# 分散演算を用いた適応フィルタの
# 構成に関する研究

**2003**

岩手大学大学院工学研究科
電子情報工学専攻

高橋 強

# 目 次

# 図目次

# 表 目 次

# 第1章　緒言

ディジタル信号処理は，音声，画像，通信，制御，医学，経済学など幅広い分野で活用されており，今日の情報通信システムにおける基盤技術となっている．

これまでの信号処理では，フィルタ設計，高速フーリエ変換など，処理方法が時間的に変化しない固定的な手法が主流であった[1]．しかし，信号やシステムの特性が時間とともに変化する場合は，固定的な処理方法では対応することはできない．そこで，信号やシステムの特性変化に応じて処理の方法を時間的に変化させようとするのが適応信号処理(Adaptive Signal Processing)である．適応信号処理は，1960年のB.Widrowの先駆的な研究に始まり[2]，その後，逐次形のLMS(Least Mean Square)アルゴリズム，Normalized LMSアルゴリズム，RLS(Recursive Least Square)アルゴリズムなどへと発展した．このアルゴリズム研究は，システム同定やスペクトル解析と密接な関係を保ちながら，信号処理の一つの中心的な課題となっている．また，応用分野も，エコー・キャンセラ，アクティブ・ノイズ・キャンセラ，アダプティブ・イコライザなど幅広く応用されている．

ディジタル信号処理の実現形態は，半導体技術の進展とともに飛躍的な進歩を遂げている．1960年代には，汎用計算機を用いたバッチ処理によるシミュレーションが主流であり，脳波・地震波の解析，声道シミュレーションなどに応用された．また，Kalmanの提案したKalman Filterは宇宙工学の分野における衛星の軌道計算や経済学の分野で応用された[3, 4]．1970年代前期には，LSI技術の進歩に伴って専用ハードウエアによる実現が可能になり，各種計測器・通信機器のディジタル化が始まった．その後も半導体技術は急速に発展を続け，近年，高精度かつコンパクトなリアルタイム信号処理システムが比較的安価に構築できるようになってきた．そのひとつが，1980年当初に出現したディジタルシグナルプロセッサ(Digital Signal Processor, DSP)と呼ばれる信号処理専用プロセッサである．DSPでは，信号処理の基本演算である積和演算を効率よく実行するための構成と専用の乗算器を有すること，プログラムバスとデータバスそしてプログラムメモリとデータメモリを分離した”ハーバード・アーキテクチャ”を採用していること，パイプライン処理により処

理速度を向上させることなどにより信号処理アルゴリズムを高速に実行することが可能である [5]．このように，DSP ではプログラム論理による汎用のリアルタイム処理が可能である．これにより，適応信号処理の適用範囲が急激に拡大し，その重要性がますます増大するきっかけとなった．しかし，汎用性を重視するアーキテクチャのために，ハードウエア規模や消費電力が大きくなる傾向がある．次に，CPLD(Complex Programmable Logic Device) やFPGA(Field Programmable Gate Array) といったプログラム可能なロジック IC が出現している．これらデバイスの動作速度，規模，消費電力などの性能向上は目覚しく，専用ハードウエアを比較的容易に実現することが可能になってきている．

最近の高速ディジタル通信網の整備と高速移動体通信の進展を背景に，適応フィルタに対する高速化，高精度化，小規模化，低消費電力化への要求が高まっている．特に，携帯性やバッテリでの動作可能時間が重要になる移動形携帯端末においては，ハードウエア規模や消費電力を小さく抑えることが望まれる．

適応フィルタを実現する際に要求される性能としては，

- 高速な収束速度を有すること
- 良好な推定精度を有すること
- 出力滞在時間 (Latency) が短いこと
- 高速なサンプリングレートを有すること
- ハードウエア規模が小さいこと
- 消費電力が小さいこと

などが挙げられる．一般に，これらの項目は適応アルゴリズムの性能とハードウエアの構成法，すなわちアーキテクチャの双方に依存している．例えば，高速な収束速度を有する RLS アルゴリズムは演算量が多いためにサンプリングレートが低く抑えられ，ハードウエア規模や消費電力が大きくなるなどのトレードオフが存在する．したがって，これらを「同

時に満たす」ことは非常に困難であるため，高性能な適応アルゴリズムや効果的なアーキテクチャが望まれている．

適応フィルタの効果的な実現に対する研究としては，乗算器を用いたパイプライン処理による構成法が提案され [6, 7, 8, 9, 10, 11, 12, 13, 14]，RLS アルゴリズム，Delayed LMS (DLMS) アルゴリズム，Pipelined LMS アルゴリズム，Kalman Filter などが用いられている．RLS に基づく構成法では，乗算，除算や平方根のような複雑な演算を多く必要とするため，非常に膨大なハードウェア量が必要になる．また，DLMS に基づく構成法では各タップ毎にレジスタを挿入してパイプライン化し，クリティカルパスを小さな値にすることによってスループットの向上を図っている．そのため，タップ数に相当する乗算器が必要となり，タップ数の増加に伴い出力滞在時間が増加する．出力滞在時間の増加は，非定常環境でのアルゴリズムの追従性の劣化を引き起こし [10]，また，フィードバック制御に用いる場合にはシステムの安定性を損なう原因になる．いずれの構成法においても多くの乗算器が必要になるが，乗算器の消費電力とハードウェア量は加算器などと比較しても非常に大きいため，高次での実現を考慮した場合，膨大な消費電力とハードウェア量が必要になる．

これに対して低消費電力，小規模ハードウェアを実現するために乗算器を使用しない，いわゆるマルチプライヤレスな構成法が提案されている．これまで，分散演算はベクトルの内積演算を効率よく計算する手法[1] として知られており，FIR フィルタや DCT(Discrete Cosine Transform) などに用いられてきた [22, 17, 18]．ところが，分散演算は係数が時間的に変化する適応フィルタ[2]においても有効な演算手法となる．Cowan らは，分散演算 (Distributed Arithmetic) を LMS アルゴリズムに適用した分散演算形 LMS アルゴリズムを導出して適応フィルタに応用した [15, 16, 17]．しかし，このアルゴリズムには実現において大きな問題点があることが明らかとなった [19, 20, 21]．それは，入力信号の符号化にオフセットバイ

[1]重み付けられた部分積の和として内積を求める．部分積はあらかじめ求められるので，これらを ROM に格納しておき，変数ベクトルをアドレス信号として ROM から部分積を読み出してシフト加算することにより内積を求める基本的な構成が提案されている．

[2]適応フィルタに応用する場合には，部分積を適応アルゴリズムにより逐次的に更新する必要があるため，部分積を RAM に格納する基本的な構成が提案されている．

ナリ形式[3]，そして内部演算の符号化に2の補数形式を用いたために推定精度が大幅に劣化することと，アルゴリズムと実現におけるビットの取扱いが異なることによって収束速度が極端に劣化することである．定係数の分散演算においては，入力信号にオフセットバイナリ形式を用いた場合にのみ関数空間[4]に奇対称性が現われることが知られている [22]．この性質を利用すると関数空間の容量を1/2に削減することが可能になるため，ハードウエア規模と消費電力を低減することが可能になる．また，アルゴリズムの導出過程において現われるマトリクスの乗算はオフセットバイナリ形式を用いると容易に簡略化できる [15]．つまり，Cowan らは適応関数空間[5]の奇対称性を利用するためとアルゴリズムの簡略化を容易にするためにオフセットバイナリ形式を用いたのである [15, 16, 17]．また，分散演算形LMS適応フィルタに対するVLSIアーキテクチャは考慮されておらず，その高速性についても検討が行われてこなかった．

本論文では，入力信号の符号化に2の補数形式を用いた高性能な分散演算形LMS適応フィルタを提案する [21, 23, 24]．まず，2の補数形式を用いた分散演算形LMSアルゴリズムを導出し，計算機シミュレーションによって推定精度および収束速度が大幅に改善されることを示す．次いで，高次の分散演算では適応関数空間の容量が膨大になるためにハードウエアと消費電力が増加し，収束速度が劣化するという問題がある．これを解決するために，マルチメモリブロック構造 [25] を適用したマルチメモリブロック構造の分散演算形LMSアルゴリズムを導出する [21, 23, 24]．最後に，従来法では検討されなかったVLSIアーキテクチャ，および，その高速性を考慮した構成法を提案する．これらより，本提案法は高次においても高速性と極めて短い出力滞在時間を維持した上で，良好な収束特性，低消費電力，小規模ハードウェアを実現できることを明らかにする．

次に，ハーフメモリアルゴリズムを適用することにより，さらに高性能化を進める．これまで，奇対称の性質はオフセットバイナリ形式に固有の性質であると考えられてきたが，

[3]通常の2進数表現における "0" に "-1" を，"1" に "1" を割り当てる方式．

[4]部分積の集合を関数空間と呼ぶ．

[5]適応フィルタにおける関数空間を適応関数空間と呼ぶ．

2の補数形式を用いた分散演算型LMS適応フィルタにおいても近似的な奇対称性[6]が成立することを初めて示す[26, 27]．そして，この性質を利用したハーフメモリアルゴリズムに基づく分散演算型LMS適応フィルタの高性能アーキテクチャを提案する[26, 27]．そして，収束特性およびVLSI評価を行い，ハーフメモリアルゴリズムを適用した構成法は消費電力およびハードウェア量がほぼ等しい条件の下で約2倍の収束速度を達成することを明らかにする．また，同等の収束速度を有する条件の下で消費電力とハードウェア量を削減可能であることも示す．

これまで，分散演算を用いたLMSアルゴリズムの収束条件に関する理論的検討はその難解さから行われてこなかった．本研究では，分散演算形LMSアルゴリズムの収束条件について初めて理論解析を行う[19, 20, 28, 29]．収束条件式を導くために，まず，更新式を全適応関数空間に拡張し，これをLMSアルゴリズムと比較することによって分散演算形LMSアルゴリズムに対する新たな入力信号ベクトル[7]を定義する．次いで，収束条件式を導出し，収束条件は拡張された入力信号ベクトルの自己相関行列の固有値に依存することを示す．さらに，従来法と提案法の固有値を解析的に評価して従来法の問題点を明らかにするとともに提案法の有効性を示す．

最後に，分散演算形LMSアルゴリズムを高速化するための検討を行う．これまで，LMSアルゴリズムの高速アルゴリズムであるブロックLMS(Block implementation of LMS, BLMS)アルゴリズムが提案されている[30, 31]．通常のLMSアルゴリズムはサンプリング時刻ごとに処理を実行するが，BLMSアルゴリズムでは$L$サンプリング時刻ごとに$L$個の入力信号を並列に処理する．これにより，BLMSアルゴリズムは1サンプルあたりの処理時間を$1/L$に減少させることが可能である．Clarkらは，演算量の削減を目的にアルゴリズムを時間領域から周波数領域に変換して用いた[30]．しかし，高速フーリエ変換を用いて入力信号やパラメータを周波数領域に変換するため，タップ数が増加するに伴い出力滞在時間が急激に増加するという問題点がある．また，これまで効果的な構成法については検討され

[6]本文では近似的に成立する奇対称性を準奇対称性と呼ぶことにする．

[7]拡張された入力信号ベクトル，あるいは拡張入力信号ベクトルと呼ぶことにする．

ていない．

BLMSアルゴリズムはLMSアルゴリズムを並列に実行するため，その時間領域アルゴリズムは高度な並列性を本質的に有している[32]．そこで，本研究では時間領域のブロックLMSアルゴリズムに対して分散演算を適用した分散演算形ブロックLMSアルゴリズム(BDAアルゴリズム)と，そのマルチメモリブロック構造に対するアルゴリズム(MBDAアルゴリズム)を導出する[33, 34]．しかし，導出したアルゴリズムは更新動作が複雑であるため，パイプライン処理にはあまり適していない．そこで，パイプライン処理に適した更新アルゴリズムとしてプライオリティ・アップデートを提案する．これにより，更新動作がスムーズに実行されることになり，高速なサンプリングレートと短い出力滞在時間を達成可能である．さらに，MBDA適応フィルタの効果的なVLSIアーキテクチャを提案し，サンプリング周波数と出力滞在時間を評価する．これらより，提案するMBDA適応フィルタは良好な収束特性，高速なサンプリングレート，そして短い出力滞在時間を有することを明らかにする．

本論文は，これらをとりまとめたものであり，以下に示す7章から構成される．

第1章は，緒言であり，本論文の背景と目的，および，概要について述べている．

第2章では，適応フィルタに対する代表的な問題であるシステム同定（パラメータ推定）について述べ，いくつかの具体的な応用例について説明している．次いで，システム同定における問題設定を行い，評価量について定義している．最後に，本論文で用いる適応アルゴリズムであるLMSアルゴリズムについて述べている．

第3章では，分散演算形LMS適応フィルタについて述べている．まず，2の補数形式に基づく分散演算形LMSアルゴリズムを導出するとともに，従来のオフセットバイナリ形式に基づく分散演算形LMSアルゴリズムの導出過程も同時に示し，従来法の問題点について言及している．さらに，提案する分散演算形LMSアルゴリズムにマルチメモリブロック構成を適用している．そして，計算機シミュレーションにより収束特性を評価している．最後に，提案するアルゴリズムを用いた適応フィルタの効果的な構成法を示し，VLSI設計システムPARTHENON[35]を用いてVLSI評価を行っている．

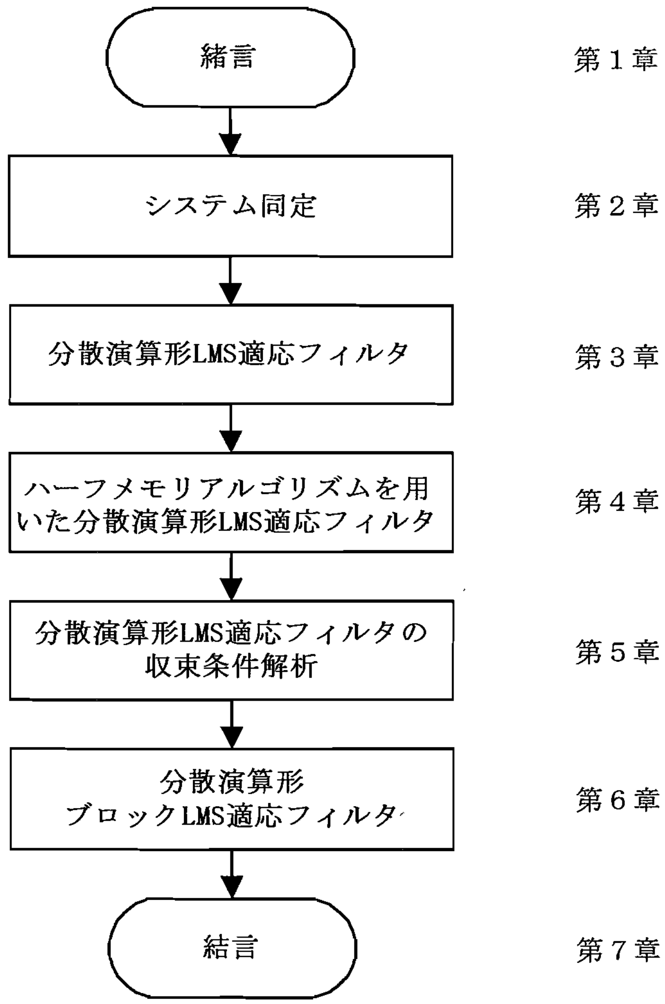
第4章では，ハーフメモリアルゴリズムについて述べている．まず，提案する2の補数形式を用いた分散演算形LMSアルゴリズムにおいても近似的な奇対称性が現れることを解析的に示す．そして，この性質を利用したハーフメモリアルゴリズムを導出している．次いで，計算機シミュレーションにより収束特性を評価している．最後に，ハーフメモリアルゴリズムに基づいた分散演算形LMSアルゴリズムを用いた適応フィルタの効果的な構成法を示し，VLSI設計評価システムPARTHENONを用いてVLSI評価を行っている．

第5章では，分散演算形LMSアルゴリズムの収束条件を解析的に示している．まず，収束条件式を導くために更新式を全適応関数空間に拡張する．そして，この全空間に拡張された更新式を用いて適応関数空間推定誤差の更新式，すなわち収束条件式を定義する．これより，収束条件を適応関数空間推定誤差が時刻の経過とともに減少するための条件として導いている．従来のアルゴリズムの収束特性が大幅に劣化すること，そして提案法の収束速度が良好であることを示すために，自己相関行列の固有値を解析的に求めている．

第6章では，分散演算形LMS適応フィルタの高速化を図る目的で，ブロックLMSアルゴリズムに分散演算を適用した分散演算形ブロックLMSアルゴリズムを導出している．導出したアルゴリズムは，その複雑さからパイプライン処理が困難であるため，パイプライン処理に適した新たな更新方法であるプライオリティ・アップデートを提案している．そして，マルチメモリブロック構造を適用したMBDAアルゴリズムを提案している．これに対して，計算機シミュレーションにより収束特性を評価し，さらに効果的なVLSIアーキテクチャを提案している．

第7章は結言である．

以上，本論文の企図するところを概説した．


緒言
第１章
システム同定
第２章
分散演算形LMS適応フィルタ
第３章
ハーフメモリアルゴリズムを用いた分散演算形LMS適応フィルタ
第４章
分散演算形LMS適応フィルタの収束条件解析
第５章
分散演算形ブロックLMS適応フィルタ
第６章
結言
第７章

# 第2章　システム同定

未知システム(Unknown System)の入力信号と出力信号から，未知システムのパラメータを逐次的に推定する，いわゆる学習機能を持ったディジタルフィルタは適応ディジタルフィルタ（Adaptive Digital Filter）[8]と呼ばれている[39, 43, 5, 47, 48, 49]．システムのパラメータとしては，インパルス応答が用いられることが多く，そのため，ディジタルフィルタとしてはインパルス応答をフィルタ係数に持つFIR（Finite Impulse Response）フィルタを用いることが多い．

本章では，パラメータ推定と適応的にフィルタ係数を逐次更新するための適応アルゴリズムについて述べる．

## 2.1　パラメータ推定

ブラックボックスで表現された未知システムについて，その入出力データから未知のシステムの構造とパラメータを推定することをシステム同定と呼んでいる．ここでは，未知システムの構造をFIR形と仮定し，そのパラメータを推定する問題について考える．

FIRフィルタの入出力関係は，次に示す畳み込み演算（Convolution)で表現される．

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

ただし，$k$はサンプリング時刻を表し，時間$t[sec]$との関係は次式で表される．

$$t = kT \tag{2}$$

ここで，$T[sec]$はサンプリング周期を表す．$x(k)$，$y(k)$ はそれぞれ時刻$k$におけるフィルタの入力信号，出力信号である．また，$h_i$は$i$番目のフィルタ係数，$N$はタップ係数の個数である．時刻$k$におけるタップ係数は，次のようにベクトル表現される．

$$\boldsymbol{h}(k) = [h_0, h_1, \cdots, h_{N-1}]^T \tag{3}$$

[8]本論文では，適応ディジタルフィルタを便宜上，適応フィルタと呼ぶことにする．

さて，インパルス応答$w_i$を有する線形時不変システム[9]を考える．このシステムの出力信号は畳み込み演算により次のように表される．

$$d(k) = \sum_{i=0}^{M-1} w_i x(k-i) \tag{4}$$

ただし，$d(k)$はシステムの出力信号である．また，タップ係数はベクトル表示により次のように表される．

$$\boldsymbol{w}(k) = [w_0, w_1, \cdots, w_{N-1}]^T \tag{5}$$

ここでは，このシステムを未知システムと呼び，出力信号を所望信号（Desired Signal）と呼ぶことにする．さて，式(1)と式(4)について，インパルス応答長に着目して議論を進める．

[$M = N$**の場合**]

このとき，

$$w_i = h_i, \quad i = 0, 1, \cdots, N-1 \tag{6}$$

であれば，両者の出力は一致する．したがって，未知システムのインパルス応答が有限でその個数が既知であると仮定すれば，同一の入力信号に対して完全に等しい出力信号を与えるFIRフィルタを構成することができる可能性を有する[47, 50]．

しかし，上に述べた仮定は一般には成立しないことが多い．

[$M = \infty$**の場合**]

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フィルタを構成することは不可能である．しかし，未知システムのインパルス応答のうちN個の値が得られれば十分である応用例も多い．この目的のためには，ある評価量を設定し，ある条件下でこの評価量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個の値を推定することができる[47, 50]．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

[9]システムの入出力関係が線形で，伝達関数が時間変化しないシステム．

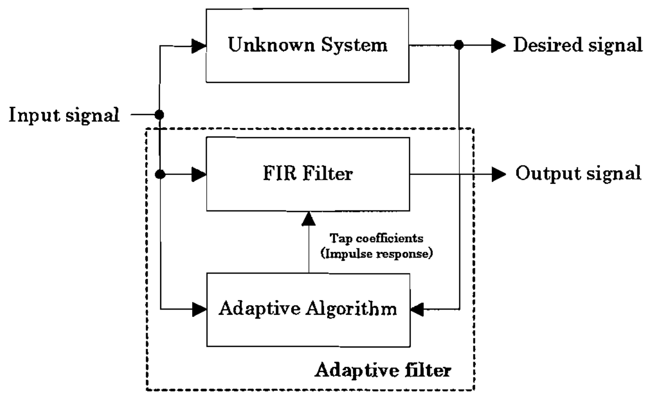



図 1: Parameter estimation.

等しい入出力関係を持ったFIRフィルタを得ることが可能である．本節では，未知システムの構造をFIRフィルタと仮定したが，この様に考えればこの仮定に対する妥当性も理解できる．

パラメータ推定概念を 図 1 に示す．

$N$タップFIR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$d(k)$を用いて修正される．係数修正アルゴリズムは適応アルゴリズム (Adaptive Algorithm)，得られたフィルタ係数は推定値 (Estimate)，また，適応アルゴリズムを含むFIRフィルタは適応フィルタと呼ばれる．未知システムの出力には観測雑音 (Observation Noise) が加えられる．以後， 図 1 と同じことを示すために， 図 2 を用いることにする．

適応フィルタは，入力信号$x(k)$と次式で定義される出力誤差信号

$$e(k) \triangleq (k) - y(k) \tag{7}$$

を用いて，$e(k)$に対する評価量を最小にするようにフィルタ係数を更新する．

## 2.2 パラメータ推定技術の応用例

ここでは，パラメータ推定技術の応用例を示し，この技術の必要性について述べる [47, 5].

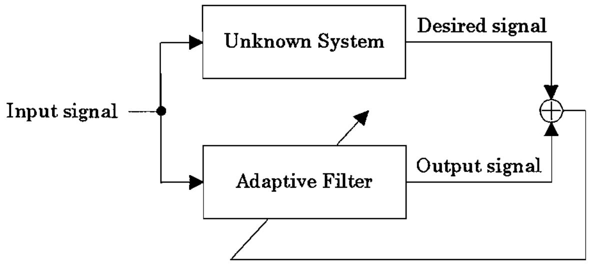



図 2: Model of system identification.

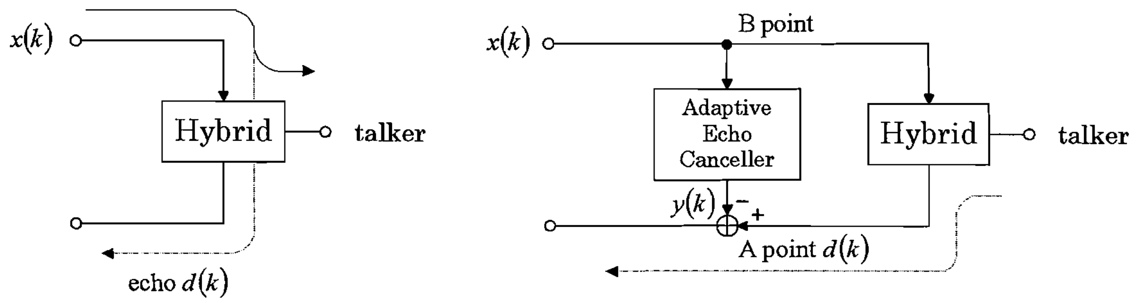
### 2.2.1 適応エコーキャンセラ

国際電話回線などのように長距離電話回線を利用する際に，自分の話声が数秒後に自分の受話器から聞こえることにより，会話が非常に困難になることがある．自分の受話器から聞こえる適当に遅延された自分の話声は，エコー (Echo) と呼ばれる．エコーの発生原因としては，2 線式と 4 線式回線の接続部に設けられたハイブリッドと呼ばれる信号分離回路のインピーダンス不整合などによる [36, 37].

この問題を解決する一手段として，パラメータ推定の概念を用いて，エコーパスを適応的に推定することにより擬似エコーを発生させる．そして，本来のエコーからこの擬似エコーを減算することによりエコー障害を抑圧しようとする方式が提案された．この方式が適応エコーキャンセラ (Adaptive echo canceller) である．概念を 図 3 に示す．エコーパスが推定できれば，擬似エコー $y(k)$ は $d(k)$ に一致し，エコーが消去される．

### 2.2.2 適応ノイズキャンセラ

所望信号が雑音に埋もれている場合に，観測信号の信号対雑音比を向上させることを考える．もし，雑音だけを純粋に取り出すことが可能であれば，この問題は次に述べる適応ノイズキャンセラの概念により効率的に解くことが可能である．図 4 に適応ノイズキャンセラの原理を示す．ここで，A 点，B 点はそれぞれ主入力端子，参照入力端子と呼ばれている．主入力端子は通常の入力端子であり，所望信号 $s(k)$ と雑音 $d(k)$ の和の信号が入力され




(a) echo path　　(b) adaptive echo canceller

図 3: Adaptive echo canceller.

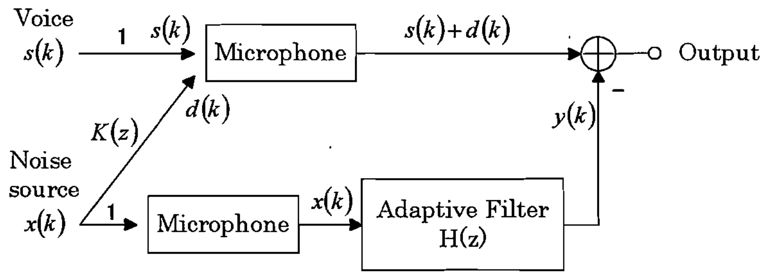



図 4: Adaptive noise canceller.

る．所望信号が参照入力端子に混入しない場合，参照入力端子の信号を適当に線形処理して主入力端子の信号から減算することにより，所望信号を抽出することが可能である．この例では，適応フィルタの伝達関数がパスの伝達関数と等しくなるとき，適応フィルタの出力と主入力端子における雑音信号は等しくなる．このとき，これらを減算することにより雑音を消去することができる．

### 2.2.3 適応等化器

位相変調を用いたディジタル伝送における適応等化器の概念を 図 5 に示す．送信信号 $d(k)$ は，伝送路を通って受信点に到達するまでに，歪や雑音の影響を受ける．このような状況下で歪を補償するのが適応等化器である．動作の概要を説明する．通信に先立って，送

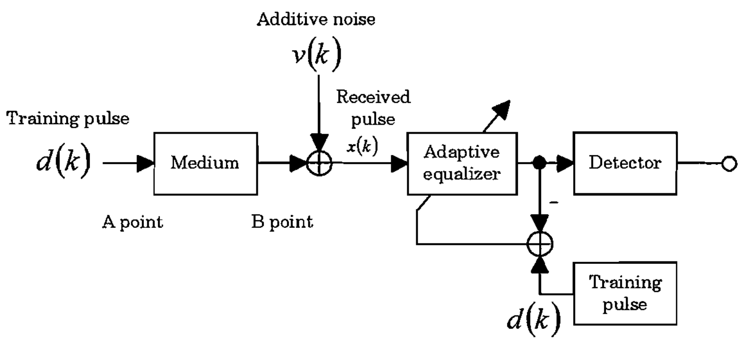



図 5: Adaptive equaliser.

信側と受信側で既知のパルス・パターンが伝送される．この期間はトレーニング期間と呼ばれる．トレーニング期間では，同期の確立，適応等化器の学習のために用いられる．この期間においては，送られているパルス・パターンが既知であるため，適応等化器出力と既知パターンとの誤差を求め，この誤差信号を用いて適応等化器を学習させる．そして，既知のパルス・パターンの後に続く通信データを等化処理する．

適応等化器の適応等化器の問題は，伝送路の伝達関数そのものを推定するのではなく逆伝達関数を推定することである．この場合，未知システムが最小位相推移系であれば，適応等化器は効率よく未知システムのインパルス応答を推定できるが，非最小位相推移系の場合には等化性能が急激に劣化する．

### 2.2.4 アクティブノイズコントローラ

図 6 にアクティブ・ノイズ・コントローラの概念を示す．雑音源 $x(k)$ の影響で，マイク周辺に雑音 $d(k)$ が存在している状況を考える．別の場所にスピーカを用意し，付加音をマイクのある位置に放射し雑音 $d(k)$ を軽減する装置をアクティブ・ノイズ・コントローラと呼ぶ．この例では,

$$H(z) = -K(z)/C(z) \tag{8}$$

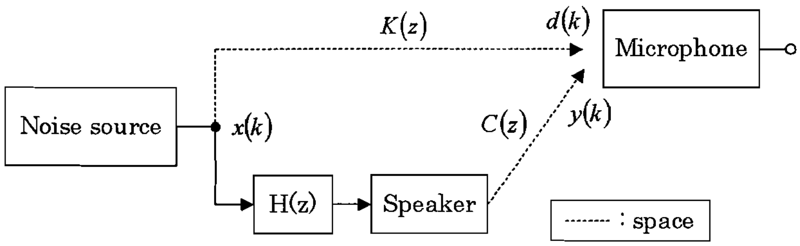



図 6: Conception of active noise controller.

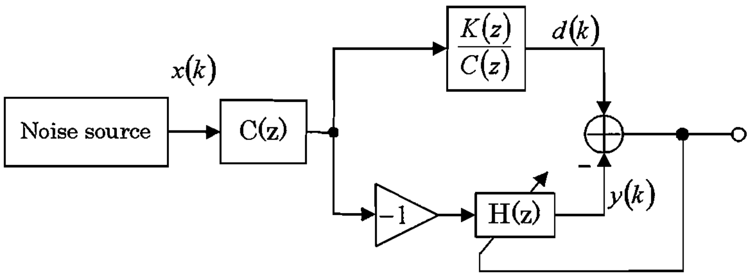



図 7: Active noise controller.

