

保 存 用



土木研究所資料  
第 3 4 3 7 号

# 土 木 研 究 所 資 料

## 建設機械の制御システムに関する研究

平成 9 年 2 月

建設省土木研究所  
材料施工部機械研究室

Copyright © (1997) by P.W.R.I.

All rights reserved. No part of this book may be reproduced by any means, nor transmitted, nor translated into a machine language without the written permission of the Director General of P.W.R.I.

この報告書は、土木研究所長の承認を得て刊行したものである。したがって、本報告書の全部又は一部転載、複製は、土木研究所長の文書による承認を得ずしてこれを行ってはならない。

# 建設機械の制御システムに関する研究

|       |       |    |        |
|-------|-------|----|--------|
| 機械研究室 | 室長    | 杉山 | 篤*     |
|       | 室長    | 吉田 | 正      |
|       | 主任研究員 | 石松 | 豊**    |
|       | 研究員   | 高津 | 知司***  |
|       | 研究員   | 宮武 | 一郎**** |
|       | 研究員   | 三村 | 茂男     |

\*現 水質源開発公団工務部機械課長

\*\*現 近畿地方建設局道路部機械課長

\*\*\*現 中部地方建設局名古屋国道工事事務所機械課長

\*\*\*\*現 四国地方建設局道路部機械課調査係長

## 要旨

一般に建設機械による施工においては、出来高や効率がその運転の善し悪しに左右されるので、熟練オペレータによる施工が望まれる。

今後の熟練オペレータの不足に対応して、操作に未熟練なオペレータでもある程度良好な施工を可能とするような建設機械のイージーオペレーションシステムの開発が必要であると考えられる。

本資料は、建設機械のイージーオペレーションシステム開発を将来的な構想として、その基本的な技術について検討した結果をとりまとめたものである。

【キーワード】：ブルドーザ、イージーオペレーション、シミュレーション技術

## ま え が き

近年の経済的・技術的な発展によって我が国の社会資本は増加しつつあり、その投資規模は高い水準で移動してきている。しかしながら、その整備水準は欧米諸国に比べて立ち後れており、経済規模に応じた豊かさを国民が実感できるレベルには未だ達していず、快適かつ安全な質の高い国民生活の実現が求められている。また、来るべき21世紀に向けてますます高度化、多様化する社会資本に的確に応えるため、環境の保全・創造、豊かな生活の実現、自然災害に対して安全な国土の建設、魅力的で活力ある地域社会の構築などの実現が基本的に求められている。このような社会資本整備を円滑に推進するにあたっては、建設産業における構造的な労働力不足、低い労働生産性と安全性等の改善を図ることが重要な課題である。また、建設労働災害の防止や作業環境の改善等により、より安全で快適な魅力ある産業として労働力を確保することが重要な課題である。

これらの課題を解決するための一方策として、建設機械の自動化・ロボット化の研究開発が進められている。その中でも、ブルドーザなどの汎用土工機械は、作業の対象物が土などであり、その操作にかなりの熟練を要するものとなっている。本資料は、建設機械のイージーオペレーションシステム開発を将来的な構想として、その基本的な技術・理論及びブルドーザ作業のコンピュータシミュレーション技術についての検討結果をとりまとめたものである。

なお、本報告書の内容は、土木研究所機械研究室にて平成3年度～平成7年度の5カ年にわたり実施された、研究課題「建設機械のAI制御システムに関する研究」により得られた成果である。

また、本研究の成果をシミュレータ開発者などが広く利用できるようにするため、本研究の成果であるブルドーザ操作シミュレータのプログラムを本資料に添付した。

# 目 次

まえがき

|                             |    |
|-----------------------------|----|
| 1. 研究目的及び方法                 | 1  |
| 2. 研究結果                     | 2  |
| 2.1 建設機械の制御手法についての検討        | 2  |
| 2.2 AIを用いたブルドーザの制御システムの検討   | 10 |
| 2.2.1 開発プロセスの提案             | 10 |
| 2.2.2 技術的課題                 | 10 |
| 2.3 ブルドーザ・オペレーション・シミュレータの製作 | 18 |
| 2.3.1 製作の目的                 | 18 |
| 2.3.2 製作に必要な技術・理論           | 18 |
| 2.3.3 機械－地盤系特性の組み込み         | 18 |
| 2.3.4 システム構成                | 27 |
| 2.3.5 シミュレーションプログラムの基本機能    | 32 |
| 2.3.6 シミュレータの操作手順           | 36 |
| 3. まとめ                      | 39 |
| おわりに                        | 40 |

## [添付資料]

- ブルドーザの挙動計測実験データ
- ブルドーザ・オペレーション・シミュレータ プログラムリスト

## 1. 研究目的及び内容

現在、種々の建設ロボットと称する自動化された建設機械の開発報告がなされている。建設機械のうちで、土工機械といわれる分野についても、自動化や遠隔操作システムなどの多くの開発努力がなされている。一般に建設機械による施工においては、出来高や効率がその運転の善し悪しに左右されるので、熟練オペレータによる施工が望まれる。しかし、熟練オペレータの育成には数年の期間を要するといわれている。そこで、今後の熟練オペレータの不足への対応として、操作に未熟練なオペレータでもある程度良好な施工を可能とするような建設機械のイメージオペレーションシステムの開発が有効と考えられる。

建設機械の操作においてはオペレータ自身が最適と思われる操作方法を実行していると考えられる。機械にこの種の情報処理機能を付加するためには、現在のオペレータの思考・判断のもととなる操作上の経験則を建設機械の制御システムに取り入れる必要がある。また、オペレータ毎に操作方法や作業の組み合わせなどに差異があり、作業の推定や支援動作の選択に固定した判断機構をもたせることは出来ないで、学習機能を有する情報処理機構を応用することが考えられる。本研究は、学習機構を有するAIを用いたイージーオペレーションシステムについて、その基本的な技術の提案を目的として実施した。

研究内容は以下の通りである。

- (1) AI (Artificial Intelligence), ファジィ, ニューラルネットを応用した制御方法についての調査
- (2) ニューラルネットを応用したブルドーザの制御システムの開発プロセス及び技術的課題の検討
- (3) ブルドーザ・オペレーション・シミュレータの開発検討, 及び試作

## 2. 研究結果

### 2.1 建設機械の制御手法についての検討

#### 2.1.1 ニューロ・ファジィ・AIの制御への応用<sup>\*)</sup>

本研究では、これまでニューラルネットをAIの一部として述べてきたが、1994年5月に発行された計測自動制御学会の「ニューロ・ファジィ・AIハンドブック」の中では、人間の脳における情報処理の原理、「知」の原理に対して、脳のソフトウェアを扱かうAI（人工知能）、パターンを扱かうニューロ、そしてその中間に位置するファジィという3つの工学的アプローチがとられてきたとして、これらを区別した取り扱いがなされている。なお、ニューロ、ファジィ、AIとニューラルネットを分けて制御へのAIの応用を述べる。なお、ニューロ、ファジィ、AIを用いて単独での問題の解決は容易でなく、それぞれの特徴を生かして統合したアプローチが必要であるというのが、現在の認識である。

##### 1) 人工知能 (artificial intelligence)

人工知能とは、人間の知能の理解及び機械による知能の実現を目標とする研究領域をいう。人工知能の研究領域内において、応用人工知能に位置づけられる代表的な研究課題として、自然言語処理、音声理解、エキスパートシステム、コンピュータビジョン、ロボティクスなどがあげられる。

近年、制御機能の多様化・高付加価値化に対する要求が高まり、取り扱うべき問題の質・量的に拡大してきている。このため、従来の固定的なアルゴリズムの自動制御システムに対して、専門の運用者の持つ経験的知識を導入し、人間の高度で柔軟な知的活動の一部を取り入れた制御システム「制御型知識システム」を構築することが求められ、この開発が行われている。

制御機能のレベルは、油圧サーボメカニズムに代表される「直接制御」、制御対象の状態を評価する指標が与えられてそれが最も望ましい値になるように最適計算を実行し操作量を決定する「最適制御」、目的が多目的であり数値的な評価関数として表現することが困難な要素も含まれるものに対する「適応制御」の3つがある。制御型知識システムの応用としては、最適制御、適応制御が主である。

#### 2.1.2 ファジィ制御

コンピュータや制御理論の発展により自動化範囲の拡大が進んでいるが、対象プロセスの非線型性のため制御範囲が狭い、モデルパラメータの同定がむずかしく制御理論を適用できないなどの理由により、オペレータの介入を必要とするシステムが少なくない、そのような状況において、熟練オペレータの操作に関する知的活動を制御規則でモデル化し、熟練オペレータと同様な知的制御をコンピュータで実現しようとする試みのひとつがファジィ制御である。ファジィ制御には種々の方式が提案されているが、次の3つの方式が多く採用されている。

##### 1) 状態評価ファジィ制御方式

制御対象のシステム状態を判断し、過去の経験からこれぐらいの操作を行ったらよいという、多くの情報に基づく制御方式である。外乱要因に基づくフィードフォワード制御や過去の操作結果に基づくフィードバック制御を協調させた制御が可能である。また、対象システムの構造が不明確でもモデル化が不要なため、適用が容易である。

本制御方式は従来の制御方式では自動化がむずかしい、プロセスの数学モデルの作成がむずかしいなどによって運転者の介入が必要な、窯業、水処理、鉄鋼、電力、化学、食品などのプロセスに多くの適用がなされている。

## 2) 予見ファジィ制御方式

現在のシステム状態でこの操作をしたならば、過去の実験からこのような動きをするはず（制御結果の予見）であり、制御対象システムの制御目標を満足する制御ができそうという判断に基づく制御方式である。現在および将来の制御目的をモデルにより予見し、それらを多次元的に評価した制御が可能である。

本制御方式は、対象システムの動特性が比較的明確であるが、その制御目的が目標パターンへの追従性だけでは満足できない対象に有効である。適用分野として交通システム、運搬システム、サービスシステムなど、状況により制御目標が変わるシステムの自動化が挙げられる。

## 3) ファジィ適応制御方式

制御アルゴリズムを変化させる制御方式である。制御規則やメンバーシップ関数を適応させることにより、制御規則が少なく済み、設計が容易である。また、特性変動に対して安定な制御範囲を広くすることができる。

本制御方式は、制御対象の特性がわかっており、その特性が操業状態により大きく変動するようなプロセスに効果がある。例えば、プラントのゲインや時定数あるいは遅れ時間が操業状態により大きく変動するプロセスや、状況により判断内容が変化するプロセスに有効である。

下水ポンプ所のポンプ、阻止扉の協調制御の事例がある。

### 2.1.3 ニューラルネットの制御への応用

ニューラルネットの応用面から見た利点は、パターン認識と情報制御／処理を同時にかつ並列に実行できるという点にある。また、ニューラルネットは未処理データの観察をもとに因果関係を学習したり、因果関係を用いて推論を行ったり、行動や運動に関する知識処理操作を実時間で実行できる可能性を装備している。これらは将来、制御の観点から行動運動の計画、それらの最適化、学習制御などに、知識の学習、その因果関係についての推論などに発展すると思われる。

ニューロンモデル（ユニット）を階層的に配置したニューラルネットを階層ニューラルネットと呼ぶ。階層ニューラルネットでは、与えられたサンプルパターンから学習によってモデルを非線型関数として表現する。この非線型関数の生成機能は、限られた数のサンプルパターンからその母集団を関数としてモデル化することに応用される。これは、一種の汎化能力であり、非線型性や冗長自由度を有する対象の処理に有用なものとなる。また、この非線型関数の生成機能はパターン識別にも応用される。

こうした機能を制御応用の立場から見ると、強い非線型性を持ち従来技術でモデル化が困難であった制御対象に対してその順モデルや逆モデルを学習で獲得するのに有用と考えることが出来る。順モデルとは、対象と同一の入出力特性を持つモデルである。逆モデルとは対象の入出力特性を逆にしたモデルである。

ニューラルネットの制御分野への重要な応用として、逆モデルの獲得がある。これらは多くの場合、

制御における学習の目的が実際の出力結果を望ましい出力結果に一致させるような制御信号を見つけ出すことにあるからである。逆モデル獲得法には、直接法、誤差フィードバック法、順逆モデル法がある。

#### 2.1.4 併用

ファジールールとニューラルネットワークの融合／併用モデルの研究は1970年代に始まっている。この研究は次のように分類することができる。

##### 1) 融合モデル

###### ① ファジィの研究（ニューロ的ファジィ）

- ・ニューラルネットワーク的学習を導入したファジールール
- ・一部をニューラルネットワーク化したファジールール

###### ② ニューラルネットワークの研究（ファジィ的ニューロ）

- ・ファジールールの構造を持ったニューラルネットワーク
- ・一部をファジィ化したニューラルネットワーク

##### 2) 併用モデル

###### ① 直列に配置

- ・ニューラルネットワークの出力をファジールールに入力（ニューロファジィ）
- ・ファジィの出力をニューラルネットワークに入力（ファジィニューロ）

###### ② 並列に配置（ニューロ／ファジィ）

###### ③ 対象ごとに個別に配置（ニューロ&ファジィ）

#### 2.1.5 建設機械の制御の事例

現在の建設機械は、機構（mechanism）と電子技術（Electronics）の合成語であるメカトロニクス（Mechatronics）化により、その制御性、操作性を大幅に改良している。

建設機械のメカトロニクス化の事例を次に2例、述べる。

##### (1) マイコン制御式大型油圧ショベル

鉱山で採掘に使用される油圧パワーショベルは大型化の一途をたどっている。このような大型化に対して、作業機を駆動する油圧系は高圧化に限界があり、ポンプ台数を増加させて大型化に対処しなければならず、油圧回路が複雑になる。これに伴い、従来のオペレータによるマニュアル操作では、機構上、操作力上、運転が難しくなり、掘削作業効率など掘削能率の低下を招く。

