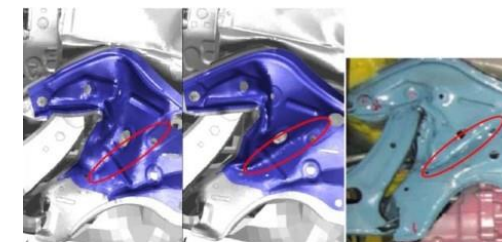
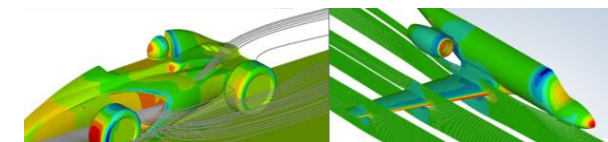
- クリロフ(Krylov)部分空間法と一部の定常反復法に必要な部品は、

入力ベクトル v に係数行列 A を掛けて出力ベクトル w を得るという

$$w = Av$$

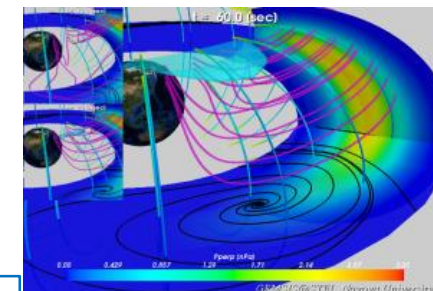
の計算部分である。係数行列を保持しなくて良くて、複数本のベクトルを保持できれば計算方法が確立する。

- 具体的なアルゴリズムの導出はここでは行わないが、大まかな考え方と連立方程式解法のアルゴリズムの教科書を読むための基礎・言葉を紹介する。



100万メッシュモデル計算結果 1000万メッシュモデル計算結果 実際の衝突実験結果写真
提供：(独)海洋研究開発機構・(社)自動車工業会

構造力学や流体の運動方程式を解くのも同じぐらいの問題



- ギリシャ文字：スカラー量
- ローマ字小文字：ベクトル量
- ローマ字大文字：行列

2. 伝搬関数を数値的に求める方法の説明

$$Ax = b$$

• 2-2. 反復法による近似解法

- 定常反復法の一番簡単なものの一つに Neumann/Taylor 展開に基づく方法がある

$f(z) = \frac{1}{1-z}$ (z : 複素数) の Neumann/Taylor 展開は

$$f(z) = 1 + z + z^2 + z^3 + \dots \quad (|z| < 1)$$

- ここで複素数 z を行列 $z \rightarrow (I - A)$ に置き換え、 ($|I - A| < 1$ 、固有値が 1 より小さい)

$$A^{-1} = (I - (I - A))^{-1} = I + (I - A) + (I - A)^2 + (I - A)^3 + \dots$$

- $Ax = b$ の解 $x = A^{-1}b$ は以下のアルゴリズムで求まる。

$$Ax = b$$

連立方程式
A: 係数行列
x: 未知ベクトル
b: 右辺ベクトル



(step0) $x = b$

(step1) $r = b - Ax$

(step2) $x = x + r$

(step3) if $|r|$ is sufficiently small then exit else goto step1.

近似解 x の初期値

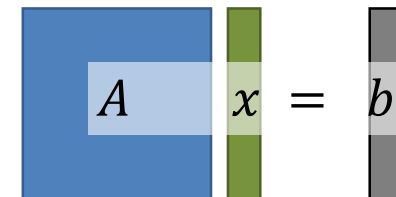
Ax を計算する。
誤差ベクトル $r = b - Ax$ を計算する。

近似解 x を修正する

$$x = A^{-1}b = b + (1 - A)b + (1 - A)^2 b + (1 - A)^3 b + \dots$$

2. 伝搬関数を数値的に求める方法の説明

$$Ax = b$$



• 2-2. 反復法による近似解法

- 定常反復法の一番簡単なものの一つに Neumann/Taylor 展開に基づく方法がある
- $Ax = b$ の解 $x = A^{-1}b$ は以下のアルゴリズムで求まる。

$$Ax = b$$

連立方程式
A: 係数行列
x: 未知ベクトル
b: 右辺ベクトル



(step0) $x = b$

(step1) $r = b - Ax$

(step2) $x = x + r$

(step3) if $|r|$ is sufficiently small then exit else goto step1.

近似解 x の初期値

Ax を計算する。
誤差ベクトル $r = b - Ax$ を計算する。

近似解 x を修正する

$$x = A^{-1}b = b + (1 - A)b + (1 - A)^2 b + (1 - A)^3 b + \dots$$

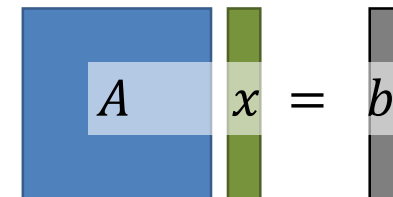
- 定常反復法・非定常反復法に共通の要素は、

- 行列掛けるベクトル $w = Av$ の計算ができればよい
- 誤差ベクトルの更新と解ベクトルの更新部分のペア

- 固有値が条件を満たさない場合は収束しないが、 $A = N + M$, $|N^{-1}M| < 1$ となる分解ができれば収束するアルゴリズムを作ることができる (詳細略)

2. 伝搬関数を数値的に求める方法の説明

$$Ax = b$$



• 2-2. 反復法による近似解法

- 定常反復法の一番簡単なものの一つに Neumann/Taylor 展開に基づく方法がある

- $Ax = b$ の解 $x = A^{-1}b$ は以下のアルゴリズムで求まる。

$$Ax = b$$

連立方程式
A: 係数行列
x: 未知ベクトル
b: 右辺ベクトル



(step0) $x = b$

(step1) $r = b - Ax$

(step2) $x = x + r$

(step3) if $|r|$ is sufficiently small then exit else goto step1.

