平成 13 年度

学士学位論文

共役勾配法における誤差の評価とその軽減

Evaluation and Reduction of Error in Conjugate Gradient Method

1020332　　山本真弘

指導教員　　福本昌弘

2002 年 2 月 8 日

高知工科大学 情報システム工学科

要 旨

共役勾配法における誤差の評価とその軽減

山本真弘

適応アルゴリズムの 1 つにブロック直交射影アルゴリズムがあり，その一実現法としては、共役勾配法を利用したものが知られている．

実際に適応アルゴリズムを利用するときには，雑音の影響は避けられない．

本研究では，雑音による推定精度の劣化を軽減することを検討している．まず共役勾配法のうち，計算手順が簡単であることから HS 版を使用している．次に，推定精度を決める主な要因が反復回数であることに着目し，シミュレーションによって推定誤差を最小にする反復回数を与えている．これにより，推定精度の劣化を軽減するための指針を与えている．

キーワード　適応アルゴリズム，共役勾配法，反復回数，推定精度，誤差，雑音

Abstract

Evaluation and Reduction of Error in Conjugate Gradient Method

Masahiro Yamamoto

The Block Orthogonal Projection algorithm is one of adaptive algorithms, and the Conjugate Gradient Method is used as a methods of achiving it. When the adaptive algorithm is actually used, it must take notice that the influence of the noise is not avoided. In this paper, it is considered that the way of reducing deterioration in it accuracy by the noise. First of all, since the HS(Hestenes and Stiefel) version is used, it has easy procedure to calculate the Conjugate Gradient Methods. Next, it is attention to the number of iterations because the main factor is decided the presumption accuracy, and then the number of iterations to minimize the presumption error is given by the computer simulations. As a result, the indicator to reduce deterioration in the presumption accuracy is provided.

key words Adaptive algorithm, Conjugate Gradient Method, Number of iterations, Presumption accuracy, Error, Noise

目次

図目次

表目次

第 1 章

序論

1.1 背景と目的

代表的な適応アルゴリズムとして学習同定法がある．しかし，これは入力信号が有色性の信号であるとき，収束速度が著しく劣化することが知られている．この解決策として，アフィン射影算法があるが，演算量が多くなってしまうという難点がある．この点を改善するために，ブロック処理を導入したブロック直交射影アルゴリズムが提案された．これを実現する一方法として共役勾配法を用いたものがある [1].

共役勾配法は，反復法でありながら，有限回のステップで厳密解に到達するという性質を持っている [2]．しかし，実際に適応アルゴリズムを用いるときには，雑音が存在する．そのため，共役勾配法を適応アルゴリズムに適用した場合には厳密解には収束せず，多くの回数を繰り返しても十分な精度が得られない．すなわち，雑音の影響により，収束速度と収束精度を両立させることが困難になる．

本研究では，何種類かある共役勾配法から適応アルゴリズムで利用するのに適したものを選び出し，収束精度を決める主な要因である反復回数を設定するための指針を示す．さらに，信号の相関係数の違いによって反復回数はどのような影響を受けるかを考察する．

1.2 本論文の概要

第 2 章では，本研究を行う上で必要な適応アルゴリズムと共役勾配法についての説明をおこなう．第 3 章では，安定性の高い共役勾配法を選び出し，シュミレーションをする．

第 2 章

適応アルゴリズムと共役勾配法

2.1　まえがき

本章では，適応アルゴリズムについて述べる．そして，適応アルゴリズムの問題点やそれを克服するための手法を示す．また問題点を克服したアルゴリズムの一実現法である，共役勾配法について述べる．

2.2　FIR ディジタルフィルタ

離散的な入力信号に対して，アナログフィルタの周波数特性と同等なフィルタリングを行うように設計されたフィルタのことをディジタルフィルタという．

FIR ディジタルフィルタの特徴として，

- フィルタのインパルス応答が有限項で表現される

  未知システムのインパルス応答が無限に続く場合は、同じ入力信号に対し完全に等しい出力を与える FIR ディジタルフィルタを見つけることはできない．
- 常に安定性を満たす

- 完全な線形位相をもつ特性がある

2.3　適応アルゴリズム

適応アルゴリズムを，図 2.1 を用いて説明する．

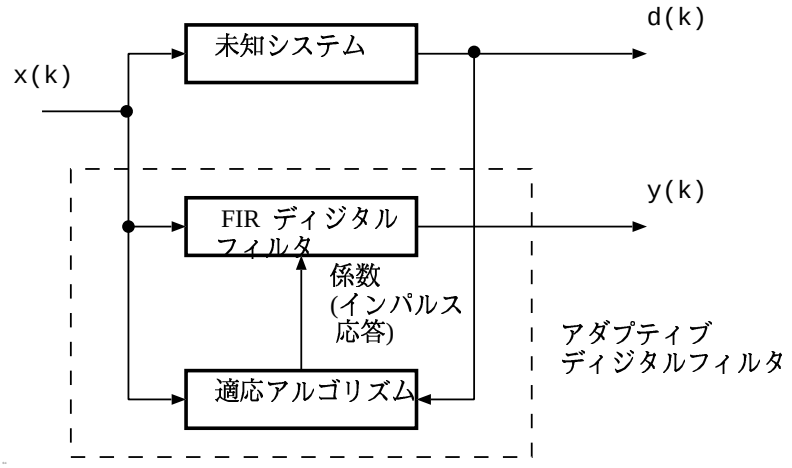

図 2.1　適応アルゴリズム

FIR ディジタルフィルタの入出力関係は，次の差分方程式

$$y = \sum_{i=0}^{N-1} h(i)x(k-i) \tag{2.1}$$

で与えられる．ただし，$x(k)$ は時刻 $kT(T$ はサンプリング周期) におけるフィルタの入力信号，$y(k)$ は時刻 $kT(T$ はサンプリング周期) におけるフィルタの出力信号である．また，$h(i)$ は i 番目のフィルタ係数，N はフィルタ係数の個数である．

次に，インパルス応答 $w(i)(i = 0, 1, 2, \cdots)$ を有する時不変線形システムの入出力信号

$$d(k) = \sum_{i=0}^{\infty} w(i)x(k-i) \tag{2.2}$$

は，このような畳み込み演算を満足させるシステムを未知システムとする．

ここで，式 (2.2) で表されたシステムのインパルス応答長が有限，すなわち，$w(i) = 0$ で $i \geq N$ のとき，FIR ディジタルフィルタとインパルス応答 $w(i)$ を有する先の未知システムは，$w(i) = h(i)(i = 0, 1, \cdots, N-1)$ であれば，常に同じ出力となる．したがって，未知システムのインパルス応答長が有限で，その個数が分かっていると仮定すれば，同じ入力信号に対し完全に等しい出力を与える FIR ディジタルフィルタを得ることができる可能性が存

在する．

次に，未知システムのインパルス応答が無限に続く場合について考える．この場合，同じ入力信号に対し常に等しい出力信号を得る FIR ディジタルフィルタを見つけることはできない．しかし，未知システムのインパルス応答のうち最初の N 個の値が得られれば十分である応用例も多い．

ある条件の下で，評価量

$$J = E[e^2(k)] = E[\{z(k) - y(k)\}^2] = E[\{(d(k) + v(k)) - y(k)\}^2] \tag{2.3}$$

を最小にすることにより，未知システムのインパルス応答の最初の N 個の値を推定できる．ただし，J は評価量，$e(k)$ は出力誤差 (所望信号からアダプティブフィルタの出力信号を引いたもの)，$z(k)$ は観測信号，$v(k)$ は誤差とする．また，N の値を限りなく大きく選べば，未知システムと限りなく等しい入出力関係を持った FIR ディジタルフィルタを得ることができる．このとき，適応フィルタの係数は，入力信号 $x(k)$，フィルタの出力信号 $y(k)$ および未知システムの出力信号 $d(k)$ を使って修正する．この係数修正アルゴリズムを適応アルゴリズムという．