このような一連の課題を解決するため、マイクロコンピュータ制御による油圧制御システムが開発された。このシステムはマイコンをコントローラに用い、操作レバーの位置信号センサ（角度、圧力）信号が入力され、電気油圧サーボ弁や電磁弁に操作信号を出力するものである。これらの技術の一部はより小形の油圧ショベルに展開され、操作性の向上に寄与している。マイコンによる主な制御内容は次のものである。

###### ① ファインコントロール

軽い操作力を可能とするための、レバーストローク、作業機角度に対応した油圧流量制御

② パワーロスの低減

コントロールレバー中立時、油圧リリーフ時に油圧ポンプの吐出力を少なくして、パワーロスを低減するポンプ制御

③ エンジン出力の効率化

油圧ポンプとバルブの組合せをコントロールして、エンジン出力を効率的に活用するエンジン出力制御

④ バケット制御

ブーム上昇中のバケット傾斜角度とアーム下降中のバケット掘削角度の自動補正など

(2) 大型ブルドーザのメカトロニクス

ブルドーザのオペレータの疲労軽減のため、操作力の低減をねらいとして操作機構に、電子油圧制御方式が採用されている。また、大型ブルドーザにおいては特に、オペレータの熟練度により大きく作業効率の差があるので、この改善のため、常に最大効率での作業ができるようにブルドーザの自動負荷制御機構が開発されている。

これは、手動操作用の電気操作レバーと、対地速度を計測するマイクロ波ドップラ式車速センサなど各種センサと、電気油圧アクチュエータとマイコン（コントローラ）より構成されている。対地車速センサ、トルコン出力軸回転センサ、ミッション変速段位置検出センサ、車体の前後傾斜角を検出する傾斜角センサなどの情報は、マイコンに入られる。マイコンには車両性能が記憶され、土質条件の把握のため、数回学習のための作業を行うと、各種センサからの情報にもとづき、その作業土質に最も適した車速が算出される。オペレータが、ブルドーザの掘削高さを決め、手動から自動掘削に切り替わると、車速センサより計測される実車速に応じて、常にその作業土質に適した車速になるよう、ブルドーザの負荷が自動的にコントロールされる。すなわちブルドーザの上下操作は不要となり、しかも最適条件での運転が可能となるため、操作性が向上し、未熟練オペレータでも、ベテランの作業効率があげることができるなど、効果は大きいとされている。

2.1.6 ブルドーザの制御

(1) ブルドーザの機構

本研究の対象とする建設機械のひとつであるブルドーザの制御対象となるのは、次のものである。このような単純な組合せであるが、作業時はブルドーザと走行系の相互作用があり、その運転には熟練を要する。次にその内容について述べる。

① 左右のクローラの起動・停止、速度制御

② ブルドーザの上昇・下降

③ ブルドーザのチルト（ない場合もある）

ブルドーザの作業の中でオペレータ側から操作できるのは、走行速度とブルドーザの操作（上昇・下降速度の調整を含む）の2要素である。これらの組み合わせにより、掘削を例にとると図2-1のように様々な掘削形状を実現することができる。例えば、図2-1のaの実線を標準的な走行速度とブルドーザ操作速度とすると、a\*はブルドーザの操作を更に長時間行ったことと考えられ、bは走行速度が速

いかブルドーザの駆動速度が遅いかのどちらかと判断される。

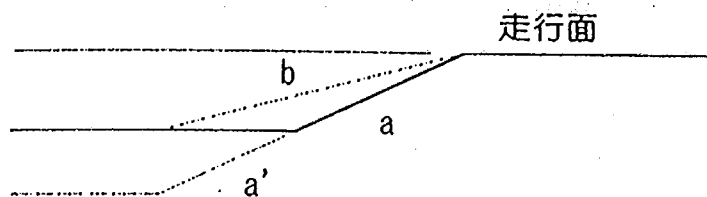


図2-1 走行速度とブレード操作の組合せ

しかし、更に掘削操作を吟味すると、掘削開始とともにエンドエフェクターであるブレードの位置を下げて土中へ貫入させることになるが、暫くするとブルドーザのクローラ部分はブレードで掘削した部位へ進入することになり、これによって車体自体が姿勢変化する。すると車体に固定されたブレード先端が入ってしまうことになる。また、同時に掘削抵抗が急激に増大し、土質によってはクローラのスリップという事態も生み出す。

このようにブルドーザは、ブレードの上下という1自由度を有する作業機が付属するのみであるが、走行・作業を含めた制御システムとしての特性は「非線型システム」として取扱を考えねばならない。また、ブレードの土中への深さやブレードの前に溜まった土により生じる掘削時の作業抵抗の変化も非線型な特性を有する。このような非線型な特性を有したシステムの制御の最適化を図るのは、困難であり、これがブルドーザの自動制御の実用化を難しくしている大きな要因と考えられる。

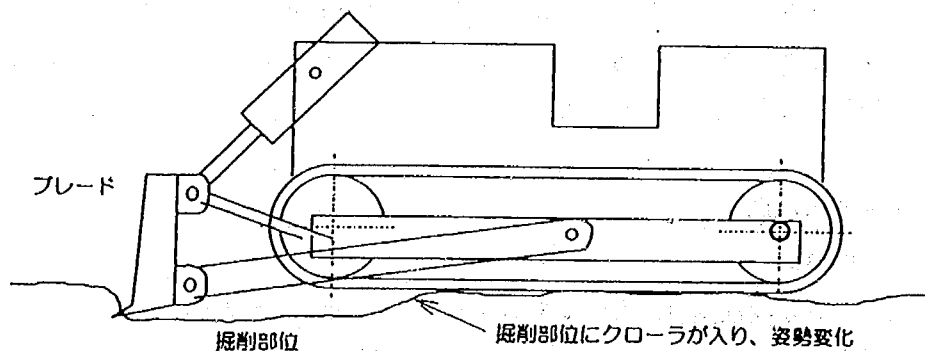


図2-2 ブルドーザの作業に伴う姿勢変化

また、ブルドーザによる作業は、掘削、押し土、盛土、整地などがある。この中で整地作業については、ブルドーザ（排出土板）に受光器を取りつけてレーザーレベルによりブルドーザの高さを自動制御するものが実現されている。また、出来型が問われない場合では上述の大型ブルドーザの自動負荷制御機構も適用が可能である。しかし、ブルドーザの基本作業である掘削、押し土、盛土の作業を組み合わせた内容には、これらの自動制御で対応することは困難であり、オペレータの技量が出来上がり品質や

施土速度に大きく影響を及ぼすことになる。

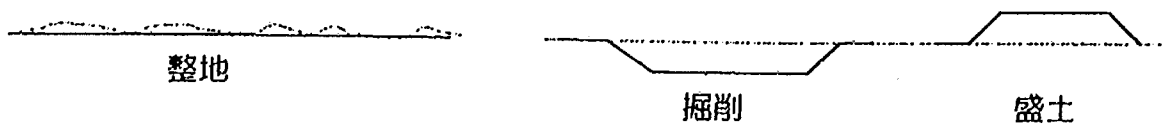


図2-3 ブルドーザによる基本作業

t2はブレードが所定の深さになったため、ブレード操作レバーを中立にした時点に相当する。これにより、ブレード駆動シリンダが停止する。

t3はブルドーザが掘削した部位に入ったために車両が傾き始め（ピッチ角に対応）、これによってブレードでさらに深く掘削するのを防ぐために、ブレード操作レバーを上げの動作開始する時点に相当する。なお、t1からt3の時間遅れはブレード先端から重心までの距離を走行速度で除したものに相当する。

t4はブルドーザの傾きがピークを過ぎる時点に相当し、ブレードを掘削深さに沿わせるようにまた、ブレード操作レバーの下げの動作を行う事になる。

t5はブルドーザが掘削部位に完全に入った履帯下面とブレード先端がほぼ、同じ高さで押土的に掘削を進めている状態である。

このようにブレード操作レバーによる入力、遅れてブルドーザの姿勢変化に影響を与え、また、ブレード操作レバーの操作入力を要求する結果となる。このようにそこで行われる（あるいは行うべき）ブレード操作の内容は明確である。しかし、t1～t2に行われたブレード操作の操作量を記憶しておいてt3～t5の操作量を決定するというのは人間にとって不得手な処理であり、また、ブルドーザの姿勢変化を補正も要求されることは、さらにこれを難しいものとしている。そこでオペレータによるブルドーザの運転では、t3～t5の操作はt1～t2の操作量に関係なく経験に基づき行われているのが実態と考えられる。

### 2.1.7 ブルドーザの制御方式の検討

#### AI制御の概要

ブルドーザの制御を考える上で、制御システムの階層構造わけ、つまり、「全体計画レベル」、「作業単位の制御レベル」、「ブレードの制御レベル」が必要と考えられる。

#### ① 全体の計画

全体の計画は総作業量、例えば掘削を何cmの深さで何m実施するのが作業指示であるが、これに対してどのようにブルドーザを運転させたらよいか計画する部分である。この計画は土の状態に応じて適時、変更される部分である。

この部分の目標達成の認識を自動化することは、レーザ測距機などを用いたブルドーザの位置計測シ

システムのデータを入力してリアルタイムで処理することが可能と考えられる。なお、外界の変化に対応したタスクプランニング（動的プランニング）については、AIの領域であると言える。

## ② 作業単位の制御

作業単位の制御は、押し土、掘削、盛土に分類された作業単位に適合した制御を行なうものである。

### 〔押し土〕

ブレード前面に土を溜めて、水平に所定の場所へ移動するものである。ブレードから土のこぼれが多くなったり、負荷が大きく履帯が空転を生じるようであればブレードを若干上げて、負荷を軽減して走行する。

### 〔掘削〕

地中にブレードを食い込ませ、掘削するものである。負荷が大きく履帯が空転を生じるようであればブレードを若干上げて、負荷を軽減して作業する。

掘削の運転は大きく次の4段階に分けられる。

a. まず、掘削開始時はブレードを下げる操作である。このブレードの下げ加減は、掘削深さという目標値と土の硬さを勘案して行なわれる。

b. 続いてブルドーザは自ら掘った穴に入り込むことになり、姿勢変化が生じる。

この時点でブレードを固定した状態に保つと深く掘削しすぎるため、ブレードを上げる必要がある（望むべくはブルドーザが水平にあった状態でのブレードの深さと同じ深さにブレードの下端の位置があるように姿勢変化中、制御されること）。

c. そしてブルドーザが自分の掘った穴に完全に入った状態では、ブレードの下端の位置は掘削土によって生じる負荷を考慮しなければ履帯の下端と同じ高さにし、走行することになる。しかし、実際はブレードの前に掘削土が滞留して走行負荷が増え、クローラに空転を生じさせるため、空転を生じた段階でブレードを上げて押し土の状態に入る。

d. 押し土に移行する段階では、自らが掘った穴を出るため、姿勢変化を生じる。この姿勢変化に合わせてブレードの高さを制御して押し土すべき土が余分にこぼれないようにするのが理想である。

### 〔盛土〕

押し土を繰り返して、所定の高さに土の山をつくることである。

以上のように実際のブルドーザの作業は、ブルドーザの姿勢、ブレードからの土砂のこぼれ具合、履帯の空転の状況等、様々な状況を即座に判断して行う必要があり、その制御を難しいものになっている。

ブルドーザのAI制御を取り入れた運転は次のものが想定される。なお、運転席に取り付けられるAI自動制御系の機器は、ブレード操作レバーに取り付ける掘削操作用のボタン、ブレードの先端の位置（掘削深さ）の表示器である。

a. ブルドーザを走行させて掘削開始位置となったら、ブレード操作レバーの掘削操作開始ボタンを押す。これにより、AI自動制御がスタートし、ブレードが下降する。

b. オペレータはブレードの先端の位置（掘削深さ）を運転席に取り付けた表示器で確認し、所定の深さになったらブレード駆動停止ボタンを押す。

c. ブルドーザが更に走行して、自らの掘った部分に入ってブルドーザの姿勢変化を生じたら、ブル

ドーザの走行速度と姿勢変化の速度を勘案した速度でブレードを自動的に上昇させ、ブルドーザが完全に自らが掘った部位に入った状態でブレードが走行面と同じ高さとする。

- d. ブレード全面に土砂が溜まって走行抵抗が増えた場合は、オペレータがブレード操作レバーを操作することで対応する。なお、上述のブレード操作支援システムはオペレータがボタン以外のブレード操作レバーの操作を行うことにより、自動制御モードを外れるものとする。

### ③ ブレードの制御

現在のブルドーザの操作系は不感帯があるなど、自動制御の対象としては極めて取り扱いにくい対象である。また、このように1台1台異なる操作系に1:1の形で自動制御システムを構成した場合、別のブルドーザでは使えないという事態を招く恐れがある。そこで特性を標準化してコンピュータによる自動制御のインターフェースの役目も果たす制御のサブシステムであるブレードの制御が必要となる。

具体的には、ブレードの絶対位置信号を与えてブレードの位置を自動制御するサーボ制御である。

AI制御の適用が一番容易なのは、上述③のレベルであるが、適用における有効性や効率性を考えるならば、ここから②に近づけることを想定する必要がある。

また、制御システムを考える上で目標値を明確にする必要があるが、従来の事例では対象物が静止しているか動いているかに関わらずその位置であるのに対して、ブルドーザの目標とするものは、出来型となる。

出来型自体が最終的な目標となることは、NC工作機械を例とするように問題はないと考えられる。しかし、NC工作機械を利用して1本のバイトで全ての成型を行なえることは少なく、一般に、何本ものバイトを自動交換して最終的な形状を得ることになる。そしてそれぞれのバイトに対して、加工に対する目標値が与えられることになる。

