NAIST-IS-DD1461007

博士論文

脳波・脳活動に基づくプログラム理解の困難さ測定

中川　尊雄

2017年 2月 2日

奈良先端科学技術大学院大学
情報科学研究科

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

中川　尊雄

審査委員：

松本 健一 教授　（主指導教員）
飯田 元 教授　（副指導教員）
門田 暁人 教授　（岡山大学）
上野 秀剛 講師　（奈良工業高等専門学校）

脳波・脳活動に基づくプログラム理解の困難さ測定*

中川　尊雄

内容梗概

本論文では，プログラミング学習者によるプログラム理解の難航度合い（理解困難度）を，脳波・脳活動により測定する手法を提案する．本手法によって，理解作業に難航する人物に対して速やかに解決策の提示や課題の変更，休息指示などが行えるようになり，学習の効率化や負荷低減に応用できる．

理解困難度の測定に関する既存研究には，(1) プログラムの解析や計測によるもの，(2) 開発者の主観的な報告・アンケートによるもの，(3) 脳波・脳活動・視線などの生体情報を用いた理解難度の測定がある．特に，被計測者の作業を阻害せずかつ個人の能力に左右されずに理解の難航を測定できる (3) の生体情報計測は近年注目を集めている．ただし，(3) に属する研究はまだ数が少なく，機械学習を用いても二値分類の判別精度が 6 割程度と低い，測定の粒度が不明である，難航の原因が異なっても測定できるか不明である，といった問題がある．

プログラム理解は記憶，言語，意味的統合などの認知プロセスから構成されることが知られており，提案手法では難航の原因がこれらのプロセスの酷使にあると仮定する．そのうえで，既存手法よりも空間分解能やノイズ耐性に優れた近赤外分光法 (NIRS) を用いた記憶の酷使に基づく理解困難度の測定，ならびに，既存手法で多用される脳波計測を用いた，原因の異なる理解困難度の測定を目指す．

20 名の被験者を対象とした NIRS による理解困難度測定の有効性検証実験の結果，記憶の酷使を要求するプログラムの理解時，前頭前野の活動が有意に活発化することが示された (20 人中 17 人, $p<0.01$)．さらに，理解難航時，すなわち理

*奈良先端科学技術大学院大学 情報科学研究科 博士論文, NAIST-IS-DD1461007, 2017 年 2 月 2 日.

解にかかった時間が長い時には，そうでない時に比べて前頭前野の活動が有意に活発化することがわかった．

13 名の被験者を対象とした手法 (b) の有効性検証実験の結果，数値や言語に関する作業記憶のような認知機能を強く要する課題と，そうでない課題の遂行中，脳波の特定周波数成分が有意に異なることを示した．ただし，脳波計測は NIRS による脳活動計測に比べてプログラム理解中の記憶の酷使に対する感度が低いことや，事前に建てた仮説と一致しない傾向が多くみられるなど，その適用可能範囲には限界があることが明らかになった．

本論文の具体的な貢献は次のとおりである．(1) 安価かつ非侵襲な計測装置を用いて，開発者が数十行のプログラムを理解する際に困難を抱えているかどうかを判別できると示した．(2) 安価かつ非侵襲な計測装置を用いて，プログラムの理解に要求される知的作業の種類がある程度特定できる可能性を示した．

キーワード

プログラム理解，理解難度，脳血流計測，脳波計測，被験者実験

Real-time Observation of Difficulty during Program Comprehension based on Brain Measurement*

Takao Nakagawa

Abstract

This paper proposes a method for real-time measuring of difficulty during program comprehension based on phasic brain wave and brain activation.

Program comprehension is a fundamental activity in software development while it cannot be measured easily as it is performed inside the human brain. However some recent studies investigate the cognitive processes involved in the program comprehension activity. According to them, program comprehension is realized by combining cognitive processes such as working memory, semantic and syntactic integration, attention together.

In this paper, we assume that developers who face to the extreme difficulty were required to use cognitive processes beyond their capacity. Proposed method measures the amount and the type of cognitive processes required during program comprehension process.

The method consists of (a) measurement of difficulty during program comprehension based on brain activation using NIRS (Near-Infrared Spectroscopy), and (b) Discrimination of types of difficulty (cognitive processes required for comprehension) based on phasic brain wave measurement.

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1461007, February 2, 2017.

As a result of subject experiment for (a), we found that the brain activation around prefrontal cortex during difficult tasks is larger than easy tasks (17 in 20 participants, $p < 0.01$). As a result of subject experiment for (b), we found that the power spectrum of phasic brain wave is significantly different between tasks. For example ,the power of theta band during the task which requires operation of working memory increases compared to other tasks.

To understand the human aspect of program comprehension, researchers have used indirect measurement such as interview, questionnaire or 'think-aloud' protocol and discuss about mental model of developers during program comprehension. Most of legacy methods assume that the difficulty of program comprehension doesn't change over time, or inhibit the developers' program comprehension process. Therefore recent studies have tried to solve these problems by using bio-metrics measurement. However the existing study which adopt the bio-metrics sensors to measure difficulty during program comprehension don't discuss about the meaning of measured value, and has low noise resistance.

The main contributions of this paper are (1) we found that the proposed method is able to discriminate whether developer faces the difficulty during program comprehension with non-invasive commercial brain measurement device, and (2) we indicated that there is a possibility that the proposed method can discriminate cognitive processes which are extremely required in program comprehension in the laboratory environment.

Keywords:

Program Comprehension, Difficulty, Cerebral Blood Flow Measurement, EEG, Participant Experiments

関連発表論文

学術論文

1. 中川 尊雄, 亀井 靖高, 上野 秀剛, 門田 暁人, 鵜林 尚靖, 松本 健一, “脳活動に基づくプログラム理解の困難さ測定,” コンピュータソフトウェア, Vol.33, No.2, pp.139-150, 2016-06. (3 章に関連)

2. 中川 尊雄, 亀井 靖高, 上野 秀剛, 門田 暁人, 松本 健一, “プログラム理解の困難さの脳血流による計測の試み,” コンピュータソフトウェア, Vol. 31, No.3, pp.270-276, 2014-08. (3 章に関連)

国際会議

1. Takao Nakagawa, Yasutaka Kamei, Hidetake Uwano, Akito Monden, Kenichi Matsumoto and Daniel M. German, “Quantifying Programmers’ Mental Workload during Program Comprehension Based on Cerebral Blood Flow Measurement: A Controlled Experiment,” In 36th International Conference on Software Engineering (ICSE2014): New Ideas and Emerging Research (NIER) Track,, 2014-05. (3 章に関連)

その他の発表論文

学術論文

1. 中川 尊雄, 伊原 彰紀, 松本 健一, “ソフトウェア開発記録の多次元データ分析に向けた可視化方式 Treemap Forest の設計と実証的評価,” SEC Journal, No.45, pp.8-15, 2016-07.

国内会議 (査読付)

1. 中川 尊雄, 藤原 新, 畑 秀明, “プログラミング学習者向けソースコード提示システム: TAMBA,” ソフトウェアエンジニアリングシンポジウム 2016 (SES2016), pp.34-41, 2016-09.

目 次

図 目 次

表 目 次

第1章　はじめに

1　はじめに

プログラム理解はソフトウェアの保守において重要な役割を持ち，保守コストの多くを占める難解な作業である．プログラム理解の効率化による保守コストの低減や品質向上を目指して，理解過程のモデル化や計測，教育法の提案といった研究がこれまでに数多く行われてきた．本論文では，学習段階の開発者がプログラム理解に難航する際，上司や教員が解決策の提示・休息の指示などの介入を行えるよう，理解の難航度合（理解困難度）を脳波・脳活動により測定する手法を提案する．

従来，プログラム理解は人間の脳内で起きる内面的活動であるため，一意な計測やモデル化は難しいとされてきたが，近年では脳活動や脳波，視線など生体情報計測の技術が発達したことにより，その解明が進んでいる．生体情報計測を用いてプログラム理解を測定・モデル化しようという試みには，(1) 高価かつ制約の厳しい医療・研究用の大がかりな機械を用いてプログラム理解の実態を探ろうという基礎研究と，(2) 安価な計測装置をソフトウェア開発の教育・実施現場に適用しようという応用研究の二つのトレンドが存在する．

(1) の例として，Siegmund らはプログラム理解中の人間の脳活動を fMRI (functional Magnetic Resonance Imaging; 核磁気共鳴画像法) によって測定し，プログラム理解が記憶，言語，意味的統合，文法的統合，注意といった認知機能の複合的な利用によって実現されると明らかにした [41][42]．(2) の例として，Fritz らによる脳波，眼球運動，血圧など複数の安価な生体情報計測装置を組み合わせ，読解中のプログラムの理解困難度や開発者の感情状態を推定する取り組みが挙げられる [14].

こうした既存研究の課題として，基礎研究における知見が応用研究に十分活用されていないことが挙げられる．例えば高橋らは商用の計測・分析スイートの出力する算出方法の明らかでない変数を用いて理解困難度の計測に臨んでいる．また，Fritz らは機械学習によって得られたモデルをブラックボックス的に扱っており，推定が実現された機序の考察や，計測精度向上に向けた議論が不足している．

本論文は (2) の発展形としてこの課題の解決を目指す．具体的には，Siegmund らが挙げた認知機能の酷使がプログラム理解の難航に結び付くと仮定し，認知機能の酷使を NIRS (Near InfraRed Spectroscopy; 近赤外分光法) や EEG (Electroencephalogram, 脳波) によって捉えることを目指す．特に，3 章ではプログラム理解中の作業記憶の酷使が NIRS で測定できるかを検証し，4 章ではプログラム理解中の数値記憶，単語記憶，文法的統合，意味的統合の酷使が脳波計測で測定できるかを検証する．

NIRS と脳波計測は装置によっては同時計測が可能であり，それぞれ脳の活動を高い時間分解能で，異なる観点から観察できる．これらの装置は被計測者の体動や姿勢に対する成約が小さく，安価で，商用装置としての開発研究が進んでいるため，今後，技術の発展により教育や開発現場への適用が期待できる．

以降，本論文では続く 1.2 節においてプログラム理解やその測定に関するより詳細な背景を説明したのち，2 章において本研究の位置づけを与えるための関連研究を紹介し，未解決の課題を示す．3 章では，NIRS によって作業記憶の酷使による理解の難航をリアルタイムに計測する手法，ならびにその有効性検証実験について述べる．4 章では，EEG によって困難さの種類を判別する手法，ならびにその有効性検証実験について述べる．5 章では両手法の制約として妥当性の脅威について述べる．6 章では本論文の成果をまとめる．

2　プログラム理解

プログラム理解は人間がプログラムの機能や実行時のふるまいなどについての情報を獲得するプロセスである．より厳密には，図 1.1 のように，ソースコードに対して意味を割り当てる作業であると定義される [28][36].

これまで，多くの研究者がプログラム理解がソフトウェアの保守作業に占める重要な役割について言及してきた．Rugaber らはプログラム理解によってより多くの情報を獲得すれば，より多くのバグ発見や正しい機能追加，再利用に繋がるとしている [38]．Cornelissen らはこれを逆に，バグの発見・修正や機能追加といった保守作業を正しく果たすためには，作業の質に応じて事前に十分な量の情報を獲得していなければならないと表現している [8]．栗山らは実際に開発者のプログラム理解の成否によってバグ発見効率に 10 倍以上の差が表れることを確かめた [51]．

プログラム理解の重要性にもかかわらず，その実施コストは大きく，時にはソフトウェア保守にかける時間の 6 割がプログラム理解に費やされる [37]．また，開発者のプログラミング能力や生産性には 10 倍以上の個人差があり [29]，誰もが容易に達成できるとはいえない．例えば図 1.2 のように，同じプログラムを読んでいても，理解の進行は個々の開発者の能力や事前知識に左右される．経験の浅い開発者は自力で解決できない困難に直面することも多く [25]，こうした開発者が理解に失敗することを防ぐためには，上司や熟練者による適切な教育・介入などの支援が必要である．

適切な支援を行うためには，開発者が理解に支障を来しているタイミングにおいて，開発者がどのような困難に直面しているか，をまず知ることが必要である．しかし，介入を行うべき上司や熟練者が，開発者が理解に支障を来しているタイミングに気づいたり，その種類を判別することは難しい．その理由として，(1) プログラム理解は脳内における種々の認知プロセス (注意，記憶，推論など) の働きで実現される内面的活動であるため，外面的な観察からのみではその詳細を知り得ないこと，(2) 特定の開発者が特定のプログラムを読み進める過程で初めて困難が発生するため，事前に開発者やプログラムを調査するだけでは困難さの度合いや質を知り得ないことが挙げられる．

過去，開発者の内面的活動であるプログラム理解の進捗や成功・失敗を計測するため，インタビューやテスト，あるいは思考内容の発話を伴うアプローチが多く用いられてきた [22][12]．Karahasanovic らはこうした手法には結果の時間分解能と開発者に対する作業阻害の間にトレードオフの関係があると指摘している

```
scanf("%lf %lf", &base, &height);
Triangle triangle1 = Triangle(base,height);

scanf("%lf %lf", &base, &height);
Triangle triangle2 = Triangle(base,height);

printf("tri1: %f\n", triangle1.getArea());
printf("tri2: %f\n", triangle2.getArea());
```

一つ目の三角形の底辺と高さを入力して保存

二つ目の三角形の底辺と高さを入力して保存

各三角形の面積を標準出力に表示

図 1.1 プログラム (上) と，それに対して割り当てる意味 (下) の例

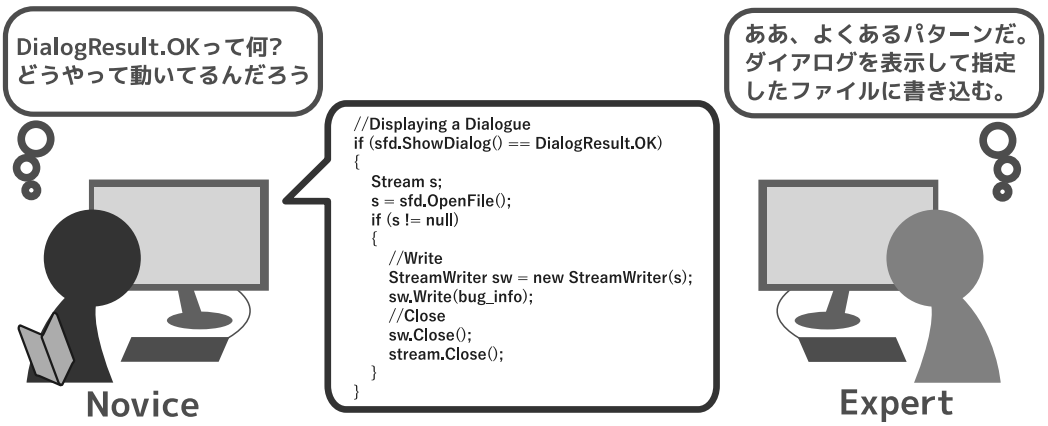

図 1.2 開発者の理解能力には個人差がある

[23]．すなわち，開発者のプログラム理解の成否を即時に知ろうとすればするほど，計測手法自体が開発者の作業を阻害してしまうという問題がある．

このような課題にたいして，前節で述べたように生体情報計測に基づいてプログラム理解の進捗や開発者の状態，プログラムの理解困難度を計測する手法が提案されている．たとえば村岸は，皮膚抵抗値 (Skin Resistance Level, SRL) を用いて C 言語プログラミング中の作業負荷計測を試み，被験者実験によって，プログラミング中の精神的な負荷が測定できることを明らかにした [59]．Busjahn らは，プログラミング初学者の支援を実現するために視線計測を導入することを提案し，そのための計測方法や，結果の解釈方法について論じた [5]．Fritz らは EEG，眼球運動，血圧など複数の安価な生体情報計測装置を組み合わせて，機械学習によってリアルタイムに開発者の感情状態を推定する取り組みを行っている [14]．

これらの手法を利用し，作業を阻害することなくプログラム理解中の困難さの度合いや種類を計測できれば，プログラム理解に行き詰った学習者に対してリアルタイムにアドバイスを与えることができ，学習段階にある開発者に対するプログラミング教育に寄与できると考えられる．

第2章　関連研究

1　プログラム理解の定義やモデル化

従来，ソフトウェア開発に必須の重要な作業であるプログラム理解を支援したり，理解過程や理解度を計測するために様々な研究が行われてきた [45][40].

1.1　古典的なメンタルモデル・理解戦略に関する研究

最も古典的なプログラム理解に関する研究は，心理学的な方法（心理学実験や内観）を用いて，プログラム理解の定義，モデル化，計測を試みるものである．例えば，Pennington らは，自身の内観や既存研究から，開発工程順に要求情報から保守に向かって具体化する流れをプログラム構成，逆順に保守段階の情報から要求に向かって抽象化する流れをプログラム理解であると位置づけた [36].

開発者がプログラムの内容についての心的表象を構築する際の戦略をモデル化，定義する研究も行われている．たとえば，Soloway らはプログラムに関する一般的知識や，理解しようとするプログラムについての前提知識がある場合，熟練者が仮説検証型の読解（トップダウン）戦略を取ることを実験により示した [43]. Pennington らは既存研究の調査から，理解戦略には仮説検証型のトップダウン戦略と，プログラムの制御構造を追うことでプログラムの意味を把握するボトムアップ戦略がありうるとし，異なる戦略の間の関係性についても論じている [36][35]. また，Letovsky はプログラム理解を構成する認知プロセスを明らかにするため，プログラム理解を行った開発者に対するアンケートや，読解中の報告を記録し，開発者のメンタルモデルを推測した [27].

理解戦略に関する議論は一時期において盛んに行われており，実験結果やイン

タビューをもとに，それぞれの戦略を統合したモデルや，各戦略が利用されるタイミングについての研究もおこなわれている [10].

1.2　プログラム理解時の脳の働きに関する考察

近年では，認知心理学や神経科学の知見を活かし，内観のみに頼らず人間の脳の働きや制約を考慮したモデル化が行われるようになり始めた.

Parnin らは，複数のプロジェクトにおいて内容の異なるプログラミング業務に携わる開発者が，開発中のプログラムについての記憶が保持される様子について認知神経科学の観点から理論的に分析を加えた [34]．彼らはまた，実験とインタビューによって，プログラムに関する短期的な記憶や長期記憶の精度，ならびに保持の様子が時間経過によってどのように変化するかを分析調査した.

中村らは，プログラム理解時の人間の短期記憶，特に変数およびコントロールフローの記憶と想起に注目し，プログラム理解のモデル構築を行った．このモデルは実際にプログラムの読解にかかった時間をよく近似することが示されている [33]．また，石黒らは中村らのモデルを改良したモデルを提案し，ある記憶を繰り返し参照することでその内容を忘れにくくなるという，リハーサルの効果が，プログラムの読解にかかる時間に影響を与えた可能性について述べている [58].

2　プログラム理解の計測

2.1　心理学的アプローチ

前節で述べたようなプログラム理解の定義・モデル化や，企業における開発者・プログラムの評価を目的に，プログラム理解を計測することが一般的に行われている.