2.3.1　適応アルゴリズムに求められる課題

- 収束速度の高速化

- 実行速度の高速化

- 収束精度の向上

2.3.2　適応アルゴリズムの歴史と種類

1960 年，Widora と whoff は適応スイッチング回路の研究において，LMS アルゴリズムと呼ばれる適応アルゴリズムを開発した．LMS アルゴリズムは，広い意味で，2 乗平均誤差を最急降下法に基づいて最小にする一方式である．演算量が少ないという理由で代表的なアルゴリズムとして利用されている．

1967 年これとは独立に，野田と南雲は学習同定法を発表した．学習同定法は LMS アルゴリズムに比べやや複雑であるが，高速な収束特性を有しており，実用的にも優れた適応アルゴリズムということができる．

これらのアルゴリズムは，与えられた信号の統計的性質が未知 (あるいはほとんど未知) の場合でも，この信号の統計量をもとに生成される Wiener-Hoff の方程式を解くことのできる反復法と見ることができる．また，推定すべきパラメータが時間とともに比較的緩やかに変動しても，パラメータの変化にある程度追従できる特徴がある．実際の応用では，このような状況はむしろ一般的と言えるので，この特徴は重要である．しかし，これらのアルゴリズムは入力信号が有色の場合，収束速度 (特に，推定すべきパラメータへの収束速度) が著しく劣化する欠点のあることが指摘されている．

この他にも，RLS アルゴリズム [1]，BLMS アルゴリズム [1]，跳躍アルゴリズム [1] がある．

2.4　学習同定法

学習同定法は 1967 年野田，南雲らによって発表された．

- 直交射影定理に基づく適応アルゴリズムである．

- LMS アルゴリズムに比べてやや複雑

- 高速な収束性を有する．

- 別名 Normalized LMS アルゴリズム

  – LMS アルゴリズムの係数修正項をフィルタの状態ベクトルノルムで正規化した形

アダプティブフィルタ：適応アルゴリズムを含む FIR ディジタルフィルタ

（注意）

・未知システムと既知システムの次数は等しい $(M = N)$

・観測雑音 $v(t)$ は存在しない $(v(t) = 0)$

時刻 k でアダプティブフィルタの出力 $y(k)$ が未知システムの出力 $d(k)$ に等しい

$$d(k) = (x(k), x(k-1), \cdots, x(k-N+1)) \begin{pmatrix} h(0) \\ h(1) \\ \vdots \\ h(N-1) \end{pmatrix} \tag{2.4}$$

$d(k)$ ： 未知システムにおける出力信号

$x(k)$ ： フィルタの入力信号

$h(k)$ ： k 番目のフィルタ係数

W_N ：未知システムのパラメータ

h_N ：アダプティブフィルタのパラメータ

式 (2.3) を満足する h_N は真値ベクトルを含むいわゆる解集合となる．式 (2.3) を満足する h_N の代表ベクトルを $h_N(k+1)$ とし，そのベクトルを適当に定めた任意の点から解集合

に下ろした垂線の足と考える. 解集合は式 (2.3) からわかるように状態ベクトル $x_N(k)$ に直交しているさらに，W_N はこの解集合に含まれるので，$h_N(k+1)$ はある点から $x_N(k)$ 方向に係数を修正したとき最も W_N に近い点である．このようなことを繰り返して $h_N(k+1)$ を W_N に接近させるためには，適当に定めた点よりも W_N により近い $h_N(k)$ を次の係数修正の初期値とすればよい.

これらのことをまとめると

$$\begin{aligned} h_N(k+1) &= h_N(k) + \{h_N(k+1) - h_N(k)\} \\ &= h_N(k) + \underbrace{\frac{\left\{W_N - h_N(k)^T\right\}\{h_N\{(k+1) - h_N(k)\}\}}{\|h_N(k+1) - h_N(k)\|}}_{\text{修正量}} \\ &\quad \times \underbrace{\frac{h_N(k+1) - h_N(k)}{\|h_N(k+1) - h_N(k)\|}}_{\text{修正方向}} \end{aligned} \tag{2.5}$$

となる．ここで,

$$\frac{h_N(k+1) - h_N(k)}{\|h_N(k+1) - h_N(k)\|} = \frac{x_N(k)}{\|x_N(k)\|} \tag{2.6}$$

$$\{W_N - h_N(k)\}^T x_N(k) = d(k) - y(k) = e(k) \tag{2.7}$$

が成立するので,

$$h_N(k+1) = h_N(k) + \frac{x_N(k)}{\|x_N(k)\|^2} e(k) \tag{2.8}$$

のように変形できる．学習同定法は上式の修正ベクトルにステップゲインを掛け

$$h_N(k+1) = h_N(k) + \alpha \frac{x_N(k)}{\|x_N(k)\|^2} e(k) \tag{2.9}$$

で与えられる.

2.4.1　学習同定法の問題点

学習同定法による係数修正は

$$h_N(k+1) = h_N(k) + \alpha \frac{x_N(k)}{\|x_N(k)\|^2} e(k) \tag{2.10}$$

で与えられる．ただし，$x_N(k)$，$e(k)$ はそれぞれ，

$$x_N(k) = [x(k), x(k-1), \cdots, x(k-N+1)]^T \tag{2.11}$$

$$e(k) = d(k) - x_N(k)^T h_N(k) \tag{2.12}$$

である．ここで，α はステップゲインと呼ばれる量である．このように学習同定法では，1本の入力信号ベクトルを用いて係数を修正するため，有色入力信号に対して著しく収束速度が劣化することが知られている．学習同定法の問題点を解決するために次のような解決策が提案された．

2.4.2　アフィン射影算法

実際の通信系や記録系で使用される多くの信号が有色性であることを考慮すると，この種の信号に対する収束速度の高速化は当然の要求といえる．そこで，雛元らと尾関らはこのような要求を満足する方式として’拡張された学習同定法’，’アフィンの射影算法’ をそれぞれ独立に提案した．これらの方式は，フィルタ係数を修正するために，ある時刻 k における入力信号ベクトル $x_N(k)$ のみならず，適当な数の過去の入力信号ベクトルをも加えた，複数個の入力ベクトルを用いている．これらの係数修正手順は，

$$h_N(k+1) = h_N(k) + A_{r,N}(k)^+ \cdot e_r(k) \tag{2.13}$$

$$\begin{aligned} e_r(k) &= d_r(k) - A_{r,N}(k) \cdot h_N(k) \\ &= A_{r,N}(k)\,[W_N - h_N(k)] \end{aligned} \tag{2.14}$$

で与えられる．ここで，$A_{r,N}(k)^+$ は $A_{r,N}(k)$ の Moore-Penrose 型一般逆行列であり，$A_{r,N}$ は

$$A_{r,N}(k) = [x_N(k-r+1), x_N(k-r+2), \cdots, x_N(k)]^T \tag{2.15}$$

で定義される．

式より

$$h_N(k+1) - h_N(k) = A_{r,N}(k)^+ A_{r,N}(k)\,[W_N - h_N(k)] \tag{2.16}$$

が成り立つ．式の右辺中の行列 $[A_{r,N}^+ A_{r,N}]$ は $A_{r,N}(k)$ の行ベクトルが張る部分空間 $S[A_{r,N}(k)^T]$ へ直行射影行列である．したがって式は $h_N(k+1)$ が W_N を $S[A_{r,N}(k)^T]$ への直行射影することによって得られる点であることを意味している．

このように，係数修正を行うのに複数個の入力信号ベクトルを用いることにより，有色信号入力時にも収束速度の高速化が可能になる．しかしながら, その代償として 1 サンプル当たりの演算量の増加を招き，ハードウェア構成の実現性に困難を生じる．

適応アルゴリズムには少ない演算量で高速な収束速度および処理速度が要求される．したがって，これらを満足するようなアルゴリズムの開発が必要である．それに対する一つの解答を与えてくれるのがブロック適応アルゴリズムである．

2.5　ブロック適応アルゴリズム

ブロック処理はフィルタリングを効率的に行う方式として，Burrus らによって提案され，この概念を適応信号処理に導入したブロック適応アルゴリズムが Clark らによって始められた．ブロック適応アルゴリズムは，入力信号と所望出力信号を有限個ずつブロック化し，そ

のデータブロックごとに 1 回だけ係数修正を行う.