これと同様にブルドーザの制御においても、それぞれの加工工程；例えば掘削、押し土、盛土；の目標値が設定されて実際の制御を行なうことになる。また、1回の走行で最終的な出来型を得られることは、殆ど不可能なことから、加工工程をどの様に組合せるかの作業計画のレベル、また、地山の硬さなどに対応して即座に掘削深さを変えるなどの操作が必要となる。これは、例えば掘削という小単位の制御で着目すると制御の目標値の変更がその都度、行なわれることを意味する。そして、この小単位のレベルの目標値の変更により、全体の加工工程；例えば往復回数などが極めて自在に変更されることになる。

### 【参考文献】

- 1) (社)計測自動制御学会：「ニューロ・ファジィ・AIハンドブック」，1994年5月，オーム社
- 2) 石坂誠一監修：「メカトロニクス 脚光をあびる複合技術」，1985年10月14日，通商産業調査会

## 2.2 AIを用いた部Rの制御システムの検討

### 2.2.1 開発プロセスの検討

ブルドーザを対象として、AIを応用した制御システムの構築方法について検討した。

図2-4にその結果を示す。

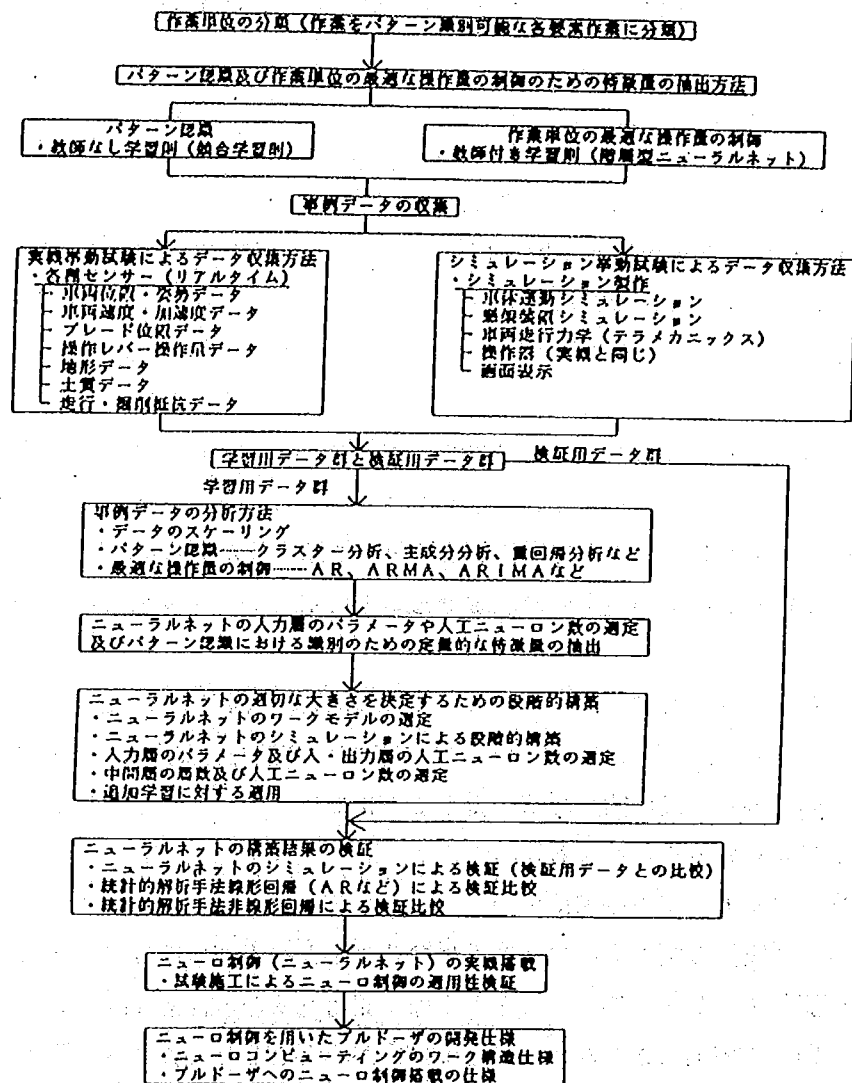


図2-4 ブルドーザのAI制御システムの開発プロセス

### 2.2.2 技術的課題

#### (1) ブルドーザの運転と制御

ブルドーザによる作業は、掘削、押し土、盛土、整地などがある。この中で整地作業については、ブレード（排出土板）に受光器を取りつけてレーザーレベルによりブレードの高さを自動制御するものが実現されている。これに対して掘削、押し土、盛土の作業はオペレータの技量が出来上がり品質や施工速度に及ぼす影響が大きい、制御として実用化されているものがない。

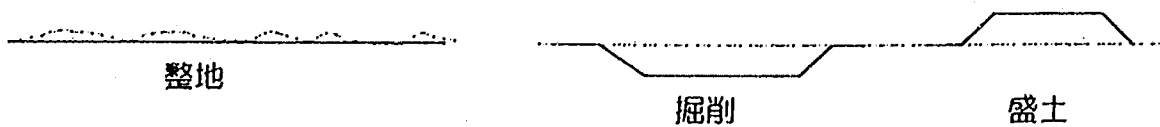


図2-5 ブルドーザによる作業

ブルドーザによる掘削に着目した場合、掘削開始とともにエンドエフェクターであるブレードの位置を下げて土中に貫入することになるが、やがてブルドーザのクローラ部分が掘削した部位へ進入することから、車体が姿勢変化してブレードの土中への貫入深さに影響を与える。このようにブルドーザは、ブレードの上下という1自由度を有する作業機が付属するのみであるが、走行・作業を含めた制御システムとしての特性は非線型システムとして取扱を考えねばならない。また、ブレードの土中への深さやブレードの前に溜まった土により生じる掘削時の作業抵抗の変化も非線型な特性を有する。このような非線型な特性を有したシステムの制御の最適化を図るのは、困難であり、これがブルドーザの自動制御の実用化を難しくしている大きな要因と考えられる。

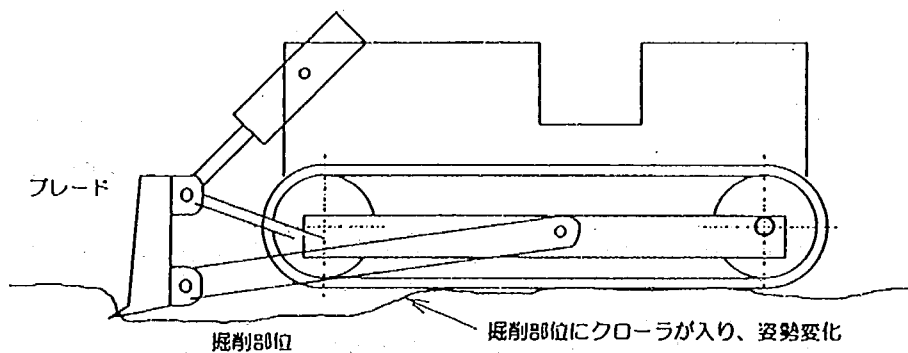


図2-6 ブルドーザの作業に伴う姿勢変化

## (2) ニューラルネットワークとブルドーザのAI制御

「ブルドーザのAI制御の研究」は、非線型の制御対象であるブルドーザをAI (Artificial Intelligence, 人工知能) の手法を用いることによって、非熟練者でも容易に掘削などの熟練のいる作業をできるようにすることを目的とするものである。具体的にはブルドーザのうまい操作方法を学習し、この操作方法をオペレータのブルドーザの制御支援システムとして利用できるようにするものである。つまり、後述の図2-8で示すニューロンのモデル中の非線型処理部をブラックボックスとして取り扱い、熟練者の入力に対する出力の関係を明らかにすることにより、制御支援システムを構築しようとするアプローチである。

近年、生物の神経細胞 (ニューロン) の本質的な機能をモデル化した要素を多数組み合わせ、ある高次の機能を実現するニューロコンピューティングの研究、そしてニューロコンピューティングをソフ

ト的に実現したニューラルネットワークの研究が進められている。久間和生・中山高著の「ニューロコンピュータ工学」(1992年, 工業調査会) からニューラルネットワークに関する部分の抜粋を次に示す。

『ニューラルネットワークは図2-7に示すように、ニューロンとよばれる多数の演算素子と、それらを結合するシナプスと呼ばれるアナログ量の分散記憶素子で構成される。このアナログの結合強度をシナプス荷重あるいはシナプス結合強度と呼ぶ。シナプス結合には、正(興奮性)と負(抑制性)の結合がある。

ニューロンは多入力/1出力型の素子で、図4に示すように、入力信号の積和演算と飽和型の非線型入出力特性を与えるモデルがよく用いられる。すなわち、結合されている他の多くのニューロンの興奮状態 $v_j$ がシナプス荷重 $W_{ij}$ を付けて荷重加算 $\sum W_{ij}v_j$ され、非線型入出力特性に従って自分の興奮状態(出力)が決まる。荷重加算された値 $\sum W_{ij}v_j$ を膜電位 $u_i$ と呼ぶ。シナプス荷重は、使用するニューラルネットワークモデルによって、あらかじめ計算によって与えられる場合と、学習によって決められる場合がある。』

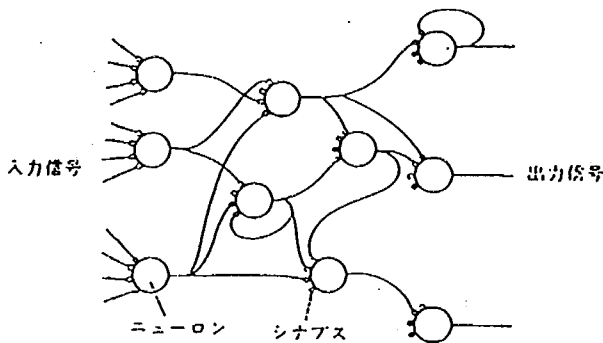


図2-7 ニューラルネットワークの構造

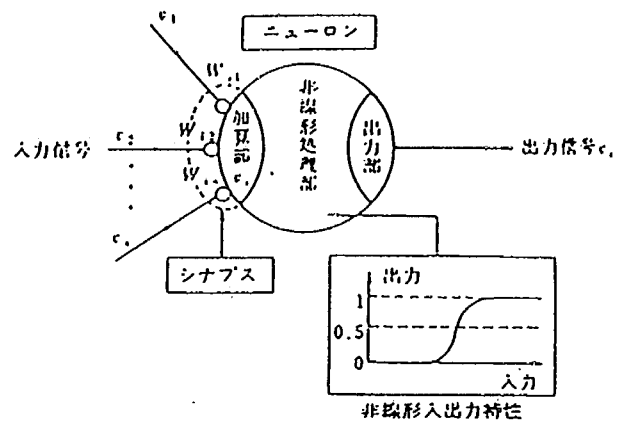


図2-8 ニューロンのモデル

そこでブルドーザの掘削などの個々の操作(入力)に対するブルドーザの挙動(出力)を結びつけるブラックボックスの特性が明らかになれば、これをブルドーザに付加することで、高度のブルドーザの運転操作が非熟練者でも可能になることが期待される。

ニューロコンピューティングの代表的なモデルとして、連想回路、誤差逆伝播学習、競合ネットワーク、最適化問題のニューラル解法、ボルツマン機械等がある。このうちで誤差逆伝播学習は、数学的には、与えられた入出力パターンを用いて、そのような入出力関係をもったニューラルネットワークを一種の最急降下法によって求めるもので、扱える工学的問題が非常に多いことから、応用上、価値が高く、最も有名である。

多方面で応用されている誤差逆伝播法において、ダイナミクスあるいはその最も単純な場合に相当する系列の学習においては、訓練パターンの提示順序自体に意味がある。そしてダイナミクスを取扱うには、過去の履歴を何らかの形でネットワーク内に保持することが必要である。履歴の保持は単純な階層型ネットワークでは不可能であり、大別すると、階層型ネットワークにフィードバック結合を追加したものと、相互結合型ネットワーク(リカレントネットワーク)がある。

階層型ネットワークのうち、ダイナミクスの学習を対象にしたものにコンテキストモデルとバッファ

ネットワークがある。コンテキストモデルは、入力層、隠れ層、出力層の他に、入力層、隠れ層、出力層の履歴を保持する状態層を導入したもので、この状態層を用いることで過去の履歴の反映した系列の学習を可能としたものである。これに対して、バッファネットワークは、最近の $n$ 時点の系列  $\{x_i, x_{i-1}, \dots, x_{i-n+1}\}$  を入力バッファとし、別の系列の現在地 $y_i$ を出力とし、両者を学習し、このネットワーク構造により、時間的認識をニューラルネットワーク構造が得意な空間的認識に変換するものである。

ここで、 $y_i$ として次時点の入力 $x_{i+1}$ を用いると、過去の系列  $\{x_{i-n+1}, \dots, x_{i-1}, x_i\}$  に基づいた次時点 $x_{i+1}$ の予測が行える。これは時系列モデルでは統計的解析手法のひとつのAR（自己回帰）モデルに相当する。また、入力の過去の系列のみならず出力の過去の系列も入力側のバッファに入れると、入出力の過去の系列から次の時点を予測できるネットワークモデルとなる。これは時系列モデルでは統計的解析手法のひとつのARMA（autoregressive moving average model, 自己回帰移動平均）モデルに相当する。

ARMAモデルは、定常時系列モデルの中でもっともよく用いられているもので、このモデルを用いて時系列を解析する具体的方法は Box and Jenkins (1970)により述べられた。

その方針は「時系列のはっきりとした変動パターン（非定常性）はデータの（比較的簡単な）変換によって取り除き、残されたそれほどはっきりしない変動パターン（定常性）をARMAモデルで記述する」というものである。この方法は Box-Jenkins 法と呼ばれており、数学的道具立てが比較的軽く、理解しやすい、実際の時系列データの解析に用いられてその有効性が報告された等から、時系列解析の普及に大きく貢献した。