近似解 x の初期値

Ax を計算する。
誤差ベクトル $r = b - Ax$ を計算する。

近似解 x を修正する

- ウィルソン型フェルミオンの場合

$$x = A^{-1}b = b + (1 - A)b + (1 - A)^2 b + (1 - A)^3 b + \dots$$

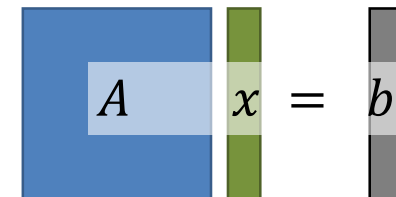
$A = I - \kappa H$, H : ホッピング行列 + クローバー項

- の形でかつ有限のクォーク質量なら $|\kappa H| < 1$ なので Neumann/Taylor 展開に基づく方法で解が求まる。

$$A^{-1}b = b + (\kappa H)b + (\kappa H)^2 b + (\kappa H)^3 b + \dots$$

2. 伝搬関数を数値的に求める方法の説明

$$Ax = b$$

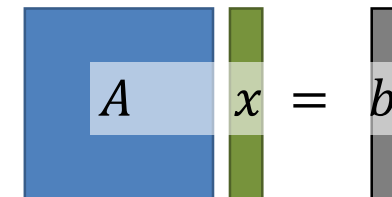


• 2-2. 反復法による近似解法

- Neumann/Taylor 展開に基づく方法以外の定常反復法には係数行列 A の成分の並びに注目して解を更新していくものがある。ガウスザイデル法やヤコビ法がそれに該当する。
- これらは、格子QCDでは主に時空格子点同士のつながり（ホッピング）に対応する部分に注目して解を更新していくものになる。
- 「行列掛けるベクトル $w = Av$ の計算ができればよい」の範疇を超える説明が必要なので本講義では説明しない。
- ひょっとすると学部の数値計算の講義とかでガウスザイデル法やヤコビ法を学習したかもしれない。それらも格子QCDのフェルミオン（クォーク）伝搬関数の計算に使うこともできることを言及しておく。

2. 伝搬関数を数値的に求める方法の説明

• 2 – 3. クリロフ(Krylov)部分空間法、非定常反復法 $Ax = b$


$$A x = b$$

– 先のNeumann/Taylor 展開に基づく方法にアルゴリズム的に似た形であるが、もっと積極的に誤差ベクトルを小さくするように解の更新ベクトルの向きや大きさを調整しながら近似解を求める方法です。

– ほぼ世の中の大規模連立方程式を解く方法はこのクリロフ(Krylov)部分空間法に基づきます。

– 格子QCDでは

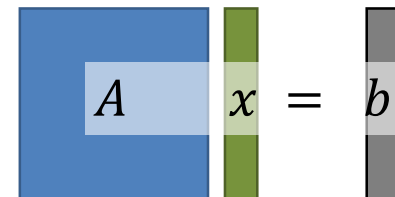
- 共役勾配法 Conjugate Gradient (CG) algorithm
- 安定化双共役勾配法 Bi-Conjugate Gradient Stabilized (BiCGStab) algorithm
- [一般化最小残差法 Generalized Minimal Residual (GMRES) algorithm]
- がよく使われます。

– アルゴリズムのエッセンスや文献を読むための豆知識を紹介します。

2. 伝搬関数を数値的に求める方法の説明

• 2-3. クリロフ(Krylov)部分空間法、非定常反復法

$$Ax = b$$



- 行列 A は有限の大きさである。その大きさを $N \times N$ とする。でもとても大きな N です。

- 行列 A の固有値方程式 (永年方程式) $p(\lambda)$ と行列の間には Cayley-Hamilton の定理があります。

$$p(\lambda) \equiv \det[\lambda I - A] = \lambda^N + c_{N-1}\lambda^{N-1} + c_{N-2}\lambda^{N-2} + \dots + c_1\lambda + c_0$$

$$p(A) \equiv A^N + c_{N-1}A^{N-1} + c_{N-2}A^{N-2} + \dots + c_1A + c_0I = 0$$

- $p(A) = 0$ より、 A^k ($k \geq N$) のべきは A^j の ($j \leq N - 1$) の冪の多項式で表現できます。

- したがって、解析的な関数 $f(z)$ の行列版 $f(A)$ を考えると、これは A の $N - 1$ じの多項式で表されるはずで。

$$f(A) = g_{N-1}A^{N-1} + g_{N-2}A^{N-2} + \dots + g_1A + g_0I$$

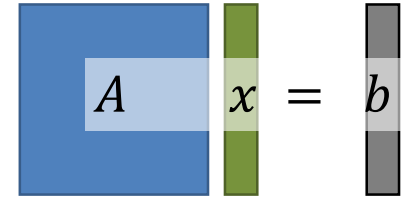
- 行列の逆行列 A^{-1} についても同様に考えることができます。

$$A^{-1} = g_{N-1}A^{N-1} + g_{N-2}A^{N-2} + \dots + g_1A + g_0I$$

2. 伝搬関数を数値的に求める方法の説明

• 2-3. クリロフ(Krylov)部分空間法、非定常反復法

$$Ax = b$$



- 行列の逆行列 A^{-1} についても同様に考えることができます。

$$A^{-1} = g_{N-1}A^{N-1} + g_{N-2}A^{N-2} + \dots + g_1A + g_0I$$

- この多項式に基づいて解を構成するのは N がとても大きいのと係数 g_j を求めるのがやはり膨大な計算が必要なので無理です。でも誤差ベクトルを必要な数値精度でゼロに近くなるように

$$r = b - Ax = b - A[d_{M-1}A^{M-1}b + d_{M-2}A^{M-2}b + \dots + d_1Ab + d_0b] \simeq 0$$

$$x = d_{M-1}A^{M-1}b + d_{M-2}A^{M-2}b + \dots + d_1Ab + d_0b \simeq A^{-1}b$$

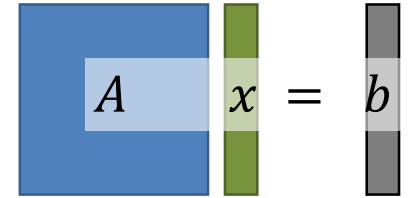
- と近似解ベクトル x を $K_M(A, b) = \text{Span}\{b, Ab, A^2b, \dots, A^{M-2}b, A^{M-1}b\}$ のベクトルをつかって適応的に展開係数 d_j を決めながら構成する方法があれば、 $M \ll N$ で解の近似を得ることができると期待します。