2.5.1　ブロック直交射影アルゴリズム

演算量の軽減を目的としてアフィン射影算法にブロック処理を導入したものが，ブロック直交射影アルゴリズムである.

2.5.2　アフィン射影算法との共通点

アフィン射影算法とブロック直交射影アルゴリズムは，1 回の係数修正に複数の入力信号ベクトルを用い，それぞれのベクトルが張る部分空間への直交射影演算に基礎を置いている．これは，基本的に入力状態行列 (一般的にランク落ちしている場合が多い) を係数行列とする連立 1 次方程式の解の中で，最小ノルム解を求めることに帰着する.

2.5.3　アフィン射影算法との相違点

アフィン射影算法では，現サンプル時刻と次のサンプル時刻での処理の対象となるそれぞれの入力状態行列は，$(r-1)$ 個の入力信号ベクトルを共に有している．一方，BOP アルゴリズムの場合は連続するブロックにおける入力状態行列は，共有している信号ベクトルを持つことはない.

2.5.4　共役勾配法の適用

直交射影演算に基づくアルゴリズムは基本的に，非正則なデータ行列を有する連立方程式を解くことに帰着し，Moore-Penrose 型一般逆行列により表現される．係数行列が対称行

列であるような方程式の一解法に共役勾配法がある．

2.6　共役勾配法

1952 年 M．R．Hestenes と E．Stifel によって発表された．大次元の問題や特殊な分野において利用されていた．1960 年ごろから非線形最適化の問題の解法としても利用されるようになる．反復法でありながら，有限回のステップで厳密解に到達するという性質を持っている．問題点として，少ない反復回数で誤差を最小にする近似解が得られるが，雑音が生じた場合，多くの回数を繰り返しても十分な精度が得られないことがあげられる．

2.6.1　共役勾配法の種類

- HS 版
- 高橋版
- 2 階版
- RG 版
- 単調版
- LI 版
- 田辺版

2.6.2　HS 版計算手順

Hestenes と Stiefel によって発表された，最も基本的なアルゴリズムである．他のアルゴリズムと区別するために，発表者の頭文字をとって，共役勾配法の HS 版とよぶ．この方法は，連立 1 次方程式

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n = b_1 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots a_{nn}x_n = b_n \end{cases} \tag{2.17}$$

を解くための公式で，係数

$$A = \begin{bmatrix} a_{11}a_{12}\cdots a_{1n} \\ \vdots \\ a_{n1}a_{n2}\cdots a_{nn} \end{bmatrix}$$

が対称行列の場合に適用できる．理論的には行列ベクトルで表現する方が扱い易い．そこで，連立 1 次方程式の係数行列，定数項および解ベクトルを

$$A = \begin{bmatrix} a_{11}a_{12}\cdots a_{1n} \\ \vdots \\ a_{n1}a_{n2}\cdots a_{nn} \end{bmatrix}$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

とすれば，方程式は

$$Ax = b$$

と表すことができる．次に，第 k 近似解，第 k 回の修正方向ベクトル，第 k 回の残差ベクトルを，

$$r = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix}$$

とすれば、計算手順は次のようになる

1. 第 0 近似解 (x_0) を適当に選ぶ
2. 第 0 近似解に対する残差を計算する

$$r_0 = b - Ax_0 \tag{2.18}$$

3. 補助変数 (p_0) の初期値を設定する

$$p_0 = r_0 \tag{2.19}$$

$$k = 0$$

4. 第 k 回の修正係数 α_k を

$$\alpha_k = \frac{(r_k, p_k)}{(p_k, Ap_k)} \tag{2.20}$$

によって求める

5. 第 $k+1$ 近似値を

$$x_{k+1} = x_k + \alpha_k p_k \tag{2.21}$$

によって求める

6. 第 $k+1$ 近似値に対する残差を

$$r_{k+1} = r_k - \alpha_k A p_k \tag{2.22}$$

という形で計算する

7. β を次のように計算する

$$\beta = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} \tag{2.23}$$

8. 補助変数の新しい値を

$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{2.24}$$

で計算する

9. 収束判定をし，まだ収束が十分でなければ手順 4 に戻る

2.6.3　高橋版の計算手順

HS 版の式 (2.23)，すなわち

$$p_{k+1} = r_{k+1} + \frac{\|r_{k+1}\|^2}{\|r_k\|^2} \tag{2.25}$$

の両辺を $|r_{k+1}|^2$ で割ると

$$\frac{p_{k+1}}{\|r_{k+1}\|^2} = \frac{r_{k+1}}{\|r_{k+1}\|^2} + \frac{p_k}{\|r_k\|^2} \tag{2.26}$$

という，きれいな形になる．そこで変数 p_k のかわりに

$$q_k = \frac{p_k}{\|r_k\|^2} \tag{2.27}$$

を用いることにすると，上記に相当する計算式は

$$q_{k+1} = q_k + \frac{r_{k+1}}{\|r_{k+1}\|^2} \tag{2.28}$$

となる．これに合わせて，q_k の式に書きなおすと

$$r_{k+1} = r_k - \frac{\|r_k\|^2}{(p_k, Ap_k)} Ap_k$$

$$= r_k - \frac{|r_k|^2}{\left(|r_k|^2 q_k, |r_k|^2 Aq_k\right)} |r_k|^2 Aq_k$$
$$= r_k - \frac{Aq_k}{(q_k, aq_k)} \tag{2.29}$$

となり，式と対称的な形になる．同様にして HS 版の式は

$$x_{k+1} = x_k + \frac{q_k}{(q_k, Aq_k)} \tag{2.30}$$

となる．HS 版よりも，はるかにすっきりすっきりしていて，計算量も少ない．

計算手順は次のようになる．

1. 第 0 近似解 x_0 を適当にとる
2. 第 0 近似解に対する残差を計算する

$$r_0 = b - Ax_0$$

3. q を求める

$$q_0 = r_0/|r_0|^2$$

$$k = 0$$

4. 第 $k+1$ 近似値を

$$x_{k+1} = x_k + \frac{q_k}{(q_k, Aq_k)}$$

5. 第 $k+1$ 近似値に対する残差を

$$r_{k+1} = r_k - \frac{Aq_k}{(q_k, Aq_k)}$$

6. q の更新

$$q_{k+1} = q_k - \frac{r_{k+1}}{|r_{k+1}|^2}$$

7. 収束判定を行い収束が十分でなければ，新 $k = k + 1$ として 4 に戻る

2.6.4　2 階版の計算手順

式 (2.27)(2.28) は，移項すると，

$$q_{k+1} - q_k = \frac{r_{r_{k+1}}}{|r_{k+1}|^2} \tag{2.31}$$

$$r_{k+1} - r_k = -\frac{Aq_k}{(q_k, A_k)} \tag{2.32}$$

という具合に，連立 1 階の差分方程式になる．こういう差分方程式は，一方の変数を消去して，単独 2 階の差分方程式になおすことができる．

まず式 (2.31) を変形して

$$(q_k, Aq_k)(r_{k+1} - r_k) = -Aq_k \tag{2.33}$$

これの添字を一つふやすと

$$(q_{k+1}, Aq_{k+1})(r_{k+2} - r_{k+1}) = -Aq_{k+1} \tag{2.34}$$

両式の差をとると

$$(q_k, Aq_k)(r_{k+1} - r_k) - (q_{k+1}, Aq_{k+1})(r_{k+2} - r_{k+1}) = A(q_{k+1} - q_k)$$