しかしながらARMAモデルは、前述のように定常時系列モデルを対象としたものであり、ブルドーザの運転操作のように定常時系列モデルと言い難いものに対して、有効なデータが得られるかは、疑問視されるところである。

よって、ARMAはニューラルネットを構築する際の入力層や人口ニューロンの選択及びパターン認識の際の各要素作業の特徴を抽出するための一手段として用いることが考えられる。そして、これら各パラメータの相関や特徴を定量的に抽出した結果を用いて、AI構築を行っていくことが考えられる。

### 3) ブルドーザへのニューラルネット制御システムの実装

ブルドーザの制御は、前項に示した制御システムの階層構造の「全体計画レベル」、「作業単位の制御レベル」、「ブレードの制御レベル」での検討が考えられる。次にその内容を示す。

#### ① 全体の計画

全体の計画は総作業量、例えば掘削を何cmの深さで何m実施するのが作業指示であるが、これに対してどのようにブルドーザを運転させたらよいか計画する部分である。この計画は土の状態に応じて適時、変更される部分である。

この部分を自動化することは、レーザ測距機などを用いられている。ブルドーザの位置計測システムのデータを入力してリアルタイムで処理することが可能と考えられる。なお、費用対効果を考える場合、オペレータの判断によることが妥当と考えられる。

#### ② 作業単位の制御

作業単位の制御は、押し土、掘削、盛土に分類された作業単位に適合した制御を行うものであ

る。

### ③ ブレードの制御

ブレードの位置制御に対応するサーボ制御に該当するものである。

以上の中で制御のレベルとして上記のうち②、③が適用の期待される範囲である。

## 4) ブルドーザの制御とシミュレータ

### ① シミュレータの必要性

本研究は、非線型の制御対象であるブルドーザを未熟練者でも容易に掘削などの熟練のいる作業をできるようにすることを目的とするものである。具体的にはブルドーザのうまい操作方法を学習し、この操作方法をオペレータのブルドーザの制御支援システムとして利用できるようにする、つまり、ニューロンのモデル中の非線型処理部をブラックボックスとして取り扱い、熟練者の入力に対する出力の関係を明らかにすることにより、制御支援システムを構築しようとするものである。そして、ブルドーザの掘削等の個々の操作（入力）に対するブルドーザの挙動（出力）を結びつけるブラックボックスの特性が明らかになれば、これをブルドーザに付加することで、高度のブルドーザの運転操作が非熟練者でも可能になることが期待される。

ブルドーザの制御へニューラルネットを応用する上で操作の学習が重要な点となる。しかし、実機を用いたデータ収集は、次のような限界がある。

・ブルドーザの運転中の走行面（＝作業面）の形状、絶対高さはブルドーザの姿勢に影響を与えることにより、重要であるが、現実的にはこれをリアルタイムで計測することは難しい。

・単にブレードを上げるという操作だけでも次の a. ～ d. のように様々な要因がある。これらを識別可能にするには極めて多量のデータが必要となろう。しかし、実機を用いた試験では、ブルドーザの往復の都度、走行条件が変化することから再現性に乏しく、学習に対して有効なデータを得ることが難しい。

a. 掘削作業で負荷が増大して履帯がスリップを開始した

b. 押土作業で負荷が増大して履帯がスリップを開始した

掘削深さが目標値より深くなったため、これを調整する。

c. 掘削作業に際してブルドーザ自らが掘った穴に入って姿勢変化をしたため、急にブレードにかかる負荷が増大した。

d. 上りの段差があるため、これにブレードがかからないようにする。

これに対して、コンピュータが作る仮想空間内で仮想のブルドーザを走行させ、ブレード操作をすると、実際のブルドーザの運動学的特性と同様の特性を表示する「ブルドーザ・オペレーション・シミュレータ」は、実際の実験では不可能な同一作業条件での掘削操作を繰返し実行することが可能にする。また、通常の掘削では目標となる掘削深さに対するブレードの先端位置はオペレータの勘により操作されているのが現状であり、制御という視点から考えると多くの雑音の中に操作データが埋没している状態となるが、シミュレータを用いれば、掘削深さという目標値および、ブレード先端の位置がシミュレータのCRT画面上に明示的に与えることができ、ブレード操作レバーの操作に対するブレードの挙動の関係が非常にクリアな形でデータとして取得できる。

ブルドーザ・オペレーション・シミュレータを構成する要素は次のものである。

a. 操作レバーと作業機の関係

ブルドーザの作業機はブレードであり、これは操作レバーの操作により制御される。この関係は、ブレードの駆動系の寸法と油圧系統の仕様が明らかであれば、数理モデルを製作するのは容易である。

b. ブルドーザと土との関係

土は、ブルドーザが走行する時の走行面として、また、ブレードによる操作での作業対象という2面で考えられる。

走行面として考える場合、その走行面の形状、強度は、ブルドーザの姿勢に影響を与える。また、クローラに対する土表面の変形やスリップによってドライブスプロケットへ入力された回転速度に対する実際の車両速度も影響を受ける。

また、作業対象として考えた場合、掘削抵抗が生じ、クローラの駆動力とのバランスを考慮する必要がある。ブレードの全面に滞留した場合、走行抵抗として影響を受けることになる。また、ブルドーザの作業として説明したように掘削によって走行面の形状が変化し、走行中の姿勢変化に影響を与える。

これらの問題を解決する手段として最初は土の取扱を簡略化して解く必要がある。つまり、走行に対して土が変形せず、また、クローラ間のスリップも生じないこと、また、掘削に対しては掘削抵抗を生じさせず、実際はブレード前面に土砂が滞留して走行の抵抗になるが、これを無視し、ブレードの入った分、掘削できるとすることである。

このようにモデルを単純化させることで、ブルドーザと土との関係の数理モデルは飛躍的に簡素化される。そしてこれを基本モデルとして、実際の土の特性を考慮して土の特性を表したモデルを付加して行くことにより、実際にブルドーザと土の関係をより正確にシミュレーションできるモデルを構築できる。なお、ブルドーザの掘削、走行における相互作用は学際的な学問であるテラメカニックスで取り扱われるが、現在の段階では掘削、あるいは走行と単独で取り扱われ、掘削と走行を結びつけて体系的に取り扱うまでにはいたっていない現状ではある。

ブルドーザのブレード操作レバーの入力に対して、走行するブルドーザのブレードの制御特性を導き出すのに、基本的には前述の基本モデルで可能である。ブレードによる掘削と、そこに走行中に入り込む状態は運動学的に表現されることによる。それと実際の差は、凸部にクローラがかかった場合、局部荷重による土の変形を考慮しなければならないといったことが上げられる。しかし、この影響は操作レバーに対する走行するブルドーザシステム全体としてのブレードの制御特性をフィルターをかけて穏やかなものとする方向に作用することは明らかであろう。

そしてブルドーザ・オペレーション・シミュレータは、コンピュータ・プログラム上にブルドーザと地面のモデルを構成し、オペレータの操作を画面上のブルドーザの走行、ブレードの動きに反映させ、掘削などの作業を模擬して、この時のブルドーザの姿勢変化、地形の

形状変化などのデータを収集を可能とする。

また、本システムを使用することで、人工的に同一の作業環境を作り出すことができ、実際の実験のように作業面積などの制限を受けることなく、操作データを収集できる。また、走行面の変化など、通常の実験では収集が困難なデータを容易に収集することが期待される。

#### 5) 定性推論とブルドーザ・オペレーション・シミュレータ

AI研究の中の「定性推論 (qualitative reasoning)」とは、定性的な思考過程に焦点をあて、その分析、モデリング、ソフトウェア化、応用について研究する研究領域である。定性推論の研究では、「どのようなタイプの定性的思考があるか」、「それによってどのような問題が解決できるか」「定性的思考とほかのタイプの思考とどのように相互作用するか」、などの問題について、次のような側面から取組みが行われている。「認知科学的側面：人間の定性的思考がどのようなものか分析し、モデル化を行う」、「数学的側面：定性的思考の形式化を行う」、「計算論的側面：定性的思考を計算論的視点から形式化し、定性的な推論を行うための推論方式を設計する」、「応用的側面：定性推論の手法を解析、診断、制御、設計、教育などの知的問題解決に応用する」

定性推論の主な研究課題は土のとおりである。

##### ① 定性的なモデリング

現象を定性的に記述するための言語の研究。因果ネットワーク、定性微分方程式、定性プロセス理論などが研究されている。

##### ② 因果解析

因果関係が陽に表されていないモデルから因果関係を推定する。因果的順序づけ、ビューリスティックを用いた因果解析などが提案されている。

##### ③ 定性微分方式と定性シミュレーション

定性モデルの記号 (代数的) シミュレーションの研究。一般に定性的な推論だけでは挙動解析に曖昧性が発生するので、それを抑えるために様々な手法が提案されている。

##### ④ 比較解析

パラメータを含む動的システムの族の挙動が、パラメータを変化させたときどのように変化するかを推論する技術。DQ解析、誇張解析、積分方程式を用いる方法などが提案されている。

##### ⑤ 力学系の理論と挙動の自動解析

数値計算、数式処理、知識処理、計算幾何学などを統合して、微分方程式や差分方程式の定性的な挙動を解析する。

##### ⑥ 計算運動学

形状の相互作用を解析して可能な挙動を導出する。定性運動学、計算運動学などが提案されている。

##### ⑦ 自動モデリング

現象を説明するモデルを自動生成する研究。合成的モデリング、モデルグラフなどが提案されている。

##### ⑧ 応用

定性推論が記号推論を中心とした従来の人工知能技術と微分方程式などを中心とした定量的なモ

デリングを結びつけることを利用した応用システムの研究。

ブルドーザ・オペレーション・シミュレータを構成する重要な要素の一つに走行面に対するブルドーザの姿勢の決定がある。これは、地面の形状とクローラの形状から取り得る姿勢を計算するという問題の類似性から、⑥の計算運動学の範疇の研究課題と位置づけられる。

## 2. 3 ブルドーザ・オペレーション・シミュレータの製作

### 2.3.1 製作の目的

作業時のブルドーザの挙動について、再現性の高い条件で繰り返し収集が可能なシミュレータ開発に着目し、その基礎段階として、ブレード操作レバーを外部入力装置とした簡易なブルドーザ機械-地盤系の挙動計算によるシミュレータの開発を検討した。

### 2.3.2 シミュレータ開発に必要な技術・理論

本シミュレータを構成するために必要なものとして次のものがある。

#### 【テラメカニックス】

テラメカニックスに対する日本語の適切な訳はない状態であるが、オフロード、不整地面における機械性能と地表との力学的な相互関係の研究をいう。

走行車両は、走行面（地形）の変化に伴って姿勢変化する。また、ブルドーザのようにブレードを地表側に下げる場合、その貫入の反力により車輛の姿勢変化を生じる。

本シミュレータが最終的に現わすものは、「掘削深さや押し土量に対して、ブルドーザが走行可能か」ということである。ブルドーザが作業できるためには次式が成立する必要がある。なお、右辺と左辺の値が近接する際、履帯の滑りについても考慮する必要がある。

ブルドーザ  $>$  走行抵抗 + 掘削抵抗

ブルドーザが静止している状態では、その重量が履帯を通して地面に掛かることになる。ブルドーザはたわみ性履帯を使用していることから駆動輪・転輪の部分とその間では履帯の変位に差があり、地面に伝達される重量は異なる。計算上は簡易のため、剛性履帯として扱かうことが妥当であると考えられる。

ブレードによる掘削作業と走行の関係について、牽引力との関係及び、走行面の形状の関係でとらえる必要がある。前者は走行負荷とスリップの関係から密接に関係するが、後者は掘削がブルドーザの前方で行われることから、走行部の数学モデルは掘削作業と独立に考えられる。

なお、テラメカニックスにおける全ての研究を本システムに反映させることは、理想的であるが、この分野の研究は発展途上にあることから、ある程度簡略化したものとして扱かうことが实际的である。

### 2.3.3 シミュレータへの機械-地盤系特性の組み込み

前述したテラメカニックスの分野では、車体の支持力や駆動力を有効に発揮させることなどの面から地盤材料と履帯の形状等の機械と地盤の相互作用について力学的な検討がなされている。

