平成 12 年度

学士学位論文

ニューラルネットワークを用いた
筋活動電位のパターン認識システムの構築に関する研究

Construction of pattern recognition system of erectoromyogram by neural networks

学籍番号　1010419　中原　昌樹

指導教員　竹田史章　教授

2001 年 2 月 5 日

情報システム工学科

概要

腕の動作により、携帯機器等における動作識別を可能とするために、本論文では被験者の随意運動に伴う生体信号である筋活動電位（EMG：electromyogram）の認識システムを検討する。特に、筋活動電位の登録、認識には学習能力を有しかつ、非線形パターン識別が可能な階層型ニューラルネットワークを用い、シミュレーションにてその有効性と可能性を示す。

abstract

In this paper, we propose a recognition system to enable recognition of operation in portable equipment by operating the arm of testee. Especially, learning and recognition of electromyogram by using neural networks, which have the learning ability and nonlinear pattern identification, is possible.We show the effectiveness and possibility of the electromyogram recognitionby simulation.

キーワード：ニューラルネットワーク、屈筋信号、伸筋信号、生体情報

目次

１．はじめに

従来、被験者の動作を解析するためには、マウス、データグローブ、データスーツ、モーションキャプチャなどのように位置や力の情報を計測するものがある。しかしながら、これらはユーザが直接触れる機器の形状、重量がユーザの身体的特性に合っていない場合、使いにくく、ユーザに不快感を与える。具体的には、マウスが大きすぎたり、腕時計が重過ぎたりするなどである。

近年、携帯電話を代表とする携帯情報端末機器は急速に普及し、Bluetooth[(1)]などの携帯機器の無線通信規格が整いつつあり、これを利用すれば、多くの機器で使用頻度の高い操作を一つの操作装置で集約して行うことも可能である。たとえば、公共交通を利用する際に、携帯電話をマナーモードにする、あるいは、音楽プレーヤの音量を調節するなどの操作を各機器を取り出すことなく、片手で一度に行うことが実現可能であると考えられる。

現在、このように、機器の操作だけを独立した機能として持ち、携帯機器のネットワークをコントロールする装置（総合操作装置と略記）は提供されていない。そのため、総合操作装置を操作する信号には被験者の随意運動に伴い発生する生体信号である筋活動電位を検討する。

生体信号には顔、光彩、指紋、筆圧、音声などがあり、前者の 3 つは身体的特徴、後者の 2 つは行動的特徴と呼ばれる。これらの違いを述べると、行動的特徴はデータ採取に際して心理的抵抗が小さい。そして、身体的特徴は再現性が高く、同一人物から何度でも同じデータを得ることが可能であるといった特徴がある。本研究で使用する筋活動電位は行動的特徴に含まれる。

筋活動電位により制御される総合操作装置は操作性を考え、腕時計型や指輪型であることが望ましいと考えられる。そのため、指や手首の筋活動電位を採取することが考えられるが、手首から得られる筋活動電位は微弱な電位であるために、本研究では、腕からの筋活動電位において最も良好な出力結果が得られた前腕二頭筋にて認識実験を行う。

通常、細胞内側の電位は細胞外側に対して、マイナスの電位である。これは主に K^{+}の細胞内外の濃度差によってもたらされる。このマイナスに偏った電位を静止膜電位という。活動電位とは脱分極による細胞の興奮、すなわち、細胞内側の電位が静止膜電位(約－70mv)を超え、一時的(約 1ms)に正になる一連の反応をいう[(2), (3)]。筋活動電位の発生に際してはまず、遠心性神経のインパルスが神経と筋の接合部である神経筋接合部に到達する。神経インパルスが神経末端より伝達物質アセチルコリンが遊離され、これが神経筋接合部の間隙を越え、筋の膜面に達し、筋線維に興奮を引き起こす。これを膜の脱分極といい、マイナス側にある静止膜電位の平衡状態を破り、プラス側に変化させることである。いくつかの神経パルスの到達により興奮状態が高ま

ると、筋細胞内部は急速に 30〜50mV に増加する。ここで、筋に電極を装着すれば、電位変化を観測することができる。これを筋活動電位と呼ぶ[4]。筋活動電位には筋収縮レベルに応じて発生する、動作、力の入れ具合、運動レベルに応じての情報や運動の柔らかさの情報を含むといった特徴を持つ。

現状の筋活動電位の認識するための機器は、機器が大きく、センサの位置ずれなどにより検出信号が変化し、誤認識がおこる可能性がある。また、筋活動電位は個人差があり、人によっては誤認識、誤動作がおこる可能性がある。そのために機器の誤動作、誤認識は事故を引き起こすことも考えられる。本研究では、筋活動電位の特徴抽出には従来手法である高速フーリエ変換（FFT：Fast Fourier Transform）とニューラルネットワークを利用し[5].[6].[7]、誤認識、誤動作の少ない筋電位認識システムを提案する。

本研究では、手首の動作識別を検討する。実際に起こりうる様々な手首の動作の中から、手首を上下へ動かすという 2 種類の動作が識別可能であるかを検証する。被験者が右手の手首を上に上げる際に使用する筋である橈側（とうそく）手根屈筋、下に下げる際に使用する筋である尺側（しゃくそく）手根屈筋の筋活動電位を検出する。電極から検出された筋活動電位は非常に微弱である。そのため、アンプにより筋活動電位を増幅する必要がある。さらに、電極からアンプまでの配線にはノイズが混入しており、筋活動電位よりも大きな電位をもっている場合が多い。そのため、筋活動電位を採取する際のノイズ対策についても検討する。そして、ノイズ除去を行った筋活動電位を特徴抽出し、このデータをニューラルネットワークへの入力信号とする。使用するニューラルネットワークは階層型で、このニューラルネットワークは文字認識や、画像認識に適している。ニューラルネットワークの出力関数にはシグモイド関数を使用し、非線形分離能力を持たせる。本研究では学習能力を有しかつ、非線形分離能力を有する 3 層の階層型ニューラルネットワークを使用し、認識システムの有用性について検討する[5]。

2．筋電パターン認識システム

2．1 筋活動電位

筋活動電位とは筋を使うことにより、細胞内液と細胞外液のバランスが崩れ、脱分極が起こり発生する。筋活動電位は筋収縮レベルに応じて発生する、動作、力の入れ具合、運動レベルに応じての情報や運動の柔らかさの情報を含むといった特徴を持つ。つまり、個人差はあるものの、筋に力を入れるほど、大きな筋活動電位が発生する。さらに、運動に応じての特徴を持つため、この特徴をつかむことができれば、動作の認識が可能である。つぎに筋活動電位により制御される総合操作装置について述べる。

2．2 総合操作装置の構成

本章では前章で述べた総合操作装置の構成について述べる。携帯機器の操作インターフェースのみが独立した装置は現在販売されておらず、同等の機能をもつ代替品はあまりない。類似の技術として音声認識による操作装置などがあるが、騒音を問題とする公共の場では使用が限定される。本研

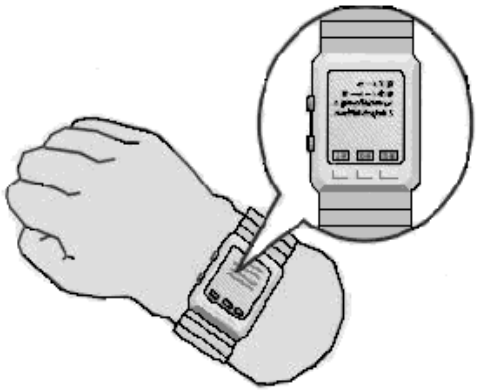


図 2.1　携帯機器総合操作装置の概略図

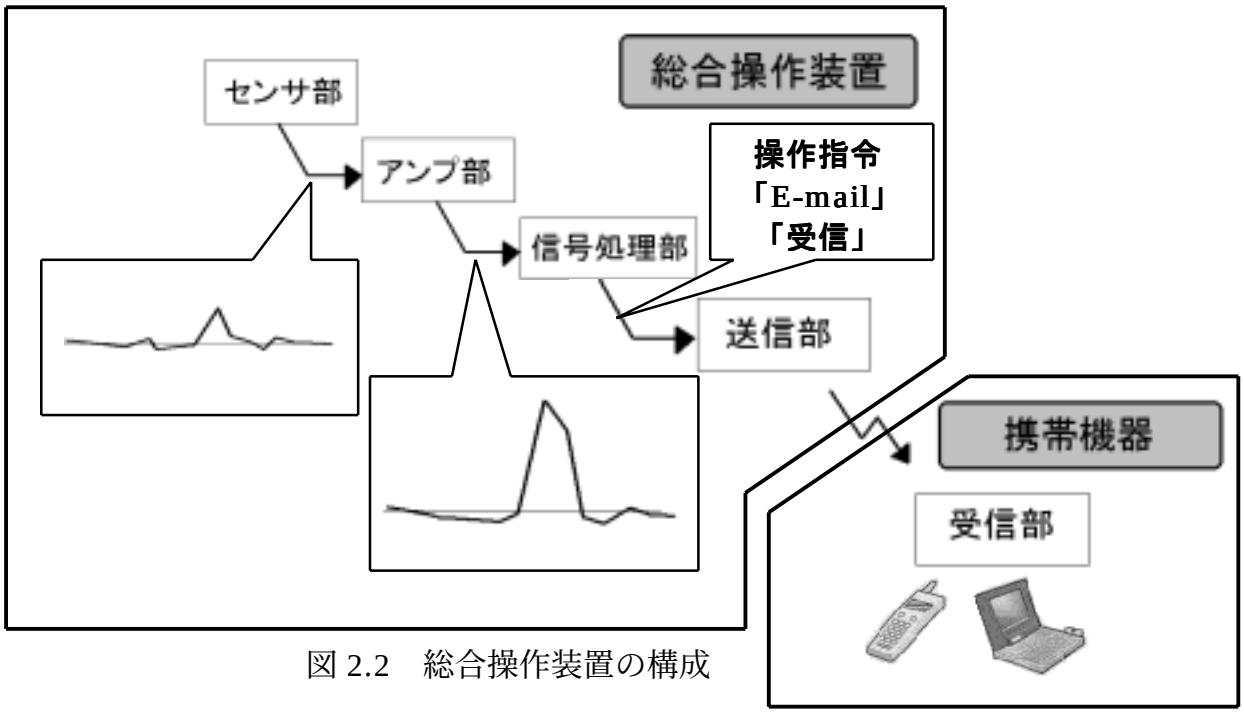


図 2.2　総合操作装置の構成

究で目標とする総合操作装置は片手で周囲に影響を与えず操作できるという点で、使用場所を制限されないというメリットを持つ。総合操作装置の機能としては携帯電話でのメールの受信や、マナーモードへの設定/解除、音楽プレーヤでの鑑賞曲の変更や音量の調節などが考えられる。現在、総合操作装置は操作性を考慮すると、図 2.1 のように腕時計型や指輪型が望ましいと考えられる。そのため、指や手首における筋活動電位を測定することが望ましい。

図 2.2 に総合操作装置の構成図を示す。センサである腕時計などから筋活動電位を検出する。腕時計を装着する手首から検出される筋活動電位は微弱な信号であるために、アンプ部で信号を増幅させ、信号による操作指令の認識を行う。そして、その認識結果を無線で携帯機器に送信し、機器が作動する仕組みである。

手首から検出される筋活動電位は小さく、良好な大きさの電位が得られないことがわかった。そこで、本研究では腕からの筋活動電位を採取するために、最も大きい筋活動電位が検出された前腕二頭筋にて実験を行う。

２．３　システム構成

本研究で提案する筋活動電位のパターン認識システムの構成を図 2.3 に示す。本システムは入力部、信号処理部、データ変換部、認識・登録部から構成される。

入力部では腕に装着する電極より採取される筋電信号はアンプにより増幅される。そして、A/D 変換され、PC（PC：Personal Computer）へ入力される。入力された筋電信号は PC 内の筋電信号出力プログラムにより時系列データ（Tdata と略記）を生成する。さらに、FFT により特徴抽出され、Tdata より FFT データ（Fdata と略記）を生成する。

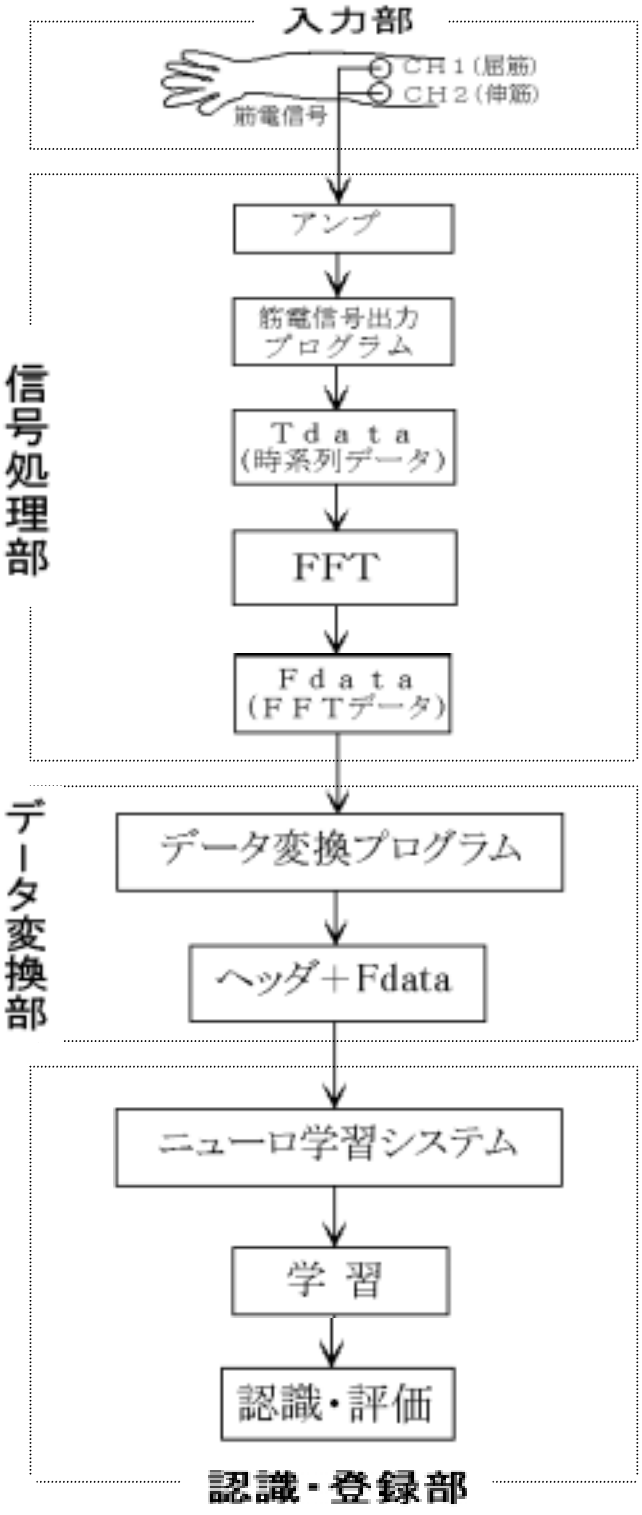


図 2.3　システム構成

出力される Tdata,Fdata は Fdata のみ PC 内に組み込まれているニューロ学習システム（学習システムと略記）に入力される。その際、ニューロ学習システムに入力するためのヘッダ情報が付加される。学習で得られたニューロウェイトを用いて、様々のデータの評価を行う。

２．４　入力部

２．４．１　電極

筋活動電位には表面電極（passive electrode）により計測される表面筋電と針電極（inserted electrode）により計測される深層筋電がある。針電極は直接筋中に刺入するために、計測される深層筋電は、近傍の活動電位のみを計測することができる。このため、特定の筋からの特徴が得やすいという利点を持つが、それと同時に被験者に拒否感を与えるという欠点も併せ持つ[6]。それに対し、表面電極は測定に苦痛を伴わず、容易に着脱でき、電極の腐食等による細胞への影響がないために生体に対して安全である。しかしながら、計測される筋活動電位は複数の筋線維の活動電位（MUAP）が混入した信号であるため、生の記録波形から筋の活動様式や運動制御を分離することは容易ではない[8]。

本研究では携帯等の情報端末の操作装置、入力装置として実用化を考慮し電極には表面電極用いる。表面電極はさらに湿式（電解クリームを使用する）と乾式（電解クリームを使用しない）に分類される。扱いやすさの点からみると、乾式が望ましいが、筋電信号を採取するためには湿式の方が現段階では確実性が高いという点により、湿式を採用する。

湿式の表面電極では金属－電解液、つまり電極の金属－汗の平衡状態の不均衡により、不安定で数 10mv 程度の大きな電位をもつ分極電位が生じるが、これは温度変化、発汗、電解クリームの変化、電極と皮膚間の動き、電解液と金属の種類や性状などにより影響を受ける。この分極電位を抑えるために電極には銀－塩化銀電極（NT-211U，8φ，日本光電（株）製）を用いる。電解クリームには脳波筋電用ペースト（エレフィックス Z-401CE，日本光電（株）製）を使用する。

ラジオや TV 放送の電波、蛍光灯などから発生する電磁波ノイズなどの影響から配線を守るために、図 2.4 のように電極の配線を束ね、

図 2.4　小型生体用電極

配線にノイズが混入しないようにアルミホイルで覆う。さらに、束ねた配線をリング状にしている。リング状にしているのも電磁波ノイズの影響を軽減するためである。配線をリング状にすると、リングの中を通過する電磁波を電流として検出し、増幅する可能性がある。この電磁波の影響を軽減するには 2 つの方法があり、1 つはリングの面積を最小にし、通過する電磁波の量を小さくする方法である。もう 1 つはリングを 2 つ作り、一方のリングに発生した電流と、もう一方に発生したリングの電流を逆向きに作用させ、相殺する方法である。本研究では、後者の方法を採用する。つまり、同じ大きさ、回数で巻いたリングを途中で逆向きにして束ねることにより、リングに通過した電磁波によって発生する電流は互いに打ち消し合い消えてしまう。

図 2.4 に実験で用いた電極を示す。2 個の測定電極 2 組と 1 個の不感電極（アース）の計 5 個を利用する双極誘導法を用いる。このため、図より 5 本の配線が見られるが、その内訳として、2 箇所の筋を測定するためにチャンネル 1，2 分の 4 個の測定電極と 1 つの不感電極を割り当てている。

２．４．２　電極配置

手首を上げる動作、下げる動作の 2 種類を検出するために、電極には 2 チャンネル用意する。橈側手根屈筋にチャンネル 1 の電極を、尺側手根屈筋にはチャンネル 2 の電極を配置する。手首を上に上げた場合、橈側手根屈筋が主に使用され、手首を下に下げた場合、尺側手根屈筋が主に使用される。本研究の電極配置は図 2.9 のように筋線維に対し、垂直に配置する。図 2.10 のように筋線維方向に沿って、測定電極を配置する。そして、筋肉の存在しない肘に不感電極を配置すると、クロストークを防ぐことができる。これは、信号チャネルを多重化することにより、他のチャンネルに信号が伝わることを防げ、特定の筋からの情報を得られるために、良好な出力が期待できる。しかしながら、電極を前者のように配置しない理由は特定の筋からの情報により、特定の動きだけを認識するのではなく、複数の筋から様々な情報を得て、腕の様々な動きの認識できることを目的としているため、電極を図 2.9 のように配置する。電極は筋線維の走行方向に沿って 2cm 間隔で固定する。不感電極はチャンネル 1，2 の電極の中央に配置する。

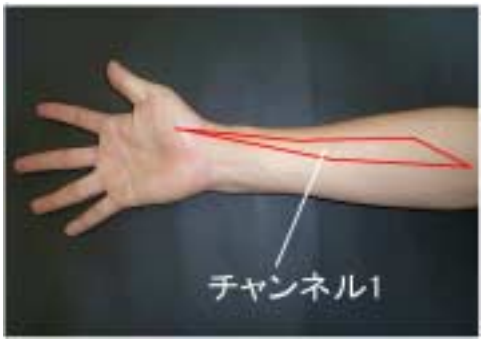


図 2.5　橈側手根屈筋（右手）

図 2.6　尺側手根屈筋（右手）

図 2.7　手首を上に上げた状態（右手）

図 2.8　手首を下に下げた状態（右手）

図 2.9　実験で採用した電極配置

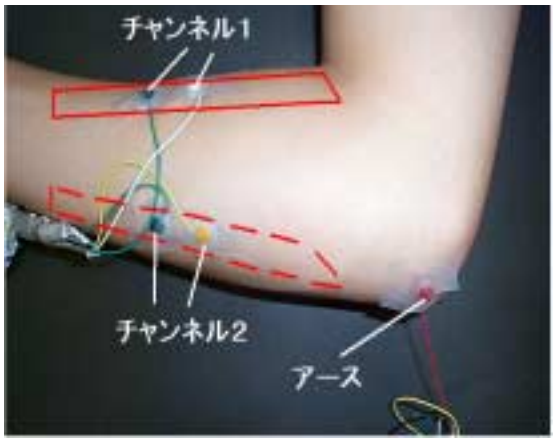


図 2.10　理想と思われる電極配置

２．５　信号処理部

２．５．１　アンプの構成

本節では、まず、アンプの構成である、A/D 変換変換回路について述べる。A/D 変換回路は図 2.11 のように構成される[2]。

はじめに、複数あるアナログ入力信号（筋電信号）の中から、変換する 1 つの信号を選択するためのマルチプレクサがある。これは、入力選択用の数ビットのディジタル信号を与えてアナログスイッチを操作させ、アナログ入力を切り替える仕組みの

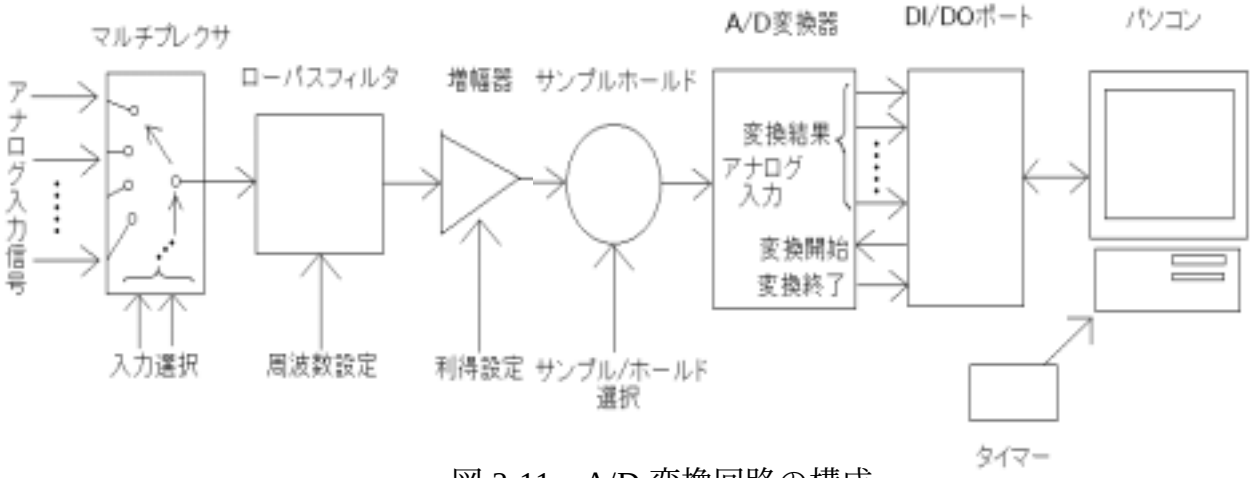


図 2.11　A/D 変換回路の構成

IC である。

フィルタは不必要な信号を取り除くために使用する。フィルタには低い周波数の信号のみを通過させて高い周波数の信号を除去するローパスフィルタ、高い周波数の信号のみを通過させて低い周波数の信号を除去するハイパスフィルタ、ある範囲の周波数の信号のみを通過させるバンドパスフィルタ、ある特定の周波数の信号のみを除去するノッチフィルタなどがある。表面筋電の場合、2、3Hz～2kHz 程度周波数成分のうち、ノイズが混入する可能性の低い 70Hz 以上の周波数成分をローパスフィルタとノッチフィルタで検出する。

