Improvement of EEG Processing System "BBFFT2ANOVA"
- To Generate Detailed Graph Data Corresponding to ANOVA Results -

Takashi Ajiro†, Koichiro Shimomura†, Hirobumi Yamamoto† and Kenichi Kamijo†

†Plant Regulation Research Center, Toyo University, 1-1-1 Izumino, Itakura, Gunma, 374-0193, Japan

Summary
At the Toyo University Plant Regulation Research Center, we have officially performed electroencephalogram (EEG) experiments a number of times. Our purpose is to investigate and analyze EEG fluctuations in individuals after they have ingested vegetables: for example, after consuming Komatsuna (also known as Japanese mustard spinach) or drinking carrot juice. To analyze EEG data derived from past experiments, we had to use two original composite methods, which were ANOVA using statistical software and fluctuation graphs using Microsoft Excel. Either "Basic Analysis" or "Detailed Analysis" was used as the latter method, and although our EEG processing system, "BBFFT2ANOVA," provided Basic Analysis, its analysis capability was not sufficient. In our past studies, we manually calculated the Detailed Analysis on Microsoft Excel. However, this is difficult for people to do if they have to operate many cells, and human errors are common. We designed and developed an improved EEG processing system called "BBFFT2ANOVA v2" that is based on the v1 system. The new system has four primary additions/improvements, the most noteworthy of which is the inclusion of Detailed Analysis instead of Basic Analysis. Furthermore, we confirmed the system's behavior is valid, comparing generated files with manually created ones. The results indicate that the operational costs as well as the amount of analysis processes required will both be considerably reduced by using this system, in EEG research project.

Key words:
ANOVA, SOC, EEG, vegetable ingestion, algorithm, analysis program, BBFFT2ANOVA

1. Introduction

Electroencephalogram (EEG) tests are frequently used for studying psychological influences, transforming original voltage fluctuation with the Fast Fourier Transformation (FFT) or directly deciphering it. Since EEG responses indicate fluctuations in the electrical activity of the human brain (in other words, psychological influences), there have been many studies that have measured and analyzed the effects of sensory inputs on EEG activity, such as hearing, smell, vision, and taste [1–6]. Wave groups, such as θ, α, β, and δ defined by representative values between frequency bands, are commonly used. More specifically, in EEG research, α and β waves are frequently used to evaluate the effects of relaxation. For example, these waves have been used to evaluate mental states while working [7–8]. We have also used these two waves (including sub-α waves) for evaluating "relaxing" and "stress" brain states in our own researches [9–11].

At the Toyo University Plant Regulation Research Center, we have officially performed EEG experiments eight times. Figure 1 shows sample vegetables that had been used in our EEG research project, and they have been improved and cultivated in vegetable research project (by another team in our research center). The Komatsuna (also known as Japanese mustard spinach) were cultivated with original fertilizers. And, the carrots ware an improved breed, and the right one ((b) called "Aroma Red") was designed by our research center to have a fruity flavor. The history of our official EEG experiments is as follows (the number of levels of factors "A" and "E" are contained within the brackets).

1) Komatsuna (A = 2), preliminary, Nov. 2008
2) Komatsuna (A = 3, E = 2), main, Dec. 2008
3) Komatsuna (A = 3, E = 2), preliminary, Oct. 2009
4) Komatsuna (A = 3, E = 2), main, Oct. 2009
5) Agar, (A=2), main, Dec. 2009
6) Komatsuna ((A = 4), (A = 2)), main, Feb. 2010
7) Carrot juice (A = 2), main, Nov. 2010
8) Carrot juice (A = 2), main, Jan. 2011

Fig. 1 Sample vegetables used in our EEG research project, Komatsuna (left) and carrots (right).

There has been no research on EEG measurements and analysis after vegetable ingestion, except our own past works [9–11]. To analyze EEG data measured from past experiments, we had used two original composite

Manuscript received October 5, 2011
Manuscript revised October 20, 2011

methods: ANOVA using statistical software and fluctuation graphs using Microsoft Excel. The former assays significant variances of defined factors and their interactions, and the latter compares the defined levels in such factors (Basic Analysis). The fluctuation graphs for interactions having significances of ANOVA represent the levels of factors related to these interactions (Detailed Analysis).

For our earlier work, we developed an original EEG processing system called "BBFFT2ANOVA" to shorten the operational time and reduce the number of processes. The system provides to generate level tables for ANOVA and to generate graph data as Basic Analysis, but the latter analysis capability is not sufficient. In our past works, we had to manually calculate the Detailed Analysis on Microsoft Excel [10–11]. However, this was difficult to do if there were a lot of cells to operate, and human errors were common. We therefore developed an improved version, called "BBFFT2ANOVA v2" based on "v1" system. This new system features four main additions/improvements, the most noteworthy of which is the implementation of Detailed Analysis instead of Basic Analysis. Furthermore, we confirmed the system's behavior is valid, comparing generated files with manually created ones. The results indicate that the operational costs as well as the number of analysis processes required can be considerably reduced by using this system, in EEG research project.

The rest of this paper is organized as follows. In Section 2, we discuss the preparation of our experiment, including an EEG measurement device and an overview of ANOVA theory. In Section 3, we give an overview of our previous EEG processing system "BBFFT2ANOVA v1" and briefly discuss the ".fft" file formats used for the processing as well as the actual program structure of the system. Our improved v2 system is introduced in Section 4, and the results of our C++ source program with a detailed description of the algorithm are presented in Section 5. In Section 6, we describe a program test of the new system to determine the accuracy of the generated data. Finally, in Section 7 we conclude with a summary of our design and development results.

2. Preparation

2.1 EEG Measurements

1) Simple EEG measurement device: In our EEG research project, we used a simple EEG measurement device called a Brain Builder Unit for the EEG measurement [12] in our EEG research project. A photograph of this device, including the electrodes, is shown in Fig. 2. The two electrodes on the headband adhere to the skin of the forehead, and the other electrode on the clips with electric wire adheres to the left ear lobe (to the right ear lobe in the case of upside-down use). The device is connected to a PC with a Windows XP operating system. EEG measurement software called Mind Sensor II controls the Brain Builder Unit [12]. The software communicates with the measurement device via a serial port, and captures and writes out the EEG data. The system measures EEG data of the left and right sides of the brain separately, and the measured data series is written to an ".fft" file that has an internal CSV format. Figure 3 shows the data visualization function of the software, which displays real-time electric patterns during the EEG capturing process. Especially, the left-top window displays important two kinds of patterns: one is the raw voltage fluctuation (unit: μV) (upside), and the other is the spectrum that indicates each power of frequency using Fast Fourier Transformation (FFT) (downside).

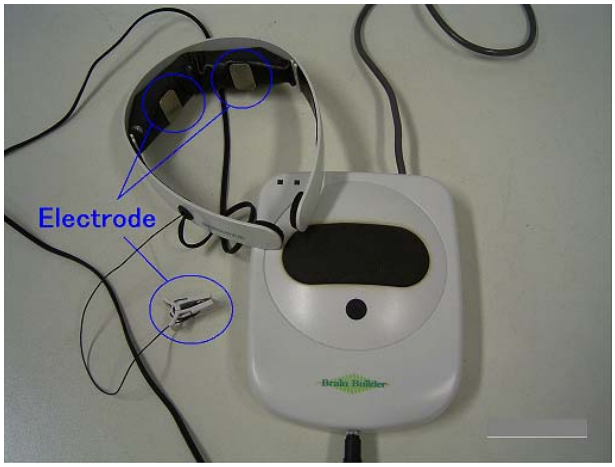

Fig. 2 The Brain Builder Unit with three electrodes.

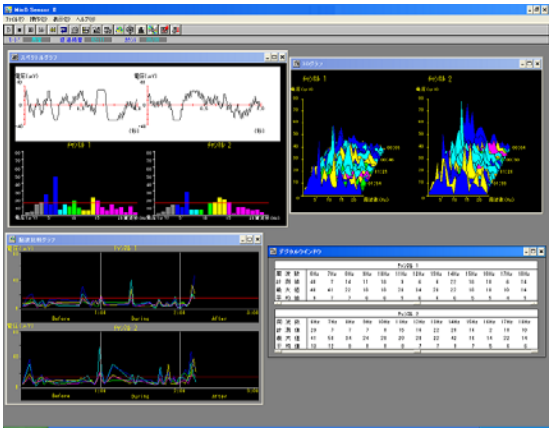

Fig. 3 Display of Mind Sensor II (in Japanese).

2) Definitions of EEG frequency groups: A categorized spectrum defined in an EEG frequency grouping is generally used in EEG analysis instead of the raw electrical fluctuation or its spectrum, even though

definitions of EEG frequency groupings vary according to the researcher. For example, some EEG works [13–14] introduce their own definitions of EEG frequency groupings. We used the definitions in the Mind Sensor II software manual (see Table 1) since they are conformed to our research and are a useful EEG grouping.