ブルドーザの作業性能は、地盤材料の種類、地盤形状、また車輛の履帯幅、長さ、シューグラウサの高さ、ピッチ、形状等によって支配される。

履帯に伝達される張力は、車輛の駆動系統の回転速度-トルク性能によるものとなる。

この張力により、履帯が地盤面に対して平行移動することにより履帯下の地盤に車体荷重による応力分布の条件下でせん断抵抗が生じ車体の駆動力となる。

その他の抵抗力を含め、ブルドーザモデルに考慮すべき要因を整理した結果を図2-9に示す。

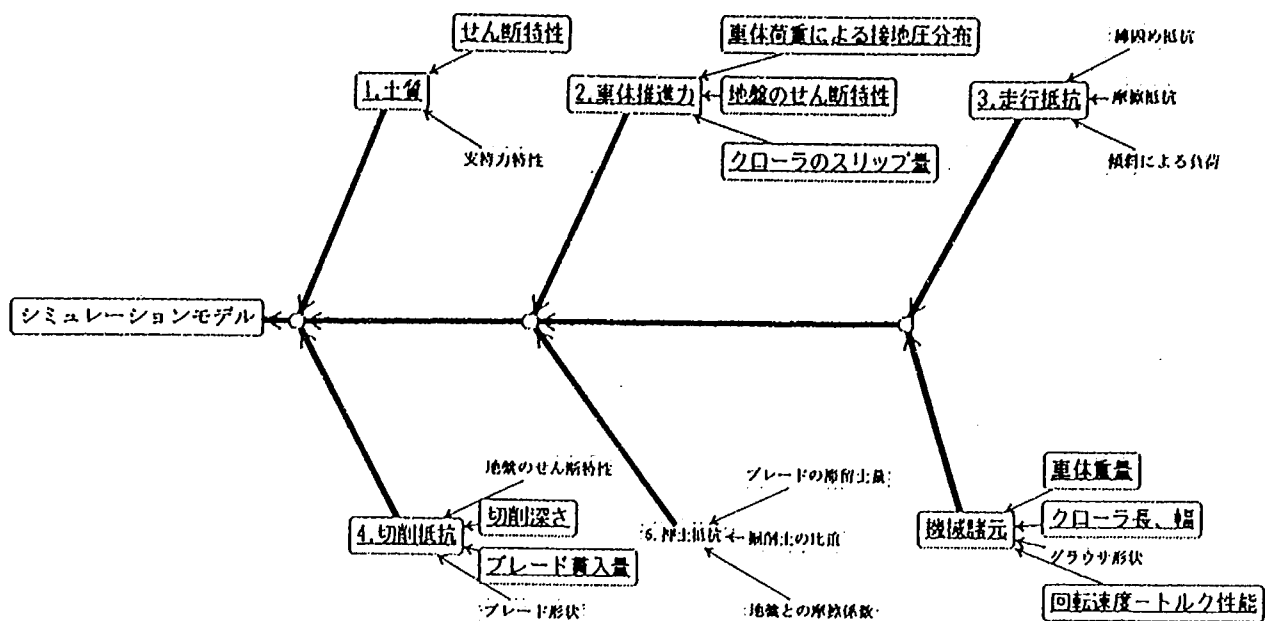


図2-9 ブルドーザの特性要因図

### 1) 土質

履带式建設機械の支持力及び牽引力等を算定するために必要な土質定数を求めるための試験法及びせん断抵抗の算定について、主なものを以下に示す。

#### ① せん断特性

個々の地盤の力学的特性は、どのような土が、どのような詰まり方をしている、どのような高速条件のもとで応力が加わるかによって影響を受ける。

一般に垂直応力 $p$ のもとでのせん断強さは、クーロンの破砕基準で、強度定数 $c$ 、 $\phi$ から次式で表される。

$$\tau_{\max} = c + p \tan \phi$$

強度定数 $c$ 及び $\phi$ はそれぞれ粘着力及び内部摩擦角であり、土質により異なるものとなる。統一土質分類による値を表2-1に示す。

これらは、地盤のせん断試験により求めることができる。

表2-1 統一土質分類

統一土質分類法による分類表 (盛土および基礎用)

| 主要区分      |                        | 分類記号 | 代 表 的 な 土                                     |
|-----------|------------------------|------|-----------------------------------------------|
| (1)       | (2)                    | (3)  | (4)                                           |
| 粗粒土       | 礫および礫質土                | GW   | 粒度分布の良い礫または礫・砂混合土、細粒分はわずかまたは欠如                |
|           |                        | GP   | 粒度分布の悪い礫または礫・砂混合土、細粒分はわずかまたは欠如                |
|           |                        | GM   | シルト質の礫、礫・砂・シルト混合土                             |
|           |                        | GC   | 粘土質の礫、礫・砂・粘土混合土                               |
|           | 砂および砂質土                | SW   | 粒度分布の良い砂または礫質の砂、細粒分はわずかまたは欠如                  |
|           |                        | SP   | 粒度分布の悪い砂または礫質の砂、細粒分はわずかまたは欠如                  |
|           |                        | SM   | シルト質の砂、砂・シルト混合土                               |
|           |                        | SC   | 粘土質の砂、砂・粘土混合土                                 |
| 細粒土       | シルトおよび粘土<br>$w_L < 50$ | ML   | 無機質のシルトおよび極微砂、岩粉、シルト質ないし粘土質の細砂または塑性の小さな粘土質シルト |
|           |                        | CL   | 塑性が普通以下の無機質粘土、微量粘土、砂質粘土、シルト質粘土、粘りけの少ない粘土      |
|           |                        | OL   | 塑性の小さな有機質のシルトおよびシルト質粘土                        |
|           | シルトおよび粘土<br>$w_L > 50$ | MH   | 無機質のシルト、雲母質またはケイソウ質の細砂質ないしシルト質土、弾性の大きいシルト     |
|           |                        | CH   | 塑性の大きい無機質粘土、粘りけの多い粘土                          |
|           |                        | OH   | 塑性が普通以上の有機質粘土、有機質シルト                          |
| きわめて有機質の土 |                        | Pt   | 泥炭およびその他のきわめて有機質の土                            |

| 分類記号  | せん断強さ                           |                                     |            | せん断強さの難易 | 施工管理の難易 | 開拓局コッパツツア突(ソール)例数 | 築造材料適合(ソール)例数 | 圧縮性(%)                    |                           |
|-------|---------------------------------|-------------------------------------|------------|----------|---------|-------------------|---------------|---------------------------|---------------------------|
|       | $c_u$<br>(kgf/cm <sup>2</sup> ) | $c_{sat}$<br>(kgf/cm <sup>2</sup> ) | $\phi$ (°) |          |         |                   |               | 1.4<br>kg/cm <sup>2</sup> | 3.5<br>kg/cm <sup>2</sup> |
| GW    | •                               | •                                   | >38        | 非常に高     | 非常に易    | —                 | 適(通)          | <1.4                      | •                         |
| GP    | •                               | •                                   | >36        | 高        | 非常に易    | —                 | 適(通)          | <0.8                      | •                         |
| GM    | •                               | •                                   | >34        | 高        | 非常に易    | 4                 | 適(不)          | <1.2                      | <3.0                      |
| GC    | •                               | •                                   | >31        | 高        | 非常に易    | 4                 | 適(不)          | <1.2                      | <2.4                      |
| SW    | 0.40±0.04                       | •                                   | 38±1       | 非常に高     | 非常に易    | —                 | 適(通)          | 1.4±•                     | •                         |
| SP    | 0.23±0.06                       | •                                   | 36±1       | 高        | 易〜中     | —                 | 適(通)          | 0.8±0.3                   | •                         |
| SM    | 0.52±0.06                       | 0.20±0.07                           | 34±1       | 高        | 易〜中     | 16                | 適(不)          | 1.2±0.1                   | 3.0±0.4                   |
| SM-SC | 0.51±0.22                       | 0.15±0.06                           | 33±3       | —        | —       | 3                 | —             | 1.4±0.3                   | 2.9±1.0                   |
| SC    | 0.76±0.15                       | 0.11±0.06                           | 31±3       | 高〜中      | 易〜中     | 7                 | 適(不)          | 1.2±0.2                   | 2.4±0.5                   |
| ML    | 0.68±0.10                       | 0.09±•                              | 32±2       | 中〜低      | 中〜非常に難  | 7                 | 適(不)          | 1.5±0.2                   | 2.6±0.3                   |
| ML-CL | 0.64±0.17                       | 0.22±•                              | 32±3       | —        | —       | —                 | —             | 1.0±0.2                   | 2.2±0.0                   |
| CL    | 0.88±0.10                       | 0.13±•                              | 28±2       | 中        | 易〜中     | 10                | 適(不)          | 1.4±0.2                   | 2.6±0.4                   |
| OL    | •                               | •                                   | •          | 低        | 中〜難     | —                 | 否             | •                         | •                         |
| MH    | 0.73±0.30                       | 0.20±0.01                           | 25±2       | 低        | 難〜非常に難  | —                 | 否             | 2.0±1.2                   | 3.8±0.8                   |
| CH    | 1.04±0.34                       | 0.11±0.06                           | 19±5       | 低〜中      | 非常に難    | 1                 | 適(不)          | 2.6±1.3                   | 3.9±1.5                   |
| OH    | •                               | •                                   | •          | —        | —       | —                 | 否             | •                         | •                         |
| Pt    | •                               | •                                   | •          | —        | 締固め不能   | —                 | 使用不能          | •                         | •                         |

(注) 1. 本表は、米国調査局、米国陸軍土木部、Earth and Earth-Rock Damsなどの資料から作成したものである。  
 2. 日本の土と完全に一致しないかも知れないが、予備設計の精度としては十分であろう。  
 3. • は資料不足を示す。  
 4.  $c_u$ : 最速含水比時の値。 $c_{sat}$ : 飽和時の値。透水係数の  $(10^{-1}-10^{-7})$  は  $(1 \times 10^{-1}-1 \times 10^{-7})$  の時である。

## ② 支持力特性

平板載荷試験では、地表面上に大きさ  $b \times \ell$  ( $b \leq 1$ ) の平板を置き鉛直荷重を  $b\ell$  で除した接地圧  $p$  平板の沈下量  $z$  との間に次式が成立する。

$$p = \left[ \frac{k_c}{b} + k_\phi \right] z^n$$

ここに、 $n$  は地盤の種類によって変化し沈下指数と呼ばれる。また、 $k_\phi$  及び  $k_c$  は土の粘着力  $c$ 、内部摩擦角  $\phi$  に支配される定数であり、平板載荷試験により求める必要がある。

実際の適用にあたっては、傾斜荷重に接する接地圧と沈下量の関係を求めるための実験を行う必要がある。また、衝撃力が作用する場合やパンチング作用の存在する場合は必ずしも上式を適用することは出来ない。

## 2) 車体推進力

履帯に伝達される張力は、車両の駆動系統の回転速度—トルク性能によるものとなる。

この張力により、履帯が地盤面に対して平行移動することにより履帯下の地盤に車体荷重による応力

分布の条件下でせん断抵抗が生じ車体の推進力となる。

以下に推進力の発生に影響する車体荷重による履帯下の地盤の応力分布と地盤のせん断特性、クローラと地盤のスリップについて述べる。

### ① 車体荷重による接地圧分布

ブルドーザが牽引力を発揮させるときの接地圧分布は、三角形状になることが実験などにより明らかにされている。これは、ブレード等による牽引力が地盤との接地面上のある高さに作用するため、車体の重心回りにモーメントが発生するためである。また、凹凸のある地盤の接地圧分布は、地盤の凸部で車体を指示することになるため、局所的に大きな接地圧が懸かることになる。このような局所的に大きな荷重が地盤に作用する場合、局所的に地盤が破壊され塑性状態になることが考えられる。

現状のテラメカニクス理論で算定可能なものとして、地盤に均等に接地した剛性履帯に重心位置から偏心して車体荷重が作用する場合の接地圧の算定について以下に述べる。車体重心に車体荷重と牽引力が作用している時の履帯接地部への合力の偏心量を $e$ とする。傾斜偏心荷重が作用しているときの地盤の接地圧は、履帯前後端の接地圧 $p_f$ ,  $p_r$ は、 $|e| < D/6$ の時、

$$p_f = \frac{W}{BD} \left[ 1 - \frac{6e}{D} \right]$$

$$p_r = \frac{W}{BD} \left[ 1 + \frac{6e}{D} \right]$$

$W$  : 車体荷重

$D, B$  : 履帯の長さ, 幅

履帯長にわたっての接地圧分布は台形となり、履帯下の地盤支持力 $Q$ は、

$$Q = BD_e \left[ \left[ 1 + 0.3 \frac{B}{D_e} \right] I_c N_c + \left[ 0.5 - 0.1 \frac{B}{D_e} \right] B I_r \gamma N_r + q I_q N_q \right]$$

ここに、 $D_e$ は有効履帯量であり偏心荷重に対する補正として履帯接地部分の有効載荷面積を考慮したものである。また、 $c$ 粘着力、 $q$ 上載荷重、 $N_c$ ,  $N_q$ ,  $N_r$ は支持力係数、 $I_c$ ,  $I_q$ ,  $I_r$ は傾斜荷重による補正係数である。

$$I_c = I_q = \left[ 1 - \frac{\alpha}{90} \right]^2$$

$$I_r = \left[ 1 - \frac{\alpha}{\phi} \right]^2, \quad \alpha = \tan^{-1} \frac{T}{W}$$

$\phi$ は地盤の内部摩擦角、 $\alpha$ は荷重 $W$ の傾斜角、 $T$ は牽引力である。

### ② 地盤のせん断特性

履帯の推進力 $T$ は、履帯下面に作用する土のせん断抵抗 $T_b$ と、履帯のグラウサ間に挟まれた土の両側面に作用する土のせん断抵抗 $T_e$ の和である。

$$T = T_b + T_s$$

Tbは次式で算定される。

$$T_b = cDB + W \tan \phi$$

また、ブラウザの両側面に作用するせん断抵抗力Tsは

$$T_s = 2D \left[ ch + \left[ \frac{Wh}{\pi BD} \cot^{-1} \left[ \frac{h}{B} \right] \right] \tan \phi \right]$$

ここに、c：粘着力、 $\phi$ ：内部摩擦角、W：車体重量、

B：履帯幅、D：履帯長、h：ブラウザ高さ

### ③ 履帯と地盤のスリップ

車両が一定速度で走行しているとき、ブラウザは車体に対して平行に移動していくため履帯下面に発生するスリップは車体後方に行く程大きくなる。

接地開始地点から距離x離れた位置でのスリップ量jは、距離x移動するまでの所要時間tまでのスリップ速度 $R\omega - v$ を積分して求めることができる。R、 $\omega$ は sprocket 半径と回転角速度である。

$$J = \int_0^t (R\omega - v) dt$$

$$= \frac{R\omega - v}{R\omega} x$$

土のせん断強さは  $\tau_{max} = c + p \tan \phi$  で表されるので、任意のスリップ量jに対して発揮されるせん断抵抗力  $\tau$  はスリップ量jの関数  $f(j)$  より次式で表される。

$$\tau = \tau_{max} f(j)$$

$f(j)$  は粘性土や砂質土に関する関係式が提案されている。例として(1)Bekkerの式、(2)Janosi-Hanamotoの式を示す。

$$(1) f(j) = \frac{\exp \{ (-K_1 + \sqrt{K_1^2 - 1}) K_1 j \} - \exp \{ (-K_1 - \sqrt{K_1^2 - 1}) K_1 j \}}{\exp \{ (-K_1 + \sqrt{K_1^2 - 1}) K_1 j_m \} - \exp \{ (-K_1 - \sqrt{K_1^2 - 1}) K_1 j_m \}} \quad (2) f(j) = 1 - \exp(-j/K_2)$$