たとえば，プログラム理解に関する人的要因（理解度，開発者の行動，理解戦略など）について知るための定性的な評価法として，レトロスペクティブ法や Think-aloud 法など心理学実験的な手法が用いられてきた　[12][40]．両手法は，

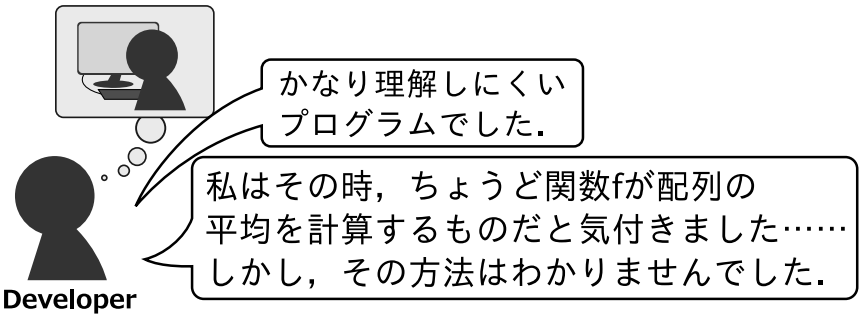

図 2.1 レトロスペクティブ法での報告

被験者が特定の作業（タスク）を実行する被験者実験において，実験中の被験者の感想や行動，思考内容について明らかにするための方法である．

レトロスペクティブ法においては，図 2.1 のように，被験者実験の終了後，あるいは実験を定期的に中断し，インタビューやアンケートを実施し，過去にさかのぼってプログラム理解の過程を答えてもらう．Think-aloud 法においては，図 2.2 被験者実験の実行時，被験者自身の思考内容や行動について随時発話することで同様の情報を得ることを目的としている．この二手法は古典的な方法ではあるが，それぞれ以下のような弱点がある．

1. 結果が被験者の主観に依存するため，定量的に比較できない（共通）
2. 実験終了後の一時点での記録となり，時間分解能が存在しない（レトロスペクティブ法）
3. 実験で取り組むタスクの本質とは関係のない発話という行動を行うため，被験者への負荷が高い（シンクアラウド法）

これらの問題のうち，2) および 3) を緩和するため，Karahasanović らはレトロスペクティブ法・シンクアラウド法に代わる手法として，フィードバック・コレクション法を提案している [22]．この手法では，定期的に「いま何について考えていましたか？」と尋ねるダイアログを画面に表示し，被験者はこれに自由記述形式で解答する．

Karahasanović らによると，この手法はレトロスペクティブ法およびシンクアラウド法のいずれよりも学習コストが低く，レトロスペクティブ法に比べて理解

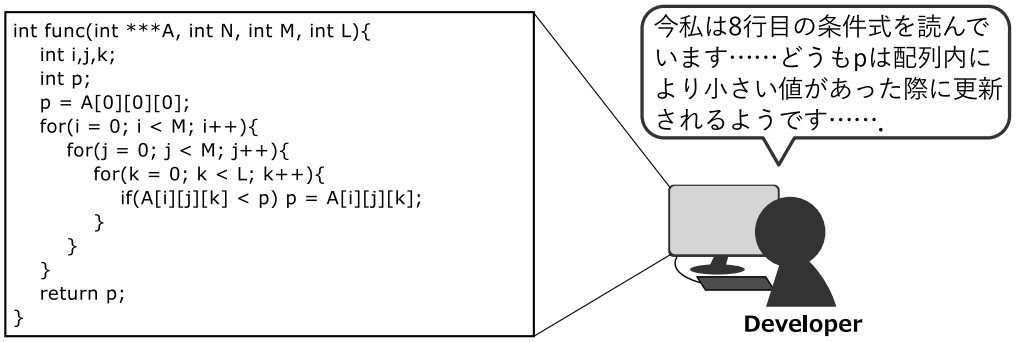

図 2.2 シンクアラウド法での報告

の妨げになる要素を多く発見できると述べている．ただし，フィードバック・コレクション法はやはりシンクアラウド法よりも時間分解能が低く，被験者の主観に依存した記録法では時間分解能と計測時に被計測者にかかる負荷の間にトレードオフの関係があることが示唆された [23]．すなわち，開発者のプログラム理解の成否を即時的に知ろうとすればするほど，計測手法自体が開発者の作業を阻害してしまうと言える．

また三輪らは，プログラミング初学者が試行錯誤しながら目的のプログラムを構成するまでの過程を，セマンティックプロトコル法と呼ばれる高次思考過程の記録方法で詳細に記録し，分析している [53]．この手法では，詳細かつ高時間分解能な記録が可能であるが，訓練を積んだ記録者が被験者を常に監視し続ける必要があり，また被験者も監視され続ける必要があるため，実験者や被験者の負荷が高い．

被験者の主観に依存しない定量的な計測法として，プログラム理解終了後の試験を用いる方法や，プログラム理解にかかった時間を測定する手法が用いられてきた．栗山らは，コードレビューにおけるバグ発見効率とプログラム理解度の関係を調べるため，レビュー終了後に 77 問のテストを課し，高得点を取った被験者ほどバグ発見効率が高くなるという結果を得た [51]．Dunsmore らは，プログラム理解度を測定するためのテスト項目について，どのような問題を出題することで実際の理解度を測定できるかについての比較実験を行った [11]．

これら定量的計測法の課題は，定性的計測における問題点 (2) と同じく，時間分解能が存在しないことである．すなわち，プログラム理解中の特定の時点にお

いて，開発者がプログラムを順調に理解できているかどうかと言ったことは，この方法では知り得ない．

2.2 生体情報計測を用いたアプローチ

心理学的アプローチの制約から，プログラム理解に関する人的要因を定量的に観測可能でき，かつ時間分解能が高く，計測時の負荷が低いアプローチとして，生体情報計測を用いた手法が提案されている．

村岸は，皮膚抵抗値 (Skin Resistance Level, SRL) の計測を用いて C 言語プログラミング中の作業負荷計測を試み，被験者実験によって，プログラミング中の精神的な負荷，たとえば成果の納期が近いかどうかといった要素と SRL の値に関係があることを報告している [59]．村岸の目標である作業負荷計測は，本研究で目指す作業中に困難が生じている状態の計測と立場が近いが，本研究とは，a) 被計測値の持つ意味，b) 計測精度の二点で大きな差があると考えられる．

まず，SRL 測定によって計測できる作業負荷とは，簡単に言ってフラストレーションや精神的な緊張のことを指し，また，逆に負荷のない状態とはリラックスしている状態を指す [56]．このことは，村岸らの実験において残り時間が短い状態で，仕様変更を命じられた開発者の負荷が上昇した点にも明らかである．

これに対して，本研究のアプローチである脳活動の計測は，フラストレーションや精神的な緊張にも影響を受けるが，基本的には脳が実際に知的活動を行う際に必要としたエネルギーの量を測定していると考えられる．このことは，プログラム理解に必要とされた知的活動の量と直接関係のないストレスや精神負荷負荷が計測値にあらわれることを軽減できることを意味する．

また，SRL の持つ計測精度の問題として，測定時に起こる皮膚抵抗反射や自発性皮膚抵抗反射といった一過性の皮膚抵抗変化が計測ノイズとなり，計測時の時間分解能が低下することが挙げられる．極度に強い心理的ストレスがかかった場合に，リラックス状態の抵抗値に戻るまでに 20〜30 分かかることも報告されている報告されている [59]．

視線計測によって，画面中あるいは現実世界のどの部分を注視しているか，あるいはどこも注視していないかといった情報からプログラム理解の様子を観察す

る方法も多く採用されている．

上野らは，コードレビューを行っている人物の視線移動をリアルタイムに記録し，ある時点においてコードや仕様書のどの部分を読んでいるかを計測する，Dresrem 2 というツールを作成した．上野らは，同ツールと視線計測装置を利用した被験者実験で，はじめに全体に目を通してから一部のコードを集中して読む「スキャンパターン」が多く見受けられるほど，コードレビュー時のバグ発見効率が高くなることを発見した [55]．

Busjahn らは，プログラミング初学者の支援を実現するために視線計測を導入することを試みており，そのための計測方法や，結果の解釈方法について論じている [5]．また同じく 2015 年には，実際に被験者実験を行い，学習段階が進むにつれプログラムを上から下へ向かって読むような「線形な読み方」をしなくなる，という結果を報告している [4]．

また Turner らは，視線計測を行った結果を「バグの含まれるコードを見続けた時間」などのかたちで集計し，これを用いてプログラミング言語とバグ発見行動の間に見られる関係を導いた [46]．

視線計測を用いることで，他の計測方法では知り得ない，人間が画面上のどこに注目しているかや，その順序に関する情報を客観的に知ることができるという利点がある．一方，特定部分を読んでいる開発者が何を考えているか，どのように感じているかを知ることはできず，あくまで推測に頼ることになる．また原理的に，何かを見たり読んだりすることを伴わない知的活動を計測できないという課題もある．たとえば，作業を中断してリラックスしようとしている場合や，頭の中で思い悩んでいるような場合に適用できない．

2.3　脳波・脳血流計測を用いたアプローチ

脳血流や脳波から脳の活動を計測する機器が比較的手に入りやすくなったこともあり，近年になってプログラム理解と脳計測を組み合わせた研究が注目を集めている．こうした研究の多くは，プログラム理解・作成に含まれる認知プロセスを同定したり，開発者のストレスあるいは集中の状態を推定するものである．

ごく初期の試みとして，Siegmund らは，神経科学的アプローチをプログラム理解の計測に適用するアイデア，特に，fMRI を用いた実験計画とその準備段階の進捗を国際会議 FSE'12 の New Ideas Track で報告している [41]．同論文では，プログラム理解と脳活動の関係を探ることの重要性が主張されており，長期的には「何がいいプログラマや悪いプログラマを生む原因になるか？」のような課題に解答したいと述べている．Siegmund らはその後，実際に fMRI を用いてプログラム理解中に活動する脳領野を特定する研究を行い，プログラム理解中に問題解決，記憶，文章理解に関する脳部位が活発化することを ICSE 2014 に報告した [42].

fMRI を用いた計測は，空間分解能が高く，特定の脳部位や認知プロセス（記憶，言語など）とプログラム理解の関係性を調査するには最も信頼性の高い手法である．いっぽうで時間分解能が低く，被験者を臥位に固定し，体動を強く制限することになるため，実際のプログラミング現場を模した環境での実験には向いていない．

そこで，安価で簡易な脳波計測機やポータブル NIRS のような機器を用い，実環境に近い状態でプログラム理解を計測する試みが提案されている．

高橋らは，プログラムの作成や自然言語の文章作成などを実施している間の脳波を計測し，作業中に感じたストレスの度合いに関するアンケートと，計測機が出力する Meditation ならびに Attention という二つの値の相関を分析した．実験の結果として，アンケートでストレスが高いと答えていたほど Meditation の値が低く，有意な差が観測されたと報告がなされている．ただし，高橋らの用いた脳波計測機は，Meditation や Attention と言った値がどのような計算式によって出力される値かを明らかにしていない点で，結果の明確性や意味付けの妥当性に疑問が残る [52].

幾谷らは，ポータブル NIRS を用いて，変数と条件分岐の差が脳活動に与える影響を調査し，変数操作が含まれるコードの読解時に前頭極の脳活動が活発化することを報告している [21].

Fritz らはプログラム理解時の視線，脳波，筋電を同時計測し，機械学習を用いて計測値からタスクの性質や開発者の状態の推定を試みた [14]．具体的には，

意図的に結果を想像しにくくしたプログラム（高難度）とそうでないプログラム（低難度）を被験者に与え，その結果を予測させる問題を解いている際の計測値から，プログラムの読解難度を推定させる．

これ以降，同様に多くのセンサを用い，機械学習を用いた測定手法に関する研究が活発に行われている．Muller らはプログラム理解中の開発者の感情状態（集中・発散や幸福・不幸）を心理学的なモデルと突き合わせて推定することを試みた [32]．Zuger らはプログラム理解を試みる開発者の作業効率を下げずに作業割り込みを発生させることを目的に,「作業割り込みを行っても問題ないか？」を二値で推定することを試みた [50]．これらの研究ではいずれも脳波計測機と視線計測機の計測値が機械学習の精度向上に寄与しており，視線・脳波記録がプログラム理解の測定において一定の効果を上げていることが確認できる．

プログラム理解時における脳の活性化度合いや脳波を計測する点において，本節で紹介した研究は本稿と密接に関連しているが，以下の点で異なっている．Siegmund らは，姿勢や体動への制限が大きい fMRI を計測に用いており，実際の開発現場への適用を対象としていない．加えて，難易度の異なる課題の比較を対象としていない．本研究では，体動や姿勢に制限が少ない NIRS を用い，実環境に近い状態で，難易度の異なるプログラムを理解する際に生じる困難を判別することを目的にしている．

幾谷らは数値演算，変数操作，条件分岐の 3 種類の構文を独立に含む 10 行未満の短いコード片を対象としているが，本研究では 17〜32 行の，一定の目的を持った関数を対象としている．また，プログラムの難読化を用い，被験者の主観評価を含めて難易度の違いの判別を目的にしている点で，幾谷らの研究とは立場が異なる．

Fritz らが利用している機械学習を用いた手法は，各計測値の高低と実験条件（難易度の高低や開発者の状態）の間にある関係や理由づけを行うことが難しい．また，精度や再現率も二値分類で 6 割から 7 割程度とそう高くないため，依然さらなる調査が必要な状態であると言える．

そこで本研究では，(a) 特定部位の脳活動の活発さを測定する脳血流計測装置 (NIRS) を用いて難易度の高低を測定する手法と，(b) 既存研究によって特定の認

知プロセスと計測値の関係が明らかになっている脳波計測装置を用いて直面した困難さの種類を判別する手法を提案し，それらの有効性について検討を加える．

3　異分野における認知プロセスと脳活動の関係の分析

神経科学分野においては，各種の知的活動と脳の関係を調査する際，脳活動に関係する生体情報を計測することが一般的である．脳周辺情報の計測のために用いられる一般的な手法・機器には，脳の血流動態を測定する fMRI（functional Magnetic Resonance Imaging; 核磁気共鳴法），脳活動に起因する頭表面の電位変化（脳波）を測定する EEG，脳血流の酸化傾向を測定する NIRS などがある．

脳活動計測を用いた多くの研究では特定の知的活動（例えば暗算 [49] や自然文読解 [24]）や身体動作と，特定の脳部位 (前頭前野，頭頂葉，側頭葉) における活動量の関連を調査することに焦点を当てている．たとえば，Zago らは暗算を行う人物の脳活動について，二種の複雑度（掛け算の九九に代表される記憶の想起と，二桁以上の掛け算のような実際の計算）でどのように差分があるか PET（Positron Emission Tomography）を用いて比較している [49]．また，Buckner らは PET や fMRI によって得られた既存研究のデータを元に，作業記憶の操作や情報の統合と関連すると考えられている前頭前野について，長期記憶の操作時にも同様の活性が見られるかを調査し，長期記憶に保存されたエピソード記憶や意味記憶の想起と前頭前野の活動に関連があることを明らかにした [3]．こうした分析を行った研究の数は非常に多く，各脳部位とどういった認知プロセスが関係しているかについての知見は工学的に十分と言える程度に集まっている．Cabeza らは 275 件の fMRI/PET データを用いた研究をサーベイし，認知プロセスと脳部位の対応を詳細に報告した [6].

これらの既存研究に共通の特徴として，非常に単純な課題（例：一つの単語を発話する，一桁の掛け算を行う，二桁の掛け算）と非常に細かい脳部位の関係について，時間分解能が低い環境で述べているものが多いことがあげられる．一方，本研究で観測の対象とするプログラム理解は，推論，記憶の操作，情報への注意など多種多様な認知プロセスが組み合わさって行われる複雑な知的活動であると

考えられ，理解に際して活動する脳部位も広い範囲にわたると考えられる．そこで本研究では特に，プログラム理解時に間違いなく行われているいくつかの活動，記憶の操作や推論などと関連があるとされる前頭前野 [48][54] の全体に注目し，高時間分解能での計測を試みる．

ただし，前述した実験の多くは，計測機器の制約により，臥位姿勢かつ無運動で課題を遂行するものが大半であり，本研究で目指すような複雑な課題を遂行する場合，被験者に過大な負荷がかかることが想定される．

本論文では，開発者の作業記憶の酷使を前頭前野の脳活動から測定するために 3 章で NIRS を，開発者の種々の認知プロセスにかかる負荷を認知活動の状態から測定するために 4 章で EEG を用いる．これらの装置に共通する特徴は，fMRI や PET など他の計測装置と比較して時間分解能が高く，脳領域に対する空間分解能が低いこと，計測対象者の姿勢・体動への制限が少ないこと，および安価・軽量な商用計測装置の開発が進んでいることである．すなわち，脳のどの部位が活動したかについての詳細な分析には向かないが，脳がいつ・どのように活動したかについての情報を応用的な目的で計測するのには向いている．そのため，両装置の利用は本研究の遠目的である学習者への教育現場における計測の適用に適していると考えられる．

より具体的には，NIRS は脳が活動した際の神経活動に応じて酸素を供給するために脳血流量とその酸化度合が上昇する現象を利用して脳活動量や部位を特定する手法である．血中の酸化ヘモグロビン (oxy-Hb) と脱酸化ヘモグロビン (deoxy-Hb) の吸光特性が異なることを利用し，頭皮表面に照射した近赤外光の反射成分を検知することで脳表層における活動を計測する．fMRI や MEG (Magnetoencephalography) などの他装置と比較して，計測に必要な準備が少なく取り付けが容易で，姿勢や体動の制限が少ないほか，時間分解能 (計測値の時間方向への精度) が高いという利点がある．

NIRS による計測の弱点として，頭部の動揺や極端な運動によるノイズが知られているが，頭部の動揺についてはノイズ除去法が提案されている．また運動についても，歩行中の脳活動や手の操作を伴う実験で十分な結果が得られていることから，着座状態で数時間程度プログラムの読解を試みる環境では大きな問題に

ならないと考えられる．

EEG は脳が活動した際の神経活動に応じて発生する電気的活動を，頭表面の電位変化としてとらえる手法である．脳活動の発生した具体的な部位から頭皮までの距離が長いため，具体的な思考の内容や活動した脳部位の特定は難しいが，多くの既存研究によって，周波数成分と認知的活動の強度や被験者の精神状態の関係性が報告されている．

EEG による計測の弱点として，まばたきや唾の嚥下などの小さな体動でも大きなノイズが発生し，その影響を受けることが知られている．これは事前に周波数フィルタリングや窓関数を適用し，数分程度の時間をかけた場合の代表値を取得すればある程度除外できると考えられる．また，EEG の計測値は実際にプログラム理解の測定を目指す複数の関連研究においてブラックボックス的に用いられており，そもそも本装置によって理解課題中の認知機能の酷使を捉えることができるのかどうかを調査することは今後の研究を考えるうえで重要な課題である．

第3章　脳血流計測によるプログラム理解時の困難さの度合い判別

1　あらまし

これまでに述べたように，プログラム理解は人間の頭の中で行われる高度な知的活動であり，より詳細には記憶，言語，意味的統合，文法的統合，注意といった認知プロセスの働きによって成立している．本論文では，プログラム理解の難航はこのれらの認知プロセスの酷使によって捉えられると仮定し，教育現場への応用を見据えた計測法を提案・検証する．特に本章では，既存研究で取り上げられてこなかったNIRS装置を用い，プログラム理解中の作業記憶の酷使を脳血流の酸化度の変化としてとらえられるという仮定のもと，理解困難度の計測手法を提案する．

具体的には，提案手法の妥当性や適用範囲を調査するため，被験者実験によって次に挙げる3つのリサーチクエスチョンに解答することを目指す．

RQ1：プログラム理解に困難が生じている状態を，理解時の脳血流によって判別できるか？

過去の研究において，記憶の操作や意識の統合に代表される高次の認知機能を担うとされる脳前部 (前頭前野) [54] の活発化が報告されている．RQ1では，理解時に記憶の酷使を要求するプログラムによって理解困難状態（理解が停滞・遅延する状態）を引き起こし，記憶の酷使を要求しないプログラムと比べて脳活動が変化するかを検証する．

RQ2：プログラム理解の困難さの度合いを，脳血流計測によって区別できるか?

RQ2 では，本手法で測定できる理解困難度の粒度や，その境界などを明らかにするため，難易度の異なる (3 段階) 課題プログラムを用意し，被験者実験によって計測値の差を検証する．

RQ3：脳血流計測の結果は，被験者が感じる主観的な難易度を反映しているか?