増幅器は微弱なアナログ信号を A/D 変換器が扱いやすい大きさまで増幅する。サンプルホールドは正しく A/D 変換を行うために A/D 変換器が変換処理中にアナログ信号が変化しないよう、ある瞬間の値を保持させる。DI/IO ポートは A/D 変換器からの変換出力や、A/D 変換器の制御信号を PC とやりとりするための専用制御回路である。タイマーはサンプリング周期を発生させるものである。タイマー出力がパソコンへ向いているのは変換開始制御を割り込み処理として行うためである。

２．５．２　アンプの設計仕様

筋電信号の主な周波数は、表面電極で測定した場合、数 Ｈｚ（2，3Hz）～2kHz と報告されている。表 2.12 からわかるように、商用周波数ノイズ（70Hz）の影響を避けるため、筋電信号のうち 70Hz～2kHz を抽出する。ノイズ混入を軽減するために差動増幅という手法を利用する。これは双極誘導である 2 個の測定電極 2 組と 1 個の不感電極の間に生じる電位差には筋活動電位は逆相（信号の位相が 180 度ずれる）、周囲ノイズは同相（信号の位相が同じ）で入る。そこで、この測定電極と不感電極間の電位差の差をとると、出力には同相の周囲ノイズは打ち消されるため現れず、逆相

である筋活動電位のみが現れ、ノイズを除去することができる。そして、アンプの筐体はアルミ製である。これは電磁波ノイズの影響を軽減するためである。表 2.12 中の CMRR(common mode rejection ratio, 同相信号除去比）とはノイズ除去能力を示している。出力される筋電信号は、一般にいわれる EMG である。

表 2.12　アンプの設計仕様

低域遮断周波数	70Hz（6次フィルタ）
高域遮断周波数	2kHz（6次フィルタ）
増幅率（ゲイン）	約2000倍 （65dB）
EMG	4ch計測可
出力最大電圧	±5V
入力定格電圧	±2.5mV
CMRR	60dB 以上

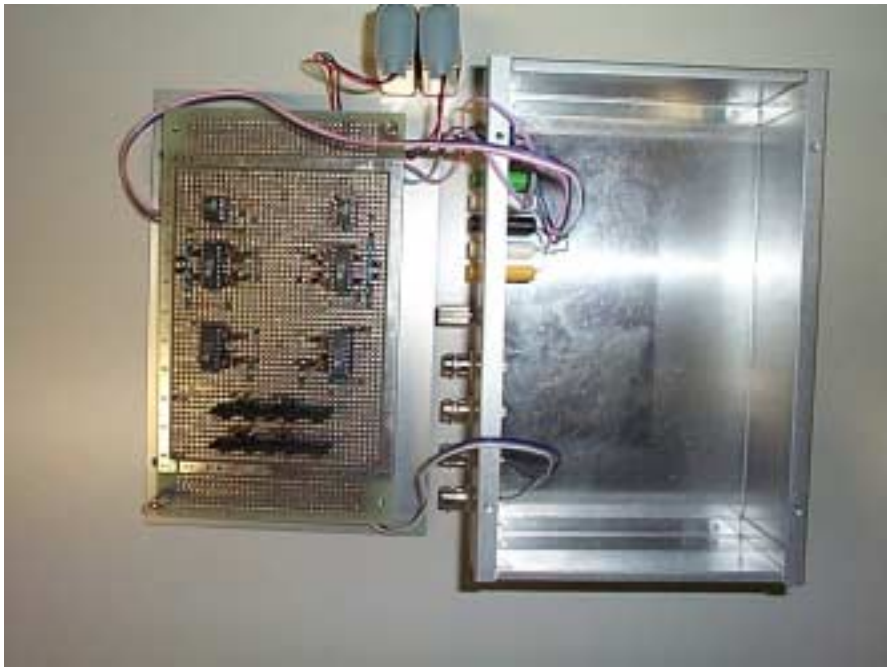


図 2.13　アンプの内部

A/D 変換カードには A/D 変換カード（MREX-5054B，RATOC Systems Inc.製）を用いた。図 2.14 に A/D 変換カードと接続用 BNC コネクタ付 BOX を示す。

図 2.15 に実験で用いるアンプを示す。左から電源スイッチ、電極に接続するためのチャンネル 1 のプラス、マイナス、アース、チャンネル 2 のマイナス、プラス、そして、採取する筋活動電位にノイズが多く混入する場合に使用するアース、A/D変換カードに接続するためのチャンネル 4，3，2，1 である。チャンネル 3，4 はデータ採取部位を増やす際の予備であり、本研究では使用しない。

図 2.14　A/D 変換カードと接続用 BNC コネクタ付 BOX

図 2.15　アンプの外観

図 2.16　システムの完成状態

２．５．３　ＤＦＴ、ＦＦＴ

本項ではまず、筋電信号出力プログラムに組み込まれた FFT について述べる。FFT とは離散フーリエ変換（DFT：Discrete Fourier Transfomer）の演算時間を大幅に短縮させたものである。DFT とはサンプリングされた有限個のディジタル信号に対するフーリエ変換のことである。フーリエ変換とはフーリエ係数を求める処理で、x(t)のフーリエ級数展開は

$$x(t) = a_0 + \sum_{n=1}^{\infty}(a_n \cos 2\pi nt/T + b_n \sin 2\pi nt/T) \tag{2.1}$$

と表せる。a_n, b_n はフーリエ係数で、a_0 は直流成分を表す。直流成分は入力データの平均値で、フーリエ係数は信号 x(t)に含まれる各成分の割合を示す。
この x(t)は式(2.2)のように複素数で表現するので、

$$x(n) = x_r(n) + jx_i(n) \tag{2.2}$$

DFT による変換結果も複素数となり、

$$X(k) = \sum(x_r(n) + jx_i(n))(\cos(2\pi nk/N) - j\sin(2\pi nk/N) \tag{2.3}$$

$$= \sum\{(x_r(n)\cos(2\pi nk/N) + x_i(n)\sin(2\pi nk/N)) + j(x_i(n)\cos(2\pi nk/N) - x_r(n)\sin(2\pi nk/N))\} \tag{2.4}$$

と表される。

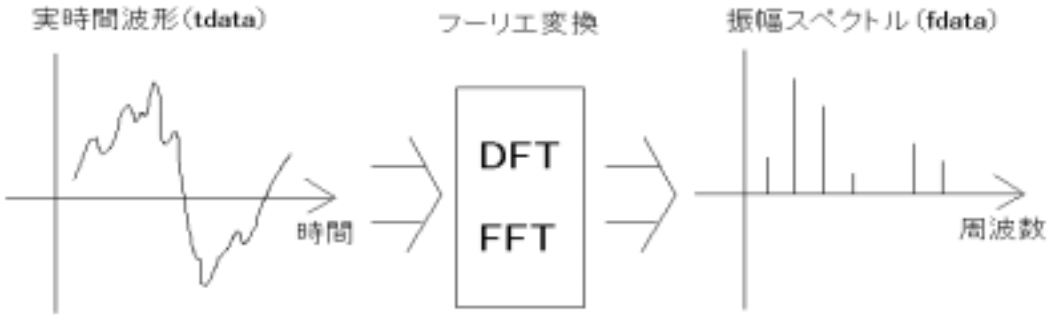


図 2.17　フーリエ変換

FFT の高速化の仕掛けは、演算の過程でビット並びが逆転する現象が起こるために正常な順に並べ替える処理であるビット逆転処理と FFT の基本演算におけるデータの流れであるバタフライ演算の 2 つの手法により複素数の乗算処理を共用できるようになるため、乗算回数が激減し、高速演算処理が可能となる。[9),(10] 表 2.18 に DFT と FFT の演算回数の比較を示す。

表 2.18　DFT と FFT の演算回数

データ数	複素乗算回数		
	DFTの場合	FFTの場合	DFT/FFT
128	16,384	448	37
256	65,536	1,024	64
512	262,114	2,304	114
1024	1,048,578	5,120	205
2048	4,194,304	11,264	372
4096	16,777,216	24,576	683

データ数n個のフーリエ変換を行う場合、通常、n^2回の積和演算が必要とするが、FFT アルゴリズムを用いると $n\log_2 n$ 回と大幅に減らすことができ、膨大な時間を要す DFT の演算時間が短縮できる。

２．５．４　筋電信号出力プログラム

ここで、筋電信号出力プログラムにより出力される Tdata に対し、FFT 処理を行ったものを Fdata とする。Fdata は振幅スペクトルといい、

$$Xn = \sqrt{a_n^{\,2} + b_n^{\,2}} \qquad (2.5)$$

と表す。

手首を上げると橈側手根屈筋が主に使用され、橈側手根屈筋であるチャンネル 1 からの筋活動電位が増加する。ここで、手首を上げた時の波形を図 2.19，図 2.20 に示す。図 2.19，図 2.20 において大きな波形を示すのはチャンネル 1 である橈側手根屈筋の出力波形である。逆に小さな波形を示すのはチャンネル 2 である尺側手根屈筋の出力波形である。図 2.19 は被験者が手首を上げた時の実時間波形（Tdata）を図 2.20 は被験者が手首を下げた時の振幅スペクトル（Fdata）を示す。

同様に、手首を下げると尺側手根屈筋が主に使用され、尺側手根屈筋であるチャン

ネル 2 からの筋活動電位が増加する。図 2.21，図 2.22 において大きな波形を示すのはチャンネル 2 である尺側手根屈筋の出力波形である。逆に小さな波形を示すのはチャンネル 1 である橈側手根屈筋の出力波形である。

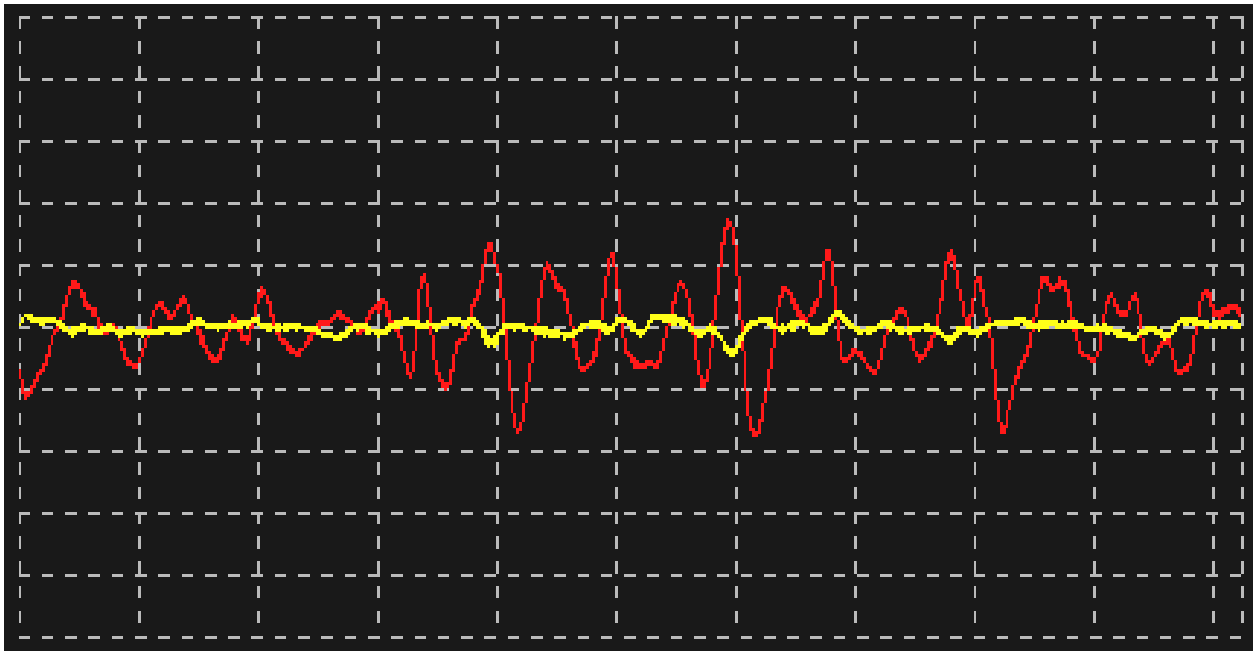


図 2.19　手首を上げた時の実時間波形（Tdata）

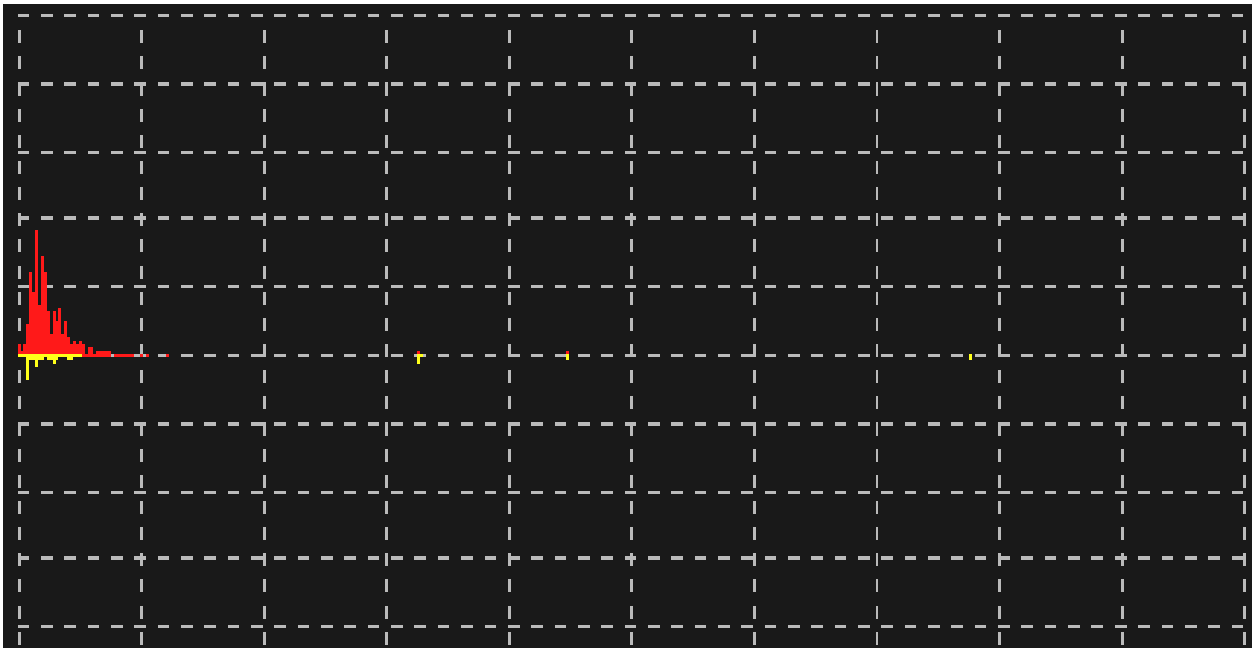


図 2.20　手首を上げた時の振幅スペクトル（Fdata）

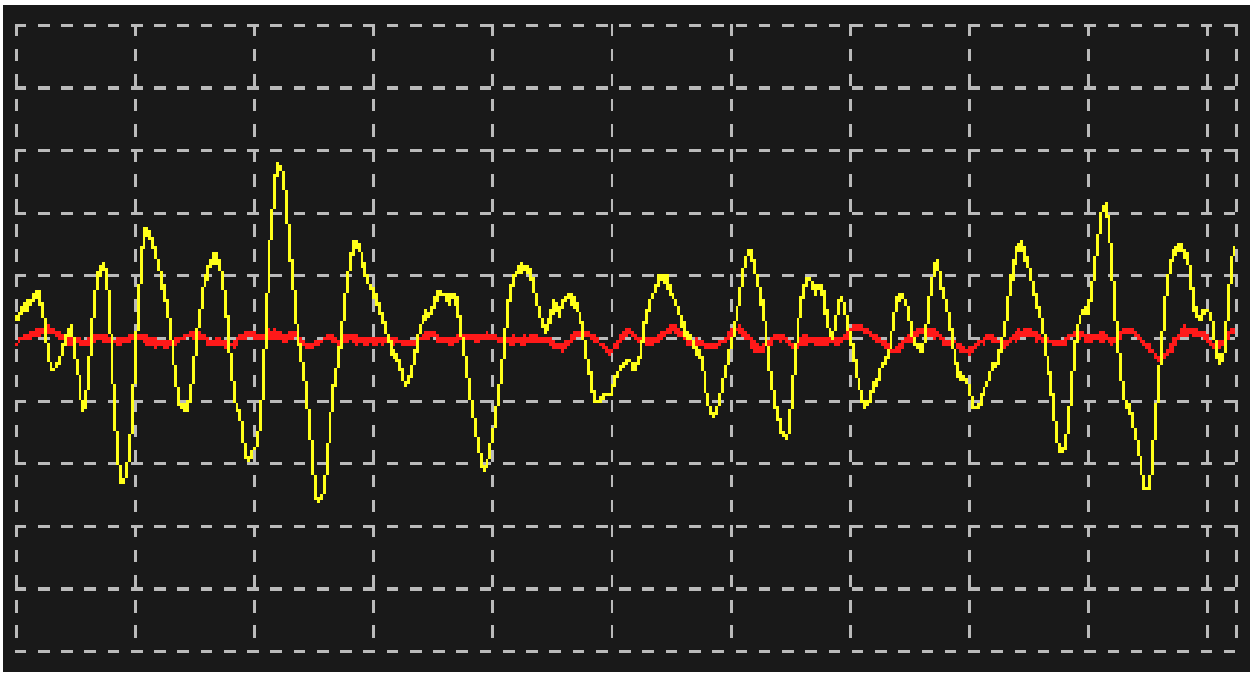


図 2.21　手首を下げた時の実時間波形（Tdata）

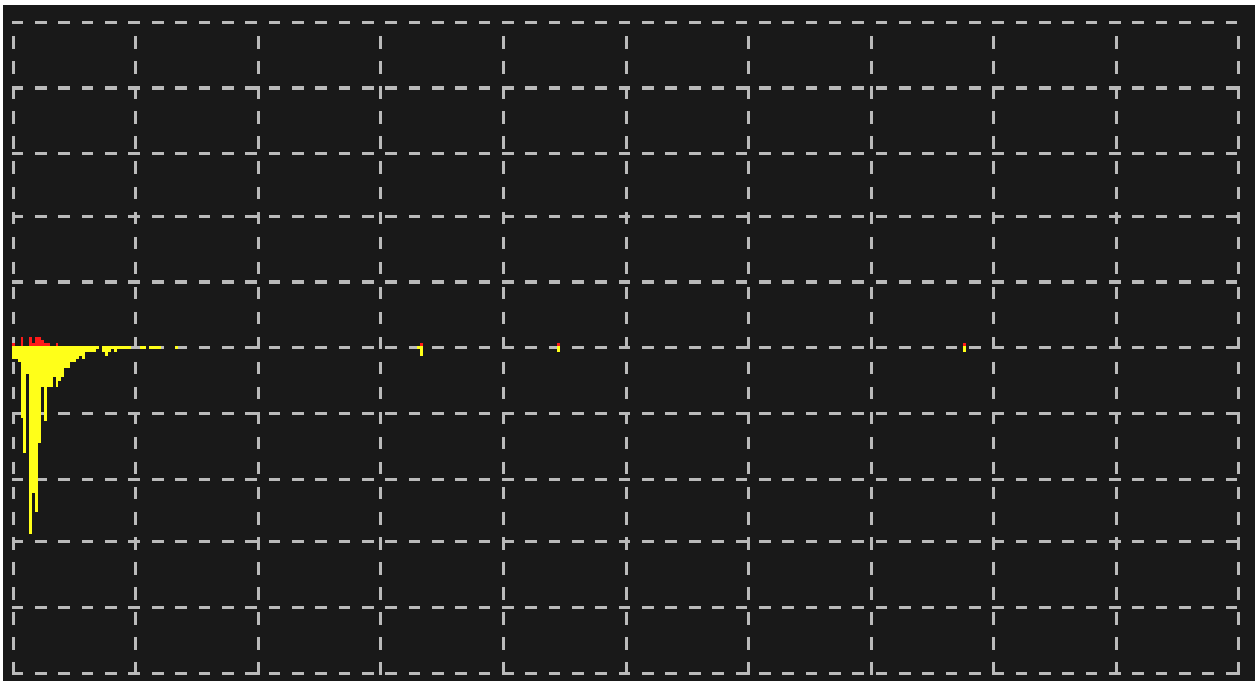


図 2.22　手首を下げた時の振幅スペクトル（Fdata）

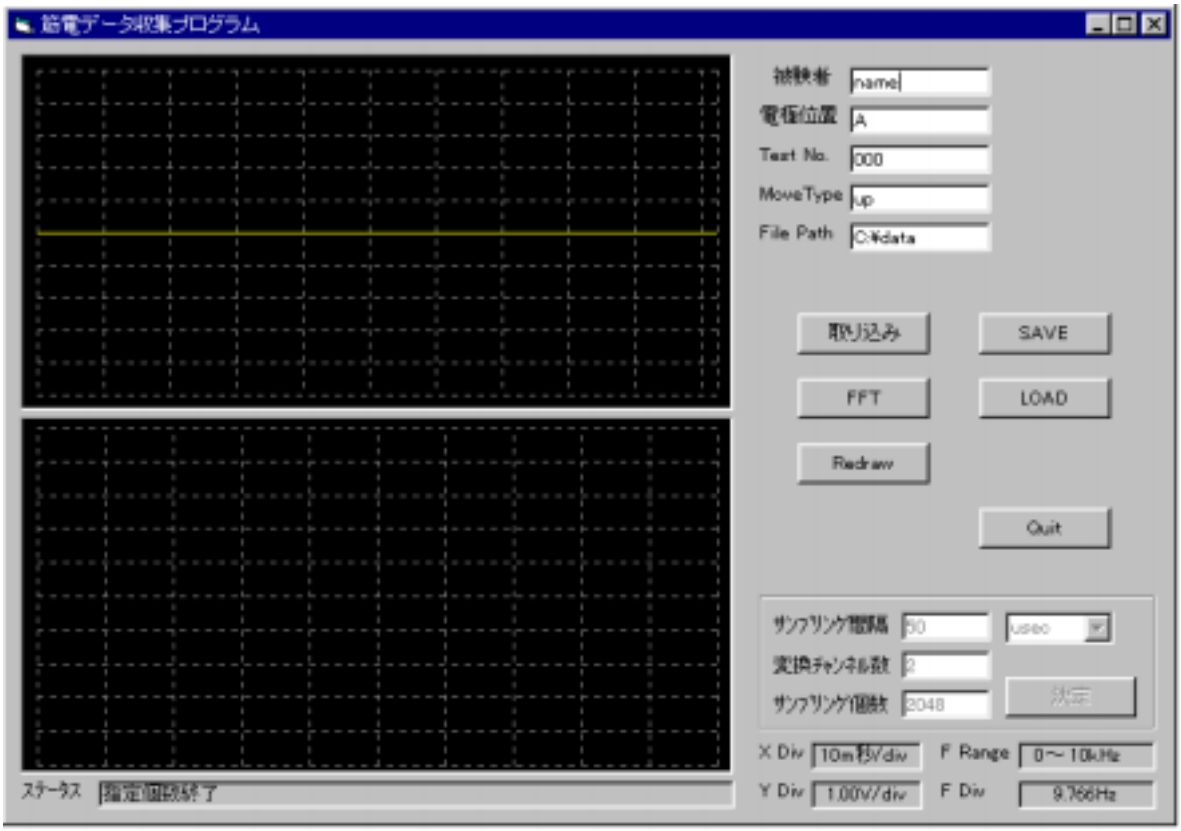


図 2.23　筋電信号サンプルプログラム実行画面

Fdata は(2.5)式よりマイナスの値をとらないが、図 2.21，図 2.22 のようにチャンネル 2 の Fdata がマイナスの値を示す理由はチャンネル 1，2 の波形が重なり、見づらいため、チャンネル 2 から出力されるの Fdata に－1 を掛け、グラフを反転させる。