The δ wave group (hereafter, a wave group is called a "wave") appears in deep sleep, but we could not use this wave since the EEG measurement device mixes electrical noise. We could use the θ wave as only sub-information since the participants in our study are awake. The area of the α wave is separated into three groupings (α-type waves), since these frequencies are the most important in terms of examining relaxation effects. In addition, fluctuations in these frequencies are active only in awake humans. The β wave appears in the attention state, and we compared it with α-type waves. Although the general boundary of the β wave is considered to be 40 Hz, our definition is 23 Hz due to the limitation of the measurement device.

Table 1 EEG types and corresponding mental states

EEG Type		Frequency	Mental state	
δ		1 – 3 Hz	deep sleep	
θ		4 – 6 Hz	light sleep, meditation	
α	slow α	7 – 8 Hz	relaxing with depressed consciousness	relaxing, creativity uplifting
	mid α	9 – 11 Hz	relaxing with concentration	
	fast α	12 – 14 Hz	concentrating with stress	
β		15 – (23 Hz)	attention, concentration	

2.2 ANOVA Software for Variance Analysis

For the ANOVA calculations, captured EEG data in the ".fft" file is processed and input into a software application called JUSE-QCAS Version 7 [15]. This software supports many statistical operations, including the ANOVA function; Figure 4 is a snapshot of its data-editing mode. In this mode, the user can input data to cells that accept integer values, real values, and characters as text labels. The system recognizes the columns of these cells as two kinds of variables: "quality" and "quantity." The quality variables include the levels of factors, and the quantity variables include the analysis data. In Fig. 4, C3–C5 are quantity columns and N6–N7 are quality columns. The system analyzes the quality values (measurement data) in accordance with the factors and allocations of the quantity values (levels of factors). The software then uses the results to generate a table of ANOVA data (called an ANOVA table).

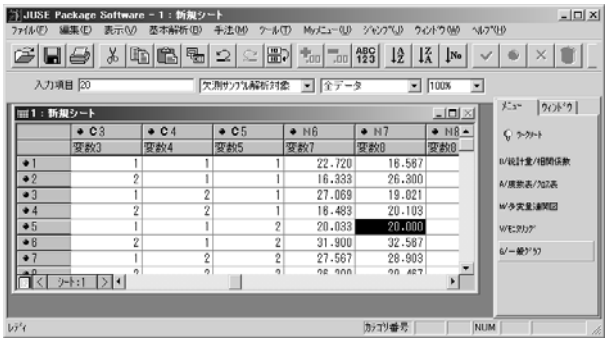

Fig. 4 Display of JUSE-QCAS Version 7 (in Japanese).

2.4 Basic Concept of ANOVA

The ANOVA statistical analysis method is based on the dependencies of factors related to the movement of measurement values. In this method, independent factors are called "main effects" and factors generated by mixing independent factors are called "interactions." The main factors are denoted as A, B, and C, and the interactions are denoted as A × B, B × C, and A × C. The calculation results are called "p-values" (probability values) and are determined using a function of this method called an "assay." The result of the assay is shown as "**" if the p-value is less than 0.01 and by "*" if it is less than 0.05. The p-value indicates the reliability of the significance. For example, a p-value of 0.5 indicates statistical singular values of 5% that are included in the numerous measured values. In other words, it indicates 95% reliability of the analysis results.

The core concept of this theory is the expression of measurement data by the sum of squares that include all factors, errors of measurement, and total accidental errors. The formulation, which is called "structure expression," is a five-way layout of analysis variance and is defined in Fig. 5. Its variance meanings are as follows: y is measurement data, a, b, c, d, and e are the level values of factors, i, j, k, l, and m are identical suffixes for factors, n is an identical suffix for iteration, ε is the total accidental error, and μ is the error of measurement. A more detailed overview of the theory of the analysis of variance can be found in other literature [16–18].

$$
\begin{aligned}
y_{ijklmn} = \mu &+ a_i + b_j + (ab)_{ij} + c_k + (ac)_{ik} + (bc)_{jk} + (abc)_{ijk} \\
&+ d_l + (ad)_{il} + (bd)_{jl} + (abd)_{ijl} + (cd)_{kl} + (acd)_{ikl} + (bcd)_{jkl} \\
&+ e_m + (ae)_{im} + (be)_{jm} + (abe)_{ijm} + (ce)_{km} + (ace)_{ikm} + (bce)_{jkm} \\
&+ (de)_{lm} + (ade)_{ilm} + (bde)_{jlm} + (cde)_{klm} + (abcde)_{ijklm} + \varepsilon_{ijklmn}
\end{aligned}
$$

Fig. 5 Structure expression of five-way layout.

2.5 Experiment Condition (in Case of Eating)

Our EEG measurement experiment sequence is defined as "Sitting down and being silent (before phase) → eating pieces of vegetables (during phase) → sitting down and being silent (after phase)." Participants sit down and are silent during the "before" phase to stabilize their psychological state. The experiments are executed in accordance with a predefined order, e.g., "pair 1 male (kind 1), pair 1 female (kind 1), pair 2 male (kind 1), pair 2 female (kind 1), pair 3 male (kind 2) ...." Both the participants and the researchers are blinded so they can not see which vegetables are being eaten (double-blind experiment). The details of the experimental conditions are as follows.

- Participants put on the Brain Builder Unit electrodes.
- All measurements are conducted in a tent-enclosed space that includes a table for eating the samples. Participants are instructed to sit down quietly with their eyes open to avoid α-wave noise.
- The layout of the experiment table is shown in Fig. 6. An edible sample is placed on a sheet of paper resting atop a cup. An alarm timer is located at the corner of the table.
- Participants are paired (5–10 pairs in general), with one pair consisting of one male and one female.
- The experiment sequence is "before → during → after." In the "before" and "after" phases, participants sit quietly for a number of seconds (20–40). In the "during" phase, which lasts 20–40 seconds, examinees eat a vegetable sample.
- Participants masticate and swallow the vegetable sample in the "during" phase. They are instructed to masticate more than 10 times before swallowing.
- Participants are instructed to eat the samples individually since being fed by another person is a rare action that may generate an invalid EEG measurement due to the psychological effect.

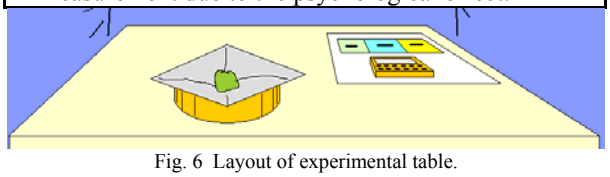

Fig. 6 Layout of experimental table.

3. Overview of the Previous EEG Processing System (BBFFT2ANOVA v1)

3.1 Basic Concept of Analysis Flow

In our analysis environment, we primarily used an EEG measurement device and the four software programs previously introduced (including Mind Sensor II and JUSE-QCAS). Figure 7 shows the analysis flow of our research project, and the rectangles with round corners indicate software used on different analysis phases. BBFT2ANOVA v1 was used in an early phase of the analysis flow and output two kinds of data files. (Note: This system was not named until 2011; in our previous papers, we refer to it as the no-name system [9-11].) The hexagonal shapes indicate final analysis results for a particular research focus of the EEG experiments. Both ANOVA Results and Graph Results are Excel data in the ".xls" format created by individuals, most of whom were students at our university.

The main purpose of v1 was to reduce the expense of and the duration of human work as well as the number of operations necessary to analyze the ".fft" data. In most of our studies, the system was mainly used to support the generation of formatted data for ANOVA because our graph data was insufficient and only included "Absolute Values" and "Basic Analysis." In these cases, we had to manually create Excel graphs for the "Relative Values" and "Detailed Analysis" data. In the next section we will introduce our new version of BBFFT2ANOVA, and will explain this illustrated analysis flow is completely implemented by the improved system.

We used JUSE-QCAS for the analysis to ensure the reliability of the statistical calculations and to take advantage of the more advanced analysis method the software has, and we made the graphs by hand because it easy to process graph data with spreadsheet software and because we wanted to use our original graphs, in which the x-axis was three (or two) phases and the y-axis was the EEG voltages.

3.1 The Format of the ".fft" File and the Meanings of Cells as ".csv" Format

The structure of a ".fft" file is compatible with the CSV format, which separates items with commas. One exception is the second item (number as measuring time), which is enclosed by double quote marks and includes a pre-space. Figure 8 shows how the ".fft" file data appears on a text editor. The first item on a line means the channel from which that line's data was obtained. In this study, we treated channel 1 as a level of the left brain and channel 2 as a level of the right brain.