RQ3 では，本手法で測定した理解困難度が，被験者アンケートによる主観的な難易度とどのような関係にあるかを分析する．

2　実験

異なる難易度のプログラム理解を試みる際の脳活動を NIRS で計測する被験者実験を行った．被験者は奈良先端科学技術大学院大学，奈良工業高等専門学校，九州大学の学部生・大学院生計 20 名で，全員が 20 歳から 24 歳の男性で，プログラミング経験が 3 年以上であった．

2.1　手順

実験において被験者は，NIRS 装置を装着した状態で，理解難易度の異なる 2 種類の暗算タスクと 3 種類のプログラム理解 (easy，medium，difficult) タスクを 1 つずつ，計 5 タスクを実施する．

暗算タスクには，難易度ごとに前頭前野の脳活動に差があると報じた文献 [49] で用いられたものを利用する．本研究では，事前に知られた 2 種類の暗算タスクの間に見られる脳活動の差と，難易度の異なるプログラム理解タスクの間に見られる脳活動の差を比較し，その一致度を見ることで，後者の結果の信頼性を補強することをはかる．

プログラム理解タスクには，機能の異なるプログラムを用いる．各プログラムに対して理解しやすさを変更した 3 つのソースコード (easy, medium, difficult) を

作成し，タスクとして用いる．学習・順序効果の影響を防ぐため，easy，medium，difficult で同じ機能のプログラムが提示されないよう配慮し，実施する難易度・機能の順番も均等にランダム化する．また，実験に支障のないよう，プログラム理解タスクと同等の練習タスクを設け，この間は被験者からの説明を受け付ける．

表 3.1 に実験手順を示す．実験の最初と各タスクの間には，前後のタスクが計測値に与える影響を抑えるために，紙に印刷した十字の模様を二分間注視してもらう．

2.2 タスク

プログラム理解タスク: プログラム理解タスクにおいては，理解中に生じる困難によって脳活動に変化があるかを調査するため，理解困難状態を人為的に誘発する．被験者の能力にかかわらず理解困難状態を誘発できるように，本タスクでは理解が困難となる要因と，理解のための戦略に制約を加える．

まず，理解が困難となる要因には，アルゴリズム自体の複雑さや，保守の過程におけるコードの劣化などが考えられる．本研究の目的は学習段階の開発者を対象に理解困難度の定量化するものであるため，初等プログラミング教育で一般的にみられるような小規模プログラムにおけるアルゴリズムの複雑さに着目する．一般に，制御フローが複雑であるほどプログラム理解が困難となり，バグ混入の可能性が高まることが知られている [1].

そこで本タスクでは次の手順で難読化手法によって制御フローの複雑さに起因する理解困難状態を誘発することを試みる．

1. 十数行の簡単な C プログラムに制御フローを複雑化する難読化手法を適用して，その複雑度を調整する．

2. 被験者に，制御フローを理解するための理解戦略（メンタルシミュレーション）を使うよう要請したうえで，十分程度のプログラム理解課題を課す．

3. 理解課題遂行中の一定期間中，制御フローが複雑すぎるために被験者は理解困難状態に陥る．

表 3.1 実験手順

手順	内容	時間	備考
1	実験内容の説明		
2	装置の取り付け		
3	注視点の凝視	2 分	装置キャリブレーションを実施
4	簡単な四則演算	2 分	問題を解き終われば次の手順へ
5	注視点の凝視	2 分	
6	難しい四則演算	2 分	問題を解き終われば次の手順へ
7	注視点の凝視	2 分	
8	練習タスク (p-hard)	10 分	問題を解き終われば次の手順へ
9	注視点の凝視	2 分	
10	実施タスク 1(各難度のうちいずれか)	10 分	問題を解き終われば次の手順へ
11	注視点の凝視	2 分	
12	実施タスク 2(各難度のうちいずれか)	10 分	問題を解き終われば次の手順へ
13	注視点の凝視	2 分	
14	実施タスク 3(各難度のうちいずれか)	10 分	問題を解き終われば次の手順へ
15	注視点の凝視	2 分	
16	装置の取り外しとアンケート		

ここで適用する難読化手法は，難読化前と難読化後の間で，理解にかかる時間が 6 倍以上変化するようなものを用いており，それでいてプログラムの実現する機能や実行時に通過するステップ数は変化しないことが確認されている [31]．このことから難読化前に比べて難読化後のプログラム理解過程では，一定の期間にわたって被験者の理解は停滞・遅延しているとみなせる．ただし，難航が起きるタイミングは理解の進み具合や理解能力による個人差があるため，分析においては，脳活動の大きさを十分間のタスク中の分布をあらわす箱ひげ図を用いて比較する．このようにすることで，プログラム理解の難航を含む時系列と，含まない時系列の間で脳活動の大きさの変化を捉えることができる．

プログラム理解戦略は理解時の方針を抽象的に分類したものから，具体性の高い方法まで幅広く提唱されており，開発者はこれらの方法を適宜切り替えたり組み合わせたりしながらプログラムを理解してゆく [45]．

抽象的な分類としては，理解の方針をトップダウン戦略とボトムアップ戦略に分類する方法が知られている [34]．トップダウン戦略では，開発者はプログラムの実現しようとする機能やアルゴリズムについての既存知識（ドメイン知識）を用いて，プログラムの構造・意味・ふるまいについての仮説を立て，それを検証するように読み解くとされる．逆にボトムアップ戦略においては，事前知識に頼らず，プログラムコードを行ごとに読み解いていき，その結果を統合することでプログラムの構造・意味・ふるまいについての情報を頭の中に構築していくとされる．

より具体的な理解の手順としては，モジュール間の呼び出し関係を調べる方法 [13] や，データの流れを追跡する方法 [13]，プログラムの Goal と Plan に対する仮定を立てて検証する方法 [53]，プログラムの実行過程を追跡する方法（メンタルシミュレーション）[36] などが提唱されてきた．

本タスクでは，制御フローのふるまいを理解するために一般的に用いられており，適用範囲が広く，被験者に事前知識を必要としないことから，理解戦略をメンタルシミュレーションに限定する．プログラムについての事前知識を持たない開発者は，メンタルシミュレーションを行う過程でプログラムが実現している機能や，その目的に関する情報を獲得できるとされる．また，プログラムについて

表 3.2 解答用紙の記入例 (4 行目が誤り)

A[0][0]	A[0][1]	A[1][0]	A[1][1]	A[2][0]	A[2][1]	N	M	i	j	x	位置
3	5	4	2	1	8	3	2				(0)
										3	(1)
								0	0	5	(2)
								0	1	5	(2)
								2	1	8	(2)
								3	2		(3)

事前知識を持つ開発者でも，デバッグやコードレビューなどの過程において，プログラムが真に正しく所望の機能を実現しているかを人の手で検証するために必要な作業であるとされる．

上記の条件を踏まえ，本タスクでは紙に印刷された 17〜32 行の C 言語の関数と引数のペアを元に，被験者が動作をメンタルシミュレーションし，その過程を解答用紙に記入する．図 3.1，3.2 にタスクで用いるプログラムの例を，表 3.2 に解答用紙の記入例を示す．

被験者はメンタルシミュレーションが目印 (ソースコードの右側に書かれた番号) の行に到達するたびに，各変数の値と目印の番号を解答用紙に記入する．実験者は随時回答を確認し，値がすべて正しければシミュレーションを続けてもらう．そうでなければ，前回の正答時点からシミュレーションをやり直し，再度解答用紙に記入してもらう．目印はループ中に含まれることもあるため，同じ目印に対して複数回の回答をすることもある．タスクは，メンタルシミュレーションが終了するか，10 分の制限時間を迎えた時点で終了とする．

タスクで用いるプログラムの概要を表 3.3 に示す．easy で提示されるソースコードはそれぞれの機能の一般的な実装であり，medium，difficult は，easy に制御フローを複雑にする難読化手法 [31] を適用することで作成する．medium に用いる手法は，プログラム内の文を複製し，条件分岐のリンクを付け替えることで制御

```
A[0][0] = 3, A[0][1] = 5
A[1][0] = 4, A[1][1] = 2
A[2][0] = 1, A[2][1] = 8
N = 3
M = 2
```

図 3.1 実験で提示した引数の例

```
01: int func(int **A, int N, int M)
02: {
03:     int i,j;
04:     int x;
05:
06:     x = A[0][0];
07:     for(i=0; i<N; i++)
08:     {
09:         for(j=0; j<M; j++)
10:         {
11:             if(A[i][j] > x)
12:             {
13:                 x = A[i][j];
14:             }
15:         }
16:     }
17:     return x;
18: }
```

図 3.2 実験で提示したソースコードの例

表 3.3 使用したプログラムの概要

名前	難易度	機能	行数
p0	practice	配列内の最大値の探索	18
p3	practice	配列内の最大値の探索	36
eA	easy	配列内の最小値の探索	18
eB	easy	配列内の数値の合算	21
eC	easy	配列に含まれる特定文字の数え上げ	18
mA	medium	配列内の最小値の探索	25
mB	medium	配列内の数値の合算	23
mC	medium	配列に含まれる特定文字の数え上げ	23
dA	difficult	配列内の最小値の探索	33
dB	difficult	配列内の数値の合算	28
dC	difficult	配列に含まれる特定文字の数え上げ	25

構造を複雑化する．difficult に用いる手法ではこれに加え，ループ条件式のうち，通常は定数である部分を変化させることで，文の評価順を極めて複雑にする．図 3.3 に，図 3.2 で提示した例に難読化手法を適用した後のコードの例を示す．難読化後のコードでは，for 文の初期化子が書かれていない，ループ中でループ条件に使われている変数が変わる，あるいはループカウンタが減少するなど一般的に見られない制御構造に変容していることがわかる．難読化手法の提案者らによる評価では，medium に用いる難読化では理解に要する時間が平均 161 秒から 652 秒に，difficult に用いる難読化手法では 2921 秒に増加すると報告されている．

暗算タスク: 暗算タスクは，一桁どうしの掛け算（簡単な暗算）と，二桁どうしの掛け算（難しい暗算）からなる．Zago らの研究 [49] によると，簡単な暗算は「九九」のように計算結果を長期記憶から想起するだけで，実際の計算は行われない．一方で，難しい暗算においては，長期記憶の想起に加え，実際の計算を行うために一時的な記憶領域が利用され，簡単な暗算に比べて前頭前野が活発化する．

暗算問題は難易度ごとにそれぞれ 150 問用意し，A4 用紙に 50 問ずつ印刷して回答してもらった．問題は計算機上でランダムに生成したが，Zago らの研究と同

```
01: int func(int **A, int N, int M)
02: {
03:     int i,j;
04:     int x;
05:
06:     x = A[0][0];
07:     i = 0;
08:     if(i<N){
09:         j = 0;
10:         if(j<M){
11:             for(;;){
12:                 if(A[i][j]>x) x = A[i][j];
13:                 i++;
14:                 if(i>=M){
15:                     for(; i<N; i++){
16:                         if(A[i][j]>x) x = A[i][j];
17:                     }
18:                     j++;
19:                     if(j>=M) break;
20:                     i = j;
21:                     continue;
22:                 }
23:                 if(A[j][i]>x) x = A[j][i];
24:                 if(i>=N){
25:                     for(; i<M; i++){
26:                         if(A[i][j]>x) x = A[j][i];
27:                     }
28:                     j++;
29:                     if(j>=N) break;
30:                     i = j;
31:                 }
32:             }
32:         }
33:     }
34:     return x;
35: }
```

図 3.3 図 3.2 に難読化を施した例

じく，同じ数字が連続して使われないように注意したほか，一桁どうしの掛け算では 1 が出現しないように調整している．

なお，暗算に関する既存研究においては，6 以上の数が出ないようにしているものがあるが，本実験ではそのような制約を置いていない．これは，日本においては義務教育課程で九九をすべて暗記する教育がなされていることを前提にしており，妥当性確保のため次節のアンケートにおいて九九を暗記しているかを被験者に尋ねた．

2.3 アンケート

各タスクの終了後，被験者に対して質問用紙を用いたアンケートを実施する．アンケートでは，提示された課題それぞれの難しさを「簡単」から「難しい」まで 5 段階のリッカート尺度で回答してもらう．また，日頃プログラムを読む頻度や，プログラミング経験，利き腕，九九を覚えているか尋ねたほか，アンケートには自由記述欄を用意した．

なお，アンケートの記入時，被験者が課題の内容を忘れた場合を考慮し，タスクで用いた問題用紙を実施順に並べて被験者に提示する．

2.4 実験環境

実験には日立製ウェアラブル光トポグラフィ WOT-220 を用いる．本装置は 22 チャンネルのレーザ発光／受光装置を備えており，前頭前野の脳血流量を測ることが可能である．また，他の NIRS 装置と比較して軽く，帽子状の形をしているため装着が容易，計測器との通信を無線で行うため被験者の体動を制限しないという特徴を持っている．

図 3.4 に装置と装着時の外観を示す．

実験は被験者 1 名と実験者 1 名のみが居る静かな部屋において，椅子に座った状態で行われた．姿勢の変化によるノイズを防ぐため，ソースコードを印刷した紙と解答用紙を書見台に置き，楽な姿勢で読めるよう椅子の位置や高さを調整し

図 3.4 装置と装着時の外観

てもらう．また，頭部の動きによる血流変化を防ぐため，事前に頭を激しく動かさないよう被験者に伝える．

2.5 計測値の整形

本研究では，NIRS で計測した値に対してノイズ除去と標準化を適用して求められる標準化 oxy-Hb を用いて，被験者ごとのタスク種類ごとにおける脳活動量の差を議論する．

装置に由来するノイズの除去:装置由来のノイズとして，a) 測定値に恒常的に含まれるバイアスと，b) 工学的特性によるスパイクノイズがあげられる．
このうち前者は，測定前のキャリブレーションによって取り除く．後者のスパイクノイズは，標準移動平均 (SMA; Simple Moving Average) をとることで除去する．本実験における最も短いタスク (2 分) に対して十分短い 4 秒の SMA を計算する．機器の測定周波数は 5[Hz] なので，oxy-Hb と deoxy-Hb の計測値に対し，式 (1) と式 (2) を用いて n=20 での信号平滑化を行う．

$$oxy_{SMA}(t) = \frac{1}{n}\sum_{k=1}^{n} oxyHb(t) \tag{3.1}$$

$$deoxy_{SMA}(t) = \frac{1}{n}\sum_{k=1}^{n} deoxyHb(t) \tag{3.2}$$

生体に由来するノイズの除去:生体由来のノイズとして，心拍，呼吸，血圧や体調に依存する変化があげられる．このうち心拍や呼吸などによるノイズは短周期な一定の変動であるため SMA によって除去される．
一方，血圧や体調の変化などを含むタスク実施時間より長周期なノイズについては，原信号から指数移動平均 (EMA; Exponential Moving Average) を減算する手法 [47] を用いる．本実験における最も長いタスク (10 分) の 2 倍にあたる 20 分の

EMA を計算する．機器の測定周波数は 5[Hz] なので，oxy_{SMA} と $deoxy_{SMA}$ に対し，式 (3) と式 (4) を用いて n=6000 での信号平滑化を行う．

$$\begin{aligned} oxy_{MA}(t) = &\frac{1}{n} oxy_{SMA}(t) \\ &+ (1 - \frac{1}{n}) oxy_{MA}(t-1) \end{aligned} \tag{3.3}$$

$$\begin{aligned} deoxy_{MA}(t) = &\frac{1}{n} deoxy_{SMA} \\ &+ (1 - \frac{1}{n}) deoxy_{MA}(t-1) \end{aligned} \tag{3.4}$$

長期計測に由来するノイズの除去:NIRS の測定値は頭部や体の動きでセンサー位置が変化することでノイズが加算される．本実験では，問題用紙の読解や解答用紙への記入に合わせて頭部が上下する可能性がある．そこで，Cui らによる CBSI 法 [9] を採用し，頭部・身体動作に起因するノイズの除去を試みる．

CBSI 法は，脳活動が起きた際の脳血流における oxy-Hb と deoxy-Hb の変動特性を利用したノイズ低減手法である．同手法を用いることで，体動をモーションセンサなどで取得せずとも，NIRS 原信号のみを用いてノイズ除去を行うことができる．たとえば頭を頷くように前に傾けた場合，前頭前野の oxy-Hb と deoxy-Hb はともに重力の影響を受けて増大するが，脳活動が起きた際はまず deoxy-Hb が減少し，後に oxy-Hb を多く含む血流量の増大が起きる．CBSI 法ではこのような oxy-Hb と deoxy-Hb の関係を利用し，統計的にノイズ除去を行う．

分析に際しては，SMA と EMA による信号平滑化を行った oxy-Hb と deoxy-Hb(式中 oxy_{MA}，$deoxy_{MA}$) に対して，次の式 (5) を適用し，真の oxy-Hb と deoxy-Hb(式中 oxy_{CBSI}，$deoxy_{CBSI}$) を求める．

$$
\begin{aligned}
\alpha &= \frac{sd(oxy_{MA}(t))}{sd(deoxy_{MA}(t))} \\
oxy_{CBSI}(t) &= \frac{1}{2}(oxy_{MA}(t) - \alpha deoxy_{MA}(t)) \\
deoxy_{CBSI}(t) &= -\frac{1}{\alpha} oxy_{CBSI}(t)
\end{aligned}
\tag{3.5}
$$

計測値の標準化: NIRS による計測値は，計測開始時からの脳活動の相対的な変化量を表すため被験者間の比較が難しい．
本稿ではある時点 t における脳活動の強さを，oxy_{CBSI} 値の平均と分散を用いて正規化した activation(次式) の形式で表現することで被験者間の比較を行う．

$$
activation(t) = \frac{oxy_{CBSI}(t) - mean(oxy_{CBSI})}{sd(oxy_{CBSI})} \tag{3.6}
$$

$activation(t)$ は，時刻 t における脳活動の強さを表す．ここでの $oxy_{CBSI}(t)$ の平均値と標準偏差は被験者ごとに計算されるため，activation は被験者ごとに平均 0, 分散 1 の形に正規化された数値である．

算出された activation の例: ノイズ除去および計測値の標準化を行った後に求まる activation の時系列変化の例を図 3.5 に示す．結果の節においては，この activation のとる値を，タスク区間ごとに集計し，その分布や値の差を箱ひげ図や代表値を用いて議論する．

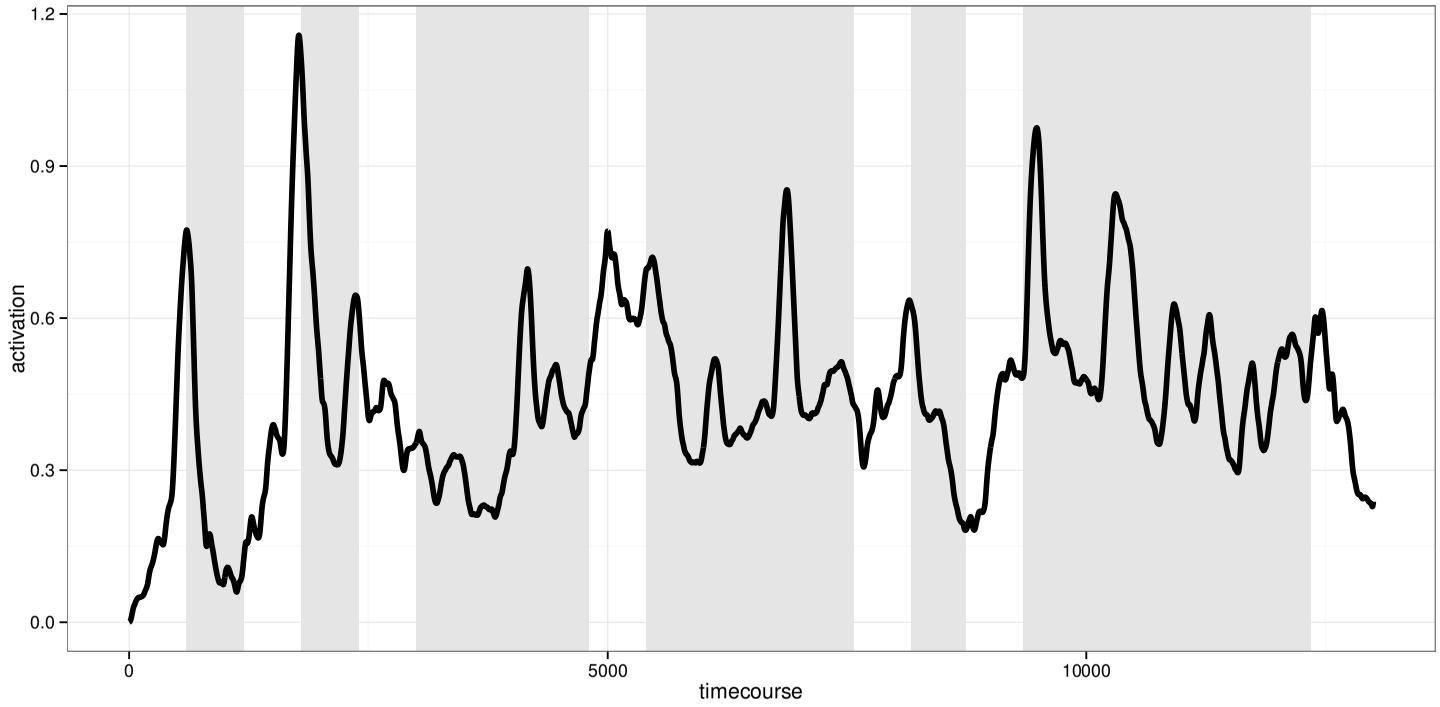

図 3.5 activation の時系列変化の例

3　結果

20 名の被験者のうち，3 名の被験者は実験中に装置が大幅にずれたため結果から除外した．

3.1　各タスクごとの脳活動値

本実験は，1 タスクの時間が最長で 10 分と長く，脳が活性化するタイミングが被験者ごと，タスクごとに異なることから，平均値や波形による議論が難しい．そのため，RQ1 の検証にあたって，実施したタスク難易度ごとの activation の分布を表した箱ひげ図を図 3.6 に示し，議論を行う．