となれば，スピーカから放射された音は，マイクの付近で$-d(k)$となり，騒音を除去可能である．次に，伝達関数$H(z)$の推定について述べる．

図 6 の概念図を 図 7 のように考えると，この問題はパラメータ推定問題となる．ただし，スピーカからマイクの間の伝達関数はあらかじめ求めておく必要がある．

以上，簡単に適応フィルタの代表的な応用例を述べた．エコーの消去，雑音の消去の基本は波形そのものの推定ではなく，その波形の経路に存在する何らかの線形システムの伝達関数（インパルス応答）の推定であることに，これらの応用例における共通点がある．言い換えれば，未知システムのパラメータを推定することにより不要な波形の消去が可能になるということであり，これら応用例を中心として，近年，適応アルゴリズムに関する関心が急速に高まっている．

## 2.3 問題の設定と評価量

適応アルゴリズムに要求される性能としては，

- 高速な収束速度
- 少ない演算量
- 小規模ハードウエア

などがある．収束速度の高速化については，推定すべきパラメータへの収束速度と評価量の収束値への収束速度の２点が考えられる．可能な限り短い語長でプロセッサを構成する試みは，実行速度の高速化とハードウエアの小規模化を同時に満足するが，実行速度の高速化をハードウエア面積に依存させる並列処理では，これらは相反する要求となる．また，一般的に収束速度の向上は，ハードウエアの複雑化につながる．この様に適応アルゴリズムの課題は，相互に関連し，全てを満足するアルゴリズムを開発することは非常に困難である．さらに，動作の安定性についても考慮する必要がある．

本節では，適応アルゴリズムを導出するために必要な問題設定と評価量について述べる[47]．誤差の定義は次の３種類が考えられる．

(1) 出力誤差

最もよく用いられる誤差の定義で，図８(a)に示したように，観測信号$z(k)$と推定システムの出力$y(k)$の差を誤差信号$e(k)$とする．

(2) 入力誤差

図８(b)に示したように，入力信号$x(k)$と観測信号$z(k)$を入力とする推定システムの出力$y(k)$の差を誤差信号$e(k)$とする．このとき，推定システムは未知システムの逆システムとなる．

(3) 一般化誤差

図８(c)に示すように出力誤差と入力誤差を組み合わせて定義される．推定システム１と推定システム２の伝達関数を，それぞれ$G_1(z)$，$G_2(z)$として次のように表す．

$$G_1(z) = a(0) + a(1)z^{-1} + \cdots + a(m)z^{-m} \tag{9}$$

$$G_2(Z) = 1 + b(1)z^{-1} + \cdots + b(n)z^{-n} \tag{10}$$

一般化誤差 $e(k)$ は

$$\begin{aligned} e(k) &= z(k) + b(1)z(k-1) + \cdots + b(n)z(k-n) \\ &\quad -a(0)x(k) - a(1)x(k-1) - \cdots - a(m)x(k-m) \end{aligned} \tag{11}$$

となる.

次に，問題設定について述べる．なお，ここでは次式で表される未知システムを推定対象として考える.

$$d(k) = \sum_{i=0}^{M-1} w_i x(k-i) \tag{12}$$

ここで，$k$は時刻，$w_i, i = 0, 1, \cdots, M-1$は推定対象となる未知システムのインパルス応答，$M$はインパルス応答の個数，$x(k)$は時刻$k$における入力信号である．未知システムのインパルス応答系列をベクトル表現し

$$\boldsymbol{w} = [w_0, w_1, \cdots, w_{M-1}]^T \tag{13}$$

と表す．ここで，添え字$T$はベクトルの転置を表す.

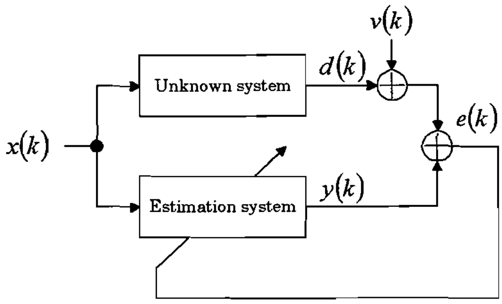
これに対して，次の FIR フィルタを考える.

$$y(k) = \sum_{i=0}^{N-1} h_i(k) x(k-i) \tag{14}$$

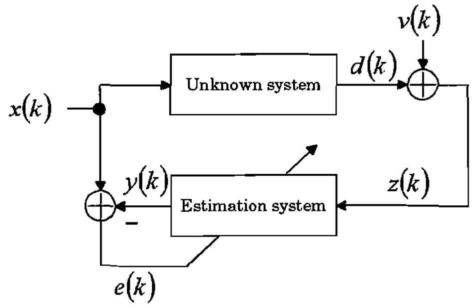
ここで，

$$\boldsymbol{h}(k) = [h_0(k), h_1(k), \cdots, h_{N-1}(k)]^T \tag{15}$$

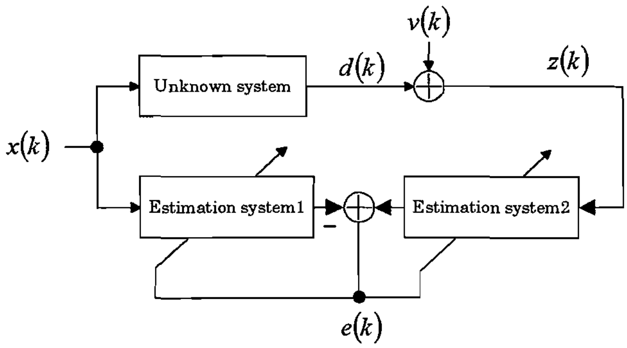
である．このとき，式(13)と式(15)の間に距離に関する評価量が定義され，かつこの評価量を最小にするように FIR フィルタの係数$h_i$を修正できる場合，このフィルタを適応フィルタ，得られた係数ベクトル$\boldsymbol{h}(k)$は推定値と呼ばれる.




(a) Output error




(b) Input error




(c) Generalized error

図 8: Definition of error signals.

さて，問題は評価量$J$をどのように選ぶかである．ここで，$J$として 図 8 (a) に示した出力誤差の2乗平均値を考えることにする． 図 8 (a) は誤差信号の2乗平均値が最小になるように，適応フィルタの係数が更新されることを示している． 図 8 (a) において，評価量$J$は

$$\begin{aligned} J &= E[e^2(k)] \\ &= E[z(k) - y(k)^2] \\ &= E[(d(k) + v(k)) - y(k)^2] \end{aligned} \tag{16}$$

で与えられる．パラメータ推定の目的からすれば，評価量$J$は未知システムのパラメータと推定システムのパラメータを直接評価することが望ましいが，未知システムのパラメータは文字どおり未知であるため，出力誤差の2乗平均値を評価量として用いることが多い．

これより，本論文における適応アルゴリズムの評価は，入力信号サンプルに対するアルゴリズムの収束を評価する観点から，サンプリング時刻$k$に対する出力誤差信号の2乗平均値 (Mean-Square-Error, MSE) を用いて行うことにする [5, 47]．したがって，収束特性を評価する際に用いるアルゴリズムの繰り返し回数は，適応動作が開始した後の入力信号サンプル数である．

## 2.4 最急降下法とLMSアルゴリズム

適応フィルタの係数を更新するために用いる適応アルゴリズムとしては，LMSアルゴリズム，学習同定法，逐次最小2乗法などさまざまなアルゴリズムが提案されている [48]．ここでは，本論文で用いるLMSアルゴリズムについて述べ，他のアルゴリズムについては参考文献を参照されたい．

まず，LMSアルゴリズムの基礎となっている最急降下法について述べる．任意の係数ベクトル$\boldsymbol{h}(k)$における勾配ベクトル$\boldsymbol{G}(\boldsymbol{h}(k))$を

$$\boldsymbol{G}(\boldsymbol{h}(k)) \triangleq 2\,\boldsymbol{R}\boldsymbol{h}(k) - 2\,\boldsymbol{v} \tag{17}$$

と定義する．ここで，

$$\boldsymbol{R} = E[\,\boldsymbol{x}(k)\,\boldsymbol{x}^T(k)] \tag{18}$$

$$\boldsymbol{v} = E[\boldsymbol{x}(k)z(k)] \tag{19}$$

$$z(k) = d(k) + v(k) \tag{20}$$

である．なお，$v(k)$ は観測雑音を表す．式 (17) はパラメータ $\boldsymbol{h}(k)$ の 2 次形式になっており，評価量 $J$を最小にする $\boldsymbol{h}(k)$ はただひとつ存在する．図 9 はこの様子を $N=2$ の場合について説明したものである． 図 9 に示した曲線は，係数 $h_0(k)$，$h_1(k)$ の変化に対して $J$の値の等しい集合である．また，$\boldsymbol{G}(\boldsymbol{h}(\boldsymbol{k}))$ は任意の係数 $\boldsymbol{h}(\boldsymbol{k})$における勾配に等しく，等高線上の法線方向に一致している．したがって，任意の点 $\boldsymbol{h}(0)$ を初期値として，$\boldsymbol{h}(0)$ を $-\boldsymbol{G}(\boldsymbol{h}(0))$ 方向に適当に移動することによって $\boldsymbol{h}(1)$ における $J$の値を $\boldsymbol{h}(0)$ における $J$の値よりも小さくすることができる．ただし，$\boldsymbol{h}(j)$ は $\boldsymbol{h}$の $j$番目の修正値である．これを繰り返せば，$\boldsymbol{h}_j$ は $\boldsymbol{h}_{opt}$に限りなく近づく．以上のアルゴリズムをまとめると，

$$\boldsymbol{h}(j) = \boldsymbol{h}(j-1) - 0.5\mu(j)\,\boldsymbol{G}\{\boldsymbol{h}(j-1)\}, \quad j = 1, 2, \cdots \tag{21}$$

となる．式 (21) は最急降下法，$\mu(j)$ はステップサイズパラメータと呼ばれる．また，ステップサイズパラメータの係数 0.5 は，後の式変形を簡略化するためのものであり，特に意味はない．構成を 図 10 に示す．

次に，Widrow と Hoff により提案された LMS について述べる [2, 48]．最急降下法では，パラメータ推定に先立って入力出力信号の統計的性質が既知であることが前提となってい

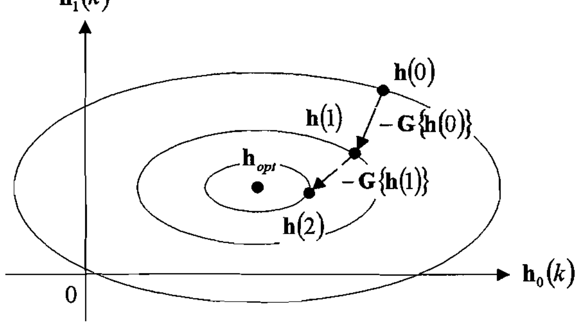



図 9: Contour line of mean square error.

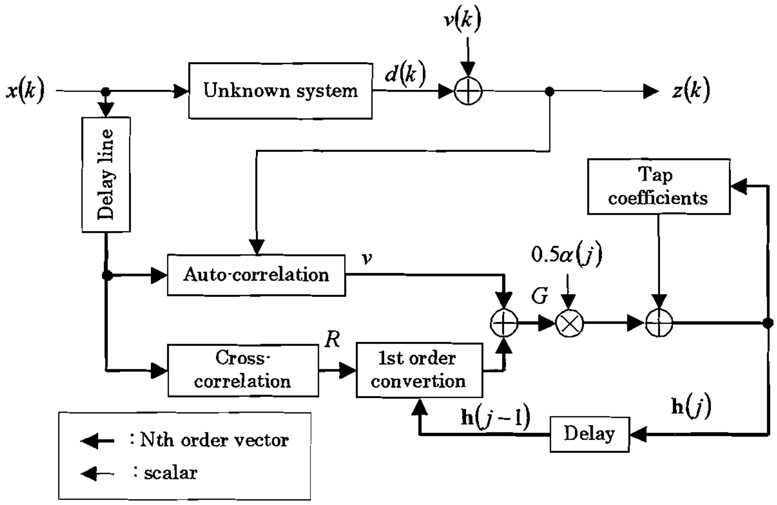



図 10: Structure of Steepest descent method.

る．しかし，実際の応用ではこれらの統計量を計算するだけの時間を許されない場合も多い．そこで，式(21)から平均操作を省略すると，式(21)は

$$\boldsymbol{h}_j = \boldsymbol{h}_{j-1} - 0.5\mu(j)\,\boldsymbol{G}\{\boldsymbol{h}_{j-1}\}, \quad j = 1, 2, \cdots \tag{22}$$

と変形される．LMSアルゴリズムは，式(22)において$j = k+1$，および$\mu(j) = \mu$とすれば得られる．すなわち，

$$\begin{aligned} \boldsymbol{h}(k+1) &= \boldsymbol{h}(k) - \mu\{y(k) - z(k)\}\,\boldsymbol{x}(k) & (23) \\ &= \boldsymbol{h}(k) + \mu e(k)\,\boldsymbol{x}(k) & (24) \end{aligned}$$

である．このように，時刻$k$におけるデータから次の時刻に使用する推定ベクトルが繰り返し得られる．また，ステップサイズパラメータが次式を満たすとき評価量$J$が零に近づくことが知られている．

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{25}$$

ただし，$\lambda_{max}$は入力信号ベクトル$\boldsymbol{x}(k)$の自己相関行列の最大固有値である．

# 第3章　分散演算形LMS適応フィルタ

## 3.1　はじめに

これまで，分散演算は定係数ベクトルの内積演算を効率的に行うための計算手法として用いられてきたが，係数が時変となる適応信号処理においても有効な演算手法となる．

本章では，まず，2の補数形式に基づいた定係数の分散演算の原理について述べ，次にLMSアルゴリズムに分散演算を適用して分散演算型LMSアルゴリズム(以下，DAアルゴリズムと呼ぶ)を導出する．さらに，従来法において入力信号に特殊な符号化を用いたことによって生じる問題点について述べる．

## 3.2　分散演算型LMSアルゴリズム

### 3.2.1　提案する2の補数形式に基づいた定係数の分散演算

分散演算は，内積演算をテーブルルックアップによって実現する計算手法である [18].

ここで，項数$N$の定係数ベクトル$\boldsymbol{a} = (a_1, \ldots, a_N)$と変数ベクトル$\boldsymbol{v} = (v_1, \ldots, v_N)$との内積演算

$$y = \boldsymbol{a}\,\boldsymbol{v} = \sum_{i=1}^{N} a_i v_i \tag{26}$$

を考える．但し，$-1 \leq v_i < 1$で，$v_i$は$B$ビットの固定小数点形の2の補数表示である．つまり，

$$v_i = -v_i^0 + \sum_{k=1}^{B-1} v_i^k 2^{-k} \tag{27}$$

と表される．ここで，$v_i^k$は$v_i$の$k$ビット目の値で'0'または'1'である．式(27)を式(26)に代入すれば，内積演算$\boldsymbol{av}$は次式で示される．

$$\boldsymbol{y} = -\Phi(v_1^0, \ldots, v_N^0) + \sum_{k=1}^{B-1} \Phi(v_1^k, \ldots, v_N^k) 2^{-k} \tag{28}$$

ここで，$\Phi$は引数$(v_1^k, \ldots, v_N^k)$に対する部分積を返す関数であり

$$\Phi(v_1^k, \ldots, v_N^k) = \sum_{i=1}^{N} a_i v_i^k \tag{29}$$

$$
\begin{array}{rl}
 & \Phi\left(v_1^{B-1},\cdots,v_N^{B-1}\right) \\
 & \quad \cdot^{\cdot^{\cdot}} \\
 & \Phi\left(v_1^{1},\cdots,v_N^{1}\right) \\
+) & -\Phi\left(v_1^{0},\cdots,v_N^{0}\right) \\
\hline
 & y
\end{array}
$$

図 11: Fundamentals of distributed arithmetic.

である．

定係数の分散演算を用いて内積を求めるためには，アドレス $(v_1^k,...,v_N^k)$ に対応した部分積を格納した関数Φを用意しておき，アドレスを指定して関数Φから部分積を読み出し，シフト操作と加算を実行する．これを $B$回行うことによって内積を求めることができる．分散演算の原理図を図11に示し，分散演算のアーキテクチャを図12に示す．関数Φのテーブルは $(v_1^k,\ldots,v_N^k)$ をアドレスとするROMで実現でき，右シフト加算は加算器とレジスタによって実現できる．

以上より，原理的には処理時間が項数 $N$に依存せず，語長 $B$にのみに依存する構成が実現可能となる．また出力滞在時間[10]も項数に依存せず，一定の値を保つことができる．さらに，乗算器を使用せずに内積演算の結果を求めることができるため，大幅に消費電力およびハードウェア量を削減することが可能となる．このようにハードウェアの効率性を考慮した場合，分散演算は非常に有効な手法となる．

### 3.2.2　2の補数形式を用いた分散演算形LMSアルゴリズムの導出

$N$次の入力信号ベクトルを

$$\boldsymbol{S}(k)=[s(k),s(k-1),\ldots,s(k-N+1)]^T \tag{30}$$

[10]本論文では，出力滞在時間を「あるサンプルがシステムに入力されてから，そのサンプルに対する演算結果が出力されるまでの時間」と定義する．

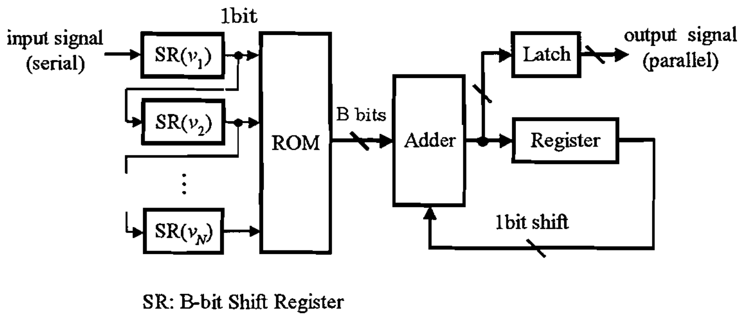



図 12: Basic structure of distributed arithmetic for constant coefficients.

タップ数$N$の係数ベクトルを

$$\boldsymbol{W}(k) = [w_0(k), w_1(k), \ldots, w_{N-1}(k)]^T \tag{31}$$

とする．ここで，$k$はサンプリング時刻を表し，時間$t$との関係はサンプリング周期を$T$とすると次のように表される．

$$t = kT \tag{32}$$

入力信号ベクトル$\boldsymbol{S}(k)$を

$$\boldsymbol{S}(k) \triangleq \boldsymbol{A}(k)\,\boldsymbol{F} \tag{33}$$

と定義すると，フィルタの出力計算式は次式で表される．

$$y(k) = \boldsymbol{S}^T(k)\,\boldsymbol{W}(k) = \boldsymbol{F}^T\,\boldsymbol{A}^T(k)\,\boldsymbol{W}(k) \tag{34}$$

式(33)，(34)において，$\boldsymbol{A}(k)$は$\boldsymbol{S}(k)$のビットパターンを要素とするアドレスマトリクスで

$$\boldsymbol{A}(k) = \begin{bmatrix} b_0(k) & b_0(k-1) & \cdots & b_0(k-N+1) \\ b_1(k) & b_1(k-1) & \cdots & b_1(k-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ b_{B-1}(k) & b_{B-1}(k-1) & \cdots & b_{B-1}(k-N+1) \end{bmatrix}^T \tag{35}$$

$\boldsymbol{F}$は2の補数形式におけるビット重みを要素とするスケーリングベクトルであり，

$$\boldsymbol{F} = [-2^0, 2^{-1}, \ldots, 2^{-(B-1)}]^T \tag{36}$$

と表される．ここで，$b_i(k)$ は時刻 $k$ における入力信号 $s(k)$ の $i$ ビット目の値である．式 (34) において

$$\boldsymbol{P}(k) \triangleq \boldsymbol{A}^T(k)\,\boldsymbol{W}(k) \tag{37}$$

と定義することによって，出力計算式は

$$y(k) = \boldsymbol{F}^T\,\boldsymbol{P}(k) \tag{38}$$

となる．ここで，関数 $\boldsymbol{P}(k)$ は

$$\boldsymbol{P}(k) = [p_0(k), p_1(k), \ldots, p_{B-1}(k)]^T \tag{39}$$

である．

次に，2 の補数形式を用いた分散演算形 LMS アルゴリズムを導出する．LMS アルゴリズムは次式で表される．

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\mu e(k)\,\boldsymbol{S}(k) \tag{40}$$

また，誤差信号 $e(k)$ は次式で求められる．

$$e(k) = d(k) - y(k) \tag{41}$$

ここで，$d(k)$ は未知システムの所望信号を表す．式 (40) の両辺に左から $\boldsymbol{A}^T(k)$ を掛けることにより，式 (42) が得られる．

$$\boldsymbol{A}^T(k)\,\boldsymbol{W}(k+1) = \boldsymbol{A}^T(k)\{\boldsymbol{W}(k) + 2\mu e(k)\,\boldsymbol{A}(k)\,\boldsymbol{F}\} \tag{42}$$

さらに，式 (42) において

$$\boldsymbol{P}(k+1) \triangleq \boldsymbol{A}^T(k)\,\boldsymbol{W}(k+1) \tag{43}$$

$$\boldsymbol{P}(k) \triangleq \boldsymbol{A}^T(k)\,\boldsymbol{W}(k) \tag{44}$$

と定義することによって，次の更新式が得られる．

$$\boldsymbol{P}(k+1) = \boldsymbol{P}(k) + 2\mu e(k)\,\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)\,\boldsymbol{F} \tag{45}$$

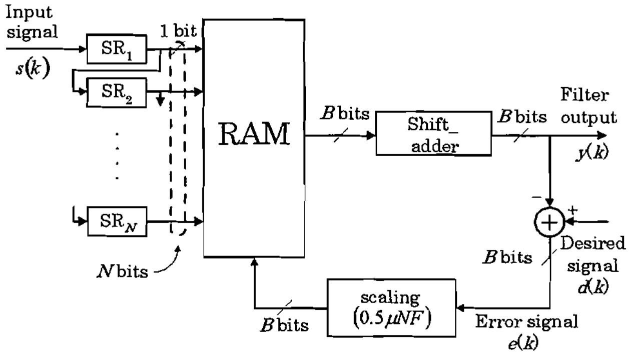



図 13: Block diagram of DA adaptive filter.

なお，関数 $\boldsymbol{P}(k)$ は時刻 $k$において更新対象になる部分積を要素に持つ $B$次ベクトルである．この関数 $\boldsymbol{P}(k)$ を適応関数空間 (Adaptive Function Space) と呼ぶことにする．また，$N$次アドレスベクトルに対する部分積数は $2^N$個であるが，この $2^N$個の全ての要素から構成される適応関数空間を全適応関数空間 (Whole Adaptive Function Space)WAFS と呼ぶことにする．定係数ベクトルの分散演算における関数空間はあらかじめ決定されるのに対し，適応アルゴリズムにおける分散演算では，適応関数空間を逐次的に更新して評価関数を最小にする最適な適応関数空間を推定する．以上で導出した分散演算形 LMS アルゴリズムを適用したフィルタを分散演算形 LMS 適応フィルタ（以下，DA 適応フィルタ）と呼ぶ．DA 適応フィルタの更新動作は通常の LMS 適応フィルタとは大きく異なる．LMS アルゴリズムは，式 (40) のように係数 $\boldsymbol{W}(k)$ を直接更新するのに対し，分散演算形 LMS アルゴリズムでは，式 (45) のように，フィルタ出力を計算するために用いた部分積に対して更新動作を実行する．なお，これらの部分積はアドレスマトリクス $\boldsymbol{A}^T(k)$ の行ベクトル $(b_i(k),\ldots,b_i(k-N+1))$ によって指定される．

さて，式 (45) では，更新値を求める際に $\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$ という行列の乗算を行う必要があるため，リアルタイム処理が困難になるという問題点がある．この問題を解決するために，スケーリングベクトル $\boldsymbol{F}$を含めた $\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)\,\boldsymbol{F}$を次のように置き換えられることを初めて示して適用した．この導出については付録 A に示す．

$$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)\,\boldsymbol{F} = 0.25N\,\boldsymbol{F} \tag{46}$$

これを式(45)に適用すると更新式は

$$\boldsymbol{P}(k+1) = \boldsymbol{P}(k) + 0.5\mu N e(k)\,\boldsymbol{F} \tag{47}$$

となる．この更新式を用いた場合にも，多くの計算機シミュレーションにより収束することが確認されている．また，提案法と同様に $\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$ を対角化した従来の分散演算形適応フィルタは，実際にディジタル電話回線の適応キャンセラ等に用いられている[15].

また，$N$および$\mu$を2のべき乗で考えた場合，更新値を誤差$e(k)$に対するシフト操作のみで求めることが可能となる．これにより，ハードウエア規模や演算時間の比較的大きな乗算器が不要になるため，高速なリアルタイム処理，小規模ハードウエア，そして低消費電力が実現できる．DA適応フィルタの構成を図13に示す．フィルタ出力を求める部分は，分散演算の基本構成をほぼそのまま用いることができ，適応関数空間はRAM（Random Access Memory）によって実現することができる．

### 3.2.3　従来法の問題点

ここでは，従来法と提案法におけるアルゴリズムの相違を示し，従来法の問題点について述べる．

従来法では，入力信号$s(k)$の符号化に特殊な符号化を用いていた．それは，各入力信号$s(k)$をオフセットバイナリ形式で符号化し，その各ビットにおいて論理値 “0” が値 “-1” を，論理値 “1” が値 “1” を有する符号化形式である．そのとき，入力信号ベクトルは以下の式で表される．

$$\boldsymbol{S}(k) = \boldsymbol{A}'(k)\,\boldsymbol{F}' \tag{48}$$

上式において，アドレスマトリクス$\boldsymbol{A}'(k)$は，

$$\boldsymbol{A}'(k) = \begin{bmatrix} b'_1(k) & b'_1(k-1) & \cdots & b'_1(k-N+1) \\ b'_2(k) & b'_2(k-1) & \cdots & b'_2(k-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ b'_B(k) & b'_B(k-1) & \cdots & b'_B(k-N+1) \end{bmatrix}^T \tag{49}$$

スケーリングベクトル $\boldsymbol{F}'$は，

$$\boldsymbol{F}' = [2^{-1}, 2^{-2}, \ldots, 2^{-B}]^T \tag{50}$$

である．

次に，LMSアルゴリズムからDA適応フィルタの更新式を導出する過程を示す．式(40)の両辺に左から $\boldsymbol{A}'^T(k)$ を掛けることにより，式(51)が得られる．

$$\boldsymbol{P}'(k+1) = \boldsymbol{P}'(k) + 2\mu e'(k)\, \boldsymbol{A}'^T(k)\, \boldsymbol{A}'(k)\, \boldsymbol{F}' \tag{51}$$

ここで，式(51)の $\boldsymbol{A}'^T(k)\, \boldsymbol{A}'(k)$ は入力信号の統計的性質により次のようになる．入力信号が平均0の白色信号であり，入力信号を構成する各ビットが互いに無相関である条件のもとで期待値をとると，$\boldsymbol{A}'^T(k)\, \boldsymbol{A}'(k)$ は容易に対角化され，

$$\boldsymbol{A}'^T(k)\, \boldsymbol{A}'(k) = \begin{bmatrix} N & 0 & \cdots & 0 \\ 0 & N & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & N \end{bmatrix} \tag{52}$$

となる．この結果を式(51)に適用すると，更新式は

$$\boldsymbol{P}'(k+1) = \boldsymbol{P}'(k) + 2\mu N e'(k)\, \boldsymbol{F}' \tag{53}$$

となる．

従来法においてこの符号化を用いた理由は，式(52)で示した $\boldsymbol{A}'^T(k)\, \boldsymbol{A}'(k)$ の対角化を容易に行い，式展開を簡単化するためである．これに対して，2の補数形式を用いた場合には従来法のように $\boldsymbol{A}^T(k)\, \boldsymbol{A}(k)$ を対角化することは不可能となる．しかし，スケーリングベクトル $\boldsymbol{F}$を含めて考えることによって，2の補数形式においても $\boldsymbol{A}^T(k)\, \boldsymbol{A}(k)$ の対角化が可能となることを初めて示した．

次に，従来法の問題点について述べる．従来法においては，収束特性（収束速度と推定精度）が極端に劣化するという問題を計算機シミュレーションにより明らかにしてきた．この原因は，入力信号の符号化と内部演算の符号化の違いにより演算精度が劣化するためで

ある．さらに，実際の適応動作においては入力信号の各ビットは“1”と“-1”ではなく“1”と“0”として扱われる．そのため，すべての入力信号にオフセットが含まれる状態になり，符号化された入力信号の自己相関行列の固有値が本来よりも大きな値になる．これによって，ステップサイズが小さい値に設定されるため，収束特性が大幅に劣化していた [19, 20].