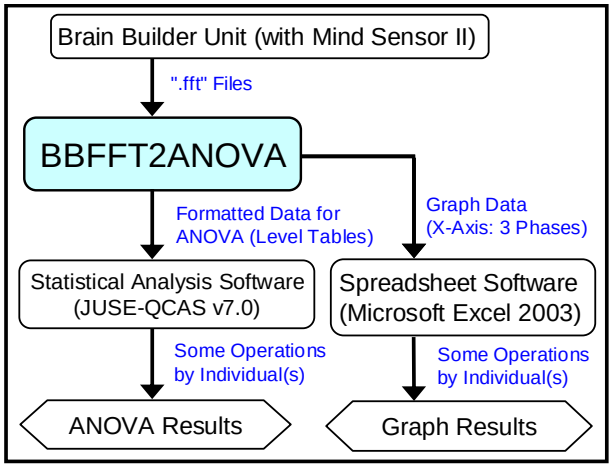

Fig. 7 Analysis flow of our research project.

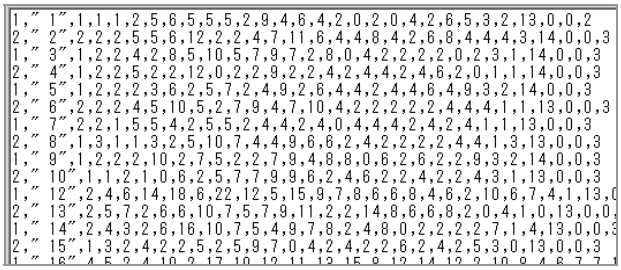

Fig. 8 Appearance of ".fft" file data on a text editor.

Table 2 shows the meanings of items of EEG data in the ".fft" file on Excel. "Cell labels" corresponds to columns on a spreadsheet area, and "meanings of data" explains the meaning of each cell's value. For example, on the values of cell A, the item explains the correspondence relationship of values of the different brain sides. "Wave type" refers to the different waves in EEG. These waves generally have each frequency band as a number of electric voltages.

The B cells include the values of measurement time as the unit s, and generally speaking, these cells have a non-redundant value each other. Cells C–Z include integer voltage values as the unit uV, but the data of C–F could not be used, mainly because of noise (according to the instruction manual). Cells AA–AD include meaningful data—for instance, myoelectric potential—but these cells are not necessary for the purposes of our research and our developed system discards them. The bands of θ waves are ignored in v1 of the system but their data are calculated in the improved system.

3.2 Program Structure and Flow of the System

The program source of BBFFT2ANOVA v1 is a normal procedural structure in the style of C language but using the C++ programming language. This structure is what makes the program flow tractable. We have previously devised and proposed a fundamental algorithm for this system [9]. Thus, the program source was written on the basis of the algorithm, and the flow was also procedural and had no object-oriented basis. Figure 9 shows the structure of the program source (left) and illustrates its program flow (right). The representation of structure shows meaningful groupings of the source program, for instance, the definition part of constant numbers (as defined macros), the declaration part of types and function prototypes, and the definition part of functions. The diagram of the program flow illustrates the execution sequence of the entire program. A rectangle means routine (called "function") and arrowed vertical lines attached to it mean its program flows. In particular, an attached line to the bottom of the rectangle is the main flow of the routine, and horizontal line(s) from the main flow are subroutine call(s).

Table 2 Item meanings of EEG data in ".fft" file on Excel

Cell Labels	Meanings of Data	Wave Type
A	left brain (ch1) = 1, right brain (ch2) = 2	
B	measurement time from start up (s)	
C – F	voltages of 0 – 3Hz (μ V)	noise mainly
G – I	voltages of 4 – 6Hz (μ V)	θ
J – K	voltages of 7 – 8Hz (μ V)	slow α
L – N	voltages of 9 – 11Hz (μ V)	mid α
O – Q	voltages of 12 – 14Hz (μ V)	fast α
R – Z	voltages of 15 – 23Hz (μ V)	β
AA – AD	etc.	

In C language, the main routine determines the main flow of operations in the entire program. The main routine of BBFFT2ANOVA v1 controls key input from a user, by means of an overall nesting loop structure. The subroutine "Proc_eeg" calculates data from one ".fft" file and stores it in the array variable "db" as a seven-dimensional array. In the subroutine, "average_ddata" and "selmax_freq" are called sequentially, and the former routine averages data per phase and per brain side. The latter routine invariably selects the maximum value in EEG frequency bands, since this version supports only the ave-max method.

A subroutine "write_db" writes the data in a "db" array to a file, "data.csv," creating and opening the file with writing mode. The subroutine consists of a whole nesting loop to write the data in a specified order so as to correspond to our conventional manual method. The "write_gr" routine also writes data to a "gr.csv" file, but this data is processed as graph data and stored in the "grdata" array variable as a three-dimensional array by the "gen_grdata" routine, which calculates the Basic Analysis

named in our previous paper [11]. This method is a simple calculation implemented by averaging all the factors except "kind" and "phase." The resulting data has applicable forms to the Excel graph drawing function.

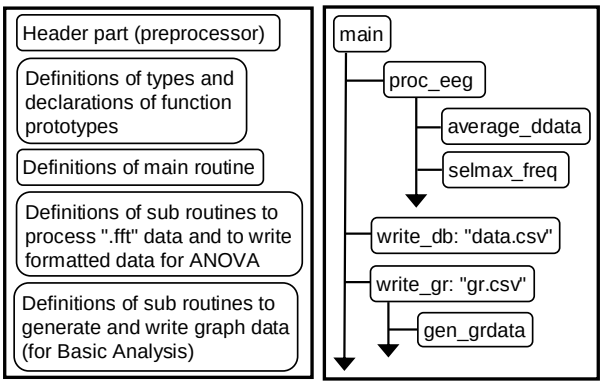

Fig. 9 Structure of program source (left) and program flow (right).

3.3 Generating Files and Their Formats

The internal analyzed data of BBFFT2ANOVA v1 stores the variable "db" as a seven-dimensional array on the returned state to the main routine from "proc_eeg." The system writes out level tables in a specified format for simple pasting to statistical software such as JUSE-QCAS. These formatted data can be used to execute the analysis command simply. Figure 10 shows a combination sequence of levels on a level table, including the calculation results on a slow α wave. The left part of the table shows the sequence of all combinations of levels on factors, with the level values changing cyclically (1, 2, 3, 1, 2, 3, 1 ...). And, the right part shows the sequence of all the calculated values, which are placed in a suitable position corresponding to combinations of level values. These calculated values belonging to different pairs are put alongside each other: for example, "20.6, 26.2, ..." correspond to the labels "<pair 1> , <pair 2>, ... <pair n>." This version of the system writes out level tables that only include calculated absolute values for slow α, mid α, fast α, and β waves by spacing a row.

The cyclic order of the combination sequence is as follows: sex → brain → phase → iteration → kind. At first, the level value of the "sex" factor is incremented by one per data line. When the level value arrives at 2, the next level value of the data line arrives at 1 cyclically and the level value of the "brain" factor is incremented by one. Briefly, the level value of a factor that is ordered later (tail of the right arrow) is increased by one when the level value of a factor that was ordered earlier (head of the left arrow) is cyclically reset to 1. This cyclic order was conventionally determined prior to our past work [9]; however, it is not a rational order. The improved system, as we explain in the next section, writes out level tables by a refined cyclic order.

slow α

kind	sex	brain	phase	iteration	<pair 1>	<pair 2>	<pair 3>	<pair 4>	<pair 5>
1	1	1	1	1	20.6	26.2	19.2	19.5	9.2
1	2	1	1	1	18.3	16.8	7.5	14.2	9.111111
1	1	2	1	1	26.11111	23.77778	23.66667	30.11111	11.11111
1	2	2	1	1	21.33333	14.11111	7.444444	20.33333	11.25
1	1	1	2	1	36.9	35.6	33.2	38.5	24.9
1	2	1	2	1	34.2	25.6	18.9	26.2	18.1
1	1	2	2	1	42.5	31.3	31.1	23.8	22.5
1	2	2	2	1	31.1	25.8	14.1	20.7	19.4
1	1	1	3	1	25.55556	27.77778	27.11111	28.33333	10.77778
1	2	1	3	1	27.55556	11.88889	15.22222	18.66667	13.125
1	1	2	3	1	23.9	34.2	27.9	25.7	17.5
1	2	2	3	1	26.2	18.6	7.8	20	12.77778
1	1	1	1	2	10	15.4	12	18.1	11.2
1	2	1	1	2	18.4	14.1	11.9	15.3	14.6
1	1	2	1	2	6.111111	12.33333	12.88889	14.33333	8.888889
1	2	2	1	2	16	8	11.44444	12.88889	16.22222
1	1	1	2	2	38	40.1	33.7	12.3	19.5
1	2	1	2	2	24.5	36.1	22.1	25.3	18.5
1	1	2	2	2	27.6	37.8	31.6	19.1	23.5
1	2	2	2	2	29.9	31.5	14	36	20.9
1	1	1	3	2	11.33333	26.44444	22.55556	20.88889	9.888889
1	2	1	3	2	23.88889	17.33333	11.44444	11.55556	10.44444
1	1	2	3	2	8.888889	26.1	28.7	14.3	8.8
1	2	2	3	2	19.9	17.5	9.5	21	20.4
2	1	1	1	1	7.888889	13.6	22.3	17.7	20.2
2	2	1	1	1	14.2	10.8	9.8	9.6	16.5
2	1	2	1	1	7.333333	8.222222	12.77778	23.66667	17.77778