図からは，17 人中 16 人について easy より difficult の中央値が高いことが読み取れる．また，17 人中 15 人について easy より medium の中央値が高いことも読み取れる．17 人中 16 人，ならびに 17 人中 15 人という結果について，正確二項検定を実施したところ，いずれも $p < 0.01$ で有意な偏りであった．このことは，「プログラム理解に困難が生じている状態を，理解時の脳血流計測によって計測できる」という仮説を支持する．

> RQ1 への回答：理解難易度の異なる様々なプログラムを読んだ際，被験者に生じる困難を，脳血流計測によって判別できる．

ただし，本研究で採用した難読化手法 [31] で生成されたプログラムの難易度が，現実世界におけるプログラムの難易度を反映しているかどうかは不明であり，今後の検討課題である．

また，同様に平均値を見た場合も，17 人中 16 人について easy より difficult が，17 人全員について easy より medium が高かった．一方，medium と difficult の平均値を比較すると，difficult の方が高かったものが 10 名，medium の方が高かったものが 7 名であった．

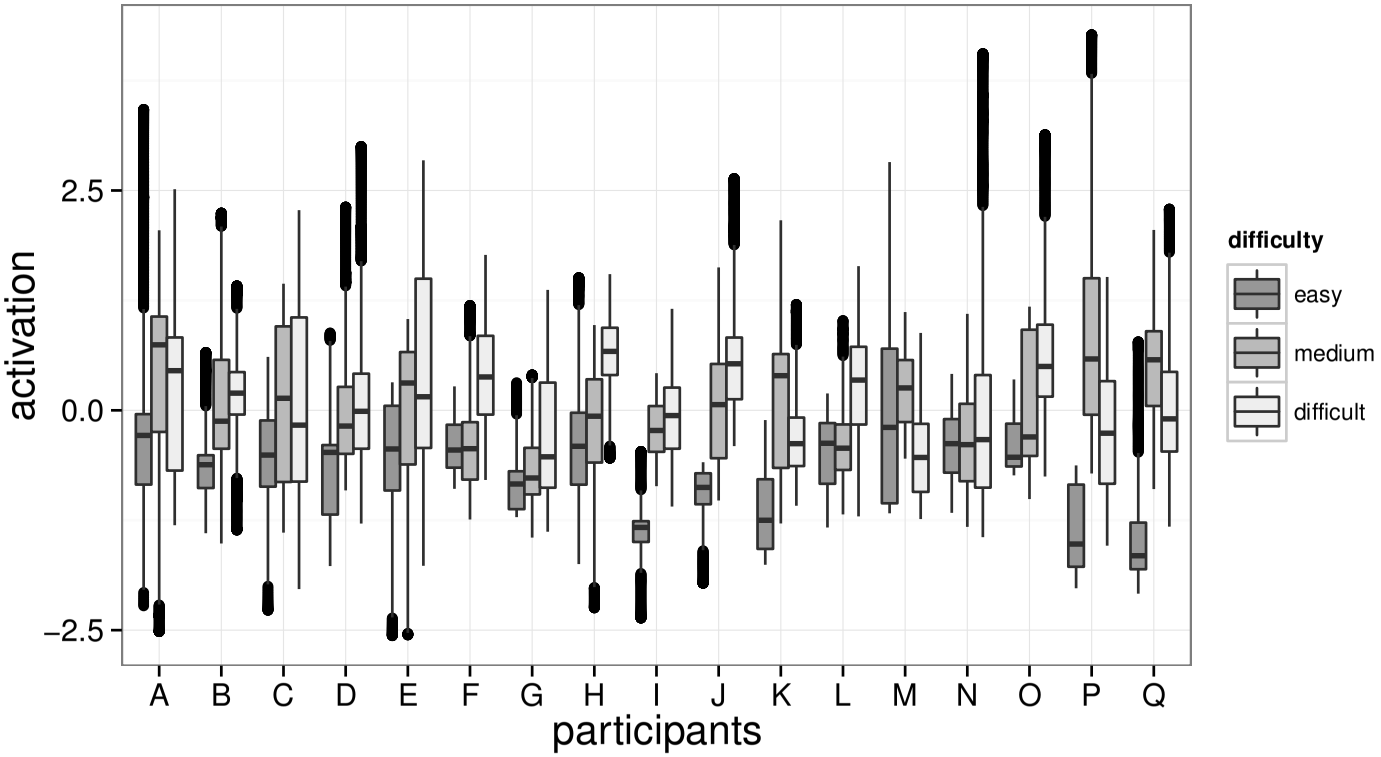

図 3.6 各被験者の難易度別 activation

3.2　被験者に実施したアンケートの結果

被験者に実施したアンケートの結果得られた，被験者が各課題に対して感じた主観的な難易度 (5 段階, リッカート尺度) を表 3.4 に示す．参考として，主観的難度にあわせて難易度別 activation の中央値を併記しておく．表からは，被験者が高難易度タスクほど難しく感じたと回答していることが読み取れる．

難易度別の 3 群間に有意な差があるかどうかをクラスカル＝ウォリス検定によって確かめた結果，群間の差は $p < 0.01$ で有意であった．

3.3　暗算タスクの脳活動値

NIRS 機器の計測値や，その加工 (activation の算出) についての妥当性を調査すべく，関連研究 [49] において，高難度時に脳活動が活発化すると報告されている暗算課題を再現し，その差を調べた．

その結果，17 人中 14 人について，簡単な暗算課題実施中より，難しい暗算課題実施中の activation が高いことが確認された．この結果について，正確二項検定を実施したところ，$p < 0.05$ で有意であった．

本結果は，17 名中 3 名の例外を含むものの，NIRS 機器による activation の計測が一定の妥当性を持つことを支持するものと考えられる．

表 3.4 各課題に対する主観的難易度と activation の中央値

被験者		a	b	c	d	e	f	g	h	i
主観評価	easy	3	1	2	1	1	1	1	1	1
	medium	2	2	3	2	4	2	4	2	4
	difficult	5	5	5	4	3	5	3	4	5
activation	easy	-0.3	-0.8	-0.5	-1.3	-0.5	-0.4	-0.4	-0.9	-0.4
	medium	0.7	-0.8	-0.2	-0.2	-0.4	0.3	-0.1	0.1	-0.4
	difficult	0.5	-0.5	0.0	-0.1	0.4	0.2	0.7	0.5	0.3

被験者		j	k	l	m	n	o	p	q	mean
主観評価	easy	2	1	1	1	1	2	2	1	1.35
	medium	2	2	5	3	2	5	4	3	3.00
	difficult	5	4	4	4	4	4	4	5	4.29
activation	easy	-1.3	-0.2	-0.4	-0.5	-1.5	-1.7	-0.6	-0.5	-
	medium	0.4	0.3	-0.4	-0.3	0.6	0.6	-0.1	0.1	-
	difficult	-0.4	-0.5	-0.3	0.5	-0.3	-0.1	0.2	-0.3	-

4　考察

4.1　異なる難易度における脳血流の分析

設定したタスク難易度と脳活動量の関係を分析するために，アンケートと同じく難易度別の 3 群間に有意な差がみられるかクラスカル＝ウォリス検定によって確かめた結果，群間の差は $p < 0.01$ で有意であった．このことは，脳血流の分布から難易度の差を判別できることを示す．その一方で，medium と difficult の二群間の中央値の差について，ウィルコクソンの符号付き順位和検定を実施したところ，有意な差は見られなかった．

2.2 節で述べたように medium の条件は「制御構造に対する難読化が施されている」という点が difficult と重複しており，これらの間で見られる差が小さくなったものと考えられる．すなわち，制御フローに対する難読化が，プログラムの処理内容に対するトップダウンな理解を妨げることで，結果的に複雑度に関わらず被験者は「記述内容を一行ずつ解釈する」ことになり，単位時間あたりに要する思考の質に大きな差があらわれなかった可能性がある．

この結果は，プログラム理解中の開発者に対する脳血流計測では，ある一定以上負荷が高くなると，それ以上の負荷は判別できないことを示す．

RQ2 への回答：プログラムの構造を簡単に理解できたかどうかを判別できる．ただし，トップダウンな理解が難しい課題における，極度の負荷とそれ以下の負荷の判別はできない．

4.2　被験者の主観的難易度と脳活動量の関係

本研究では理解困難状態を理解の難航，すなわち同様の機能を理解するのに必要な時間が増大する状態として定義している．これは，理解の難航時に速やかな解決策の提示・休息の指示といった介入を行うことでプログラム理解作業・教育の効率化を目指す本手法の立場からくる定義である．

理解の困難さを捉える他の観点として，たとえば被験者の感じる主観的な困難さがある．実際に既存研究のいくらかは被験者にかかる精神的な負荷や感情の悪化に言及している [59][14]．そこで，本手法で計測された activation が理解の難航を反映するものなのか，それとも主観的な困難さを反映するものなのかを明らかにするため，難易度の主観的評価と activation の関係について調べる．

図 3.7 は被験者の感じた主観的難易度 (5 段階) を横軸，activation の中央値を縦軸にとった散布図である．全体として課題の難易度が高くなるほど主観的評価と activation の値が高くなる比例関係が見られる．activation と主観的難易度の関係について，Spearman の相関係数をとったところ，相関係数 0.42　($p < 0.01$) で有意な相関がみられた．

easy に着目すると，主観的難易度と activation が一致して共に低く，他の難易度と分かれていることがわかる．一方で，主観的難易度が 2 と回答された課題に着目すると，easy の activation が低く，medium が高くなるなど明瞭に分かれている．すなわち，本手法によって計測される理解困難状態は，被験者の主観的な難易度評価と関連する一方，事前に定めたタスク難易度の別，すなわち理解に要する時間を反映していることを示唆する．これについては，次節でより詳細に検証する．

また，最も activation が活発だった時の値，すなわち各タスク実施中の activation の最大値と主観的難易度の関係を示した散布図を図 3.8 に示す．最大値を見ると，medium より difficult のほうが activation の値が高い場合が多い．Spearman の相関係数は 0.59　($p < 0.01$)　と中央値を用いた場合よりも強い，有意な相関がみられた．

このことは，もっとも難易度の高い部分を読解する必要がある時点においては，difficult のほうが medium より高い activation を示す可能性があることを示唆する．ただし，最大値については，easy でも高い値が見られ，全体的にばらつきが大きいため，activation の最大値が課題遂行中の困難を示すか確かめるには至らなかった．

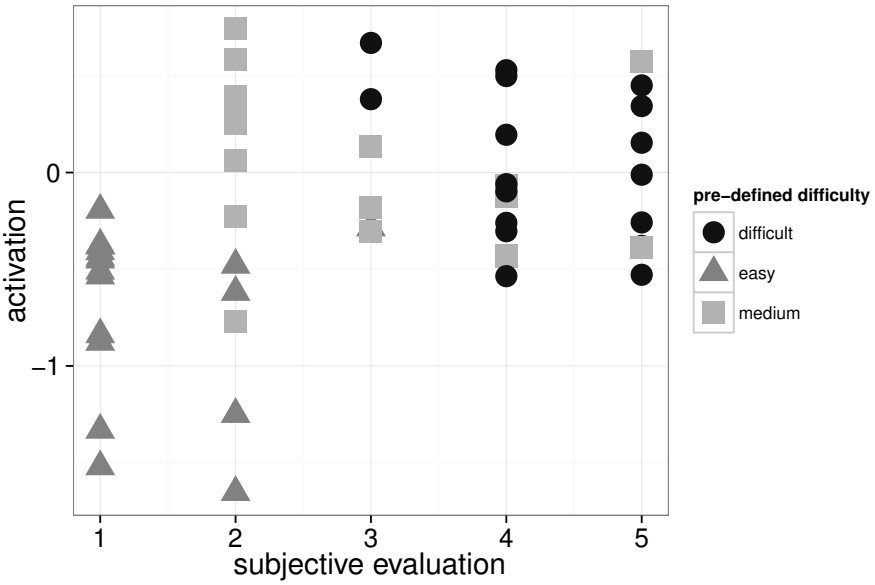

図 3.7 主観的難易度と activation の中央値

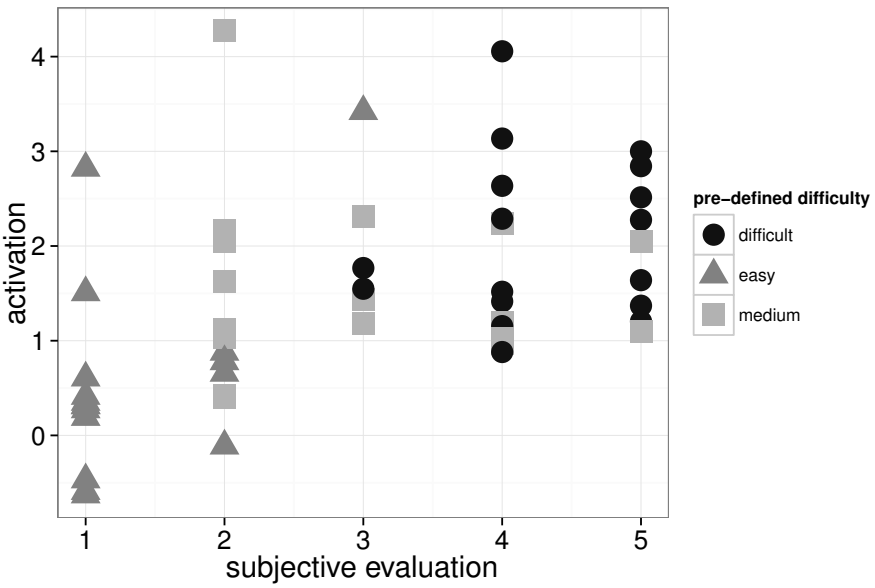

図 3.8 主観的難易度と activation の最大値

RQ3 への回答：activation の中央値は主観的難易度と有意な相関があり，特に 5 段階評価で 2 を下回る場合，明瞭に区別できる．最大値を見ると，さらに相関が強いが，ばらつきが大きい．

4.3 理解にかかった時間と脳活動量の関係

前節での分析の結果，activation は被験者の感じる主観的難易度よりも，タスク難易度の別，すなわち理解にかかる時間をよく反映している可能性があることがわかった．そこで，前節と同じく activation の中央値と最大値に注目し，それぞれタスク完了に要した時間との関係を調査する．

図 3.9 および 3.10 はそれぞれタスク完了時間を横軸に，activation の中央値および最大値を縦軸に取ったものである．実験ではタスクの制限時間を 10 分 (600 秒) に設定したため，完全な図であるとは言えず読み取りに注意を要するが，タスク完了時間と activation の間には正の相関がみられる．activation と主観的難易度の関係について，Spearman の相関係数をとったところ，中央値では相関係数 0.57　($p < 0.01$)，最大値では相関係数 0.60 ($p < 0.01$) の有意な相関がみられた．

各難易度に注目すると，easy,medium においては同一難易度内でもタスク完了時間の違いによる activation の差が見られる．本結果は同一難易度内で主観難度が異なっても activation に目立つ差が見られなかった前節の結果とは対照的であり，activation が理解にかかる時間の長さをよりよく反映することを示唆している．このことから，本手法は被験者が感じる主観難度によらず，理解の難航が発生しているかどうかを特定できる可能性があると言える．

4.4 逆傾向を示す被験者

難読化された medium，difficult ではなく，easy のほうが activation が高くなる被験者が一定数 (17 名中 1〜2 名) 見られた．このような逆傾向を示す被験者の例

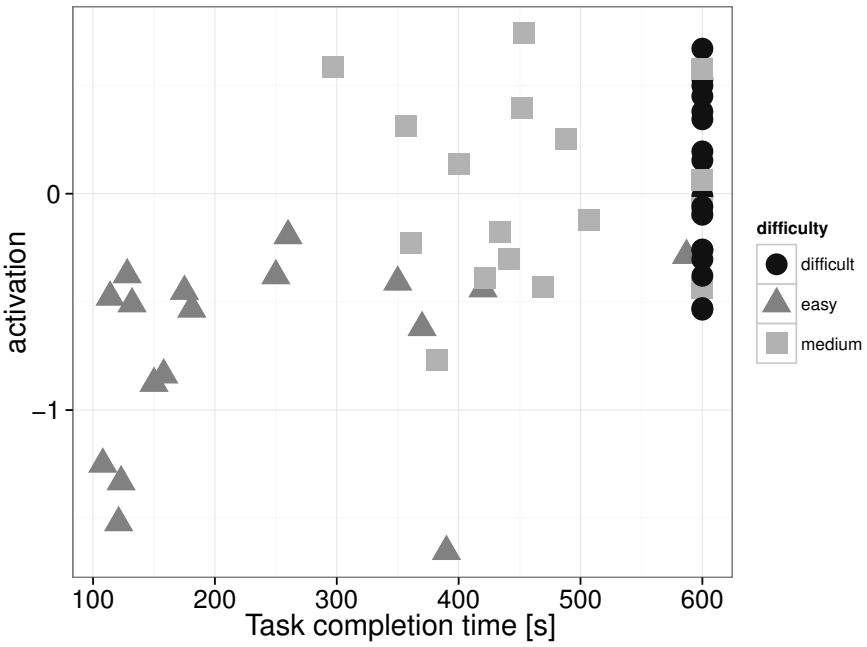

図 3.9 タスク完了時間と activation の中央値

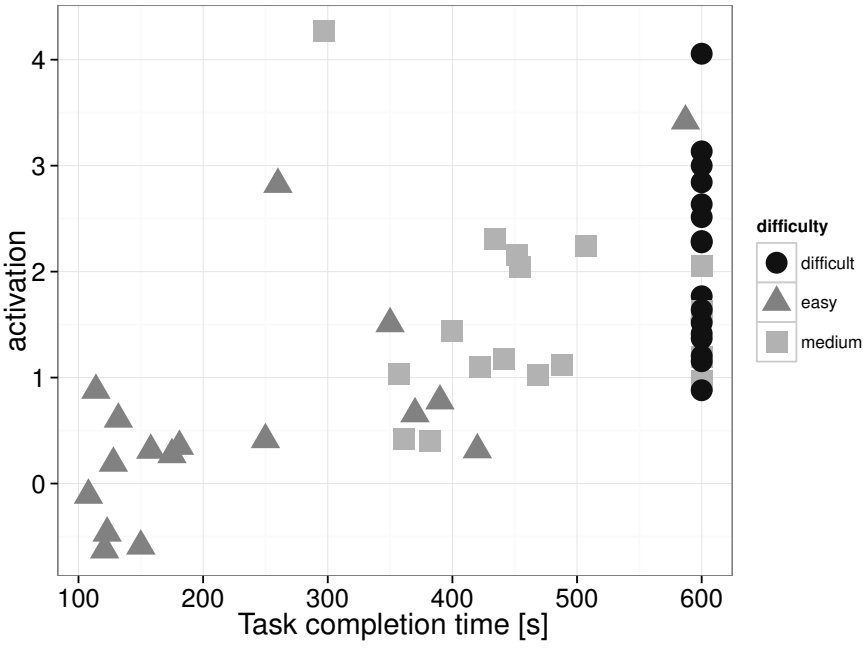

図 3.10 タスク完了時間と activation の最大値

は，過去の NIRS に関する研究でも 10 人に 1 人程度の割合で含まれるとする指摘があるが，いずれも原因は特定されていない [57] [20].