提案法では，この問題点を解決するために入力信号の符号化に2の補数形式を用いて全てのアルゴリズムを導出した．これによって，入力信号の符号化と内部演算の符号化を統一的に行えるため，推定精度を大幅に改善することができる．さらに，従来法のように実際の適応動作において入力信号にオフセットがかからないため，入力信号の自己相関行列の固有値を正確に求めることができ，ステップサイズを最適な値に設定することができる．これにより，収束速度を大幅に改善することが可能となる．

## 3.3 マルチメモリブロック構造を有する分散演算形LMSアルゴリズム

### 3.3.1 適応関数空間の容量と収束速度

ここでは，タップ数 $N$ の増加に伴い収束速度が劣化する原因について述べる．

時刻 $k$ において更新対象となる適応関数空間要素は，アドレスマトリクス $\boldsymbol{A}^T(k)$ の $N$ 次行ベクトルをアドレスとして指定される．全適応関数空間（容量 $2^N$）が1つのアドレスベクトル当たりに更新される確率 $Pr_{update}$ は

$$Pr_{update} = \frac{1}{2^N} \tag{54}$$

となる．タップ数 $N$ の増加は，適応関数空間の容量を増加させるために更新確率が減少し，収束状態に達するまでには多くの繰り返しが必要になる．また，容量の大きな適応関数空間をRAMで実現することは，消費電力やハードウエア量の点で非常に不利となる．

この問題点を解決するために，適応関数空間を分割する手法が提案されてきた [25]．これは，容量 $2^N$ の適応関数空間を $M$ 個に分割することにより，個々の空間を小容量化して更新確率を向上させる手法である．この際，分割された適応関数空間のアドレスビット数 $R$ は $(N/M)$ となるので，分割された適応関数空間の容量は $2^R$ になる．したがって，分割さ

れた適応関数空間が1つのアドレスベクトル当たりに更新される確率$Pr'_{update}$は

$$Pr'_{update} = \frac{1}{2^R} \tag{55}$$

となり，分割しない場合の$2^{N-R}$倍に向上する．このように，分割化による更新確率の向上に伴って収束速度が改善され，消費電力やハードウエア量も減少させることが可能になる．ところが，この分割化の手法を従来法に適用しても，消費電力やハードウエア量は減少するが，収束速度の改善はあまり見られず大きく劣化したままである．

そこでDA適応フィルタと同様に，この適応関数空間を分割化した場合の構成法に対しても2の補数形式を用いた．この場合にも，2の補数形式を適用することができるのは，適応フィルタの場合と同様にアドレスマトリクスの乗算を対角化したからである．これによって，従来法に対して収束特性を大幅に改善し，さらに適応関数空間の分割数が多いほど収束速度を速くすることを可能とする．この適応関数空間を分割した構成をマルチメモリブロック構造(Multi-Memory Block Structure)と呼び，この構造を適用した分散演算形LMS適応フィルタをMDA適応フィルタと呼ぶことにする．

### 3.3.2 MDAアルゴリズム

$N$タップの分散演算形LMS適応フィルタの適応関数空間を$M$個に分割した場合のMDAアルゴリズムについて考える．各フィルタ係数ベクトルを，

$$\boldsymbol{W}_m(k) = [w_{m0}(k), w_{m1}(k), \ldots, w_{m(R-1)}(k)]^T \tag{56}$$

分割された適応関数ベクトルを，

$$\boldsymbol{P}_m(k) \triangleq [p_{m0}(k), p_{m1}(k), ..., p_{m(B-1)}(k)]^T \tag{57}$$

$$m = 0, 1, \ldots, M-1 \tag{58}$$

と定義する．ここで，$R$は各適応関数空間のアドレスのビット数を表す．各適応関数ベクトルを式(59)のように表すことによって，フィルタ出力は式(60)で求めることができる．

$$\boldsymbol{P}_m(k) = \boldsymbol{A}_m^T(k)\, \boldsymbol{W}_m(k) \tag{59}$$

$$y(k) = \sum_{m=0}^{M-1} \boldsymbol{F}^T \boldsymbol{P}_m(k) \tag{60}$$

式(59)において，アドレスマトリクス $\boldsymbol{A}_m(k)$ を

$$\boldsymbol{A}_m(k) \triangleq \begin{bmatrix} b_{m0}(k) & b_{m0}(k-1) & \cdots & b_{m0}(k-R+1) \\ b_{m1}(k) & b_{m1}(k-1) & \cdots & b_{m1}(k-R+1) \\ \vdots & \vdots & \ddots & \vdots \\ b_{m(B-1)}(k) & b_{m(B-1)}(k-1) & \cdots & b_{m(B-1)}(k-R+1) \end{bmatrix}^T$$

と定義する．

次に，MDA アルゴリズムを以下に示す．

$$\boldsymbol{P}_m(k+1) = \boldsymbol{P}_m(k) + 0.5\mu Re(k)\,\boldsymbol{F} \tag{61}$$

$$m = 0, 1, \ldots, M-1 \tag{62}$$

MDA 適応フィルタの構成を図14に示す．この構成では，良好な収束速度，小規模ハードウエア，低消費電力を実現可能である．

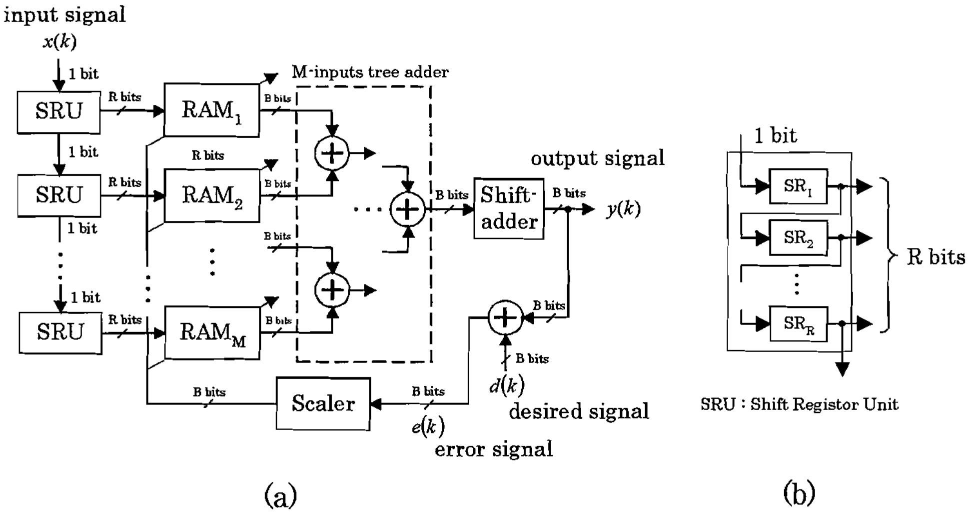



図 14: Block diagram of MDA adaptive filter.

## 3.4　シミュレーションによる収束特性の比較

収束速度における提案法の有効性を検証する．計算機シミュレーションは，図15に示されるシステム同定問題について行った．入力は平均0，分散0.01の白色信号，未知システム出力 $d(k)$ には$-75$[dB] の白色信号を観測雑音 $v(k)$ として加えた．ステップサイズ$\mu$は，同一のMSEを達成するという条件で最も良好な収束速度を示す値を設定した．内部演算語長は20ビット，結果は独立した20回の試行の集合平均である．なお，**2.3**節で述べたように収束特性の繰り返し回数は入力信号のサンプル数を表し，MSEは誤差の2乗平均値を表す．

図16はタップ数が16の場合に対する提案法と従来法の適応アルゴリズムの収束特性である．これより，それほどタップ数が大きくない場合でも，従来法の推定精度および収束速度が大幅に劣化していることがわかる．これに対して提案法では収束速度および推定精度を大幅に改善されている．なお，多くの計算例からタップ数を大きくするほど従来法が大きく劣化することを確認している [19].

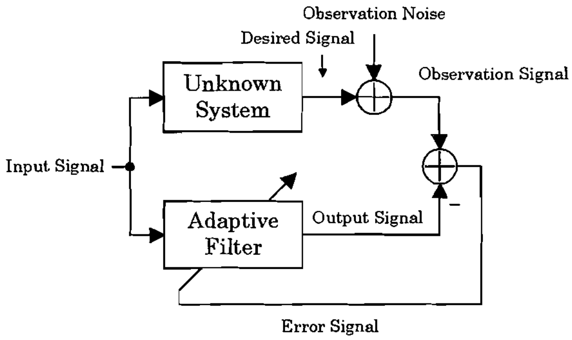



図 15: Simulation Model.

次に，提案法のMDAアルゴリズムの収束特性を明らかにするために，LMSアルゴリズム，およびVLSI評価の比較対象として用いるDLMSアルゴリズムの収束特性との比較を行う [8]．図17にタップ数が64の場合の提案法，LMS，そしてDLMSの収束特性のシミュレーション結果を示す．この結果から，提案法の収束特性において適応関数空間のアドレスが2ビット ($R = 2$) の場合には，DLMSアルゴリズムの収束特性とほぼ一致している．さらにアドレスが1ビット ($R = 1$) の場合には，LMSと同等の収束特性を得ることができる．このように提案するMDAアルゴリズムには，適応関数空間のアドレスのビット数が小さいほど良好な収束速度を示すという特長がある．

さらに，有色信号に対する収束特性を図18に示す．ここで，有色信号は係数0.99の1次AR過程を用いて生成した．なお，DA適応フィルタについては適応関数空間のアドレスが1ビット ($R = 1$) の場合である．この結果より，提案法はLMSアルゴリズムと同様に，適切なステップサイズパラメータを選択することにより有色信号に対しても収束することがわかる．しかし，従来法のDA適応フィルタでは入力信号のオフセットの影響により，収束速度は大きく劣化したままである．

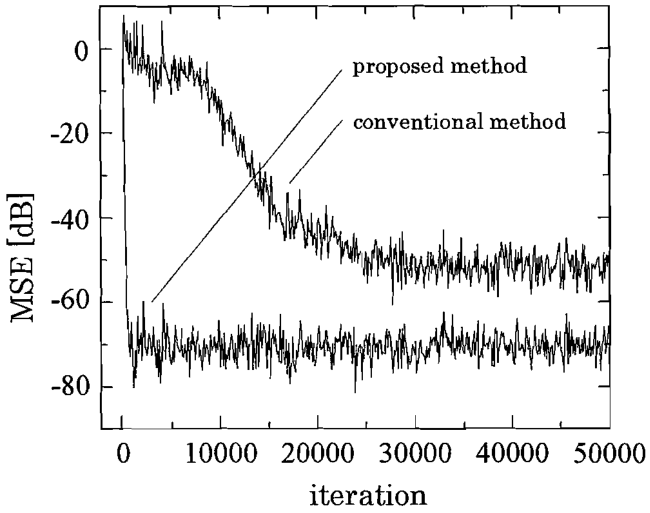



図 16: Comparison of convergence characteristics between proposed and conventional method.

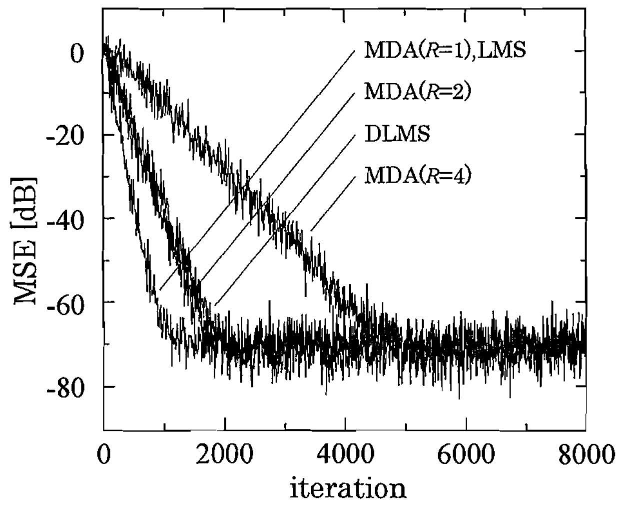



図 17: Convergence characteristics of MDA for various division numbers.

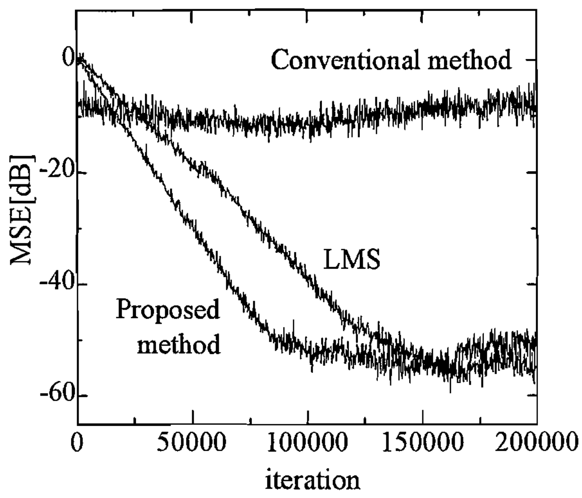



図 18: Convergence characteristics of LMS, proposed MDA, and conventional MDA algorithm when the input signal was the colored process.

## 3.5 MDA 適応フィルタの構成法

これまで，分散演算を用いた適応フィルタの VLSI アーキテクチャ，およびその高速性に関する検討は行われてこなかった．ここでは，高速性を考慮した MDA 適応フィルタのハードウェア構成を提案する．高速処理が可能な構成を図 19に示す．なお，これは分割数が 2 の場合の構成である．以下，この構成を MDA 適応フィルタの高速形と呼ぶことにする．

MDA 適応フィルタの構成のアルゴリズムは，フィルタ出力計算と適応関数空間の更新の異なる 2 つのステージから成る．

**[フィルタ出力計算ステージ]**

以下に示す動作を語長回繰り返すことによってフィルタ出力を求める．

[step1] 入力部に入力信号 $s(k)$ を 1 ビット入力する．

[step2] 入力部から出力されるビットパターンによって RAM のアドレスを指定し，関数 $\boldsymbol{P}(k)$ の値を読み出す．

[step3] 読み出された関数 $\boldsymbol{P}(k)$ の値をシフト加算部で累積加算する．

さらに，フィルタ出力 $y(k)$ と所望信号 $d(k)$ から誤差 $e(k)$ を求めている．また，更新ステージで用いるアドレス信号を生成するための入力信号ベクトル $\boldsymbol{S}(k)$ と，そのアドレスから読み出された関数 $\boldsymbol{P}(k)$ をレジスタで保持する．

**[更新ステージ]**

レジスタに保持されている入力 $\boldsymbol{S}(k)$ によって更新すべき RAM のアドレスを指定し，保持しておいた関数 $\boldsymbol{P}(k)$ とシフトされた誤差 $e(k)$ の和をそのアドレスに書き込む．この動作を語長回行うことによって適応関数空間の更新を行う．

従来法の構成では，外部に保持しているのは入力信号ベクトル $\boldsymbol{S}(k)$ のみで，関数 $\boldsymbol{P}(k)$ は保持していない．そのため，更新動作においては，RAM から関数 $\boldsymbol{P}(k)$ を読み出し，更新値を加算し，その加算結果を RAM に書き込むという動作が必要であった．これにより，パイプラインのピッチが大きくなり，処理速度が大幅に低下していた．しかし，提案する構成では関数 $\boldsymbol{P}(k)$ の値も保持する構成にしたため，RAM から関数を読み出す必要はなく，保持しておいた関数 $\boldsymbol{P}(k)$ と更新値の和を RAM に書き込む動作だけになる．従って，パイプラインのピッチをほぼ加算器の演算時間にまで減少することができるため，従来法よりも高速な実現が可能となる．この構成のタイムチャートを図 20 に示す．同図より，次数を増加させた場合でもパイプラインの加算段数が数段増加するだけである．パイプラインの加算段数は語長に対して占める割合が小さいため，この構成法では処理速度および滞在時間をほぼ一定の値に保つことが可能となる．

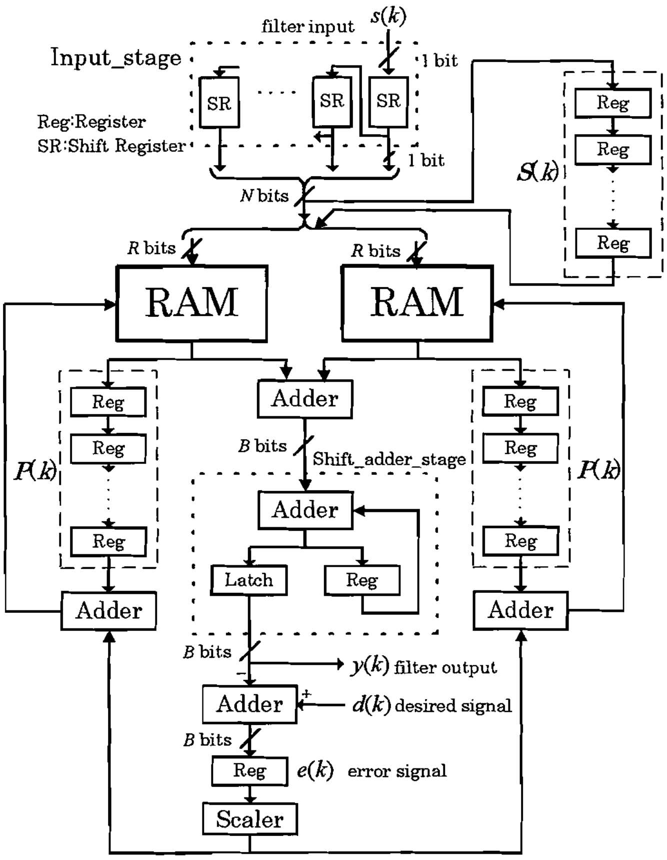



図 19: High-speed structure of MDA adaptive filter.

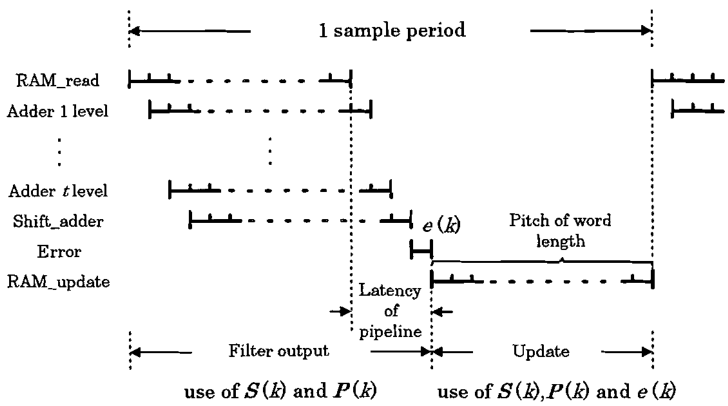



図 20: Timing chart of high-speed structure of MDA adaptive filter.

## 3.6 VLSI評価

マルチプライヤレスな構成法であるMDA適応フィルタの高速形の構成法に対してVLSI設計システムPARTHENONを用いてVLSI評価を行う[35]．MDA適応フィルタの高速形の構成において，タップ数が32，64，128の場合の構成に対するVLSI評価の結果を表1に示す．ここで，実部品として用いたセルライブラリの設計ルールは0.8$\mu$CMOSスタンダードセル(VLSIテクノロジ社)であり，電源電圧は5.0[V]である．ここでの設計仕様は，RAMの分割数をタップ数32の場合は8，64の場合は16，128の場合は32とする．また，演算に用いるデータ形式は2の補数表現による語長16ビットの固定小数点とする．

ここでは，提案した構成法の比較対象に，乗算器を使用したパイプライン構成の代表例としてDLMSアルゴリズムに基づくパイプライン適応フィルタを用いた[8]．この構成法に対するVLSI評価の結果を表2に示す．このDLMSパイプライン適応フィルタは，各タップ毎にパイプライン処理を実行する構成になっており，その各タップは乗算器，加算器，遅延器で構成されている．ここで使用した乗算器は，Boothのアルゴリズムに Wallace tree方式とCLA加算器を用いたものである．

この結果から，提案するMDA適応フィルタの高速形の構成法はタップ数128という高次においても，1.06MHzという高い処理速度と540nsという極めて小さい滞在時間を実現できることがわかる．さらに，DLMSパイプライン適応フィルタの構成に対して消費電力を68.2%，面積を76.0%，ゲート数を82.1%と大幅に削減することが可能となる．また，滞在時間に関しても約93.3%と大幅に減少でき，極めて小さい値に抑えることができる．より高次の場合に対しても，この構成法は処理速度と滞在時間をほぼ一定の値に保つことが可能である．

極めて滞在時間が小さい構成法として，HaradaらのLMSパイプライン適応フィルタの構成法についても比較を行う[12]．この構成法に対するVLSI評価の結果を表3に示す．なお，ここでは滞在時間が最小となる構成法であるArc1に対して比較を行った．この構成法では，ルックアヘッド変換を適用することによりLMSアルゴリズムのパイプライン処理化を可能とした．これによって，タップ数に依存しない高速な処理速度，LMSと同等の収束

表 1: VLSI evaluation for high-speed structure of MDA adaptive filter.

| | | | |
|---|---|---|---|
| Number of taps | 32 | 64 | 128 |
| Machine cycle[ns] | 27 | 27 | 27 |
| Sampling rate[MHz] | 1.122 | 1.089 | 1.058 |
| Latency[ns] | 486 | 513 | 540 |
| Power dissipation[W] | 2.176 | 4.184 | 8.205 |
| Area[mm$^2$] | 12.857 | 24.919 | 49.118 |
| Number of gates | 47,331 | 90,834 | 178,185 |

表 2: VLSI evaluation for the DLMS-pipelined adaptive filter.

| | | | |
|---|---|---|---|
| Number of taps | 32 | 64 | 128 |
| Machine cycle[ns] | 63 | 63 | 63 |
| Sampling rate[MHz] | 15.873 | 15.873 | 15.873 |
| Latency[ns] | 2016 | 4032 | 8064 |
| Power dissipation[W] | 6.446 | 12.892 | 25.785 |
| Area[mm$^2$] | 51.203 | 102.406 | 204.812 |
| Number of gates | 249,440 | 496,960 | 997,760 |

表 3: VLSI evaluation for the LMS-pipelined adaptive filter.

| Number of taps | 32 | 64 | 128 |
|---|---|---|---|
| Machine cycle[ns] | 131 | 131 | 131 |
| Sampling rate[MHz] | 7.634 | 7.634 | 7.634 |
| Latency[ns] | 131 | 131 | 131 |
| Power dissipation[W] | 6.833 | 13.666 | 27.333 |
| Area[mm$^2$] | 107.350 | 214.700 | 429.399 |
| Number of gates | 520,576 | 1,041,152 | 2,082,304 |

特性を維持した上で，131ns以下という本構成法よりも小さな滞在時間で実現が可能となる．しかし，その実現にはDLMSパイプライン適応フィルタの構成法において必要となるハードウエア量の2倍以上を必要とするため，高次における実現が大きな問題となる。これに対して，本構成法では約1/10程度のハードウエア量で実現が可能である。

## 3.7 まとめ

本章では，従来の分散演算を用いた適応フィルタに対して，入力信号の符号化に対しても2の補数形式を適用した高性能なマルチプライヤレスVLSIアーキテクチャを提案した．まず，従来の分散演算を用いた適応フィルタの収束特性が，入力信号の特殊な符号化によって大幅に劣化していることを明らかにし，これに対して，提案法ではすべてのアルゴリズムを2の補数形式で一般化することによって，収束速度および推定精度を大幅に改善することが可能であることを示した．さらに，従来法では検討されてこなかったVLSIアーキテクチャおよびその高速性を考慮した構成法についても提案した．最後に，提案の構成に対してVLSI設計システムPARTHENONを用いてVLSI評価を行った．その結果，提案法がタップ数128という高次においても，1.06MHz(0.8$\mu$CMOSスタンダードセル）という高い処理速度と540nsという極めて小さい滞在時間を実現できることを明らかにした．また，乗算器を用いたパイプライン構成の代表例であるDLMSパイプライン適応フィルタの構成法に対して，消費電力を68%，面積を76%，ゲート数を82%削減することを可能とした．さ

らに，より高次の構成に対しても処理速度および滞在時間をほぼ一定の値に保つことができることを示した．

以上より，本提案法では，高次に対しても高速性と極めて小さい滞在時間を維持した上で，低消費電力，低ハードウェア量を実現可能である．

# 第4章　ハーフメモリアルゴリズムを用いた分散演算形LMS適応フィルタ

## 4.1　はじめに

定係数の分散演算においては，入力信号の符号化方式にオフセットバイナリ形式が用いられてきた[15, 16, 25, 17, 22]．その理由は，適応関数空間に生じる“奇対称性”[11]を利用するためである．この性質を利用すれば，どちらか片側の空間を用いて適応関数空間を実現することができるため，RAMの容量を1/2に減少させることが可能である．また，前述のようにアルゴリズムの導出が容易であることも大きな理由の一つである．

一方，2の補数形式を用いた場合には，定係数と同様にDA適応フィルタでも適応関数空間に奇対称性は現れないと考えられてきた．

本章では，符号化方式に2の補数形式を用いた場合においても，適応関数空間が準奇対称性[12]を有することを解析的に示し，この性質を利用したハーフメモリアルゴリズムを導出する．次いで，収束特性を計算機シミュレーションにより評価し，最後に，提案するVLSIアーキテクチャを評価する[26]．

## 4.2　準奇対称性を利用したハーフメモリアルゴリズム

### 4.2.1　適応関数空間の準奇対称性

まず，タップ数が1の場合のDA適応フィルタについて適応関数空間が準奇対称性を有することを示し，その結果を利用してタップ数$N$および分割数$M$の場合の適応関数空間が準奇対称性を有することを示す．

**[タップ数1の適応関数空間]**

[11]あるアドレスの内容とそのアドレスビットを反転したアドレスの内容が互いに異符号で絶対値が等しい関係．

[12]本論文では，近似的に成立する奇対称性を準奇対称性と呼ぶ．

タップ数1の場合の入力信号，係数ベクトルを

$$\boldsymbol{S}(k) = s(k) \tag{63}$$

$$\boldsymbol{W}(k) = w_0(k) \tag{64}$$

とすると，フィルタ出力は次式で表される．

$$y(k) = w_0(k)s(k) = w_0(k)\,\boldsymbol{A}(k)\,\boldsymbol{F} \tag{65}$$

ここで，入力信号$s(k)$とアドレスマトリクス$\boldsymbol{A}(k)$は

$$s(k) = \boldsymbol{A}(k)\,\boldsymbol{F} \tag{66}$$

$$\boldsymbol{A}(k) = [b_0(k), b_1(k), \ldots, b_{B-1}(k)] \tag{67}$$

である．また，適応関数空間の更新式は式(47)において$N=1$と置き，

$$\boldsymbol{P}(k+1) = \boldsymbol{P}(k) + 0.5\mu e(k)\,\boldsymbol{F} \tag{68}$$

となる．この式のベクトルを要素で表すと

$$\begin{bmatrix} p_0(k+1) \\ p_1(k+1) \\ \vdots \\ p_{B-1}(k+1) \end{bmatrix} = \begin{bmatrix} p_0(k) \\ p_1(k) \\ \vdots \\ p_{B-1}(k) \end{bmatrix} + 0.5\mu e(k) \begin{bmatrix} -2^0 \\ 2^{-1} \\ \vdots \\ 2^{-B+1} \end{bmatrix} \tag{69}$$

となる．これは，時刻$k$において更新される任意の適応関数空間の要素に対する更新値を示した式であり，適応関数空間の要素全体を表した式ではない．そこで，準奇対称性を検討するために，この更新式を適応関数空間の要素全体を表現する式に拡張する．タップ数が1の場合，適応関数空間のアドレスは（ビットの値である）1と0の2種類であるため，それぞれに対応する適応関数空間を$P^1(k)$，$P^0(k)$とする．式(69)において，$p_0(k)$から$p_{B-1}(k)$までの各更新値$0.5\mu e(k)\,\boldsymbol{F}$は，アドレスマトリクス$\boldsymbol{A}(k)$の各ビットによって指定される適応関数空間に加えられる．つまり，アドレスマトリクス$\boldsymbol{A}(k)$を構成する各ビッ

ト $b_0(k), b_1(k), \ldots, b_{B-1}(k)$ の値が 1 の場合には $P^1(k)$ に，0 の場合には $P^0(k)$ にスケーリングベクトル $\boldsymbol{F}$の重みに応じた更新値を加える．従って更新式は

$$\begin{aligned} P^1(k+1) &= P^1(k) + 0.5\mu e(k)\,\boldsymbol{A}(k)\,\boldsymbol{F} \\ &= P^1(k) + 0.5\mu e(k) s(k) \end{aligned} \tag{70}$$

$$\begin{aligned} P^0(k+1) &= P^0(k) + 0.5\mu e(k)\,\bar{\boldsymbol{A}}(k)\,\boldsymbol{F} \\ &= P^0(k) + 0.5\mu e(k) \bar{s}(k) \end{aligned} \tag{71}$$

と表すことができる．ここで，

$$\bar{\boldsymbol{A}}(k) = [\bar{b}_0(k), \bar{b}_1(k), \ldots, \bar{b}_{B-1}(k)] \tag{72}$$

$$\bar{s}(k) = \bar{\boldsymbol{A}}(k)\,\boldsymbol{F} \tag{73}$$

であり，$\bar{\boldsymbol{A}}(k)$ の各要素$\bar{b}_i(k)$ は $\boldsymbol{A}(k)$ の各要素 $b_i(k)$ をビット反転したものである．次に，2 の補数形式における $s(k)$ と$\bar{s}(k)$ には

$$\bar{s}(k) = -[s(k) + 2^{-B+1}] \tag{74}$$

の関係が成立する．ここで，値$2^{-B+1}$は 2 の補数形式における最小ビット重みを表す．これより，係数更新式の式 (70)，(71) は

$$P^1(k+1) = P^1(k) + 0.5\mu e(k) s(k) \tag{75}$$

$$P^0(k+1) = P^0(k) - 0.5\mu e(k) s(k) - 0.5\mu e(k) \cdot 2^{-B+1} \tag{76}$$

となる．

ここで，式 (75) と (76) の更新値をそれぞれ $u_1(k)$,$u_0(k)$ とする．すなわち，

$$u_1(k) = 0.5\mu e(k) s(k) \tag{77}$$

$$\begin{aligned} u_0(k) &= 0.5\mu e(k) s(k) + 0.5\mu e(k) \cdot 2^{-B+1} \\ &= u_1(k) + 0.5\mu e(k) \cdot 2^{-B+1} \end{aligned} \tag{78}$$

である．そして，式(78) の右辺第２項と第１項の比を求めると，

$$\frac{u_0(k)-u_1(k)}{u_1(k)} = \frac{0.5\mu e(k)\cdot 2^{-B+1}}{0.5\mu e(k)s(k)} \tag{79}$$

となり，これを統計的に評価するために分子と分母の２乗平均を求めると，

$$\begin{aligned}\frac{E[(0.5\mu e(k)\cdot 2^{-B+1})^2]}{E[(0.5\mu e(k)s(k))^2]} &= \frac{(0.5\mu)^2\cdot(2^{-B+1})^2\cdot E[e^2(k)]}{(0.5\mu)^2\cdot E[e^2(k)]\cdot E[s^2(k)]}\\ &= \frac{(2^{-B+1})^2}{E[s^2(k)]}\\ &= \frac{2^{-2B+2}}{\sigma^2}\end{aligned} \tag{80}$$

となる．なお，入力信号 $s(k)$ は平均 0，分散 $\sigma^2$ の白色信号であり，LMS アルゴリズムにおける直交性の原理より入力信号 $s(k)$ と誤差信号 $e(k)$ は独立であると仮定した [39]．式 (80) は

$$\sigma^2 \gg 2^{-2B+2} \tag{81}$$

の場合に 0 と見なすことができるため，式 (78) の右辺第２項が第１項に対して無視でき

$$u_1(k) \approx u_0(k) \tag{82}$$

となる．そして，式(82)，式(75) と (76) より，

$$P^0(k) \approx -P^1(k) \tag{83}$$

が成立する．これは，アドレスが反転の関係にある適応関数空間 $P^0(k)$ と $P^1(k)$ の内容は異符号で近似的に絶対値が等しくなることを表している．

以上より，これまで２の補数形式を用いた DA 適応フィルタの適応関数空間は奇対称とは全く異なると考えられてきたが，式(81) の条件のもとに適応関数空間には近似的奇対称性つまり準奇対称性が存在することが明らかになった．なお，準奇対称性による誤差の影響は非常に小さく，比較的語長の短い 6 ビットにおいても良好な収束特性を有することを計算機シミュレーションによって確認している．

[タップ数Nの適応関数空間]

タップ数$N$のDA適応フィルタの適応関数空間について検討するために，まず適応関数空間を$N$個に分割したMDA適応フィルタについて検討し，その結果を用いて$N$タップ適応フィルタの適応関数空間の準奇対称性を示す．