$K_2$  は土の変形係数

$$K_1 = \frac{1}{j_m} \frac{\log(-K_1 - \sqrt{K_1^2 - 1})}{\sqrt{K_1^2 - 1}}$$

$K_1, K_2$  は土の変形係数

履帯の発揮する推進力は任意の地点xに作用するせん断抵抗力  $\tau_x$  の履帯長にわたっての総和となる。

### ④ シミュレータへの組み込みの検討

凹凸のある地盤を走行する履带式車両の履帯下の接地圧分布及びその沈下量を算出することは、現状のテラメカの理論からでは困難である。即ち前述した地盤支持力と沈下量の関係を、凹凸のある地盤を走行する履带式車両に適用することは現実的でない。

そこで、

・今回開発を検討したシミュレータでは、走行地盤を剛体鋼体として取扱い、地盤は車体が走

行することに対して沈下や変形はしないこととして取り扱った。

・走行地盤のせん断抵抗の算出にあたっては、以下に示す方法により地盤の接地長を算出し、その接地長をせん断抵抗の算出における有効履帯長とした。また、接地圧はその有効履帯長の範囲で均等に分布することとして簡略化した。

#### a. クローラの接地長の計算

凹凸のある地盤をブルドーザが走行する場合、クローラの一部が地面に接しながら移動することになる。しかし、今回は走行地盤は剛体、すなわち車体の走行による地盤の沈下や変形を考慮しないこととしている。そのため、履帯と地盤の接地部は、重心に懸かる車体荷重が履帯下面に作用する地点を中心としてその前後の1点または2点のみで車体が支持されることとなる。そこで、クローラの接地長（クローラが地盤に接している長さ）を算出するため、グラウザの先端を結んだ線上にせん断によるすべり面が生じることとして、そのすべり面に接する地盤の長さを接地長として利用した。その任意の接地長での地盤のせん断抵抗力をクーロンの破碎基準式により求めることとした。以下に接地長の計算手順を示す。

クローラ後端oからの距離xの地点の履帯の高さyxは、

$$y_x = y_o + x \tan \theta$$

ただし、

$y_o$  : クローラ後端の高さ

$\theta$  : 車体の傾斜角

で表される。

地形データの感覚  $\Delta x$ 、クローラ長さ  $l_c$  とすると、

$i = 0, 1, \dots, n$ ,  $n$ ,  $l_c / \Delta x$  の地点の履帯の高さ  $y_i$  は、

$$y_i = y_o + i \cdot \Delta x \tan \theta$$

であり、地点  $i$  での地盤とクローラ面との距離  $s_i$  は、

$$s_i = h_i - y_i = h_i - (y_o + i \cdot \Delta x \tan \theta)$$

上式について、グラウサの先端を結ぶ線上にせん断滑りが発生するとして、グラウサ高さ  $l_g$  を考慮して、

$$s_i = h_i + l_g - (y_o + i \cdot \Delta x \tan \theta)$$

の上式の  $s_i$  を用いて、地盤とクローラの接地長  $d_s$  を次式により算出する。

$$d_s = \sum_{i=0}^n \delta_i \cdot \Delta x$$

ただし、

$$\delta_i = \begin{cases} 1 & ; s_i \geq 0 \text{ (クローラが地盤に接している場合)} \\ 0 & ; s_i < 0 \text{ (クローラが地盤に接していない場合)} \end{cases}$$

#### b. 任意の接地長で発揮可能な車体推進力の算出

凹凸のある地盤を走行するブルドーザが発揮可能な推進力  $T$  を次式により算出する。

$$T = \eta / (2 \cdot c \cdot B \cdot ds + W \cdot \tan \phi)$$

ここで、Wは車体荷重、Bは履帯幅、c及びφは地盤の粘着力及び内部摩擦角

下図は、上式において土の粘着力c及び内部摩擦角φをそれぞれ変えた場合の接地長dsと推進力Tの関係である。

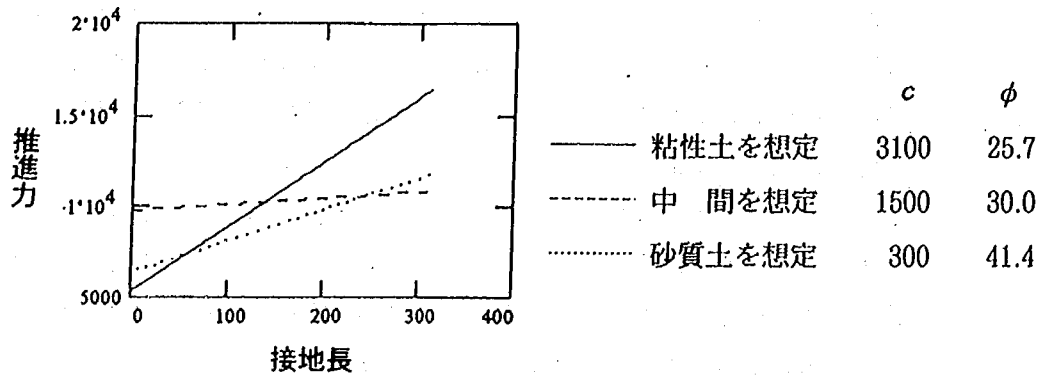


図 2-10 接地長と推進力の関係

#### c. スリップ率の算出

クローラと地盤面との滑りは、ブルドーザの走行・掘削に対する抵抗力が、地盤の發揮できる最大の牽引力と同じか超えた場合に滑り率が“1”，すなわちクローラが地盤に対して完全に滑った状態になることとして、(走行抵抗+掘削抵抗)／推進力(以下・牽引比Hという)に対して、次式により滑り率*i*を*i*=H<sup>k</sup>として算出することとした。指数係数kは地盤の土質などにより任意に設定するものとした。

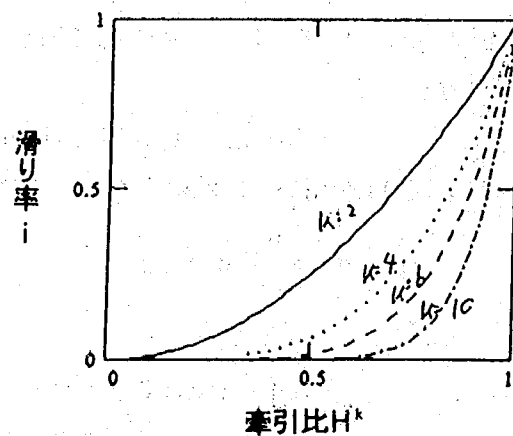


図 2-11 牽引比と滑り率の関係

### 3) 走行抵抗力

車体が移動することに対して地盤の抵抗力が作用する。

走行抵抗としては、締め固め抵抗、摩擦抵抗、車体が傾斜することによる走行負荷等が考えられている。

車体荷重Wによる静的沈下量は地盤の支持力に対応して発生する。このような地盤の変形によって履帯には推進力が発生するが、それと同時に走行に伴って地表面を締め固め、わだちを形成するのに必要な走行抵抗も発生する。この抵抗を締め固め抵抗という。

また、摩擦抵抗は履帯の足回り部分に付着した粘性土などが走行時に自然地盤と接触しおこる摩擦によるものである。

#### ① 締め固め抵抗

履帯幅B、履帯長Dの剛性履帯が距離Lだけ進行するときの仕事量は、 $B \times L$ の面積の載荷板が深さzまで沈下し地盤を締め固めるのに必要な仕事量と等しいとして、またこの仕事量は履帯が走行抵抗 $R_c$ に打勝って距離Lだけ進行するときの仕事量と等価であると仮定した場合次式となる。

$$BL \int_0^z p dz = L \cdot R_c$$

ここに、

$R_c$  : 締め固め抵抗,  $p$  : 接地圧,  $z$  : 沈下量,  $B$  : 履帯幅,

$D$  : 履帯長,  $L$  : 移動距離

載荷板の接地圧は平板載荷試験から与えられるので、締め固め抵抗 $R_c$ は、

$$\begin{aligned} LR_c &= B \int_0^z \left[ \frac{k_c}{B} + k_\phi \right] z^n dz \\ &= \frac{1}{(n+1)(k_c + Bk_\phi)^{1/n}} \frac{1}{D} \end{aligned}$$

ここに、

$n$  : 沈下指数,  $k_c$ ,  $k_\phi$  : 粘着力及び内部摩擦角に支配される定数

$W$  : 車体重量

#### ② 摩擦抵抗

剛性履帯の足回り部分に付着した粘性土が履帯走行中に自然地盤と接触して摩擦するときの摩擦抵抗 $R_f$ は、接触面積を履帯長Dと沈下量zの積であると仮定して、次式で表される。

$$R_f = 2Dz(c + K_o \gamma z \tan \phi)$$

ただし、 $\gamma$  : 土の単位体積重量,  $K_o$  : 静止土圧係数

#### ③ 傾斜による走行負荷

車体の傾斜角 $\theta$ 、車体荷重Wとして、傾斜による車体荷重の進行方向に対する走行負荷 $R_w$ は次式で表される。

$$R_w = W \sin \theta$$

#### ④ シミュレータへの組み込みの検討

本シミュレータの検討では、接地圧分布と沈下量を算定しないので、地盤の締め固め量や履帯と自然地盤との接触面積を算出できない。そのため、クローラで地盤を締め固め轍を形成しながら

走行する際に発生する締固め抵抗やその他をまとめて、走行抵抗力＝一定値として、本シミュレータに考慮することとした。

また、車体の傾斜角 $\theta$ 、車体荷重 $W$ として、傾斜による車体荷重の進行方向に対する抵抗力 $R_w$ を次式で算定することとした。

$$R_w = W \sin \theta$$

#### 4) 掘削抵抗力

##### ① 切削抵抗

地盤の掘削機構についてはいまだよく分かっていない。

地盤の掘削による抵抗としては、刃先が土を削ってゆくときの抵抗を切削抵抗、運土抵抗等を含んだものを掘削抵抗という。

切削抵抗を求める方法として、土圧論を用いる方法がある。土圧論による方法では、ブレードに懸かる土圧は深さ方向とともに直線的に増加することとしているが、実際には刃先付近に応力集中が起こるため、抵抗合力の着点点が異なったものとなる。

次に実験による方法は、土固有の特性値と抵抗との関係を実験によって明らかにし実験式の形で適用するものである。幅5cmの鋼板を深さ $t$ だけ土中に押し込み、地上に設けた固定支点 $o$ のまわりを掘り返すときの最大モーメント $P_1$ を用いて次式に示す $R_s$ が提案されている。

$$R_s = \frac{P_1}{5t^3}$$

$R_s$ は、

地盤を掘り返すときに土中部分の鋼板に作用する平均応力の深さ $t$ に関する比例定数である。 $R_s$ は土の比重、締固め度、粒度分布、及び粒径に関係すると思われるが、 $R_s$ の最大の特徴は切削抵抗に正比例することである。

##### ② シミュレータへの組み込みの検討

ブルドーザの掘削作業は、ブレードで地場の連続的に掘削するものである。土固有の特性値と抵抗との関係、また掘削深さなどの掘削条件との関係を実験式の形で適用することが考えられる。

本シミュレータでの切削抵抗の算定方法を以下に示す。

切削抵抗力は、ブレードが前進することで、ブレード前方の地盤に発生する円弧滑り面にせん断応力が生じることとして考える。ただし、計算においては、円弧滑り面を直線として簡単化した。

図2-12に示す通り、ブレード前方の地盤には、走行方向に対して $\alpha$ の角度で滑り面が発生することとして、車体の傾斜角 $\theta$ と地盤の形状により、滑り面の長さ $l_s$ が変化することとした。計算フローを図2-13に示す。なお、フロー図において $y_o$ はブレード下端の高さ、 $\Delta x$ は地形データの計算間隔、 $h_i$ はブレード下端から前方 $i$ 番目にあたる地形データを表す。

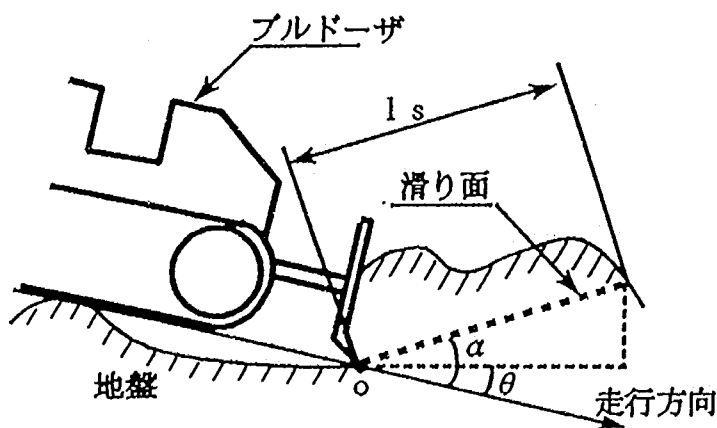


図 2-12 セン断による滑り面の考え方

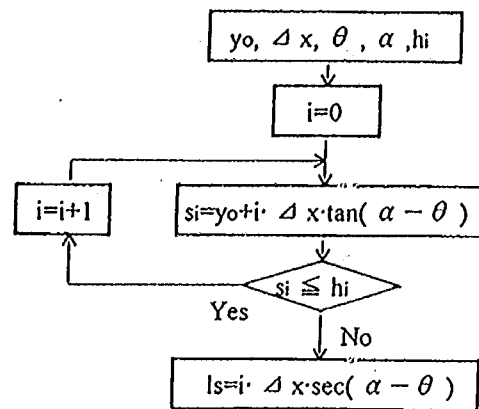


図 2-13 滑り面の長さの計算フロー

滑り面の面積は、上記で算出した長さ $ls$ とブレード幅 $db$ の積で表すとして、切削抵抗 $T_c$ を、次式により算出する。また、 $l$ は各種地盤のせん断特性等から任意に設定することとした。

$$T_c = \tau \cdot ls \cdot db$$

### 5) 押土抵抗

押土抵抗の考え方として、摩擦係数と押土重量の積で表現する方法が考えられる。摩擦係数そのものは土の性状の関数となり、土質、含水比、締め固め状態等の各種の要素で構成されるものと思われる。また押土重量は、ブレード前面に溜まった掘り返した後の土量、とその単位体積重量を算定することが必要であるが、それぞれ土のブレードからのこぼれの問題や土の掘り返しによる密度の変化等があり、困難と考えられる。