ここで式を用いると

$$= \left(1/\left|r_{k+1}\right|^2\right) Ar_{k+1} \tag{2.35}$$

したがって

$$r_{k+2} = r_{k+1} + \frac{(q_k, Aq_k)}{(q_{k+1}, Aq_{k+1})}\left(r_{k+1} - r_k\right) - \frac{Ar_{k+1}}{(q_{k+1}, Aq_{k+1})\left|r_{k+1}\right|^2} \tag{2.36}$$

を得る．しかし，このままだと，係数に q が入っているから，ぐあいが悪い．そこで，これを q を使わないで計算することを考える．それには,

$$(q_{k+1}, Aq_{k+1}) = \frac{(r_{k+1}, Ar_{k+1})}{\left|r_{k+1}\right|^4} - (q_k, Aq_k) \tag{2.37}$$

という漸化式を用いればよい．その正当性は，式 (2.30) を用いて，次のようにして証明できる．

$$\begin{aligned}(r_{k+1}, Ar_{k+1}) &= \left(\left|r_{k+1}\right|^2 (q_{k+1} - q_k), A\left|r_{k+1}\right|^2 (q_{k+1} - q_k)\right) \\ &= \left|r_{k+1}\right|^4 \{(q_{k+1}, Aq_{k+1}) - 2(q_{k+1}, Aq_{k+1}) + (q_k, Aq_k)\}\end{aligned} \tag{2.38}$$

p_i の直交性 $(p_i, Ap_j) = 0 (i \neq j)$ および式 (2.26) から

$$(q_{k+1}, Aq_k) = 0 \tag{2.39}$$

したがって

$$(r_{k+1}, Ar_{k+1}) = \left|r_{k+1}\right|^4 \{(q_{k+1}, Aq_{k+1}) + (q_k, Aq_k)\} \tag{2.40}$$

これを移項したものが式 (2.36) である．なお，出発時には

$$r_1 = p_1 = \left|r_1\right|^2 q_1 \tag{2.41}$$

であるから

$$(q_1, Aq_1) = (r_1, Ar_1) / |r_1|^4 \tag{2.42}$$

とすればよい．これは式 (2.36) において

$$(q_0, Aq_0) = 0 \tag{2.43}$$

とすることに相当する．こうする方がプログラムは簡単になる．

一方，x の計算をするために式 (2.29) を変形すれば

$$(q_k, Aq_k)(x_{k+1} - x_k) = q_k \tag{2.44}$$

その添字を一つふやすと

$$(q_{k+1}, Aq_{k+1})(x_{k+2} - x_{k+1}) = q_{k+1} \tag{2.45}$$

両式の差を作ると

$$\begin{aligned}(q_{k+1}, Aq_{k+1})(x_{k+2} - x_{k+1}) - (q_k, Aq_k)(x_{k+1} - x_k) &= q_{k+1} - q_k \\ &= r_{k+1} |r_{k+1}|^2 q_k \\ &= r_{k+1} |r_{k+1}|^2 \end{aligned} \tag{2.46}$$

したがって

$$x_{k+2} = x_{k+1} + \frac{(q_k, Aq_k)}{(q_{k+1}, Aq_{k+1})}(x_{k+1} - x_k) + \frac{r_{k+1}}{(q_{k+1}, Aq_{k+1}) |r_{k+1}|^2} \tag{2.47}$$

となる．ここで

$$a_k = (q_k, Aq_k) \tag{2.48}$$

$$b_k = |r_k|^2 \tag{2.49}$$

と書いて計算式をもとめると次のようになる．

1. x_0 を適当にとる
2. 第 0 近似解に対する残差を計算する

$$r_0 = b - Ax_0$$

3. 第 k 回の修正係数 α_k を

$$\alpha = \frac{(r_0, r_0)}{r_0, Ar_0}$$

4. x の更新

$$x_1 = x_0 + \alpha_0 r_0$$

5. r の更新

$$r_1 = r_0 - \alpha A r_0$$

6. a の計算

$$(r_0, Ar_0) \,/\, |r_0|^4$$

7. b の計算

$$b_0 = |r_0|^2$$

$$k = 0$$

8. a の更新

$$a_{k+1} = \frac{(r_{k+1}, Ar_{k+1})}{|r_{k+1}|^4} - a_k$$

9. b の更新

$$b_{k+1} = |r_{k+1}|^2$$

10. x の更新

$$x_{x+2} = \left(1 + \frac{a_k}{a_{k+1}}\right) x_{k+1} - \frac{a_k}{a_{k+1}} x_k + \frac{r_{k+1}}{a_{k+1} b_{k+1}} \tag{2.50}$$

11. r の更新

$$r_{k+2} = \left(1 + \frac{a_k}{a_{k+1}}\right) r_{k+1} - \frac{a_k}{a_{k+1}} x_k + \frac{Ar_{k+1}}{a_{k+1} b_{k+1}} \tag{2.51}$$

12. 収束判定を行い収束が十分でなければ，新 $k = k + 1$ として 8 に戻る

2.6.5　RG 版の計算手順

式 (2.49) および (2.50) を移項すると

$$x_{k+2} - x_{k+1} = \frac{a_k}{a_{k+1}} (x_{k+1} - x_k) + \frac{r_{k+1}}{a_{k+1} b_{k+1}} \tag{2.52}$$

$$r_{k+2} - r_{k+1} = \frac{a_k}{a_{k+1}} (r_{k+1} - r_k) - \frac{Ar_{k+1}}{a_{k+1} b_{k+1}} \tag{2.53}$$