$N$個に分割したMDA適応フィルタはアドレス線数が1ビットであるため，$N$個の各タップにおける適応関数空間はタップ数1における結果を適用して

$$P_m^0(k) \approx -P_m^1(k), \quad (m = 0, 1, \ldots, N-1)$$

の関係が成立する．ここで，$P_m^b(k)$は時刻$k$における$m$番目の適応関数空間で，アドレスが$b$であることを表す．DA適応フィルタのフィルタ出力計算式

$$y(k) = \boldsymbol{F}^T \boldsymbol{P}(k) \tag{84}$$

とMDA適応フィルタのフィルタ出力計算式

$$y(k) = \sum_{m=0}^{M-1} \boldsymbol{F}^T \boldsymbol{P}_m(k)$$

を比較することによって，DA適応フィルタの適応関数空間は$N$個に分割したMDA適応フィルタの適応関数空間を用いて

$$P^{[b_0, b_1, \ldots, b_{N-1}]}(k) = P_0^{b_0}(k) + P_1^{b_1}(k) + \cdots + P_{N-1}^{b_{N-1}}(k) \tag{85}$$

と表すことができる．ここで，左辺の$\boldsymbol{P}(k)$の添え字$[b_0, b_1, \ldots, b_{N-1}]$はタップ数$N$に対するDA適応フィルタの適応関数空間のアドレスを表し，右辺の各$\boldsymbol{P}_m(k)$に対する添え字$b_0, b_1, \ldots, b_{N-1}$はタップ数$N$，分割数$N$に対するMDA適応フィルタの各適応関数空間のアドレスを表している．次に，DA適応フィルタにおいて反転したビットパターンをアドレスとする適応関数空間は

$$P^{[\bar{b}_0, \bar{b}_1, \ldots, \bar{b}_{N-1}]}(k) = P_0^{\bar{b}_0}(k) + P_1^{\bar{b}_1}(k) + \cdots + P_{N-1}^{\bar{b}_{N-1}}(k) \tag{86}$$

となり，準奇対称性を考慮すると

$$P^{[\bar{b}_0,\bar{b}_1,\ldots,\bar{b}_{N-1}]}(k) \approx -P_0^{b_0}(k) - P_1^{b_1}(k) - \cdots - P_{N-1}^{b_{N-1}}(k) \tag{87}$$

$$= -P^{[b_0,b_1,\ldots,b_{N-1}]}(k) \tag{88}$$

となる．従って，タップ数$N$のDA適応フィルタもタップ数1の場合と同様に適応関数空間が準奇対称性を有することがわかる．

**[分割数Mの適応関数空間]**

適応関数空間を分割するMDA適応フィルタの適応関数空間の準奇対称性を示す．ここで，タップ数を$N$，分割数を$M$，各適応関数空間のアドレス線数を$R(N/M)$とする．

式(85)の右辺を$R$個毎にまとめると，

$$P^{[b_0,b_1,\ldots,b_{N-1}]}(k) = [P_0^{b_0}(k)+\cdots+P_{R-1}^{b_{R-1}}(k)]+\cdots+[P_{N-R-2}^{b_{N-R-2}}(k)+\cdots+P_{N-1}^{b_{N-1}}(k)] \tag{89}$$

ここで，$R$個毎にまとめた$m$番目の適応関数空間を

$$P_m^{[b_{m\times R},\ldots,b_{m\times R+R-1}]}(k) = P_{m\times R}^{b_{m\times R}}(k)+P_{m\times R+1}^{b_{m\times R+1}}(k)+\cdots+P_{m\times R+R-1}^{b_{m\times R+R-1}}(k)$$
$$(m=0,1,\cdots,M-1) \tag{90}$$

とおくと，式(89)は，

$$P^{[b_0,b_1,\ldots,b_{N-1}]}(k) = P_0^{[b_0,\cdots,b_{R-1}]}(k)+\cdots+P_{M-1}^{[b_{(M-1)R},\cdots,b_{MR-1}]}(k)$$

と表される．式(90)で表される分割された各適応関数空間において，アドレスのビットを反転させた適応関数空間は，式(83)を考慮すると次のようになる．

$$\begin{aligned}
P_m^{[\bar{b}_{m\times R},\ldots,\bar{b}_{m\times R+R-1}]}(k) &= P_{m\times R}^{\bar{b}_{m\times R}}(k)+P_{m\times R+1}^{\bar{b}_{m\times R+1}}(k)+\cdots+P_{m\times R+R-1}^{\bar{b}_{m\times R+R-1}}(k) \\
&\approx -P_{m\times R}^{b_{m\times R}}(k)-P_{m\times R+1}^{b_{m\times R+1}}(k)-\cdots-P_{m\times R+R-1}^{b_{m\times R+R-1}}(k) \\
&= -P_m^{[b_{m\times R},\ldots,b_{m\times R+R-1}]}(k) \qquad (m=0,1,\cdots,M-1)
\end{aligned}$$

これより，$M$分割された各適応関数空間においても準奇対称性が成立することが示された．

### 4.2.2 ハーフメモリアルゴリズム

提案した分散演算型LMSアルゴリズム（以下，フルメモリアルゴリズムとよぶ）では，前節の検討から適応関数空間が準奇対称性を有することがわかった．以下に示すハーフメモリアルゴリズムでは，準奇対称性を利用して適応関数空間の容量を1/2とし，収束速度を向上させることが可能となる．ここで，フルメモリアルゴリズムにおける適応関数空間のアドレス線数を$R$ビットとすると，ハーフメモリアルゴリズムにおけるアドレス線数は最上位ビットを除いた$R-1$ビットになる．

以下にハーフメモリアルゴリズムにおける適応関数空間からの読み出し・更新方法を以下に示す．

**・適応関数空間からの読み出し**

```
begin
    for i := 1 to B do
        begin
            if adrs_MSB = 0 then
                (R−1 ビットのアドレスを用いて
                    適応関数空間から関数値を読み出す);
            if adrs_MSB = 1 then
                (R−1 ビットのアドレスをビット反転する);
                (そのアドレスで指定した適応関数空間
                    から関数値を読み出す);
                (読み出した値の正負を反転させる);
        end
end
```

・**適応関数空間の更新**

```
begin
  for i := 1 to B do
    begin
      if adrs_MSB = 0 then
        (R−1ビットのアドレスで指定された
          適応関数空間の値に更新値を加える);
      if adrs_MSB = 1 then
        (R−1ビットのアドレスをビット反転する);
        (更新値の正負を反転させる);
        (ビット反転したアドレスで指定した
          適応関数空間の値に更新値を加算する);
    end
end
```

ここで$adrs_{MSB}$はアドレスの最上位ビットを表す．以上の変更点をフルメモリアルゴリズムに適用することによって，フルメモリの場合と同様に適応動作が可能となる．

## 4.3 シミュレーションによる収束特性の比較

本節では分散演算型適応フィルタにおいて，フルメモリアルゴリズムを用いた場合とハーフメモリアルゴリズムを用いた場合の収束特性の比較を行う．ここで，シミュレーションを行う条件として，フィルタのタップ数を32，入力信号を平均0，分散0.05の白色ガウス信号，そして信号語長を16ビットとした．また，所望信号$d(k)$には-80[dB]の白色信号を観測雑音として加えた．結果は独立した20回の試行の集合平均としている．それぞれのステップサイズは，収束状態において同一のMSEを達成する条件のもとで，収束速度が最も高速になる値を設定した．なお，**2.3**節で述べたように収束特性の繰り返し回数は入力信号のサンプル数を表し，MSEは誤差の2乗平均値を表す．

図21，22から，同じ分割数$M$においてハーフメモリの収束速度がフルメモリに対して約2倍収束が速くなっていることがわかる．これはハーフメモリの適応関数空間の容量がフルメモリの1/2になったことによって，各空間の更新確率が2倍になったためである．

また，ハーフメモリとフルメモリの収束速度がほぼ同等である場合を比較すると，それぞれのアドレス線数$R$が等しい場合に収束速度がほぼ一致することがわかった．この結果からハーフメモリの構成法では，フルメモリの構成法と同等の収束速度をより少ない分割数で実現できるため，より消費電力，ハードウェア量を削減することが可能となる．

さらに提案法の収束特性を明らかにするために，LMSアルゴリズム，および**4.4**においてVLSI評価の比較対象として用いるDLMSアルゴリズムの収束特性との比較を行う[8]．図23にタップ数が32の場合のLMS，そしてDLMSの収束特性のシミュレーション結果を示す．この結果から，ハーフメモリアルゴリズムではフルメモリと等しい分割数($M=16$)において，フルメモリの約2倍の収束速度が得られ，さらにLMSアルゴリズムと同等の収束速度を得られることがわかる．

最後に，式(83)で示した準奇対称性について語長の影響を検討する．図24に，語長$B=6\sim16$ビットに対するハーフメモリとフルメモリの収束特性を示す．なお，語長の影響を検討するために観測雑音は加えていない．これより，各語長におけるハーフメモリアルゴリズムのMSEは同一語長のフルメモリアルゴリズムのMSEと同程度であること，また，各語長におけるハーフメモリアルゴリズムの収束速度は同一語長のフルメモリアルゴリズムの収束速度の約2倍であることがわかる．特に6ビットという比較的短い語長においても良好な特性を示していることから，奇対称性が大きく崩れることはなく，語長の影響は非常に小さいことがわかる．なお，多くの計算例より同様の結果が得られることを確認している．

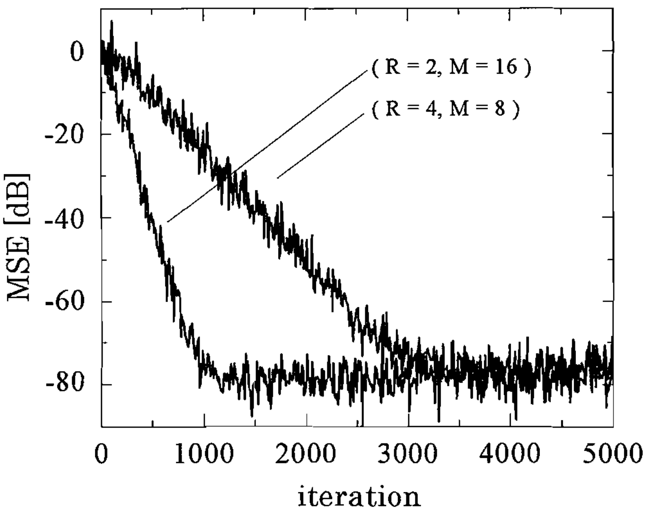



図 21: Convergence characteristics of full-memory algorithm.

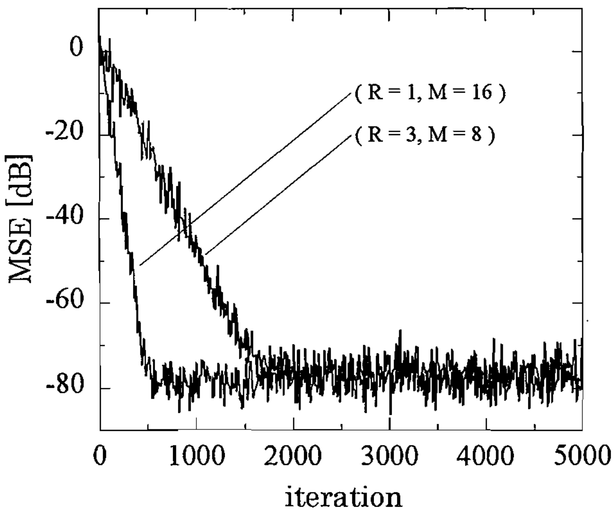



図 22: Convergence characteristics of half-memory algorithm.

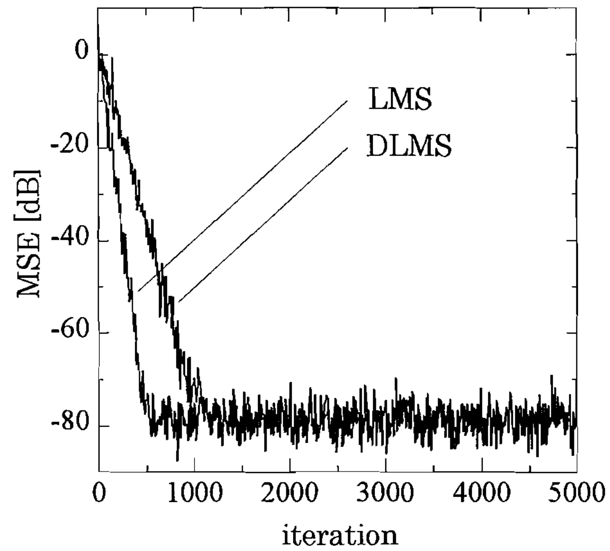



図 23: Convergence characteristics of LMS and DLMS algorithm.

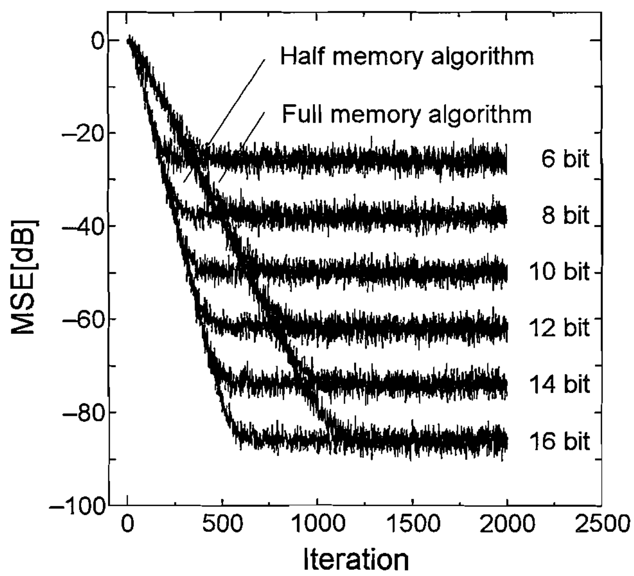



図 24: Convergence characteristics of half-memory and full-memory algorithm for various word length.

## 4.4 VLSI評価

本節では，まず，分散演算型適応フィルタにおいてフルメモリアルゴリズムを用いた構成法とハーフメモリアルゴリズムを用いた構成法について述る．次いで，それらの構成法に対してVLSI設計システムPARTHENONを用いてVLSI評価を行った結果を示す[35].

### 4.4.1 フルメモリの構成法

比較対象となるフルメモリアルゴリズムを用いた分散演算型LMS適応フィルタの構成を再記し簡単に説明する．これまで処理速度の向上を図る構成として図25のような構成を提案してきた[21]．この構成の動作はフィルタ出力を求める動作とRAMの更新を行う動作の2つのステージに分けることができる．

**[出力計算ステージ]**

ここでは，分散演算の基本動作通り入力のビットパターンによってRAMから読み出された値を語長回の累積加算を行うことによってフィルタ出力を求めている．さらにこのステージでは，得られたフィルタ出力$y(k)$と所望信号$d(k)$との差をとり，誤差$e(k)$を求めている．また，次のステージでRAMの更新を行うために必要なデータであるRAMの読み出し時のアドレス指定に用いた入力$\boldsymbol{S}(k)$と，そのアドレスから読み出された関数$\boldsymbol{P}(k)$の値をレジスタで保持している．

**[更新ステージ]**

前のステージで保持した関数$\boldsymbol{P}(k)$の値と，誤差$e(k)$をシフトした値を加算器によって加え合わせる．この値を前のステージで保持した入力$\boldsymbol{S}(k)$によって読み出し時に指定されたアドレスに書き込む．この動作を語長回行うことによってRAMの更新を行う．

この構成のタイムチャートを図26に示す．同図からわかるようにこの構成のメリットとして，次数を大きく増加させた場合にもパイプラインの加算段数がわずかに数段増加するだけなので，処理速度および滞在時間をほぼ一定の値に保つことができる．

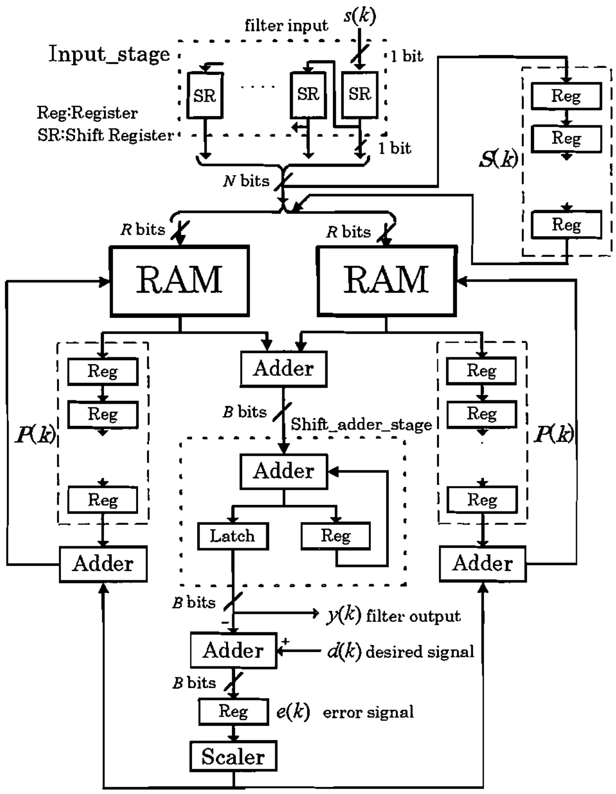



図 25: Structure for full-memory algorithm.

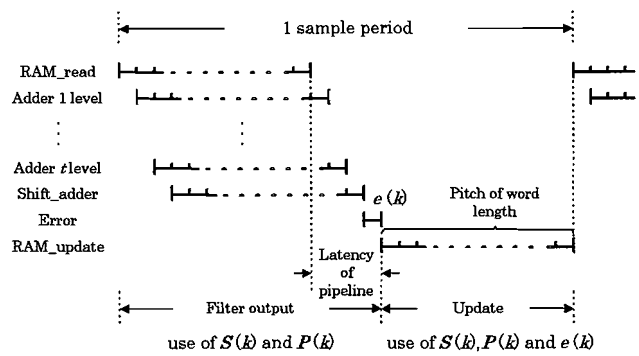



図 26: Timing chart of full-memory architecture.

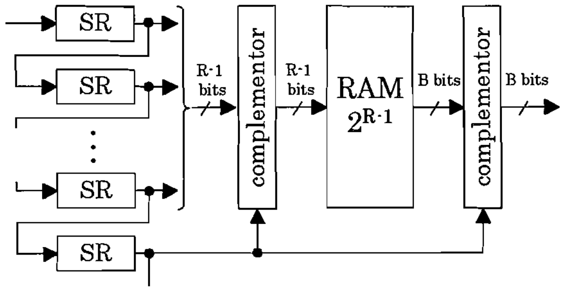



図 27: Input part for structure of half-memory algorithm.

### 4.4.2 ハーフメモリの構成法

フルメモリの構成法に対して適応関数空間の準奇対称性を利用するハーフメモリアルゴリズムを用いた場合の構成法について説明する．フルメモリの構成からの変更点は2箇所である．

1つ目の変更点は入力部である．フルメモリの構成法ではシフトレジスタからの出力をそのままRAMのアドレスとしてRAMの関数を読み出していた．そのためアドレス線が $R$ 本の場合では適応関数空間の容量も $2^R$ となる．これに対して，ハーフメモリの構成法では図27のように入力部のシフトレジスタにおける最下段の出力を補数器の制御用信号として用いる．この信号が0であればそのままのアドレスを用いて値を読み出し，その値をそのまま出力する．また，信号が1であればアドレスに対してビット反転を行ったビットパターンをアドレスとして値を読み出し，その値をビット反転し正負を反転させた値を出力するという構成にしている．これによって適応関数空間のアドレス線数を $R-1$ 本にすることができ，フルメモリの構成法に対して適応関数空間の容量を $2^{R-1}$ と 1/2 にすることが可能となる．

2つ目の変更点は更新部である．更新部も入力部と同様に最下段のアドレスを補数器の制御用信号として用い，この信号が0であれば更新値をそのままRAMに加え合わせ，1であればビット反転を行い正負を反転させた値をRAMに加え合わせる構成にしている．

### 4.4.3 評価結果

フルメモリ，ハーフメモリの各構成法に対してVLSI評価を行う．また，比較対象として乗算器を使用したパイプライン構成の代表例であるDLMSアルゴリズムに基づくパイプライン適応フィルタ[8]と信号処理装置の代表的な実現方法であるDSPモデルを用いた[42]．DLMSパイプライン適応フィルタは，各タップ毎にパイプライン処理を実行する構成になっており，その各タップは乗算器，加算器，そして遅延器で構成されている．DSPモデルは，積和乗算を効率よく計算する演算回路とデータRAM，プログラムRAM，制御回路から構成されるが，このモデルはLMSアルゴリズムを実行するために必要な最小限の構成とした．比較対象において使用した乗算器は，BoothのアルゴリズムにWallace tree方式とCLA加算器を用いたものである．また，実部品として用いたセルライブラリの設計ルールは0.8$\mu$mCMOSスタンダードセル（VLSIテクノロジ社）であり，電源電圧は5.0[V]である．また，演算に用いるデータ形式は2の補数表現による語長16ビットの固定小数点である．

タップ数60におけるフルメモリ，ハーフメモリ，そしてDLMSパイプライン適応フィルタの構成に対する評価結果を，表4，表5，表6に示す．まず，フルメモリとハーフメモリを比較すると，アドレス線数が等しい（収束速度は同程度）場合には，アドレス線数4，5に対して消費電力を22.1%，18.7%，面積を14.7%，11.3%，そしてゲート数を13.0%，9.6%削減可能である．また，フルメモリよりもアドレス線数が1だけ少ないハーフメモリ（フルメモリよりも約2倍高速な収束速度を示す）は，同等の消費電力，面積，そしてゲート数を示している．DLMSパイプライン適応フィルタとハーフメモリを比較すると，アドレス線数が4，5に対して，ハーフメモリは消費電力を75.7%，79.2%，面積を80.8%，82.6%，ゲート数を84.7%，86.3%削減可能である．また，出力滞在時間も85.6%，85.1%と大幅に削減することが可能であり，極めて小さい値を示している．

次に，タップ数120におけるフルメモリ，ハーフメモリ，そしてDLMSパイプライン適応フィルタの構成に対する評価結果を表7，表8，表9に示す．フルメモリとハーフメモリを比較すると，アドレス線数が等しい（収束速度は同程度）場合は，アドレス線数が4，5本に対

表 4: VLSI evaluation for structure of full-memory algorithm for N=60.

| Number of Taps | 60 | | |
|---|---|---|---|
| Number of address lines | 4 | 5 | 6 |
| Division number | 15 | 12 | 10 |
| Machine cycle[ns] | 27 | 28 | 29 |
| Sampling rate[MHz] | 1.089 | 1.050 | 1.014 |
| Latency[ns] | 513 | 532 | 551 |
| Power dissipation[W] | 3.946 | 3.239 | 2.770 |
| Area[$mm^2$] | 23.487 | 20.459 | 19.159 |
| Number of gates | 85,696 | 74,136 | 67,768 |

表 5: VLSI evaluation for structure of half-memory algorithm for N=60.

| Number of Taps | 60 | |
|---|---|---|
| Number of address lines | 4 | 5 |
| Division number | 12 | 10 |
| Machine cycle[ns] | 30 | 31 |
| Sampling rate[MHz] | 0.980 | 0.949 |
| Latency[ns] | 570 | 589 |
| Power dissipation[W] | 3.073 | 2.634 |
| Area[$mm^2$] | 20.043 | 18.153 |
| Number of gates | 74,529 | 67,016 |

表 6: VLSI evaluation for DLMS-pipelined adaptive filter for N=60.

| Number of Taps | 60 |
|---|---|
| Machine cycle[ns] | 63 |
| Sampling rate[MHz] | 15.873 |
| Latency[ns] | 3945 |
| Power dissipation[W] | 12.653 |
| Area[$mm^2$] | 104.510 |
| Number of gates | 487,811 |

表 7: VLSI evaluation for structure of full-memory algorithm for N=120.

| Number of Taps | 120 | | |
|---|---|---|---|
| Number of address lines | 4 | 5 | 6 |
| Division number | 30 | 24 | 20 |
| Machine cycle[ns] | 29 | 29 | 29 |
| Sampling rate[MHz] | 0.985 | 0.985 | 0.985 |
| Latency[ns] | 580 | 580 | 580 |
| Power dissipation[W] | 7.165 | 6.080 | 5.365 |
| Area[$mm^2$] | 46.032 | 40.044 | 37.442 |
| Number of gates | 166,979 | 144,070 | 131,320 |

表 8: VLSI evaluation for structure of half-memory algorithm for N=120.

| Number of Taps | 120 | |
|---|---|---|
| Number of address lines | 4 | 5 |
| Division number | 24 | 20 |
| Machine cycle[ns] | 30 | 31 |
| Sampling rate[MHz] | 0.952 | 0.922 |
| Latency[ns] | 600 | 620 |
| Power dissipation[W] | 5.986 | 5.106 |
| Area[$mm^2$] | 39.234 | 35.429 |
| Number of gates | 144,953 | 129,784 |

表 9: VLSI evaluation for DLMS-pipelined adaptive filter for N=120.

| Number of Taps | 120 |
|---|---|
| Machine cycle[ns] | 63 |
| Sampling rate[MHz] | 15.873 |
| Latency[ns] | 7560 |
| Power dissipation[W] | 24.173 |
| Area[$mm^2$] | 192.011 |
| Number of gates | 935,400 |

して，消費電力を16.5%，16.0%，面積を14.8%，11.5%，そしてゲート数を13.2%，9.9%削減可能である．また，フルメモリよりもアドレス線数が1だけ少ないハーフメモリ（フルメモリよりも約2倍高速な収束速度を示す）は，同等の消費電力，面積，そしてゲート数を示している．また，DLMSパイプライン適応フィルタとハーフメモリを比較すると，アドレス線数が4，5に対して，消費電力を75.2%，78.9%，面積を80.0%，81.5%，ゲート数を84.5%，86.1%と大幅に削減することが可能となる．また，出力滞在時間も91.2%，91.8%と大幅に削減することが可能で，極めて小さい値を示している．

以上より，提案するハーフメモリの構成法はフルメモリと比較してより小さな消費電力，面積，そしてゲート数を達成可能であり，より高次に対しても処理速度と出力滞在時間をほぼ一定の値に保つことが可能である．これより，提案するハーフメモリの構成は，適応フィルタに要求される性能を同時に満たすことが可能であることがわかる．

最後に，DSPモデルに対する評価結果を表10に示す．タップ数に対してサンプリングレートは減少し，出力滞在時間は増加していることがわかる．そして，タップ数120のサンプリングレートは28.4kHzでありハーフメモリの3.0%，出力滞在時間は35,265nsでありハーフメモリの58.7倍である．これは，DSPモデルではプログラムにより演算をシーケンシャルに実行するため，処理時間と出力滞在時間はタップ数に大きく依存するためである．これに対して，ハーフメモリアルゴリズムのサンプリングレートと出力滞在時間はタップ数に依存しないため高次においても良好な処理能力を有する．タップ数60に対するハーフメモリとDSPモデルを比較する．アドレス線数が4，5に対して，ハーフメモリは消費電力が194.5%，166.7%，面積は109.8%，99.4%，ゲート数は82.6%，74.2%，タップ数120に対しては，消費電力が237.5%，202.6%，面積は129.9%，117.3%，ゲート数は97.6%，87.34%である．ゲート数はいずれにおいてもDSPより少なく，面積はタップ数60では同程度，タップ数120では25%程度上回る．消費電力は2倍程度大きいが，これはハーフメモリのマシンサイクルがDSPより2.2倍大きいためである．以上より，ハードウエアに関してはDSPはハーフメモリと同程度かやや上回る性能を示しているものの，サンプリングレートと出力滞在時間が極端に劣るため，リアルタイム処理を目的とする場合の応用範囲が極端に限

表 10: VLSI evaluation for DSP model for N=120,60

| Number of Taps | 120 | 60 |
|---|---|---|
| Machine cycle[ns] | 67 | 67 |
| Sampling rate[MHz] | 0.0284 | 0.0527 |
| Latency[ns] | 35,265 | 18,957 |
| Power dissipation[W] | 2.520 | 1.580 |
| Area[mm$^2$] | 30.210 | 18.260 |
| Number of gates | 148,589 | 90,269 |

定される.