図 2.23 は筋電信号出力プログラム実行画面を示す。上の画面には実時間波形を、下の画面には振幅スペクトルを出力する。このプログラムにより保存されるファイル名は”nameA001u.txt”となり、name が被験者の名前、Aは採取位置、001 は採取カウント数、u は手首を上に上げた状態、d は手首を下げた状態でのデータを示す。

図 2.24 は筋電信号出力プログラムから出力されるデータの一部である。データはアスキー形式で、左列から順に時間間隔、チャンネル 1 から出力される Tdata、チャンネル 2 の Tdata、周波数のサンプリング間隔、チャンネル 1 の Fdata、チャンネル 2 の Fdata である。

サンプリング点が 2048 個のため、採取されるデータは 2048 行出力される。Fdata は 2048 の半分である 1024 個のデータが必要であればよいので、1025 行目以降のチャンネル 1，2 の Fdata には 0 を代入する[11]。

0	4.8828124	-.10742187	0	16.501464	1.3256835
.00005	.14404296	-.11962890	9.765625	8.4778502	1.7077239
.0001	.18798828	-4.3945312	19.53125	8.8456195	1.5369357
.00015	.23925781	-4.8828125	29.296875	10.517392	1.8422488
.0002	.22705078	-3.4179687	39.0625	14.418730	1.1642197
.00025	.25146484	-8.7890625	48.828125	19.366519	3.3676475
.0003	.33203125	-.11962890	58.59375	54.627328	15.834719
.00035	.36376953	-5.8593750	68.359375	82.145941	15.136828
.0004	.40039062	-7.3242187	78.125	61.806792	12.065231
.00045	.37109375	-7.3242187	87.890625	113.03237	9.8222154
.0005	.38818359	-7.8125000	97.65625	128.10333	8.9472939
.00055	.43457031	-9.5214843	107.42187	133.86509	6.4699400
.0006	.43457031	-6.3476562	117.1875	5.1465763	23.716577
.00065	.45898437	-8.5449218	126.95312	184.37469	18.267764
.0007	.44433593	-.10009765	136.71875	121.33034	11.624175
.00075	.43212890	-4.3945312	146.48437	54.691657	2.7300767
.0008	.46142578	-4.1503906	156.25	64.655926	6.0394018
.00085	.4296875	-3.4179687	166.01562	151.67315	34.640464
.0009	.42236328	-8.7890625	175.78125	182.24655	19.869483
.00095	.44677734	-9.7656250	185.54687	101.13924	31.929321
.001	.43701171	-2.6855468	195.3125	135.14544	15.692424
.00105	.45410156	-1.2207031	205.07812	59.416526	21.793864

図 2.24　筋電信号出力プログラムの出力データ（一部）

２．６　データ変換部

本節では前項により出力された Fdata を学習システムに入力するため、ヘッダ情報を Fdata に付加するデータ変換プログラムについて述べる。

２．６．１　データ変換プログラム

筋電信号出力プログラムに出力されるデータは表 2.24 で示すように時間間隔、チャンネル 1，2 の Tdata、周波数のサンプリング間隔、チャンネル 1，2 の Fdata が記述されている。

学習システムへの入力データは表 2.25 に示されるようにヘッダ情報、画像データ、スラブ値データで構成される。

スラブ値データとは筋電信号出力プログラムにより出力されるチャンネル 1，2 の Fdata を示す。

図 2.26 に示すように、学習システムで設定するニューラルネットワークの入力層細胞数は 50 である。各 1024 ずつ存在するチャンネル 1，2 の Fdata からそれぞれ 25 個ずつ計 50 個のデータをニューラルネットワークへの入力データとする。このデータを SLAB1～50 とする。SLAB1～25 にはチャンネル 1 の Fdata から抽出する 25 個、SLAB26～50 はチャンネル 2 の Fdata から抽出する 25 個を割り当てる。た

とえば､図 2.24 の場合､チャンネル 1, 2 の Fdata の上から 25 個を抜き出すと SLAB1 は 16.501464、 SLAB2 は 8.4778502、 SLAB26 は 1.3256835 となる。この SLAB1～50 の抽出方法については 2.6.2 項で触れる。

さらに、図 2.24 に示すチャンネル 1，2 の Fdata はアスキー形式のため、学習システムへの入力形式であるバイナリ形式に変換する。

表 2.25　データ１枚分のフォーマット

ヘッダ情報	（64byte）
ダミーデータ	（4byte）
スラブ値データ	（200byte）

データの解説をすると、ヘッダに 64byte、ダミーデータに 4byte（float 型）を確保する｡ダミーデータとは学習システムの汎用性を保つために本来は画像データが入力される。しかしながら、本研究である筋電パターン認識システムには不必要であるため、領域の確保のみを行う。Fdata はプログラム中、float 型（4byte）で宣言されているため、スラブ値データには $(25+25)\times 4 = 200$ byte を確保している。

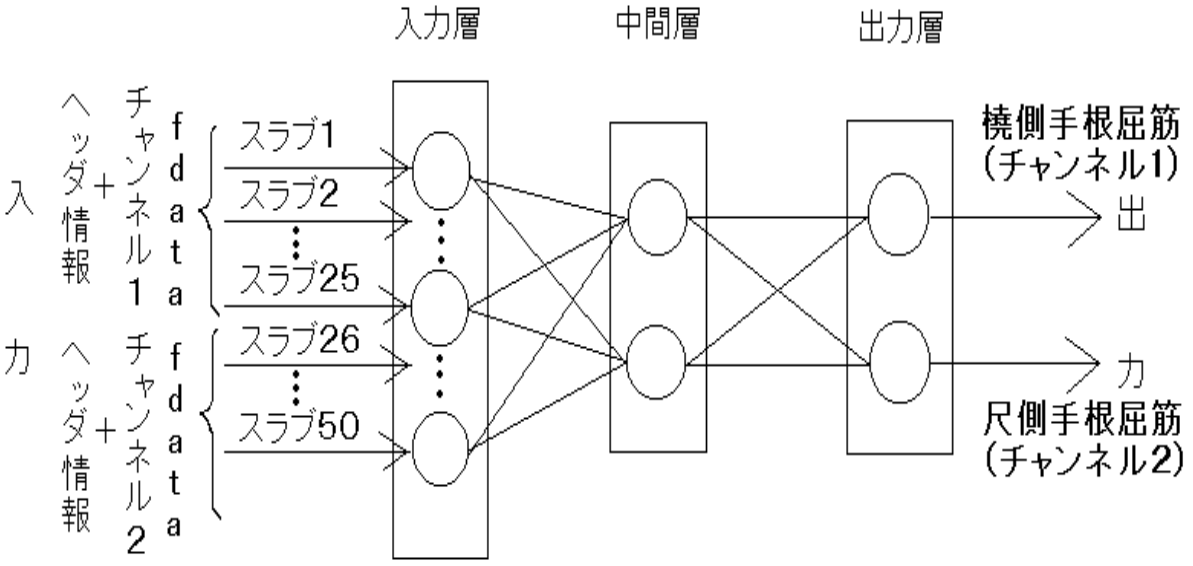


図 2.26　ニューラルネットワークへの入力データモデル

表 2.27　ヘッダ情報のフォーマット

予約領域1	(4byte)
データ1枚分の総サイズ	(4byte)
パターン番号	(1byte)
金種	(1byte)
方向	(1byte)
国情報	(1byte)
通し番号	(2byte)
機種	(1byte)
号機	(1byte)
種別	(1byte)
センサ番号	(1byte)
画像のXサイズ(画素数)	(2byte)
画像のYサイズ(画素数)	(2byte)
センサ有効域チャネル番号(L)	(2byte)
センサ有効域チャネル番号(H)	(2byte)
予約領域2	(2byte)
センサ分解能(X方向)	(1byte)
センサ分解能(Y方向)	(1byte)
予約領域3	(6byte)
紙幣の中心画素のX座標	(2byte)
紙幣の中心画素のY座標	(2byte)
画像データのオフセット	(4byte)
予約領域4	(4byte)
スラブ値の種類数	(1byte)
1種類目のスラブ値のマスクID	(1byte)
1種類目のスラブ値の個数	(1byte)
予約領域5	(5byte)
スラブ値のオフセット	(4byte)
予約領域6	(4byte)

表 2.27 は表 2.25 のデータ 1 枚分のヘッダ部分の構成を示すものであり、表 2.27 は紙幣の場合を示し、本研究では筋電パターン認識システム用に応用する。

・予約領域

学習システムを拡張するために用意している。

・データ 1 枚分の総サイズ

上記で示したヘッダ情報 64byte、画像データ 4byte、スラブ値データ 200byte の計 268byte である。

・画像フレームの X サイズ、Y サイズ

画像データの大きさ。

・画像データのオフセット

ヘッダの先頭を 0 バイト目とする際、画像データの先頭からのバイト数。

・1 パターン目のスラブ値の数

ニューラルネットへの入力層数。

・スラブ値データのオフセット

ヘッダの先頭を 0 バイト目とする際、スラブ値データの先頭からのバイト数。

ここで、採取したデータの集合をデータファイルとし、データファイルの構成を図 2.28 に示す。

データファイル名の”emg01.db”で同じ動作のファイルを 1 つに管理する。

図 2.29、図 2.30 に、このプログラムの入力画面と出力結果を示す。

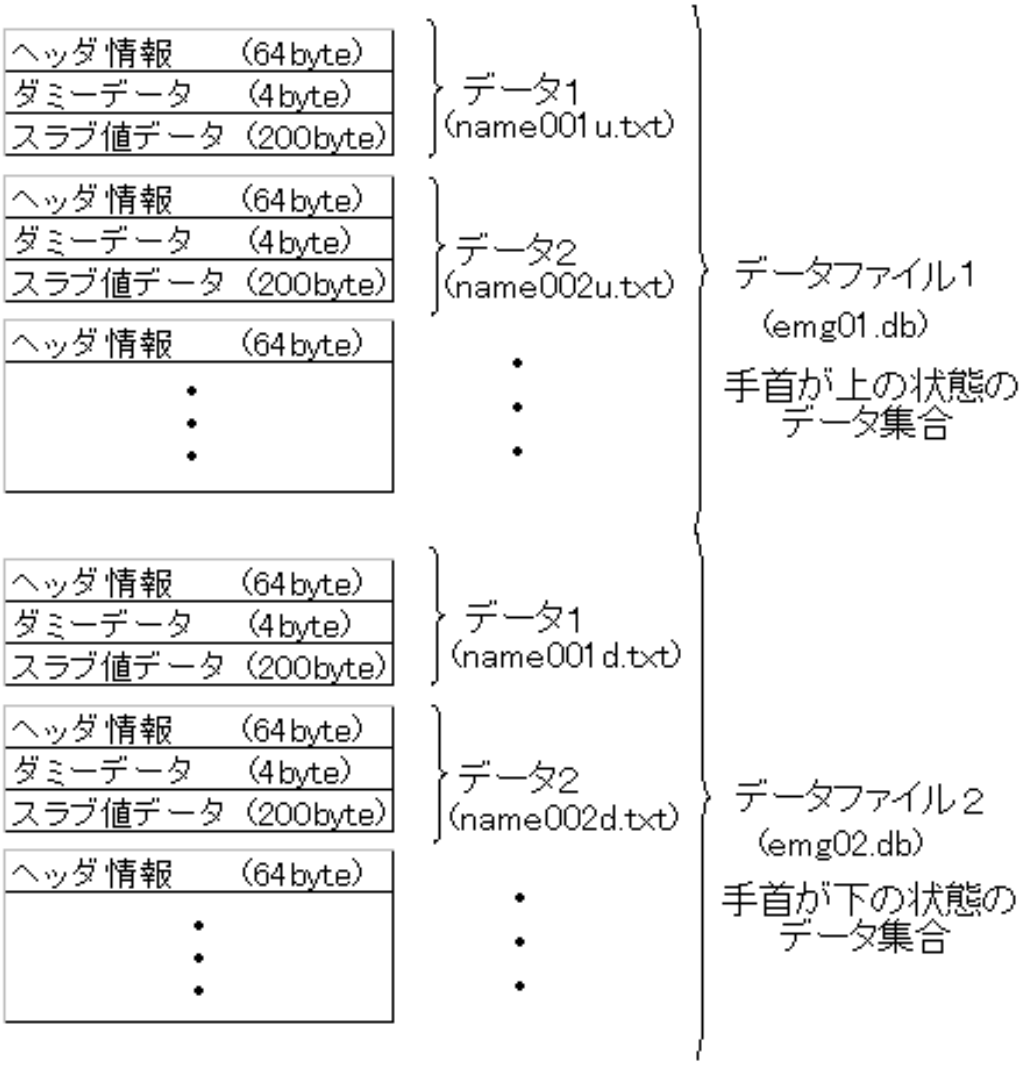


図 2.28　データファイルの構成

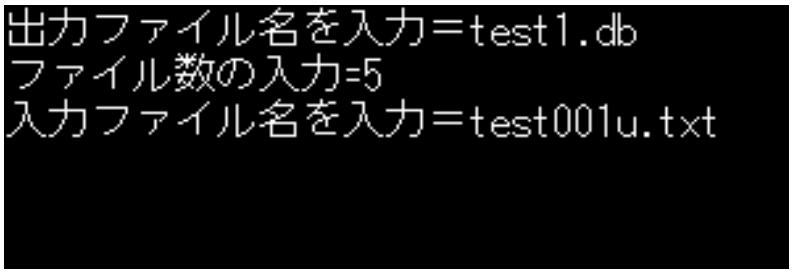


図 2.29　入力画面例

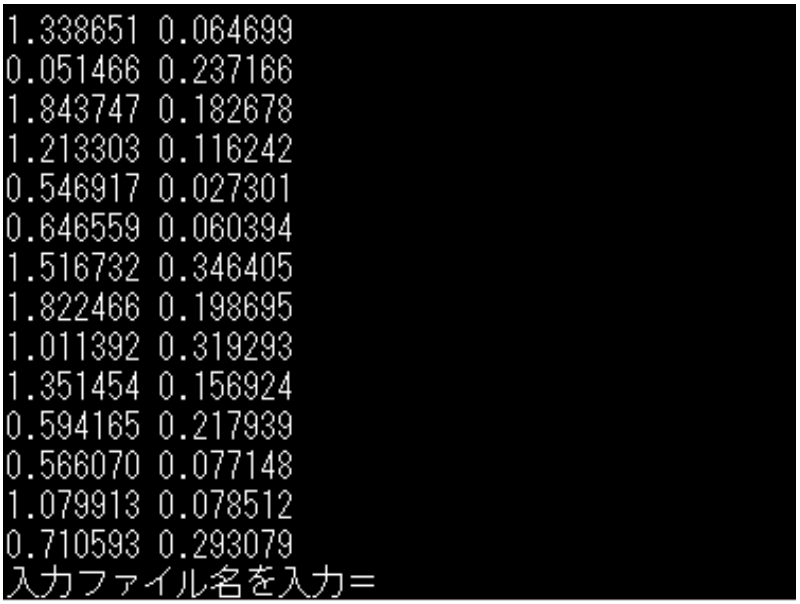


図 2.30　出力画面例

図 2.30 の左列がチャンネル 1 の Fdata、右列がチャンネル 2 の Fdata である。

２．６．２　データ抽出

採取される Fdata は 2 チャンネルで 1024×2=2048 個である。ニューラルネットワークの入力層数が 50 のため、2048 のデータから、いかにして 50 個の情報を得るかについて検討する。

ここで、2 種類の処理方法を提案する。チャンネル 1，2 が存在するために、チャンネル 1 つに対して、SLAB1～25、SLAB26～50 の各 25 ずつの入力を行う。まず、1 つ目は商用周波数ノイズ(1～60Hz)を避けるために、それぞれチャンネル 1，2 の 60~1019 個目の Fdata を抽出する。そして、各チャンネルの総和を 24 等分し、それぞれの区間の和をとる。それぞれの区間の和を各チャンネルの総和で割った数値 24×2=48 個を SLAB1～24，SLAB26～49 とする。SLAB25，SLAB50 には各チャンネルの総和を適当な値である 30000 で割る。これは学習システムに対し、入力の

ダイナミックレンジを合わせるためである。この手法を手法 1 と呼ぶこととする。

2 つ目は前者で行った Fdata60～1019 個目のデータの範囲を半分に狭め、60～515 個目対し、同様の処理を行う。これを手法 2 と呼ぶこととする。

上記の 2 手法以外に各チャンネルの Fdata1～25 の各点であるデータを SLAB1～25 と設定する場合も良好な結果が得られることがわかった。しかしながら、この手法を採用しない理由は 60Hz 以下はノイズが混入しやすく、さらに、周波数のデータは 10Hz や 20Hｚの範囲での振幅の変化が大きく、そのために Fdata の 1 周波数（1 つの数字）をそのまま使う場合、データごとの誤差に弱い検出になるからである。

２．７　登録・認識部

２．７．１　ニューラルネットワーク

ニューラルネットワークとは人間の脳の情報処理を工学的にモデル化したネットワークのことである。

本節では、学習システムに用いている階層型ニューラルネットワークについて解説する。

ニューラルネットワークの計算メカニズムには、大きく分類するとして 2 つあり、一方は階層的型ネットワーク（層状結合ネットワーク）で、もう一方は相互結合のある非階層的なネットワークである[12]。

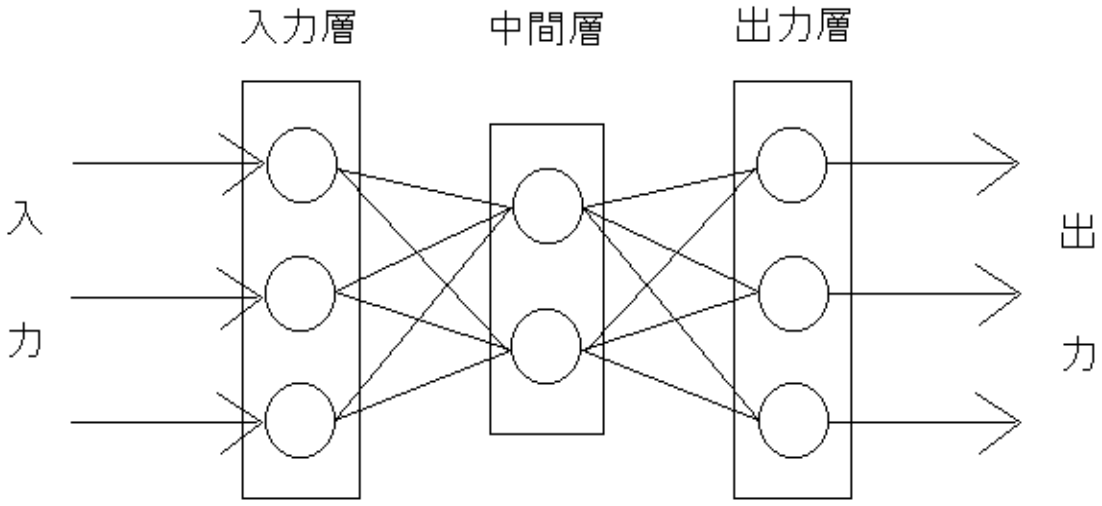


図 2.31　階層型ネットワーク

本研究で用いた計算アルゴリズムは中間層が 1 つからなる 3 層の階層型ネットワークである。図 2.31 の丸印で表されたのがニューロンで、ネットワークのノードにあたる処理要素で、多入力、一出力の素子である。

各ニューロンは、入力層、中間層、出力層のそれぞれ独立した層に配置する。各層のニューロンはそれより 1 つ前の層のニューロンから入力を受ける。各層間では、すべてのニューロンが結びついている。

＜情報の伝達＞

ニューラルネットワークにおいて、入力層に与えられた入力パターンが、いかにして伝達され、出力層から出力されるかについて説明する。

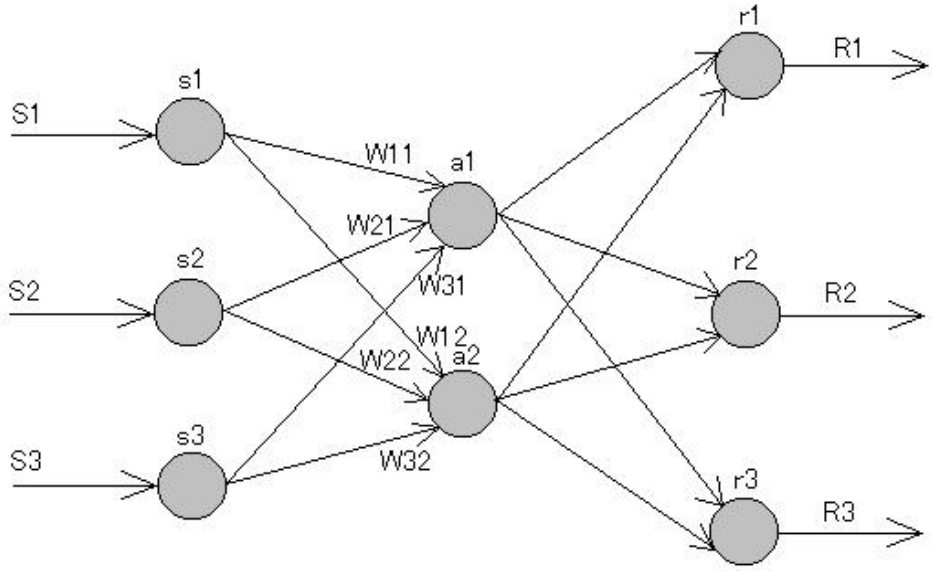


図 2.32　情報の伝達

図 2.32 で、s1，s2，s3 はそれぞれ入力層の第 1，2，3 ユニット、a1，a2 はそれぞれ中間層の第１,２ユニット、r1，r2，r3 はそれぞれ出力層の第 1，2，3 ユニットである。また、S1，S2，S3 は入力層への入力値（スラブ値)、R1，R2，R3 は出力層の出力値を表す。

入力層の第 I ユニットから中間層の第 j ユニットへの結合の重み（ウェイト）を Wij とする。ここで、ウェイトとはその結合が結ぶユニット間の相互作用の強さを表す。