Fig. 10 Combination sequence of levels on a level table

Figure 11 shows generated graph data that is included in a "gr.csv" file. These data are separated by commas and line feed codes according to CSV format. The first table (top-left) shows the graph data composed of absolute values for a slow α wave, and the top-right table shows the graph data composed of relative values. The bottom tables show the graph data for a mid α wave. Graph data for other EEG waves are sequentially listed in the file by spacing a row. BBFFT2ANOVA v1 can output the graph data of slow α, mid α, fast α, and β waves as well as write out level tables. In this system, relative values are generated by absolute data, subtracting the "before" phase values from any values. In the table used with this version, the x-axis is electric voltage, y-axis is "phase," and the type of series is "kind." Basic Analysis only calculates the differences between "kind" and "phase" levels, meaning that these generated graphs cannot indicate the differences between the levels of other factors.

slow α

	kind 1	kind 2	kind 3		kind 1	kind 2	kind 3
before	15.24931	15.1925	16.04475	before	0	0	0
during	27.41	25.50556	24.655	during	12.16069	10.31306	8.610248
after	19.03646	17.25472	18.37528	after	3.787153	2.062222	2.330526

mid α

	kind 1	kind 2	kind 3		kind 1	kind 2	kind 3
before	12.13326	11.24222	11.60281	before	0	0	0

Fig. 11 Generated graph data .

4. Improved System: BBFFT2ANOVA v2

4.1 Improvements over the v1 System

In this section, we introduce our new EEG analysis (supporting) system, BBFFT2ANOVA v2, which is based on the algorithm and program source of BBFFT2ANOVA v1. The main improvements to the system are:

1) Supports a new type of calculation method, "ave-ave," in addition to the previous method, "ave-max."
2) Provides additional functions to calculate θ and α waves, and writes out six kinds of EEG waves, including these two.
3) Completely calculates and writes out relative values from individual absolute values (per participant).
4) Executes Detailed Analysis and writes out the results for all sufficient combinations of factors and levels as graph data.

As for the first point, the "ave-ave" method averages original data per frequency and subsequently averages these data per EEG band. It has been used in many studies and is less influenced by abnormal values than other methods. The difference between the "ave-ave" and "ave-max" methods is in the second calculation only: "ave-max" selects maximum values from data that have already been calculated. As for the second point, the calculation of new kinds of waves enables us to investigate more facets of the psychological state of a human being. For example, the θ wave indicates whether a participant is in a meditation state, and the α wave can comprehensively reflect the relaxing state of a human as the averaged (or maximum) values of three sub-α waves. As for the third point, the v1 system could only calculate relative values of graph data from the absolute values of graph data, but the v2 system calculates both relative values of level tables and graph data. Since these data are generated from individual absolute values, the new system can write out both kinds of data files (two each, for a total of four files) by absolute and by relative values. Finally, as for the last point, the addition of the Detailed Analysis function is the largest feature in the v2 system. This function calculates and writes out graph data for all sufficient combinations of levels of factors, but only graph data corresponding to significant factors (including interactions) of the ANOVA results are used for graph analysis. We will explain how these combinations of factors work in detail in a later section.

4.2 Program Structure and Flow of the System

The program structure of BBFFT2ANOVA v2 is quite similar to the previous system since, as stated before, we modified the program source of the v1 system and added new programs for the new features. Figure 12 shows the structure of the program source (left) and its program flow (right). In the structure of the program source, we added a routine to calculate relational values and modified the routine to generate graph data for providing the Detailed Analysis. The program flow has been changed on many points: the routines of writing out data are executed twice for absolute data and relative data, and a "makerel" routine is executed to generate relative data before the second writing process. In the figure, rectangles with broken lines indicate executions (calls) of routines that are already represented in other flows. Since "write_db" and "write_gr" are given by file pointer, they execute calculations for absolute and relative data in the same way.

"Calculate_freq" is a routine that calculates values with the "ave-ave" or "ave-max" method. It integrates the "average_ddata" and "selmax_freq" routines of v1, enabling us to eliminate the two dimensional array-type "DData." "Gen_grpredata" is a new routine in the v2 system that averages pair data before the main calculation process by "gen_grdata." The "makerel" routine is also new, and it calculates relative data from absolute data and stores it in the same array by a reverse loop that decreases the counter variable for the "phase" factor. "Gen_grdata" is an improved routine, and it calculates a part of the Detailed Analysis using processed data from "gen_grpredata" instead of the Basic Analysis used in v1 system.

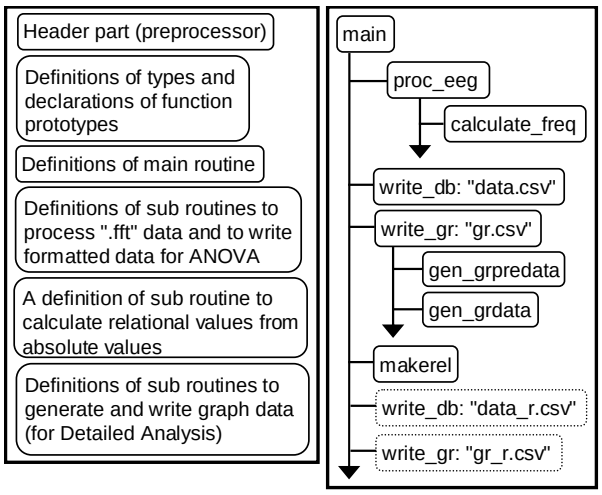

Fig. 12 Structure of program source (left) and program flow (right).

4.3 Graph Data Generation for Detailed Analysis

The "gen_grdata" routine calculates averages specified by the argument "avecode," which has a pointer to a string. For example, a pointer to "BCE" means that the routine first averages factor B, then moves on to C, and finally finishes with E. Briefly, the averaging function is coded by a string to be interpreted with "gen_grdata," and the Detailed Analysis is implemented by an iterative call of "gen_grdata" from "write_gr."

In our program, the averaging codes are defined by an array of strings (a two dimensional array of "char"). The defined codes and their corresponding analysis meanings are as follows.

BCE: AD, CE: AD-B, BE: AD-C, BC: AD-E(ED-A), E: AD-BC, C: AD-BE(ED-BA), B: AD-CE(ED-CA)

The meaning of a description item is "<averaging code>: <series type><factor of x-axis>-<combination of factors>(<another item generated by this averaging code>)," and multiple items are separated by a comma. For example, "BCE: AD" means that the given data are sequentially averaged by factors B, C, and E, and the generating graph consists of A-series and an x-axis allocated for "phase." As another example, "C: AD-BE(ED-BA)" means that the given data are averaged by the C factor, and the generating graphs consist of either A-series and an x-axis allocated for "phase" or E-series and an x-axis allocated for "phase." Parts of -BE and -BA generate individual graphs for combinations of these levels of factors. For example, -BE generates graphs for B×E: (male, attribute 1), (female, attribute 1), (male, attribute 2), (female, attribute 2), ..., (female, attribute n). Briefly, the above codes are an example of the definition of specifications in the Detailed Analysis.

4.4 Generating Files and Their Formats

The improved system writes out level tables differently than the previous system: concretely, in a combination sequence. The cyclic order of the combination sequence is "phase → brain → attribute → kind → sex. " This changed formatting optimizes the act of placing data from the same file close to each other, the main advantage of which is that it is easy to manually locate and obtain a particular file's data. Figure 13 shows a level table (whole), including its combination sequence (left). BBFFTANOVA v2 generates and writes out a total of six level tables for θ, slow α, mid α, fast α, β, and α waves.

theta kind	sex	brain	phase	attribute	<pair 1>	<pair 2>	<pair 3>	<pair 4>	<pair 5>
1	1	1	1	1	27.2	24.4	18.9	14.2	12.4
1	1	1	2	1	23.6	26.4	20.8	24.2	19.4
1	1	1	3	1	20.44444	24	21.66667	20.88889	7.333333
1	1	2	1	1	23.44444	16.77778	21.44444	20	13.77778
1	1	2	2	1	19.7	30.8	23.4	18.8	13.8
1	1	2	3	1	19.4	23.8	24.4	16	11.6
1	1	1	1	2	5.111111	14.4	12.4	18.8	11.8
1	1	1	2	2	13.2	28	21.8	13.4	16.2
1	1	1	3	2	7.555556	24.88889	19.11111	18.44444	10.66667
1	1	2	1	2	4.888889	9.555556	16.88889	10.66667	11.55556
1	1	2	2	2	12.4	27	26	18.3	20.1
1	1	2	3	2	5.555556	29.4	24.4	12.7	8.4
2	1	1	1	1	6.777778	18.6	20.6	15.2	11.5
2	1	1	2	1	18.9	23.8	14	23.4	20.4
2	1	1	3	1	6.222222	14.22222	21	20.22222	8.888889
2	1	2	1	1	8	10.22222	13.55556	17.33333	12.66667
2	1	2	2	1	22.8	22.6	28.8	18.2	19.4

Fig. 13 Combination sequence of levels on a level table.