となるから，x_k や r_k のかわりに，その増分

$$\Delta x_k = x_{k+1} - x_k \tag{2.54}$$

$$\Delta r_k = r_{k+1-r_k} \tag{2.55}$$

を主変数にとれば，もっと単純な形になるであろう．ついでに

$$\frac{1}{a_k b_k} = \frac{1}{\varepsilon} \tag{2.56}$$

$$\frac{a_{k-1}}{b_k} = \frac{\varepsilon_{k-1}}{\delta_k} \tag{2.57}$$

すなわち

$$\delta_k = a_k b_k \tag{2.58}$$

$$\varepsilon_{k-1} = a_{k-1} b_k \tag{2.59}$$

と置き，添字を一つずらして書くと，計算手順は次のようになる．

1. x_0 を適当にとる
2. 第 0 近似解に対する残差を計算する

$$r_0 = b - Ax_0$$

3. Δx_0 の定義

$$\Delta x_0 = 0$$

4. Δr_0 の定義

$$\Delta r_0 = 0$$

2.6　共役勾配法

$$k = 0$$

5. ε_{k-1} の定義

$k = 0$ のとき

$$\varepsilon_{k-1} = 0$$

$k > 0$ のとき

$$\varepsilon_{k-1} = \delta_{k-1} \frac{|r_k|^2}{|r_{k-1}|^2}$$

6. δ_k の計算

$$\delta_k = \frac{(r_k, Ar_k)}{|r_k|^2} - \varepsilon_{k-1}$$

7. Δx_k の更新

$$\Delta x_k = (1/\delta)\left(\varepsilon_{k-1}\Delta x_{k-1} + r_k\right)$$

8. Δr_k の更新

$$\Delta r_k = (1/\delta)\left(\varepsilon_{k-1}\Delta r_{k-1} + Ar_k\right)$$

9. x の更新

$$x_{k+1} = x_k + \Delta x_k$$

10. r の更新

$$r_{k+1} = r_k + \Delta r_k$$

11. 収束判定を行い収束が十分でなければ，新 $k = k + 1$ として 5 に戻る

2.6.6　単調版の計算手順

高橋版，2 階版，RG 版は HS 版の変形であり，本質的には同じ公式である．したがって，それらの内のどの公式で計算しても，近似解の列は

$$x_0, x_1, \cdots, x_m$$

は同じものが得られる．また，残差

$$r_0, r_1, \cdots, r_m$$

も共通である．その「大きさ」は，

$$\left(r_k, A^{-1} r_k\right)$$

すなわち，エネルギーを尺度として用いれば，単調に減少するが，

$$(r_k, r_k)$$

すなわち，残差 2 乗和の意味では単調に減少しない．

しかし，できるとこならば，残差 2 乗和 (r_k, r_k) が反復の毎回，単調に減少する方が望ましい．

公式の骨子としてはこれまでの公式と同じ形を使用し，ただ，そこに現れる係数を少し変えることにより，残差 2 乗和が単調に減少する公式を作る．

骨組となる公式としては，この場合，RG 版

$$\Delta x_k = (1/\delta_k)\left(\varepsilon_{k-1} \Delta x_{k-1} + r_k\right) \tag{2.60}$$

$$x_{k+1} = x_k + \Delta x_k \tag{2.61}$$

を用いるのが便利である．この係数 ε_{k-1},δ_k をどのように決めても

$$\Delta r_k = (1/\delta_k)\left(\varepsilon_{k-1}\Delta r_{k-1} - Ar_k\right) \tag{2.62}$$

$$r_{k+1} = r_k + \Delta r_k \tag{2.63}$$

となることは容易にわかる．いうまでもなく r_k は x_k に対する残差

$$r_k = b - Ax_k \tag{2.64}$$

である．これにより

$$r_{k+1} = \left(1 + \frac{\varepsilon_{k-1}}{\delta_k}\right) r_k - \frac{\varepsilon_{k-1}}{\delta_k} r_k - \frac{1}{\delta_k} Ar_k \tag{2.65}$$

を得る．両辺に δ_k を掛けて，移行すれば，次のように書ける．

$$Ar_k = -\delta_k r_{k+1} + \left(\delta_k + \varepsilon_{k-1}\right) r_k - \varepsilon_{k-1} r_{k-1} \tag{2.66}$$

行列 A を対称かつ正定値とする．その固有値を

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

対応する固有ベクトルを長さ 1 に正規化したものを

$$v_1, v_2, \cdots, v_n$$

とする．第 0 近似解に対する残差 r_0 をこの基底で表したものを

$$r_0 = \sum_{i=1}^{n} c_i v_i \tag{2.67}$$

とする．以後の残差は，ε や δ をどのようにとっても，

$$r_k = \varphi_k(A) r_0 \tag{2.68}$$

ただし φ_k は k 次多項式の形で表すことができる．この φ_k が残差多項式である．式 (2.64) に式 (2.67) を入れれば，φ_k に関する漸化式

$$\varphi_{k+1}(t) = \left(1 + \frac{\varepsilon_{k-1}}{\delta_k}\right)\varphi_k(t) - \frac{\varepsilon_{k-1}}{\delta_k}\varphi_{k-1}(t) - \frac{1}{\delta_k} t\varphi_k(t) \tag{2.69}$$

が得られる．

$$\varphi_0(t) + 1 \tag{2.70}$$

$$\varphi_1(t) = 1 - (1/\delta_k)\, t \tag{2.71}$$

で出発して，式で作られる残差多項式はすべての定数項が 1 となる．

定数項が 1 の，あらゆる k 次多項式の中でノルム $\sqrt{<\varphi_k, \varphi_k>}$ が，最小になるものを作るには，$\varphi_0, \varphi)1, \varphi_2, \cdots$ が, 次式

$$f, g = \sum_{i=0}^{n} t_i w_i f(t_i) g(t_i) = \int_a^b f(t) g(t) t\rho(t) dt$$

の意味で直交関数系になるようにすればよい．今の場合，最小化したいものは

$$(r_k, r_k) =< \varphi_k, \varphi_k > \tag{2.72}$$

ただし

$$< f, g >= \sum_{i+1}^{2} c_i^2 f(\lambda_i) g(\lambda_i) \tag{2.73}$$

であるから，残差多項式の方の重み

$$\lambda_1 c_1^2 \lambda_2 c_2^2 \cdots \lambda_n c_n^2$$

に関して直交，すなわち

$i \neq j$ のとき

$$\{\varphi_i, \varphi_j\} = \sum_{j=1}^{n} \lambda_i c_i^2 \varphi_i(\lambda_i)\varphi_j(\lambda_i) = 0 \tag{2.74}$$

となるように定めなければならない．それには漸化式

$$f_{k+1}(t) = tf_k(t) - \frac{< tf_k, f_k >}{< f_k, f_k >} f_k(t) - \frac{< tf_k, f_{k-1} >}{< f_{k-1}, f_{k-1} >} f_{k-1}(t)$$

を使えばよいはずであるが，上式そのままでは式 (2.68) の形に合わない．式は $t\varphi_k(t)$ に係数 $-1/\delta_k$ が掛かっている．よく考えてみると，この項は，いわば「$k+1$ 次式の材料」であるからここに係数が掛かっても，それに合わせて

$$\varphi_{k+1}(t) = \frac{1}{\delta_k} t\varphi_k(t) - \frac{\{-t\varphi_k/\delta_k, \varphi_k\}}{\{\varphi_k, \varphi_k\}} \varphi_k(t) - \frac{\{-t\varphi_k/\delta_k, \varphi_{k-1}\}}{\{\varphi_{k-1}, \varphi_{k-1}\}} \varphi_{k-1}(t) \tag{2.75}$$

というように直交化してやればよい．この式の第 2 項の係数を G_k，第 3 項の係数を h_k とすると

$$\delta_k G_k = \frac{\{t\varphi_k, \varphi_k\}}{\{\varphi_k, \varphi_k\}} = \frac{\sum_{i=1}^{n} \lambda_i^2 c_i^2 \varphi_k(\lambda_i)^2}{\sum_{i=1}^{n} \lambda_i c^2 c_i \varphi_k(\lambda_i)^2} = \frac{(Ar_k, Ar_k)}{(r_k, Ar_k)} \tag{2.76}$$

これと式を比較すると，まず $\varphi_{k-1}(t)$ の係数から

$$\epsilon_{k-1} = -\frac{(Ar_k, At_{k-1})}{(r_{k-1}, Ar_{k-1})} \tag{2.77}$$

また $\varphi_l(t)$ の係数から

$$\delta_k + \varepsilon_{k-1} = \frac{(Ar_k, Ar_k)}{(r_k, Ar_k)} \tag{2.78}$$

$$\delta_k = \frac{(Ar_k, Ar_k)}{(r_k, Ar_k)} - \varepsilon_{k-1} \tag{2.79}$$

これで計算式が確定した．なお,ε_{k-1} の計算式は次のように書くこともできる．

$$\varepsilon_{k-1} = \frac{(r_k, Ar_k)}{(r_{k-1}, Ar_{k-1})}\delta_{k-1} \tag{2.80}$$

実際, 式 (2.73) より，一般に

$$i \neq j$$

ならば

$$(r_i, Ar_j) = 0$$

が成立するから, 式を移項した

$$\delta_k r_{k+1} = (\delta_k + \varepsilon_{k-1})r_k - \varepsilon_{k-1}r_{k-1} + Ar_k \tag{2.81}$$