したがって、今回のシミュレータでは考慮しないこととした。

## 2.3.4 システム構成

### 1) ハードウェア構成

図 2-14に本シミュレータのハードウェアの構成を示す。

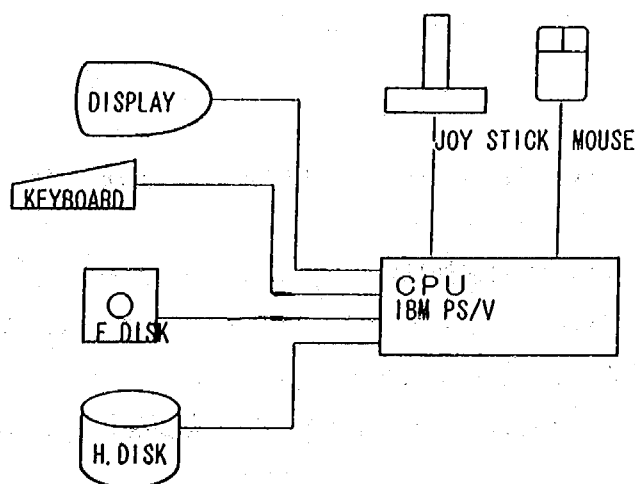


図 2-14 ハードウェア構成

a. パーソナルコンピュータ

各入出力装置の管理とプログラムの実行を行うもので、本シミュレータTでは、IBMPS/Vを用いた。

表2-2にシミュレータに用いるパーソナルコンピュータの仕様を示す。

表2-2 パソコンの仕様

|            |                       |
|------------|-----------------------|
| メーカ名       | IBM                   |
| 型式         | PS/V Master           |
| CPU        | Pentium 90MHz         |
| RAMメモリ     | 8MB (最大40MB)          |
| ハードディスク    | 720MB                 |
| ビデオボード     | PCIバス Vision 864      |
| 拡張インターフェイス | ISA, PCI              |
| その他装備品     | Sund Blascr16, CD-ROM |

b. ディスプレイ

ブルドーザ及び地形を表示し、オペレータに対する視覚的インターフェースとなる。

c. キーボード

シミュレーションプログラムの立ち上げ及びシミュレーション条件の入力装置として使用する。

d. ハードディスク

シミュレータプログラム、及びオペレータの操作データ、地盤形状のデータ等を記録する。

e. 各種印刷に使用する。

f. ジョイスティック

本シミュレータにおける、ブルドーザの操作におけるブレードリフトレバーに相当するものとする。

2) ソフトウェア

シミュレータプログラムは、C++で記述し、MS-DOS/V Ver6.0上で動作するものとした。

3) 入力装置

ブルドーザの操作具としては、燃料調整レバー、走行レバー、変速レバー、メインクラッチペダル、ブレーキペダル、ブレード操作レバーがある。

この中で、掘削などの作業中の主な操作具はブレード操作レバーとなる。

このブレード操作レバーは保持状態から上げまたは下げに移行するのに対して、不感帯を設けることにより操作がシビアになるのを防ぎ操作の誤りを防いでいる。

本シミュレータでは、ブレード操作レバーを唯一の操作入力装置として、不感帯等の特性を考慮することとした。

#### 4) ブルドーザ

##### ① 機種及び型式

ブルドーザの重量による分類は機種型式の増加に伴って必ずしも明確ではないが、一般には10 t以下のものを小型、15t前後のものを中型、20t以上のものを大型とよび、さらに3 t以下のものを超小型、60 t以上のものを超大型とよんでいる。我が国のブルドーザは、現在3 tのものから90 t程度のもので制作されている。

本シミュレータが対象とするブルドーザの寸法を図2-15に示す。

##### ② ブレードシリンダ系統の性能

排土板はCフレームにつけられ、Cフレームはトラックフレームに取り付けられている。排土板の上げ下げは油圧式による油圧シリンダで行う。排土板は角度が変化できる構造になっている。油圧シリンダは油タンクの中よりギアポンプによって送油される。それを運転席のレバーによって、操作弁のスプールを動かし送油の通路の切り替えを行う。

操作弁を上下位置にすると土工板は油圧シリンダの工程の長さだけ上下に動き、中立位置にすると土工板は静止する。

油圧シリンダの推力はシリンダの有効面積に比例し、ピストン速度は有効面積に反比例する。

ブレードの操作にあたっては、微少な操作レバーの操作でブレードが動く場合、操作がシビアになる。また、走行中の振動等より誤操作の危険性が高くなる。そこで、ブレード操作レバーの中立位置の近傍はスプールにより、「不感帯」を設け、この範囲で操作レバーを動かしてもブレードが動かないようにしている。また、不感帯からブレード上げに切り替える領域では、ブレードの微少な操作が可能ないようにブレードレバーの変位に対するブレードシリンダの伸縮速度を低いものにしている。

本シミュレータでは、これらのブレード操作レバーの特性を考慮し、図2-16に示すブレードレバーシリンダ伸縮速度線図を用いた。

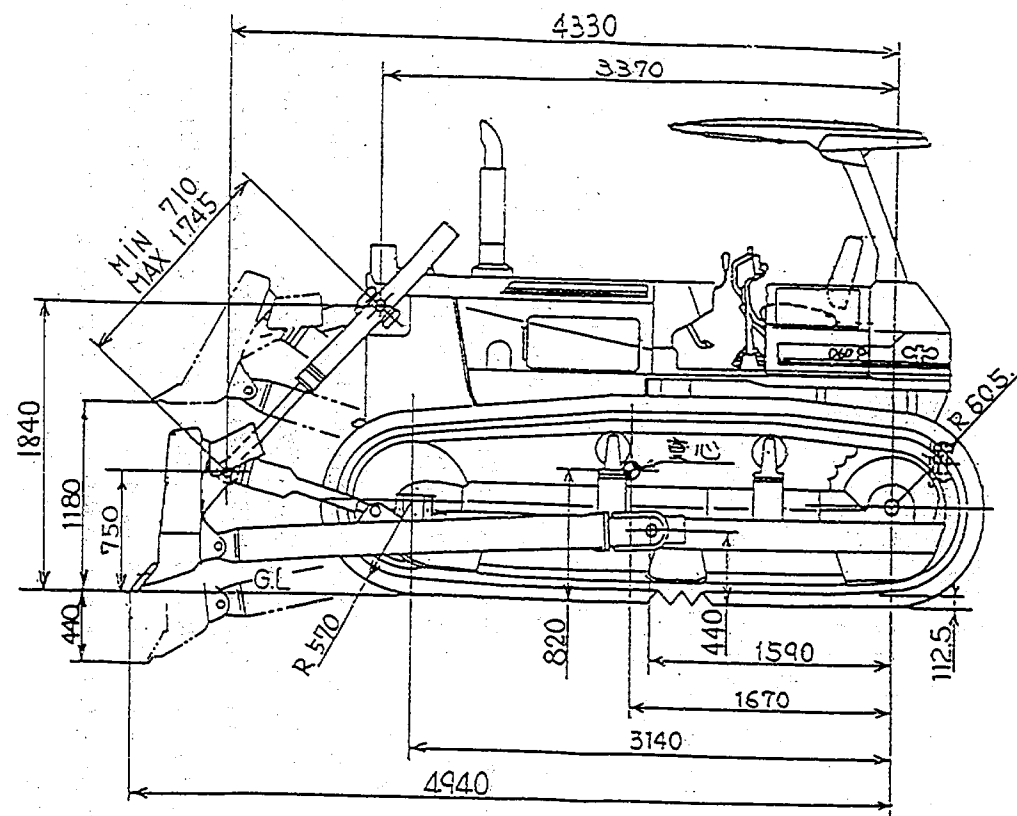


図2-15 ブルドーザの寸法

# ブレードレバー—シリンダ伸縮速度線図（無負荷、エンジン回転フル）

レバー変位～シリンダ変位 応答時間  
0.15 sec

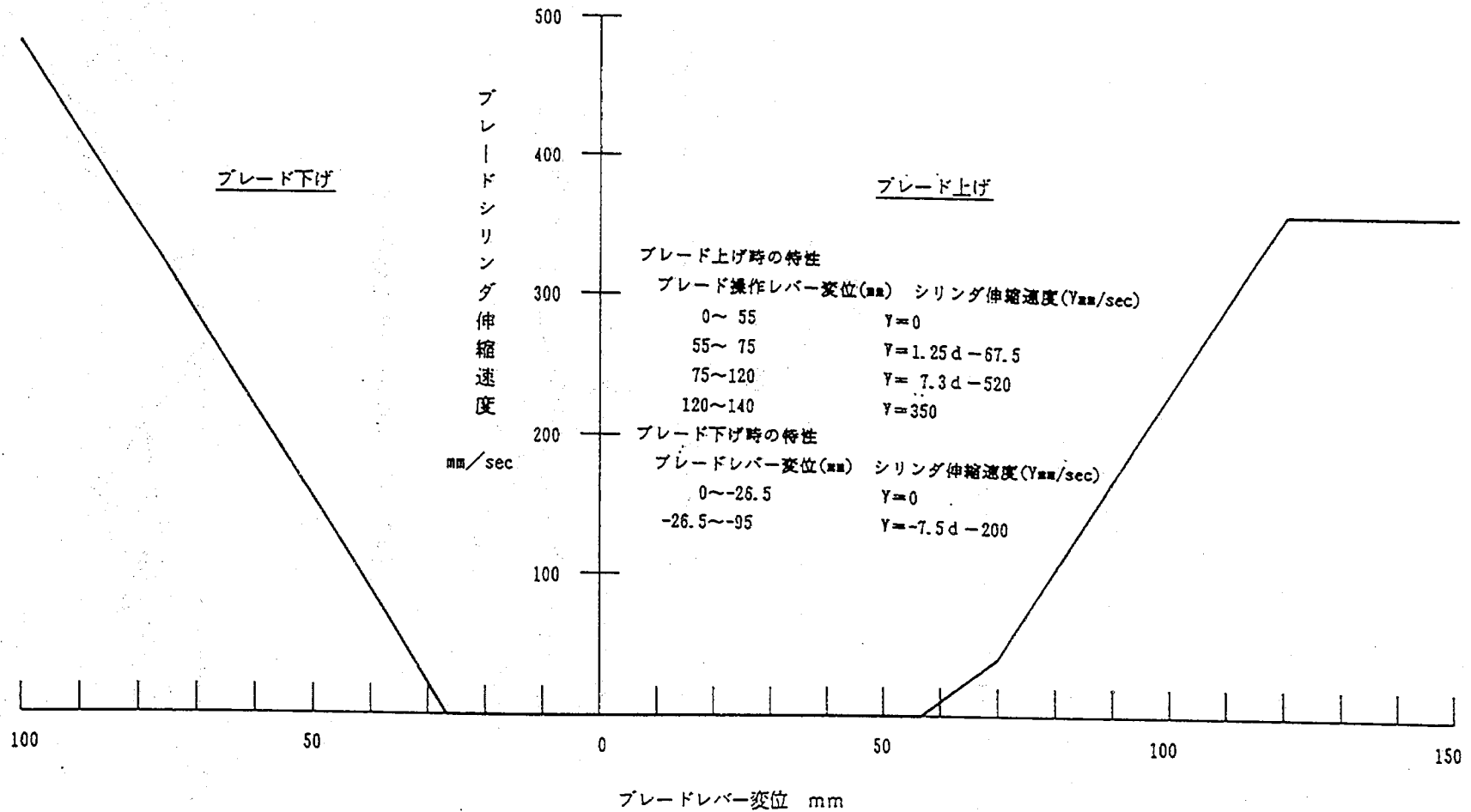


図2-16 ブレードレバーシリンダ伸縮速度線図

### 2.3.5 シミュレーションプログラムの基本機能

#### 1) 車体の姿勢を求める処理

車体の姿勢は「車体の重心がクローラと地面の接地点で構成される三角形を含む多角形の内側に存在することで安定となる。そして、車体の姿勢はクローラと接地点により決定されるということができる。

ブルドーザを2次元モデルとして扱い、また履帯を剛性履帯とした場合、その姿勢角は重心を挟んで前後の地盤の凸部と履帯の接触する2点により決定される。

この接触する部分を求めるのは、一般には非線形問題の解析の手法が必要となる。ここでは、ソフトウェア的手法によりブルドーザの姿勢角を求める手法を示す。

基本的な考え方は、地形データ列における隣り合う点と更に隣り合う点の位置を比較して中央にある点が低い位置にある場合、その中央にある点を処理対象からはずす。これを履帯の先端から後端まで行う。そして一連の処理を終えた地形を代表する線の傾斜がブルドーザの傾斜となる。この処理は、ブルドーザの姿勢を求めるために使用するものであり、ブルドーザの進行の都度同様の処理を繰り返すこととなる。