Figure 14 shows graph data in "gr.csv" generated by the v2 system. The data format of the file differs from the v1 system on three points: 1) the graph data by absolute values are put on a line, 2) both graph data of A- and E-series are generated, and 3) graph data of sufficient combinations of factors are written out. As for the first point, the improved system generates two kinds of files—"gr_r.csv" for relative values in addition to "gr.csv"—instead of two types of graph data on a line. As for the second point, the list of written graph data is strictly compliant with the Detailed Analysis instead of the Basic Analysis in v1 system. As for the last point, the graph data of the A-series consist of a number of kind series while the graph data of the E-series consist of a number of attribute series. The A-series appears before the E-series in the file.

theta: AD			
	series 1	series 2	series 3
before	14.03847	13.25278	13.47653
during	20.1775	20.19722	19.4675
after	15.08389	13.81583	14.74
slow alpha: AD			
	series 1	series 2	series 3
before	14.91618	15.095	16.04475
during	27.3225	25.25556	24.27

Fig. 14 Generated graph data.

5. Details of the Improved System

5.1 Header Part of the Program

In this section, we introduce and discuss the improved system's algorithms, releasing the program source with detailed comments. Figure 15 shows the header part of BBFFT2ANOVA v2 program. These lines are placed on the top of the source file and, via the preprocessor, prepare declarations of external functions and define constant values. These constant values are commonly used to allocate memories for arrays, since the size of the arrays is fixed at the time of compilation. The constants are defined to be of sufficient size, and "-MAX"

means the maximum size of a factor. For example, "KINDMAX" is defined as 10 while the number of levels of the factor "kind" tends to be 4 or 5 generally.

```
#include<stdio.h>
#include<stdlib.h>
#include<memory.h>
//constant values
#define PHNUM 3 //number of phases
#define MFNUM 2 //number of sexes
#define BRNUM 2 //number of brain sides
#define FREQNUM 24 //number of frequencies
#define EEGNUM 6 //number of EEG types
#define KINDMAX 10 //maximum of kinds
#define PAIRMAX 10 //maximum of pairs
#define ATTMAX 10 //maximum of attributes
```

Fig. 15 The header part of BBFFT2ANOVA v2.

5.2 Definitions of Types and Declarations of Function Prototypes

In our system, the types of complicated multiple arrays are redefined by the "typeder" feature of the language, and the system's routines are declared to avoid name reference problems in function calls. Figure 16 and 17 show definitions of types and declarations of function prototypes, respectively. The type of "DBaseSub" is used as a definition of sub-arrays of the "Dbase" array, which stores regular data. An array of "TmpData" is used to store EEG data from an ".fft" file, and "GrData" is the type for graph data to write out to a ".csv" file. The "GrData"-type structure is a factual sub-array of the "DBase"-type structure. The "DBase" array averaged by pair is stored in a "GrData" array by a "gr_predata" routine.

5.3 Definition of Main Routine

A few parts of v2's main routine have been modified from v1's. The main modification, which was inputting the cyclic order of data by console, has been optimized (changed) to the same order as the cyclic order of the level tables. Figure 18 and 19 show definitions of the main routine as a "main function" of C++. The "try–catch" block composes an abnormal flow for exception solution, and an occurring error in the internal block executes a "throw" statement (depending on an "if" evaluation).

5.4 Definitions of Calculation Routines

In addition to the main routine, some parts of v2's calculation routines have also been modified. First, the "calculate_freq" routine integrates the "average_ddata" and "selmax_freq" routines in the previous program and adds a function to select two calculating methods ("ave-max" and "ave-ave") for calculating the given data. Second, the "makerel" routine is added and implemented to generate complete relative data from individual absolute data. Finally, "write_db" is modified to write out six kinds of EEG data: θ, slow α, mid α, fast α, β, and α waves (data of 4–23 Hz are used). Figures 20–23 show definitions of four routines as the "four functions" of C++. In "calculate_freq," an EEG range (from, to) is specified by the two dimensional array for simpler program flow. In "write_db," the composition of the nested loop determines the cyclic order. And, in "makerel," the loop for "phase" works as the reverse loop, since the relative values are calculated by the "before" values. If the "phase" loop is an incremental loop, the "before" values will first be cleared by self-values.

5.5 Definitions of Graph Data Generation Routines

Graph data generation routines consist of added or modified parts to implement the Detailed Analysis instead of the Basic Analysis in the v1 system. Figures 24–26 show definitions of three routines as the "three functions" of C++. The "write_gr" routine sequentially calls the two subroutines, which are "gen_grpredata" and "gen_grdata." However, the sub-main routine "write_gr" also executes complicated calculations and writes out processed data from the two subroutines. For example, multiple nested loops are composed in the routine, and "gen_grdata" is called whenever an interpretation flow on the loops for a new averaging code starts. "Gen_grpredata" is called once on the top of the main flow, and the generated data in the "grbase" array are used as original data to be processed by "gen_grdata." Additionally, in "Gen_grdata," multiple nested loops average data by any factor, depending on an interpreted letter A–E from "avecode." This mechanism is simply implemented by two arrays "cnt" and "tmp" that consist of a dimension. Concretely, the specified element of "tmp" consistently holds to "0" in the nested loops, depending on an interpreted letter. Elements of "cnt" are used as counter variables for these loops, and elements of "grdata" specified by "tmp" are added to values specified by "cnt." Then, the added elements are divided by the number of the target factor after the nested loop for the target factor ends.

```
//////////////////////////////////////Definitions of types//////////////////////////////////////
typedef double DBase[PAIRMAX][MFNUM][KINDMAX][ATTMAX][BRNUM][PHNUM][EEGNUM];
 //array for database (regular data)
typedef double DBaseSub[BRNUM][PHNUM][EEGNUM];
  //sub array of DBase (used to operate a part of data that has DBase type)
typedef double TmpData[BRNUM][PHNUM][FREQNUM];
  //data in a ".fft" file separated by phase, brain and frequency
typedef double GrData[MFNUM][KINDMAX][ATTMAX][BRNUM][PHNUM][EEGNUM];
  //data to generate graphs (sub array of DBase)
```

Fig. 16 The type definition part of BBFFT2ANOVA v2.

```
//////////////////////////////////////Declarations of fuctions//////////////////////////////////////
bool proc_eeg(FILE *file, int phtime[PHNUM], DBaseSub dbsub, bool avemax);
  //load a file, and process EEG data and store it to "dbsub"
bool calculate_freq(TmpData tmpdata, DBaseSub dbsub, bool avemax);
  //select maximum values of averages per frequency in EEG types from "tmpdata" (avemax==true)
  //average values per frequency in EEG types from "tmpdata" (avemax==false)
bool makerel(int pair, int kind, int att, DBase db); //make the data in "db" relational data
bool write_db(FILE *file, int pair, int kind, int att, DBase db);
  //write "db" to a file as level tables for ANOVA
bool write_gr(FILE *file, int pair, int kind, int att, DBase db);
  //write "db" to a file as graph data
bool gen_grpredata(int pair, int kind, int att, DBase db, GrData grbase);
  //average "db" of the pair part
bool gen_grdata(int fcnum[5], char *avecode, GrData grdata);
  //generate graph data and resubstitute them to "grdata", operating the values of "grdata"
  //"fcnum" means numbers of levels of factors, "avecode" is a string to average data of "grdata"
```