と Ar_{k+1} の内積を作ると

$$\delta_k(r_{k+1}, Ar_{k+1}) = -(Ar_k, A_{k+1}) \tag{2.82}$$

$$\delta_k = -\frac{(Ar_{k+1}, Ar_k)}{(r_{k+1}, Ar_{k+1})} \tag{2.83}$$

これと式から式を得る

計算手順は次のようになる

2.6　共役勾配法

1. x_0 を適当にとる
2. 第 0 近似解に対する残差を計算する

$$r_0 = b - Ax_0$$

3. Δx_0 の定義

$$\Delta x_0 = 0$$

4. Δr_0 の定義

$$\Delta r_0 = 0$$

$$k = 0$$

5. ε_{k-1} の定義

$k = 0$ のとき

$$\varepsilon_{k-1} = 0$$

$k > 0$ のとき

$$\varepsilon_{k-1} = \frac{(r_k, Ar_k)}{(r_{k-1}, Ar_{k-1})}\delta_{k-1}$$

6. δ_k の計算

$$\delta_k = \frac{(Ar_k, Ar_k)}{(r_k, Ar_k)} - \varepsilon_{k-1}$$

7. Δx_k の更新

$$\Delta x_k = (1/\delta_k)(\varepsilon_{k-1}\Delta x_{k-1} + r_k)$$

8. Δr_k の更新

$$\Delta r_k = (1/\delta_k)(\varepsilon_{k-1}\Delta r_{k-1} - Ar_k)$$

9. x の更新

$$x_{k+1} = x_k + \Delta x_k$$

10. r の更新

$$r_{k+1} = r_k + \Delta r_k$$

11. 収束判定を行い収束が十分でなければ，新 $k = k + 1$ として 5 に戻る．

第 3 章

共役勾配法の選択と評価

3.1　まえがき

適応アルゴリズムで共役勾配法を利用する場合，雑音が存在することを考える必要がある．共役勾配法は雑音の影響を受けることにより，収束速度と収束精度を両立させることが困難になる．このため，適応アルゴリズムで利用する場合の共役勾配法には安定性が必要となる．そのために本章では，前章で述べた HS 版，高橋版，2 階版，RG 版，単調版，LI 版，田辺版，これらの方式から適したものを選ぶ.

3.2　共役勾配法の条件

第 2 章で示した共役勾配法を実行するための条件として次のことがあげられる．

共役勾配法は連立 1 次方程式の係数行列

$$A = \begin{bmatrix} a_{11}a_{12}\cdots a_{1n} \\ \vdots \\ a_{n1}a_{n2}\cdots a_{nn} \end{bmatrix}$$

が対称行列の場合に適用できる．ブロック直交射影アルゴリズムでは，係数行列は入力信号の自己相関行列となるため，この条件を満たしている．適応アルゴリズムでの利用を考え，シミュレーションを行うときの元の数 (適応フィルタのインパルス応答長) は 100 とする．

3.3　反復回数と誤差

共役勾配法の反復回数が適応アルゴリズムの収束値を決める主な要因になる．

共役勾配法では，n 元の方程式なら n 回の反復で厳密解が求まることが知られている．しかしながら，適応アルゴリズムに適用する際には通常，雑音があるため，n 回以下の反復で誤差が最小になる近似解が求まり，それ以上の反復は，誤差の増大を招く．

シミュレーション結果，計算量さらに公式の構造などをもとに，共役勾配法の反復回数を決める．

3.4　共役勾配法の選択

3.4.1　問題点

- 2 解版

  常微分方程式の数値解析における多段法に相当し，そのため出発に手間がかかり，記憶すべきベクトルの本数も多いため，実用上有利とはいえない．

- RG 版

  計算量が他の物よりも多く実用的ではない．更に丸め誤差に弱いという性質を持っている．これは，計算方式に問題があり，情報落ち，桁落ちを引き起こす可能性があり，それにより，一種の誤差が混入されたり，誤差が相対的に拡大されたりするためである．

- 単調版

  大きい固有値に対応する成分を重視するという性格をとっており，そのため絶対誤差の減少が HS 版よりもやや遅くなるという欠点がある．そのため，HS 版に比べ，反復回数が 1 から 2 割多くなる．

- LI 版

  HS 版から比べると反復回数が多くなる．線形反復法を併用すれば良いが，それだと計算時間が多くなり，実用的とはいえない．

- 田辺版

  LI 版を簡単にした物だがこれもやはり LI 版と理論的には変わらないので実用性に欠ける．

HS 版と高橋版は，他種と比較すると手順が簡単で実用的であるため，適応アルゴリズムで用いるのに適している．計算機シミュレーションにより，HS 版と高橋版を比較してみると，HS 版，高橋版のどちらも同じ結果が得られた．大次元の問題になると違いが出るかも知れないが，手順もほとんど変わらないので違いは少ないと思われる．よって一般的に広く用いられている HS 版の反復回数について検討する．

3.5　最適な反復回数

計算機シミュレーション結果から，S/N 比と反復回数について考察する．また，信号の相関係数を 0，0.5，0.9，0.999 にし，違いを見る．ここでの S/N 比とは信号成分と雑音との比である．

- 相関係数を 0 にした場合

  S/N 比が 11.5(dB) より小さい値では，全て 1 回の反復で近似解を得た．また，反復回数が 100 に近くなるほど S/N 比の値も下記のグラフのような波で大きくなっていった．そして S/N 比の値が大きくなればなるほど，反復回数の変動が激しくなる．

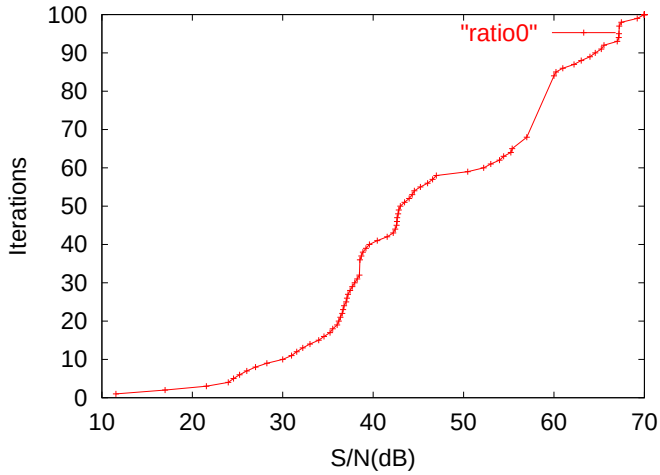

図 3.1　反復回数と S/N 比

- 相関係数を 0.5 にした場合

  S/N 比が 7(dB) より小さい値では，全て 1 回の反復で近似解を得た．やはり，反復回数が 100 に近付くと S/N 比も大きくなった．ただ 1 つだけ異なっているのは，反復回数が 41 から 100 の間において S/N 比の値がほとんど変化していないという事だ．

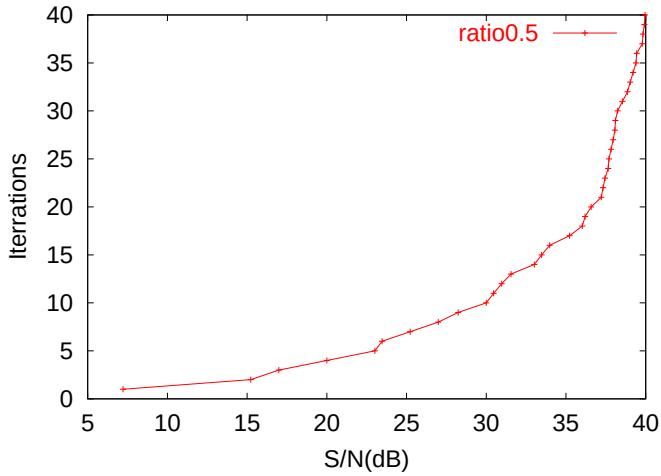

図 3.2　反復回数と S/N 比

- 相関係数を 0.9 にした場合

  S/N 比が-8(dB) より小さい値では、全て 1 回の反復で近似解が得られた．やはり，反復回数が 100 に近付くと S/N 比の値も大きくなった．また反復回数が 90 から 100 の

間は、S/N 比の値は，あまり変化していない.