主観評価においては一例を除いて easy は他のプログラムより簡単であると評価されており，その一例も activation で逆傾向を示した被験者と一致しない．また，NIRS の計測値の根拠となる脳血流の酸化傾向と脳活動の関連は，神経科学分野において繰り返し検証されてきたものであり，その妥当性は非常に高いと考えられる．そのため，実際に被験者が困難を感じていなかった，あるいは困難を感じたが脳血流の酸化傾向が変化しなかったとは考えにくく，装置の特性，あるいはその装着時の接触不良などの影響を受けている可能性が高い．

いずれにせよ，実際に装置を用いた計測で逆傾向が見られたことは計測値に基づく困難さ判定の限界を示すと考えられる．ただし，本手法は企業・学校等におけるプログラミング学習段階の人物に対する支援を目指すものであり，理解困難状態にある開発者を発見したのちは，実際に人間の監督者が教育を行うなどの形で対処することが予想される．そのため，判別の精度が必ずしも 100%である必要はなく，作業を阻害せずに理解困難度を定量化できる本手法の有効性が損なわれることはない．

また，過去の報告では確認されていないが，装置の特性ではなく，人物の特性として常に逆の傾向を示す傾向がある場合も考えられる．仮にこのような人物が存在した場合，不要な教育を何度も繰り返すことになる可能性があるため，装置の適用にあたっては逆傾向が存在する可能性を十分考慮したうえで適用する必要がある．

5　まとめ

本章では，脳活動に基づいてリアルタイムに開発者にとっての理解困難度を計測する手法を提案し，その有効性を実験によって検証した．

実験では，被験者がプログラム理解に確実に支障を来すような難読プログラムを作成し，そのメンタルシミュレーションを行ってもらうことで，人工的に理解困難度の高い状況を作り出し，その際の前頭前野の脳活動が仮説通り上昇するか

や，脳活動の値と被験者の感じる主観的な難易度の関係を分析した．

20 名を対象とした被験者実験の結果，脳活動に基づく理解困難度のリアルタイム計測手法は，開発者がプログラムを簡単に理解できない状態にあるかどうかを判別できることを明らかにした．ただし，トップダウンな理解が難しい課題における，極度の負荷とそれ以下の負荷の判別は難しいという制約も明らかになった．被験者の感じる主観的な難易度との関係として，脳活動量の中央値は主観的難易度と有意な相関があり，特に 5 段階評価で 2 を下回る場合，明瞭に区別できることがわかった．また，最大値を見ると，さらに相関が強く 0.59 となるが，ばらつきが大きくなることがわかった．

将来の発展的課題として，前頭前野以外の脳部位を対象としたり，より精度のよい NIRS 装置を用いることによって，結果の制約となった測定粒度を改善することが挙げられる．また，課題の難易度だけでなく，どのような原因で困難さが生じているかをある程度判別できるような手法について検討することも興味深い．なお，生体情報計測に基づく困難さの種類判別については，続く 4 章において，人間の認知機能とプログラム理解の関係に注目した先駆的調査を行う．

第4章　脳波計測によるプログラム理解時の困難さの種類の判別

1　あらまし

3章ではプログラム理解を構成する認知機能のうち特に記憶に着目し，その酷使がNIRS装置によってある程度捉えられること，またそれが人間の感じる主観的な困難さとも関連することを示した．

一方で，既存研究においてはより安価な脳波計測装置を用い，周波数解析結果を利用してプログラム理解の困難度を測定する試みが行われてきた [52][14]．既存研究ではいずれも，困難度の二値判別における精度が70%を下回る，あるいは課題プログラムに含まれる困難さの種類に言及していない，といった課題があった．

特にプログラム理解中のどの知的活動が，脳波のどの成分と対応して困難度を測定できるかについての議論は著者の知る限り行われておらず，特定の困難さは測定できるが，別の困難さは測定できないといった可能性も十分考えられる．そこで我々は既存研究で用いられてきた簡易かつ商用の脳波計測装置を用い，記憶，意味的統合，文法的統合など複数の認知機能を個別に要求する実験的なプログラムを用いて，3章と同様の理解困難度計測がどの程度可能かを調査する．

実験では作業記憶，意味的統合，文法的統合などの認知機能を強く要求すると考えられるプログラムを読む課題 (target 課題) と，target 課題に類似するが要求しないと考えられるプログラムを読む課題 (control 課題) を用意し（2種× 4認知機能=8種），被験者にそれらの理解を試みてもらう．target 課題と control 課題は認知心理学・神経科学の分野で特定の認知機能に関係するとの報告がある実験課題を参考に，理解を進める途中で知的活動が誘発されるよう調整されている．

分析においては，各課題の target-control 間で，脳波の原信号に含まれる代表的な周波数成分の大きさに差があるかを調べる．具体的には，課題ごとに α 帯，β 帯，θ 帯，γ 帯のパワースペクトル密度を計算し，その差を比較することで，各認知機能に対する負荷の差を信号として抽出できるかどうかを確認する．

2　実験

脳波計測装置によって開発者がプログラム理解に支障を来している際の困難の種類を特定できるかどうかを調査するため，被験者実験を行った．奈良先端科学技術大学院大学，ならびに奈良工業高等専門学校の学生 14 名が被験者として実験に参加した．被験者は全員男性で，年齢は 21 歳から 25 歳，Java プログラミングの経験を有している．

2.1　手順

表 4.1 に実験手順を示す．実験において被験者は，脳波計測装置を装着した状態で 8 件のプログラム理解（出力値の回答）を試みる．8 件中 4 件のプログラムは，認知心理学分野で特定の認知機能に強い負荷がかかるとされている実験課題を模したプログラム (target タスク) である．残り 4 件はそれらのプログラムによく似た機能を実現するが，認知機能に強い負荷がかからないようなプログラム (control タスク) である．また，全ての被験者が全課題（8 件）の読解を行う．順序効果を考慮し，提示順序をランダム化した上で，被験者間の被りがないように工夫する．

各理解タスクの制限時間は 5 分となっており，被験者がそれよりもはやくプログラムを理解できた場合は，即座に口頭で解答し，注視点の凝視タスクにうつる．制限時間を迎えた場合，その時点で可能ならば口頭で出力値を回答するよう画面指示 (5000ms) を出す．実験時間は，最大の場合（被験者がすべてのプログラムを制限時間内に理解できなかった場合）で 49 分 40 秒となる．なお，被験者に回答結果の正誤は伝えないものとした．

表 4.1 実験手順

手順	内容	時間	備考
1	実験内容の説明		
2	装置キャリブレーションの実施		
3	注視点の凝視	1 分	
4	実施タスク 1	5 分	問題を解き終われば，口頭で回答の後次の手順へ
5	注視点の凝視	1 分	
6	実施タスク 2	5 分	問題を解き終われば，口頭で回答の後次の手順へ
7	注視点の凝視	1 分	
8	実施タスク 3	5 分	問題を解き終われば，口頭で回答の後次の手順へ
9	注視点の凝視	1 分	
10	実施タスク 4	5 分	問題を解き終われば，口頭で回答の後次の手順へ
11	注視点の凝視	1 分	
12	実施タスク 5	5 分	問題を解き終われば，口頭で回答の後次の手順へ
13	注視点の凝視	1 分	
14	実施タスク 6	5 分	問題を解き終われば，口頭で回答の後次の手順へ
15	注視点の凝視	1 分	
16	実施タスク 7	5 分	問題を解き終われば，口頭で回答の後次の手順へ
17	注視点の凝視	1 分	
18	実施タスク 8	5 分	問題を解き終われば，口頭で回答の後次の手順へ
19	注視点の凝視	1 分	
20	装置の取り外しとアンケート		

2.2 タスク

タスクは Java 言語で書かれた一つのメソッドからなるプログラムである．タスク提示前の説明で，被験者には課題プログラムを読み，メソッドを実行した際の返り値を回答するように指示する．また本実験においては制御構造の複雑さを対象とした 3 章と異なり，多様な困難さを対象としていることから，実験指示においてプログラム理解手法の限定は行わなかった．

実験に先立って，プログラム理解中に必要とされる可能性が高い認知機能を洗い出すため，Siegmund らによる fMRI を用いた研究や，自然言語の理解に関する研究を調査した [42][44]．調査の結果，Working Memory, Attention, Language, Semantic Memory Retrieval, Problem Solving, Semantic Integration などの認知機能はプログラム理解と関連する可能性が高い．

これらの認知機能のうちいくつかは既に認知心理学・神経科学分野において「特定の認知機能を強く利用する実験課題 (target 課題)・利用しない実験課題 (control 課題)」を比較する実験が行われている．そこで本実験ではこうした先行研究の実験デザインを参考に，特定の認知機能を強く利用しなくては読めないプログラムと，そうでないプログラムを作成した．本節では Working Memory を酷使する二つのタスク（変動する数値記憶タスク，変数名とその意味・数値の記憶タスク）と，言語に関する操作である意味的統合タスク，文法的統合タスクのそれぞれについて説明する．

なお，特に前出の Working Memory を酷使する 2 タスクについては，模倣元の心理学実験課題について脳波の周波数特性解析によってある程度差分が抽出できるという既存研究があり，結果において比較を行う．残り二つのタスクについては，fMRI や MEG を利用した研究では活動部位の議論がされているが，脳波を用いた議論は著者の知る限り存在せず，より探索的な課題となっている．

A) 変動する数値の記憶タスク: Siegmund らは，プログラム理解に必須の認知プロセスのひとつとして，プログラム中に出てくる値や文を記憶するための Working Memory の操作を挙げた [42]．石黒ら，中村らも同様にプログラムの理解難度に Working Memory の参照回数が影響するという前提でモデルを構築している

[58][33]．Working Memory に記憶できる情報の量は有限であるため，配列の中身など，多くの値を記憶しなくてはならない場合は理解の混乱や失敗が予想される．

この能力が強く求められる場合に適切な支援ツールとして，デバッガの利用が挙げられる．また，適切に数値の推移をメモなどのかたちで記録することにも一定の効果があると考えられる．

target 課題を図 4.1 に示す．本課題では，過去に参照された数値の系列を参照する n-back タスクを模倣することで，プログラム理解中に Working Memory が強く刺激される状態を再現する．n-back は，認知心理学などの分野で作業記憶を強く利用する課題として知られており，つぎつぎと画面に提示されていく数字を見ながら,「n 個前に表示されていた数値」と一致するかどうかを検査する課題である．本課題は単に特定の法則に従ってに配列の数値を足し合わせていくプログラムであるが，返り値の変動を追跡・回答ためには，3 ループ前に参照された値を思い出す必要があり，3-back の模倣となっている．

control 課題はランダムに配列の数値を足し合わせていくプログラムであるが，読解中に過去のループに遡った数値の想起を必要としないため，target 課題に比べて Working Memory への負荷は低いと考えられる．

target 課題実施中に増大が期待される脳波の周波数成分として，θ 波が挙げられる．特に Frontal Midline Theta(Fm θ) とも呼ばれる前頭部の θ 波は，難易度の高い n-back タスクの実施中に観測されることでよく知られている [16][18]．

B) 変数名やその意味・数値の記憶タスク: プログラム理解において，変数名や関数名の定義・用途を理解し，記憶することはプログラム理解において重要な手がかりとされる [7]．ソフトウェア工学の研究や書籍などにおいては，有意味かつ他の項目と差別化しやすい識別子を用いることがが推奨されている [2][26]．すなわち，単語ではない記号や，用途とかい離した不適切な名前の識別子は開発者のプログラム理解を妨げる要因となり得る．

この能力が強く求められる場合に適切な対処として，変数名・関数名の変更やコメントの追加が挙げられる．

target 課題を図 4.2 に示す．本課題では，全ての変数名においてランダムに作

```
01: public static int function(){
02:     int data1[] = {1,2,3,3,2,6,6,9,5,5};
03:     int data2[] = {9,0,1,7,1,6,8,1,8,8};
04:     int a[] = {0,0,0,0,0,0,0,0,0,0};
05:     int i,b,c;
06:
07:     for(i=0; i<10; i++){
08:         b = data1[i];
09:         c = (i+10-3)%10;
10:         a[i] = a[c] + data2[b];
11:     }
12:     return a[9];
13: }
```

図 4.1 変動する数値の記憶 (target タスク)

成した意味を持たない語を利用することで，プログラム理解中に新たな語の記憶を要する状態を再現する．control 課題では，通常の意味を持った変数名が利用されており，プログラムの理解において新たな単語と定義の結びつきを記憶しなおす必要が少ない．

本課題に対応する心理学分野での研究として，特定の単語の記憶課題や，単語と意味，単語と数値といった結びつきの記憶（対連合学習）課題について調べたものがある．これらの研究でも，単語そのものや，単語と他の情報の関連を記憶する際には，記憶するべき語，あるいは数値の有意味度が記憶の容易さに影響することが知られている [15][39]．

本課題においても，新たな学習のために作業記憶を酷使する target 課題においては変動する数値の記憶タスクと同じく Frontal Midline Theta の増大が確認できる可能性がある．加えて，対連合学習課題の実施中に γ 波が増大するという報告 [17] や，単語学習課題において一定時間後に単語を記憶し続けられていた被験者でのみ β 波が増大したという報告 [19] がある．

```
01: public static int function(){
02:     String chalph = "wuxwwxxu";
03:     int dromph = 0;
04:     char wroggy = '?';
05:     int valalm = chalph.length();
06:     for(int sheyvyl = 0; sheyvyl < valalm-1; sheyvyl++){
07:         if(chalph.charAt(sheyvyl) == wroggy){
08:             dromph += 1;
09:         }
10:         wroggy = chalph.charAt(sheyvyl);
11:     }
12:     return dromph;
13: }
```

図 4.2 変数名と値の関連付け (target タスク)

C) 意味的統合タスク: Siegmund らはまた，プログラム理解中に，複数の文や文節からより抽象的な意味を見出す”Semantic Integration”に関連する脳部位が活発化したと報告している [42]．あまりにも抽象度の低いプログラムは，関数名やコメントなどで抽象的な意味が最初から付与されたプログラムに比べて，理解が困難であると考えられる．

この能力が強く求められるプログラムに対する適切な対処として，適切な関数名を与えることや，メソッド抽出リファクタリング等によってプログラムの可読性を高めることが挙げられる．

自然言語理解の分野における既存研究では，特定のパラグラフを読解する際，タイトルの有無がその読解難度と強く関係することに着目し，意図的に Semantic Integration を行わせる課題が提案されている [44]．この実験設定を参考に，図 4.3 に示す target 課題は，関数名として適切なアルゴリズム名が与えられないプログラムの読解によって，Semantic Integration が必要とされる状況を誘発する．control 課題では，適切なアルゴリズム名がつけられたプログラムが提示され，負荷課題に比べて Semantic Integration への負荷は低い．

```
01: public static int function(){
02:     int data[] = {6,0,1,5,8,9,0,9,5,3};
03:     int a, b, i, j;
04:     a = 4;
05:     while(a>0){
06:         for(i=0; i<10; i++){
07:             j = i;
08:             b = data[i];
09:             while( (j>=a) && (data[j-a]>b) ){
10:                 data[j] = data[j-a];
11:                 j = j-a;
12:             }
13:             data[j] = b;
14:         }
15:         if(a/2 != 0)
16:             a = a/2;
17:         else
18:             a = 0;
19:     }
20:     return data[7];
21: }
```

図 4.3 意味的統合 (target タスク)

D) 文法的統合タスク: 自然言語文の理解では，先ほど述べたような関係節に関する文法などが比較用のタスクとして取り上げられている．Meltzer らは，英語や他の多くの言語では一般的に動作主が被動主より先行する強いバイアスがあると述べ，文法的に正しくとも，被動主が動作主に先行すれば文法的複雑性が増すと述べている．自然言語文の文法的複雑性をそのままプログラムに置き換えるのは難しいが，同様のバイアスが考えられる例として，構造化プログラミング（パラダイム）における反復の本来の意図に反する利用がある．

この能力が強く求められるプログラムに対する適切な対処として，より一般的

```
01: public static double function(){
02:     int data[] = {4,2,1,1,1,5,9,6,1,6};
03:     double a = 0.0;
04:     int b = 20;
05:
06:     for(int i=0; i<b;){
07:         b--;
08:         a = (a*i+data[i])/++i;
09:     }
10:
11:     return a;
12: }
```

図 4.4 文法的統合 (target タスク)

な文法や計算式への置き換えや，ソースコードに対するコメントの追加，メソッド抽出リファクタリングが挙げられる．

ここでは，文法的に誤りではないが可読性が低い，一般的でないとされるような文法を用いたプログラムを target 課題として設定し，そうでないものを control 課題とする．target 課題は，ループカウンタが更新されず，平均値を求める処理が総和/全数ではなく逐次処理になっており，ループの条件に使われる変数 w が条件によって加算されていく．

2.3 実験環境

実験は，実験実施者 1 名または 2 名と，被験者のみが居る静かな部屋において，椅子に座った状態で行った．課題は PsychoPy を用いて 13 インチのモニタに順次提示し，その際の EEG 信号を Emotiv EPOC+によって測定する．

Emotiv EPOC+は商用の簡易な脳波計測装置で，頭部表面の電位を計測できる．図 4.5 に，利用した実験装置と計測環境の例を示す．同装置は本研究の目的と同様の，簡易かつ安価な装置による脳情報計測の記録・分析用途で広く用いら

図 4.5 実験装置 (Emotiv EPOC+)

れている [52][30]．より一般的な脳波計測においては，国際 10-20 法などの配置図に基づいて頭部の各位置に電極を配置するが，EPOC+では予めその一部 (14 箇所) に電極が配置されるようなキャップ形状になっている．

実験中，被験者には着座姿勢を保ち，あまり激しい動きをしないように伝え，また，基本的に発話は課題の回答の際のみに行うよう指示した．

2.4 分析

脳波を計測し，そこから人間の認知活動を計測する主要な方法として，特定の認知活動が発生してから 1 秒以内に発生する脳波のパルスを測定する ERP(Event Related Potential) と，脳波の周波数特性を α 波，β 波など特定の周波数帯に分割し，その強さ（パワー）を見ることで認知活動の量やストレス状態などを測定する方法がよく知られている．

本実験の目的は，特定のタスクに取り組んでいる最中の認知機能の利用量を知ることで，困難さの種類が判別可能かを調査することであるため，後者の周波数成分を調べる方法を採用する．周波数成分の基礎的な計算方法として計測信号に対するフーリエ変換を利用する方法が知られている．この方法では，高速フーリエ変換によって原信号に含まれる特定の周波数帯域の成分が，パワースペクトルと

して表現される．本研究では同方法の発展形として知られる Welch の方法を用いる．Welch の方法では，単位周波数あたりのパワースペクトルの分布が PSD（パワースペクトル密度）の形式で算出される．分析時の周波数帯域は，慣習に従い，国際脳波学会によって定められた θ 帯 (4-7Hz)，α 帯 (8-13Hz)，β 帯 (14-30Hz)，γ 帯 (30Hz-) に注目して議論を行う．

フィルタリングによるノイズ除去，および FFT による周波数成分の計算は，EEG/MEG 信号の解析を行うための Python ライブラリ MNE を利用した．ノイズ除去では，計測対象周波数の下限の半分である 2Hz 以下の信号，並びに計測が行われた地区における交流電力の影響を除外するため，60Hz 以上の信号をバンドパスフィルタにより除去した．

また，一般に FFT による脳波データの時系列解析では，数秒の長さを持つスライド窓を適用して計測対象信号を区切ることが行われている．解析対象データのレコード数が 2 の n 乗であることを要求する FFT の性質上，本研究では 2 秒 (256 点) の長さを持つスライド窓 (ハン窓) を用いた．

結果の節においては，これらの方法で取得された各周波数帯域のパワー (単位:dB/Hz) を被験者ごとに z-score で正規化し，タスクの節で示した課題別に，target 課題と control 課題の間でその強さの比較を行う．特に，課題と対応する認知機能の利用と脳波の周波数解析についての既存研究がある場合は，そこで示された周波数帯に着目し，そうでない場合は特徴的な差が見られた指標について論じる．