中間層と出力層の間の情報伝達方法も全く同じであるため、入力層と中間層を例にとり説明する。

<ユニットの入力値>

入力層の第 I ユニットの出力値を Si とすると（入力層のユニットは入力値と出力値が同じ）、中間層の第 j ユニットの入力の総和 Ij はつぎのようにして求められる。

$$I_1 = W_{11}S_1 + W_{21}S_2 + W_{31}S_3$$
$$I_2 = W_{12}S_1 + W_{22}S_2 + W_{32}S_3 \qquad (2.6)$$

となり一般的に、

$$I_j = \sum_i W_{ij}S_i \qquad (2.7)$$

と表すことができる。

<ユニットの出力値>

中間層の第 j ユニットの入力の総和を Ij とすると、中間層の第 j ユニットの出力値 Oj はつぎのようにして求めることができる。

$$O_j = f(I_j) \quad \text{(f は入出力関数)} \qquad (2.8)$$

このようにして、入力層の各ユニットの出力値と、入力層の中間層の各ユニット間のウェイトから中間層の各ユニットの出力値を得ることができる。

同様にして、中間層の各ユニットの出力値と、中間層と出力層の各ユニット間のウェイトから出力層の各ユニットの出力値を得ることができる。

＜中間層・出力層の入出力関数＞

中間層・出力層の各ユニットの入出力関数としてはシグモイド関数

$$f(x)=\frac{1}{1+\exp\left(\frac{-x+\theta}{T}\right)} \tag{2.9}$$

を使用する。(2.9)式において、x は各ユニットへの入力値で、f(x)はそのユニットの出力値です。T はネットワークの温度と呼ばれる正の数で、T が大きくなるほどグラフはなだらかになり、θはユニット単位のしきい値を表す。

ここでシグモイド関数のグラフを示す。（θ=0）

T は後述する総合誤差に比例して、1.3 から 0.7 まで変化させる。この操作は学習プログラムが自動的に行う。

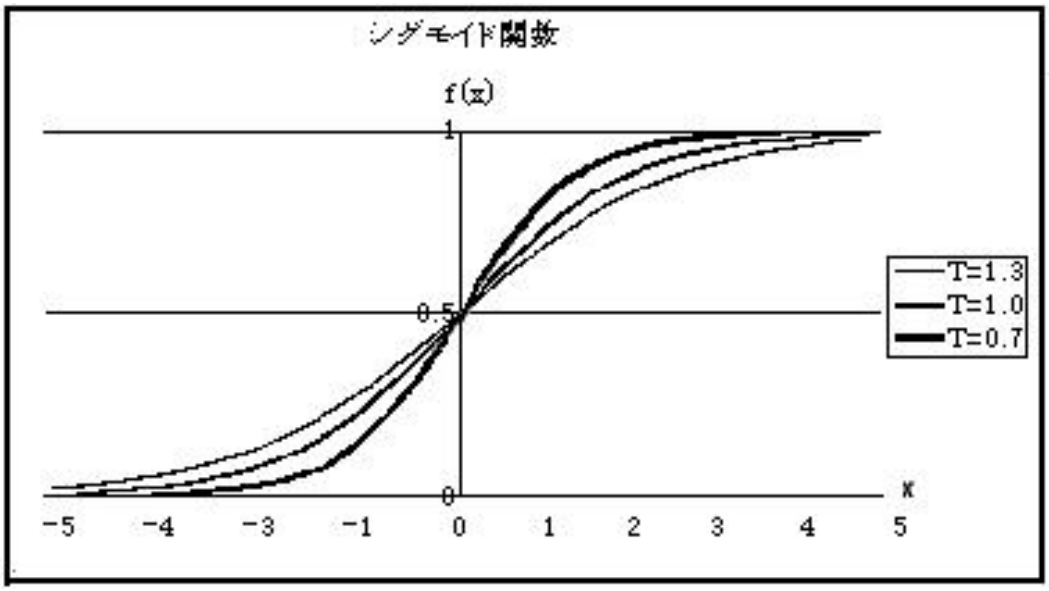


図 2.33　シグモイド関数

＜学習アルゴリズム＞

先に述べた情報の伝達の項で、入力層に与えられた入力パターンが中間層を経て変換され、出力層から出力パターンが得られることを示したが、入力パターンに対して期待する出力パターンを得るためには、各ユニット間のウェイトを適切な値にする必要がある。本項ではこのウェイトを適切な値に設定するための学習アルゴリズムにつ

いて述べる。

学習方法としては、誤差逆伝搬（ＢＰ：Error Back Propagation）アルゴリズムを使用する。具体的には、いくつかの入力パターンの例（学習データ）を与え、その時の出力パターンと期待する出力パターン（教師値）との誤差が減少するようにウェイトを修正する。

ある入力パターンを与えた時の出力層の第 j ユニットの出力を Oj、このときの出力層の第 j ユニットの期待値（教師値）を Tj とすると、第 j ユニットの誤差 Ej はつぎのようにして求める。

$$E_j = \frac{1}{2}(T_j - O_j)^2 \tag{2.10}$$

したがって、1 つの学習パターン P における出力層の誤差 Ep はつぎのようにして求める。

$$E_p = \frac{1}{2}\sum_j (T_j - O_j)^2 \tag{2.11}$$

ここで、全学習パターンの誤差の総和を E とし、これを総合誤差と呼ぶこととする。

$$E = \sum_p E_p = \frac{1}{2}\sum_p \sum_j (T_j - O_j)^2 \tag{2.12}$$

上記の総合誤差 E が最小になるように各ユニット間のウェイトを修正する。
修正の方法としては最急降下法を用いる。具体的には、各学習パターンごとに、各ユニットの誤差 Ej が最小となる方向へ微小な変更を加えていく。

t 回目の学習における k-1 層の第 I ユニットから k 層の第 j ユニットへのウェイト Wij の修正量Δwij(k-1,k)(t)は(2.13)式のようにして求める。

$$\Delta W_{ij}^{k-1,k}(t) = -\varepsilon \delta_j^k O_j^{k-1} + \alpha \Delta W_{ij}^{k-1,k}(t-1) + \beta \Delta W_{ij}^{k-1,k}(t-2) \tag{2.13}$$

上式において”ε”は学習定数、”α”は慣性定数、”β”は振動定数である。この式を改良型ＢＰ法と呼ぶこととする。

また、”δj(k)”は k 層の第 j ユニットの一般化誤差で、k 層が出力層の場合と中間層の場合によって算出方法が異なる。以下に一般化誤差の算出方法を示す。

k 層が出力層の場合、(Ij(k)は k 層の第 j ユニットの入力総和)

$$\delta_j^k = \left(T_j - O_j^k\right) f'\left(I_j^k\right) \tag{2.14}$$

k 層が中間層の場合、（ただし、m は出力層のユニット番号）

$$\delta_j^k = \left(\sum_m W_{jm}^{k,k-1} \delta_m^{k+1} \right) f'\left(I_j^k\right) \tag{2.15}$$

＜学習パラメータ＞

前項で示した(2.13)式の“ε”の項は大きな値を設定するとウェイトの修正量が大きくなり、学習は早くなるが、限度をこえると逆に学習が収束しなくなる。総合誤差が上下するときは学習定数を小さくし、誤差の減少速度が小さいときは学習定数を大きくする必要がある。この操作は学習プログラムが自動的に行うため、ユーザーは学習開始時の学習定数の初期値を後述する環境設定ファイルで設定する必要がある。“α”の項は総合誤差の振動を減らし、学習の収束を加速させる働きをする。“β”の項は総合誤差を上下に振動させ、極小値から脱出させる働きをする。

２．７．２　学習システム

この学習システムは後述に示すプログラムを集合体であるバッチファイルである。各プログラムの動作の説明を行い、学習システム全体の流れを説明する。

２．７．３　環境設定ファイル

ネットワークの構成はこの環境設定ファイルで設定する。設定項目は入力層の細胞数、中間層の数、中間層の細胞数、出力層の細胞数、データファイルのパターン数、1 パターンのデータの個数、(2.13)式における学習定数、慣性定数、振動定数等を環境設定ファイル（テキスト形式）で決定する。

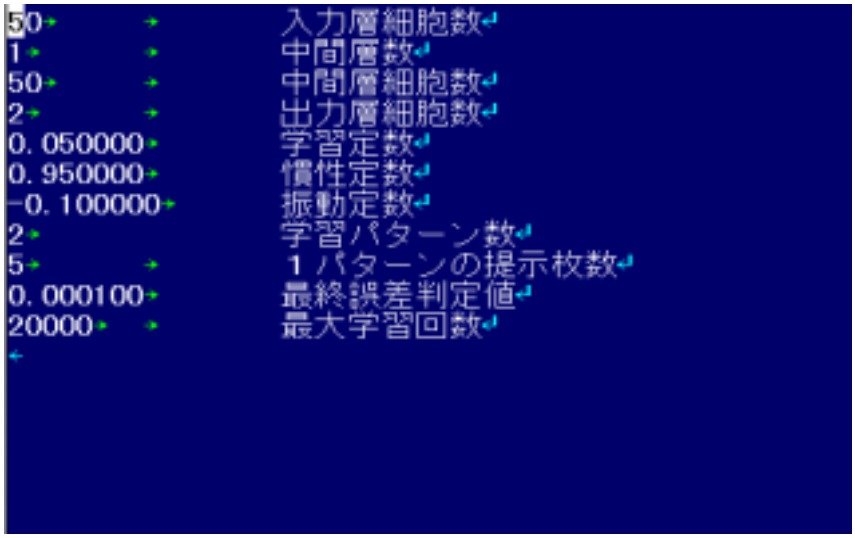


図 2.34　環境設定ファイル

２．７．４　データベース設定ファイル

データファイルのパス名をパターン番号順に 1 行に 1 ファイルずつデータベース設定ファイル（テキスト形式）に記述する。

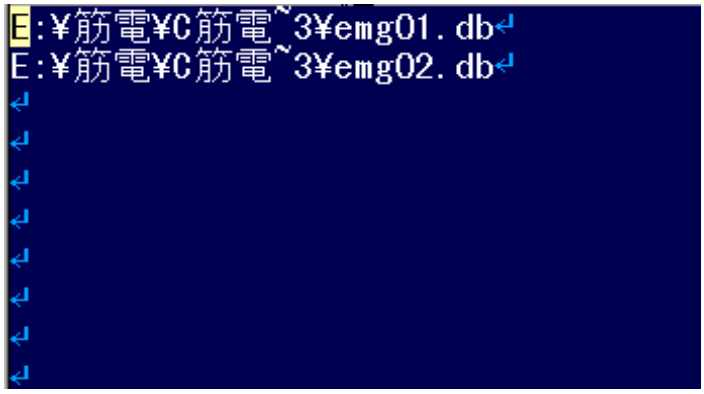


図 2.35　データベース設定ファイル

２．７．５　学習データ抽出

ここでは環境設定ファイルで指定した 1 パターンの学習枚数分だけデータベース設定ファイルに登録されたデータファイルの先頭から取り出し、学習データファイル（LDATA1，2）を作成する。

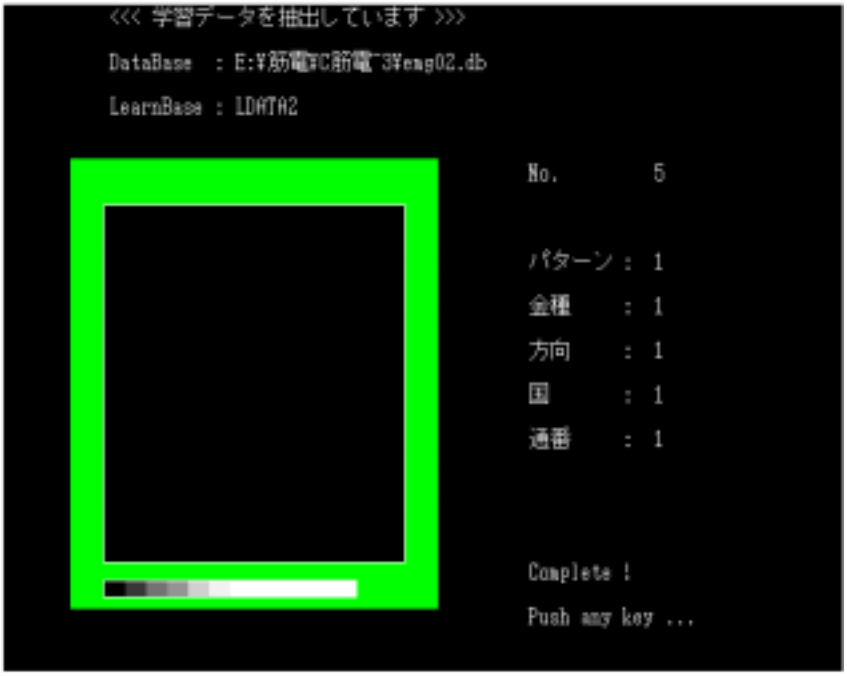


図 2.36　学習データ抽出画面

２．７．６ スラブ値ファイル作成

学習データファイルのスラブ値を取り出して、スラブファイルを作成する。スラブファイル（SLAB1~50）の作成中の画面を図 2.37 に示す。

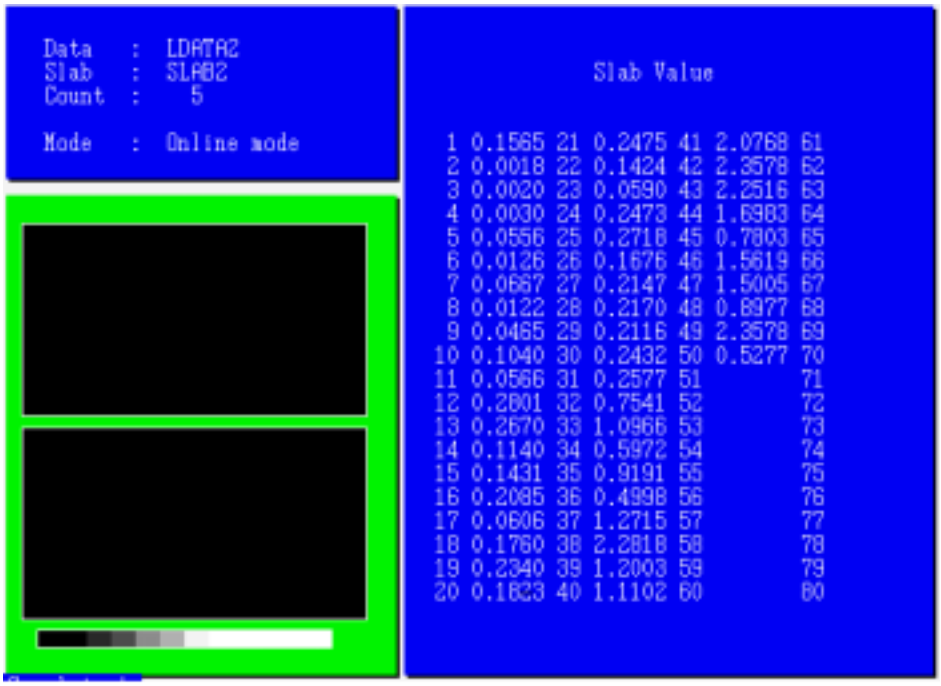


図 2.37　スラブファイル作成画面

２．７．７ スラブファイルの並び替え

学習収束を早めるために、各パターンのスラブファイル（SLAB1～50）を並び替え、学習ファイル(LEARN.S)を作成する。学習ファイルと各スラブファイルの関係は図 2.38 に示す。図中のｎは 1 パターンの学習枚数である。

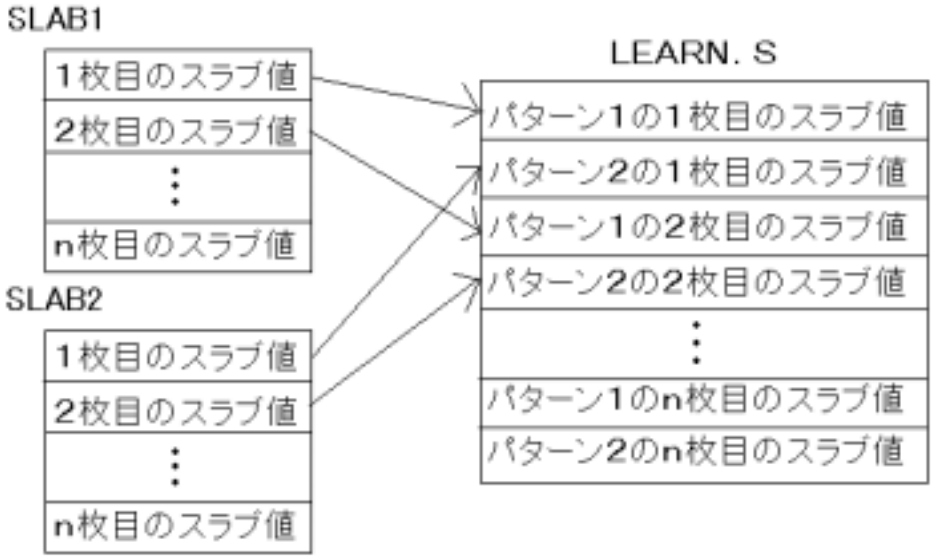


図 2.38　学習ファイルと各スラブファイルの関係

２．７．８ 教師ファイル作成

環境設定ファイルで指定されたパターン数に従って教師ファイルを作成する。正解ならば 1、不正解ならば 0 を教師する行列を作成する。

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.16}$$

作成された行列はパターン数をｎとし、n×n 行列である。本研究の場合、手首が上の状態と下の状態の 2 パターンであるため、２×２行列が作成される。

２．７．９ 学習

ニューロ学習を実行し、ウェイトファイルを作成する。学習プログラムは環境設定ファイルの最終誤差判定値または学習回数のどちらかの条件を満たした時に終了する。図 2.39 中の左下は相互誤差を表し、右半分はそれぞれのパターンの出力ユニット値を表す。そのため、本研究では腕の上下の 2 パターンを学習するので、2 つの出力が表示される。

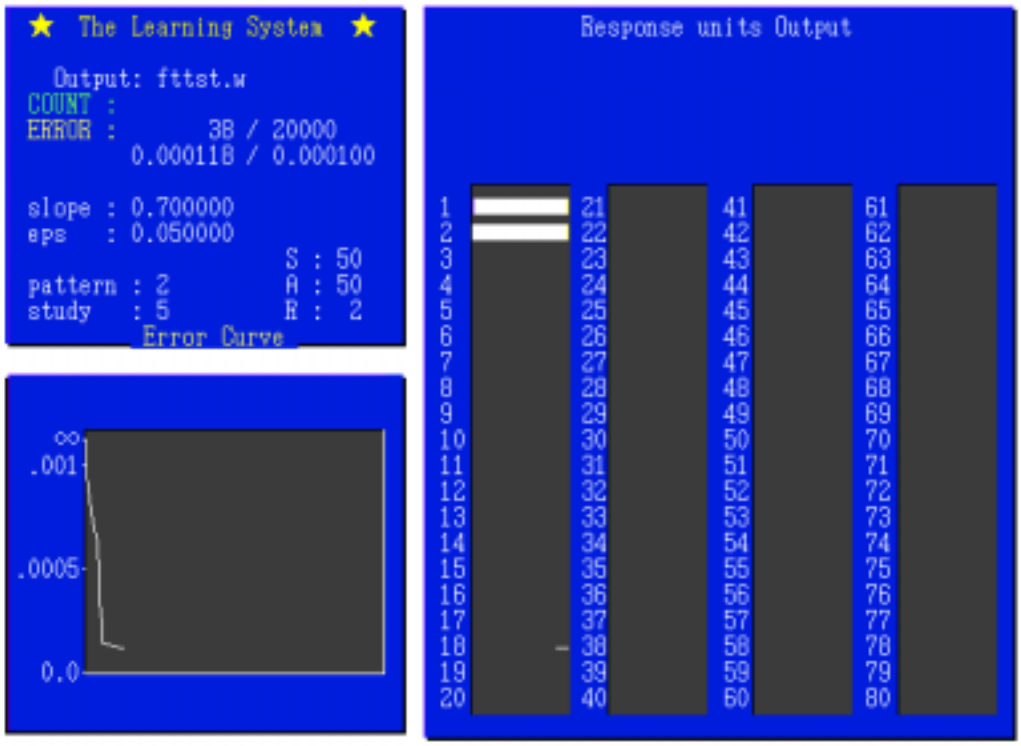


図 2.39　学習画面

２．７．１０ 認識・評価

ウェイトファイルを用いてデータファイルを評価し、評価結果としてログファイルを出力する。

図 2.40 の評価実行時画面中の右半分は１行目がパターン１の反応値、２行目がパターン２の反応値を表す。パターン 1 は手首を上に上げた時のデータ、パターン 2 は手首を下に下げた時のデータを示す。したがって、図 2.40 の評価中のデータはパターン２に反応していることから、手首を下げたデータであると判明する。

図 2.40　評価実行時画面

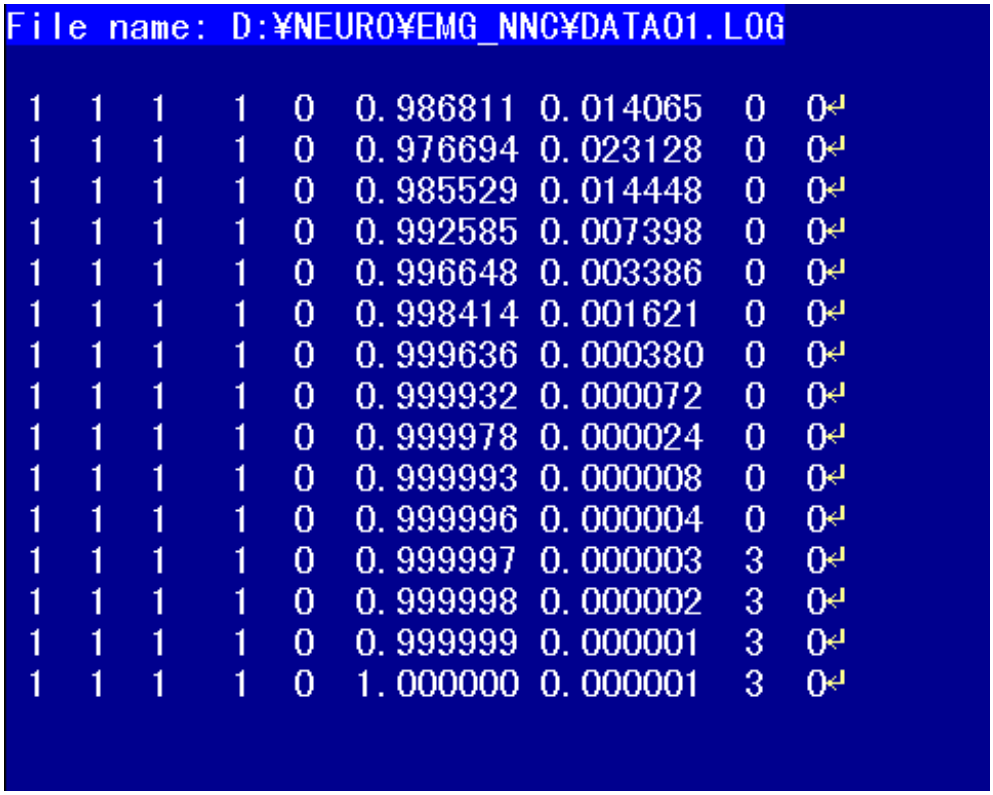


図 2.41　ログファイル

ログファイル中の小数値を出力する 2 列は左列から順にパターン 1 の反応値、パターン 2 の反応値を表す。尚、これらの値の最大値は 1 である。これより、評価データは手首を上にあげたデータであると判定できる。