- この $K_M(A, b)$ を、行列 A とベクトル b が張る M 次元のクリロフ(Krylov)部分空間といいます。

2. 伝搬関数を数値的に求める方法の説明

- 2-3. クリロフ(Krylov)部分空間法、非定常反復法

$$Ax = b$$



–クリロフ(Krylov)部分空間法とはこのような残差ベクトルを最小化するような多項式係数を反復中に適応的に決めていく方法の総称です。

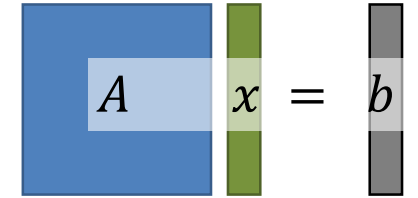
$$r = b - Ax = b - A[d_{M-1}A^{M-1}b + d_{M-2}A^{M-2}b + \dots + d_1Ab + d_0b] \simeq 0$$

$$x = d_{M-1}A^{M-1}b + d_{M-2}A^{M-2}b + \dots + d_1Ab + d_0b \simeq A^{-1}b$$

2. 伝搬関数を数値的に求める方法の説明

• 2-3. クリロフ(Krylov)部分空間法、非定常反復法

$$Ax = b$$



- クリロフ(Krylov)部分空間法に基づくアルゴリズムの概形

(Step0) 解の初期値 $x = x^{(0)}$

(Step1) 残差ベクトルの初期値 $r = b - Ax$

(Step2) 反復法の始まり:

...

(StepX) スカラー係数 (多項式係数) α とベクトル v をここまで計算してきた反復中のベクトルを使って計算する。
(内積、ベクトルの足し算等の線形計算を使う)
どのような計算を行なうかはアルゴリズムによるが下のステップでの残差が小さくなるように決めるものとする。

(StepX+1) 残差ベクトルを更新する。 $r = r + \alpha Av$

(StepX+2) 解ベクトルを更新する。 $x = x - \alpha v$

....

(StepZ) 残差ベクトルのノルム $|r|$ が十分小さくないなら Step2 へ行く。

係数 α やベクトル v の作り方の詳細は最適化・適応化の方法・アルゴリズムに依存する。 v がすでに $A^k r$ の多項式になっていることが重要。

行列ベクトル積の回数を増やさないためには、解と残差の更新時に解と残差の関係が崩れないように行うこと。

残差 r が $r = b - Ax$ を満たしているとき
次の式で与えられる更新

$$r' = r + \alpha Av$$

$$x' = x - \alpha v$$

は、更新された量の間関係

$$r' = b - Ax'$$

を保つ。

- ギリシャ文字: スカラー量
- ローマ字小文字: ベクトル量
- ローマ字大文字: 行列

2. 伝搬関数を数値的に求める方法の説明 $Ax = b$

• 2-4. 共役勾配法 Conjugate Gradient (CG) Algorithm

- CG法: どの教科書にも書いてあるアルゴリズム

- $Ax = b$ の方程式で係数行列 A がエルミート行列で正の固有値のみを含むときに方程式を解くことができる。
- 格子上的フェルミオン行列はそのままではエルミート行列ではない。CG法を使うためには次のように変形した方程式 (正規方程式) を解く。

$$D\phi = \eta \Rightarrow D^\dagger D\phi = D^\dagger \eta$$

↓

$$A \equiv D^\dagger D, x = \phi, b = D^\dagger \eta, Ax = b$$

- ただし、この方法はあまりおすすめしない。正規方程式を解くための少し改良されたCG法があるのでそちらを使うこと。

```

x = x(0); r = b - Ax; p = r
ρ0 = ⟨r | r⟩
do
  q = Ap
  α = ρ0 / ⟨p | q⟩
  x = x + αp
  r = r - αq
  ρ1 = ⟨r | r⟩
  if |r| = √ρ1 is sufficiently small exit do loop
  β = ρ1 / ρ0; ρ0 = ρ1
  p = r + βp
end do
    
```

- ギリシャ文字: スカラー量
- ローマ字小文字: ベクトル量
- ローマ字大文字: 行列

数値計算で線型方程式を解くアルゴリズムの応用数学の教科書で使われる記号の使い方のスタイル。
 注意点として複素数の線形代数ではないことが多いので、自分で実数行列から複素数の行列の解法に焼き直す必要がある。内積のところの左側と右側のベクトルの区別を自分で考える。

$$A x = b$$

• 2 - 4. 共

- CG法 : と

- $Ax = b$ の方で正の固有値とができる。

- 格子上のフュミート行列でように変形し

$$D\phi$$

$$A \equiv D^\dagger D,$$

- ただし、この式を解くためにを使うこと。

```

 $x_0$  is an initial guess,  $r_0 = b - Ax_0$ 
for  $i = 1, 2, \dots$ 
     $\rho_{i-1} = r_{i-1}^T r_{i-1}$ 
    if  $i = 1$ 
         $p_i = r_{i-1}$ 
    else
         $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$ 
         $p_i = r_{i-1} + \beta_{i-1} p_{i-1}$ 
    endif
     $q_i = Ap_i$ 
     $\alpha_i = \rho_{i-1} / p_i^T q_i$ 
     $x_i = x_{i-1} + \alpha_i p_i$ 
     $r_i = r_{i-1} - \alpha_i q_i$ 
    if  $x_i$  accurate enough then quit
end

```

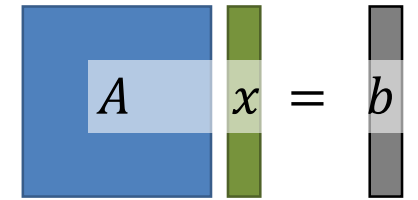
- ギリシャ文字 : スカラー量
- マ字小文字 : ベクトル量
- マ字大文字 : 行列

数値計算で線型方程式を解くアルゴリズムの応用数学の教科書で使われる記号の使い方のスタイル。
 注意点として複素数の線形代数ではないことが多いので、自分で実数行列から複素数の行列の解法に焼き直す必要がある。内積のところの左側と右側のベクトルの区別を自分で考える。

P.42 CG alg., H.A. van der Vorst, "Iterative Krylov Methods for Large Linear Systems", Cambridge Univ. Press.より

Figure 5.1. Conjugate Gradients without preconditioning.