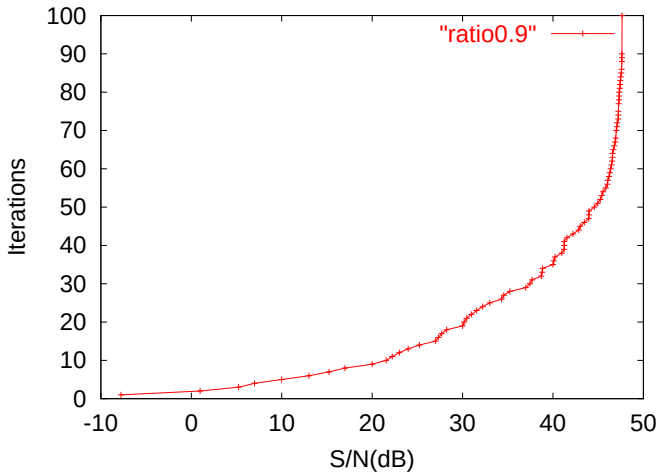

図 3.3　反復回数と S/N 比

- 相関係数を 0.999 にした場合

  S/N 比が-22(dB) より小さい値では，全て 1 回の反復で近似解が得られた．やはり，反復回数が 100 に近付くと S/N 比の値も大きくなった．しかし，ここまで相関係数を大きくすると，S/N 比に対する反復回数が予想に反した結果が出て来る事があった.

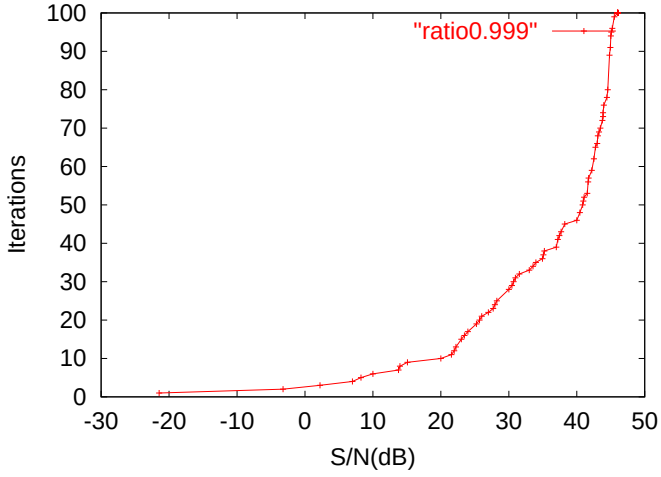

図 3.4　反復回数と S/N 比

表 3.1 に，図 3.1 から 3.4 を数値化した例を示す.

表 3.1　相関係数による違い

相関係数	S/N 比						
	-20(dB)	-10(dB)	0(dB)	10(dB)	20(dB)	30(dB)	40(dB)
0	1	1	1	1	3	10	41
0.5	1	1	1	2	4	10	40
0.9	1	1	1	5	9	19	35
0.999	2	2	2	6	10	28	46

図 3.5 に，S/N 比を 0，10，20，30(dB) にした場合の反復回数と相関係数の関係を示す．

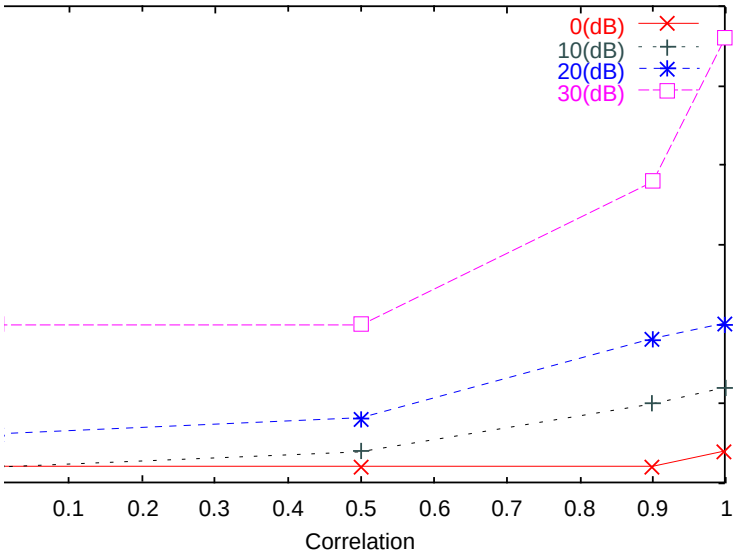

図 3.5　反復回数と相関係数

図または表から，S/N 比が大きくなる程，反復回数は 100 に近くなり，相関を強くする程，S/N 比に対する反復回数が多くなることが分かる．

第 4 章

結論

4.1 結論

本研究では，共役勾配法の問題点である，雑音が存在する場合の推定精度の劣化を軽減させるために，誤差が最小となる最適な反復回数をシミュレーションにより示した．

その傾向として，S/N 比を大きくしていくと，最適な反復回数もそれに応じて多くなり，信号の相関が強くなる程 S/N 比に対する反復回数が多くなったことがあげられる．この結果から，S/N 比と相関係数と反復回数とには規則性があり，S/N 比と相関係数を決定することにより，最適な反復回数を決めることは可能になる．例えば，相関係数を 0.3，S/N 比を 20 とした場合，最適な反復回数は 2 回と決まる．

したがって，信号の相関係数と S/N 比と反復回数との関係式を導くことにより，雑音が生じた場合における推定精度の劣化を大幅に改善することが可能である．また，その結果，実行速度の高速化にもつながる．

また，本研究では，相関係数と S/N 比が分かっている条件下での検討を行ったが，S/N 比がある程度分かれば最適な反復回数の幅を予想できる．その場合においても，n 回の反復を行わなくてよいため，推定精度の向上と実行速度の高速化にもつながる．

4.2 今後の課題

今後の課題として，本研究では，相関係数と S/N 比から最適な反復回数を決めるための指針を示したが，実際にその関係式を導き，実証することが重要になる．さらに今回は，100

4.2　今後の課題

元 1 次連立方程式でのシミュレーションにより，最適な反復回数を示したのであるが，全ての条件に対して，一般化させることが課題となる．

謝辞

本研究を行うにあたり，御指導並びに御助言を頂いた高知工科大学情報システム工学科の毒舌マスター (本当は優しい？) こと福本昌弘助教授に深く感謝致します.

本論文を御審議してくださる高知工科大学情報システム工学科の島村和典教授, 菊池豊助教授，情報システム工学科の先生方に心より感謝致します.

また、日頃からお世話になり，人生を熱く語ってくれた妻鳥助手に深く感謝の意をあらわしたいと思います.

さらに，共に助け合った坂本研究室の石持ちこと登伸一様，I 君のいじめ確信犯こと栃木隆道様，マジックポイントマスターこと亀本学様にも心より感謝致します.

最後に，御協力を賜わりました福本研究室の BOSS こと (本当は優しい) 秋山由佳様，共に苦労した嶋岡哲夫様，また，菊地研究室の BOSS こと舟橋釈仁様に深く感謝致します.

参考文献

[1] 辻井重男, 久保田一, 古川利博, 趙晋輝, 適応信号処理, 昭晃堂,1995.

[2] 戸川隼人, 共役勾配法, 教育出版,1977.

[3] 瀬尾光代,“適応アルゴリズムに適した共役勾配法,” 平成 12 年度高知工科大学学士学位論文,2001.