３．　実験

ここでは著者が行ったシミュレーション実験について述べる。図のように被験者は右腕の脇をしめ、手を軽く握り、腕を 90 度に曲げた状態で手首を上下させ、筋電信号を採取する。

３．１　電極の最適位置

電極の最適ポイントは被験者 12 人の筋活動電位を測定し、筋電信号出力プログラムより出力された各人の Tdata と Fdata から、最適ポイントを推定する。それによると、個人差はみられるものの、数人から腕を垂直に曲げた状態で上腕二頭筋から 8cm のポイントが最も良好な筋活動電位が採取できた。その最適ポイントを C と呼ぶこととする。

さらに、最適ポイントの周囲 1.5ｃｍについても同様にデータ採取を行う。図 3.2 のように手側を A、体側を B と記述する。

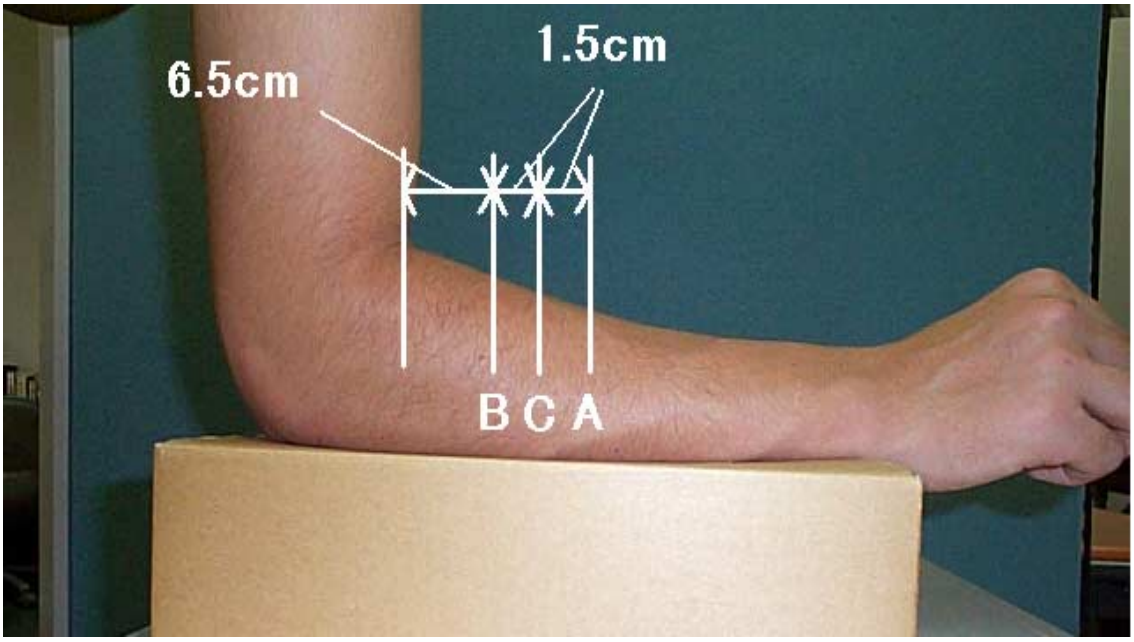


図 3.1　データ採取位置

３．２　データ採取

実験データは 5 人から採取する。

３．２．１　ノイズ対策

まず、データ採取の際して、推測されるノイズを挙げる。
(1) 電極の接触抵抗
(2) 配線での電磁誘導

これらを解決するために著者が行った対策は、
(1) 電解クリームを十分につけ、医療用テープで電極を強く押さえつける。
(2) 電極を束ね、アルミホイルでシールドする。この他に行ったノイズ対策としては、
(3) 机に電磁ノイズが混入している場合が考えられるため、電極の配線をリング状に束ねたものを空中に浮かせ、地面に触れないようにする。
(4) 電極を束ねたリング状の配線はラジオのアンテナに相当するため、様々な方向へ配線を向ける。
(5) リング状の配線に手で直接触れる。

３．２．２　実験方法

本項では提案手法による筋電信号の学習と未学習データによるシミュレーション実験を学習システムにて実施する。その実験方法について述べる

データ採取位置（A，B，C）の 3 箇所において、2.6.2 項で述べた手法 1，2 の実験を行う。さらに、学習データを全て同一人物のデータとし、他人のデータを評価したもの、全て異なる人物を学習データとし、残りの本人のデータを評価したものについての計 12 種類の実験を行う。

被験者 5 人（全員男子大学生、健常者）に対し、同じ動作を 1 人、1 箇所につき 5 個ずつ採取する。したがって、評価個数（データファイル 1）は 5×5=25 個となる。

３．２．３　実験条件

実験条件を表 3.2 に示す。

表 3.2　実験条件

チャンネル数	2
データ数	2048
入力層細胞数	50
中間層数	1
中間層細胞数	50
出力層細胞数	2
学習定数	0.05
慣性定数	0.95
振動定数	−0.1
学習パターン数	2
1パターンの提示個数	3
評価個数	25
最終誤差判定値	0.0001
最大学習回数	20000
学習アルゴリズム	改良型BP法 (2.12)式

３．３　実験結果

＜手法１で学習データ 5 枚が全て同一人物である場合＞

まず、データ採取位置を変化させ、手首を上下させた結果について述べる。また、グラフ中のチャンネル 1，2 をチャンネル 1，2 と表記している。

1）　データ採取位置 A に電極を配置する

手首を上に上げた状態のデータファイルの評価結果を(3.1)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \tag{3.1}$$

学習回数：426 回

表 3.3　チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.986653
チャンネル2の最高反応値	0.11237
チャンネル2の最低反応値	0

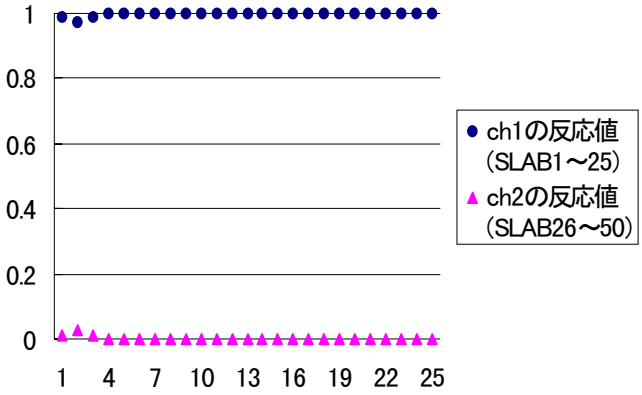


表 3.4　手首を上に上げた時の評価結果

表 3.4 は SLAB 1～25 がチャンネル 1 に反応したグラフと、SLAB 26～50 がチャンネル 2 に反応したグラフを 1 つにしたものである。縦軸が反応値、横軸がスラブ番号を表す。●印は SLAB 1～25 におけるチャンネル 1 の反応値、■印は SLAB 26～50 におけるチャンネル 2 の反応値を表す。

表 3.4 はデータ採取位置 A に電極を配置し、手首を上に上げた状態の出力結果を示す。評価データは手首を上に上げた状態である。そのため、チャンネル 1 が高い反応を示し、チャンネル 2 については低い反応を示しているので、良好な結果であることがわかる。表 3.3 のグラフにおいて、それぞれのチャンネルの反応値は最大が 1 である。しかしながら、ニューラルネットワークは完全に正解であるという答えを出力することはないと考えられる。ここでの反応値 1 とは 0.9999991 以上を示す。

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.2)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.2)$$

表 3.5 チャンネル毎の反応値

チャンネル1の最高反応値	0.025762
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.973796

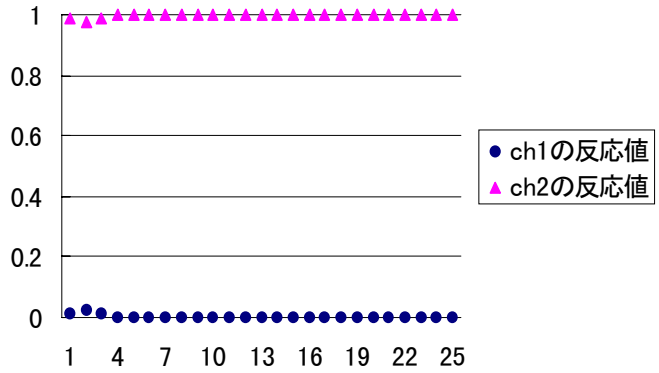


表 3.6 手首を下に下げた時の評価結果

評価データは手首を下に下げた状態である。そのため、チャンネル 1 は低い反応を、チャンネル 2 は高い反応を示していることから、良好な結果であることがわかる。

2） データ採取位置 B に電極を配置する

まず、手首を上に上げた状態のデータファイルの評価結果を(3.3)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.3)$$

学習回数：375 回

表 3.7 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.987885
チャンネル2の最高反応値	0.022073
チャンネル2の最低反応値	0

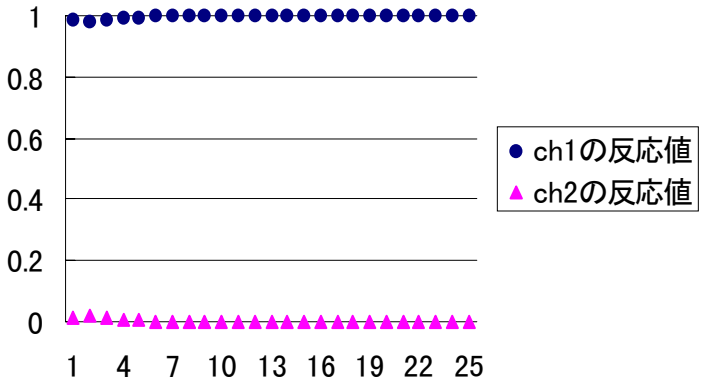


表 3.8 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.4)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \tag{3.4}$$

表 3.9 チャンネル毎の反応値

チャンネル1の最高反応値	0.034312
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.965577

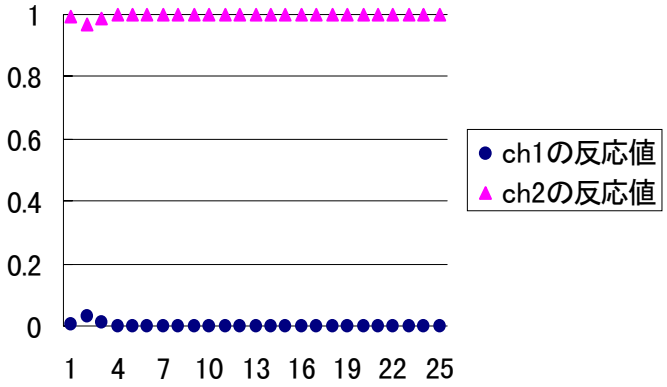


表 3.10 手首を下げた時の評価結果

3） データ採取位置 C に電極を配置する

まず、手首を上に上げた状態のデータファイルの評価結果を(3.5)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \tag{3.5}$$

学習回数：574 回

表 3.11 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.977942
チャンネル2の最高反応値	0.021741
チャンネル2の最低反応値	0

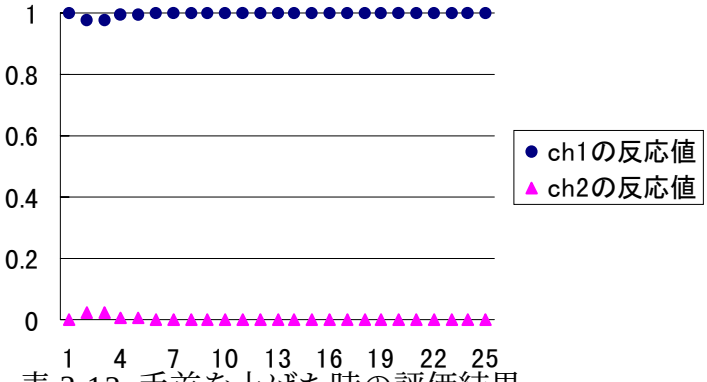


表 3.12 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.6)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.6)$$

表 3.13 チャンネル毎の反応値

チャンネル1の最高反応値	0.027013
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.973669

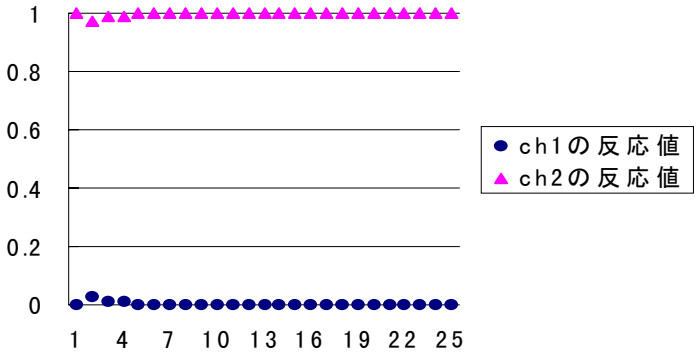


表 3.14 手首を下げた時の評価結果

この実験結果から、データ採取位置の違いにより認識率の変化がみられないことがわかる。

＜手法２で学習データ５枚が全て同一人物である場合＞

データ採取位置を変化させ、手首を上下させた結果について述べる。

４）　データ採取位置 A に電極を配置する

まず、手首を上に上げた状態のデータファイルの評価結果を(3.7)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.7)$$

学習回数：440 回

表 3.15 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.975493
チャンネル2の最高反応値	0.025992
チャンネル2の最低反応値	0

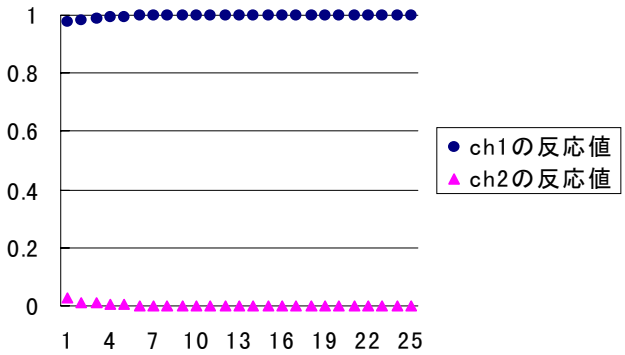


表 3.16 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.8)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \tag{3.8}$$

表 3.17 チャンネル毎の反応値

チャンネル1の最高反応値	0.024113
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.975515

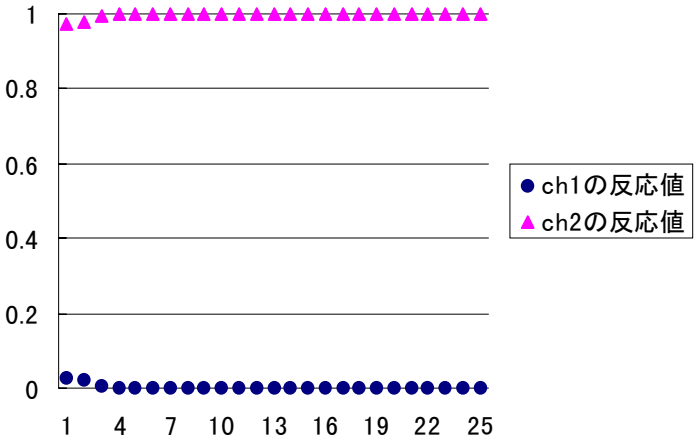


表 3.18 手首を下げた時の評価結果

5） データ採取位置 B に電極を配置する

まず、手首を上に上げた状態のデータファイルの評価結果を(3.9)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \tag{3.9}$$

学習回数：351 回

表 3.19 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.976819
チャンネル2の最高反応値	0.023977
チャンネル2の最低反応値	0

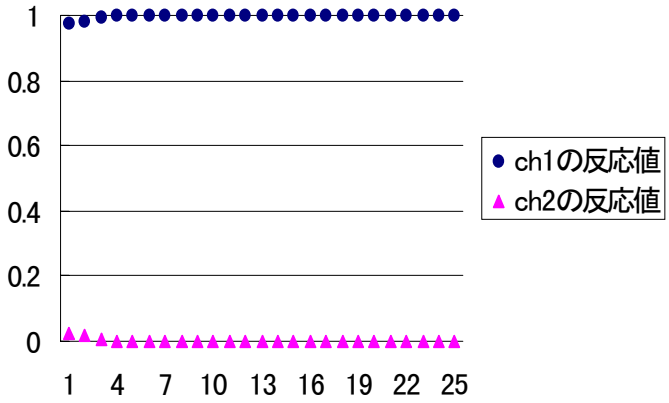


表 3.20 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.10)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.10)$$

表 3.21 チャンネル毎の反応値

チャンネル1の最高反応値	0.027833
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.971153

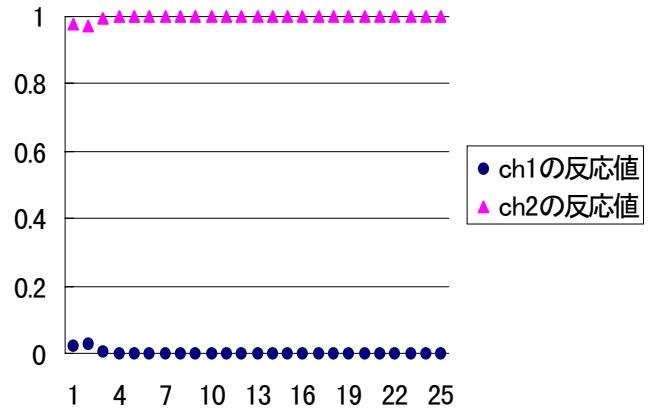


表 3.22 手首を下げた時の評価結果

6） データ採取位置 C に電極を配置する

まず、手首を上に上げた状態のデータファイルの評価結果を(3.11)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.11)$$

学習回数：445 回

表 3.23 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.975313
チャンネル2の最高反応値	0.024371
チャンネル2の最低反応値	0

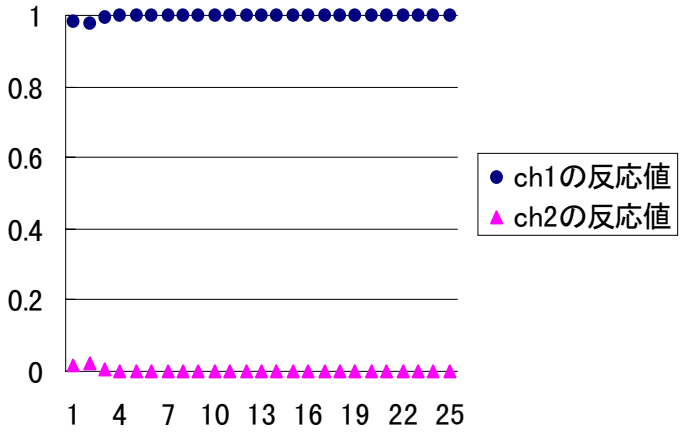


表 3.24 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.12)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.12)$$

表 3.25 チャンネル毎の反応値

チャンネル1の最高反応値	0.025157
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.975077

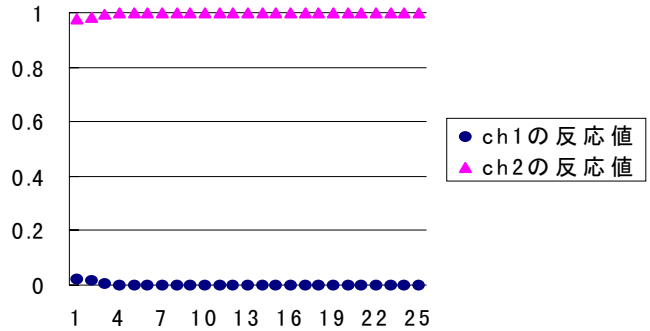


表 3.26 手首を下げた時の評価結果

この結果より、Fdata の抽出の範囲、データ採取位置の違いによって認識率に変化がみられないことがわかる。そして、１人の学習データで他人の動作識別が可能であることがわかる。さらに、その認識率は良好であることがわかる。

＜手法１で学習データ 5 枚が全て異なる人物である場合＞

まず、データ採取位置を変化させ、手首を上下させた結果について述べる。

7） データ採取位置 A に電極を配置する

手首を上に上げた状態のデータファイルの評価結果を(3.13)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.13)$$

学習回数：417 回

表 3.27 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.971661
チャンネル2の最高反応値	0.027629
チャンネル2の最低反応値	0

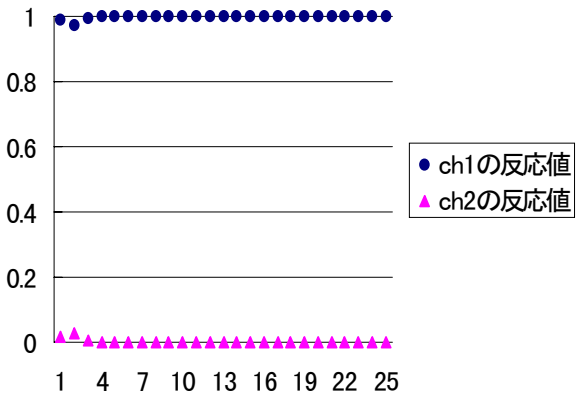


表3.28 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.14)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.14)$$

表 3.29 チャンネル毎の反応値

チャンネル1の最高反応値	0.013192
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.975268

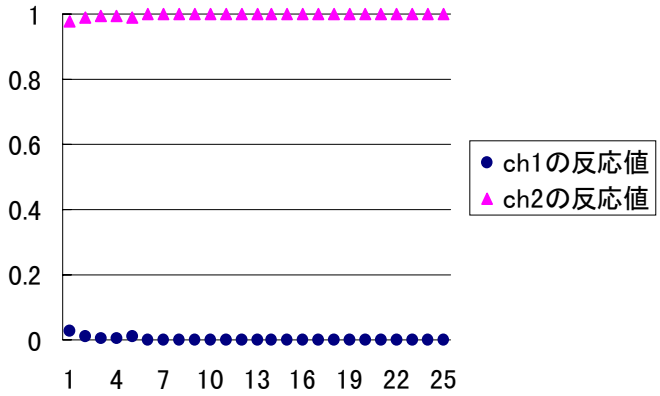


表 3.30 手首を下げた時の評価結果

8） データ採取位置 B に電極を配置する

手首を上に上げた状態のデータファイルの評価結果を(3.15)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.15)$$

学習回数：337 回

表 3.31 チャンネル毎の反応値

チャンネル1の最高反応値	0.999995
チャンネル1の最低反応値	0.970992
チャンネル2の最高反応値	0.028652
チャンネル2の最低反応値	0.000007

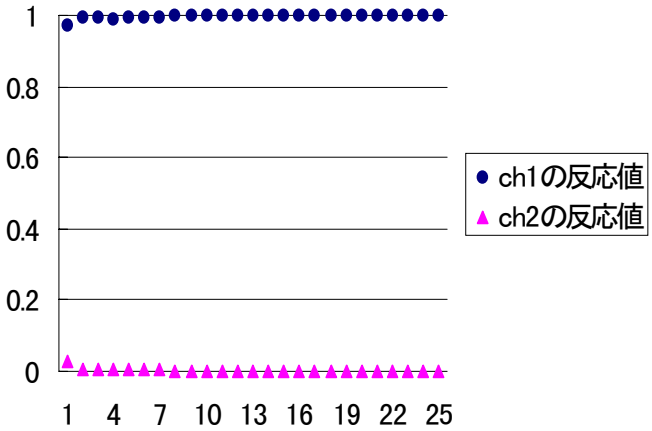


表 3.32 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.16)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.16)$$