2. 伝搬関数を数値的に求める方法の説明 $Ax = b$



• 2-4. 共役勾配法 Conjugate Gradient (CG) Algorithm

-例: 1次元のエルムホルツ方程式 $(-\Delta + \kappa^2)\phi = \rho$ を周期境界条件で解く。

N 格子点で差分化すると (格子間隔は場や係数に吸収)

$$-\phi(i+1) + (2 + \kappa^2)\phi(i) - \phi(i-1) = \rho(i), i = 1, \dots, N$$

$$\phi(N+1) = \phi(1), \phi(0) = \phi(N)$$

-行列ベクトル積 $w = Av$ の部分は

$$v(N+1) = v(1)$$

$$v(0) = v(N)$$

for $i = 1, N$

$$w(i) = -v(i+1) + f * v(i) - v(i-1)$$

end for

-Fortran コード例: `Helmholz1D.tar.gz` をSlack#09-12-1_cg-solver のところに置きました。

$$\begin{pmatrix} f & -1 & 0 & \dots & 0 & -1 \\ -1 & f & -1 & \ddots & & 0 \\ 0 & -1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & f & -1 \\ -1 & 0 & \dots & 0 & -1 & f \end{pmatrix} \begin{pmatrix} \phi(1) \\ \phi(2) \\ \vdots \\ \vdots \\ \phi(N-1) \\ \phi(N) \end{pmatrix} = \begin{pmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \vdots \\ \rho(N-1) \\ \rho(N) \end{pmatrix}$$

$f = 2 + \kappa^2$

2. 伝搬関数を数値的に求めるための説明 $Av = b$

• 2-4. 共役勾配法 Conjugate Gradient

-例: 1次元のエルムホルツ方程式 $(-\Delta + \kappa^2)\phi = b$

-行列ベクトル積 $w = Av$ の部分は

$$v(N+1) = v(1)$$

$$v(0) = v(N)$$

for $i = 1, N$

$$w(i) = -v(i+1) + f * v(i) - v(i-1)$$

end for

-Fortran コード例: [Helmholz1D.tar.gz](#)

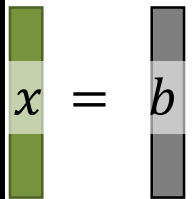
をSlack#09-12-1_cg-solver のところに置きました。

```
subroutine assign_mult_helmhz_op(k2,v,p)
!
! Return v = -Laplace(p) + k2 p
!
implicit none
real(DP), intent(in) :: k2
real(DP), intent(out) :: v(0:NSITE+1)
real(DP), intent(inout) :: p(0:NSITE+1)
integer :: i

!
! set periodic boundary condition on the ghost sites.
!
p(0) = p(NSITE)
p(NSITE+1) = p(1)

!
! Apply Laplacian on p
!
do i=1,NSITE
v(i) = -p(i+1)+(2.0_DP+k2)*p(i)-p(i-1)
enddo

return
end subroutine
```



2. 伝搬関数を数値的に求める方法の説明 $Ax = b$

• 2-4. 共役勾配法 Conjugate Gradient

- 例: 1次元のエルムホルツ方程式 $(-\Delta + \kappa^2)\phi = b$

- 行列 A $x = x^{(0)}$; $r = b - Ax$; $p = r$

$$\rho_0 = \langle r | r \rangle$$

do

$$q = Ap$$

$$\alpha = \frac{\rho_0}{\langle p | q \rangle}$$

$$x = x + \alpha p$$

$$r = r - \alpha q$$

- Fortran

$$\rho_1 = \langle r | r \rangle$$

をSlack $\text{if } |r| = \sqrt{\rho_1}$ is sufficiently small exit do loop 量きま

$$\beta = \frac{\rho_1}{\rho_0}; \rho_0 = \rho_1$$

$$p = r + \beta p$$

end do

```

iter = 0
call assign_mult_helmhz_op(k2,q,x)      ! q = A x
r(1:NSITE) = b(1:NSITE) - q(1:NSITE)  ! r = b - q = b - A x (initial residual)
p(1:NSITE) = r(1:NSITE)                ! p = r
rho0 = abs2(r)                          ! rho0 = <r|r> = |r|^2

err = sqrt(rho0)
write(*,'(I3,2ES24.15)')iter,err

do
  iter = iter + 1

  call assign_mult_helmhz_op(k2,q,p)    ! q = A p

  alpha = rho0/prod(p,q)

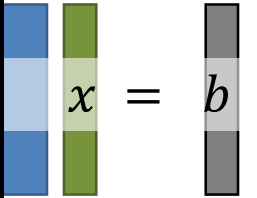
  x(1:NSITE) = x(1:NSITE) + alpha * p(1:NSITE)
  r(1:NSITE) = r(1:NSITE) - alpha * q(1:NSITE)

  rho1 = abs2(r)                        ! rho1 = <r|r> = |r|^2
  err = sqrt(rho1)
  write(*,'(I3,2ES24.15)')iter,err
  if (err < tol) exit
  if (iter > NITER) then
    write(*,'("CG algorithm did not converge.",I4)')iter
    exit
  endif

  beta = rho1/rho0
  rho0 = rho1

  p(1:NSITE) = r(1:NSITE) + beta * p(1:NSITE)

enddo
    
```