3　結果と考察

14 名の被験者のうち，3 名分のデータはデータ収集の失敗，装置のずれに起因する計測値異常のため分析から除外した．本節では，残った 11 名分のデータについて，各タスクごとの結果と考察，ならびに EEG 装置による結果を 3 章における NIRS 装置による結果と比較した場合の考察を示す．

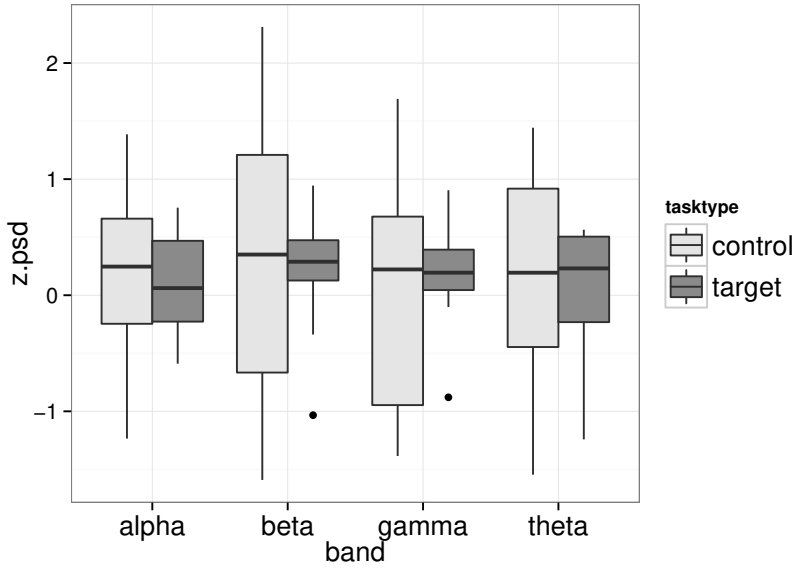

図 4.6 タスク A における課題タイプ/帯域別パワー

3.1 各タスクの結果と考察

A) 変動する数値の記憶タスク：タスク A の target/control 課題における帯域別パワーの分布を図 4.6 に示す．以降，結果の図はすべて，課題・周波数帯別に被験者の正規化後帯域別パワー（全計測チャンネルの平均）をプロットした箱ひげ図である．

タスク A の target-control 間には，どの周波数帯域においても大きな差が見られなかった．分布の差についてウィルコクソンの順位和検定を実施したところ，いずれの周波数帯域においても有意な差は確認できなかった．

この結果の原因として，target 課題で十分に困難が誘発されなかった可能性，あるいは control 課題でも作業記憶が利用されているため，脳波でその強度の差を検出できなかった可能性がある．タスク A の target 課題は control 課題に比べて被験者の誤答が多く，解答に要する平均時間も長いことから，前者のような target 課題における困難さの不足は考えにくい．そのため，脳波計測でプログラム理解中の作業記憶にかかる負荷の差を検出することは難しい可能性がある．

また，タスク A では追加の分析として，既存研究で言及のあった Fm θ の target-

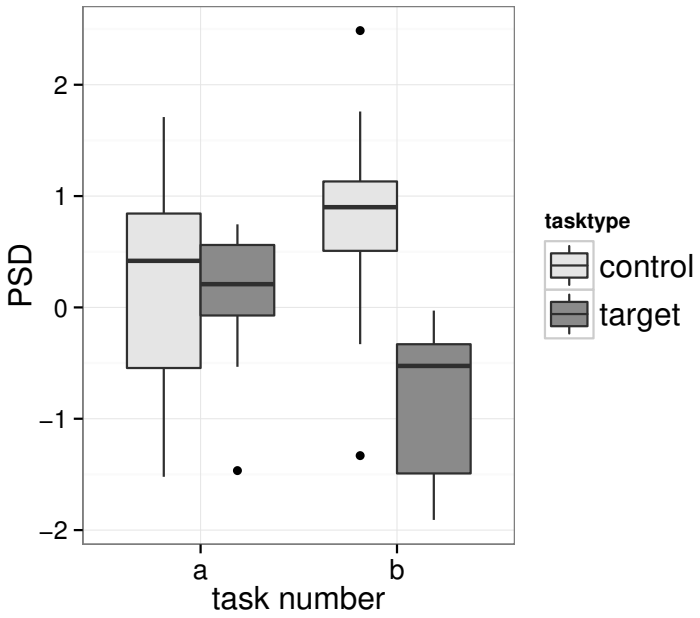

図 4.7 タスク A および B における Fm θ のパワー

control 間における比較も実施した．Fm θ の計測対象は，同様の機材を利用した McMahan らの研究を参考に決定した．追加の分析の結果，図 4.7 左のように，target 課題において Fm θ が減衰する傾向が見られたが，ここでも target-control 間には有意な差が見られなかった．

B) 変動する数値の記憶タスク： タスク B では，target 課題において α 帯と θ 帯のパワーに減衰が見られた．分布の差についてウィルコクソンの順位和検定を実施したところ，α 帯の差は有意 ($p < 0.01$) であると確認された．

関連研究から事前に予想された β 帯および γ 帯の変化は見られなかった．原因として，プログラム理解に必要な諸知的活動が，対連合学習の観点から見た困難さの差分に対して大きく，target-control 間の差を検出できなかったことが示唆される．

有意な差が見られた α 帯のパワーは一般に精神的にリラックスした状態にある場合や閉眼時に増大し，緊張しているときや，暗算などの知的活動時に減少すると言われている．この結果は，target 課題に含まれる未知の文字列が被験者に違和感を与え，さらに内容の意識的な記憶を必要とすることが，被験者の精神的な

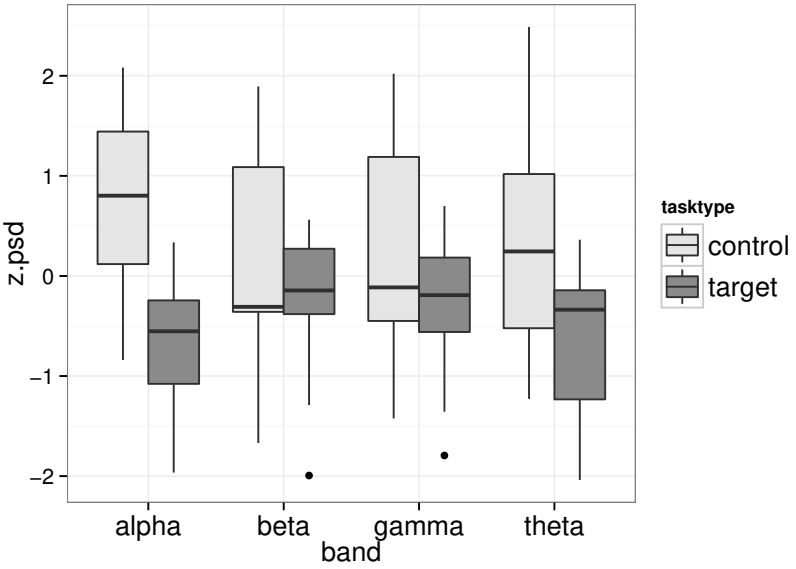

図 4.8 タスク B における課題タイプ/帯域別パワー

緊張状態につながった可能性を示唆する．

また，タスク B でもタスク A と同じく，作業記憶の操作に関連する Fm θ の target-control 間における比較を追加分析として実施した．追加の分析の結果，図 4.7 右のように，全脳の θ と比べて target 課題において大きくパワーが減衰することがわかった．この差は $p < 0.01$ で有意であり，被験者が target 課題においてはうまく変数値を記憶しておらず，逐一変数の定義に戻ってその意味を理解しなおしていたなどの可能性がある．

C) 意味的統合タスク： タスク C での target-control 間では，いずれの帯域のパワーも target 課題においてやや増大しており，γ 帯では比較的大きな差が見られた．分布の差についてウィルコクソンの順位和検定を実施したところ，いずれの周波数帯域においても有意な差は確認できなかった．

どの帯域にも有意差が見られなかった結果として，被験者のソートアルゴリズムに対する知識の不足が考えられる．特にタスク C は target 課題，control 課題ともに全タスクの中でもっとも行数が多く，コントロールフローの複雑度も高い課題であり，タイトルの有無による差が脳波の計測値に表れなかったことが示唆

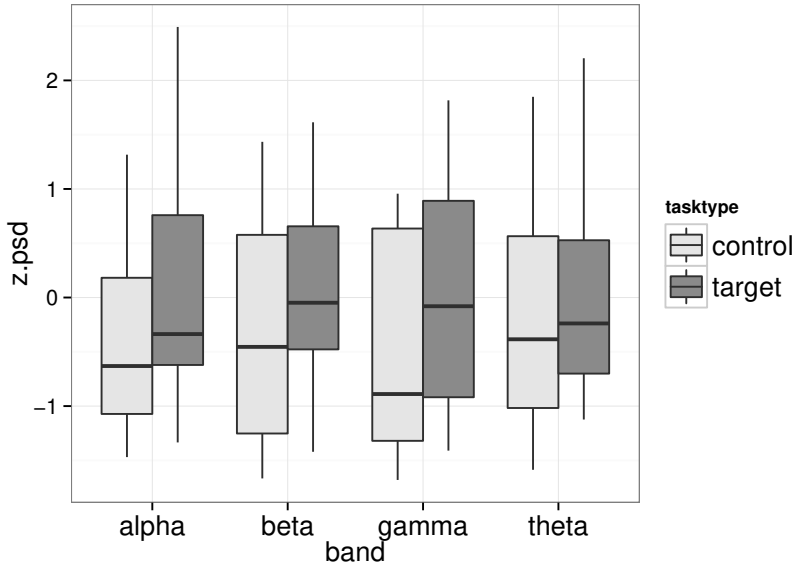

図 4.9 タスク C における課題タイプ/帯域別パワー

される．γ 帯の増大は一般に強い注意や高次の精神活動と結びついて考えられており，事前にプログラムの目的が判明している control 課題と対照的な target 課題において，被験者の探索的な理解手法を誘発した可能性がある．

D) 文法的統合タスク ： タスク D では，いずれの帯域のパワーも target 課題において増大しており，特に α 帯と θ 帯には顕著な差が見られた．分布の差についてウィルコクソンの順位和検定を実施したところ，α 帯のパワーにみられた差 ($p < 0.05$) および θ 帯の差 ($p < 0.01$) は有意であると確認された．

明らかに通常見られないプログラムである target 課題において，リラックスした状態，あるいは落ち着いた集中に関連するとされる α 帯のパワーが増大することは考えづらく，解釈が難しい．一つの解釈として，被験者がプログラムの機能を理解することを完全に諦め，原始的なやり方でプログラムを上から順に実行していったため，高次の知的活動としての負荷は逆に下がった可能性が考えられる．また，θ 帯の増大は頻繁に発生する小数点以下の演算および不規則なループ条件値の更新との関係が示唆される．control 課題では割り算の発生するタイミングはプログラム終了直前の一瞬のみであるが，target 課題では 20 度にわたって繰り下

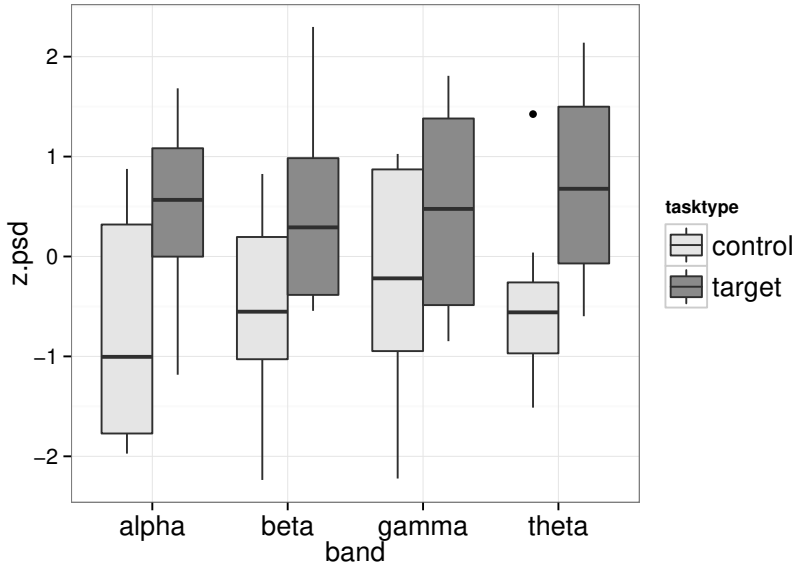

図 4.10　タスク D における課題タイプ/帯域別パワー

がり・繰り上がりのある暗算を行う必要があり，作業記憶の利用と関連する θ 帯の増大につながったと考えられる．

3.2　3 章の結果との比較

実験では心理学実験課題を参考に，各認知機能を強く要求するような人為的調整を加えたプログラムを用意したが，周波数分析の結果はほとんどの場合において事前に期待された仮説と異なった．この結果は，商用の脳波計測装置によってプログラム理解タスク中に発生する認知機能の酷使を捉えることには大きな障壁があることを示唆するものである．原因としては，脳波計測装置のノイズ耐性の低さ，プログラム理解それ自体が（認知機能の酷使を要求せずとも）ある程度複雑な知的作業であることが影響している可能性が考えられる．

この結果は 3 章で提案した NIRS 装置による脳の特定部位の計測結果で，easy とその他の難易度との間に見られた 17 人中 16 人の差と比較して著しく悪い結果であると考えられる．したがって，単一の装置を用いて認知機能の酷使からプログラム理解の難航を測定する場合においては，少なくとも作業記憶に関しては

NIRS 装置の方が実用性が高いと言える．

ただし，NIRS 装置は EEG 装置に比べて歴史が新しく，脳の複数部位の活動を計測できる簡易な計測装置はまだ登場しておらず，本論文でも他の認知機能の酷使は対象としていない．装置の価格も EEG 装置に比べて十倍以上高価である．そのためいくらかのタスクで特定の周波数成分に有意な差が計測された EEG 装置を代用的に適用する余地は残る．本研究の結果から，EEG 装置の適用においては，既存研究における他の生体情報計測装置との併用などを考慮し，計測には限界があることに留意する必要があると考えられる．また，これまでに本実験における Fm-θ のような特定の部位における周波数成分に着目した手法は提案されておらず，計測精度の向上に寄与する可能性がある．

4　まとめ

本章では，開発者が直面している困難さの種類や原因に応じた適切な支援のための第一歩として，簡易な脳波計測装置を用いて困難さの種類が判別する手法を提案し，その有効性を検証した．これまで，プログラム理解中の開発者の状態や，理解中のプログラムの困難さなどを測定するために脳活動・脳波に注目するアプローチが提案されてきた．他の生体情報も併用した機械学習による判別でも，なお二値判別の精度が 70%以下であるなど改善の余地があり，また，理解するプログラムに含まれる困難さや，被験者が実際に直面する困難さの種類については十分検討されてこなかった．

そこで，本研究では簡易な脳波計測装置によって困難さを推定するような方法がどの程度現実的なものかについて実験による調査を行った．実験では，実験室的な環境において，意図的に異なる種類の困難さを誘発するようなプログラムとそうでないプログラムを用意し，理解中の脳波に表れる周波数成分の差異を分析した．具体的には，認知機能の行使が開発者の能力を超えて求められたとき，プログラム理解の進行に支障を来すという仮説を立て，理解時に酷使される認知機能の種類が判別できるかを調べた．

実験では，既存研究においてプログラム理解と関連するとされる 4 つの認知機

能に着目し，過去の心理学実験課題を参考に，各認知機能を強く要求するよう人為的に調整を加えたプログラムを作成した．被験者 14 名から機器故障で計測に失敗した 3 名を除いた 11 名についてのデータ分析の結果，心理学分野の研究報告に沿う形で，作業記憶を強く要する理解課題の最中に θ 帯のパワーが有意に増大すること，記憶の難しい変数名を多用した課題においては既存研究と同じく β 帯のパワーが減衰する傾向が確認された．

これらの結果は，簡易かつ安価な脳波計測装置を用いて開発者が直面している困難さの種別や，プログラム中の問題点を発見することには大きな障壁があることを示唆する．一方で，いくらかの課題においては，理由は不明ながらも 11 名程度のデータで十分有意な差が検出された周波数帯もあるため，種類によっては簡易な脳波計測装置で困難さの識別ができる可能性も示唆された．

今後の発展的課題として，前頭部の脳血流計測では明確に差が見られた作業記憶への負荷（タスク A）における差が脳波では検出できないなどの特徴から，脳波・脳血流，ならびに過去の研究で取り上げられた視線・筋電など複数の生体情報計測装置を組み合わせた計測方法について考慮することは面白い．また，既存研究においては計測値の意味や困難さの種類への言及が不足しており，それらの観点に着目した上で，手法の適用範囲を徐々に確認してゆくことが望ましいと考えられる．

第5章　妥当性への脅威

1　内的妥当性への脅威

内的妥当性への脅威の一つとして，被験者は複数回繰り返して課題プログラムを読むため，実験結果は慣れの影響を受けている可能性がある．本研究では，実験中に生じる可能性のある順序効果による慣れの影響は，順序をランダム化することによって除去を試みた．また3章ではこの影響をより小さくするために，練習課題を実施した．練習課題ではソースコードの構文や読み方についての質問を受け付け，すべての被験者が課題内容に一定以上慣れたうえで実験を行った．主観評価においてはいずれの被験者も difficult の難易度に3以上 (難しい) の値をつけていたことから，難易度設定についての妥当性はある程度確保されたものと考えられる．

被験者間の経験や能力の差も内的妥当性への脅威となりうる．本研究では，理解困難状態を誘発する課題に，たとえ熟練者であっても経験したことがないと思われる，実験的なプログラムを用意することで経験の差を排除するように試みた．また,「学習段階のプログラマに対する支援」という研究目的に沿う形で被験者の経験・能力を制限した．具体的には，被験者はプログラミング教育を受けたことがあり，アルバイト以外での実務プログラミング経験をもたない大学・大学院の学生のみからなる．

プログラム理解課題の途中，被験者が理解を諦めた場合，実験結果に影響を与えると考えられる．本研究では，実験前の説明において，被験者にプログラム理解を極力諦めないように指示し，また制限時間を設けてなるべく被験者のモチベーションを下げないよう試みた．3章では被験者にメンタルシミュレーションの途中の状態を定期的に報告させることで，被験者のプログラム理解の進行を促した．

制限時間を迎えた課題においても，すべての被験者が制限時間近くまで状態の報告を行っており，被験者がプログラム理解を諦めた可能性は低いと考えられる．4章では理解手法を制限しなかったために途中状態を形式的に報告させる方法を取ることができなかったが，制限時間を5分と短く設定することで，影響を小さくすることに努めた．

3章では，タスク完了時間とactivationの関係を見ているが，実験においてタスクの制限時間を10分としているため，理解に10分以上を要する場合の結果については説明できない．特に，難易度difficultではすべての被験者のタスク完了時間が10分となっており，同タスク内の完了時間によるactivationの差は明らかでない．

4章では，内部で用いられているJavaメソッド等について実験前に一通りの説明を行うことで，被験者間の差の吸収を試みた．難易度設定については，万全とは言いにくいが，過去に心理学分野で利用された実験課題と似た状況をプログラム読解中に再現する試みによって，妥当性を補強している．また，異なる認知機能を要求するはずのプログラム間でも，ある程度は他のプログラムで要求される認知機能が利用されていることが考えられる．そのため，4章においては，特定の認知機能を強く利用すること以外は，target課題と機能や見た目の点で類似した課題(control課題)を用意し，その二者間での比較を行うことで，統制を試みた．

2　外的妥当性への脅威

本研究の外的妥当性への脅威として，現実世界のプログラム理解においては，プログラムの機能，実装法，そしてその理解戦略も非常に多様であり，実験ではそのうち一事例しか対象にできないことがある．たとえば，3章では，被験者がプログラム理解に困難を感じる状態を誘発するため，実験タスクではアルゴリズムの理解しにくさに注目し，制御フローの難読化を用いて難易度を調整している．また，理解戦略も，制御フローやデータフローの理解に用いられるメンタルシミュレーションに限定している．一方で，理解を困難にする要因には，アルゴリズムの複雑性以外にもモジュールの凝集度・結合度，不適切な変数名やコメントなど