Fig. 17 The function declaration part of BBFFT2ANOVA v2.

```
//////////////////////////////////////Definitions of functions//////////////////////////////////////
//**************************************Input control (main)**************************************
int main(int argc, char *argv[]){
  static DBase db={0}; //the database for the regular data
  int kind, pair, att, //number of kinds, number of examinee pairs, number of attributes
      phtime[PHNUM], //times of each phase (unit: second, suffix: 0=before, 1=during, 2=after)
      opmode; //EEG operation mode
  char fname[300]; //buffer to input a filename
  FILE *file=NULL; //file pointer

  printf("¥"BBFFT2ANOVA v2¥" by Takashi Ajiro¥n¥n");
  do{
    printf("Input by [pair kind att opmode(1=ave, 2=max) before during after]¥n:");
    scanf("%d%d%d%d%d%d%d", &pair, &kind, &att, &opmode, &phtime[0], &phtime[1], &phtime[2]);
  }while( //check whether input values are regular
    pair<=0 || pair>PAIRMAX || kind<=0 || kind>KINDMAX || att<=0 || att>ATTMAX ||
    phtime[0]<0 || phtime[1]<0 || phtime[2]<0 || opmode<1 || opmode>2);
  try{
    phtime[2]+=(phtime[1]+=phtime[0]); //adjust to elapsed seconds from starting time
    for(int cpair=0 ; cpair<pair ; cpair++)
      for(int cmf=0 ; cmf<MFNUM ; cmf++)
        for(int ckind=0 ; ckind<kind ; ckind++)
          for(int catt=0 ; catt<att ; catt++){
            printf("Input a filename of <pair %d, kind %d, %s, att %d>¥n:",
              cpair+1, ckind+1, (cmf==0 ? "male" : "female"), catt+1);
            if( scanf("%s", fname)<=0 ) //input a filename
              throw "Unexpected error! This program quits.";
            if( (file=fopen(fname, "rt"))==NULL ){ //open a file
              printf("proc_eeg: Cannot open a ¥".fft¥" file by the filename!¥n");
              catt--;
              continue;
            } //if
            if( !proc_eeg(file, phtime, db[cpair][cmf][ckind][catt], opmode==1) ) //process EEG
              throw "proc_eeg: Measuring times are wrong!";
            write_db(stdout, pair, kind, att, db); //output the results
            fclose(file); //close the file
          } //for catt
```

Fig. 18 The main routine (1).

```
    //-----------------------output absolate data-------------------------
    if( (file=fopen("data.csv", "wt"))==NULL )
      throw "write_db: Cannot open ¥"data.csv¥" with writing mode!";
    write_db(file, pair, kind, att, db); //write "db" as level tables for ANOVA to the file
    fclose(file); //close the file
    if( (file=fopen("gr.csv", "wt"))==NULL )
      throw "write_gr: Cannot open ¥"gr.csv¥" with writing mode!";
    write_gr(file, pair, kind, att, db); //write "db" as Excel graph data to the file
    fclose(file); //close the file
    //-----------------------output relative data-------------------------
    if( (file=fopen("data_r.csv", "wt"))==NULL )
      throw "write_db: Cannot open ¥"data_r.csv¥" with writing mode!";
    makerel(pair, kind, att, db); //make the data in "db" relational data
    write_db(file, pair, kind, att, db); //write "db" as relational level tables to the file
    fclose(file); //close the file
    if( (file=fopen("gr_r.csv", "wt"))==NULL )
      throw "write_gr: Cannot open ¥"gr_r.csv¥" with writing mode!";
    write_gr(file, pair, kind, att, db); //make relational data from "db" and write the data
    fclose(file); //close the file
  }catch(char *err){
    printf("%s¥n", err);
    if( file!=NULL )fclose(file);
  } //cth
  return 0;
}
```

Fig. 19 The main routine (2).

```
//*************************************Process EEG data*****************************************
bool proc_eeg(FILE *file, int phtime[PHNUM], DBaseSub dbsub, bool avemax){
  static TmpData tmpdata; //initialize the data area by "0"
  int num[BRNUM][PHNUM]={0}, //count numbers to average tmpdata
      br, ph, time; //brain-side, phase, a value of time-part in one line data

  memset(tmpdata, 0, sizeof(tmpdata)); //clear the elements
  for(ph=0 ; fscanf(file,"%d,", &br)!=EOF ; num[br][ph]++){ //get the brain-side in a dataline
    br--; //adjust "br" to offset from 0
    if( fscanf(file, "¥" %d¥",", &time)==0 ) //get a measuring time with ".fft" type CSV-format
      fscanf(file, "%d,", &time); //get a measuring time with normal type CSV-format
    if( phtime[ph]<=time ){ //transit to next phase if the time of current phase is over
      ph++;
      if( ph>=PHNUM )break; //escape this for-loop if current time is over
    } //if
    for(int i=0, tmp ; i<FREQNUM ; i++){ //read values from cell "C" to cell "Z"
      fscanf(file, "%d,", &tmp);
      tmpdata[br][ph][i]+=tmp;
    } //i
    while( fgetc(file)!='¥n' ); //proceed to end of line (line feed code is "¥r¥n" on Windows)
  } //for fscanf
  if( ph==PHNUM || num[br][ph]>0 ){
    for(br=0 ; br<BRNUM ; br++) //average values of "tmpdata" by "num"
      for(int ph=0 ; ph<PHNUM ; ph++)
        for(int i=0 ; i<FREQNUM ; i++)
          tmpdata[br][ph][i]/=num[br][ph];
    calculate_freq(tmpdata, dbsub, avemax); //calculate EEG values from "tmpdata"
    return true;
  } //if
  return false;
}
```

Fig. 20 The routine for processing EEG data.

```
//********************Calculate data per frequency and store them to "dbsub"************************
bool calculate_freq(TmpData tmpdata, DBaseSub dbsub, bool avemax){
  int lim[][2]={ //frequency ranges (theta, slow alpha, mid alpha, fast alpha, beta, alpha)
    {4,6},{7,8},{9,11},{12,14},{15,23},{7,14}}; //the range format is {from, to}

  for(int cbr=0 ; cbr<BRNUM ; cbr++)
    for(int cph=0 ; cph<PHNUM ; cph++)
      for(int i=0 ; i<EEGNUM ; i++){
        if( avemax ){ //-------calculation mode of average values---------
          double ave=0;

          for(int j=lim[i][0] ; j<=lim[i][1] ; j++) //sum of the values of the range
            ave+=tmpdata[cbr][cph][j];
          dbsub[cbr][cph][i]=ave/(lim[i][1]-lim[i][0]+1); //the average value is put into "dbsub"
        }else{ //---------------calculation mode of maximum values---------
          double max=tmpdata[cbr][cph][lim[i][0]];

          for(int j=lim[i][0] ; j<lim[i][1] ; j++)
            if( max<tmpdata[cbr][cph][j] ) //check whether "max" is smaller than "tmpdata"
              max=tmpdata[cbr][cph][j]; //renew the value of "max" if the above condition is true
          dbsub[cbr][cph][i]=max; // the maximum value is put into "dbsub"
        } //if
      } //i
  return true;
}
```

Fig. 21 The routine for calculating data per frequency.

```
//***************************Make relational data from the data of "db"*****************************
bool makerel(int pair, int kind, int att, DBase db){
  for(int cpair=0 ; cpair<pair ; cpair++)
    for(int cmf=0 ; cmf<MFNUM ; cmf++)
      for(int ckind=0 ; ckind<kind ; ckind++)
        for(int catt=0 ; catt<att ; catt++)
          for(int cbr=0 ; cbr<BRNUM ; cbr++)
            for(int ceeg=0 ; ceeg<EEGNUM ; ceeg++)
              for(int cph=PHNUM-1 ; cph>=0 ; cph--) //reverse loop for last clearing of "before"
                db[cpair][cmf][ckind][catt][cbr][cph][ceeg]-=
                  db[cpair][cmf][ckind][catt][cbr][0][ceeg];
  return true;
}
```

Fig. 22 The routine for making relational data.

```
//***********************************Write level tables for ANOVA***********************************
bool write_db(FILE *file, int pair, int kind, int att, DBase db){
  static char *title[]={ //strings to indicate brain-wave types
    "theta", "slow alpha", "mid alpha", "fast alpha", "beta", "alpha"};

  for(int ceeg=0 ; ceeg<EEGNUM ; ceeg++){
    //------------------------- output labels -----------------------------
    fprintf(file, "%s¥n", title[ceeg]); //display titles
    fprintf(file, "kind,sex,brain,phase,attribute");
    for(int cpair=0 ; cpair<pair ; cpair++) //data label
      fprintf(file, ",<pair %d>", cpair+1);
    fprintf(file, "¥n");
    //------------------------- output data -------------------------------
    for(int b=0 ; b<MFNUM ; b++) //sex
      for(int a=0 ; a<kind ; a++) //kind
        for(int e=0 ; e<att ; e++) //attribute
          for(int c=0 ; c<BRNUM ; c++) //brain
            for(int d=0 ; d<PHNUM ; d++){ //phase
              fprintf(file, "%d,%d,%d,%d,%d,%d", a+1, b+1, c+1, d+1, e+1); //levels of factors
              for(int cpair=0 ; cpair<pair ; cpair++) //output "cpair" pieces of data
                fprintf(file, ",%.8f", db[cpair][b][a][e][c][d][ceeg]);
              fprintf(file,"¥n"); //end of line
            } //for d
    fprintf(file, "¥n"); //print a empty line to sepalate brain-wave types
  } //for ceeg
  return true;
}
```