2. 伝搬関数を数値的に求める方法の説明 $Ax = b$

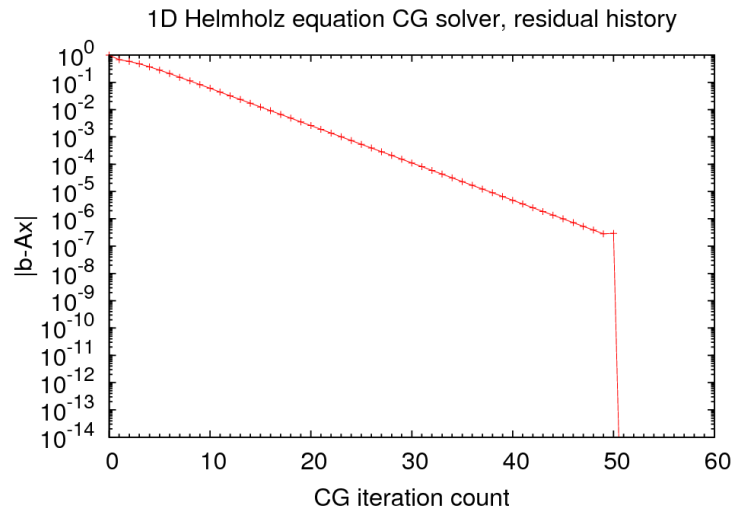
• 2-4. 共役勾配法 Conjugate Gradient (CG) Algorithm

$$A x = b$$

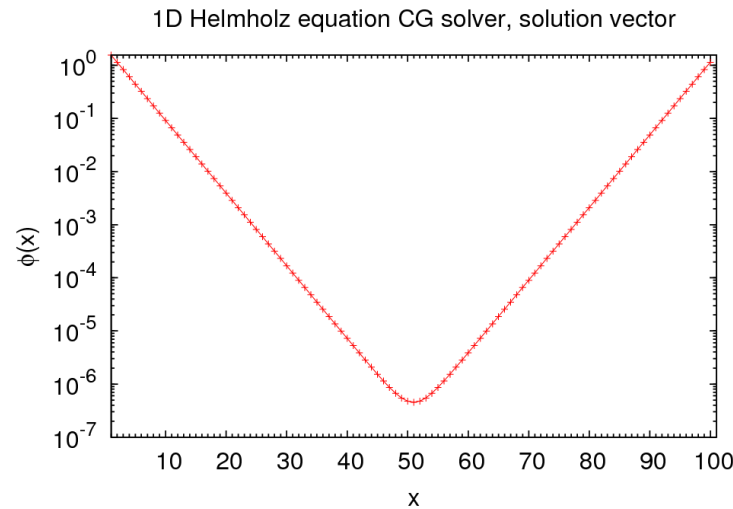
- 例: 1次元のエルムホルツ方程式 $(-\Delta + \kappa^2)\phi = \rho$ を周期境界条件で解く。

- 格子点数 $N = 100, \kappa^2 = 0.1, \rho(n) = \delta_{n,1}$ の実行例 (gfortran, make)

反復毎の残差 (Residual History)



解の様子 (Solution vector)

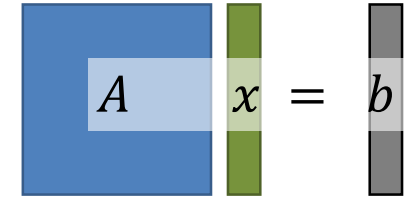


$$\begin{pmatrix} f & -1 & 0 & \dots & 0 & -1 \\ -1 & f & -1 & \ddots & & 0 \\ 0 & -1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & f & -1 \\ -1 & 0 & \dots & 0 & -1 & f \end{pmatrix} \begin{pmatrix} \phi(1) \\ \phi(2) \\ \vdots \\ \vdots \\ \phi(N-1) \\ \phi(N) \end{pmatrix} = \begin{pmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \vdots \\ \rho(N-1) \\ \rho(N) \end{pmatrix}$$

$f = 2 + \kappa^2$

- 残差の減り方は反復回数に対して指数関数的に収束する。
- ケーリーハミルトンから考えれば100次元の問題なので (クリロフ部分空間サイズ100次元) 100反復付近で完全に収束するはず。
- GC法では1反復で部分空間が1次元ずつ増えていく。実際の計算では50反復目で完全に収束した。実際にこのような突如の収束が起こることは格子QCDではまずありえない (行列サイズがとて大きいため)。

2. 伝搬関数を数値的に求める方法の説明 $Ax = b$



• 2-4. 共役勾配法 Conjugate Gradient (CG) Algorithm

- 残差の減り方 (収束の速さ) は係数行列の固有値の分布に支配される。
- 応用数学の研究により、真の解と近似解の差に対して次の不等式が成り立つことが示されている。

$$\|x^* - x^{(m)}\|_A \leq 2 \left[\frac{\sqrt{K} - 1}{\sqrt{K} + 1} \right]^m \|x^* - x^{(0)}\|_A$$

$$\|v\|_A \equiv \sqrt{\langle v | A | v \rangle} \quad Aノルム$$

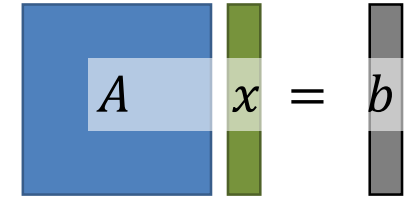
$$K = \kappa(A) \equiv \|A\| \|A^{-1}\| \quad \text{行列}A\text{の条件数}$$

x^* 真の解 $x^{(m)}$ CG法の反復 m 回目での近似解 $x^{(0)}$ 初期値

- 係数行列の条件数が大きいと収束は遅い。
 - 格子QCDでのウィルソン・クローバーフェルミオンの場合、CG法に適用される係数行列は D の2乗の $A = D^\dagger D$ である。この行列の条件数は元の D の条件数の2乗になり一般に大きい。=>反復回数が多い
- $$\kappa(D^\dagger D) = |\kappa(D)|^2$$
- 非エルミート行列用の線形方程式アルゴリズムがほしい。
 - 格子QCD計算でウィルソン・クローバーフェルミオンにもっとも使われてきたアルゴリズムがあります。
BiCGStab (Bi Conjugate Gradient Stabilized)法