表 3.33 チャンネル毎の反応値

チャンネル1の最高反応値	0.024771
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.977011

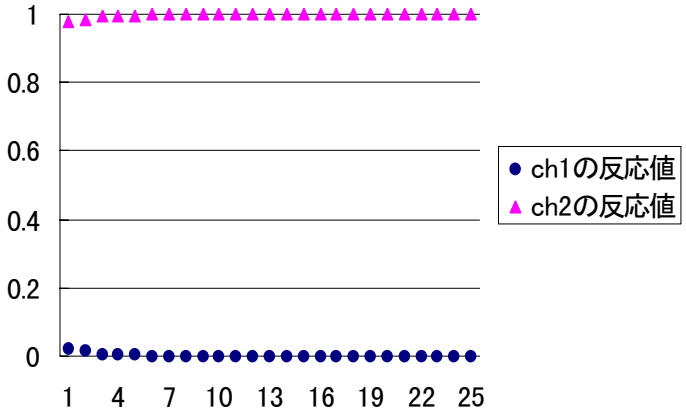


表 3.34 手首を下げた時の評価結果

9） データ採取位置 C に電極を配置する

手首を上に上げた状態のデータファイルの評価結果を(3.17)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.17)$$

学習回数：408 回

表 3.35 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.972868
チャンネル2の最高反応値	0.026415
チャンネル2の最低反応値	0

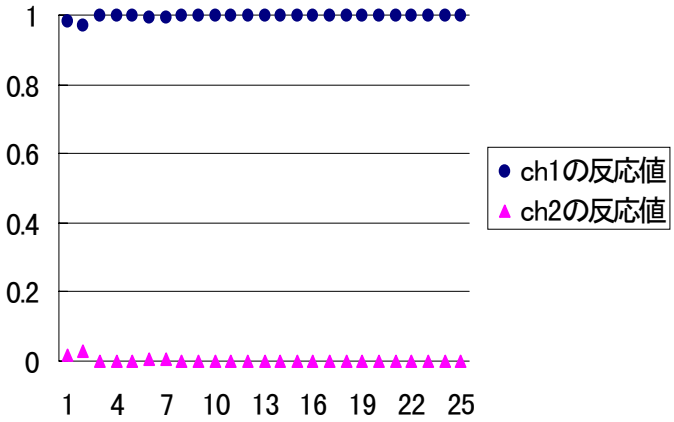


表 3.36 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.18)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.18)$$

表 3.37 チャンネル毎の反応値

チャンネル1の最高反応値	0.022835
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.978141

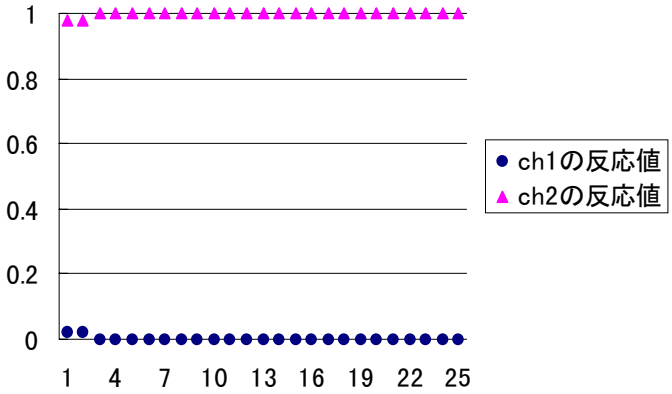


表 3.38 手首を下げた時の評価結果

この結果により、5 人の学習データでの動作識別が可能であることがわかる。さらに、その認識率は良好であることがわかる。

＜手法 2 で学習データ 5 枚が全て異なる人物である場合＞

まず、データ採取位置を変化させ、手首を上下させた結果について述べる。

１０） データ採取位置 A に電極を配置する

手首を上に上げた状態のデータファイルの評価結果を(3.19)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.19)$$

学習回数：445 回

表 3.39 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.974102
チャンネル2の最高反応値	0.027213
チャンネル2の最低反応値	0

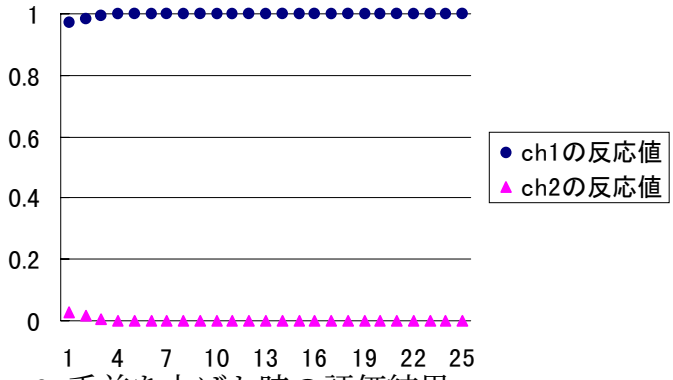


表 3.40 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.20)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.20)$$

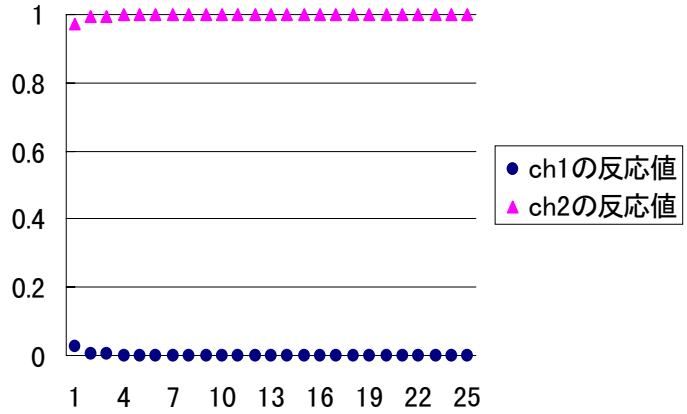


表 3.41 チャンネル毎の反応値

チャンネル1の最高反応値	0.02989
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.969489

表 3.42 手首を下げた時の評価結果

１１） データ採取位置 B に電極を配置する

手首を上に上げた状態のデータファイルの評価結果を(3.21)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.21)$$

学習回数：399 回

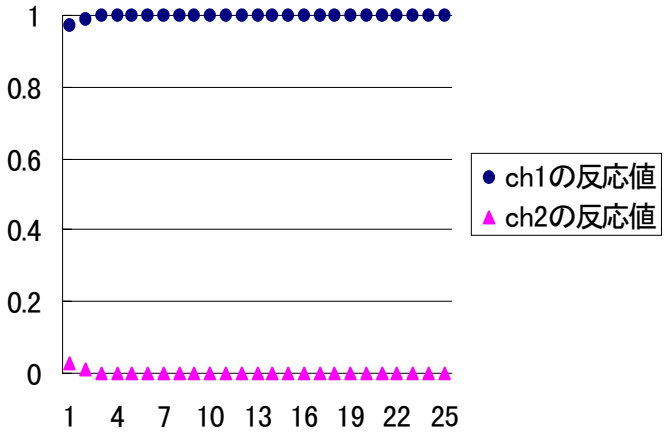


表 3.43 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.970129
チャンネル2の最高反応値	0.02996
チャンネル2の最低反応値	0

表 3.44 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.22)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.22)$$

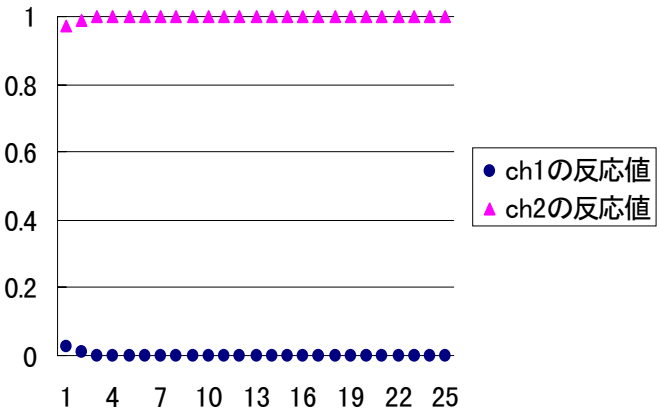


表 3.45 チャンネル毎の反応値

チャンネル1の最高反応値	0.02943
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.971033

表 3.46 手首を下げた時の評価結果

１２） データ採取位置 C に電極を配置する

手首を上に上げた状態のデータファイルの評価結果を(3.23)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.23)$$

学習回数：512 回

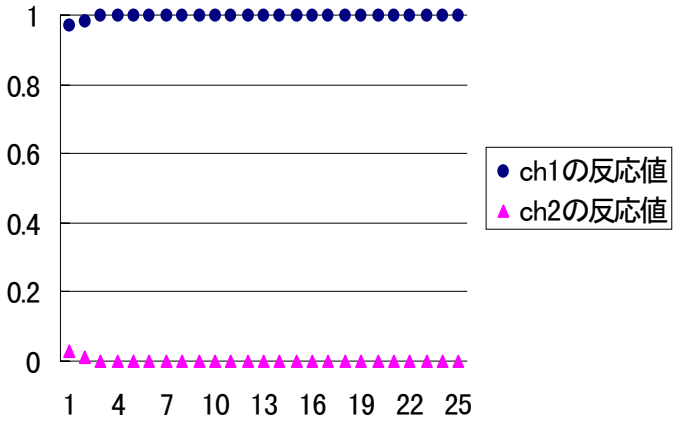


表 3.47 チャンネル毎の反応値

チャンネル1の最高反応値	1
チャンネル1の最低反応値	0.972251
チャンネル2の最高反応値	0.028461
チャンネル2の最低反応値	0

表 3.48 手首を上げた時の評価結果

つぎに、手首を下に下げた状態のデータファイルの評価結果を(3.24)式に示す。

$$\text{認識率} = \frac{\text{正しく反応した個数（25枚）}}{\text{評価個数（25枚）}} \times 100 = 100\% \qquad (3.24)$$

表 3.49 チャンネル毎の反応値

チャンネル1の最高反応値	0.031172
チャンネル1の最低反応値	0
チャンネル2の最高反応値	1
チャンネル2の最低反応値	0.968592

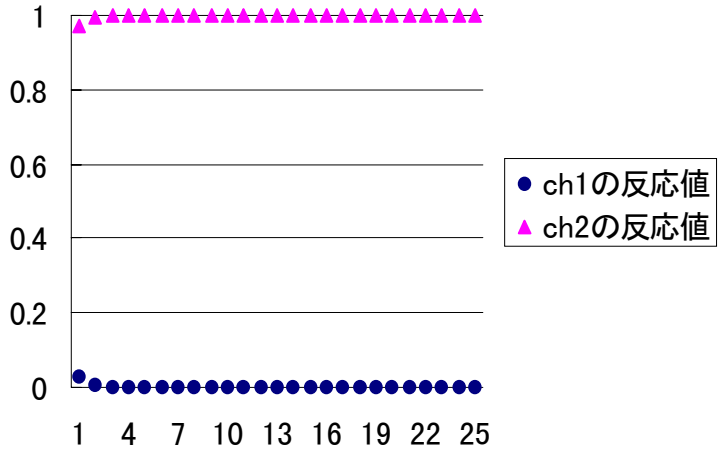


表 3.50 手首を下げた時の評価結果

Fdata の抽出の範囲を狭めたが、認識結果には変化がみられなかった。
実験結果をまとめ、表 3.51、表 3.52 に示す。

表 3.51　手法 1 の実験結果

		学習データ人数	
採取位置	動作	1人	5人
A	上	100%	100%
	下	100%	100%
B	上	100%	100%
	下	100%	100%
C	上	100%	100%
	下	100%	100%

表 3.52　手法 2 の実験結果

		学習データ人数	
採取位置	動作	1人	5人
A	上	100%	100%
	下	100%	100%
B	上	100%	100%
	下	100%	100%
C	上	100%	100%
	下	100%	100%

表 3.4～表 3.52 より採取位置の違いや Fdata の抽出範囲の違いにより、認識率の変化や反応値の大きな変化はみられなかった。また、学習データに 1 人のデータ、5 人のデータを使った場合においても、認識率の変化や反応値の大きな変化はみられなかった。そして、それぞれの動作に対し、その全てに正しい反応が得られ、その全ての反応値は高く、動作識別率も全て 100%であった。

全ての認識率が 100%であるため、どのような場合において認識率が低下するかを観測することができなかった。そのため、反応値で差異を明確にできるようデータ抽出方法を考察することが今後の課題である。

この結果により、自分の動作をニューラルネットワークにより学習し、他人の動作を評価できることが示された。

４．まとめ

本研究は、腕の動作識別において被験者の随意運動に伴う生体信号である筋活動電位を検討した。そして、筋活動電位の登録、評価には階層型ニューラルネットワークを用い、認識システムとしての検証を行った。

実験結果により、手首の動作識別には提案した筋電パターン認識システムが有効であることを確認した。これにより手首での他の動作や手首以外の部位についても同様に動作識別が可能であることが期待できる。性別、年齢の違いによる識別率の調査、電極を乾式にし、波形が大きく観測することができなかった電極の手首へ移行についての検討を今後の課題とする。

５．謝辞

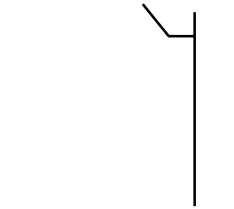
最後に本研究に御協力頂いた竹田史章教授、システム LSI(株)の山本様、グローリー工業(株)西蔭様に厚く御礼申し上げる。

６．参考文献

1) http://www2.freeweb.ne.jp/~tar4/tech/bluetooth.htm
2) http://bme.ahs.kitasato-u.ac.jp:8080/docs/qrs/phy/index.html
3) http://www.kris.sfc.keio.ac.jp/grants/mori/99/doctor/report/3/mori_found6.htm
4) 三田勝己, “ 筋電図計測 ”, BME, Vol.5, No.1, pp.33-40, 1991
5) 内田雅文, 井出英人, 横山修一, “ 筋電によるロボットアーム制御２“ , 電気学会誌, Vol.114-C, No.1, pp.269-270, 1994
6) 栗林勝利, 谷口隆雄, 清水聖治, 小江則禎, “ 前腕の筋活動電位のコマンド識別に関する研究 ”, 日本機会学会誌, pp.25-34, 1995
7) 福田修, 辻敏夫, 金子真, “ ニューラルネットを利用した筋電制御型ポインティングデバイス ”, バイオメカニズム学術講演会, pp.281-284, 1998
8) 佐鹿博信, “ 動作分析と表面筋電図 ”, 総合リハ, Vol.18-5, pp.347-355, 1990
9) 坂巻佳壽美, “ 見てわかるディジタル信号処理 ”, 工業調査会, 1998
10) http://mars.elcom.nitech.ac.jp/java-cai/signal/fft3.html
11) 竹田史章, 大松繁, 井上卓, 尾波宰三, “ フーリエ変換を前処理とするニューラルネットワークによる紙幣識別 ”, システム制御情報学会論文, Vol.5, No.7, pp.265-273, 1992
12) 麻生英樹, “ ニューラルネットワーク情報処理 ”, 産業図書, 1994
13) http://biking.taiiku.tsukuba.ac.jp/thesis/doguu/GR/GR.html
14) http://www.hip.atr.co.jp/patent/details/2557789.html
15) http://www01.u-page.so-net.ne.jp/db3/aseo/emg.htm
16) 竹田史章, 西蔭紀洋, “ 紙幣用ニューロテンプレートマッチング識別手法の開発 ”, 電気学会誌, Vol.121-C, No.1, pp.196-205, 2001
17) 竹田史章、大松繁、井上卓、尾波宰三、小西健一, “ ニューラルネットワークを用いた高速搬送紙幣の識別 ”, 電気学会論文　Vol.112-C, No.4, pp.249-258, 1992
18) http://jiten.nikkeibp.co.jp/cgi-bin/showic/~/showdata.cgi?tmpl=icd00_sh.tmpl&url=/icdic2000/0699_ni.html
19) 竹田史章, 西蔭紀洋, 内田久也, 中原昌樹, “ ニューロテンプレートによるパターンマッチング識別手法の開発とその紙幣への適用 ”, システム制御情報学会論文 Vol.112-C, No.4, pp.415-416, 2000
20) F.Takeda, M.Nakahara, Y.Ichiryu, H.Uchida, “Autonomic Neuro-Recognition Board for Paper Currency”, SPAT2000 Workshop Proceeding, pp.85-90, 2000
21) B.Kermanshahi “ ニューラルネットワークの設計と応用 ”, 昭晃堂, 1999
22) 長尾真, “ パターン情報処理 ”, コロナ社, 1983