## 4.5　まとめ

本章では，2の補数形式を用いた分散演算型LMS適応フィルタの構成法に対してハーフメモリアルゴリズムを適用したマルチプライヤレスVLSIアーキテクチャを提案した．まず，分散演算型LMS適応フィルタにおいて適応関数空間が準奇対称性を有することを解析的に明らかにした．さらに，この性質を利用したハーフメモリアルゴリズムを提案し，このアルゴリズムを適用した新しい分散演算型LMS適応フィルタの構成法を提案した．最後に，この構成法に対して収束特性の検討およびVLSI評価を行った．

その結果，これまでに提案したフルメモリアルゴリズムを用いた構成法に対して，消費電力，ハードウェア量をほぼ等しい値に抑えた上で，収束速度を約2倍高速化できることを明らかにした．また，同等の収束速度を維持した上で，低消費電力が可能となることについても明らかにした．さらに，比較対象として用いたDLMSパイプライン適応フィルタの構成法に対して，消費電力を75%，面積を79%，ゲート数を84%と大幅に削減できることを示した．

以上のことから，ハーフメモリアルゴリズムを適用した本提案法が，これまでの分散演算型LMS適応フィルタの構成法に対して，高速性および低滞在時間を維持した上で収束速度の向上，低消費電力を実現できることを明らかにした．

# 第5章　分散演算形LMSアルゴリズムの収束条件解析

## 5.1　はじめに

分散演算形LMSアルゴリズムに関する収束条件などの理論解析は，その難解さからこれまで検討されてこなかった．

本章では，分散演算を用いたLMSアルゴリズムの収束条件を初めて明らかにする．

分散演算形LMS適応フィルタは未知システムの伝達関数を適応関数空間として推定するため，収束条件式は全適応関数空間に対して定義される．しかし，更新式は時刻$k$において更新対象となる要素のみを記述しているため，この更新式から全適応関数空間に対する収束条件式を導くことはできない．そこで，まず，分散演算形LMS適応フィルタの更新式を全適応関数空間に拡張し，この拡張された更新式において入力信号ベクトルを新たに定義する．次いで，推定誤差を適応関数空間の最適値と推定値の差として定義することにより，収束条件を時刻$k$の経過とともに推定誤差が減少するための条件として導いて行く．

## 5.2　更新式の拡張

更新式の拡張手順は，まずタップ数が1と2について行い，その結果を用いてタップ数$N$に一般化する．

**[タップ数1における拡張]**

2の補数形式を用いる提案法において，一例として入力信号$s(k)$が

$$s(k) = 0 \times (-2^0) + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \tag{91}$$

の場合，ビット'0'，'1'に対応する適応関数空間要素$p0(k)$，$p1(k)$は提案法の更新式を用いて次のように更新される．

$$\begin{aligned} p0(k+1) &= p0(k)+0.5\mu Ne(k)[-2^0+2^{-2}] \\ &= p0(k) + 0.5\mu Ne(k)\bar{s}(k) \end{aligned} \tag{92}$$

表 11: Relation between symbols and bit patterns.

| Symbol | Bit pattern | Symbol | Bit pattern |
|---|---|---|---|
| $\boldsymbol{a}$ | $[00]^T$ | $\boldsymbol{c}$ | $[10]^T$ |
| $\boldsymbol{b}$ | $[01]^T$ | $\boldsymbol{d}$ | $[11]^T$ |

$$\begin{aligned} p1(k+1) &= p1(k)+0.5\mu Ne(k)[2^{-1}+2^{-3}] \\ &= p1(k) + 0.5\mu Ne(k)s(k) \end{aligned} \tag{93}$$

ここで，$\bar{s}(k)$ は $s(k)$ のビットパターンを反転した信号を表す．式 (92) と (93) より，適応関数空間要素 $p0(k)$，$p1(k)$ はそれぞれ入力信号 $\bar{s}(k)$，$s(k)$ により更新されることがわかる．任意の入力信号に対する更新式は，式 (92) と式 (93) をまとめて次式のように表される．

$$\boldsymbol{P}_w(k+1) = \boldsymbol{P}_w(k)+0.5\mu e(k)\,\boldsymbol{S}_{DA}(k) \tag{94}$$

$$\boldsymbol{S}_{DA}(k) = N[\bar{s}(k), s(k)]^T \tag{95}$$

$$\boldsymbol{P}_w(k) = [p0(k), p1(k)]^T \tag{96}$$

これを LMS アルゴリズムと比較すると，$\boldsymbol{S}_{DA}(k)$ は入力信号ベクトルに相当することがわかる．この $\boldsymbol{S}_{DA}(k)$ は，適応関数空間を更新するために用いられる新たな入力信号ベクトルであり，これを拡張入力信号ベクトルと呼ぶことにする．

**[タップ数 2 における拡張]**

入力信号ベクトルを

$$\boldsymbol{S}(k) = [s(k), s(k-1)]^T \tag{97}$$

とする．ここで，信号 $s(k)$，$s(k-1)$ を構成するビットは'0' と'1' の 2 値を持つため，適応関数空間要素を指定するアドレスベクトル $\boldsymbol{A}_{vi}(k)$ の種類は，'0' と'1' の組み合わせの 4 種類存在する．それらを 表 11 のようにアルファベットで表して区別することにする．今，一

表 12: Access pattern of adaptive function space.

| Symbol | Bit pattern | $-2^0$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ |
|---|---|---|---|---|---|
| $\boldsymbol{a}$ | $[00]^T$ | 0 | 0 | 1 | 0 |
| $\boldsymbol{b}$ | $[01]^T$ | 0 | 1 | 0 | 0 |
| $\boldsymbol{c}$ | $[10]^T$ | 1 | 0 | 0 | 0 |
| $\boldsymbol{d}$ | $[11]^T$ | 0 | 0 | 0 | 1 |

例として入力信号の組み合わせが以下の場合について考える.

$$s(k) = 1 \times (-2^0) + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

$$s(k-1) = 0 \times (-2^0) + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

この入力信号ベクトルをシンボルを用いて表すと,

$$\boldsymbol{S}_s = \boldsymbol{c} \times (-2^0) + \boldsymbol{b} \times 2^{-1} + \boldsymbol{a} \times 2^{-2} + \boldsymbol{d} \times 2^{-3} \tag{98}$$

となる. なお, 語長を4ビットとし, 入力信号は4種類の全パターンが現れるビットパターンを選択した. この場合, 適応関数空間は提案法の更新式を用いて次のように更新される.

$$pa(k+1) = pa(k) + 0.5\mu \times 2 \times e(k) 2^{-2} \tag{99}$$

$$pb(k+1) = pb(k) + 0.5\mu \times 2 \times e(k) 2^{-1} \tag{100}$$

$$pc(k+1) = pc(k) - 0.5\mu \times 2 \times e(k) 2^{0} \tag{101}$$

$$pd(k+1) = pd(k) + 0.5\mu \times 2 \times e(k) 2^{-3} \tag{102}$$

式(99)∼ 式(102) をまとめると,

$$\boldsymbol{P}_w(k+1) = \boldsymbol{P}_w(k) + 0.5\mu N e(k)\, \boldsymbol{A}_{ac}(k)\, \boldsymbol{F} \tag{103}$$

となる. ここで, 各適応関数空間と更新に寄与するスケーリングベクトル要素との対応関係を 表 12 に示す. 式(103) の $\boldsymbol{A}_{ac}(k)$ は 表 12 に相当し, この場合は以下に示す$4 \times 4$のマ

トリクスである．

$$\boldsymbol{A}_{ac}(k)=\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix} \tag{104}$$

この行列は，入力信号ベクトルのビットパターンにより一意に決定される．これを，アクセスマトリクスと呼ぶことにする．また，適応関数空間 $\boldsymbol{P}_w(k)$ は

$$\boldsymbol{P}_w(k)=[pa(k),pb(k),pc(k),pd(k)]^T \tag{105}$$

である．式(103)を任意の入力信号ベクトルに対してタップ数 $N$ に一般化すると，

$$\boldsymbol{P}_w(k+1)=\boldsymbol{P}_w(k)+0.5\mu e(k)\,\boldsymbol{S}_{DA}(k) \tag{106}$$

$$\begin{aligned}\boldsymbol{S}_{DA}(k)&=N\,\boldsymbol{A}_{ac}(k)\,\boldsymbol{F} \qquad (107)\\&=[s_{DA,0}(k),\cdots,s_{DA,2^N-1}(k)]^T\end{aligned}$$

となる．ここで，$\boldsymbol{A}_{ac}^T(k)$ は $2^N\times B$ のアクセスマトリクス，

$$\boldsymbol{P}_w(k)=[p_0(k),p_1(k),\cdots,p_{2^N-1}(k)]^T \tag{108}$$

$$\boldsymbol{F}=[-2^0,2^{-1},\cdots,2^{-B+1}]^T \tag{109}$$

である．また，出力信号 $y(k)$ と誤差信号 $e(k)$ は

$$\begin{aligned}y(k)&=\boldsymbol{F}^T\,\boldsymbol{A}_{ac}^T(k)\,\boldsymbol{P}_w(k)\\&=\frac{1}{N}\,\boldsymbol{S}_{DA}^T(k)\,\boldsymbol{P}_w(k) \qquad (110)\end{aligned}$$

$$e(k)=d(k)-y(k) \tag{111}$$

となる．式(106)とLMSアルゴリズムの式(40)を比較すると，$\boldsymbol{S}_{DA}(k)$ はLMSアルゴリズムにおける入力信号ベクトル $\boldsymbol{S}(k)$ に相当しており，適応関数空間の更新値を決定する入力信号ベクトルである．

従来法の更新式も同様の過程により導かれるため，ここでは結果のみを示す．

$$\boldsymbol{P}'_w(k+1) = \boldsymbol{P}'_w(k) + 2\mu e'(k)\,\boldsymbol{S}'_{DA}(k) \tag{112}$$

$$\begin{aligned}\boldsymbol{S}'_{DA}(k) &= N\,\boldsymbol{A}'_{ac}(k)\,\boldsymbol{F}' \\ &= [s'_{DA,0}(k), \cdots, s'_{DA,2^N-1}(k)]^T\end{aligned} \tag{113}$$

ただし，入力信号の各ビットは値'0'，'1'である．ここで，$\boldsymbol{A}'^T_{ac}(k)$は$2^N \times B$のアクセスマトリクス，

$$\boldsymbol{P}'_w(k) = [p'_0(k), p'_1(k), \cdots, p'_{2^N-1}(k)]^T \tag{114}$$

$$\boldsymbol{F}' = [2^{-1}, 2^{-2}, \cdots, 2^{-B}]^T \tag{115}$$

である．また，出力信号$y'(k)$と誤差信号$e'(k)$は

$$\begin{aligned}y'(k) &= \boldsymbol{F}'^T\,\boldsymbol{A}'^T_{ac}(k)\,\boldsymbol{P}'_w(k) \\ &= \frac{1}{N}\boldsymbol{S}'^T_{DA}(k)\,\boldsymbol{P}'_w(k)\end{aligned} \tag{116}$$

$$e'(k) = d(k) - y'(k) \tag{117}$$

となる．

これまで，入力信号は単に適応関数空間の要素を指定するために用いていると解釈されていた．しかし，更新式を全適応関数空間に拡張することにより，分散演算型LMS適応フィルタは式(106)，式(108)と式(112)，式(113)のように，入力信号$\boldsymbol{S}_{DA}(k)$，$\boldsymbol{S}'_{DA}(k)$を用いて適応関数空間を更新することが陽に示された．

## 5.3 収束条件の導出

DA適応フィルタの収束条件は以下のようにして求められる[19, 20, 28, 29]．式(110)を式(106)に代入して次式が得られる．

$$\boldsymbol{P}_w(k+1) = [\,\boldsymbol{I} - 0.5\frac{1}{N}\mu\,\boldsymbol{S}_{DA}(k)\,\boldsymbol{S}^T_{DA}(k)]\,\boldsymbol{P}_w(k) + 0.5\mu d(k)\,\boldsymbol{S}_{DA}(k) \tag{118}$$

ここで，行列$\boldsymbol{I}$は$2^N \times 2^N$の単位行列である．さて，適応関数空間の最適値を$\boldsymbol{P}_w^*$とし，適応関数空間誤差ベクトル$\boldsymbol{c}(k)$を次のように定義する．

$$\begin{aligned}\boldsymbol{c}(k) &= \boldsymbol{P}_w(k) - \boldsymbol{P}_w^* &(119)\\ \boldsymbol{P}_w^* &= N\,\boldsymbol{R}^{-1}\,\boldsymbol{q} &(120)\end{aligned}$$

なお，最適値$\boldsymbol{P}_w^*$はDAアルゴリズムの正規方程式

$$\boldsymbol{q} = \frac{1}{N}\,\boldsymbol{R}\,\boldsymbol{P}_w^* \tag{121}$$

より得られる [20]．なお，正規方程式の導出過程を付録Bに示す．ここで，$\boldsymbol{R}$は次式で定義される拡張入力信号ベクトルの自己相関行列

$$\boldsymbol{R} = E[\,\boldsymbol{S}_{DA}(k)\,\boldsymbol{S}_{DA}^T(k)] \tag{122}$$

そして，$\boldsymbol{q}$は次式で定義される所望信号と拡張入力信号ベクトルの相互相関ベクトルである．

$$\boldsymbol{q} = E[d(k)\,\boldsymbol{S}_{DA}(k)] \tag{123}$$

また，$\boldsymbol{R}^{-1}$は$\boldsymbol{R}$の逆行列を表し，$\boldsymbol{P}_w^*$は誤差信号の最小2乗平均の意味で最適である．

これらの関係を用いて，式(118)は次のように表される．

$$\begin{aligned}\boldsymbol{c}(k+1) = [\,&\boldsymbol{I} - 0.5\frac{1}{N}\mu\,\boldsymbol{S}_{DA}(k)\,\boldsymbol{S}_{DA}^T(k)]\,\boldsymbol{c}(k)\\ &+0.5\mu[d(k)\,\boldsymbol{S}_{DA}(k) - \frac{1}{N}\,\boldsymbol{S}_{DA}(k)\,\boldsymbol{S}_{DA}^T(k)\,\boldsymbol{P}_w^*]\end{aligned} \tag{124}$$

式(124)の期待値は，$\boldsymbol{c}(k)$と$\boldsymbol{S}_{DA}(k)$の独立性より

$$\begin{aligned}E[\,\boldsymbol{c}(k+1)] &= [\,\boldsymbol{I} - 0.5\frac{1}{N}\mu\,\boldsymbol{R}]E[\,\boldsymbol{c}(k)]\\ &\quad +0.5\mu[\,\boldsymbol{q} - \frac{1}{N}\,\boldsymbol{R}\,\boldsymbol{P}_w^*]\end{aligned} \tag{125}$$

となる．式(121)を式(125)に代入して，

$$\begin{aligned}E[\,\boldsymbol{c}(k+1)] &= [\,\boldsymbol{I} - 0.5\frac{1}{N}\mu\,\boldsymbol{R}]E[\,\boldsymbol{c}(k)]\\ &= [\,\boldsymbol{I} - \mu_a\,\boldsymbol{R}]E[\,\boldsymbol{c}(k)]\end{aligned} \tag{126}$$

と簡略化される．なお，

$$\mu_a = \frac{0.5}{N}\mu \tag{127}$$

とおいた．この式は適応関数空間誤差ベクトル $\boldsymbol{c}(k)$ の更新式を表しており，時刻 $k$ が経過するにつれて $\boldsymbol{c}(k)$ が減少するかどうかは $\mu_a$ および $\boldsymbol{R}$ に依存していることがわかる．この性質を明確にするために，$\boldsymbol{R}$ を次のように変形する．

$$\boldsymbol{R} = \boldsymbol{Q}\,\boldsymbol{D}\,\boldsymbol{Q}^T \tag{128}$$

ただし，$\boldsymbol{Q}$ は $\boldsymbol{R}$ の固有ベクトルを列ベクトルに持つ直交行列

$$\boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$$

である．また，$\boldsymbol{D}$ は $\boldsymbol{R}$ の固有値を対角要素とする対角行列

$$\boldsymbol{D} = Diag(\lambda_1, \lambda_2, \cdots, \lambda_{2^N})$$

である．ここで，$\lambda_i, (i = 1, \cdots, 2^N)$ は $\boldsymbol{R}$ の固有値を表す．これより，式 (126) は

$$E[\,\boldsymbol{c}(k+1)] = \boldsymbol{Q}[\,\boldsymbol{I} - \mu_a \boldsymbol{D}]\,\boldsymbol{Q}^T E[\,\boldsymbol{c}(k)] \tag{129}$$

となり，時刻 $k$ が経過するにつれて式 (129) が収束するための条件は

$$0 < \mu_a < \frac{1}{\lambda_{max}} \tag{130}$$

となる．ここで，$\lambda_{max}$ は $\boldsymbol{R}$ の最大固有値である．

従来法の収束条件も提案法と同様の過程で導かれるため，ここでは結果のみを示す．適応関数空間誤差ベクトルの平均値 $E[\,\boldsymbol{c}'(k)]$ の更新式は，

$$\begin{aligned} E[\,\boldsymbol{c}'(k+1)] &= [\,\boldsymbol{I} - 2\frac{1}{N}\mu'\,\boldsymbol{R}']E[\,\boldsymbol{c}'(k)] \\ &= [\,\boldsymbol{I} - \mu'_a\,\boldsymbol{R}']E[\,\boldsymbol{c}'(k)] \end{aligned} \tag{131}$$

である．ここで，

$$\boldsymbol{c}'(k) = \boldsymbol{P}'_w(k) - \boldsymbol{P}'^*_w \tag{132}$$

$$\mu'_a = \frac{2}{N}\mu' \tag{133}$$

である．式(131)より収束条件は，

$$0 < \mu'_a < \frac{1}{\lambda'_{max}} \tag{134}$$

となる．ここで，$\lambda'_{max}$は$\boldsymbol{R}'$の最大固有値である．

提案法と従来法のいずれの場合も，新たに定義した拡張入力信号ベクトルの自己相関行列$\boldsymbol{R}$，$\boldsymbol{R}'$の固有値分布が小さいほど高速な収束速度を示すため，提案法や従来法の固有値分布を検証することにより収束速度を評価することができる．

## 5.4 自己相関行列の固有値と収束速度

### 5.4.1 提案法の固有値

以下の議論では，入力信号$s(k)$は定常過程であり，平均値は0，連続する信号サンプルは無相関，さらに，信号サンプルを構成するビットは互いに無相関であると仮定する [16]．これより，式(107)で表される拡張した入力信号ベクトルにおいて，アクセスマトリクス$\boldsymbol{A}^T_{ac}(k)$の行には値'1'を持つ要素がランダムにただ一つ生起し，また各行は無相関である．これより，$s_{DA,i}(k)$はアクセスマトリクスの第$j$行に対応して生起する信号$s_{i,j}(k)$の和と考えることができる．ここで，信号$s_{i,j}(k)$の値はビットが'1'と'0'の値を持つ場合にわけられ，生起確率をそれぞれ$Pr_1$，$Pr_0$とすると式(107)より

$$s_{i,j}(k) = \begin{cases} N \times \boldsymbol{F}_j & \text{生起確率}\, Pr_1 \\ 0 & \text{生起確率}\, Pr_0 \end{cases}$$

である．生起確率$Pr_1$は行ベクトルの$2^N$個の要素のうち唯一の要素が'1'となる確率であり，$Pr_0$は'1'とならない確率であるため

$$\begin{aligned} Pr_1 &= \frac{1}{2^N} \\ Pr_0 &= 1 - Pr_1 = \frac{2^N - 1}{2^N} \end{aligned}$$

である．

さて，自己相関行列の対角要素の平均値と分散を求める．まず，第1列により決まる $s_{DA,i}(k)$ の平均値 $ave_0$ と分散 $var_0$ は，

$$
\begin{aligned}
ave_0 &= (-2^0 \times Pr_1 + 0 \times Pr_0) \times N \\
var_0 &= (-2^0 \times N - ave_0)^2 \times Pr_1 + (0 \times N - ave_0)^2 \times Pr_0
\end{aligned}
$$

である．次に，第2列により決まる平均値 $ave_1$ と分散 $var_1$ は，

$$
\begin{aligned}
ave_1 &= (2^{-1} \times Pr_1 + 0 \times Pr_0) \times N \\
var_1 &= (2^{-1} \times N - ave_1)^2 \times Pr_1 + (0 \times N - ave_1)^2 \times Pr_0
\end{aligned}
$$

となる．以下，第3から第 $2^N$ 列ベクトルについても同様に求められるため，ここでは省略する．拡張した入力信号ベクトルの各要素 $s_{DA,i}(k)$ は，これら各列ベクトルにより生起するランダム信号の和と見なせるため，中心極限定理より平均値 $ave$ と分散 $var$ は

$$ave = \sum_{j=0}^{B-1} ave_j \approx 0 \tag{135}$$

$$var = \sum_{j=0}^{B-1} var_j \tag{136}$$

となる [40]．

次に，$s_{DA,i}(k)$ と $s_{DA,j}(k)$，$i \neq j$ の相関値を求める．ここで，提案法の拡張入力信号ベクトルの要素には式 (107) より次の関係が成立する．

$$s_{DA,0}(k) + s_{DA,1}(k) + \cdots + s_{DA,2^N-1}(k) \approx 0 \tag{137}$$

いま，$s_{DA,0}(k)$ と $s_{DA,1}(k)$ の相互相関値を求めるために，式 (137) を $s_{DA,0}(k)$ について解き，両辺に $s_{DA,1}(k)$ を掛けると，

$$s_{DA,0}(k)s_{DA,1}(k) \approx -\{s_{DA,1}(k)s_{DA,1}(k) + \cdots + s_{DA,2^N-1}(k)s_{DA,1}(k)\}$$

となり両辺の期待値を求めると，

$$E[s_{DA,0}(k)s_{DA,1}(k)] \approx -E[s_{DA,1}(k)s_{DA,1}(k)] - \cdots - E[s_{DA,2^N-1}(k)s_{DA,1}(k)] \tag{138}$$

となる．ここで，式(107)の各要素はアクセスマトリクス$\boldsymbol{A}_{ac}^{T}(k)$のパターンにより決定されるが，それらのパターンは等確率で生起するため，相互相関$cor$は

$$E[s_{DA,i}(k)s_{DA,j}(k)] = cor,\ i \neq j$$

である．これより式(138)は,

$$cor \approx -var - (2^N - 2) \times cor$$

となり，これを$cor$について求めると

$$cor \approx -\frac{var}{2^N - 1} \tag{139}$$

となる．以上より，$\boldsymbol{R}$は対角要素に$var$，非対角要素に$cor$を持つ$2^N \times 2^N$行列である．

次に，自己相関行列の固有値を求める．$\boldsymbol{R}$を入力信号の線形性より次のように表す[41]．

$$\boldsymbol{R} = \boldsymbol{D} + \boldsymbol{Q}$$

ここで,

$$\boldsymbol{D} = diag[d_0, d_1, \cdots, d_{2^N-1}]$$

$$\boldsymbol{Q} = \begin{bmatrix} q & \cdots & q \\ \vdots & \ddots & \vdots \\ q & \cdots & q \end{bmatrix}^T$$

であり，$diag[\ ]$は行列の対角要素を表す．$\boldsymbol{R}$の対角要素と非対角要素はそれぞれ

$$E[s_{DA,i}(k)s_{DA,i}] = var$$
$$i = 0, 1, \cdots, 2^N - 1$$

そして,

$$cor \approx -\frac{var}{2^N - 1}$$

であるので，行列$\boldsymbol{D}$と$\boldsymbol{Q}$の要素はそれぞれ

$$\begin{aligned} d &= d_i \\ &= var + \frac{var}{2^N - 1} \\ &= \frac{2^N \times var}{2^N - 1}, \qquad i = 0, 1, \cdots, 2^N - 1 \\ q &= -\frac{var}{2^N - 1} \end{aligned}$$

である．対角行列$\boldsymbol{D}$の固有値は対角要素$d$に等しい．次に，行列$\boldsymbol{Q}$の対角要素の和であるトレース$tr[\boldsymbol{Q}]$は固有値の総和に等しく，

$$tr[\boldsymbol{Q}] = -2^N \times \frac{var}{2^N - 1} \tag{140}$$

であり，また階数$rank[\boldsymbol{Q}]$は非特異な部分行列のサイズであるため，

$$rank[\boldsymbol{Q}] = 1$$

である[41]．これより，行列$\boldsymbol{Q}$の固有値は1つの固有値が式(140)の$tr[\boldsymbol{Q}]$と等しく，その他の$2^N-1$個の固有値は0である．したがって，求める$\boldsymbol{R}$の固有値は

$$\begin{aligned} eig[\boldsymbol{R}] &= eig[\boldsymbol{D}] + eig[\boldsymbol{Q}] \\ &= [0, \frac{2^N \times var}{2^N - 1}, \cdots, \frac{2^N \times var}{2^N - 1}]^T \end{aligned} \tag{141}$$

となり，ここで

$$\frac{2^N \times var}{2^N - 1} = \frac{4}{3} N^2 (1 - 2^{-N+1} + 2^{-N})(2^N - 1)^{-1}$$

である．なお，$eig[\boldsymbol{R}]$は行列$\boldsymbol{R}$の固有値を表し，他の行列についても同様である．理論値と計算機シミュレーションによって自己相関行列$\boldsymbol{R}$を計算し，その固有値を求めた結果を 表 13 に示す．なお，括弧中の数は固有値の個数を表している．自己相関行列$\boldsymbol{R}$のサイズは$2^N \times 2^N$であるため，$2^N$個の固有値が存在する．入力信号$s(k)$は一様分布する平均0,

表 13: Comparison of eigenvalues for our proposed DA-ADF.

| Tap number | Theory | Simulation |
|---|---|---|
| 2 | 1.333(3),0.000(1) | 1.333(3),0.000(1) |
| 3 | 1.500(7),0.000(1) | 1.500(7),0.000(1) |
| 4 | 1.333(15),0.000(1) | 1.333(15),0.000(1) |
| 5 | 1.042(31),0.000(1) | 1.042(31),0.000(1) |

分散0.333の白色信号である．そして，語長16ビットの2の補数表現を用いて次のように表される．

$$s(k) = [b_0(k), b_1(k), \cdots, b_{15}(k)]\,\boldsymbol{F}$$
$$\boldsymbol{F} = [-2^0, 2^{-1}, \cdots, 2^{-15}]^T$$

自己相関行列は，100回の独立試行（1試行あたり$10^5$回信号を発生させている）の結果である．$\boldsymbol{R}$の固有値は，サイバネットシステム（株）社製 MATLAB ver.6.0を用いて求めた．これより，理論値と計算機シミュレーション結果はよく一致しており，1つの固有値が0になり，他の$2^N-1$個の固有値は全て同じ値を有している．そして，表 13 の0の固有値は理論的に0であるため，$\boldsymbol{R}$の階数は$2^N-1$になり，提案法における固有値は全て等しいことになる．以上より，提案法においては選択したステップサイズパラメータは全ての固有値に対して同等の収束速度を保証するため，良好な収束速度を達成することが可能である．

### 5.4.2 従来法の固有値

従来法の自己相関行列の固有値は，提案法と同様の過程で求めることができるため，ここでは結果のみを示す．$s'_{DA,i}(k)$の平均値$ave'$と分散$var'$を以下に示す．

$$ave' \approx N \times 2^{-N} \tag{142}$$
$$var' = \sum_{j=0}^{B-1} var'_j \tag{143}$$

このように，従来法における拡張した入力信号ベクトル要素は平均値$ave'(\neq 0)$を有する．

これより，$cor'$の非対角要素は

$$cor' \approx -\frac{var' + ave'^2 - N \times ave'}{2^N - 1} \tag{144}$$

となる．したがって，$\boldsymbol{R}'$は対角要素に$var' + ave'^2$，非対角要素に$ccor'$を持つ$2^N \times 2^N$行列であり，$\boldsymbol{R}'$の固有値は

$$\begin{aligned} eig[\boldsymbol{R}'] &= eig[\boldsymbol{D}'] + eig[\boldsymbol{Q}'] &(145)\\ &= [d', \cdots, d']^T + [2^N \times q', 0, \cdots, 0]^T \\ &= [d' + 2^N \times q', d', \cdots, d']^T \\ &= [N \times ave', d', \cdots, d']^T &(146) \end{aligned}$$

となる．ここで，

$$N \times ave' = N^2 \times 2^{-N} \tag{147}$$

$$d' = \frac{1}{3}N^2(1 - 2^{-N+1} + 2^{-N})(2^N - 1)^{-1} \tag{148}$$

である．理論値と計算機シミュレーションによって$\boldsymbol{R}'$を計算し，その固有値を求めた結果を 表 14 に示す．なお，括弧中の数は固有値の個数を表している[13]．入力信号$s'(k)$は一様分布する平均0，分散0.333の白色信号である．そして，語長16 ビットのオフセットバイナリ形式を用いて次のように表される．

$$s'(k) = [b'_0(k), b'_1(k), \cdots, b'_{15}(k)]\,\boldsymbol{F}'$$

$$\boldsymbol{F}' = [2^{-1}, 2^{-2}, \cdots, 2^{-16}]^T$$

自己相関行列は，100回の独立試行（1試行あたり$10^5$回信号を発生させている）の結果である．$\boldsymbol{R}'$の固有値は，サイバネットシステム（株）社製 MATLAB ver.6.0を用いて求めた．これより，理論値と計算機シミュレーション結果はよく一致しており，1つの固有値が大きく，他の$2^N-1$個の固有値は全て同じ値を有している．この原因は，$s'_{DA,i}(k)$の平均値が0とはならずに，式(142)で示されるオフセットを有するためである．

[13]自己相関行列$\boldsymbol{R}'$のサイズは$2^N \times 2^N$であるので，$2^N$個の固有値が存在する．

表 14: Comparison of eigenvalues for the conventional method.

| Tap number | Theory | Simulation |
|---|---|---|
| 2 | 0.333(3) ,1.000(1) | 0.333(3) ,1.000(1) |
| 3 | 0.375(7) ,1.125(1) | 0.375(7) ,1.125(1) |
| 4 | 0.333(15),1.000(1) | 0.333(15),1.000(1) |
| 5 | 0.260(31),0.781(1) | 0.260(31),0.781(1) |

### 5.4.3 収束条件式による収束速度

収束条件式を用いて収束速度を評価する．図 28 に収束特性を示す．なお，タップ数は 4，ステップサイズパラメータは

$$\mu = 0.1 \times \frac{1}{\lambda_{max}}$$
$$\mu' = 0.1 \times \frac{1}{\lambda'_{max}}$$

である．これより，従来法は提案法より収束速度が劣化していることがわかる．

以上より，従来法の収束速度が極端に劣化する原因は，入力信号の符号化用いたオフセットバイナリ形式が原因となり入力信号にオフセットバイアスが加わるため，拡張入力信号ベクトルの自己相関行列の固有値が広く分布するためである．一方，提案法ではこのような問題は発生せず，全ての固有値が等しいため良好な収束速度を有することが示された．

## 5.5 まとめ

本章では，DA アルゴリズムの収束条件を理論的に解析した．解析のために，

(1) DA アルゴリズムを全空間を対象にする式に拡張し，新たに拡張入力信号ベクトルを定義し，

(2) 推定誤差を WAFS の最適値と推定値の差として定義し，

(3) 収束条件を，推定誤差が繰り返しとともに減少するための条件として定義した．

その結果，アルゴリズムの収束は拡張入力信号ベクトルの自己相関行列の固有値とステップ

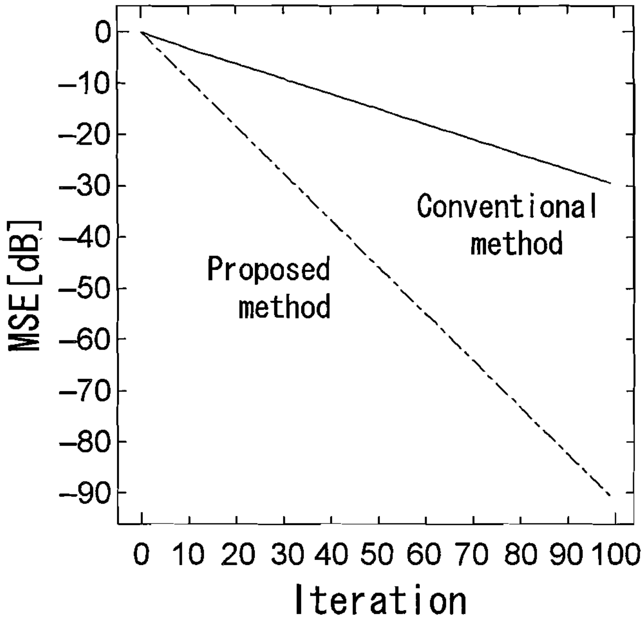



図 28: Comparison of convergence speed using convergence equation.