2. 伝搬関数を数値的に求める方法の説明 $Ax = b$

- 2 – 5. 安定化双共役勾配法 Bi-Conjugate Gradient Stabilized (BiCGStab) Algorithm [BiCGStab: van der Vorst (1992)]

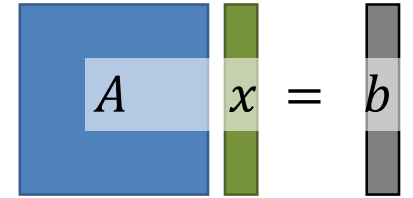

$$Ax = b$$

- BiCG法を安定化させたもの。
- 非エルミート行列（ウィルソクローバフェルミオンの行列）に対して用いることができるアルゴリズム。ただし、行列の固有値に負の実部をもつものがあると解けなくなる。
- ウィルソクローバフェルミオンの行列に対するアルゴリズムとしてはメモリの的にもスピード的にも最も速いことが知られている。
For LQCD Wilson quarks, introduced by [Frommer et al. Int.J.Mod.Phys. C5 (1994) 1073]
- 近年開発された様々な工夫と組み合わせるとほかの方法が速い時もある
- アルゴリズムの流れはCG法と似ている。しかし、1反復内で**行列ベクトル積を2回行う**ところが違う。これは行列の非エルミート性のためである。

2. 伝搬関数を数値的に求める方法の説明

$$Ax = b$$

- 2 - 5. 安定化双共役勾配法 Bi-Conjugate Gradient Stabilized (BiCGStab) Algorithm [BiCGStab: van der Vorst (1992)]



- アルゴリズム

```
x = x(0); r = b - Ax; p = r;  
Shadow vector  $\tilde{r}$  can be arbitrary.  
A simple choice is set  $\tilde{r} = r$ .  
 $\rho_0 = \langle \tilde{r} | r \rangle$   
do  
  q = Ap  
   $\alpha = \rho_0 / \langle \tilde{r} | q \rangle$   
  x = x +  $\alpha p$   
  r = r -  $\alpha q$   
  if |r| is sufficiently small exit do loop
```

```
t = Ar  
 $\omega = \langle t | r \rangle / \langle t | t \rangle$   
x = x +  $\omega r$   
r = r -  $\omega t$   
if |r| is sufficiently small exit do loop  
 $\rho_1 = \langle \tilde{r} | r \rangle$   
 $\beta = (\alpha / \omega) (\rho_1 / \rho_0); \rho_0 = \rho_1$   
p = r +  $\beta(p - \omega q)$   
end do
```

- 係数とベクトルを計算して残差と近似解を更新する。1反復内で2か所行っている。
- このアルゴリズムの実行例プログラム例を次に示す。

2. 伝¹³⁶

9. Bi-CGSTAB

の説明 $Ax = b$

• 2-5. 安定化双共役 (BiCGStab)

- アルゴリズム

```

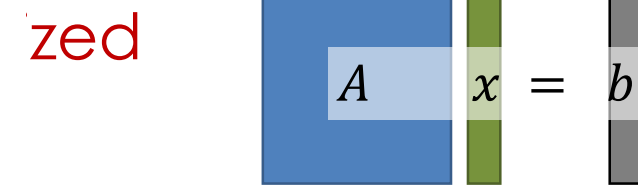
x = x(0); r
Shadow v
A simple
ρ0 = ⟨r̃ | r
do
  q = Ap
  α = ρ0 /
  x = x +
  r = r -
  if |r| is
  
```



- 係数とベクトルを計算し
- このアルゴリズムの実行

```

x0 is an initial guess; r0 = b - Ax0
Choose r̃, for example, r̂ = r0
for i = 1, 2, ...
  ρi-1 = r̃T ri-1
  if ρi-1 = 0 method fails
  if i = 1
    pi = ri-1
  else
    βi-1 = (ρi-1/ρi-2)(αi-1/ωi-1)
    pi = ri-1 + βi-1(pi-1 - ωi-1vi-1)
  endif
  vi = Api;
  αi = ρi-1/r̃T vi
  s = ri-1 - αivi
  check ||s||2, if small enough: xi = xi-1 + αipi and stop
  t = As, ωi = tTs/tTt
  xi = xi-1 + αipi + ωis
  ri = s - ωit
  check convergence; continue if necessary
  for continuation it is necessary that ωi ≠ 0
end
  
```



Il exit do loop
ρ₁

P.136 Bi-CGSTAB alg., H.A. van der Vorst, "Iterative Krylov Methods for Large Linear Systems", Cambridge Univ. Press.より

る。

Figure 9.1. The Bi-CGSTAB algorithm.

2. 伝搬関数を数値的に求める方法の説明 $Ax = b$

- 2 - 5. 安定化双共役勾配法 Bi-Conjugate Gradient Stabilized (BiCGStab) Algorithm [BiCGStab: van der Vorst (1992)]

- 1次元のウィルソンフェルミオンみたいな係数行列 (非エルミート) の例

$$(D\phi)(i) \equiv \phi(i) - \kappa[(1 - \sigma_3)\phi(i + 1) + (1 + \sigma_3)\phi(i - 1)] = \rho(i)$$

$$\phi(N + 1) = \phi(1), \phi(0) = \phi(N)$$

$$\phi(i) = (\phi_1(i), \phi_2(i))^T, \sigma_3: \text{パウリ行列の3つめ}$$

$$\begin{pmatrix} E & -F & 0 & \cdots & 0 & -B \\ -B & E & -F & \ddots & & 0 \\ 0 & -B & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & E & -F \\ -F & 0 & \cdots & 0 & -B & E \end{pmatrix} \begin{pmatrix} \phi(1) \\ \phi(2) \\ \vdots \\ \vdots \\ \phi(N-1) \\ \phi(N) \end{pmatrix} = \begin{pmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \vdots \\ \rho(N-1) \\ \rho(N) \end{pmatrix}$$