付録

txt_myolog1.vbp



・フォーム
　　Myo1.frm
・標準モジュール
　　FFTKModule2.bas
　　Vbadlib.bas

Myo1.frm

```
'変数の定義
Option Explicit

'AD 変換用変数
Private Type ADCPARAM
MyCardType As Integer          ' ｶｰﾄﾞ型式（1:REX5054U, 2:REX5054B）
MyIOBase As Integer
MyIrqNo As Integer
SampTime As Integer            ' ｻﾝﾌﾟﾘﾝｸﾞ間隔
TimeUnitNo As Integer          ' 時間単位
AdcChannel As Integer          ' 変換ﾁｬﾝﾈﾙ数
NCount As Long                 ' 1 ﾁｬﾝﾈﾙ当たりの変換ﾃﾞｰﾀ個数
MaxSampDataNum As Long         ' 全 A/D 変換ﾃﾞｰﾀ個数
Pdata() As Integer             ' 変換ﾃﾞｰﾀ格納先ﾒﾓﾘｰ先頭ｱﾄﾞﾚｽ
End Type
Private Adc As ADCPARAM
Private OleHandle As Long      'MBOX.OCX ﾊﾝﾄﾞﾙ
Private AdCount As Long        'Ad 変換ﾃﾞｰﾀ個数
Private DllMem As Long         ' ﾄﾞﾗｲﾊﾞが FIFO から取り出したﾃﾞｰﾀを格納するｱﾄﾞﾚｽ
Private Sequence(4) As Integer

'グラフ表示用定数および変数
Private Const YDIV_NUM = 10
Private Const XDIV_NUM = 10
Private Xdiv As Double         ' x 座標の目盛(sec/div)
Private Ydiv As Double         ' y 座標の目盛(V/div)
Private XWidth As Single       ' 仮想ｳｨﾝﾄﾞｳにおけるｸﾞﾗﾌ論理座標最大値
Private Yheight As Single      ' 仮想ｳｨﾝﾄﾞｳにおけるｸﾞﾗﾌ論理座標最大値
Private px As Single
Private py As Single
Private ChColor(6) As Long     ' 各ﾁｬﾝﾈﾙ描画色のｶﾗｰ番号
Private CurrentWind As Integer

'データ保管用グローバル変数
Private Ttime(8192) As Double, Ffreq(4096) As Double
Private Tdata(1 To 2, 8192) As Double, Fdata(1 To 2, 4096) As Double
Private Pdata(1 To 2, 4096) As Double
Private Ave(4096) As Double

Private fftScale As Integer

Private Sub Form_Load()
Dim SlotNo As Integer
Dim i As Integer

' 開始ﾎﾞﾀﾝを無効にする
IDB_START.Enabled = False
cmdLoad.Enabled = False

cmdSave.Enabled = False
cmdReDraw.Enabled = False
cmdFFT.Enabled = False

' ｽﾛｯﾄ 0 またはｽﾛｯﾄ 1 に挿入されている自分のｶｰﾄﾞのﾘｿｰｽ情報を取得する
For SlotNo = 0 To 3 Step 1
```

```
If AdGetCardResource(hwnd, SlotNo, Adc.MyCardType, Adc.MyIOBase, Adc.MyIrqNo) = 0 Then
Exit For
End If
Next SlotNo
Debug.Print "Card Check", hwnd, SlotNo, Adc.MyCardType, Adc.MyIOBase, Adc.MyIrqNo

' ｶｰﾄﾞが挿入されていない場合のｴﾗｰ処理
If SlotNo >= 4 Then
IDC_STATUS.Caption = "REX-5054U/B AD ｶｰﾄﾞ検出ｴﾗｰ"
Exit Sub
End If

Select Case Adc.MyCardType
Case REX5054U
IDC_STATUS.Caption = "REX-5054U 正常検出 I/O ﾍﾞｰｽ:0x" + Hex(Adc.MyIOBase) + " IRQ 番号:" + Str(Adc.MyIrqNo)
'IDS_YRANGE.Caption = "0-2.5V"
Case REX5054B
IDC_STATUS.Caption = "REX-5054B 正常検出 I/O ﾍﾞｰｽ:0x" + Hex(Adc.MyIOBase) + " IRQ 番号:" + Str(Adc.MyIrqNo)
'IDS_YRANGE = "±5V"
End Select

'SamplingTime の初期値を 50usec(20kS)に設定
IDE_SAMPTIME.Text = 50

'SamplingTime の単位選択ｺﾝﾎﾞﾘｽﾄ作成
IDCB_TIMEUNIT.AddItem "sec"
IDCB_TIMEUNIT.AddItem "msec"
IDCB_TIMEUNIT.AddItem "usec"
'  ｺﾝﾎﾞﾘｽﾄの初期設定(usec)
IDCB_TIMEUNIT.ListIndex = 2

' AD 変換ﾁｬﾝﾈﾙ数の初期値を 2 に設定
IDE_CHAN.Text = "2"

' AD 変換個数の初期値を 2048 に設定
' (計測時間約 100m 秒)
IDE_SAMPCOUNT.Text = "2048"
CurrentWind = 0

'ﾋﾟｸﾁｬﾎﾞｯｸｽの初期値の取得、設定
PicGraph(0).ForeColor = QBColor(1)      ' 前景色を設定
PicGraph(1).ForeColor = QBColor(1)      ' 前景色を設定

' 各チャンネルグラフ描画色設定
ChColor(0) = 15          ' 明るい白
ChColor(1) = 12          ' 明るい赤
ChColor(2) = 14          ' 明るい黄
ChColor(3) = 9           ' 明るい青
ChColor(4) = 10          ' 明るい緑
ChColor(5) = 11          ' 明るいシアン
ChColor(6) = 13          ' 明るいマゼンタ

IDC_STATUS.Caption = "サンプリング間隔などを決定してください"
Debug.Print "Form Module loaded"

End Sub

Private Sub cmdDet_Click()
Dim IXdiv As Integer, IYdiv As String
Dim IFrange As Integer, IFdiv As Double
Dim RetCode As Long

cmdDet.Enabled = False
IDE_SAMPTIME.Enabled = False
IDCB_TIMEUNIT.Enabled = False
IDE_CHAN.Enabled = False
IDE_SAMPCOUNT.Enabled = False
```

```
' AD 変換時間
Adc.SampTime = Val(IDE_SAMPTIME.Text)
' ｻﾝﾌﾟﾘﾝｸﾞ時間
Adc.TimeUnitNo = Val(IDCB_TIMEUNIT.ListIndex)
' AD 変換ﾁｬﾝﾈﾙ数
Adc.AdcChannel = Val(IDE_CHAN.Text)
' AD 変換個数
Adc.NCount = Val(IDE_SAMPCOUNT.Text)

'ｸﾞﾗﾌの目盛(Xdiv,Ydiv)を設定
Xdiv = (Adc.NCount * Adc.SampTime / XDIV_NUM / 1.024) *
10# ^ (-3# * Adc.TimeUnitNo)
Select Case Adc.MyCardType
Case REX5054B    ' ±5Volt
Ydiv = 10# / YDIV_NUM
Case REX5054U    ' 0-2.5Volt
Ydiv = 2.5 / YDIV_NUM
End Select

' ｽｹｰﾙの単位表示
If Xdiv < 10# ^ -3# Then
IXdiv = (Xdiv * 10# ^ 6#)
IDS_XDIV.Caption = Str(IXdiv) + "μ秒/div"
ElseIf Xdiv < 10# ^ 0# Then
IXdiv = (Xdiv * 10# ^ 3#)
IDS_XDIV.Caption = Str(IXdiv) + "m 秒/div"
Else
IXdiv = Xdiv
IDS_XDIV.Caption = Str(IXdiv) + "秒/div"
End If

IYdiv = "   " + Format(Ydiv, "0.00") + "V/div"
IDS_YDIV.Caption = IYdiv

IFrange = 1# / (2# * Adc.SampTime * 10# ^ (-3# *
Adc.TimeUnitNo))
If IFrange > 1000# Then
IFrange = IFrange / 1000
IDS_FRANGE.Caption = "   0～" + Str(IFrange) + "kHz"
ElseIf IFrange > 1# Then
IDS_FRANGE.Caption = "   0～" + Str(IFrange) + "Hz"
End If

IFdiv = (1# / (Adc.NCount * Adc.SampTime * 10# ^ (-3# *
Adc.TimeUnitNo)))
IDS_FDIV.Caption = "       " + Format(IFdiv, "0.000") + "Hz"

TFinp

'=======================================
RetCode = AdSetParam(hwnd, Adc.MyCardType,
INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo)
Debug.Print "SetParam", hwnd, Adc.MyCardType,
INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo, RetCode

If RetCode <> 0 Then
IDC_STATUS.Caption = "AD ｶｰﾄﾞﾊﾟﾗﾒｰﾀ設定ｴﾗｰ[CODE:" +
Str(RetCode) + "]"
Exit Sub
End If

'=======================================
Sequence(1) = 0: Sequence(2) = 1: Sequence(3) = 2: Sequence(4)
= 3

' 変換ﾁｬﾝﾈﾙ数設定
RetCode = AdSetChannel(Adc.AdcChannel, Sequence(1))
Debug.Print "ADSetChannel", Adc.AdcChannel, Sequence(1),
RetCode

If RetCode <> Adc.AdcChannel Then
IDC_STATUS.Caption = "AD 変換ﾁｬﾝﾈﾙ設定誤り[CODE:" +
```

```
Str(RetCode) + "]"
Exit Sub
End If

'================================================
' 通常のﾓｰﾄﾞで開始終了
AdSetTrigger TRIGGER_DISABLE, PULSE_RISING

'================================================
' ｻﾝﾌﾟﾘﾝｸﾞ間隔設定
RetCode = AdSetFreq(hwnd, Adc.SampTime, Adc.TimeUnitNo)
Debug.Print "ADSetFreq", hwnd, Adc.SampTime,
Adc.TimeUnitNo, RetCode

If RetCode <> 0 Then
IDC_STATUS.Caption = "ｻﾝﾌﾟﾘﾝｸﾞ間隔設定誤り[CODE:" +
Str(RetCode) + "]"
Exit Sub
End If

'========================================
' VB ｱﾌﾟﾘｹｰｼｮﾝ側でﾃﾞｰﾀを保管するための配列を確保
Adc.MaxSampDataNum = Adc.AdcChannel * Adc.NCount
ReDim Adc.Pdata(Adc.MaxSampDataNum) As Integer

' AD ﾃﾞｰﾀ格納ﾒﾓﾘｰをｱﾛｹｰﾄ(1 ﾁｬﾝﾈﾙ当たりの変換個数を指定する)
DllMem = AdAllocDataMem(hwnd, Adc.NCount)
If DllMem = 0 Then
IDC_STATUS.Caption = "変換ﾃﾞｰﾀ格納用ﾒﾓﾘﾌﾞﾛｯｸｱﾛｹｰｼｮﾝｴﾗｰ"
Exit Sub
End If

'================================================
'OLE のウィンドウハンドル取得
OleHandle = MBOX1.GetMboxWnd()
If (OleHandle = 0) Then
MsgBox "OLE のハンドルが取得できません。", vbOKOnly +
vbCritical, "エラー"
Exit Sub
End If

'Debug.Print "OleHandle", OleHandle
'================================================

IDB_START.Enabled = True
cmdLoad.Enabled = True

cmdSave.Enabled = False
cmdReDraw.Enabled = False
cmdFFT.Enabled = False

IDC_STATUS.Caption = ""

End Sub

Private Sub IDB_START_Click()
'ﾎﾟｰﾘﾝｸﾞﾓｰﾄﾞ A/D 変換実行
AdStartPolModeThread OleHandle
End Sub

Private Sub IDQUIT_Click()
'AD ﾃﾞｰﾀ格納ﾒﾓﾘｰ開放
AdFreeDataMem
Unload Me
End Sub

Private Sub MBOX1_OnMsgPost(ByVal wParam As Integer,
ByVal lParam As Long)
Dim AdcStat As Integer          ' AD 変換ｽﾃｰﾀｽ
Dim ChNo As Integer             ' CH 番号変数
Dim DataNo  As Long
Dim i As Long

' lParam -> AD 変換ﾃﾞｰﾀｶｳﾝﾄ数
```

```
AdCount = lParam

' wParam -> AD 変換開始終了ｺｰﾄﾞ
AdcStat = wParam

Debug.Print "StartEndCode", AdcStat

Select Case AdcStat
Case AD_MODE_RUN
IDC_STATUS.Caption = "変換開始..."
Case AD_MODE_TRGRUN
IDC_STATUS.Caption = "ﾄﾘｶﾞｰ変換開始"
Case AD_MODE_STOP
IDC_STATUS.Caption = "指定個数終了"

' ﾄﾞﾗｲﾊﾞがｾｯﾄした変換ﾃﾞｰﾀを自分のﾒﾓﾘへ転送保管します
' 1 個の変換ﾃﾞｰﾀは 2 ﾊﾞｲﾄですので転送先のｱﾄﾞﾚｽに注意して下さい
VbMemCopy Adc.Pdata(0), DllMem, AdCount * 2

' 各 CH のﾃﾞｰﾀを電圧値に変換する
For ChNo = 1 To Adc.AdcChannel
For i = 0 To Adc.NCount - 1
DataNo = i * Adc.AdcChannel + ChNo - 1
'Tdata(ChNo, I) = (CLng(Adc.pData(DataNo) And &H1FFF) * 8)
/ CLng(32768) * CLng(10) - CLng(5)
Tdata(ChNo, i) = CLng(Adc.Pdata(DataNo) And &H1FFF) /
409.6 - 5#
Next i
Next ChNo

PlotTData                ' グラフ描画

cmdFFT.Enabled = True

Case AD_MODE_PRESTOP:
IDC_STATUS.Caption = "ﾄﾘｶﾞｰ終了"
Case AD_MODE_OVRUN:
IDC_STATUS.Caption = "FIFO ｵｰﾊﾞｰﾗﾝ停止"
End Select

End Sub

Private Sub cmdFFT_Click()
Dim i As Integer

For i = 1 To Adc.AdcChannel
F_FFT Tdata(), Fdata(), Pdata(), i, CInt(Adc.NCount), 0
Next i

PlotFData
cmdSave.Enabled = True
cmdReDraw.Enabled = True

End Sub

Private Sub cmdSave_Click()
Dim i As Integer, J As Integer
Dim Fnum As Integer
Dim Filename As String * 256
Dim Pos As String * 1
Dim TestNum As String * 3
Dim MType As String * 1
Dim A As String * 10, B As String * 10, D As String * 10, E As
String * 10, F As String * 10, G As String * 10, H As String * 10

If (ManName.Text = "" Or PosInitial.Text = "") Then
IDC_STATUS.Caption = "ファイルが生成できません"
Exit Sub
End If
If (TestNo.Text = "" Or MoveType.Text = "") Then
IDC_STATUS.Caption = "ファイルが生成できません"
Exit Sub
End If
```

```
Pos = PosInitial.Text
TestNum = TestNo.Text
MType = MoveType.Text
Filename = ManName.Text + Pos + TestNum + MType + ".txt"
Filename = FilePath.Text + "¥" + Filename
Debug.Print "save file = " + Filename

' ﾌｧｲﾙを開く
Fnum = FreeFile()
Open Filename For Binary As Fnum

'for ループ　2048 回
For i = 0 To Adc.NCount - 1

't_No の記述
A = Str(Ttime(i))
Put #Fnum, , A
Put #Fnum, , "    "

'for ループ 2 回　tch1,tch2 の記述
For J = 1 To Adc.AdcChannel
B = Str(Tdata(J, i))
Put #Fnum, , B
Put #Fnum, , "    "
Next J

'i が 1024 回以下なら fdata も記述
If i < Adc.NCount / 2 Then

'f_No の記述
D = Str(Ffreq(i))
Put #Fnum, , D
Put #Fnum, , "    "

'for ループ　1 回　fch1 の記述
For J = 1 To Adc.AdcChannel - 1
E = Str(Fdata(J, i + 1))
Put #Fnum, , E
Put #Fnum, , "    "
Next J

'fch2 の記述
F = Str(Fdata(Adc.AdcChannel, i + 1))
Put #Fnum, , F
Put #Fnum, , "    "
Put #Fnum, , vbCrLf

'i が 1025 以上の時の処理(f_No,fch1,fch2 を全て"0"にする)
Else

'for ループ　2 回　f_No,fch1 に"0"を記述
For J = 0 To Adc.AdcChannel - 1
G = Str(0)
Put #Fnum, , G
Put #Fnum, , "    "
Next J
'+1 回 fch2 (計 2+1 回)"0"を記述
H = Str(0)
Put #Fnum, , H
Put #Fnum, , vbCrLf
End If

Next i

Close #Fnum
IDC_STATUS.Caption = "saved " + Filename

End Sub

Private Sub cmdload_Click()
Dim i As Integer, J As Integer
Dim Fnum As Integer, dammy As String
Dim Filename As String * 256
```

```
Dim Pos As String * 1
Dim TestNum As String * 3
Dim MType As String * 1

If (ManName.Text = "" Or PosInitial.Text = "") Then
IDC_STATUS.Caption = "ファイルが生成できません"
Exit Sub
End If
If (TestNo.Text = "" Or MoveType.Text = "") Then
IDC_STATUS.Caption = "ファイルが生成できません"
Exit Sub
End If

Pos = PosInitial.Text
TestNum = TestNo.Text
MType = MoveType.Text
Filename = ManName.Text + Pos + TestNum + MType + ".txt"
Filename = FilePath.Text + "¥" + Filename
Debug.Print "load file = " + Filename

' ﾌｧｲﾙを開く
Fnum = FreeFile()
Open Filename For Input As Fnum

For i = 0 To Adc.NCount - 1
Input #Fnum, dammy
Ttime(i) = CDbl(dammy)
For J = 1 To Adc.AdcChannel
Input #Fnum, dammy
Tdata(J, i) = CDbl(dammy)
Next J
If i < Adc.NCount / 2 Then
Input #Fnum, dammy
Ffreq(i) = CDbl(dammy)
For J = 1 To Adc.AdcChannel
Input #Fnum, dammy
Fdata(J, i + 1) = CDbl(dammy)
Next J
Else
For J = 0 To Adc.AdcChannel
Input #Fnum, dammy
Next J
End If
Next i

Close #Fnum

cmdReDraw_Click
IDC_STATUS.Caption = "loaded " + Filename

cmdSave.Enabled = True
cmdReDraw.Enabled = True
cmdFFT.Enabled = True

End Sub

Public Sub PlotTData()          '時間データのグラフ描画
Dim i As Long, ChNo As Integer
Dim Y As Double

CurrentWind = 0
DrawTScale                  'スケールの描画

PicGraph(CurrentWind).DrawStyle = 0

Debug.Print "adc.adcchannel = "; Adc.AdcChannel
For ChNo = 1 To Adc.AdcChannel
For i = 0 To Adc.NCount - 1 Step 1
Y = Tdata(ChNo, i) * py / Ydiv
If i = 0 Then
PicGraph(CurrentWind).PSet (i, Y), QBColor(ChColor(ChNo))
Else
PicGraph(CurrentWind).Line -(i, Y), QBColor(ChColor(ChNo))
End If
```

```
Next i
Next ChNo

End Sub

Public Sub DrawTScale()         '時間データのスケール描画
Dim i As Single
Dim SS As Integer

SS = 40 'グラフの余白
Yheight = 1000
XWidth = Adc.NCount
PicGraph(CurrentWind).Cls
PicGraph(CurrentWind).Scale (0 - SS, Yheight / 2 + SS)-(XWidth
+ SS, -Yheight / 2 - SS)
py = Yheight / YDIV_NUM
px = (Adc.NCount / 1.024) / XDIV_NUM

PicGraph(CurrentWind).DrawStyle = vbDot

' ｘ軸方向の点線
For i = -YDIV_NUM / 2 To YDIV_NUM / 2 Step 1
PicGraph(CurrentWind).Line (0, i * py)-(XWidth, i * py),
QBColor(8)
Next

' ｙ軸方向の点線
For i = 0 To XDIV_NUM
PicGraph(CurrentWind).Line (i * px, Yheight / 2)-(i * px, -
Yheight / 2), QBColor(8)
Next
PicGraph(CurrentWind).Line (XWidth, Yheight / 2)-(XWidth, -
Yheight / 2), QBColor(8)

End Sub

Public Sub PlotFData()          'FFT データのグラフ描画
Dim i As Long, ChNo As Integer
Dim Y As Double, Fdiv As Double
Dim max(1) As Double, min(1) As Double
Dim Ysig(2) As Double

Ysig(1) = 1#
Ysig(2) = -1#

CurrentWind = 1
DrawFScale                  'スケールの描画
fftScale = 1

PicGraph(CurrentWind).DrawStyle = 0

For ChNo = 1 To 2
'F_MaxMin Fdata(), ChNo, CInt(Adc.NCount), max(), min()
'Fdiv = max(0)
Fdiv = 1000
'Debug.Print "ch"; ChNo; "    FFT Max : "; max(0)
For i = 1 To Adc.NCount Step 1
Y = Fdata(ChNo, i) * Yheight / Fdiv
PicGraph(CurrentWind).Line ((i - 1) * 2 * fftScale, 0)-((i - 1) * 2 *
fftScale, Y * Ysig(ChNo)), QBColor(ChColor(ChNo))
Next i
Next ChNo

End Sub

Public Sub DrawFScale()         'FFT データのスケール描画
Dim i As Single
Dim SS As Integer

SS = 40 'グラフの余白
Yheight = 1000
XWidth = Adc.NCount
PicGraph(CurrentWind).Cls
PicGraph(CurrentWind).Scale (0 - SS, Yheight + SS)-(XWidth +
```

```
SS, -Yheight - SS)
py = Yheight / YDIV_NUM * 2
px = Adc.NCount / XDIV_NUM

PicGraph(CurrentWind).DrawStyle = vbDot

' ｘ軸方向の点線
For i = -YDIV_NUM / 2 To YDIV_NUM / 2 Step 1
PicGraph(CurrentWind).Line (0, i * py)-(XWidth, i * py),
QBColor(8)
Next

' ｙ軸方向の点線
For i = 0 To XDIV_NUM
PicGraph(CurrentWind).Line (i * px, -Yheight)-(i * px, Yheight),
QBColor(8)
Next

End Sub

Public Sub PlotPData()        '位相データのグラフ描画
Dim i As Long, ChNo As Integer
Dim Y As Double, Pdiv As Double
Dim max(1) As Double, min(1) As Double
Dim Yoffset(2) As Double, Ysig(2) As Double
Dim CurrentDNum As Integer

Yoffset(1) = 0#
Yoffset(2) = 0#

Ysig(1) = 0
Ysig(2) = 1

'ChNo = CurrentFFT
CurrentWind = 1
DrawPScale                'スケールの描画
PicGraph(CurrentWind).DrawStyle = 0
Pdiv = 2000# / (8# * Atn(1#))

For ChNo = 1 To 2
'Debug.Print "ch"; ChNo; "    FFT Max : "; max(0)
For i = 0 To Adc.NCount Step 1
Y = Pdata(ChNo, i) * Pdiv
PicGraph(CurrentWind).Line ((i * 2 + Ysig(ChNo)) * fftScale,
Yoffset(ChNo))-((i * 2 + Ysig(ChNo)) * fftScale, Y +
Yoffset(ChNo)), QBColor(ChColor(ChNo))
Next i
Next ChNo

'IDS_WLABEL(CurrentWind).Caption = " FFTdata / Phase" +
Format(CurrentDNum, "00") + " / ALL"

End Sub

Public Sub DrawPScale()        '位相データのスケール描画
Dim i As Single
Dim SS As Integer

SS = 40 'グラフの余白
Yheight = 1000
XWidth = Adc.NCount
PicGraph(CurrentWind).Cls
PicGraph(CurrentWind).Scale (0 - SS, Yheight + SS)-(XWidth +
SS, -Yheight - SS)
py = Yheight / YDIV_NUM * 2
px = Adc.NCount / XDIV_NUM

PicGraph(CurrentWind).DrawStyle = vbDot

' ｘ軸方向の点線
For i = -YDIV_NUM / 2 To YDIV_NUM / 2 Step 1
PicGraph(CurrentWind).Line (0, i * py)-(XWidth, i * py),
QBColor(8)
```

```
Next

' y軸方向の点線
For i = 0 To XDIV_NUM
PicGraph(CurrentWind).Line (i * px, -Yheight)-(i * px, Yheight),
QBColor(8)
Next

End Sub

Private Sub TFinp()
Dim i As Integer, J As Integer
Dim TInterval As Double, FInterval As Double
TInterval = Adc.SampTime * 10# ^ (-3# * Adc.TimeUnitNo)
FInterval = (1# / (Adc.NCount * TInterval))

For i = 1 To Adc.NCount
Ttime(i) = i * TInterval
Next i

For i = 1 To Adc.NCount / 2
Ffreq(i) = i * FInterval
Next i

End Sub

Private Sub cmdReDraw_Click()
PlotTData
PlotFData
End Sub

Private Sub Wave1_Click()
Dim i As Integer, dt1 As Double, dt2 As Double, dt3 As Double
Dim pi As Double
pi = 4 * Atn(1)
dt1 = 2 * pi * 4 / Adc.NCount
dt2 = 2 * pi * 56 / Adc.NCount
dt3 = 2 * pi * 100 / Adc.NCount

For i = 0 To Adc.NCount
Tdata(1, i) = Sin(dt1 * (i - 1)) + 0.3 * Sin(dt2 * (i - 1) + pi / 4) + 0.7
* Sin(dt3 * (i - 1))
Tdata(2, i) = 0
Next i

PlotTData

End Sub
```

FFTKModule2.bas

```
Public Sub F_FFT(Tdata() As Double, Fdata() As Double, Pdata()
As Double, _
CurrentNo As Integer, N As Integer, wind As Integer)
'wind = 0 window なし
'wind = 1 Hamming window
'wind = 2 Hanning window
'
Dim x(1, 8192) As Double 'x()は、FFT 変換用変数

F_Window Tdata(), x(), CurrentNo, N, wind
fftCalc x(), CurrentNo, N, 1#
fftAbs Fdata(), Pdata(), x(), CurrentNo, N / 2

End Sub

Private Sub fftCalc(x() As Double, CurrentNo As Integer, N As
```

```
Integer, id As Double)
' N     データ数、2 のべき乗であること  128,256,512,1024,...
' id= 1.0 FFT の計算
' id=-1.0 逆 FFT(RFT)の計算
' 入力データ：x(0,N) 実数値データ:x(1,N) 虚数値データ x(2,N)
' 出力  x(1,N)と x(2,N)に結果が入る  (x(0,N)の内容は保持される)
Dim i As Integer, i0 As Integer, i1 As Integer, J As Integer
Dim ns As Integer, k As Integer, arg As Integer
Dim s As Double, c As Double, sc As Double, X1 As Double, y1 As Double

ns = N / 2: sc = 2 * 4 * Atn(1#) / N
For i = 0 To N: x(1, i) = 0: Next i
Do While ns >= 1
arg = 0
For J = 1 To N Step 2 * ns
k = N / 4
c = Cos(sc * arg): s = Sin(id * sc * arg)
For i0 = J To J + ns - 1
i1 = i0 + ns
X1 = x(0, i1) * c - x(1, i1) * s: y1 = x(1, i1) * c + x(0, i1) * s
x(0, i1) = x(0, i0) - X1: x(1, i1) = x(1, i0) - y1
x(0, i0) = x(0, i0) + X1: x(1, i0) = x(1, i0) + y1
Next i0
Do While k <= arg
arg = arg - k: k = k / 2
If k = 0 Then Exit Do
Loop
arg = arg + k
Next J
ns = ns / 2
Loop
If id < 0 Then
For i = 1 To N
x(0, i) = x(0, i) / N: x(1, i) = x(1, i) / N
Next i
End If
J = 1
For i = 1 To N - 1
If i <= J Then
X1 = x(0, i): x(0, i) = x(0, J): x(0, J) = X1
y1 = x(1, i): x(1, i) = x(1, J): x(1, J) = y1
End If
k = N / 2
Do While k < J
J = J - k: k = k / 2
Loop
J = J + k
Next i
End Sub

Private Sub F_Window(Tdata() As Double, x() As Double,
CurrentNo As Integer, _
N As Integer, win As Integer)
Dim i As Integer

Select Case win
Case 1
fftHamming Tdata(), x(), CurrentNo, N
Case 2
fftHanning Tdata(), x(), CurrentNo, N
Case Else
For i = 1 To N
x(0, i) = Tdata(CurrentNo, i)
x(1, i) = 0
Next i
```

```
End Select

End Sub

Private Sub fftHanning(Tdata() As Double, x() As Double,
CurrentNo As Integer, N As Integer)
Dim i As Integer, sc As Double
sc = 2 * 4 * Atn(1#) / N
For i = 1 To N
x(0, i) = 0.5 * Tdata(CurrentNo, i) * (1# - Cos(sc * (i - 1)))
x(1, i) = 0
Next i
End Sub

Private Sub fftHamming(Tdata() As Double, x() As Double,
CurrentNo As Integer, N As Integer)
Dim i As Integer, sc As Double
sc = 2 * 4 * Atn(1#) / N
For i = 1 To N
x(0, i) = Tdata(CurrentNo, i) * (0.54 - 0.46 * Cos(sc * (i -
1)))
x(1, i) = 0
Next i
End Sub

Private Sub fftAbs(Fdata() As Double, Pdata() As Double, x()
As Double, CurrentNo As Integer, N As Integer)
' z()=sqrt(x()^2+y()^2)
Dim i As Integer
For i = 1 To N
Fdata(CurrentNo, i) = Sqr(x(0, i) ^ 2 + x(1, i) ^ 2)
'Pdata(CurrentNo, i) = Atn(x(1, i) / x(0, i))
Next i
End Sub
```

```
Public Sub F_MaxMin(x() As Double, CurrentNo As Integer, N As
Integer, max() As Double, min() As Double)
Dim i As Integer
max(0) = x(CurrentNo, 1): min(0) = x(CurrentNo, 1)
max(1) = 0: min(1) = 0

'Debug.Print "CurrentNo = "; Str(CurrentNo)
For i = 2 To N
If x(CurrentNo, i) > max(0) Then
max(0) = x(CurrentNo, i)
max(1) = i
End If
If x(CurrentNo, i) < min(0) Then
min(0) = x(CurrentNo, i)
min(1) = i
End If
Next i

End Sub
```