サイズパラメータに依存することを示した．そして，DA アルゴリズムの拡張により，DA アルゴリズムは拡張入力信号ベクトルを用いて更新されるという新たな解釈を示した．さらに，自己相関行列の固有値を理論的に示した．その結果，提案法の固有値は全て等しく，良好な収束速度を有することがわかった．一方，従来法の固有値はひとつの固有値が非常に大きくなることがわかった．これは，入力信号の符号化にオフセットバイナリ形式を用いているために，符号化した入力信号にオフセットが加わるためである．これより，従来法の収束速度は極端に劣化する．

# 第6章　分散演算形ブロックLMS適応フィルタ

## 6.1　はじめに

近年，テレビ電話やテレビ会議システムなどのオーディオビジュアル(Audiovisual，AV)通信システムなどを含むマルチメディア通信システムの構築が進んでおり，ディジタル信号処理はマルチメディアを支える中心技術となっている[44, 45, 46]．マルチメディアに使用される通信ネットワークの一つであるブロードバンドISDNは，140Mbit/sという高速な通信網であるため，これらの高速通信網に対するディジタル信号処理システムに対しては，特に高速性が要求される．

これまで提案してきた分散演算形LMS適応フィルタは，高次においても高速性と極めて小さい滞在時間を維持した上で，低消費電力，低ハードウェア量を実現可能な高性能適応フィルタである．しかし，サンプリングレートは高々数MHz程度であるため，マルチメディア通信のような超高速通信網に対しては適用範囲が限定される．

LMSアルゴリズムの高速アルゴリズムとしてブロックLMS(BLMS)アルゴリズムが提案されている[30, 31, 38]．通常のLMSアルゴリズムは入力信号をサンプリング時刻ごとに処理するのに対して，BLMSアルゴリズムは$L$サンプリング時刻ごとに$L$個の入力信号ベクトルを並列に処理するため，1サンプルあたりの処理時間を短縮することが可能である．$Clark$らは，演算量の削減を目的にアルゴリズムを時間領域から周波数領域に変換して用いた．しかし，入力信号やパラメータを周波数領域に変換するために高速フーリエ変換を用いることになり，タップ数が増加するにしたがい出力滞在時間が急激に増加するという問題点がある．

本章では，分散演算形ブロックLMS適応フィルタを初めて提案する．ブロックLMSアルゴリズムは，LMSアルゴリズムの並列実現であるため，時間領域におけるアルゴリズムは本質的に高度な並列性を有している．この点に着目して，分散演算の適用は時間領域のブロックLMSアルゴリズムに対して行う．まず，分散演算をブロックLMS適応フィルタに適用することにより分散演算形ブロックLMSアルゴリズム（BDAアルゴリズム）を

導出し，次いで，マルチメモリブロック構造を適用したアルゴリズム（MBDA アルゴリズム）を導出する．そして，これらのアルゴリズムにおいて，2種類の更新方法，すなわち通常の更新方法とパイプライン処理に適した更新方法(プライオリティ・アップデート)を提案する．MBDA アルゴリズムの収束特性を計算機シミュレーションで評価した結果，マルチメモリブロック構成の分散演算形 LMS アルゴリズム(MDA アルゴリズム)とほぼ同等の収束特性を有することが明らかになった．さらに，プライオリティアップデートを用いた MBDA 適応フィルタの効果的な VLSI アーキテクチャを検討した．その結果，提案するアーキテクチャは高速なサンプリングレートを有し，しかも出力滞在時間が短いことが明らかになった．

## 6.2 ブロック LMS アルゴリズム

### 6.2.1 LMS アルゴリズム

Widrow らの LMS(Least mean square) アルゴリズムを以下に再記する [37]．入力信号を $x(k)$ とすると $p$ 次入力信号ベクトル $\boldsymbol{\varphi}(k)$ は

$$\boldsymbol{\varphi}(k) = [x(k), x(k-1), \cdots, x(k-p+1)]^T \tag{149}$$

と表される．次に，$p$ タップ FIR フィルタの出力 $y(k)$ は次式で求められる．

$$y(k) = \boldsymbol{\varphi}^T(k)\, \boldsymbol{w}(k) \tag{150}$$

ここで，$\boldsymbol{w}(k)$ は FIR フィルタのタップ係数で

$$\boldsymbol{w}(k) = [w_0(k), w_1(k), \cdots, w_{p-1}(k)]^T \tag{151}$$

である．誤差信号を $e(k)$ とすると，LMS アルゴリズムは次の様に表される．

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + 2\mu e(k)\, \boldsymbol{\varphi}(k) \tag{152}$$

なお，$d(k)$ は所望信号を表し，

$$e(k) = d(k) - y(k) \tag{153}$$

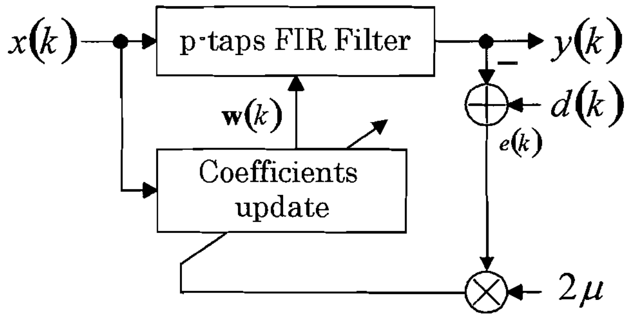



図 29: Structure of LMS adaptive filter.

である．LMSアルゴリズムはサンプリング時刻$k$ごとに出力計算と更新動作を実行する．LMS適応フィルタの基本構成を 図 29 に示す．

### 6.2.2 ブロックLMSアルゴリズムの導出

ブロックLMSアルゴリズムでは，$L$サンプル時刻毎に$L$個の入力信号ベクトルを並列に処理することにより，1入力サンプルあたりのサンプリング周期を$1/L$に短縮することが可能になる．ブロック長$L$，ブロック番号$j$，タップ数$p$に対して，入力信号ベクトル$\boldsymbol{\varphi}_{j,i}$を次の様に表す．

$$\boldsymbol{\varphi}_{j,i} = \left[x_{j,i}, x_{j,(i-1)}, \cdots, x_{j,(i-p+1)}\right]^T$$

ここで，サンプリング時刻$k$は，

$$k = jL + i, \quad i = 0, -1, \cdots, -L+1$$

であり，$i$はブロック内における時刻を表す．BLMSアルゴリズムの入力信号マトリクスは次の様に表される．

$$\boldsymbol{\Gamma}_j = \left[\boldsymbol{\varphi}_{j,0}, \boldsymbol{\varphi}_{j,(-1)}, \cdots, \boldsymbol{\varphi}_{j,(-L+1)}\right]^T$$

$$
= \begin{bmatrix} x_{j,0} & x_{j,(-1)} \cdots & x_{j,(-L+1)} \\ x_{j,(-1)} & x_{j,(-2)} \cdots & x_{j,(-L)} \\ \vdots & \vdots \quad \ddots & \vdots \\ x_{j,(-p+1)} & x_{j,(-p)} \cdots & x_{j,(-L-p+2)} \end{bmatrix}^T
$$

出力信号$\boldsymbol{y}_j$と誤差信号$\boldsymbol{e}_j$は，次式で求められる．

$$
\begin{aligned} \boldsymbol{y}_j &= \boldsymbol{\Gamma}_j \boldsymbol{w}_j \\ \boldsymbol{e}_j &= \boldsymbol{d}_j - \boldsymbol{y}_j \end{aligned}
$$

ここで，出力信号$\boldsymbol{y}_j$,所望信号$\boldsymbol{d}_j$,誤差信号$\boldsymbol{e}_j$そして，タップ係数$\boldsymbol{w}_j$は

$$
\boldsymbol{y}_j = [y_{j,0}, y_{j,(-1)}, \cdots, y_{j,(-L+1)}]^T \tag{154}
$$

$$
\boldsymbol{d}_j = [d_{j,0}, d_{j,(-1)}, \cdots, d_{j,(-L+1)}]^T \tag{155}
$$

$$
\boldsymbol{e}_j = [e_{j,0}, e_{j,(-1)}, \cdots, e_{j,(-L+1)}]^T \tag{156}
$$

$$
\boldsymbol{w}_j = [w_j(0), w_j(1), \cdots, w_j(-p+1)]^T \tag{157}
$$

である．これらより，ブロックLMSアルゴリズムは次のように表される．

$$
\begin{aligned} \boldsymbol{w}_{(j+1)} &= \boldsymbol{w}_j + \frac{2\mu_B}{L} \boldsymbol{\Gamma}_j^T \boldsymbol{e}_j \\ &= \boldsymbol{w}_j + \frac{2\mu_B}{L} \boldsymbol{q}_j \end{aligned}
$$

ここで，$\mu_B$は収束速度と推定精度を決定するステップサイズパラメータ，そして

$$
\begin{aligned} \boldsymbol{q}_j &= [q_{j,0}, q_{j,(-1)}, \cdots, q_{j,(-L+1)}]^T \\ &= \boldsymbol{\Gamma}_j^T \boldsymbol{e}_j \\ &= \sum_{i=0}^{-L+1} \boldsymbol{\varphi}_{j,i} e_{j,i} \end{aligned}
$$

である．ブロック長$L = 3$のBLMS-ADFの基本構成を 図 30 に示す．新たな信号が$L$サンプル得られた時点で$L$次入力信号ベクトル$\boldsymbol{\Gamma}_j$が確定し，アルゴリズムは動作を開始す

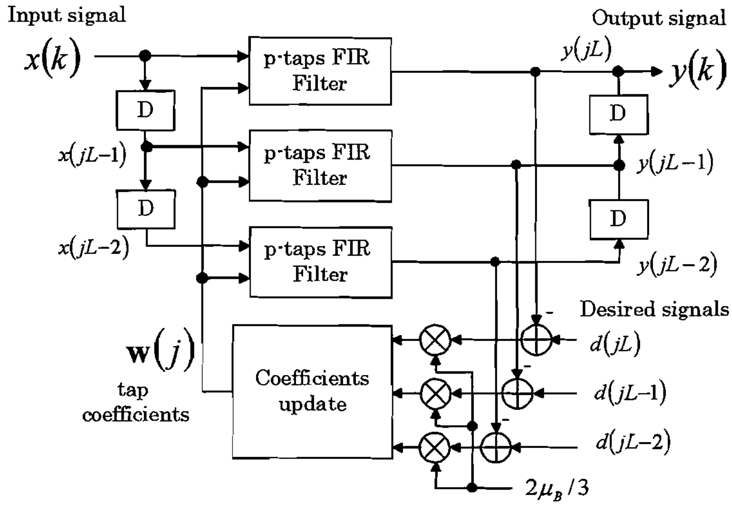



図 30: Block diagram of BLMS-ADF with L=3.

る．まず，$L$ 個の FIR フィルタが並列に動作し，$L$ 個の出力信号が同時に得られる．次いで，誤差信号の計算とスケーリングがそれぞれ並列に実行される．得られた $L$ 個の誤差信号と入力信号ベクトルを用いて更新値が求められ，タップ係数が更新される．なお，タップ係数は各 FIR フィルタに共通であるため，更新動作は (a) $L$ 個の更新値ベクトルの和を求めた後にタップ係数を更新する，(b) タップ係数と更新値ベクトルの和を $L$ 回繰り返す方法が考えられる．次に，LMS-ADF と BLMS-ADF($L=3$) の動作タイミングを 図 31 に示す．通常の LMS-ADF はサンプリング時刻ごとに出力計算と係数更新動作を行うのに対して，BLMS-ADF は 3 サンプリング時刻ごとに適応動作を並列に実行する．したがって，BLMS-ADF は 1 サンプルあたりの処理時間を 1/3 に短縮することが可能になる．

## 6.3 分散演算形ブロック LMS アルゴリズム

### 6.3.1 分散演算形 LMS アルゴリズム

分散演算は定係数ベクトルの内積演算を効率よく求める手法としてよく知られていが，係数が変化する適応フィルタにおいても有効である [18, 15, 25, 16]．分散演算における内積

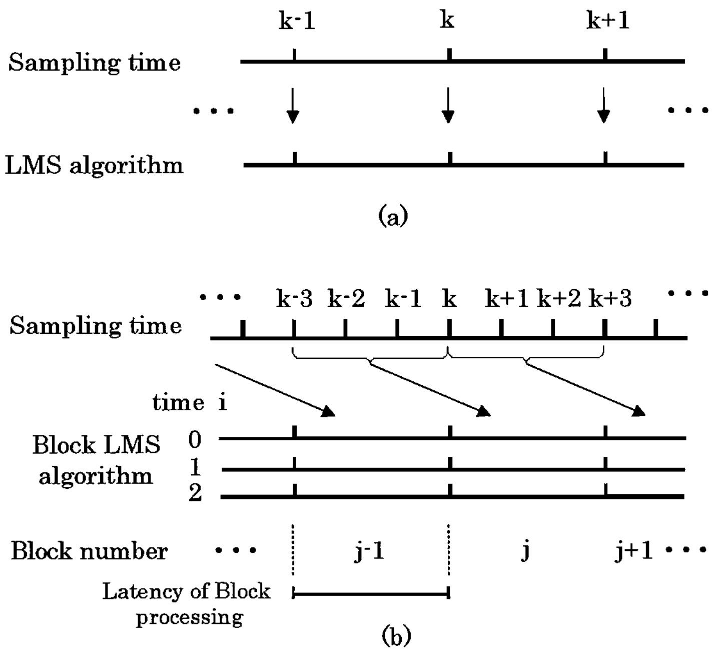



図 31: Comparison of processing timing between LMS and block LMS algorithm. (a) LMS algorithm, (b) Block LMS algorithm with L=3.

演算は部分積のシフト加算により実行されるが，$p$次ベクトルの内積演算における部分積数は，そのパターン数に応じた$2^p$個である．この集合を全適応関数空間と呼び，WAFS(Whole Adaptive Function Space)と表すことにする．WAFSは，定係数の分散演算ではあらかじめ決定されるが，適応フィルタでは逐次的に最適値を推定する．LMSアルゴリズムに分散演算を適用した分散演算形LMS適応フィルタ(DA-ADF)の基本構成を 図 32 に示す．出力信号$y(k)$は，WAFS(RAMを用いて実現される)から指定された部分積を語長回数だけ読み出し，これらをシフト加算することにより得られる．そして，WAFSの部分積はスケーリングされた誤差信号を用いて更新される．この際，スケーリング値を2のべき乗で近似することにより，乗算器を用いない構成が可能になる．なお，WAFSの要素を指定するアドレス信号には，入力信号ベクトルのビットパターンから構成される$p$次のアドレスベクトルを用いる．更新式と出力計算式を以下に示す．

第3章で示した分散演算形LMSアルゴリズムを以下に再記する．適応関数空間$\boldsymbol{P}(k)$の

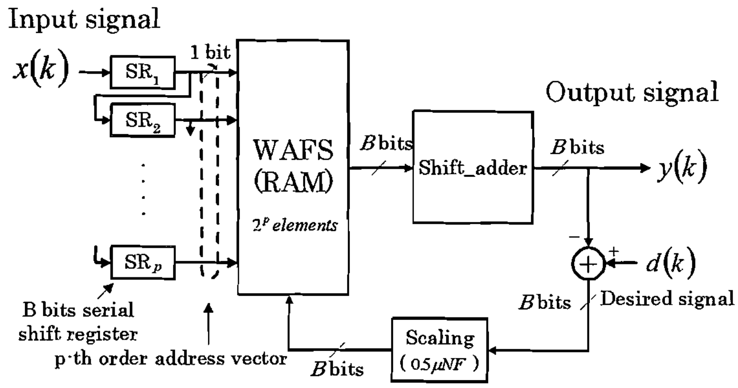



図 32: Block diagram of DA-ADF.

更新式と出力計算式は

$$\boldsymbol{P}(k+1) = \boldsymbol{P}(k) + 0.5\mu e(k)\,\boldsymbol{F} \tag{158}$$

$$y(k) = \boldsymbol{F}^T \boldsymbol{P}(k) \tag{159}$$

である．ここで，$\boldsymbol{P}(k)$は次に示す$B$次のベクトルで

$$\boldsymbol{P}(k) = [p_0(k), p_1(k), \cdots, p_{B-1}(k)]^T \tag{160}$$

スケーリングベクトル$\boldsymbol{F}$は

$$\boldsymbol{F} = [-2^0, 2^{-1}, \ldots, 2^{-(B-1)}]^T \tag{161}$$

誤差信号$e(k)$は

$$e(k) = d(k) - y(k) \tag{162}$$

である．

### 6.3.2 分散演算形ブロックLMSアルゴリズムの導出

分散演算形ブロックLMSアルゴリズム(BDA)は，通常のBLMSアルゴリズムに分散演算を適用して以下のように求められる．$p$次の入力信号ベクトル$\boldsymbol{\varphi}_{j,i}$を次のように表す．

$$\boldsymbol{\varphi}_{j,i} = \boldsymbol{A}_{j,i}\boldsymbol{F}, \quad i = 0, -1, \cdots, -L+1 \tag{163}$$

$\boldsymbol{A}_{j,i}$は入力信号のビットパターンを要素に持つアドレスマトリクスであり，$\boldsymbol{F}$はスケーリングベクトルである．すなわち，

$$\boldsymbol{A}_{j,i} = \begin{bmatrix} b_{j,i}(0) & \cdots & b_{j,(i-p+1)}(0) \\ b_{j,i}(1) & \cdots & b_{j,(i-p+1)}(1) \\ \vdots & \ddots & \vdots \\ b_{j,i}(B-1) & \cdots & b_{j,(i-p+1)}(B-1) \end{bmatrix}^T$$

$$\boldsymbol{F} = [-2^0, 2^{-1}, \cdots, 2^{-(B-1)}]^T$$

である．ここで$b_{j,i}(l)$は，ブロック$j$の時刻$i$における入力信号$x_{j,i}$の第$l$ビットを表す．なお，$l = 0, 1, \cdots, B-1$である．また，アドレスマトリクスの列ベクトル

$$\boldsymbol{Av}_{j,i}(l) = [b_{j,i}(l), b_{j,(i-1)}(l), \cdots, b_{j,(i-p+1)}(l)]^T, \quad l = 0, 1, \cdots, B-1$$

をアドレスベクトルと呼び，その値を次のように定義する．

$$\begin{aligned} Av_{j,i}(m) &\triangleq \boldsymbol{Av}_{j,i}^T(m)\, \boldsymbol{F}_A \\ \boldsymbol{F}_A &\triangleq [2^{(p-1)}, 2^{(p-2)}, \cdots, 2^0]^T \end{aligned}$$

分散演算における内積演算では，アドレスベクトルに対して部分積が定義されるため，アドレスベクトルは出力計算と更新動作においてWAFSの要素を指定するために用いられる．これらの関係を用いて，通常のBLMSアルゴリズムは次のように表される．

$$\boldsymbol{w}_{(j+1)} = \boldsymbol{w}_j + \frac{2\mu_B}{L} \sum_{i=0}^{-L+1} \boldsymbol{A}_{j,i}\, \boldsymbol{F} e_{j,i}$$

ブロック$j$内の時刻$i$=0,1,$\cdots$,$-L$+1について，この式を個別に表すと

$$\boldsymbol{w}_{j,(-L+2)} = \boldsymbol{w}_{j,(-L+1)} + \frac{2\mu_B}{L} \boldsymbol{A}_{j,(-L+1)} \boldsymbol{F} e_{j,(-L+1)} \tag{164}$$

$$\vdots$$

$$\boldsymbol{w}_{j,0} = \boldsymbol{w}_{j,(-1)} + \frac{2\mu_B}{L} \boldsymbol{A}_{j,(-1)} \boldsymbol{F} e_{j,(-1)} \tag{165}$$

$$\boldsymbol{w}_{(j+1),(-L+1)} = \boldsymbol{w}_{j,0} + \frac{2\mu_B}{L} \boldsymbol{A}_{j,0} \boldsymbol{F} e_{j,0} \tag{166}$$

となる．ここで，$\boldsymbol{w}_{j,i}$はブロック$j$の時刻$i$におけるタップ係数ベクトルであり次の関係を有する．

$$\boldsymbol{w}_{(j+1)} = \boldsymbol{w}_{(j+1),(-L+1)}$$

この式の両辺に左から$\boldsymbol{A}_{j,i}^T$を掛けると，式(164)から式(166)は

$$\begin{aligned}
\boldsymbol{A}_{j,(-L+1)}^T \boldsymbol{w}_{j,(-L+2)} &= \boldsymbol{A}_{j,(-L+1)}^T \boldsymbol{w}_{j,(-L+1)} + \frac{2\mu_B}{L} \boldsymbol{A}_{j,(-L+1)}^T \boldsymbol{A}_{j,(-L+1)} \boldsymbol{F} e_{j,(-L+1)} && (167)\\
&\vdots \\
\boldsymbol{A}_{j,(-1)}^T \boldsymbol{w}_{j,0} &= \boldsymbol{A}_{j,(-1)}^T \boldsymbol{w}_{j,(-1)} + \frac{2\mu_B}{L} \boldsymbol{A}_{j,(-1)}^T \boldsymbol{A}_{j,(-1)} \boldsymbol{F} e_{j,(-1)} && (168)\\
\boldsymbol{A}_{j,0}^T \boldsymbol{w}_{(j+1),(-L+1)} &= \boldsymbol{A}_{j,0}^T \boldsymbol{w}_{j,0} + \frac{2\mu_B}{L} \boldsymbol{A}_{j,0}^T \boldsymbol{A}_{j,0} \boldsymbol{F} e_{j,0} && (169)
\end{aligned}$$

となる．ここで，

$$\begin{aligned}
\boldsymbol{P}_{j,i}^i &\triangleq \boldsymbol{A}_{j,i}^T \boldsymbol{w}_{j,i} \\
&= [p_{j,i}(Av_{j,i}(0)), \cdots, p_{j,i}(Av_{j,i}(B-1))]^T \\
\boldsymbol{P}_{j,(i+1)}^i &\triangleq \boldsymbol{A}_{j,i}^T \boldsymbol{w}_{j,(i+1)} \\
&= [p_{j,(i+1)}(Av_{j,i}(0)), \cdots, p_{j,(i+1)}(Av_{j,i}(B-1))]^T
\end{aligned}$$

と定義する．なお，$\boldsymbol{P}_{j,i}^{i'}$はB次のベクトルで，アドレスマトリクス$\boldsymbol{A}_{j,i'}^T$に関する適応関数空間AFSである．これらの定義より，式(167)から式(169)は次のように表される．

$$\begin{aligned}
\boldsymbol{P}_{j,(-L+2)}^{(-L+1)} &= \boldsymbol{P}_{j,(-L+1)}^{(-L+1)} + \frac{2\mu_B}{L} \boldsymbol{A}_{j,(-L+1)}^T \boldsymbol{A}_{j,(-L+1)} \boldsymbol{F} e_{j,(-L+1)} && (170)\\
&\vdots \\
\boldsymbol{P}_{j,0}^{(-1)} &= \boldsymbol{P}_{j,(-1)}^{(-1)} + \frac{2\mu_B}{L} \boldsymbol{A}_{j,(-1)}^T \boldsymbol{A}_{j,(-1)} \boldsymbol{F} e_{j,(-1)} && (171)\\
\boldsymbol{P}_{(j+1),(-L+1)}^0 &= \boldsymbol{P}_{j,0}^0 + \frac{2\mu_B}{L} \boldsymbol{A}_{j,0}^T \boldsymbol{A}_{j,0} \boldsymbol{F} e_{j,0} && (172)
\end{aligned}$$

入力信号を平均0の白色信号と仮定すると，$\boldsymbol{A}_{j,i}^T \boldsymbol{A}_{j,i}$の平均値は

$$E[\boldsymbol{A}_{j,i}^T \boldsymbol{A}_{j,i}] = 0.25p\,\boldsymbol{F}$$

となり [24]，これを式 (170) から式 (172) に代入すると，次のように簡略化される．

$$\boldsymbol{P}^{(-L+1)}_{j,(-L+2)} = \boldsymbol{P}^{(-L+1)}_{j,(-L+1)} + \boldsymbol{u}_{j,(-L+1)} \tag{173}$$

$$\vdots$$

$$\boldsymbol{P}^{(-1)}_{j,0} = \boldsymbol{P}^{(-1)}_{j,(-1)} + \boldsymbol{u}_{j,(-1)} \tag{174}$$

$$\boldsymbol{P}^{0}_{(j+1),(-L+1)} = \boldsymbol{P}^{0}_{j,0} + \boldsymbol{u}_{j,0} \tag{175}$$

ここで，

$$\begin{aligned} \boldsymbol{u}_{j,i} &= 0.5p\frac{\mu_B}{L}\boldsymbol{F}e_{j,i} \\ &= [u_{j,i}(0), u_{j,i}(1), \cdots, u_{j,i}(B-1)]^T \\ & \quad i = 0, -1, \cdots, -L+1 \end{aligned}$$

である．これら $L$ 個の更新式は並列に実行され，共有する WAFS の要素をアドレスベクトルによって順次選択して更新する．また，誤差信号のスケーリング値を 2 のべき乗で近似することにより，乗算器を用いない構成が可能になる．なお，WAFS は次のように表される．

$$\boldsymbol{Pw}_{j,i} = [p_{j,i}(0), p_{j,i}(1), \cdots, p_{j,i}(2^p - 1)]^T$$

次に，BLMS アルゴリズムの出力方程式は

$$\begin{aligned} \boldsymbol{y}_j &= [y_{j,0}, y_{j,(-1)}, \cdots, y_{j,(-L+1)}]^T \\ &= [\boldsymbol{\varphi}^T_{j,0}\,\boldsymbol{w}_j,\ \boldsymbol{\varphi}^T_{j,(-1)}\,\boldsymbol{w}_j, \cdots,\ \boldsymbol{\varphi}^T_{j,(-L+1)}\,\boldsymbol{w}_j]^T \end{aligned}$$

であるが，式 (163) を適用して

$$\begin{aligned} \boldsymbol{y}_j &= [\,\boldsymbol{F}^T\boldsymbol{A}^T_{j,0}\,\boldsymbol{w}_j, \cdots,\ \boldsymbol{F}^T\boldsymbol{A}^T_{j,(-L+1)}\,\boldsymbol{w}_j]^T \\ &= [\,\boldsymbol{F}^T\boldsymbol{P}^0_{j,(-L+1)}, \cdots,\ \boldsymbol{F}^T\boldsymbol{P}^{(-L+1)}_{j,(-L+1)}]^T \end{aligned} \tag{176}$$

となる．ここで，

$$\boldsymbol{P}^i_j \triangleq \boldsymbol{P}^i_{j,(-L+1)}, \quad i = 0, -1, \cdots, -L+1$$

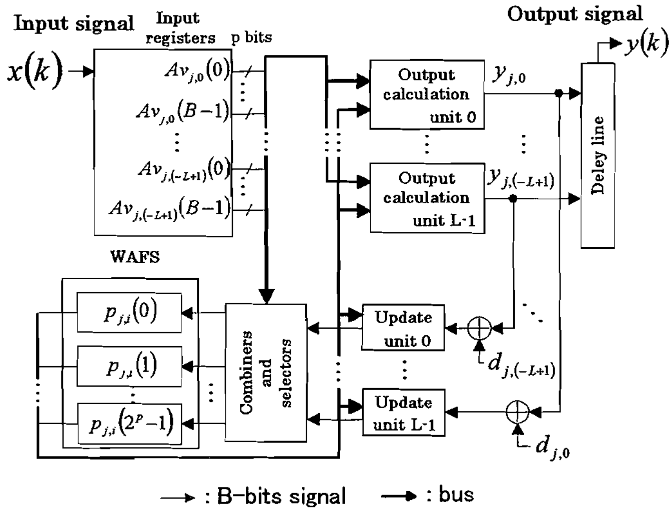



図 33: Block diagram of BDA-ADF.

と定義して，これを式 (176) に適用する．これより，出力計算式は

$$
\begin{aligned}
\boldsymbol{y}_j &= [y_{j,0}, y_{j,(-1)}, \cdots, y_{j,(-L+1)}]^T \\
&= [\boldsymbol{F}^T\boldsymbol{P}_j^0, \boldsymbol{F}^T\boldsymbol{P}_j^{(-1)}, \cdots, \boldsymbol{F}^T\boldsymbol{P}_j^{(-L+1)}]^T
\end{aligned} \tag{177}
$$

となり，各要素は同時に計算可能である．BDA-ADF の基本構成を 図 33 に示す.