- 係数行列 D は非対称行列

- CG方は $D\phi = \rho$ を解くためには直接は使えない。

$$E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, F = \begin{pmatrix} 0 & 0 \\ 0 & 2\kappa \end{pmatrix}, B = \begin{pmatrix} 2\kappa & 0 \\ 0 & 0 \end{pmatrix}$$

- Fortran プログラム例: Slack #19-12-1_cg-solver に Wilson1DBiCGStab.tar.gz を置きました。

(gfortran,make)

2. 伝搬関数を数値的に求める方法の説明 $Ax = b$

• 2-5. 安定化双共役勾配法 Bi-Conjugate Gradient (BiCGStab) Algorithm [BiCGStab: va

- 1次元のウィルソンの方法 (Wilson's method) $x = x^{(0)}; r = b - Ax; p = r;$
 - Fortran プログラムの Shadow vector \tilde{r} can be arbitrary. r (非零) r (非零)
- A simple choice is set $\tilde{r} = r$.
- $$\rho_0 = \langle \tilde{r} | r \rangle$$
- do
- $$q = Ap$$
- $$\alpha = \frac{\rho_0}{\langle \tilde{r} | q \rangle}$$
- $$x = x + \alpha p$$
- $$r = r - \alpha q$$
- if $|r|$ is sufficiently small exit do loop

```

iter = 0
call assign_mult_wd_op(kp,q,x)      ! q = A x
r(:,1:NSITE) = b(:,1:NSITE) - q(:,1:NSITE) ! r = b - q = b - A x (initial residual)
p(:,1:NSITE) = r(:,1:NSITE)      ! p = r
call RANDOM_NUMBER(rt)           ! rt <= random / normalized vector
rt(:,1:NSITE) = rt(:,1:NSITE)/abs2(rt)
rho0 = prod(rt,r)                 ! rho0 = <rt|r>

err = sqrt(abs2(r))
write(*,'(I3,2ES24.15)')iter,err
errmax = err

do
  iter = iter + 1

  call assign_mult_wd_op(kp,q,p) ! q = A p

  alpha = rho0/prod(rt,q)

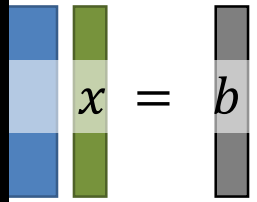
  x(:,1:NSITE) = x(:,1:NSITE) + alpha * p(:,1:NSITE)
  r(:,1:NSITE) = r(:,1:NSITE) - alpha * q(:,1:NSITE)

  err = sqrt(abs2(r))
  errmax = MAX(err,errmax)
  write(*,'(I3,2ES24.15)')iter,err

  if ( err < errmax*delta ) then
    !
    ! compute true residual
    !
    call assign_mult_wd_op(kp,t,x)      ! t = A x
    r(:,1:NSITE) = b(:,1:NSITE) - t(:,1:NSITE) ! r = b - t = b - A x
    err = sqrt(abs2(r))
    errmax = err
  endif

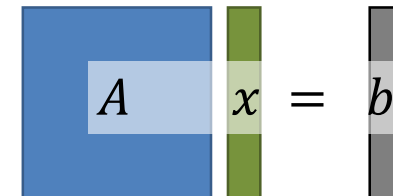
  if (err < tol) exit
  if (iter >= NITER) then
    write*,'("# BiCGStab algorithm did not converge.",I4)'iter
    exit
  endif
enddo

```



2. 伝搬関数を数値的に求める方法の説明

$$Ax = b$$



• 2 - 5

安定化双共役勾配法 Bi-Conjugate Gradient Stabilized

[BiCGStab: van der Vorst (1992)]

- 1次元
- Fortran

```

endif
iter = iter + 1

call assign_mult_wd_op(kp,t,r) ! t = A r
omega = prod(t,r)/abs2(t)

x(:,1:NSITE) = x(:,1:NSITE) + omega * r(:,1:NSITE)
r(:,1:NSITE) = r(:,1:NSITE) - omega * t(:,1:NSITE)

err = sqrt(abs2(r))
errmax = MAX(err,errmax)
write(*,'(I3,2ES24.15)')iter,err

if ( err < errmax*delta ) then
!
! compute true residual
!
call assign_mult_wd_op(kp,t,x) ! t = A x
r(:,1:NSITE) = b(:,1:NSITE) - t(:,1:NSITE) ! r = b - t = b - A x
err = sqrt(abs2(r))
errmax = err
endif

if (err < tol) exit
if (iter > NITER) then
write(*,'(BiCGStab algorithm did not converge.",I4)')iter
exit
endif

rho1 = prod(rt,r)
beta = (alpha/omega)*(rho1/rho0)
rho0 = rho1

p(:,1:NSITE) = r(:,1:NSITE) + beta * ( p(:,1:NSITE) - omega * q(:,1:NSITE))

enddo
    
```

列 (非
er (2
pp

$$t = Ar$$

$$\omega = \frac{\langle t | r \rangle}{\langle t | t \rangle}$$

$$x = x + \omega r$$

$$r = r - \omega t$$

if $|r|$ is sufficiently small exit do loop

$$\rho_1 = \langle \tilde{r} | r \rangle$$

$$\beta = \left(\frac{\alpha}{\omega} \right) \left(\frac{\rho_1}{\rho_0} \right); \rho_0 = \rho_1$$

$$p = r + \beta(p - \omega q)$$

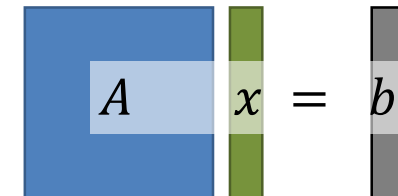
end do

ました。
ran,make)