様々な種類があり，理解戦略もそれに応じて多様なものが用いられる．

これらの制約のため，現時点における 3 章の成果は，小規模プログラムにおけるアルゴリズムの理解が困難な状況の検出・支援に限定され，また，現実のプログラムの読みにくさを反映しているとは限らない．より大規模で現実的なプログラムの理解を想定し，編集に伴うモジュールの役割変化，不適切なコメントに代表されるコード劣化など，多様な困難を誘発する状況下での実験を行うことが，今後の発展的課題となる．

多様な困難を扱えないというリスクの削減を目的として，4 章においては 4 種類の認知機能を酷使する異なる課題を用意し，脳波の周波数特性がそれぞれの課題間でどのように異なるかを調べた．しかし，本実験は心理学分野で利用された実験課題を強く意識したもので，一部のプログラムは非常に非日常的で実験室的な処理構造や見た目をしている．たとえば，変数名の有意味度と記憶を扱ったタスク (図 4.2) においては，存在しない語（非語）を変数名に用いることで，意図的にプログラムの理解を困難にしており，現実環境より厳しい条件になっている．そのため，4 章における結果は，現実におけるプログラミング環境で同様の結果が確認できることを保証せず，あくまでプログラム理解中に発生した困難さの種類が，脳波の周波数特性に波及し，それを検出できる可能性を示唆するにとどまると考えられる．

また，本研究では実験の被験者がすべて学生であった．このことは，プログラム開発に熟練した開発者を計測する場合などを想定した場合，外的妥当性への脅威となりうる．ただし，実験タスクに用いられた構文等は学生に理解できる程度のものであり，影響は小さいものと考えられる．今後，発展的な課題においてより現実的な環境での実験を試行する際には，被験者の技術やドメイン知識に関する事前調査を行ったうえでの適用が望ましい．

第6章　おわりに

プログラム理解はソフトウェアの保守において重要な役割を持ち，そのコストの多くを占める難解な作業である．プログラム理解は長い時間をかけて行われる作業であり，時に難航（理解作業の遅延・停滞）し，コスト増大につながる．また，特に学習段階の開発者については，一人では乗り越えられないような問題や誤解に直面することもある．

仮にこのような理解の難航度合いを測定することができれば，プログラム理解に行き詰った学習者に対してリアルタイムにアドバイスを与えることで，プログラミング教育の効率化や学習者の負荷低減が実現できると考えられる．そこで本論文では，プログラム理解を試みる開発者が作業の遂行に支障を来している度合い（理解困難度）を，脳波・脳活動によりリアルタイムに測定する手法を提案した．

本手法では理解困難度が上昇している間，開発者はその限界を超えた思考量を求められていると仮定し，脳波・脳活動により思考を構成する認知プロセスの量や質を調べ，間接的に理解困難度を計測する．具体的には，3 章における「近赤外分光法 (NIRS) による脳活動計測に基づく理解困難度の測定」および 4 章「脳波の周波数解析に基づく理解困難の原因（理解に必要な認知プロセスの種類）測定」からなる．

3 章においては，プログラム理解に支障をきたしている（理解困難度が高い状態の）開発者は，頭の中でなんとかプログラムを理解しようと種々の認知機能を駆使するはずである，という仮定のもと，脳活動の量を計測することで，間接的にプログラム理解中の理解困難度を計測する手法を提案した．被験者実験では，人間の手によって意図的に理解困難度を変更したプログラムを 3 種類（easy, middle, difficult) 用意し，その理解中の前頭前野における脳活動量を比較した．実験の結果，17 名の被験者のうち 16 名について，difficult 課題遂行中の脳活動が easy 課

題遂行中に比べて活発であることがわかった．この結果は「プログラム理解に困難が生じている状態を，理解時の脳血流計測によって計測できる.」という仮説を支持するものであり，本手法が理解困難度の測定に適用できる可能性が示された．ただし，difficult 課題遂行中の脳活動量と medium 課題遂行中の脳活動量には有意な差が見られず，極端な困難さと中程度の困難さの判別は行えないことも明らかとなった．

また，プログラム理解中の脳活動の中央値・最大値と，アンケートで聴取した主観的なプログラム理解の難しさの間にはそれぞれ相関係数 0.42, 0.59 の有意な相関が見られた．このことから，本手法で計測された理解困難度は，被験者がプログラム理解中に感じた困難さの量をよく再現していることが示唆される．

4 章においては，困難さの種類に応じた適切な支援などを目指した先駆的調査として，理解時に酷使される認知機能の種類とその利用量を脳波の周波数成分から知ることができるかを被験者実験によって調査した，実験に先立って，既存研究においてプログラム理解と関連するとされる 4 つの認知機能に着目し，それぞれを強く利用するプログラムとそうでないプログラムを，過去の心理学実験課題を参考に作成した．被験者 7 名から機器故障で計測に失敗した 2 名を除いた 5 名についてのデータ分析の結果，作業記憶域の酷使が予想されるタスクの一部において，θ 帯の有意な増大が確認された．しかし，一方で事前に予想された結果と異なり，困難度が異なると考えられる二つの課題間で脳波の周波数成分にほとんど差がみられない課題もあったことから，簡易な脳波計測装置で測定可能な困難さの種類は限られていることが明らかになった．

本研究で示した結果は，いずれも実験設定に示した環境のもとで得られたものであり，その適用可能範囲は数あるプログラム理解活動のうち，一部に限られる．そこで，今後の発展的な展望として，支援を必要とするプログラム理解活動という観点から問題を整理したうえで，計測手法の適用先や有効性を検討していくことが課題となる．例えば，プログラム実行，デバッガ等のツールの利用，設計書の参照等を含むプログラム理解活動に対して，脳活動計測による具体的な支援を検討していきたい．具体的な方針としては，プログラムの実行やデバッガに代表されるツールの利用，あるいは設計書の参照など，他の理解活動について考慮し

た調査が求められる．

本研究と関連する今後の発展的課題として，より現実的な環境において，特定の認知機能の酷使を検出したり，またその差を特定できるかについての調査が望まれる．このような調査は，脳波計測装置単体では難しいものと考えられるが，たとえばNIRSやfMRIといったより高度な装置との併用計測や，機械学習的な手法との組み合わせなどによって実現できる可能性がある．

謝辞

本研究を進めるにあたり，大変多くの方々からご指導，ご協力，ご助言をいただきました．この研究にかかわってくださった全ての方に対して，心から感謝いたします．誠にありがとうございました．

奈良先端科学技術大学院大学　情報科学研究科　松本 健一 教授には，本論文の主指導教員をご担当いただきました．先生からは,「そもそも研究とは何か，論文はどうあるべきか」といった大局的な部分にはじまり，発表資料や論文における日本語書法といった詳細な部分まで，包括的で示唆に富んだご指導・ご意見を賜りました．先生の時に妥協のない鋭いご指摘や，暖かいご助言といった熱意ある指導なくして，本論文はなかったものと強く確信しております．ここに，心からの深い感謝を申し上げます．

奈良先端科学技術大学院大学　情報科学研究科　飯田 元 教授には，本論文の副指導教員をご担当いただきました．先生からは，本研究の発表において研究の目的や方向性について鋭いご指摘を賜りました．また，研究室の異なる学生であるにも関わらず，平時にも研究者としてのあり方についてご助言をいただきました．ここに深く感謝申し上げます．

岡山大学　自然科学研究科　門田 暁人 教授には，本論文の審査委員をご担当いただきました．先生は私の博士前期課程時代から，一貫して研究におけるストーリーと説得力の大切さについてご指導くださいました．また実験結果の解釈や，論文執筆法，査読対応法といった研究を進めるうえで重要な技術について，何度も心強いご指導をいただきました．ここに心から感謝を申し上げます．

奈良工業高等専門学校　情報工学科　上野 秀剛 講師には，本論文の審査委員をご担当いただきました．先生は，研究における実験デザインや結果の解釈，考察の部分について的確なご指摘・ご助言をくださいました．本研究テーマについての厳しくも熱意ある真剣な視線からは，あるべき研究者の姿勢を学ばせていただきました．深く感謝申し上げます．

九州大学 大学院システム情報科学研究院 鵜林 尚靖 教授は，九州大学でのインターンシップを快く承諾してくださり，さらに，研究の将来的な方向性から分

析手法，論文の構成まで，多岐にわたるご助言・ご指摘をいただきました．心から感謝申し上げます．

九州大学 大学院システム情報科学研究院 亀井 靖高 准教授は，九州大学でのインターンシップの際，私の研究内容に対して親身にご指導してくださりました．その後も論文の構成や執筆時の細かな技法にいたるまで，丁寧にご指導くださいました．また，時折下さる暖かな激励は研究を進めるうえでの精神的な支えとなりました．深く感謝申し上げます．

ヴィクトリア大学 Department of Computer Science， Daniel M Germán 教授は，本研究に対して多大な興味を示してくださり，研究の方向性や実験のアイデアから，英語での論文執筆法について多大なご意見・ご協力を賜りました．深く感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 伊原 彰紀 助教には，博士前期課程の入学時より，研究内容，研究生活，研究者としての姿勢について親身にご助言いただきました．先生の前向きであり，かつ時にストイックなご意見は研究を進めるうえで大変参考になりました．心から感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 畑 秀明 助教は，本研究と関連する研究論文や書籍，また新たなアイデアや観点を次々と提供してくださいました．また，私の研究テーマに強い興味と肯定を示してくださった上で，鋭いご意見をくださいました．深く感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 久保 孝富 特任准教授は，本論文の発表において神経科学の専門家としての立場から鋭く，かつ建設的なご意見をくださりました．心から感謝申し上げます．

奈良先端科学技術大学院大学　情報科学研究科 崔 恩瀞 助教には，私の研究内容にご興味を持っていただき，学会発表や論文公聴会において多大なご指摘・ご助言をいただきました．深く感謝申し上げます．

豊田工業高等専門学校　情報工学科 藤原 賢二 助教は，奈良先端科学技術大学院大学におけるソフトウェア工学分野の先輩として，研究内容や研究生活について多数のご助言をいただきました．ここに感謝申し上げます．

奈良先端科学技術大学院大学　情報科学研究科 Passakorn Phannachitta 氏に

は，博士後期課程の先輩として，研究に取り組む姿勢や物事の捉え方についてご助言をいただきました．ここに感謝申し上げます．

奈良先端科学技術大学院　情報科学研究科 ソフトウェア工学研究室ならびにソフトウェア設計学研究室の学生の皆様には，日ごろから研究生活を共にして頂きました．充実した5年間を送ることができたのはひとえに皆様のおかげです．ありがとうございました．特に，博士後期課程学生として立場を同じくし，精神的な支えとなってくださいました尾上 紗野 氏，坂口 英司 氏，北川 愼人 氏，上村 恭平 氏，槇原 絵里奈氏に感謝を申し上げます．また，数少ない隣接テーマの研究者として私の研究内容に関する相談に乗り，また鋭い質問をしてくださいました幾谷 吉晴 氏に感謝を申し上げます．

最後に，研究を進めるにあたって様々な問題にぶつかる私を精神的に支えて下さり，暖かく見守ってくれた家族に感謝いたします．

参考文献

[1] Erik Arisholm and Lionel C Briand. Predicting fault-prone components in a java legacy system. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pp. 8–17. ACM, 2006.

[2] Dustin Boswell and Trevor Foucher. リーダブルコード - より良いコードを書くためのシンプルで実践的なテクニック. O'Reilly Japan, 2012.

[3] Randy L Buckner and Steven E Petersen. What does neuroimaging tell us about the role of prefrontal cortex in memory retrieval? In *Seminars in Neuroscience*, Vol. 8, pp. 47–55. Elsevier, 1996.

[4] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H. Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. Eye movements in code reading: Relaxing the linear order. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*, ICPC '15, pp. 255–265, Piscataway, NJ, USA, 2015. IEEE Press.

[5] Teresa Busjahn, Carsten Schulte, Bonita Sharif, Simon, Andrew Begel, Michael Hansen, Roman Bednarik, Paul Orlov, Petri Ihantola, Galina Shchekotova, and Maria Antropova. Eye tracking in computing education. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*, ICER '14, pp. 3–10, New York, NY, USA, 2014. ACM.

[6] Roberto Cabeza and Lars Nyberg. Imaging cognition ii: An empirical review of 275 pet and fmri studies. *Journal of cognitive neuroscience*, Vol. 12, No. 1, pp. 1–47, 2000.

[7] C. Caprile and P. Tonella. Nomen est omen: analyzing the language of function identifiers. In *Sixth Working Conference on Reverse Engineering (Cat. No.PR00303)*, pp. 112–122, Oct 1999.

[8] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke. A systematic survey of program comprehension through dynamic analysis. *IEEE Transactions on Software Engineering*, Vol. 35, No. 5, pp. 684–702, Sept 2009.

[9] Xu Cui, Signe Bray, and Allan L. Reiss. Functional near infrared spectroscopy (nirs) signal improvement based on negative correlation between oxygenated and deoxygenated hemoglobin dynamics. *NeuroImage*, Vol. 49, No. 4, pp. 3039–3046, 2010.

[10] Françoise Détienne. Expert programming knowledge: a schema-based approach. *Psychology of programming*, pp. 205–222, 1990.

[11] Alastair Dunsmore and Marc Roper. A comparative evaluation of program comprehension measures. *The Journal of Systems and Software*, Vol. 52, No. 3, pp. 121–129, 2000.

[12] K Anders Ericsson and Herbert A Simon. Verbal reports as data. *Psychological review*, Vol. 87, No. 3, 1980.

[13] Vikki Fix, Susan Wiedenbeck, and Jean Scholtz. Mental representations of programs by novices and experts. In *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*, pp. 74–79, Amsterdam, The Netherlands, The Netherlands, 1993. IOS Press.

[14] Thomas Fritz, Andrew Begel, Sebastian C Müller, Serap Yigit-Elliott, and Manuela Züger. Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*, pp. 402–413. ACM, 2014.

[15] John M Gardiner and Rosalind I Java. Recollective experience in word and nonword recognition. *Memory & Cognition*, Vol. 18, No. 1, pp. 23–30, 1990.

[16] Alan Gevins, Michael E Smith, Linda McEvoy, and Daphne Yu. High-resolution eeg mapping of cortical activation related to working memory: effects of task difficulty, type of processing, and practice. *Cerebral cortex*, Vol. 7, No. 4, pp. 374–385, 1997.

[17] Thomas Gruber, Andreas Keil, and Matthias M Müller. Modulation of induced gamma band responses and phase synchrony in a paired associate learning task in the human eeg. *Neuroscience letters*, Vol. 316, No. 1, pp. 29–32, 2001.

[18] Alexander Gundel and Glenn F Wilson. Topographical changes in the ongoing eeg related to the difficulty of mental tasks. *Brain topography*, Vol. 5, No. 1, pp. 17–25, 1992.

[19] Simon Hanslmayr, Gregor Volberg, Maria Wimber, Markus Raabe, Mark W Greenlee, and Karl-Heinz T Bäuml. The relationship between brain oscillations and bold signal during memory formation: a combined eeg–fmri study. *The Journal of Neuroscience*, Vol. 31, No. 44, pp. 15674–15680, 2011.

[20] Yoko Hoshi and Mamoru Tamura. Near-infrared optical detection of sequential brain activation in the prefrontal cortex during mental tasks. *NeuroImage*, Vol. 5, No. 4, pp. 292–297, 1997.

[21] Yoshiharu Ikutani and Hidetake Uwano. Brain activity measurement during program comprehension with nirs. *International Journal of Networked and Distributed Computing*, Vol. 2, No. 4, pp. 259–268, Nov. 2014.

[22] Amela Karahasanović, Unni Nyhamar Hinkel, Dag I.K. Sjøberg, and Richard Thomas. Comparing of feedback-collection and think-aloud methods in program comprehension studies. *Behaviour & Information Technology*, Vol. 28, No. 2, pp. 139–164, 2009.

[23] Amela Karahasanović, Annette Kristin Levine, and Richard Thomas. Comprehension strategies and difficulties in maintaining object-oriented systems:

An explorative study. *Journal of Systems and Software*, Vol. 80, No. 9, pp. 1541–1559, 2007.

[24] Timothy A. Keller, Patricia A. Carpenter, and Marcel Adam Just. The neural bases of sentence comprehension: a fmri examination of syntactic and lexical processing. *Cerebral Cortex*, Vol. 11, No. 3, pp. 223–237, 2001.

[25] Andrew J. Ko, Brad A. Myers, and Htet Htet Aung. Six learning barriers in end-user programming systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing*, VLHCC '04, pp. 199–206, Washington, DC, USA, 2004. IEEE Computer Society.

[26] D. Lawrie, C. Morrell, H. Feild, and D. Binkley. What's in a name? a study of identifiers. In *14th IEEE International Conference on Program Comprehension (ICPC'06)*, pp. 3–12, June 2006.

[27] Stanley Letovsky. Cognitive processes in program comprehension. *Journal of Systems and software*, Vol. 7, No. 4, pp. 325–339, 1987.

[28] A. Von Mayrhauser and A. M. Vans. Identification of dynamic comprehension processes during large scale maintenance. *IEEE Transactions on Software Engineering*, Vol. 22, No. 6, pp. 424–437, Jun 1996.

[29] Steve McConnel. What does 10x mean? measuring variations in programmer productivity. Making Software, O ' Reilly, Vol. 16, , 2011.

[30] Timothy McMahan, Ian Parberry, and Thomas D Parsons. Evaluating electroencephalography engagement indices during video game play. In *Proceedings of the Foundations of Digital Games Conference*, 2015.

[31] Akito Monden, Yoshihiro Takada, and Koji Torii. Method for scrambling programs containing loops. *IEICE Trans. on Information and Systems*, Vol. 80, No. 7, pp. 644–652, Jul. 1997.

[32] S. C. Muller and T. Fritz. Stuck and frustrated or in flow and happy: Sensing developers' emotions and progress. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1, pp. 688–699, May 2015.

[33] Masahide Nakamura, Akito Monden, Tomoaki Itoh, Keníchi Matsumoto, Yuichiro Kanzaki, and Hirotsugu Satoh. Queue-based cost evaluation of mental simulation process in program comprehension. In *Proceedings of 9th IEEE International Software Metrics Symposium (METRICS 2003)*, pp. 351–360, Sep. 2003.

[34] Chris Parnin. A cognitive neuroscience perspective on memory for programming tasks. In *Proceedings of 22nd Annual Meeting of the Psychology of Programming Interest Group (PPIG)*, 2010.

[35] Nancy Pennington. Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive psychology*, Vol. 19, No. 3, pp. 295–341, 1987.

[36] Nancy Pennington and Beatrice Grabowski. The tasks of programming. *Psychology of programming*, Vol. 307, pp. 45–62, 1990.

[37] T. Richner and S. Ducasse. Recovering high-level views of object-oriented applications from static and dynamic information. In *Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on*, pp. 13–22, 1999.

[38] Spencer Rugaber. Program comprehension. *Encyclopedia of Computer Science and Technology*, Vol. 35, No. 20, pp. 341–368, 1995.

[39] Jean Saint-Aubin and Marie Poirier. Immediate serial recall of words and nonwords: Tests of the retrieval-based hypothesis. *Psychonomic Bulletin & Review*, Vol. 7, No. 2, pp. 332–340, 2000.

[40] Janet Siegmund. Program comprehension: Past, present, and future. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 5, pp. 13–20. IEEE, 2016.

[41] Janet Siegmund, André Brechmann, Sven Apel, Christian Kästner, Jörg Liebig, Thomas Leich, and Gunter Saake. Toward measuring program comprehension with functional magnetic resonance imaging. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE '12)*, pp. 24:1–24:4. ACM, 2012.

[42] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. Understanding understanding source code with functional magnetic resonance imaging. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*, pp. 378–389, New York, NY, USA, 2014. ACM.

[43] Elliot Soloway and Kate Ehrlich. Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering*, No. 5, pp. 595–609, 1984.