Fig. 23 The routine for writing tables for ANOVA.

```
//*************************************Write graph data*********************************************
bool write_gr(FILE *file, int pair, int kind, int att, DBase db){
  static GrData grbase, grdata; //data area for graph data
  static char *title[]={ //strings to indicate EEG types
    "theta", "slow alpha", "mid alpha", "fast alpha", "beta", "alpha"};
  static char *phtitle[]={"before", "during", "after"}, //strings to indicate phases
    *brtitle[]={"left brain", "right brain"}, *mftitle[]={"male", "female"};
  static char avcode[][10]={ //AD, AD-B, AD-C, AD-E(ED-A), AD-BC, AD-BE(ED-BA), AD-CE(ED-CA)
    "BCE", "CE", "BE", "BC", "E", "C", "B"}; //factors to average
  static char grlabel[][7][10]={ //this sequence is the same as "avcode"
    {"AD", "AD-B", "AD-C", "AD-E", "AD-BC", "AD-BE", "AD-CE"},
    {  "",      "",      "", "ED-A",      "", "ED-BA", "ED-CA"}};
  int fcnum[5]; //numbers of levels of factors

  memset(grbase, 0, sizeof(grbase)); //clear the elements of "grbase"
  gen_grpredata(pair, kind, att, db, grbase); //generate predata of graphs
  for(int i=0 ; i<sizeof(avcode)/sizeof(avcode[0]) ; i++){
    fcnum[0]=kind,fcnum[1]=MFNUM,fcnum[2]=BRNUM,fcnum[3]=PHNUM,fcnum[4]=att; //initialize
    memcpy(grdata, grbase, sizeof(grdata)); //copy the values of "grbase" into "grdata"
    gen_grdata(fcnum, avcode[i], grdata); //generate graph data
    for(int mode=0 ; mode<=4 ; mode+=4){ //0 means series of kinds, 4 means series of attributes
      if( fcnum[mode]<=1 )continue; //skip the mode if the number of kind or att is 1
      for(int ceeg=0 ; ceeg<EEGNUM ; ceeg++) //the loop for types of EEG
        for(int cmf=0 ; cmf<fcnum[1] ; cmf++ ) //graphs per sex
          for(int cbr=0 ; cbr<fcnum[2] ; cbr++) //graphs per brain-side
            for(int ckora=0 ; ckora<fcnum[4-mode] ; ckora++){ //graphs per kind or attribute
              fprintf(file, "%s", title[ceeg]); //write EEG type
              fprintf(file, ": %s", grlabel[mode/4][i]); //write the name of a series
              fprintf(file, (fcnum[1]>1 ? ": %s" : ""), mftitle[cmf]); //write "male" or "female"
              fprintf(file, (fcnum[2]>1 ? ": %s" : ""), brtitle[cbr]); //write a label of brainside
              fprintf(file, (fcnum[4-mode]<=1 ? "¥n" : //write a label of kinds or attributes
                (mode==0 ? ": att %d¥n" : ": kind %d¥n")), ckora+1);
              for(int cph=-1 ; cph<fcnum[3] ; cph++){ //output kind labels on -1
                for(int cser=-1 ; cser<fcnum[mode] ; cser++) //output a phase label on -1
                  if( cph<0 )
                    fprintf(file, (cser<0 ? "," : "series %d,"), cser+1); //kind or attribute label
                  else if( cser<0 )
                    fprintf(file, "%s,", phtitle[cph]); //phase label
                  else
                    fprintf(file, "%.8f,", (mode==0 ?
                      grdata[cmf][cser][ckora][cbr][cph][ceeg] : //series of kinds
                      grdata[cmf][ckora][cser][cbr][cph][ceeg])); //series of attributes
                fprintf(file, "¥n"); //the end of a row
              } //for cph
              fprintf(file, "¥n"); //the end of a row
            } //for kora
      fprintf(file, "¥n"); //the end of a row
    } //for mode
  } //for i
  return true;
}
```

Fig. 24 The routine for writing graph data.

```
//*************************************Generate predata of graphs*********************************
bool gen_grpredata(int pair, int kind, int att, DBase db, GrData grbase){
  for(int ceeg=0 ; ceeg<EEGNUM ; ceeg++)
    for(int d=0 ; d<PHNUM ; d++) //phase
      for(int c=0 ; c<BRNUM ; c++) //brain
        for(int e=0 ; e<att ; e++) //attribute
          for(int a=0 ; a<kind ; a++) //kind
            for(int b=0 ; b<MFNUM ; b++){ //sex
              for(int cpair=0 ; cpair<pair ; cpair++) //examinee pair
                grbase[b][a][e][c][d][ceeg]+=db[cpair][b][a][e][c][d][ceeg];
              grbase[b][a][e][c][d][ceeg]/=pair; //average the data by the number of pairs
            } //b
  return true;
}
```

Fig. 25 The routine for generating pre-data of graphs (to average pairs).

```
//*************************************Generate graph data*************************************
bool gen_grdata(int fcnum[5], char *avecode, GrData grdata){
  int cnt[5], tmp[5], val; //[0]=kind, [1]=sex, [2]=brain, [3]=phase, [4]=attribute

  for(int i=0 ; avecode[i]!='¥0' ; i++){
    if( avecode[i]>='A' && avecode[i]<='E' ){ //average the factor specified by avecode[i]
      val=avecode[i]-'A'; //"val" points the first elements to be averaged (index 0)
      for(int ceeg=0 ; ceeg<EEGNUM ; ceeg++)
        for(cnt[0]=fcnum[0]-1 ; cnt[0]>=0 ; cnt[0]--)
          for(cnt[1]=fcnum[1]-1 ; cnt[1]>=0 ; cnt[1]--)
            for(cnt[2]=fcnum[2]-1 ; cnt[2]>=0 ; cnt[2]--)
              for(cnt[3]=fcnum[3]-1 ; cnt[3]>=0 ; cnt[3]--)
                for(cnt[4]=fcnum[4]-1 ; cnt[4]>=0 ; cnt[4]--){
                  memcpy(tmp, cnt, sizeof(tmp)); //copy values of cnt to "tmp"
                  tmp[val]=0; //set the index value of first element
                  if( cnt[val]>0 ) //add the value to the first element
                    grdata[tmp[1]][tmp[0]][tmp[4]][tmp[2]][tmp[3]][ceeg]+=
                      grdata[cnt[1]][cnt[0]][cnt[4]][cnt[2]][cnt[3]][ceeg];
                  else //average the first element of the factor
                    grdata[tmp[1]][tmp[0]][tmp[4]][tmp[2]][tmp[3]][ceeg]/=fcnum[val];
                } //for cont[4]
      fcnum[val]=1;
    }else //if
      return false;
  } //for i
  return true;
}
```

Fig. 26 The routine for generating graph data.

6. Execution of BBFFT2ANOVA v2 and Verification of Results

6.1 Level Table and Corresponding ".fft" Files for the Program Test

We executed and tested BBFFT2ANOVA v2 system with our manual processing procedure to verify accuracy of generated files by the system. The manual processing procedure is precisely defined as operations on spreadsheets to enable the same data to be made by any user. The program (including the algorithm) of the v2 system also refers to the procedure sequence. Thus, we can confirm the validity of the system's behavior, by comparing the data in result files that were generated automatically with those that were made manually, from the same test files. Table 3 shows the level table of an imaginary experiment plan used to test the validity of the v2 system, and table 4 lists the ".fft" files used in the actual program test. The naming rule of the files is based on the allocation table of the imaginary experiment plan, and the general form is "test <pair><sex><kind><attribute>.fft." The <sex> part is replaced by "m" or "f," and the other parts are replaced by their respective defined names. These ".fft" files include random numbers from 0–9999 in the section of values per frequency that were generated by Microsoft Excel's "RAND" function.

Table 3 Allocation table of an imaginary experiment plan for ANOVA

Factor		Level		
Label	**Name**	**1**	**2**	**3**
A	kind	kind a	kind b	
B	sex	male	female	
C	brain	left	right	
D	phase	before	during	after
E	attribute	attribute 1	attribute 2	

Table 4 ".fft" files to verify the execution results of BBFFT2ANOVA v2

Pair	Sex	Kind	Attribute	Filename
1	male	a	1	test1ma1.fft
1	male	a	2	test1ma2.fft
1	male	b	1	test1mb1.fft
1	male	b	2	test1mb2.fft
1	female	a	1	test1fa1.fft
1	female	a	2	test1fa2.fft
1	female	b	1	test1mba.fft
1	female	b	2	test1mb2.fft
2	male	a	1	test2ma1.fft
2	male	a	2	test2ma2.fft
2	male	b	1	test2mb1.fft
2	male	b	2	test2mb2.fft
2	female	a	1	test2fa1.fft
2	female	a	2	test2fa2.fft
2	female	b	1	test2fb1.fft
2	female	b	2	test2fb2.fft

6.2 Execution Results

BBFFT2ANOVA v2 was built as a console Win32 application with Visual Studio 2005 Professional. Figure 27 is a screen grab of the system in execution—specifically, the first scene, which is the input of a calculation specification and required filenames. The execution file is named "bbfftqanova.exe" and the system starts execution simply by inputting the name on "command prompt."