### 6.3.3 プライオリティアップデート

式 (173)~式 (175) を並列に動作させる場合についてその問題点を明らかにする．BDA アルゴリズムが適応関数空間を更新する様子を 図 34 に示す．なお，これはブロック長 $L=4$, 語長 $B=3$, そしてタップ数 $p=2$ に対する例であり，更新値のボックス内に表現されている適応関数空間要素は更新対象を表している． 図 34 (a) において，4 つの更新式は並列に動作するため，更新ステップは語長方向に 3 つの Phase に分けられる．(b) は Phase1 を表し，全ての更新式が $p_j(0)$ を更新対象にする．(c) は Phase2 を表し，更新式 1 と 4 は $p_j(2)$ を，更新式 2 と 3 は $p_j(1)$ を更新対象にする．(d) は Phase3 を表し，更新式 1 は $p_j(1)$，更新式 2 は $p_j(2)$，更新式 3 は $p_j(0)$，そして更新式 4 は $p_j(3)$ を更新対象にする．複数の更新式が

同じ適応関数空間要素を同時にアクセスする場合には，あらかじめ全ての更新値の和を求めておいてから1回だけ更新する必要がある．しかし，同じ要素を更新対象にする更新式の数は$2 \sim L$と幅があるため，処理時間は不規則になりパイプライン処理には適していない．さらに，更新動作は語長に相当するPhaseをシリアルに実行するため，更新処理時間が長く必要になる．

そこで，更新動作を規則的にし，収束特性を極力劣化させない新たな更新方法を検討する．まず，更新値ベクトル$\boldsymbol{u}_{i,j}$の要素は誤差信号に対するスケーリングベクトルの重み付けで決定されるため，絶対値が大きい要素ほどアルゴリズムが収束するための効果が大きい．そこで，個々の更新式においてWAFSのある要素を更新する際に，最も絶対値の大きい更新値を用いて更新動作を実行することを考える．さらに，更新動作を規則的に実行するために式(173)から式(175)をWAFSを対象とする更新式に拡張し，要素を0番目から$2^p-1$番目まで順番に更新することを考える．拡張された更新式を以下に示す．

$$\boldsymbol{Pw}_{j,(-L+2)} = \boldsymbol{Pw}_{j,(-L+1)} + \boldsymbol{U}_{j,(-L+1)} \tag{178}$$

$$\vdots$$

$$\boldsymbol{Pw}_{j,0} = \boldsymbol{Pw}_{j,(-1)} + \boldsymbol{U}_{j,(-1)} \tag{179}$$

$$\boldsymbol{Pw}_{(j+1),(-L+1)} = \boldsymbol{Pw}_{j,0} + \boldsymbol{U}_{j,0} \tag{180}$$

ここで，$\boldsymbol{Pw}_{j,i}$と$\boldsymbol{U}_{j,i}$は，それぞれブロック$j$の時刻$i$におけるWAFSとその更新値を表し，

$$\begin{aligned}\boldsymbol{U}_{j,i} &= [u_{j,i}(0), u_{j,i}(1), \cdots, u_{j,i}(2^p-1)]^T \\ &= \boldsymbol{T}_{j,i}[0.5p\frac{\mu_B}{L}\boldsymbol{F}e_{j,i}]\end{aligned}$$

である．なお，$\boldsymbol{T}_{j,i}$は$2^p \times B$の変換行列である．ここで，個々の更新式において，$B$個のアクセスベクトルの中で，いくつかのアクセスベクトルが同じ値を有する場合には，それらのなかで最も大きな更新値を有するアクセスベクトルを選択する．これより，$\boldsymbol{T}_{j,i}$の列ベクトルの要素には一つの'1'が存在し，その他は全て'0'となる．そして，更新動作をWAFSの要素$p_{j,i}(0) \sim p_{j,i}(2^p-1)$の順に実行する．この更新方法をプライオリティ・アップデート

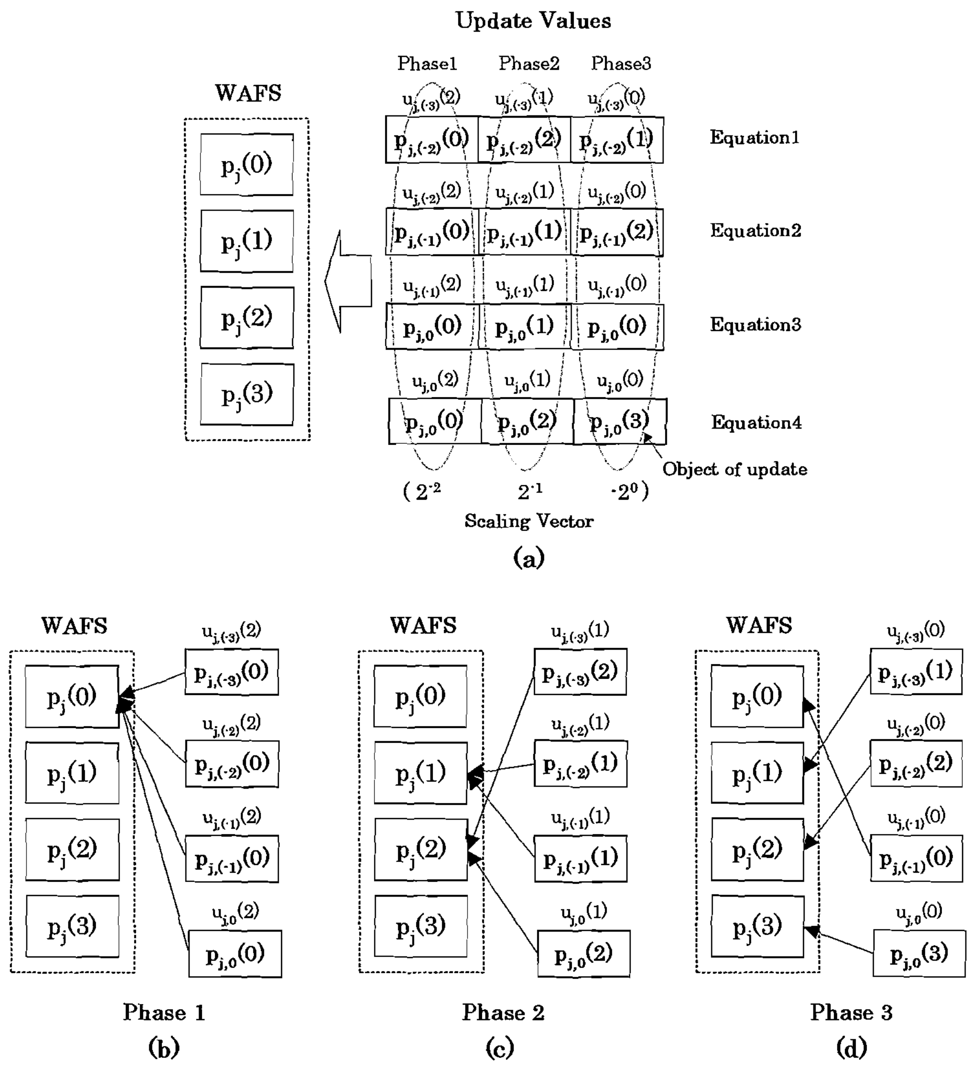



図 34: Example of update procedure of BDA algorithm for L=4, B=3, and p=2.

と呼ぶことにする．更新式は，式(178)から式(180)へ順次代入して

$$\boldsymbol{Pw}_{(j+1),(-L+1)} = \boldsymbol{Pw}_{j,(-L+1)} + \sum_{i=0}^{-L+1} \boldsymbol{U}_{j,i}$$

となり，さらに

$$\begin{aligned}\boldsymbol{Pw}_{(j+1)} &\triangleq \boldsymbol{Pw}_{(j+1),(-L+1)}\\ &= [p_{(j+1)}(0), \cdots, p_{(j+1)}(2^p-1)]^T\\ \boldsymbol{Pw}_j &\triangleq \boldsymbol{Pw}_{j,(-L+1)}\\ &= [p_j(0), \cdots, p_j(2^p-1)]^T\end{aligned}$$

と置くことにより，次式のように表される．

$$\boldsymbol{Pw}_{(j+1)} = \boldsymbol{Pw}_j + \sum_{i=0}^{-L+1} \boldsymbol{U}_{j,i} \tag{181}$$

プライオリティアップデートを用いた更新例を 図 34 に示す．なお，ブロック長4，語長6，タップ数2である．適応関数空間の要素数は$4(=2^p)$個，更新値は$4(=L)$つの更新式においてそれぞれ$6(=B)$個存在する．なお，実線で囲まれた要素は更新に用いられる更新値，点線で囲まれた要素は更新に使用しない更新値を表す．更新に使用されない更新値が存在する理由は，同じ要素を更新対象にする重みのより大きい更新値が存在するためである．これらの更新値は，対応するスケーリングベクトルの要素によって重み付けられる．さて，適応関数空間の更新は，$p_j(0)$から$p_j(3)$の順に実行される．各適応関数空間要素を更新するために用いられる更新値を以下に示す．なお，更新値は左から更新式1～4の順に示している．

1. $p_j(0)$の更新：$u_{j,(-3)}(2)$,$u_{j,(-2)}(2)$,$u_{j,(-1)}(2)$,$u_{j,0}(2)$

2. $p_j(1)$の更新：$u_{j,(-3)}(0)$,$u_{j,(-2)}(4)$,$u_{j,(-1)}(1)$,$u_{j,0}(1)$

3. $p_j(2)$の更新：$u_{j,(-3)}(4)$,$u_{j,(-2)}(0)$, なし ,$u_{j,0}(0)$

4. $p_j(3)$の更新：$u_{j,(-3)}(1)$,$u_{j,(-2)}(1)$,$u_{j,(-1)}(0)$,$u_{j,0}(3)$

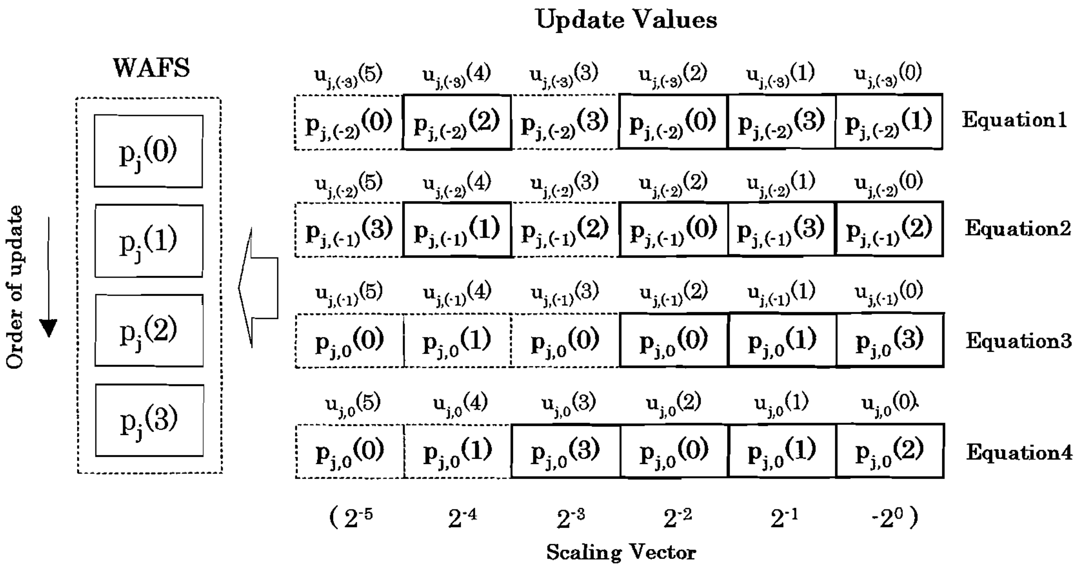



図 35: Example of priority update method for L=4, B=6, and p=2.The boxes of bold line indicate the update values used in the priority update method.

### 6.3.4 マルチメモリブロック構造

BDA アルゴリズムの WAFS の容量は $2^p$words であるため，タップ数 $p$ が増加するにしたがい容量が急激に増加する．これにより，ハードウエア規模と消費電力は増加し，収束速度が大幅に劣化する．この問題を解決するために，マルチメモリブロック構造 (Multi-memory block structure) が提案されている [25]．マルチメモリブロック構造では，$p$ 次のタップ係数を $M$ 個に分割した $(p/M)$ 次のベクトルに対してそれぞれ WAFS を定義する．これにより，個々の容量は $2^{(p/M)}$words，総容量は $M \cdot 2^{(p/M)}$words と小容量になるため，小規模なハードウエアと低消費電力を実現でき，収束速度を大幅に改善することが可能である．

マルチメモリブロック構造を適用した BDA アルゴリズム (MBDA アルゴリズム) は以下のように表される．分割されたタップ係数と WAFS を次のように定義する．

$$\boldsymbol{w}_j^m \triangleq [w_j^m(0), w_j^m(1), \cdots, w_j^m(R-1)]^T$$
$$\boldsymbol{Pw}_{j,i}^m \triangleq [p_{j,i}^m(0), p_{j,i}^m(1), \cdots, p_{j,i}^m(2^R-1)]^T$$

$$
\begin{aligned}
m &= 0, 1, \cdots, M-1 \\
i &= 0, -1, \cdots, -L+1
\end{aligned}
$$

ここで,

$$R = p/M$$

である．ブロック番号$j$の時刻$i$における適応関数空間と出力信号はそれぞれ次の様に表される．

$$
\begin{aligned}
\boldsymbol{P}_j^{i,m} &= \boldsymbol{A}_{j,i}^{m\,T}\,\boldsymbol{w}_j^m \\
y_{j,i} &= \sum_{m=0}^{M-1} \boldsymbol{F}^T \boldsymbol{P}_j^{i,m} \\
&\qquad m = 0, 1, \cdots, M-1
\end{aligned}
$$

ここで,

$$
\boldsymbol{A}_{j,i}^m = \begin{bmatrix} b_{j,i}^m(0) & \cdots & b_{j,(i-R+1)}^m(0) \\ b_{j,i}^m(1) & \cdots & b_{j,(i-R+1)}^m(1) \\ \vdots & \ddots & \vdots \\ b_{j,i}^m(B-1) & \cdots & b_{j,(i-R+1)}^m(B-1) \end{bmatrix}^T
$$

である．MBDAアルゴリズムの更新式は

$$\boldsymbol{P}_{j,(-L+2)}^{(-L+1),m} = \boldsymbol{P}_{j,(-L+1)}^{(-L+1),m} + 0.5R\frac{\mu_B}{L}\,\boldsymbol{F}e_{j,(-L+1)} \tag{182}$$

$$\vdots$$

$$\boldsymbol{P}_{j,0}^{(-1),m} = \boldsymbol{P}_{j,(-1)}^{(-1),m} + 0.5R\frac{\mu_B}{L}\,\boldsymbol{F}e_{j,(-1)} \tag{183}$$

$$\boldsymbol{P}_{(j+1),(-L+1)}^{0,m} = \boldsymbol{P}_{j,0}^{0,m} + 0.5R\frac{\mu_B}{L}\,\boldsymbol{F}e_{j,0} \tag{184}$$

である．これらの更新式は，次式で表される$m$番目のWAFSの中から$B$個の要素を選択して更新することを表している．

$$\boldsymbol{Pw}_{j,i}^m = [p_{j,i}^m(0), p_{j,i}^m(1), \cdots, p_{j,i}^m(2^R-1)]^T$$

式(182)から式(184)をWAFSに拡張してプライオリティ・アップデートを適用する．

$$\boldsymbol{Pw}^m_{j,(-L+2)} = \boldsymbol{Pw}^m_{j,(-L+1)} + \boldsymbol{U}^m_{j,(-L+1)} \tag{185}$$

$$\vdots$$

$$\boldsymbol{Pw}^m_{j,0} = \boldsymbol{Pw}^m_{j,(-1)} + \boldsymbol{U}^m_{j,(-1)} \tag{186}$$

$$\boldsymbol{Pw}^m_{(j+1),(-L+1)} = \boldsymbol{Pw}^m_{j,0} + \boldsymbol{U}^m_{j,0} \tag{187}$$

ここで，$\boldsymbol{U}^m_{j,i}$は$m$番目のWAFSに対する更新値を表しており，

$$\begin{aligned}\boldsymbol{U}^m_{j,i} &= [u^m_{j,i}(0), u^m_{j,i}(1), \cdots, u^m_{j,i}(2^R-1)]^T \\ &= \boldsymbol{T}^m_{j,i}[0.5R\frac{\mu_B}{L}\boldsymbol{F}e_{j,i}]\end{aligned}$$

である．なお，$\boldsymbol{T}^m_{j,i}$は$2^R \times B$の変換行列である．更新式は，式(185)から式(187)へ順次代入して

$$\boldsymbol{Pw}^m_{j,1} = \boldsymbol{Pw}^m_{j,(-L+1)} + \sum_{i=0}^{-L+1} \boldsymbol{U}^m_{j,i}$$

となり，さらに

$$\begin{aligned}\boldsymbol{Pw}^m_{(j+1)} &\triangleq \boldsymbol{Pw}^m_{(j+1),(-L+1)} \\ &= [p^m_{(j+1)}(0), p^m_{(j+1)}(1), \cdots, p^m_{(j+1)}(2^R-1)]^T \\ \boldsymbol{Pw}^m_j &\triangleq \boldsymbol{Pw}^m_{j,(-L+1)} \\ &= [p^m_j(0), p^m_j(1), \cdots, p^m_j(2^p-1)]^T\end{aligned}$$

と定義することにより，更新式は次式のように表される．

$$\boldsymbol{Pw}^m_{(j+1)} = \boldsymbol{Pw}^m_j + \sum_{i=0}^{-L+1} \boldsymbol{U}^m_{j,i} \tag{188}$$

となる．MBDA-ADFの基本構成を 図 36 に示す．なお，ここではBDA-ADFの基本構成からの変更箇所のみを示している．

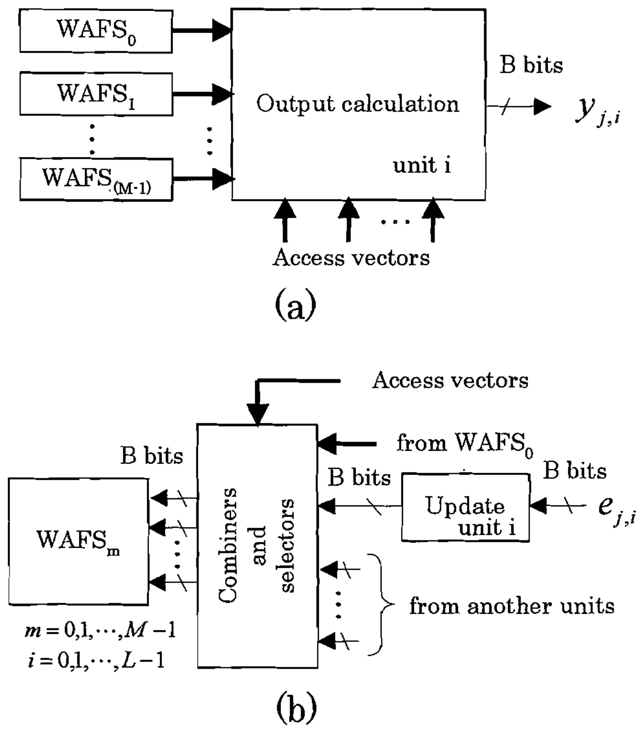



図 36: Block diagram of MBDA-ADF.(a) modification of output calculation. (b) modification of update procedure.

## 6.4 シミュレーションによる収束特性の比較

計算機シミュレーションにより，提案するMBDAアルゴリズムの収束特性を検証する．シミュレーションモデルは 図 37 に示されるシステム同定問題である．未知システムはタップ数8の低域通過FIRフィルタ，入力信号は平均0，分散0.05の白色ガウス信号，そして観測雑音として平均0，分散$1.50998 \times 10^{-6}$の入力信号とは無相関の白色ガウス信号を加えた． 図 38 は分割数$M = 1, 2, 4, 8$のMDAアルゴリズムの収束特性である．なお，分割数$M = 8$のMDAアルゴリズムの収束特性はBLMSアルゴリズムの収束特性と同等である．図 39 は，ブロック長$L = 4$，分割数$M = 1, 2, 4, 8$に対するプライオリティ・アップデートを用いたMBDAアルゴリズムの収束特性である． 表 15 にシミュレーションに用いたステップサイズを示す．なお，ステップサイズの値は同じMSEを達成し，かつ最も高速な収

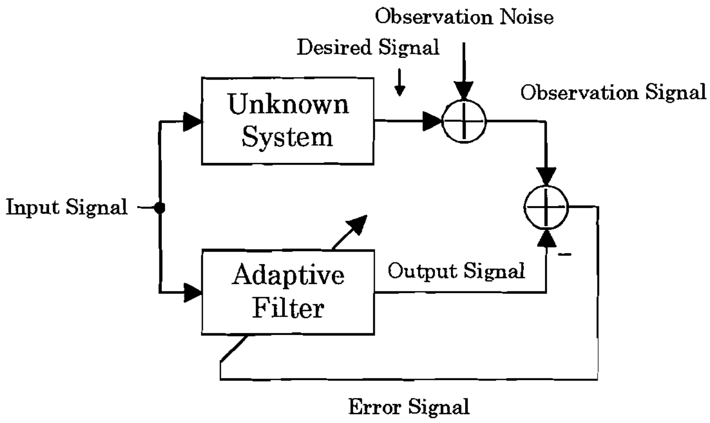



図 37: Simulation model.

表 15: Step size parameters. M indicates the division number.

| M | MDA algorithm | MBDA algorithm |
|---|---|---|
| 1 | $2^4$ | $2^2$ |
| 2 | $2^3$ | $2^2$ |
| 4 | $2^3$ | $2^1$ |
| 8 | $2^3$ | $2^0$ |

束速度を示す値を選択した．これらより，提案するMBDAアルゴリズムはMDAアルゴリズムと比較してやや収束速度は劣化しているものの，良好な収束特性を有している．次に，図40は分割数$M = 4$，ブロック数$L = 1, 2, 3, 4$に対するMBDA-ADFの収束特性である．これより，MBDAアルゴリズムは異なるブロック数に対しても良好な収束特性を有することがわかる．なお，多くのシミュレーションにおいても同様の結果を得ている．

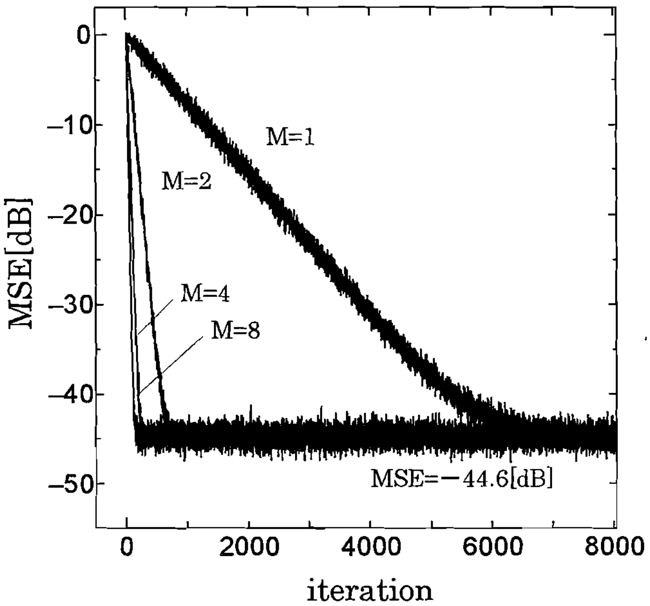



図 38: Convergence characteristics of MDA.

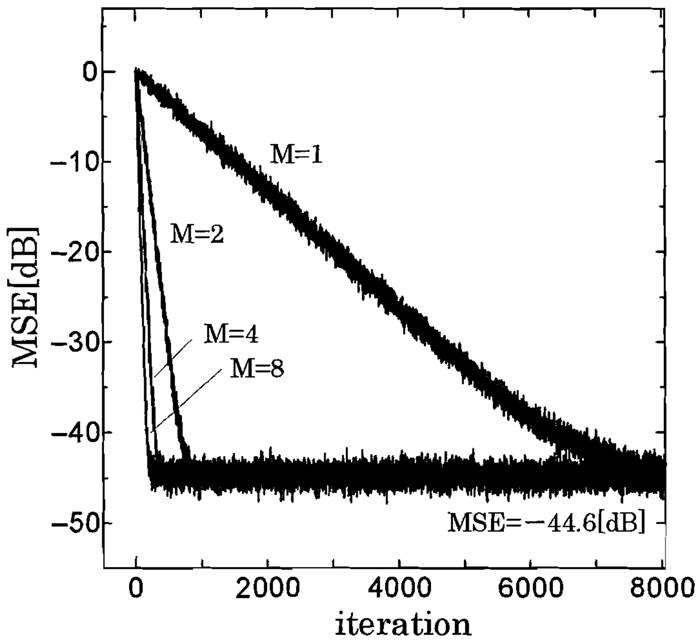



図 39: Convergence characteristics of MBDA using priority update method.

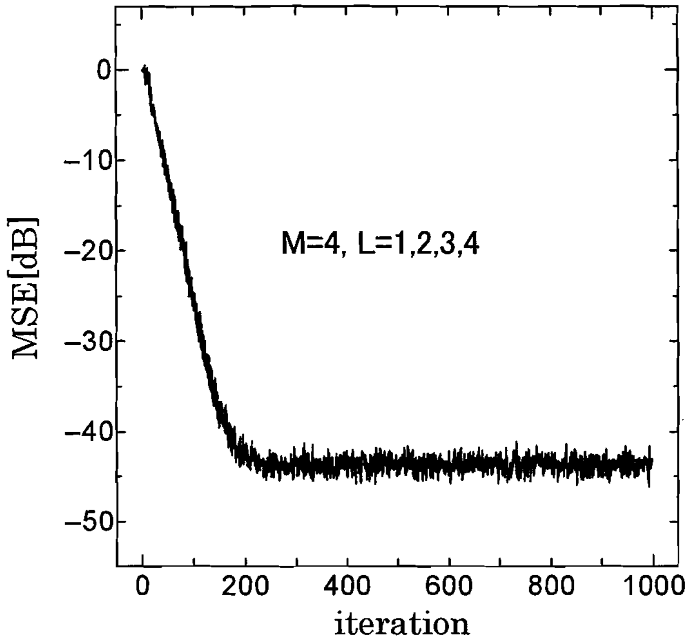



図 40: Convergence characteristics of MBDA using priority update method for various L.

## 6.5 VLSI アーキテクチャ

新たな更新方法であるプライオリティ・アップデートを用いた MBDA-ADF の高性能アーキテクチャを 図 41 に示す．なお，ブロック長と分割数はどちらも 2 とした．プライオリティ・アップデートを用いた更新式と出力計算式を以下に示し，動作を説明する．

$$\boldsymbol{Pw}^{m}_{(j+1)} = \boldsymbol{Pw}^{m}_{j} + \sum_{i=0}^{1} \boldsymbol{U}^{m}_{j,i} \tag{189}$$

$$i = 0,1 \quad m = 0,1$$

この更新式はブロック長と等しい 2 つの更新式から構成される．

$$\boldsymbol{Pw}^{m}_{j,(0)} = \boldsymbol{Pw}^{m}_{j,(-1)} + \boldsymbol{U}^{m}_{j,(-1)} \tag{190}$$

$$\boldsymbol{Pw}^{m}_{(j+1),(-1)} = \boldsymbol{Pw}^{m}_{j,(0)} + \boldsymbol{U}^{m}_{j,0} \tag{191}$$

$$m = 0,1$$

また，出力計算式は次式で表される．

$$\begin{aligned}\boldsymbol{y}_j &= [y_{j,0}, y_{j,(-1)}]^T \\ &= [\sum_{m=0}^{1} \boldsymbol{F}^T \boldsymbol{P}_j^{0,m}, \sum_{m=0}^{1} \boldsymbol{F}^T \boldsymbol{P}_j^{(-1),m}]^T \end{aligned} \quad (192)$$

入力信号レジスタは$(p+L-1)\times B$個の1ビットシフトレジスタ(SR)で構成され，アドレスベクトルを生成する．Controllerは，全てのアドレスベクトルを入力信号とし，selector-0, selector-1，selector-2の制御信号を生成する．本構成は，ブロック長が2であるため中央を境として右側と左側に同じ構成を2つ有している．

MBDAアルゴリズムは，フィルタ出力の計算とWAFSの更新の二つのステージに分けられる．

**[出力計算]**

出力信号は，式(192)を用いて求められるが，2つの出力はそれぞれアドレスベクトルが指定する2つの適応関数空間$\mathrm{WAFS}_0$と$\mathrm{WAFS}_1$の要素（部分積）の和を$B$回シフト加算することによって求められる．本構成では，適応関数空間は$2^R$個のregisterから構成されるため，その出力に配置された"Selector-0"によって1つの要素を指定する．そして，$\mathrm{WAFS}_0$と$\mathrm{WAFS}_1$の要素をそれぞれ1つづつ選択してそれらの和を求め，シフト加算を行う．この動作を$B$回実行することによって出力$y_{j,0}$と$y_{j,(-1)}$が同時に求められる．これまで，WAFSはRAM(Random Access Memory)を用いて実現されてきた．しかし，RAMでは複数要素を同時に読み出すことができないため，出力計算における並列処理には不適当である．そこで，本構成ではregisterを用いてWAFSを実現することにより並列処理を可能にしている．

**[WAFSの更新]**

WAFSの更新動作は，式(189)を実行する．まず，所望信号から出力信号を減算して誤差信号が得られ，次いで更新値を生成するために，Scalerは誤差信号を$0\sim B-1$ビットの右シフトした信号を出力する．WAFSの更新は，$p_j^m(0)\sim p_j^m(2^R-1)$の順に実行されるが，この際，更新の対象になるWAFSの要素はSelector-0，そして更新値（シフトされた誤差信号）はSelector-1によって選択される．まず，$p_j^0(0)$の更新は，各更新式においてこの要

素を更新対象にする最も重みの大きい更新値（誤差信号）をSelector-1は選択する（プライオリティ・アップデート）．更新値は各更新式において最大1個，そしてこの例ではブロック長が2であるため総数は最大2である．もし，この要素が更新対象にならなければ更新値として0を用いる．そして，これらを加算した後，Selector-2によって対象となる要素を選択して更新する．

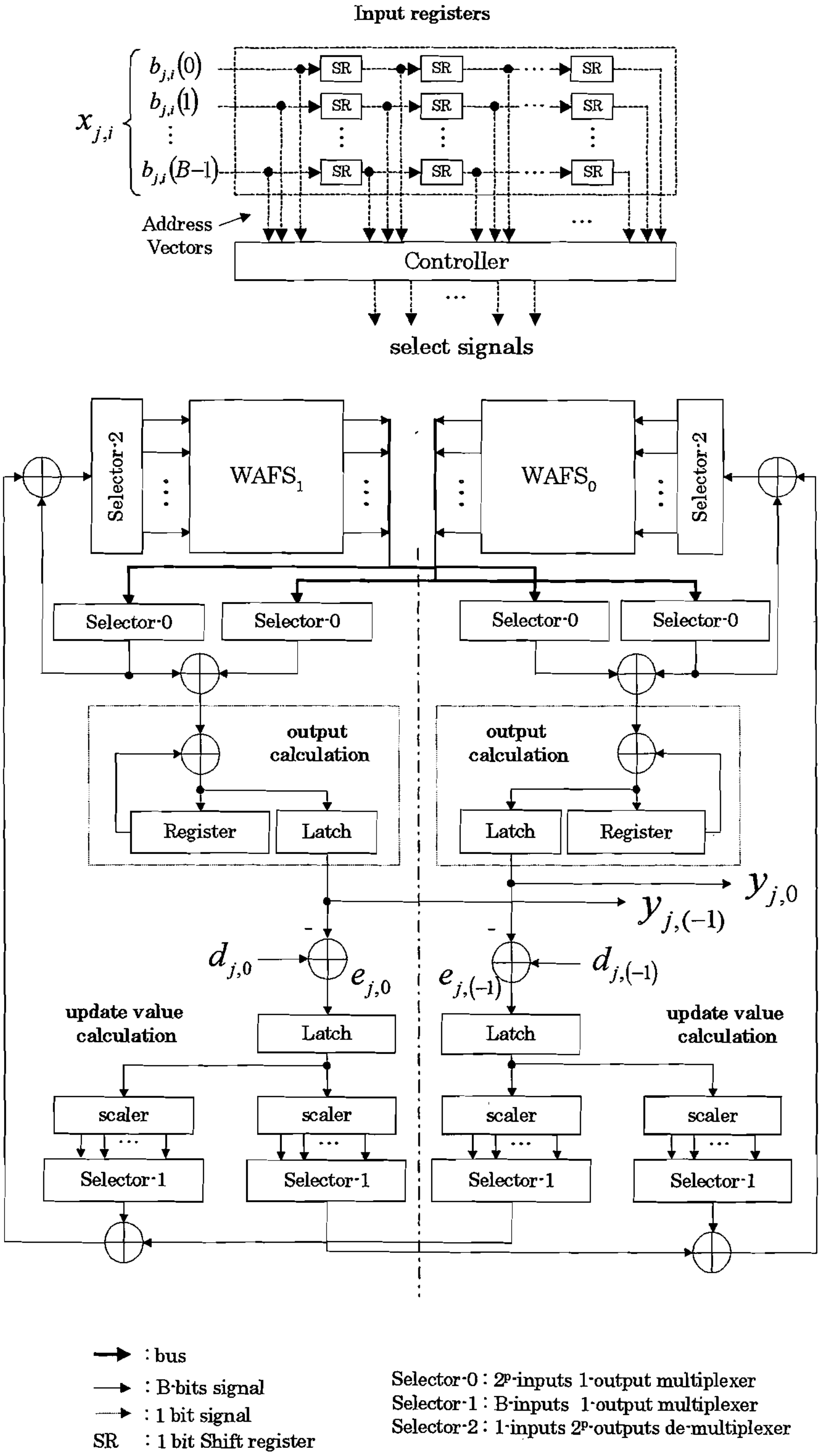



図 41: Block diagram of proposed MBDA-ADF with L=M=2.