[44] Marie St George, M Kutas, A Martinez, and MI Sereno. Semantic integration in reading: engagement of the right hemisphere during discourse processing. *Brain*, Vol. 122, No. 7, pp. 1317–1325, 1999.

[45] M-A Storey. Theories, methods and tools in program comprehension: Past, present and future. In *13th International Workshop on Program Comprehension (IWPC'05)*, pp. 181–191. IEEE, 2005.

[46] Rachel Turner, Michael Falcone, Bonita Sharif, and Alina Lazar. An eye-tracking study assessing the comprehension of c++ and python source code. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, pp. 231–234, New York, NY, USA, 2014. ACM.

[47] K. Utsugi, A. Obata, H. Sato, T. Katsura, K. Sagara, A. Maki, and H. Koizumi. Development of an optical brain-machine interface. In *29th*

Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2007 (EMBS 2007), pp. 5338–5341, Aug. 2007.

[48] Yaling Yang and Adrian Raine. Prefrontal structural and functional brain imaging findings in antisocial, violent, and psychopathic individuals: A meta-analysis. *Psychiatry Research: Neuroimaging*, Vol. 174, No. 2, pp. 81–88, 2009.

[49] Laure Zago, Mauro Pesenti, Emmanuel Mellet, Fabrice Crivello, Bernard Mazoyer, and Nathalie Tzourio-Mazoyer. Neural correlates of simple and complex mental calculation. *NeuroImage*, Vol. 13, No. 2, pp. 314–327, 2001.

[50] Manuela Züger and Thomas Fritz. Interruptibility of software developers and its prediction using psycho-physiological sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 2981–2990, New York, NY, USA, 2015. ACM.

[51] 栗山進, 大平雅雄, 門田暁人, 松本健一. プログラム理解度がコードレビュー達成度に及ぼす影響の分析. 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス, Vol. 104, No. 571, pp. 17–22, Jan. 2005.

[52] 高橋圭一. 簡易脳波測定装置を用いたプログラミング活動のストレス測定に関する実験. ソフトウェア工学の基礎 XX 日本ソフトウェア科学会 FOSE 2013. 近代科学社, Nov. 2013.

[53] 三輪和久, 杉江昇. 学習の初期段階における計算機プログラミングの動的過程. 人工知能学会誌, Vol. 7, No. 1, pp. 138–148, Jan. 1992.

[54] 山口修平. 前頭葉と記憶. 高次脳機能研究, Vol. 27, No. 3, pp. 222–230, 2007.

[55] 上野秀剛, 中村匡秀, 門田暁人, 松本健一. プログラマの視線を用いたコードレビュー性能の要因分析. ソフトウェア工学の基礎 XIII 日本ソフトウェア科学会 FOSE2006, pp. 103–112, Nov. 2006.

[56] 新美良純, 白藤美隆. 皮膚電気反射—基礎と応用. 医歯薬出版, 1969.

[57] 西牧謙吾, 国立特殊教育総合研究所. 脳科学と障害のある子どもの教育に関する研究. 国立特殊教育総合研究所, 2007.

[58] 石黒誉久, 井垣宏, 中村匡秀, 門田暁人, 松本健一. 変数更新の回数と分散に基づくプログラムのメンタルシミュレーションコスト評価. 電子情報通信学会技術報告, ソフトウェアサイエンス研究会, 第 SS2004-32 巻, pp. 37–42, Nov. 2004.

[59] 村岸厳. 皮膚抵抗値によるソフトウェア開発者の負荷評価に関する研究. Master's thesis, 奈良先端科学技術大学院大学, 1998.

付録

A　三章の実験で用いた課題プログラムの一覧

以下に参照の実験で用いた課題プログラムの一覧を掲載する．ここでは節タイトルに問題難易度が併記されているが，被験者への影響を排除するため問題提示時には難易度を記載していない．

A.1　練習問題 easy

下のような引数を与えて関数 func を実行させると、戻り値として 8 が返されることを示せ。

$$A[0][0] = 3 \quad A[0][1] = 5$$
$$A[1][0] = 4 \quad A[1][1] = 2$$
$$A[2][0] = 1 \quad A[2][1] = 8$$

$$N = 3$$
$$M = 2$$

```
int func (int **A, int N, int M)
{
    int i, j;
    int x;
    (0)
    x = A[0][0]; (1)
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < M; j++)
        {
            if (A[i][j] > x)
```

```
            {
                x = A[i][j]; (2)
            }
        }
    }
    (3) return x;
}
```

A.2 練習問題 difficult

下のような引数を与えて関数 func を実行させると、戻り値として 8 が返されることを示せ。

$$A[0][0] = 3 \quad A[0][1] = 5$$
$$A[1][0] = 4 \quad A[1][1] = 2$$
$$A[2][0] = 1 \quad A[2][1] = 8$$

$$N = 3$$
$$M = 2$$

```
int func (int **A, int N, int M)
{
    int i, j;
    int x;
    (0)
    x = A[0][0]; (1)
    i = 0;
    if(i < N){
        j = 0;
```

```
        if(j < M){
            for(;;){
                if (A[i][j] > x) x = A[i][j]; (2)
                i++;
                if(i >= M){
                    for(; i < N; i++){
                        if (A[i][j] > x) x = A[i][j]; (3)
                    }
                    j++;
                    if(j >= M) break;
                    i = j;
                    continue;
                }
                if (A[j][i] > x) x = A[j][i]; (4)
                if(i >= N){
                    for(; i < M; i++){
                        if (A[j][i] > x) x = A[j][i]; (5)
                    }
                    j++;
                    if(j >= N) break;
                    i = j;
                }
            }
        }
    }
    (6) return x;
}
```

A.3 問題 A easy

下のような引数を与えて関数 func を実行させると、戻り値として 11 が返されることを示せ.

$$
\begin{array}{lll}
A[0][0][0] = 97 & A[0][0][1] = 48 & \\
A[0][1][0] = 52 & A[0][1][1] = 71 & \\
A[1][0][0] = 17 & A[1][0][1] = 64 & \\
A[1][1][0] = 11 & A[1][1][1] = 32 & \\
A[2][0][0] = 20 & A[2][0][1] = 22 & \\
A[2][1][0] = 48 & A[2][1][1] = 86 & \\
N = 3 & M = 2 & L = 2
\end{array}
$$

int func(int ***A, int N, int M, int L)
{
    int i, j, k;
    int p;
    (0)
    $p = A[0][0][0]$; (1)
    for ($i = 0$; $i < N$; $i++$)
    {
        for ($j = 0$; $j < M$; $j++$)
        {
            for ($k = 0$; $k < L$; $k++$)
            {
                if($A[i][j][k] < p$) $p = A[i][j][k]$; (2)
            }
        }
    }
    (3) return p;

}

A.4 問題 A medium

下のような引数を与えて関数 func を実行させると、戻り値として 11 が返されることを示せ.

$$
\begin{array}{lll}
A[0][0][0] = 97 & A[0][0][1] = 48 & \\
A[0][1][0] = 52 & A[0][1][1] = 71 & \\
A[1][0][0] = 17 & A[1][0][1] = 64 & \\
A[1][1][0] = 11 & A[1][1][1] = 32 & \\
A[2][0][0] = 20 & A[2][0][1] = 22 & \\
A[2][1][0] = 48 & A[2][1][1] = 86 & \\
N = 3 & M = 2 & L = 2
\end{array}
$$

```
int func(int ***A, int N, int M, int L)
{
    int i, j, k;
    int p;
    (0)
    p = A[0][0][0]; (1)
    i = 0;
    for(;;){
        if(i >= N) break;
        j = 0;
        if(j < M){
L1:         for (k = 0; k < L; k++){
                if(A[i][j][k] < p) p = A[i][j][k]; (2)
            }
```

```
            j++;
        }else{
            i++;
            if(i >= N) break;
            j = 0;
        }
        if(j >= M) i++;
        else goto L1;
    }
    (3) return p;
}
```

A.5 問題 A difficult

下のような引数を与えて関数 func を実行させると、戻り値として 11 が返されることを示せ.

$$
\begin{array}{lll}
A[0][0][0] = 97 & A[0][0][1] = 48 & \\
A[0][1][0] = 52 & A[0][1][1] = 71 & \\
A[1][0][0] = 17 & A[1][0][1] = 64 & \\
A[1][1][0] = 11 & A[1][1][1] = 32 & \\
A[2][0][0] = 20 & A[2][0][1] = 22 & \\
A[2][1][0] = 48 & A[2][1][1] = 86 & \\
N = 3 & M = 2 & L = 2
\end{array}
$$

```
int func(int ***A, int N, int M, int L)
{
    int i, j, k, l;
    int p;
```

```
(0)
p = A[0][0][0]; (1)
for (i = 0; i < N; i++){
    l = M;
    for (j = 0; j < M; j++){
        M--;
        for (k = 0; k < L; k++){
            if(A[i][j][k] < p) p = A[i][j][k]; (2)
        }
        if(j >= M) break;
        for (k = 0; k < L; k++){
            if(A[i][M][k] < p) p = A[i][M][k]; (3)
        }
    }
    N--;
    if(i >= N) break;
    for(; M < l; M++){
        for (k = 0; k < L; k++){
            if(A[N][M][k] < p) p = A[N][M][k]; (4)
        }
        if(j <= 0) break;
        j--;
        for (k = 0; k < L; k++){
            if(A[N][j][k] < p) p = A[N][j][k]; (5)
        }
    }
}
(6) return p;
}
```

A.6 問題B easy

下のような引数を与えて関数 func を実行させると、$*neg$ が -45 になることを示せ.

$$\begin{array}{llll} A[0][0] = -3 & A[0][1] = 3 & A[0][2] = 0 & A[0][3] = -3 \\ A[1][0] = 1 & A[1][1] = -1 & A[1][2] = -7 & A[1][3] = -8 \\ A[2][0] = -5 & A[2][1] = 0 & A[2][2] = -6 & A[2][3] = 5 \\ A[3][0] = -2 & A[3][1] = 9 & A[3][2] = -4 & A[3][3] = -6 \end{array}$$

$$\begin{aligned} N &= 4 \\ M &= 4 \end{aligned}$$

```
void func (int **A, int N, int M, int *pos, int *neg)
{
    int i, j;
    (0)
    *pos =*neg = 0; (1)
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < M; j++)
        {
            if(A[i][j] > 0)
            {
                *pos+ = A[i][j]; (2)
            }
            else if(A[i][j] < 0)
```

```
            {
                *neg+ = A[i][j]; (3)
            }
        }
    }
    (4) return;
}
```

A.7 問題B medium

下のような引数を与えて関数 func を実行させると、$*neg$ が -45 になることを示せ.

$$\begin{array}{llll} A[0][0] = -3 & A[0][1] = 3 & A[0][2] = 0 & A[0][3] = -3 \\ A[1][0] = 1 & A[1][1] = -1 & A[1][2] = -7 & A[1][3] = -8 \\ A[2][0] = -5 & A[2][1] = 0 & A[2][2] = -6 & A[2][3] = 5 \\ A[3][0] = -2 & A[3][1] = 9 & A[3][2] = -4 & A[3][3] = -6 \end{array}$$

$$\begin{aligned} N &= 4 \\ M &= 4 \end{aligned}$$

```
void func (int **A, int N, int M, int *pos, int *neg)
{
    int i, j;
    (0)
    *pos =*neg = 0; (1)
    i = 0
    for(;;){
```

```
        if(i >= N) break;
        j = 0;
        if(j < M){
L1:         if(A[i][j] > 0) *pos+ = A[i][j]; (2)
            else if(A[i][j] < 0) *neg+ = A[i][j]; (3)
            j++;
        }else{
            i++;
            if(i >= N) break;
            j = 0;
        }
        if(j >= M) i++;
        else goto L1;
    }
    (4) return;
}
```

A.8 問題B difficult

下のような引数を与えて関数 func を実行させると、$*neg$ が -45 になることを示せ.

$$\begin{array}{llll} A[0][0] = -3 & A[0][1] = 3 & A[0][2] = 0 & A[0][3] = -3 \\ A[1][0] = 1 & A[1][1] = -1 & A[1][2] = -7 & A[1][3] = -8 \\ A[2][0] = -5 & A[2][1] = 0 & A[2][2] = -6 & A[2][3] = 5 \\ A[3][0] = -2 & A[3][1] = 9 & A[3][2] = -4 & A[3][3] = -6 \end{array}$$

$$N = 4\ ,\ M = 4$$

```
void func (int **A, int N, int M, int *pos, int *neg)
{
    int i, j, l;
    (0)
    *pos =*neg = 0; (1)
    for (i = 0; i < N; i++){
        l = M;
        for (j = 0; j < M; j++){
            M--;
            if(A[i][j] > 0) *pos+ = A[i][j]; (2)
            else if(A[i][j] < 0) *neg+ = A[i][j]; (3)
            if(j >= M) break;
            if(A[i][M] > 0) *pos+ = A[i][M]; (4)
            else if(A[i][M] < 0) *neg+ = A[i][M]; (5)
        }
        N--;
        if(i >= N) break;
        for(; M < l; M++){
            if(A[N][M] > 0) *pos+ = A[N][M]; (6)
            else if(A[N][M] < 0) *neg+ = A[N][M]; (7)
            if(j <= 0) break;
            j--;
            if(A[N][j] > 0) *pos+ = A[N][j]; (8)
            else if(A[N][j] < 0) *neg+ = A[N][j]; (9)
        }
    }
    (10) return;
}
```

A.9 問題C easy

下のような引数を与えて関数 func を実行させると、戻り値として 4 が返されることを示せ.

$$
\begin{array}{llll}
A[0][0] =' A' & A[0][1] =' H' & A[0][2] =' X' & A[0][3] =' G' \\
A[1][0] =' D' & A[1][1] =' J' & A[1][2] =' A' & A[1][3] =' X' \\
A[2][0] =' X' & A[2][1] =' K' & A[2][2] =' F' & A[2][3] =' J' \\
A[3][0] =' F' & A[3][1] =' X' & A[3][2] =' Q' & A[3][3] =' L'
\end{array}
$$

$$
\begin{aligned}
N &= 4 \\
M &= 4
\end{aligned}
$$

```
void func (int **A, int N, int M)
{
    int i, j;
    int x;
    (0)
    x = 0; (1)
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < M; j++)
        {
            if(A[i][j] ==' X')
            {
                x++; (2)
            }
        }
    }
    (3) return x;
```

}

A.10 問題C medium

下のような引数を与えて関数 func を実行させると、戻り値として 4 が返されることを示せ.

$$\begin{array}{llll} A[0][0] =' A' & A[0][1] =' H' & A[0][2] =' X' & A[0][3] =' G' \\ A[1][0] =' D' & A[1][1] =' J' & A[1][2] =' A' & A[1][3] =' X' \\ A[2][0] =' X' & A[2][1] =' K' & A[2][2] =' F' & A[2][3] =' J' \\ A[3][0] =' F' & A[3][1] =' X' & A[3][2] =' Q' & A[3][3] =' L' \end{array}$$

$$\begin{aligned} N &= 4 \\ M &= 4 \end{aligned}$$

```
void func (int **A, int N, int M)
{
    int i, j;
    int x;
    (0)
    x = 0; (1)
    i = 0;
    for(;;){
        if(i >= N) break;
        j = 0;
        if(j < M){
L1:         if(A[i][j] ==' X') x++; (2)
            j++;
```

```
        }else{
            i++;
            if(i >= N) break;
            j = 0;
        }
        if(j >= M) i++;
        else goto L1;
    }
    (3) return x;
}
```

A.11 問題C difficult

下のような引数を与えて関数 func を実行させると、戻り値として 4 が返されることを示せ.

$$
\begin{array}{llll}
A[0][0] =' A' & A[0][1] =' H' & A[0][2] =' X' & A[0][3] =' G' \\
A[1][0] =' D' & A[1][1] =' J' & A[1][2] =' A' & A[1][3] =' X' \\
A[2][0] =' X' & A[2][1] =' K' & A[2][2] =' F' & A[2][3] =' J' \\
A[3][0] =' F' & A[3][1] =' X' & A[3][2] =' Q' & A[3][3] =' L'
\end{array}
$$

$$
\begin{aligned}
N &= 4 \\
M &= 4
\end{aligned}
$$

```
void func (int **A, int N, int M)
{
    int i, j, l;
    int x;
```

```
(0)
x = 0; (1)
for (i = 0; i < N; i++){
    l = M;
    for (j = 0; j < M; j++){
        M--;
        if(A[i][j] =='X') x++; (2)
        if(j >= M) break;
        if(A[i][M] =='X') x++; (3)
    }
    N--;
    if(i >= N) break;
    for(; M < l; M++){
        if(A[N][M] =='X') x++; (4)
        if(j <= 0) break;
        j--;
        if(A[N][j] =='X') x++; (5)
    }
}
(6) return x;
}
```

B　四章の実験で用いた課題プログラム (control 課題のみ) の一覧

ここでは四章の実験で用いた課題プログラムの一覧を示す．target 課題については，実験設計に影響する項目であるために本文中に掲載した，付録では control 課題のみ掲載する．

B.1　タスク A control 課題

```
public static int function(){
    String long_str = "aabcabc";
    String short_str = "ab";
    int result = 0;

    for(int i = 0; i < long_str.length()-1; i++){
        String tmp_str = long_str.substring(i,i + short_str.length());
        if(short_str.equals(tmp_str)){
            result++;
        }
    }
    return result;
}
```

B.2　タスク B control 課題

```
public static int function(){
    int data1[] = {4,2,1,1,1,5,9,6,1,6};
    int data2[] = {4,3,3,3,1,0,4,5,0,8};
    int a,b,i;

    b = 0;
    for(i=0; i<10; i++){
        a = data1[i];
```

```
        b = b + data2[a];
    }

    return b;
}
```

B.3 タスクC control課題

```
public static int simpleSelectionSort(){
    int data[] = {6,6,4,8,3,4,0,7,1,4};
    int a, b, i, j, k;
    for (i=0; i<10; i++){
        a = data[i];
        k = i;
        for (j=i+1; j<10; j++){
            if (data[j] < a){
                a = data[j];
                k = j;
            }
        }
        b = data[i];
        data[i] = data[k];
        data[k] = b;
    }
    return data[7];
}
```

B.4 タスクD control課題

```
public static double function(){
    int data[] = {2,7,7,8,2,4,8,6,4,0};
    int a,i;
    double b = 0.0;

    for(i=0; i < 10; i++){
```

```
        b = b + data[i];
    }
    b = b/i;
    return b;
}
```

C　三章の実験で用いたアンケートの内容

実験アンケート

1. あなたは日ごろ，どの位の頻度でプログラムのソースコードを読みますか？

　1. 毎日 2. 数日に一回程度 3. 週に一回程度 4. 月に一回程度 5. 数か月に一回程度

2. あなたのプログラミング経験はどの程度ですか？

　1. 一年以内 2. 二年以内 3. 三年以内 4. 5 年以内 5. それ以上

3. 普段使っているプログラミング言語は何ですか？(複数筆記可)

4. あなたは左利きですか？　右利きですか？

　1. 左利き 2. 右利き

5. あなたは掛け算の九九をすべて覚えていますか？

　1. はい 2. いいえ

6. （各工程終了後）提示されたプログラムの理解がどの程度難しかったか、五段階でお答えください。

	←簡単だった		ニュートラル		難しかった→
一個目	1	2	3	4	5
二個目	1	2	3	4	5
三個目	1	2	3	4	5

7. 他に感想、意見などがあれば以下に書いてください (自由記述)

D　四章の実験で用いたアンケートの内容

実験アンケート

1. あなたは日ごろ，どの位の頻度でプログラムのソースコードを読みますか？
   1. 毎日 2. 数日に一回程度 3. 週に一回程度 4. 月に一回程度 5. 数か月に一回程度

2. あなたのプログラミング経験はどの程度ですか？
   1. 一年以内 2. 二年以内 3. 三年以内 4. 5 年以内 5. それ以上

3. 普段使っているプログラミング言語は何ですか？(複数筆記可)

4. あなたは左利きですか？　右利きですか？
   1. 左利き 2. 右利き

5. 年齢をお答えください。

6. 性別をお答えください。

7. （終了後）感想、意見などがあれば以下に書いてください (自由記述)