付録 A

共役勾配法-HS 版-

```
/**共役勾配法-HS 版**/
/**gcc -o -lm hs hs.c**/
#include <stdio.h>
#include <math.h>

void hs();

main()
{
  int i,j,k,n;
  int nor;
  double nsr;
  double d[100],w[100],h[100],cen[100];
  double X[100][100],X0[100][100];

  printf("n = ");
  scanf("%d", &n);
  printf("repeat = ");
  scanf("%d", &nor);
  printf("N/S = ");
```

```
  scanf("%lf", &nsr);

  for(i=0;i<n;i++)
    {
      w[i] = sqrt( 1.0/(double)n );
      X0[n-1][i] = (double)rand()/2147483648.0;
    }

  for (i=n-2; i>=0; i--)
    {
      for(j=0; j<n-1; j++)
X0[i][j] = X0[i+1][j+1];

      X0[i][n-1] = (double)rand()/2147483648.0;
    }

  for (i=0; i<n; i++)
    {
    for(j=0; j<n; j++)
      {
      for( k=0, X[i][j]=0.0; k<n; k++ )
X[i][j] += X0[i][k] * X0[j][k];
    }
  }

  for (i=0; i<n; i++)
```

```
    h[i] = 0.0;

  for (i=0; i<n; i++)
    {
    d[i] = 0.0;

    for (j=0; j<n; j++)
      {
      d[i] += X[i][j] * w[j];
    }
    d[i] += (double)rand()/2147483648.0 * nsr; /* 雑音付加 */
  }

  hs( n, nor, X, h, d ,w, cen );

  for(i=0; i<nor; i++)
      printf(" %4d %g \n", i+1, cen[i]);
}




/*   HS type Conjugate Gradient Method   */

void hs(n,nor,a,x,b,w,cen)
     int n,nor;
```

```
     double a[][100],x[],b[],w[],cen[];
{
  /*変数の定義*/
  int i,j,repeat;/**計算用変数**/
  double d,e,f,h,l,m,o,k;/**計算用変数**/
  double alpha,beta;/**修正係数**/
  double g[100],r[100],p[100];/**残差, 修正ベクトル**/

  /*r=b-ax(第 0 近似解に対する残差) の計算*/
  for(i=0; i<n; i++)
    {
      d=b[i];
      for(j=0; j<n; j++)
{
d -= a[i][j]*x[j];
      }
      /*p(補助変数) の定義*/
      r[i]=d;
      p[i]=r[i];
    }

  /*繰り返し*/
  for(repeat=0;repeat<nor;repeat++)
    {
      /*alpha(修正係数)=((r,p)=e/(p,Ap)=f) の計算*/
      /*g(Ap) の計算*/
```

```
      m=0;
      e=0;
      f=0;
      for(i=0; i<n; i++)
{
  g[i]=0;
  for(j=0; j<n; j++)
    {
      g[i]+=a[i][j]*p[j];
    }

  /*e(rp の内積)&f(p,Ap) の計算*/
  e+=r[i]*p[i];
  f+=p[i]*g[i];
  /*m(r) の内積*/
  m+=r[i]*r[i];
}


      /*alpha(修正係数) の計算*/
      alpha=e/f;
      /*x(近似値) の更新*/
      for(i=0; i<n; i++)
{
  h=x[i];
  h+=alpha*p[i];
```

```
  x[i]=h;
  /*r(近似値に関する残差) の更新*/
  k=r[i];
  k-=alpha*g[i];
  r[i]=k;
}

      /* 誤差ノルム (|| h_N(i) - w_N ||^2) の計算 */
      for(i=0, cen[repeat]=0.0; i<n; i++)
cen[repeat] += (x[i]-w[i])*(x[i]-w[i]);

      /*beta の計算*/
      /*l の計算*/
      l=0;
      for(i=0; i<n; i++)
{
  l+=r[i]*r[i];
}
      beta=l/m;

      /*p(補助変数) の更新*/
      for(i=0; i<n; i++)
{
  o=r[i];
  o+=beta*p[i];
  p[i]=o;
```

```
}


    }
}
```

付録 B

共役勾配法-高橋版-

```
/**共役勾配法-高橋版**/
/**gcc -o -lm tk takahashi.c**/
#include <stdio.h>
#include <math.h>

void cgm_tk();

main()
{
  int i,j,k,n;
  int nor;
  double nsr;
  double d[100],w[100],h[100],cen[100];
  double X[100][100],X0[100][100];

  printf("n = ");
  scanf("%d", &n);
  printf("repeat = ");
  scanf("%d", &nor);
  printf("N/S = ");
```

```
  scanf("%lf", &nsr);

  for(i=0;i<n;i++)
    {
      w[i] = sqrt( 1.0/(double)n );
      X0[n-1][i] = (double)rand()/2147483648.0;
    }

  for (i=n-2; i>=0; i--)
    {
      for(j=0; j<n-1; j++)
X0[i][j] = X0[i+1][j+1];
        X0[i][n-1] = (double)rand()/2147483648.0;
    }

  for (i=0; i<n; i++)
    {
    for(j=0; j<n; j++)
      {
      for( k=0, X[i][j]=0.0; k<n; k++ )
X[i][j] += X0[i][k] * X0[j][k];
    }
  }

  for (i=0; i<n; i++)
    h[i] = 0.0;
```

```
  for (i=0; i<n; i++)
    {
    d[i] = 0.0;

    for (j=0; j<n; j++)
      {
      d[i] += X[i][j] * w[j];
    }
    d[i] += (double)rand()/2147483648.0 * nsr; /* 雑音付加 */
  }

  cgm_tk( n, nor, X, h, d ,w, cen );

  for(i=0; i<nor; i++)
      printf(" %4d %g \n", i+1, cen[i]);
}

/*   takahashi type Conjugate Gradient Method   */

void cgm_tk(n,nor,A,x,B,w,cen)
     int n,nor;
     double A[][100],x[],B[],w[],cen[];
{

/*変数の定義*/
```

```
  int i,j,repeat;/**計算用変数**/
  double d,e,rr;/**計算用変数**/
  double g[100],r[100],q[100];/**残差ベクトル, 修正ベクトル**/

  /*r=b-ax(第 0 近似解に対する残差) の計算*/
  for(i=0; i<n; i++)
    {
      d=B[i];
      for(j=0; j<n; j++)
{
  d -= A[i][j]*x[j];
}
      r[i]=d;
    }

  /*q[i]=(r/r*r) の計算*/
  /*rr=(r*r) の計算*/
  rr=0;
  for(i=0;i<n;i++)
    {
      rr+=r[i]*r[i];
    }
  /*q[i] の設定*/
  for(i=0;i<n;i++)
    {
      q[i]=r[i]/rr;
```

```
    }

  /*繰り返し*/
  for(repeat=0;repeat<nor;repeat++)
    {
      /*x(近似解) の更新*/
      /*g[i]=(Aq) の計算*/
      for(i=0; i<n; i++)
{
  g[i]=0;
  for(j=0; j<n; j++)
    {
      g[i]+=A[i][j]*q[j];
    }
}


      /*e=(q,Aq) の内積の計算*/
      e=0;
      for(i=0;i<n;i++)
{
  e+=q[i]*g[i];
}
      /*x(近似解) の計算*/
      for(i=0;i<n;i++)
{
  x[i]+=q[i]/e;
```

```
}

      /* 誤差ノルム (|| h_N(i) - w_N ||^2) の計算 */
      for(i=0, cen[repeat]=0.0; i<n; i++)
cen[repeat] += (x[i]-w[i])*(x[i]-w[i]);

      /*r(近似解に関する残差) の更新*/
      for(i=0;i<n;i++)
{
  r[i]-=g[i]/e;
}

      /*rr の更新*/
      rr=0;
      for(i=0;i<n;i++)
{
  rr+=r[i]*r[i];
}

      /*q[i] の更新*/
      for(i=0;i<n;i++)
{
  q[i]+=r[i]/rr;
}
    }
}
```