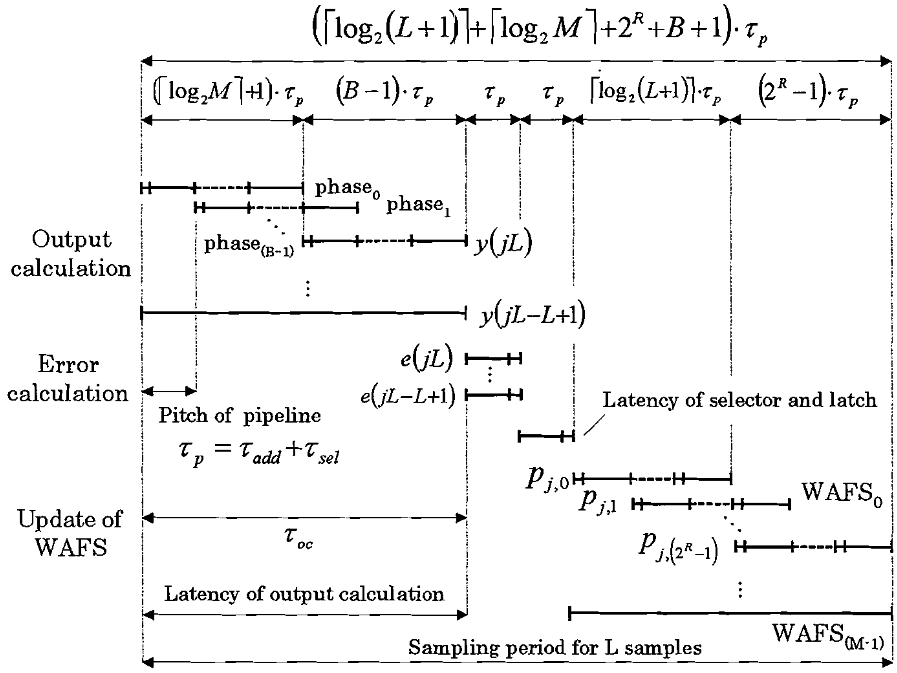



図 42: Timing chart of proposed MBDA-ADF.

タイミングチャートを 図 42 に示す．$L$個の入力信号サンプルに対するサンプリング周期$Ts^L$とサンプリングレート$Fs^L$は

$$Ts^L = (\lceil log_2(L+1)\rceil + \lceil log_2 M\rceil + 2^R + B + 1)\cdot\tau_p,$$

$$Fs^L = 1/Ts^L,$$

である．1サンプルあたりのサンプリング周期$Ts$は，$L$で割り

$$Ts = Ts^L/L$$

となるので，サンプリングレート$Fs$は

$$Fs = 1/Ts$$

である．出力滞在時間$\tau_o$は，$Ts^L$と出力計算時間$\tau_{oc}$の和であるので，

$$\tau_{oc} = (\lceil log_2 M\rceil + B)\cdot\tau_p$$

$$
\begin{aligned}
\tau_o &= Ts + \tau_{oc} \\
&= (\lceil log_2(L+1)\rceil + 2\lceil log_2 M\rceil + 2^R + 2B + 1)\cdot\tau_p
\end{aligned}
$$

となる．ここで，$\lceil x \rceil$ は $x$ 以上の最小の整数であり，$\tau_{add}$ と $\tau_{sel}$ はそれぞれ加算器とセレクタの出力滞在時間を表す．ブロック長に対するサンプリングレートと出力滞在時間を 表 16 に示す．なお，タップ数は $p = 128$，$\tau_{add} = 15ns$，$\tau_{sel} = 7ns$ とした．ブロック長 $L$ が増加するにしたがって，サンプリングレートと出力滞在時間もともに増加するが，サンプリングレートの増加に対して出力滞在時間の増加は非常に小さいことがわかる．これは，並列処理のため出力計算時間はブロック長に依存しないこと，そしてWAFSの更新動作においては，$L$ 個の更新値を加算するために"Tree-adder"（加算器をツリー状に配置した構造を有する）を使用して更新時間の増加を極力抑制しているためである．分割数が32から64に増加すると，サンプリングレートは増加し，出力滞在時間は減少している．マルチメモリブロック構造の出力計算においては，$M$ 個のWAFSから選択される要素を加算する処理時間が新たに追加される．これは，サンプリングレートの減少と出力滞在時間の増加をもたらす．しかし，本構成では"Tree-adder"を用いるため，分割による出力計算時間 $\tau_{oc}$ の増加は極めて少ない．一方，更新時間はWAFSの容量に依存しており

$$(\lceil log_2(L+1)\rceil + 2^R - 1)\cdot\tau_p$$

であるが，分割数の増加により各適応関数空間の容量 $(2^R)$ が減少するために更新時間も減少する．これらより，分割数 $M = 64$ においてサンプリングレートが増加し，出力滞在時間は減少するのである．さらに大きなブロック長を選択すると，MBDA-ADFはより高速なサンプリングレートを実現可能である．

表 17 にClarkらの周波数領域アルゴリズムによるBLMS-ADF[30]との比較を示す．Clarkらの周波数領域におけるBLMS-ADFは，高速フーリエ変換を用いるためにタップ数とブロック長を同じ2のべき乗に選択する必要がある．そこで，比較においてはブロック長 $L$ とタップ数 $p$ を2から128までの2のべき乗を選択した．我々の提案するMBDA-ADFは，$L(=p) = 128$ において165.5MHzのサンプリングレートを達成可能である．これは，BLMS-

表 16: Sampling rate $Fs$ and output latency $\tau_o$ of MBDA-ADF for M=32,64, p=128 and B=16.

| | M=32(R=4) | | M=64(R=2) | |
|---|---|---|---|---|
| L | Fs [MHz] | $\tau_o$ [ns] | Fs [MHz] | $\tau_o$ [ns] |
| 1 | 1.16 | 1326.0 | 1.61 | 1105.0 |
| 2 | 2.26 | 1348.1 | 3.12 | 1127.1 |
| 4 | 4.41 | 1370.2 | 6.03 | 1149.2 |
| 8 | 8.62 | 1392.3 | 11.68 | 1171.3 |
| 16 | 16.84 | 1414.4 | 22.62 | 1193.4 |
| 32 | 32.90 | 1436.5 | 43.88 | 1215.5 |

表 17: Comparison of sampling rate $Fs$ and output latency $\tau_o$ between MBDA-ADF with $M = 64$ and BLMS-ADF. The word length B=16.

| | MBDA | | BLMS | |
|---|---|---|---|---|
| L(=p) | Fs [MHz] | $\tau_o$ [ns] | Fs [MHz] | $\tau_o$ [ns] |
| 2 | 3.93 | 861.9 | 3.16 | 918.0 |
| 4 | 7.24 | 928.2 | 4.52 | 1296.0 |
| 8 | 13.4 | 994.5 | 7.04 | 1674.0 |
| 16 | 25.0 | 1060.8 | 11.5 | 2052.0 |
| 32 | 46.7 | 1127.1 | 19.5 | 2430.0 |
| 64 | 87.8 | 1193.4 | 33.8 | 2808.0 |
| 128 | 165.5 | 1259.7 | 59.6 | 3186.0 |

ADFの約277.7%である．出力滞在時間もBLMS-ADFより短く，約39.5%の1259.7nsである．また，$L(=p)$に対する増加も極めて少ない．より大きな$L(=p)$に対して，さらに高速なサンプリング・レートを達成可能である．

## 6.6 まとめ

本章では，分散演算形ブロックLMSアルゴリズム(BDAアルゴリズム)と，マルチメモリブロック構造を適用したアルゴリズム(MBDAアルゴリズム)を初めて提案した．さらに，パイプライン処理向きの更新方法であるプライオリティ・アップデートを提案した．提案したアルゴリズムは良好な収束速度と推定精度を有することを計算機シミュレーションにより確認した．さらに，効果的なVLSIアーキテクチャを検討し，サンプリングレートと出力滞在時間を評価した．提案するMBDA-ADFは，タップ数128，分割数64，ブロック長128において165.5MHzのサンプリングレートを達成可能である．これは，従来法の約277.7%である．また，出力滞在時間も1259.7nsと非常に短く，従来法の39.5%である．このように，提案したMBDA-ADFはブロックLMSアルゴリズムの本来有する並列性を有効に利用した高速性と短い出力滞在時間を有し，さらに分散演算の適用によりタップ数に対する出力滞在時間の増加を最小限に抑えることが可能な高性能適応フィルタである．

# 第7章　結言

以上，第3章から第6章にわたり2の補数形式に基づく分散演算形LMSアルゴリズムと適応フィルタの効果的な構成法を提案し，収束条件の解析を行った．さらに，高速化を目的として分散演算形ブロックLMSアルゴリズムと適応フィルタの効果的アーキテクチャを提案した．

第3章では，Cowanらの分散演算形LMSアルゴリズムの問題点を明らかにし，これを解決する方法として，入力信号の符号化に2の補数形式を用いた分散演算形LMSアルゴリズムを導出した．次いで，高次における問題点を解決するためにマルチメモリブロック構成を適用した分散演算形LMSアルゴリズムを導出した．これは，高次においても良好な収束速度と推定精度を有する高性能な適応アルゴリズムである．さらに，このアルゴリズムを適用したマルチメモリブロック構造を有する分散演算形LMS適応フィルタの効果的なアーキテクチャを提案した．提案する適応フィルタは，良好な収束速度を有し，タップ数にほとんど依存しない短い出力滞在時間，高速なサンプリングレート，小規模なハードウエア，そして低消費電力を実現することが可能である．また，ステップサイズを適切に設定することにより，有色性を有する入力信号に対してもLMSアルゴリズムと同様に収束することを確認した．

第4章では，さらに高速な収束速度，小規模なハードウエア，低消費電力を達成可能なハーフメモリアルゴリズムと，これを実現する高性能アーキテクチャを提案した．ハーフメモリアルゴリズムは，適応関数空間の奇対称性を利用することにより可能となるが，2の補数形式を用いた場合にも適応関数空間には準奇対称性が現われることを解析的に初めて示した．これにより，適応関数空間は1/2の容量で実現可能になるためハードウエア規模と消費電力を削減することが可能になり，そして適応関数空間のアクセス確率が2倍に増加するため収束速度を約2倍に向上させることが可能になる．

第5章では，分散演算形LMSアルゴリズムの収束条件を理論的に解析した．収束条件式の導出は更新式を全適応関数空間に拡張することにより初めて可能になり，その結果，収束

条件は拡張入力信号ベクトルの自己相関行列の最大固有値に依存することが明らかになった．また，Cowanらが提案したオフセットバイナリ形式に基づくアルゴリズムの問題点について理論的に示すとともに提案法の有効性を示した．

第6章では，飛躍的に高速化が可能な分散演算形ブロックLMSアルゴリズムとパイプライン処理を用いた高性能な並列VLSIアーキテクチャを提案した．このアルゴリズムにおいては，パイプライン処理を可能にする新たな更新方法であるプライオリティアップデートを用いている．この更新方法では，更新値の数に制限を加えること，適応関数空間の全要素を順に更新することにより，パイプライン処理を用いた規則的な更新動作が可能になる．提案した構成法では，タップ数128，分割数64，ブロック長128において165.5MHzのサンプリングレートを達成可能である．また，出力滞在時間も1259.7nsと非常に短い．

本論文は，今後とも広範囲な応用が期待される適応フィルタに対して検討を行ったものであるが，前提として効率的なハードウエア実現に目標を定めて議論を進めてきた．そのために，アルゴリズムと効果的なアーキテクチャという2つの視点よりアプローチしたが，それらは個別の問題として存在するのではなく，相互に密接な関連を有している．これまで数多く研究されてきた乗算器を用いた構成とは異なり，分散演算の手法に基づく適応アルゴリズムは加算器とレジスタを用いたシンプルな構成により実現可能であるため多くの自由度を有する．このことは，サンプリングレートや出力滞在時間などの様々な要求が課せられる適応フィルタの実現においては，それぞれの要求に適切に対応するためにも極めて重要な特徴である．例えば，出力計算では部分積のシフト加算を語長回数だけ実行するが，シフト加算モジュールを複数個用いた並列演算により，出力滞在時間をさらに減少させることも可能である．これは，語長が大きい場合において有効であり，出力滞在時間の短縮とサンプリングレートの向上につながる．

本論文は，これ自体で完成したものではなく，今後の研究に待つべき問題も多い．

まず第1に，本論文では，分散演算形LMS適応フィルタとそのブロック実現についてVLSIアーキテクチャを提案しているが，レイアウト設計などの実際にVLSI化を行った場合の詳細な評価を行う必要がある．

第2に，分散演算形LMSアルゴリズムについて収束条件を明らかにしたが，マルチメモリブロック構成やハーフメモリアルゴリズムの収束条件についても明らかにする必要がある．また，推定精度や有色信号に対する検討も必要である．

第3に，本論文では固定小数点演算形式に対する検討を行ってきたが，浮動小数点形式を用いた場合についても検討する必要がある．

第4に，本論文では実係数の適応フィルタに対する検討を行ってきたが，解析信号を処理するための複素係数についても検討を進める必要がある．また，他の適応アルゴリズムに対する分散演算の適用についても検討を進める必要がある．

第5に，地震波の解析，生体信号処理，画像信号処理などの多次元信号処理への応用も検討する必要がある．

しかしながら，適応ディジタルフィルタのアルゴリズムや構成に関して，以上の4章にわたる内容も参考になるところが少なくないと考え，ここに報告する次第である．

# 付録　A

2の補数形式においても式(45)の$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$を対角化できることを明らかにする．以下にその導出の過程を示す．

$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$の行列の乗算は以下のように表される．

$$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k) = \begin{bmatrix} b_0(k) & b_0(k-1) & \cdots & b_0(k-N+1) \\ b_1(k) & b_1(k-1) & \cdots & b_1(k-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ b_{B-1}(k) & b_{B-1}(k-1) & \cdots & b_{B-1}(k-N+1) \end{bmatrix} \begin{bmatrix} b_0(k) & b_0(k-1) & \cdots & b_0(k-N+1) \\ b_1(k) & b_1(k-1) & \cdots & b_1(k-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ b_{B-1}(k) & b_{B-1}(k-1) & \cdots & b_{B-1}(k-N+1) \end{bmatrix}^T \tag{A·1}$$

この行列における各要素は，それぞれ0または1の値をとる．ここで，入力信号が平均0の白色信号であり，入力信号を構成する各ビットが互いに無相関である条件のもとで期待値をとると，$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$の行列の乗算は次のように表される．

$$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k) = \begin{bmatrix} 0.5N & 0.25N & \cdots & 0.25N \\ 0.25N & 0.5N & \cdots & 0.25N \\ \vdots & \vdots & \ddots & \vdots \\ 0.25N & 0.25N & \cdots & 0.5N \end{bmatrix} \tag{A·2}$$

ここで，$\boldsymbol{A}^T(k)$が$B\times N$の行列で，$\boldsymbol{A}(k)$が$N\times B$の行列であるので，$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$は$B\times B$の行列となる．このように，2の補数形式では$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$の対角化を行うことが不可能である．

そこで，この$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$に対してスケーリングベクトル$\boldsymbol{F}$を含めて考えることによって，以下のように置き換えることが可能となる．

$$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)\,\boldsymbol{F}$$

$$
= 0.25N \begin{bmatrix} 2 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 2 \end{bmatrix} \begin{bmatrix} -2^0 \\ 2^{-1} \\ \vdots \\ 2^{-B+1} \end{bmatrix}
$$

$$
= 0.25N \begin{bmatrix} -2\times 2^0 + \sum_{i=1}^{B-1} 2^{-i} \\ -1\times 2^0 + 2^{-1} + \sum_{i=1}^{B-1} 2^{-i} \\ \vdots \\ -1\times 2^0 + 2^{-B+1} + \sum_{i=1}^{B-1} 2^{-i} \end{bmatrix} \tag{A·3}
$$

ここで，語長$B$がある程度大きい場合には$\sum_{i=1}^{B-1} 2^{-i}$を近似的に1として扱うことができるから

$$
\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)\,\boldsymbol{F}
$$

$$
\approx 0.25N \begin{bmatrix} -2\times 2^0 + 1 \\ -1\times 2^0 + 2^{-1} + 1 \\ \vdots \\ -1\times 2^0 + 2^{-B+1} + 1 \end{bmatrix}
$$

$$
= 0.25N \begin{bmatrix} -2^0 \\ 2^{-1} \\ \vdots \\ 2^{-B+1} \end{bmatrix}
$$

$$
= \begin{bmatrix} 0.25N & 0 & \cdots & 0 \\ 0 & 0.25N & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0.25N \end{bmatrix} \begin{bmatrix} -2^0 \\ 2^{-1} \\ \vdots \\ 2^{-B+1} \end{bmatrix} \tag{A·4}
$$

となる．

このように，2の補数形式においても$\boldsymbol{A}^T(k)\,\boldsymbol{A}(k)$を，対角要素のみが$0.25N$という値を持つ対角行列に置き換えて考えることが可能となる．

# 付録　B

DA 適応アルゴリズムの正規方程式を導く．

誤差の2乗平均値を最小化することを考えると評価量$J$は，

$$J = E[e^2(k)] = E[(d(k) - y(k))^2] \tag{B·1}$$

で与えられる．ここで，$d(k)$ は所望信号，$y(k)$ はフィルタ出力を表す．これに出力式

$$y(k) = \frac{1}{N}\boldsymbol{S}_{DA}^T(k)\,\boldsymbol{P}_w(k) \tag{B·2}$$

を代入すると，

$$\begin{aligned}
J = & E[\frac{1}{N^2}\boldsymbol{P}_w^T(k)\,\boldsymbol{S}_{DA}(k)\,\boldsymbol{S}_{DA}^T(k)\,\boldsymbol{P}_w(k)] \\
& - 2E[\frac{1}{N}d(k)\,\boldsymbol{S}_{DA}^T(k)\,\boldsymbol{P}_w(k)] + E[d^2(k)] && \text{(B·3)} \\
= & \frac{1}{N^2}\boldsymbol{P}_w^T(k)E[\,\boldsymbol{S}_{DA}(k)\,\boldsymbol{S}_{DA}^T(k)]\,\boldsymbol{P}_w(k) \\
& - 2\frac{1}{N}E[d(k)\,\boldsymbol{S}_{DA}^T(k)]\,\boldsymbol{P}_w(k) + E[d^2(k)] && \text{(B·4)} \\
= & \frac{1}{N^2}\boldsymbol{P}_w^T(k)\,\boldsymbol{R}\boldsymbol{P}_w(k) \\
& - 2\frac{1}{N}\boldsymbol{q}\,\boldsymbol{P}_w(k) + E[d^2(k)] && \text{(B·5)}
\end{aligned}$$

となる．ここで，

$$\boldsymbol{R} = E[\,\boldsymbol{S}_{DA}(k)\,\boldsymbol{S}_{DA}^T(k)] \tag{B·6}$$

$$\boldsymbol{q} = E[d(k)\,\boldsymbol{S}_{DA}^T(k)] \tag{B·7}$$

とおいた．(B·5) 式は適応関数空間に関する2次式になっていることがわかる．したがって，$J$は $\boldsymbol{P}_w(k)$ に関する凸関数で唯一の最小値を持つ．ここでは，時刻$k$において$J$を最小にする推定ベクトルを$\boldsymbol{P}_w^*$と表記する．$\boldsymbol{P}_w^*$は，(B·5) 式の両辺を $\boldsymbol{P}_w(k)$ で偏微分して，

$$\frac{\partial J}{\partial \boldsymbol{P}_w(k)} = 2\frac{1}{N^2}\boldsymbol{R}\boldsymbol{P}_w(k) - 2\frac{1}{N}\boldsymbol{q} \tag{B·8}$$

となり，これを零とおいてDA適応アルゴリズムの正規方程式は

$$\boldsymbol{q} = \frac{1}{N}\boldsymbol{R}\boldsymbol{P}_w^* \tag{B$\cdot$9}$$

となる.

# 参考文献

[1] A. V. Oppenheim, R. W. Schafer,"DIGITAL SIGNAL PROCESSING,"Prentice-Hall, Engewood Cliffs, New Jersey, 1975.

[2] B. Widrow and M. E. Hoff,'Adaptive Switching Circuit," IRE EWSCON Conv. Rec.,pp 96-104, 1960.

[3] 片山 徹, "応用カルマンフィルタ," 朝倉書店, 1983.

[4] 有本 卓, "カルマンフィルター," 産業図書, 1977.

[5] 電子情報通信学会編, "ディジタル信号処理ハンドブック," オーム社, 1993.

[6] K.J. Raghunath and K.K. Parhi, "High-speed RLS using scaled tangent rotation (star), " Proc. IEEE ISCAS'93, Chicago, USA, pp.1959–1962, May 1993.

[7] K.J. Raghunath and K.K. Parhi, "A 100 MHz pipe-lined RLS adaptive filter," Proc. IEEE ICASSP'95, Detroit, Michigan, pp.3187–3190, May 1995.

[8] M.D. Meyer and D.P. Agrawal, "A high sampling rate delayed LMS filter architecture," IEEE Trans. Circuits & Syst. II, vol.40, no.11, pp.727-729, Nov. 1993.

[9] C.L. Wang, "Bit-serial VLSI implementation of delayed LMS transversal adaptive filters," IEEE Trans. Signal Processing, vol.42, no.8, pp.2169–2175, Aug. 1994.

[10] 松原勝重, 西川清史, 貴家仁志, "Delayed LMS アルゴリズムに基づくパイプライン適応フィルタ," 信学論 (A), vol.J79-A, no.5, pp.1050–1057, May 1996.

[11] Flavio Lorenzelli, Kung Yao"A Linear Systolic Array for Recursive Least Squares," IEEE Trans. Signal Processing, vol.43, no.12, pp.3014–3021, Dec. 1995.

[12] A. Harada, K. Nishikawa and H. Kiya, "Pipelined architecture of the LMS adaptive digital filter with the minimum output latency," IEICE Trans. Fundamentals, vol.E81-A, no.8, pp.1578-1585, Aug. 1998.

[13] T. Kimijima, K. Nishikawa and H. Kiya, "An Effective Architecture of the Pipelined LMS adaptive filters," IEICE Trans. Fundamentals, vol.E82-A, no.8, pp.1428-1434, Aug. 1999.

[14] 恒川佳隆，高橋　強，三浦　守，"FADDEEVAアルゴリズムに基づく滞在時間最小型カルマンフィルタのVLSIアーキテクチャ，" 計測自動制御学会論文集，Vol.34, N0.12, pp.1913-1921, Dec. 1998.

[15] C.F.N. Cowan and J. Mavor, "New digital adaptive-filter implementation using distributed-arithmetic techniques," IEE Proc., vol.128, Pt.F, no.4, pp.225–230, Aug. 1981.

[16] C.F.N. Cowan, S.G. Smith and J.H. Elliott, "A digital adaptive filter using a memory-accumulator architecture:theory and realization," IEEE Trans. Acoust., Speech & Signal Process., vol.31, no.3, pp.541–549, Jun. 1983.

[17] C. F. N. Cowan and P. M. Grant, "Adaptive Filters," Prentice-Hall, Inc.,New Jersey, 1985.

[18] A. Peled and B. Liu, "A new hardware realization of digital filters," IEEE Trans. Acoust., Speech & Signal Process., vol.22, no.12, pp.456–462, Dec. 1974.

[19] 高橋　強，豊田真嗣，恒川佳隆，三浦　守，"分散演算型LMS適応フィルタの収束特性解析，" 計測自動制御学会東北支部第178回研究集会，178-3, Nov. 1998.

[20] 高橋　強，恒川佳隆，三浦　守，関 享士郎，"分散演算を用いたLMS適応フィルタの収束条件，" 計測自動制御学会東北支部第187回研究集会，187-3, June 2000.

[21] 豊田真嗣，高橋　強，恒川佳隆，三浦　守，“分散演算型LMS適応フィルタのVLSI実現,” 第12回ディジタル信号処理シンポジウム講演論文集，B8-3，pp.645–650，Nov. 1997.

[22] Keshab K.,Parhi,“VLSI digital signal processing systems:design and implementation, ”John Wiley & Sons,Inc., New York, 1999.

[23] 恒川佳隆，高橋 強，三浦 守，“Multiplierless LMS適応フィルタの高性能VLSIアーキテクチャ,”,SICE'99予稿集，204A-2, June 1999.

[24] 恒川佳隆，高橋　強，豊田真嗣，三浦　守，“分散演算によるマルチプライヤレスLMS適応フィルタの高性能アーキテクチャ，” 信学論(A)，vol.J-82-A, no.10, pp.1518-1528, Oct. 1999.

[25] C.H. Wei, J.J. Lou, “Multimemory block structure for implementing a digital adaptive filter using distributed arithmetic,” IEE Proc., vol.133, Pt.G, no.1, pp.19–26, Feb. 1986.

[26] 豊田真嗣，高橋　強，恒川佳隆，三浦　守，“ハーフメモリアルゴリズムを用いた分散演算型LMS適応フィルタのVLSI実現,” 信学技報，DSP98-23，May 1998.

[27] 高橋強，恒川佳隆，豊田真嗣，三浦守,“ハーフメモリアルゴリズムに基づく分散演算形LMS適応フィルタの高性能アーキテクチャ,” 信学論(A)，vol.J84-A, no.6, pp.777-787, June 2001.

[28] K. Takahashi, Y. Tsunekawa, N. Tayama, K. Seki,“Analysis of the Convergence Condition of LMS Adaptive Filter Using Distributed Arithmetic,” Proceedings of ITC-CSCC'01, vol.1, pp.576-579, July 2001.

[29] K. Takahashi, Y. Tsunekawa, N. Tayama, K. Seki,“Analysis of the Convergence Condition of LMS Adaptive Filter Using Distributed Arithmetic,” IEICE TRANS. FUN-

DAMENTALS, vol.E85-A, NO.6, pp.151-158, JUNE 2002.

[30] Gregory A. Clark, Sanjit K. Mitra, Sydney R. Parker,"Block Implementation of Adaptive Digital Filters," IEEE Trans. Circuits and Syst., vol. CAS-28, pp.584–592, June. 1981.

[31] A. Feuer, "Performance Analysis of the Block Least Mean Square Algorithm," IEEE Trans. Circuits and Syst., vol. CAS-32, pp.960–963, Sept. 1985.

[32] 高橋　強, 豊田真嗣, 恒川佳隆, 三浦　守, "ブロックLMS適応フィルタの超高速VLSIアーキテクチャ," 計測自動制御学会東北支部第195回研究集会, 195-8, June 2001.

[33] 高橋　強, 豊田真嗣, 恒川佳隆, 三浦　守, "分散演算を用いたブロックLMS適応フィルタの超高速VLSIアーキテクチャ," 計測自動制御学会東北支部第201回研究集会, 201-6, May 2002.

[34] K. Takahashi, Y. Tsunekawa, N. Tayama,"Very High-Speed VLSI Architecture of Block LMS Adaptive Filter Using Distributed Arithmetic," Proceedings of ITC-CSCC'02, pp.678-681, July 2002.

[35] NTTデータ通信株式会社, "PARTHENON User's Manual," 1990.

[36] 牧野昭二, 小泉宣夫, "エコーキャンセラの室内音場における適応特性の改善について," 信学論(A), vol.J71-A, no.12, pp.2212–2214, Dec. 1988.

[37] B. Widrow, J. R. Glover,Jr., J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong,Jr., and R. C. Goodlin,"Adaptive noise cancelling : Principles and applications," Proc. IEEE, vol.63, pp.1692–1716, Dec.1975.

[38] A. Feuer, E. Weinstein, "Convergence Analysis of LMS Filters with Uncorrelated Gaussian DATA," IEEE Trans. Acoustics, Speech, and Signal Processing, vol. ASSP-33, pp.222–230, Feb. 1985.

[39] Simon Haykin,"Introduction to Adaptive Filters," Macmillan publishing Company,New York,1984.

[40] L. Maisel,"Probability, Statics and Random Process,"Simon & Schuster,Inc.,New York,1971.

[41] G.Strang,"Linear Algebra and its Application, "Academic Press,Inc.,New York,1976

[42] 日本テキサス・インスツルメンツ株式会社, "TMS320C5xユーザーズマニュアル ディジタル・シグナル・プロセッサ," 1994.

[43] 谷萩隆嗣, "ディジタル信号処理の理論1 ～3," コロナ社, 1985.

[44] 谷萩隆嗣, "マルチメディアとディジタル信号処理," コロナ社, 1997.

[45] 谷萩隆嗣, "情報通信とディジタル信号処理," コロナ社, 1999.

[46] 電子情報通信学会編, "ディジタル信号処理の応用," コロナ社, 1981.

[47] 辻井重男, "適応信号処理," 昭晃堂, 1995.

[48] 飯國洋二, "適応信号処理アルゴリズム," 培風館, 2000.

[49] 中溝高好, "信号解析とシステム同定," コロナ社, 1988.

[50] 佐藤洋一, "線形等化理論," 丸善株式会社, 1990.

[51] 樋口龍雄, "高度並列信号処理," 昭晃堂, 1992.

## 謝 辞

本論文は，著者が岩手大学工学部電気電子工学科恒川研究室ならびに岩手県立産業技術短期大学校電子技術科において，これまで行ってきた研究成果を取りまとめたものである。

本研究をまとめるに至ったのは，ひとえに恩師岩手大学恒川佳隆助教授，田山典男教授，前情報工学科三浦守教授，前電気電子工学科関享士郎教授の懇切なるご指導と暖かいご配慮によるものであり，ここに衷心より感謝申し上げます。

また，御多忙な時間を割いて御討論いただき，貴重な御意見と御教示を賜った，岩手大学安倍正人教授，柏葉安兵衛教授に深く感謝申し上げます。

岩手大学大学院における研究活動の機会を下さった前岩手県立産業技術短期大学校安藤厚校長に深く感謝申し上げると同時に，これまで暖かい励ましとご助力を頂きました岩手県立産業技術短期大学校職員の方々に厚く御礼申し上げます。

岩手大学電気電子工学科恒川研究室に所属して卒業研究を行い，本研究をなす機会を共にした豊田真嗣氏はじめ卒業生諸氏，並びに本研究をまとめるにあたりご協力を惜しまなかった恒川研究室諸氏のご厚意に心から御礼申し上げます。