2. 伝搬関数を数値的に求める方法の説明

$$Ax = b$$

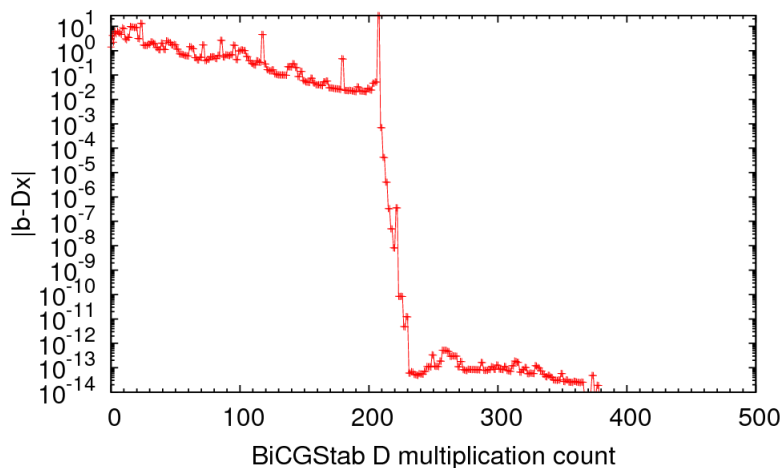
• 2 – 5. 安定化双共役勾配法 Bi-Conjugate Gradient Stabilized (BiCGStab) Algorithm [BiCGStab: van der Vorst (1992)]



- 1次元のウィルソンフェルミオンみたいな係数行列（非エルミート）の例
- 格子点数：N = 100, パラメータ：K=0.49, $\rho^a(n) = \delta_{n,1}$

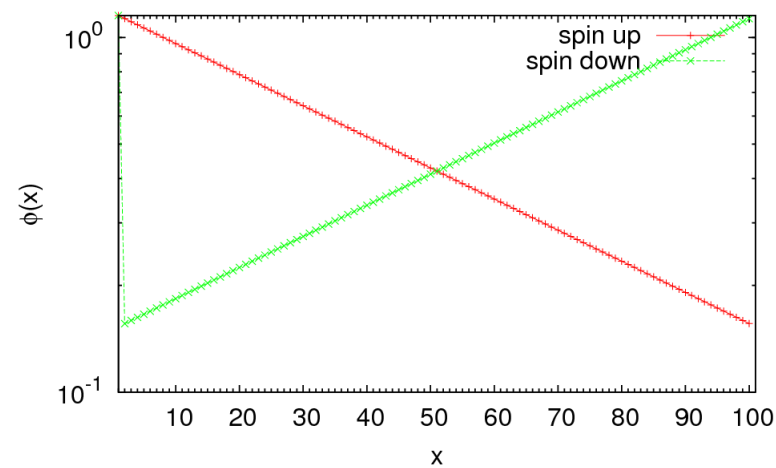
反復毎の残差 (Residual History)

1D Wilson like equation ($\kappa=0.49$) BiCGStab solver residual history



解の様子 (Solution vector)

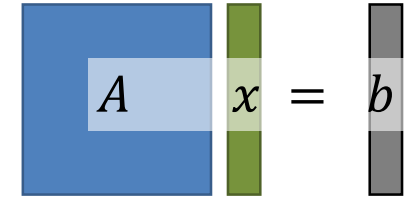
1D Wilson like equation ($\kappa=0.49$) BiCGStab solver solution vector



- BiCGStab 法は残差の大きさがふらついて単調ではない。この振る舞いは BiCGStab 法では一般的である。(紹介しないがCGNE 法（正規方程式に対するCG法）は単調に残差が減少する。)
- 格子QCDのウィルソン・クローバーフェルミオンの場合はBiCGStab法の方が経験的にはCGNE 法より2倍速い。

2. 伝搬関数を数値的に求める方法の説明 $Ax = b$

•まとめ


$$A x = b$$

- ✓ 格子場の理論の伝搬関数は運動方程式を差分化した場合の行列の逆行列。
- ✓ 格子点数が多い場合は逆行列そのものを求めることは不可能。格子QCDでは現在のところ計算時間が長くなり不可能。逆行列の一部の列を解くことになるが、この場合線形方程式を解くことに対応。
- ✓ 線型方程式を数値的に解く方法に、定常反復法と非定常反復法がある。
- ✓ 格子QCDにおいて定常反復法をそのまま使うことはまれ。非定常反復法（クリロフ部分空間法）と組み合わせて使う。非定常反復法のうちクリロフ部分空間法と呼ばれる方法が良くつかわれる。
- ✓ ウィルソン・クローバーフェルミオンに対してはクリロフ部分空間法のうちCG法と、BiCGStab法が使われてきた。これらのアルゴリズムが基礎。
- ✓ 話せなかったこと：前処理方法、数値精度のこと、収束条件の設定方法、高速化法、4次元での実際の実装方法。

2. 伝搬関数を数値的に求める方法の説明 $Ax = b$

$$A x = b$$

• 参考文献

- ✓ 藤野清次、張紹良 著, 「反復法の数理」応用数値計算ライブラリ、朝倉書店
- ✓ R.Barrett et al 著, 長谷川里美、長谷川秀彦、藤野清次 訳, 「反復法Templates」応用数値計算ライブラリ、朝倉書店
- ✓ Henk A. van der Vorst, “Iterative Krylov Methods for Large Linear Systems”, Cambridge Monographs on Applied and Computational Mathematics, Cambridge Univ Press
- ✓ Yousef Saad, “Iterative Methods for Sparse Linear Systems”, SIAM, (2nd ed) ISBN 0-89871-534-2 (pbk.)
- ✓ Gene H. Golub and Charles F. Van Loan, “Matrix Computations”, The Johns Hopkins Univ. Press, (3rd ed ISBN 0-8018-5414-8 (pbk.)

クォークソルバー_{について}

- (学校なので) 問題

- ① 例に出てきたプログラムを実行してみよう。
- ② 1次元の Helmholtz 方程式ソルバーを2次元や3次元の問題へ拡張してみよう。
- ③ 1次元のウィルソンフェルミオン方程式ソルバーを4次元の自由場のウィルソンフェルミオン方程式に拡張してみよう。(自由場 = ゲージ場 $U=1$)

おしまい