Figure 28 shows Excel snapshots of four files generated by the EEG processing system. "data_r.csv" and "gr_r.csv" include relative data based on the absolute data of "data.csv" and "gr_r.csv." We compared the calculated data of these files with the manually made ones and confirmed that all their values were the same. Since the main data of the ".fft" files are generated by random numbers, there is a very low probability that all values in the generated files will be the same as the manually created ones. This result thus indicates that the system's behavior is valid and that its generating data are accurate.

Fig. 27 Execution state of BBFFT2ANOVA v2 (first scene).

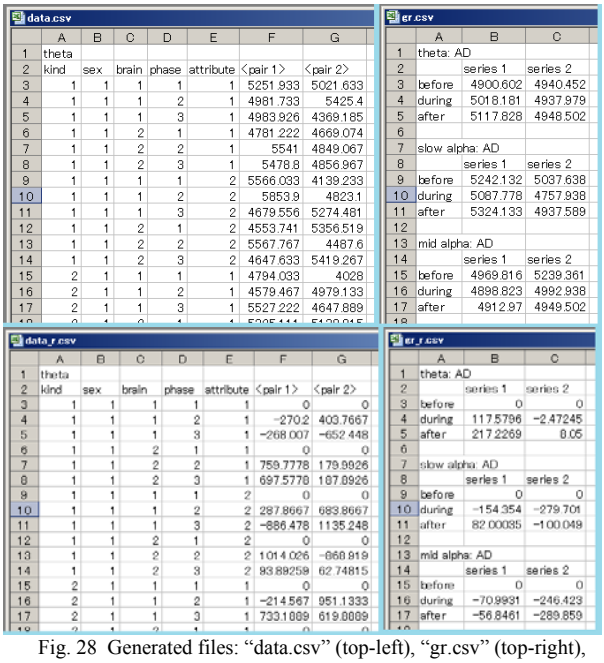

Fig. 28 Generated files: "data.csv" (top-left), "gr.csv" (top-right), "data_r.csv" (bottom-left), and "gr_r.csv" (bottom-right).

7. Conclusion and Future Works

We designed and developed an EEG processing system called BBFFT2ANOVA v2, which is an improvement on our previous version (v1). The improved system featured four additions/improvements: 1) a new type of calculation method, called "ave-ave," 2) additional functions to calculate and write out θ and α waves, 3) the complete function to calculate related values from absolute values, and 4) a new function to execute and write out a Detailed Analysis that calculates graph data for all sufficient combinations of factors and levels. We described the program structure and flow of the v2 system (which was written in C++ with a C-like style) with detailed comments, in Section 4 and 5. And, we tested the improved system to verify generated files by it, and confirmed the accuracy by comparing the result data in these files with result data in files made by the manual operations, from the same test files. The results of our works in this paper can improve costs of working time and quantity in our EEG analysis project and other related EEG researches that use our method, by the new system's four features.

For our future work, first, we will design and develop a more flexible system based on the v2 system. This new system will provide a feature for various experimental plans using a setting file that is CSV or another format and has a calculation specification and filename information for EEG data. And, strings for labels will be set into output data via the setting file's description. Finally, we will redesign and redevelop it as a Graphical User Interface-based system for ease of use by general operators such as university students belonging to non-ICT departments.

References

[1] G. Neil Martin, "Human electroencephalographic (EEG) response to olfactory stimulation. Two experiments using the aroma of food," Int J Psychophysiol, Vol. 30, pp. 287–302, 1998.

[2] J. C. Hashida, A. C. de S. Silva, S. Souto, and E. José Xavier Costa, "EEG pattern discrimination between salty and sweet taste using adaptive Gabor transform," Neurocomputing, Vol. 68, pp. 251–257, Oct. 2005.

[3] M. Nuki, K. Nagata, and H. Kawakami, "The Relations among EEG, Mood, Preference, Personality and Spectrum Power analysis in Listening to Healing music," IPSJ SIG Technical Report, Vol. 2004, No. 111, pp. 35–40, Nov. 2004. (in Japanese)

[4] T. Sakurai and M. Nakagawa, "A Study of EEG Dynamics with Photic-Stimulation," IEICE Technical Report, Vol. 96, No. 569 (NLP96-159), pp. 1–8, Mar. 1997. (in Japanese)

[5] M. Onoda and T. Noji, "Verification of Effect of Smell Healing by Brain Wave," Proceedings of the IEICE General Conference, Vol. 2006 Engineering Sciences, p. 225, Mar. 2006. (in Japanese)

[6] Y. Matsuo, "EEG changes by odors of preferable drinks: the effects of coffee and whisky odors on α-wave," The Japanese journal of taste and smell research, Vol. 6, No. 2, pp. 203–210, Aug. 1999. (in Japanese)
[7] A. Oshima, "A Five-year Study on the Relationships between α-Waves and Business Performance for a Japanese Businessman," Journal of International Society of Life Information Science, Vol. 18, No. 1, pp. 232–241, Mar. 2000.
[8] T. A. Lin, L. R. John, "Quantifying Mental Relaxation with EEG for use in Computer Games," ICOMP 2006, pp. 409–415, June 2006.
[9] T. Ajiro, A. Yamanouchi, K. Shimomura, H. Yamamoto, and K. Kamijo, "A Method for Structure Analysis of EEG Data -Application to ANOVA in Vegetable Ingestion-," IJCSNS, Vol. 9, No. 9, pp. 70–82, Sep. 2009.
[10] T. Ajiro, K. Shimomura, H. Yamamoto, and K. Kamijo, "A Method for EEG Fluctuation Processing -Application to Fertilizer Difference Analysis in Vegetable Ingestion-," IJCSNS, Vol. 10, No. 10, pp. 66–77, Oct. 2010.
[11] T. Ajiro, K. Shimomura, H. Yamamoto, and K. Kamijo, "A Method for EEG Fluctuation Processing II -Application to Pesticide Difference Analysis in Vegetable Ingestion-," IJCSNS, Vol. 10, No. 11, pp. 99–110, Nov. 2010.
[12] Brain Function Research Center Inc., http://www.alphacom.co.jp/.
[13] T. Okuma, "Rinsho-Nohagaku," Igaku Shoin Co., Ltd., Tokyo, Nov. 1963. (in Japanese)
[14] A. J. Rowan and E. Tolunsky, "Primer of EEG: With A Mini-Atras," Butterworth-Heinemann, Mar. 2003.
[15] I-JUSE: The Institute of Japanese Union of Scientists & Engineers, http://www.i-juse.co.jp/.
[16] K. Kamijo, K. Maekawa, and C. Nakabasami, "Introduction to Informatics by Personal Computer," Kougaku Tosho Co., Ltd., Tokyo, 1999. (in Japanese)
[17] H. Nakazato, K. Kawasaki, N. Hirakuri, and A. Otaki, "A Text for Design of Experiments Method for Quality Control (revision and new edition)," Union of Japanese Scientists and Engineers, Tokyo, 1993. (in Japanese)
[18] H. Scheffé Henry, "The Analysis of Variance," Wiley-Interscience, New York, Feb. 1950.

Takashi Ajiro was born in 1980. He received his Ph.D. degree in Engineering from Toyo University in 2008. He has been a research assistant at the Plant Regulation Research Center, Toyo University, since 2008. His main research interests are information science and engineering, especially models of computation, programming languages, and visual language environments and systems.

Koichiro Shimomura was born in 1951. He received his Ph.D. degree in Pharmacy in 1981 from Kyushu University. He joined National Institute of Health Sciences. He is now a professor at Faculty of Life Sciences, Toyo University from 2000. His research interest is mainly antioxidative compounds produced by plants. He is a member of Pharmaceutical Society of Japan, Japan Society for Bioscience, Biotechnology, and Agrochemistry and Japanese Society for Plant Cell and Molecular Biology.

Hirobumi Yamamoto was born in 1960. He received his Ph.D. degree in Pharmacy in 1989 from Kyoto University. He was an assistant professor in Faculty of Pharmaceutical Sciences, Nagasaki University. He is now a professor at Faculty of Life Sciences, Toyo University from 2003. His research interests are biochemistry and metabolic engineering in plant. He is a member of Pharmaceutical Society of Japan and Japanese Society for Plant Cell and Molecular Biology.

Kenichi Kamijo was born in 1949. He received his Ph.D. degree in Geophysics from Kyoto University in 1994. He is now a professor at Faculty of Life Sciences, Toyo University. His research interests include complex systems in informatics, geoinformatics and bioinformatics. He is a member of IEICE, JSAI, Meteorological Society of Japan and Geodetic Society of Japan.