Vbadlib.bas

```
CurrentNo As Integer, N As Integer, wind As Integer)
'wind = 0 window なし
'wind = 1 Hamming window
'wind = 2 Hanning window
'
Dim x(1, 8192) As Double 'x()は、FFT 変換用変数
```

```
F_Window Tdata(), x(), CurrentNo, N, wind
fftCalc x(), CurrentNo, N, 1#
fftAbs Fdata(), Pdata(), x(), CurrentNo, N / 2

End Sub

Private Sub fftCalc(x() As Double, CurrentNo As Integer, N As
Integer, id As Double)
' N      データ数、2 のべき乗であること　128,256,512,1024,...
' id= 1.0 FFT の計算
' id=-1.0 逆 FFT(RFT)の計算
' 入力データ：x(0,N) 実数値データ:x(1,N) 虚数値データ x(2,N)
' 出力　x(1,N)と x(2,N)に結果が入る　(x(0,N)の内容は保持され
る)
Dim i As Integer, i0 As Integer, i1 As Integer, J As Integer
Dim ns As Integer, k As Integer, arg As Integer
Dim s As Double, c As Double, sc As Double, X1 As Double, y1 As
Double

ns = N / 2: sc = 2 * 4 * Atn(1#) / N
For i = 0 To N: x(1, i) = 0: Next i
Do While ns >= 1
arg = 0
For J = 1 To N Step 2 * ns
k = N / 4
c = Cos(sc * arg): s = Sin(id * sc * arg)
For i0 = J To J + ns - 1
i1 = i0 + ns
X1 = x(0, i1) * c - x(1, i1) * s: y1 = x(1, i1) * c + x(0, i1) * s
x(0, i1) = x(0, i0) - X1: x(1, i1) = x(1, i0) - y1
x(0, i0) = x(0, i0) + X1: x(1, i0) = x(1, i0) + y1
Next i0
Do While k <= arg
arg = arg - k: k = k / 2
If k = 0 Then Exit Do
Loop
arg = arg + k
Next J
ns = ns / 2
Loop
If id < 0 Then
For i = 1 To N
x(0, i) = x(0, i) / N: x(1, i) = x(1, i) / N
Next i
End If
J = 1
For i = 1 To N - 1
If i <= J Then
X1 = x(0, i): x(0, i) = x(0, J): x(0, J) = X1
y1 = x(1, i): x(1, i) = x(1, J): x(1, J) = y1
End If
k = N / 2
Do While k < J
J = J - k: k = k / 2
Loop
J = J + k
Next i
End Sub

Private Sub F_Window(Tdata() As Double, x() As Double,
CurrentNo As Integer, _
N As Integer, win As Integer)
Dim i As Integer

Select Case win
Case 1
fftHamming Tdata(), x(), CurrentNo, N
Case 2
fftHanning Tdata(), x(), CurrentNo, N
Case Else
For i = 1 To N
x(0, i) = Tdata(CurrentNo, i)
x(1, i) = 0
Next i
```

```
End Select

End Sub

Private Sub fftHanning(Tdata() As Double, x() As Double, CurrentNo As Integer, N As Integer)
Dim i As Integer, sc As Double
sc = 2 * 4 * Atn(1#) / N
For i = 1 To N
x(0, i) = 0.5 * Tdata(CurrentNo, i) * (1# - Cos(sc * (i - 1)))
x(1, i) = 0
Next i
End Sub

Private Sub fftHamming(Tdata() As Double, x() As Double, CurrentNo As Integer, N As Integer)
Dim i As Integer, sc As Double
sc = 2 * 4 * Atn(1#) / N
For i = 1 To N
x(0, i) = Tdata(CurrentNo, i) * (0.54 - 0.46 * Cos(sc * (i - 1)))
x(1, i) = 0
Next i
End Sub

Private Sub fftAbs(Fdata() As Double, Pdata() As Double, x() As Double, CurrentNo As Integer, N As Integer)
' z()=sqrt(x()^2+y()^2)
Dim i As Integer
For i = 1 To N
Fdata(CurrentNo, i) = Sqr(x(0, i) ^ 2 + x(1, i) ^ 2)
'Pdata(CurrentNo, i) = Atn(x(1, i) / x(0, i))
Next i
End Sub

Public Sub F_MaxMin(x() As Double, CurrentNo As Integer, N As Integer, max() As Double, min() As Double)
Dim i As Integer
max(0) = x(CurrentNo, 1): min(0) = x(CurrentNo, 1)
max(1) = 0: min(1) = 0

'Debug.Print "CurrentNo = "; Str(CurrentNo)
For i = 2 To N
If x(CurrentNo, i) > max(0) Then
max(0) = x(CurrentNo, i)
max(1) = i
End If
If x(CurrentNo, i) < min(0) Then
min(0) = x(CurrentNo, i)
min(1) = i
End If
Next i

End Sub
```

手法１のプログラム (f_sum.c)

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
unsigned long    id1;    // 予約領域１
unsigned long    id2;    // このデータ１枚分のサイズ
unsigned char    id3;    // パターン番号
unsigned char    id4;    // 金種
unsigned char    id5;    // 方向
unsigned char    id6;    // 国情報
unsigned int     id7;    // 通し番号
unsigned char    id8;    // 機種
unsigned char    id9;    // 号機
```

```
unsigned char    id10;    // 種別（０：元画像　１：補正後）
unsigned char    id11;    // センサ番号
unsigned int     id12;    // 画像フレームのＸサイズ
unsigned int     id13;    // 画像フレームのＹサイズ
unsigned int     id14;    // センサ有効域チャネル番号（Ｌ）
unsigned int     id15;    // センサ有効域チャネル番号（Ｈ）
unsigned int     id16;    // 予約領域２
unsigned char    id17;    // センサ分解能Ｘ
unsigned char    id18;    // センサ分解能Ｙ
unsigned char    id19;    // 予約領域３
unsigned int     id20;    // 中心画素のＸ座標
unsigned int     id21;    // 中心画素のＹ座標
unsigned long    id22;    // 画像データのオフセット
unsigned long    id23;    // 予約領域４
unsigned char    id24;    // 格納されているスラブ種類数
unsigned char    id25;    // １種類目のスラブ値のマスクｉｄ
unsigned char    id26;    // １種類目のスラブ値の数
unsigned char    id27;    // 予約領域５
unsigned long    id28;    // スラブ値データのオフセット
unsigned long    id29;    // 予約領域６(id1～id29=ヘッダ)

float                     id30;    // 画像データ４byte

int      i,j,n,k,l,m;
float    fch1[1024],fch2[1024],tch1[2048],tch2[2048];
float    t_No[2048],f_No[1024];
FILE     *output_pt,*input_pt;
char     infile[20],outfile[20];

//       float    f1_slab[1024],f2_slab[1024];

float    f1_sum_div=0.0,f2_sum_div=0.0;
float    f1_sum[60]={0.0},f2_sum[60]={0.0};
float    f1_in[60],f2_in[60];
float    f1_input[30],f2_input[30];
float    f1div,f2div;
```

```
void main( void )
{
id1=0;              // 予約領域１
id2=664; // このデータ１枚分のサイズ
id3=1;              // パターン番号
id4=1;              // 金種
id5=1;              // 方向
id6=1;              // 国情報
id7=1;              // 通し番号
id8=1;              // 機種
id9=1;              // 号機
id10=0;             // 種別（０：元画像　１：補正後）
id11=0;             // センサ番号
id12=20; // 画像フレームのＸサイズ
id13=20; // 画像フレームのＹサイズ
id14=4;             // センサ有効域チャネル番号（Ｌ）
id15=201;           // センサ有効域チャネル番号（Ｈ）
id16=0;             // 予約領域２
id17=10; // センサ分解能Ｘ
id18=40; // センサ分解能Ｙ
id19=0;             // 予約領域３
id20=0;             // 中心画素のＸ座標
id21=0;             // 中心画素のＹ座標
id22=64; // 画像データのオフセット
id23=0;             // 予約領域４
id24=1;             // 格納されているスラブ種類数
id25=1;             // １種類目のスラブ値のマスクｉｄ
id26=50; // １種類目のスラブ値の数
id27=0;             // 予約領域５
id28=464;           // スラブ値データのオフセット
id29=0;             // 予約領域６

id30=1;             // 画像データ
```

```
// ファイルオープン (出力)
printf("出力ファイル名を入力＝");
scanf("%s",&outfile);

output_pt=fopen(outfile,"wb");

// REPEAT

printf("ファイル数の入力=");
scanf("%d",&n);

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

// ファイルオープン (入力)
printf("入力ファイル名を入力＝");
scanf("%s",&infile);

input_pt=fopen(infile,"ra");



// ASCII ファイルのﾃﾞｰﾀをリード(No.,tch1(2048),tch2(2048),No.,
fch1(1024),fch2(1024))
for(j=60;j<=1019;j++){
fscanf(input_pt,"%f%f%f%f%f%f",&t_No[j],&tch1[j],&tch2[j],&f_
No[j],&fch1[j],&fch2[j]);
}

for(k=0;k<=24;k++){
f1_sum[k]=0.0 ;
f2_sum[k]=0.0;
for(l=1;l<=40;l++){
f1_sum[k] += fch1[59+l+(40*k)];
f2_sum[k] += fch2[59+l+(40*k)];
}
}
```

```
for(k=0;k<=24;k++){
f1_sum_div=f1_sum_div+f1_sum[k];
f2_sum_div=f2_sum_div+f2_sum[k];

f1_in[k]= f1_sum[k] / f1_sum_div;
f2_in[k]= f2_sum[k] / f2_sum_div;
}

for(k=0;k<=24;k++){
f1_input[k]=f1_in[k+1];
f2_input[k]=f2_in[k+1];
}

f1div=f1_sum_div / 30000;
f2div=f2_sum_div / 30000;

printf("¥n¥n¥n f1_input[1]= %f",f1_input[1]);
printf("      ");
printf(" f2_input[1]= %f¥n",f2_input[1]);

printf(" f1_input[2]= %f",f1_input[2]);
printf("      ");
printf(" f2_input[2]= %f¥n",f2_input[2]);

printf(" f1_sum[1]= %f",f1_sum[1]);
printf(" f2_sum[1]= %f¥n",f2_sum[1]);

printf(" f1div= %f",f1div);
printf("      ");
printf(" f2div= %f¥n",f2div);

// ID の準備(64)
//予約領域１
fwrite(&id1,      4,      1,      output_pt);      //4byte
//このデータ（１枚分の総サイズ）
fwrite(&id2,      4,      1,      output_pt);      //4byte
//パターン番号
```

```
fwrite(&id3,    1,    1,    output_pt);    //1byte
//金種
fwrite(&id4,    1,    1,    output_pt);    //1byte --
-- 10 byte
//方向
fwrite(&id5,    1,    1,    output_pt);    //1byte
//国情報
fwrite(&id6,    1,    1,    output_pt);    //1byte
//通し番号
fwrite(&id7,    2,    1,    output_pt);    //2byte
//機種
fwrite(&id8,    1,    1,    output_pt);    //1byte
//号機
fwrite(&id9,    1,    1,    output_pt);    //1byte
//種別
fwrite(&id10,   1,    1,    output_pt);    //1byte
//センサ番号
fwrite(&id11,   1,    1,    output_pt);    //1byte
//画像のＸサイズ（画素数）
fwrite(&id12,   2,    1,    output_pt);    //2byte --
-- 20
//画像のＹサイズ（画素数）
fwrite(&id13,   2,    1,    output_pt);    //2byte
//センサ有効域チャネル番号（Ｌ）
fwrite(&id14,   2,    1,    output_pt);    //2byte
//センサ有効域チャネル番号（Ｈ）
fwrite(&id15,   2,    1,    output_pt);    //2byte
//予約領域２
fwrite(&id16,   2,    1,    output_pt);    //2byte
//センサ分解能（Ｘ方向）
fwrite(&id17,   1,    1,    output_pt);    //1byte
//センサ分解能（Ｙ方向）
fwrite(&id18,   1,    1,    output_pt);    //1byte --
-- 30
//予約領域３
fwrite(&id19,   1,    6,    output_pt);    //6byte
//紙幣の中心画素のＸ座標
fwrite(&id20,   2,    1,    output_pt);    //2byte
//紙幣の中心画素のＹ座標
fwrite(&id21,   2,    1,    output_pt);    //2byte --
-- 40
//画像データのオフセット
fwrite(&id22,   4,    1,    output_pt);    //4byte
//予約領域４
fwrite(&id23,   4,    1,    output_pt);    //4byte
//スラブ値の種類数
fwrite(&id24,   1,    1,    output_pt);    //1byte
//１種類目のスラブ値のマスクＩＤ
fwrite(&id25,   1,    1,    output_pt);    //1byte --
-- 50
//１種類目のスラブ値の個数
fwrite(&id26,   1,    1,    output_pt);    //1byte
//予約領域５
fwrite(&id27,   1,    5,    output_pt);    //5byte
//スラブ値のオフセット
fwrite(&id28,   4,    1,    output_pt);    //4byte --
-- 60
//予約領域６
fwrite(&id29,   4,    1,    output_pt);    //4byte --
-- 64 (ヘッダの終了)

//画像データ
fwrite(&id30,   4,    100,output_pt);    //4byte- 68 (画像デ
ータの終了)

//(この後にデータとなるスラブ値が書かれる)

// ID +  画像データ  + f1_slab(25) + f2_slab(25)  をファイル出力
//                                  ―――――――    ―――――
//                                        スラブ値

fwrite(&f1_input, 4,    24,    output_pt);
```

```
fwrite(&f1div,      4,      1,      output_pt);
fwrite(&f2_input,   4,      24,     output_pt);
fwrite(&f2div,      4,      1,      output_pt);

// ファイルクローズ(入力)
fclose(input_pt);
}
// go to REPEAT

// ファイルクローズ(出力)
fclose(output_pt);
}
```

手法２のプログラム　(f_sum1.c)

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
unsigned long     id1;      // 予約領域１
unsigned long     id2;      // このデータ１枚分のサイズ
unsigned char     id3;      // パターン番号
unsigned char     id4;      // 金種
unsigned char     id5;      // 方向
unsigned char     id6;      // 国情報
unsigned int      id7;      // 通し番号
unsigned char     id8;      // 機種
unsigned char     id9;      // 号機
unsigned char     id10;     // 種別（０：元画像　１：補正後）
unsigned char     id11;     // センサ番号
unsigned int      id12;     // 画像フレームのＸサイズ
unsigned int      id13;     // 画像フレームのＹサイズ
unsigned int      id14;     // センサ有効域チャネル番号（Ｌ）
unsigned int      id15;     // センサ有効域チャネル番号（Ｈ）
unsigned int      id16;     // 予約領域２
unsigned char     id17;     // センサ分解能Ｘ
unsigned char     id18;     // センサ分解能Ｙ
unsigned char     id19;     // 予約領域３
unsigned int      id20;     // 中心画素のＸ座標
unsigned int      id21;     // 中心画素のＹ座標
unsigned long     id22;     // 画像データのオフセット
unsigned long     id23;     // 予約領域４
unsigned char     id24;     // 格納されているスラブ種類数
unsigned char     id25;     // １種類目のスラブ値のマスクｉｄ
unsigned char     id26;     // １種類目のスラブ値の数
unsigned char     id27;     // 予約領域５
unsigned long     id28;     // スラブ値データのオフセット
unsigned long     id29;     // 予約領域６(id1～id29=ヘッダ)

float                       id30;     // 画像データ４byte

int       i,j,n,k,l,m;
float     fch1[1024],fch2[1024],tch1[2048],tch2[2048];
float     t_No[2048],f_No[1024];
FILE      *output_pt,*input_pt;
char      infile[20],outfile[20];

//        float     f1_slab[1024],f2_slab[1024];

float     f1_sum_div=0.0,f2_sum_div=0.0;
float     f1_sum[60]={0.0},f2_sum[60]={0.0};
float     f1_in[60],f2_in[60];
float     f1_input[30],f2_input[30];
float     f1div,f2div;

void main( void )
{
```

```
id1=0;              // 予約領域１
id2=664; // このデータ１枚分のサイズ
id3=1;              // パターン番号
id4=1;              // 金種
id5=1;              // 方向
id6=1;              // 国情報
id7=1;              // 通し番号
id8=1;              // 機種
id9=1;              // 号機
id10=0;             // 種別（０：元画像　１：補正後）
id11=0;             // センサ番号
id12=20; // 画像フレームのＸサイズ
id13=20; // 画像フレームのＹサイズ
id14=4;             // センサ有効域チャネル番号（Ｌ）
id15=201;           // センサ有効域チャネル番号（Ｈ）
id16=0;             // 予約領域２
id17=10; // センサ分解能Ｘ
id18=40; // センサ分解能Ｙ
id19=0;             // 予約領域３
id20=0;             // 中心画素のＸ座標
id21=0;             // 中心画素のＹ座標
id22=64; // 画像データのオフセット
id23=0;             // 予約領域４
id24=1;             // 格納されているスラブ種類数
id25=1;             // １種類目のスラブ値のマスクｉｄ
id26=50; //  １種類目のスラブ値の数
id27=0;             // 予約領域５
id28=464;           // スラブ値データのオフセット
id29=0;             // 予約領域６

id30=1;             // 画像データ

// ファイルオープン (出力）
printf("出力ファイル名を入力＝");
scanf("%s",&outfile);

output_pt=fopen(outfile,"wb");

// REPEAT

printf("ファイル数の入力=");
scanf("%d",&n);

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

// ファイルオープン (入力）
printf("入力ファイル名を入力＝");
scanf("%s",&infile);

input_pt=fopen(infile,"ra");



// ASCII ファイルのﾃﾞｰﾀをリード(No.,tch1(2048),tch2(2048),No.,
fch1(1024),fch2(1024))
for(j=60;j<=515;j++){
fscanf(input_pt,"%f%f%f%f%f%f",&t_No[j],&tch1[j],&tch2[j],&f_
No[j],&fch1[j],&fch2[j]);
}

for(k=0;k<=24;k++){
f1_sum[k]=0.0 ;
f2_sum[k]=0.0;
for(l=1;l<=28;l++){
f1_sum[k] += fch1[59+l+(28*k)];
f2_sum[k] += fch2[59+l+(28*k)];
}
}

for(k=0;k<=24;k++){
f1_sum_div=f1_sum_div+f1_sum[k];
f2_sum_div=f2_sum_div+f2_sum[k];
```

```
f1_in[k]= f1_sum[k] / f1_sum_div;
f2_in[k]= f2_sum[k] / f2_sum_div;
}

for(k=0;k<=24;k++){
f1_input[k]=f1_in[k+1];
f2_input[k]=f2_in[k+1];
}

f1div=f1_sum_div / 30000;
f2div=f2_sum_div / 30000;

printf("¥n¥n¥n f1_input[1]= %f",f1_input[1]);
printf("      ");
printf(" f2_input[1]= %f¥n",f2_input[1]);

printf(" f1_input[2]= %f",f1_input[2]);
printf("      ");
printf(" f2_input[2]= %f¥n",f2_input[2]);

printf(" f1_sum[1]= %f",f1_sum[1]);
printf(" f2_sum[1]= %f¥n",f2_sum[1]);

printf(" f1div= %f",f1div);
printf("      ");
printf(" f2div= %f¥n",f2div);

// ID の準備(64)
//予約領域１
fwrite(&id1,      4,      1,      output_pt);      //4byte
//このデータ（１枚分の総サイズ）
fwrite(&id2,      4,      1,      output_pt);      //4byte
//パターン番号
fwrite(&id3,      1,      1,      output_pt);      //1byte
//金種
fwrite(&id4,      1,      1,      output_pt);      //1byte --
-- 10 byte
//方向
fwrite(&id5,      1,      1,      output_pt);      //1byte
//国情報
fwrite(&id6,      1,      1,      output_pt);      //1byte
//通し番号
fwrite(&id7,      2,      1,      output_pt);      //2byte
//機種
fwrite(&id8,      1,      1,      output_pt);      //1byte
//号機
fwrite(&id9,      1,      1,      output_pt);      //1byte
//種別
fwrite(&id10,     1,      1,      output_pt);      //1byte
//センサ番号
fwrite(&id11,     1,      1,      output_pt);      //1byte
//画像のＸサイズ（画素数）
fwrite(&id12,     2,      1,      output_pt);      //2byte --
-- 20
//画像のＹサイズ（画素数）
fwrite(&id13,     2,      1,      output_pt);      //2byte
//センサ有効域チャネル番号（Ｌ）
fwrite(&id14,     2,      1,      output_pt);      //2byte
//センサ有効域チャネル番号（Ｈ）
fwrite(&id15,     2,      1,      output_pt);      //2byte
//予約領域２
fwrite(&id16,     2,      1,      output_pt);      //2byte
//センサ分解能（Ｘ方向）
fwrite(&id17,     1,      1,      output_pt);      //1byte
//センサ分解能（Ｙ方向）
fwrite(&id18,     1,      1,      output_pt);      //1byte --
-- 30
//予約領域３
fwrite(&id19,     1,      6,      output_pt);      //6byte
//紙幣の中心画素のＸ座標
fwrite(&id20,     2,      1,      output_pt);      //2byte
//紙幣の中心画素のＹ座標
```

```
fwrite(&id21,      2,        1,        output_pt);        //2byte --
-- 40
//画像データのオフセット
fwrite(&id22,      4,        1,        output_pt);        //4byte
//予約領域４
fwrite(&id23,      4,        1,        output_pt);        //4byte
//スラブ値の種類数
fwrite(&id24,      1,        1,        output_pt);        //1byte
//１種類目のスラブ値のマスクＩＤ
fwrite(&id25,      1,        1,        output_pt);        //1byte --
-- 50
//１種類目のスラブ値の個数
fwrite(&id26,      1,        1,        output_pt);        //1byte
//予約領域５
fwrite(&id27,      1,        5,        output_pt);        //5byte
//スラブ値のオフセット
fwrite(&id28,      4,        1,        output_pt);        //4byte --
-- 60
//予約領域６
fwrite(&id29,      4,        1,        output_pt);        //4byte --
-- 64 (ヘッダの終了)

//画像データ
fwrite(&id30,      4,        100,output_pt);    //4byte- 68 (画像デ
ータの終了)

//(この後にデータとなるスラブ値が書かれる)

// ID +  画像データ  + f1_slab(25) + f2_slab(25)  をファイル出力
//                                ―――――――     ―――――
//                                        スラブ値

fwrite(&f1_input, 4,        24,       output_pt);
fwrite(&f1div,    4,        1,        output_pt);
fwrite(&f2_input, 4,        24,       output_pt);
fwrite(&f2div,    4,        1,        output_pt);

// ファイルクローズ(入力)
fclose(input_pt);
}
// go to REPEAT

// ファイルクローズ(出力)
fclose(output_pt);
}
```