図2-17～図2-19に処理の概略を図示する。

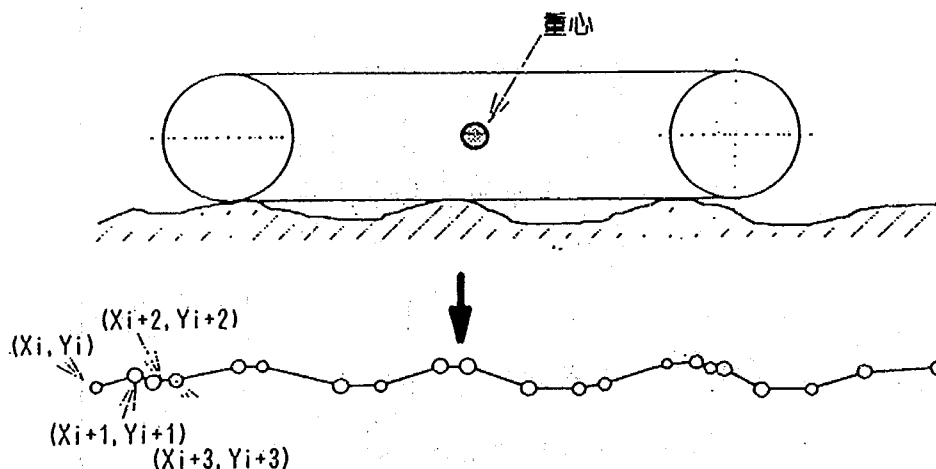


図2-17 ブルドーザの姿勢と地形

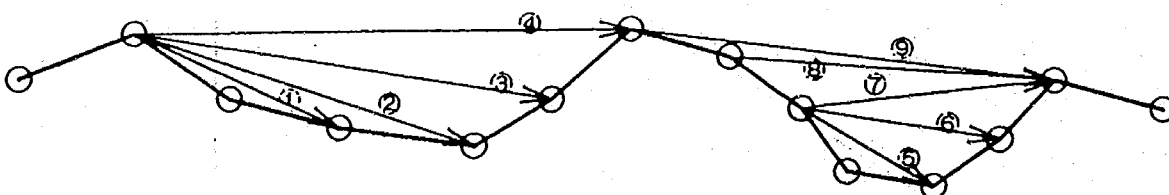


図2-18 ブルドーザの姿勢を求めるための地形処理

$$\frac{y_{i+1}-y_i}{x_{i+1}-x_i} \cdot \frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}}$$

$$\frac{y_{i+1}-y_i}{x_{i+1}-x_i} \geq \frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}} \Rightarrow go\_ (A)$$

$$\frac{y_{i+1}-y_i}{x_{i+1}-x_i} < \frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}} \Rightarrow neglect\_ x_{i+1}, y_{i+1} \Rightarrow$$

$$\frac{y_{i+2}-y_i}{x_{i+2}-x_i} \cdot \frac{y_{i+3}-y_{i+2}}{x_{i+3}-x_{i+2}}$$

$$\frac{y_{i+2}-y_i}{x_{i+2}-x_i} \geq \frac{y_{i+3}-y_{i+2}}{x_{i+3}-x_{i+2}} \Rightarrow go\_ (B)$$

$$\frac{y_{i+2}-y_i}{x_{i+2}-x_i} < \frac{y_{i+3}-y_{i+2}}{x_{i+3}-x_{i+2}} \Rightarrow neglect\_ x_{i+2}, y_{i+2}$$

(A)

$$\frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}} \cdot \frac{y_{i+3}-y_{i+2}}{x_{i+3}-x_{i+2}}$$

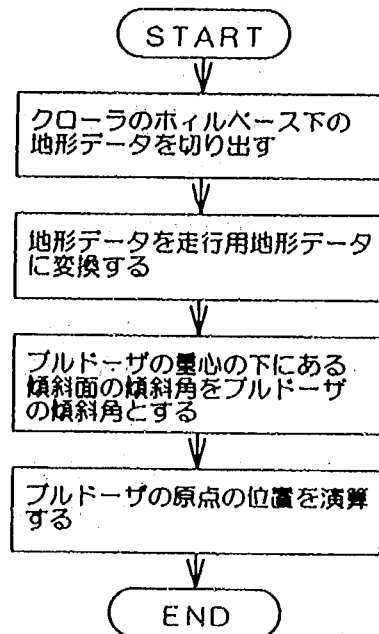


図 2-19 ブルドーザの姿勢角を求める処理フロー

ブルドーザが凸部の頂点を越える際、車体の前部は浮いた状態となり、車体の重心が凸部の頂点を越えると車体前部が前方の地盤に接地するので、後方の接地点を中心とした回転をすることとなる。この回転速度は、水平方向の速度ベクトル $v$ と落下速度ベクトル $g \cdot t$ のうち、履帯の接地点 $A$ を中心として重心 $W$ までを半径とし、これに直角の方向のベクトルの合成によって現わされる。この半径は車体の前進によって増加することとなり、また、車体自体の姿勢が変化することからベクトルの後方も変化する。本シュミレータにおいても同様の考え方でプログラムに組み込むこととした。

## 2) ブレードの地盤への貫入深さ計算

ブレードの先端の位置は以下により決定される。

- ・ブレード操作レバーの操作
- ・地形とブルドーザの履帯の相互関係により決定されるブルドーザ車体の姿勢

すなわち、前項で求めた車体の姿勢とオペレータのブレードレバー操作によるブレードの上下移動により、地盤に対するブレードの相対位置が計算できる。

そこで、本シュミレータでは、現在の地表面の高さとブレード先端の高さを比較して、後者が低い位置にあれば地盤を掘削していることとし、掘削によって逐次地盤を構成する地形データを後進することとしている。このブレードによって車体前方で掘削された地盤は、ブルドーザが前進していくとブルドーザ本体を支持することとなるので、前項に示した車体姿勢を求める処理に利用されることになる。

## 3) 画面表示処理

ブルドーザの位置・姿勢データとブレードの角度データ（あるいは油圧シリンダのストローク）を与えることにより、ブルドーザの走行の姿をCRT画面上に表示する。

CRT画面という限られた表示範囲でブルドーザの姿勢変化を示すため、ブルドーザの表示上の水平方向の位置は固定として、ブルドーザの位置姿勢データの内の水平方向のデータは走行する地盤の水平方向の表示範囲を逐次後進していくために用いる。

ブルドーザは仮想原点（スプロケット中心）の高さの変化とそこを中心とした傾斜角度分の回転座標変換により表示を行う。

表示する地盤は、シミュレーション中の核時刻の地盤の形状データを上記により与えられた表示範囲でCRTに表示することとする。

この地形データは、水平方向に一定ピッチ（1 cm）で離散化したものを用い、地盤の始点からの  $i$  cm にあたる地盤の高さを  $y_i$  として、パソコンのメモリ内に格納する。

#### ① ブルドーザ本体の表示

与えられた車体原点の絶対位置、及び車体の傾きをもとに、ブルドーザの輪郭をCRTに表示する。

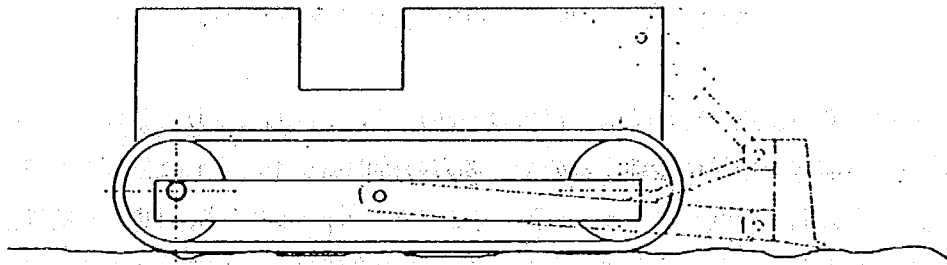


図 2-20 ブルドーザの車体の表示

#### ② ブレードの表示

車体原点（スプロケット中心の絶対位置）及び車体傾斜とブレードシリンダの伸縮長より計算したブレードの位置から、画面に表示するものであり、ブレードのアームやシリンダ等を簡略化

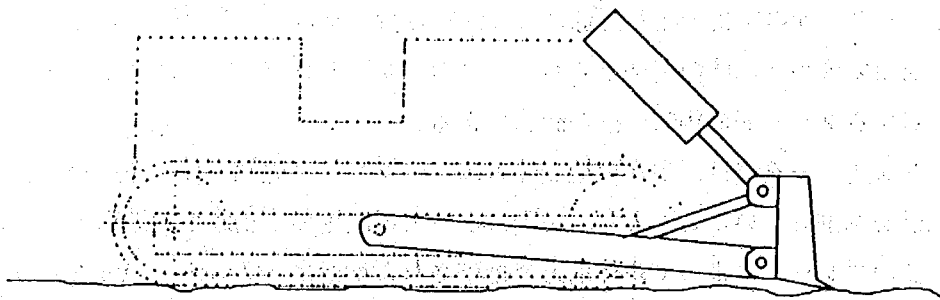


図 2-21 グレードの表示

### ③ 地形の表示

掘削等により逐次更新される地形データを表示させる。

### ④ 掘削目標値の表示

オペレータに対して、シミュレーション実行中に掘削量の目標として提示するためのもので、シミュレータの設定画面により任意の掘削目標値が表示できることとしている。

## 4) 操作データファイルの作製

シミュレーション実行中の各種操作データは、パソコンのメモリに記憶される。

シミュレーション終了時にこれらのデータは、ハードディスクに任意のファイル名で記録される。

記録されるデータ項目を以下に示す。

### ① 操作データファイル

- スプロケット中心の絶対位置（水平方向、鉛直方向）
- 車体傾斜角
- 走行速度（水平方向）
- ブレード操作レバー操作量
- ブレードリフトシリン伸縮長
- ブレード下端の絶対位置

### ② 地形データファイル

シミュレーション終了後の地形

## 5) シミュレーション実行画面

写真 2-1 及び写真 2-2 にシミュレータの外観及び実行中の画面を示す。



写真 2-1 シミュレータ外観

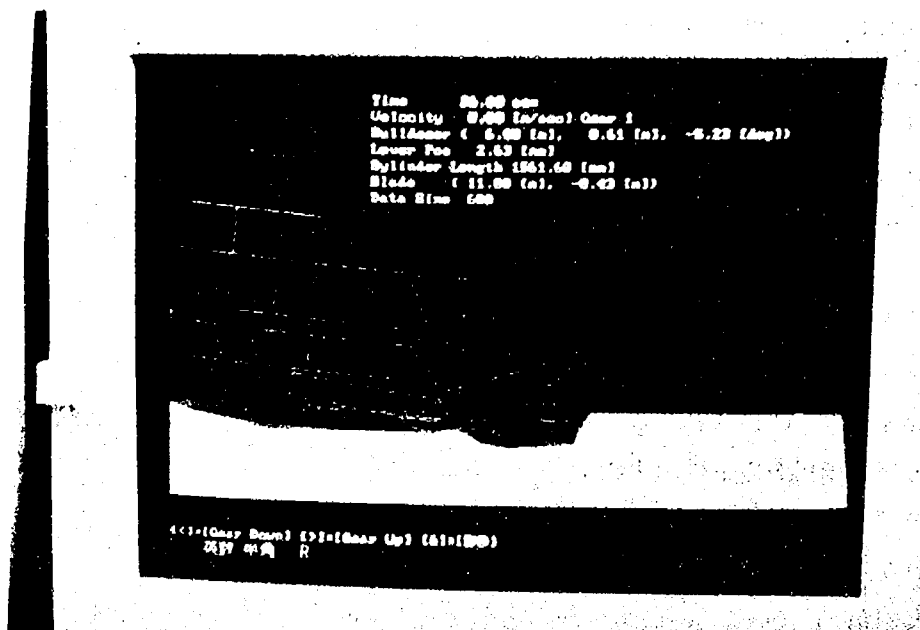


写真 2 - 2 実行中の画面

### 2.3.6 シミュレータの操作手順

#### 1) 起動

DOSプロンプトから実行ファイル名を入力することにより、シミュレーションプログラムを立ち上げることとする。ファイルのディレクトリ名及びファイル名は以下の通りである。

・ディレクトリ名

E: ¥MEITECY¥BULL[ENTER]

・ファイル名

MAIN. EXE

図 2 - 22に起動画面を示す。

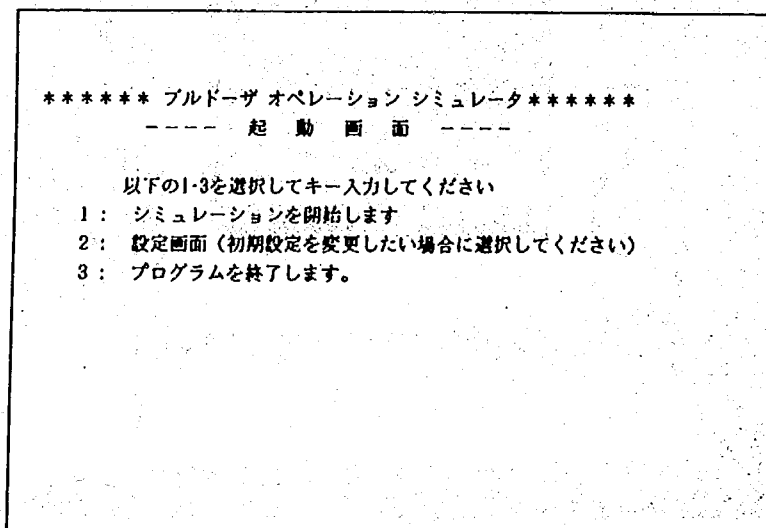


図 2 - 22 起動画面

## 2) シミュレーション条件の設定

本プログラムはシミュレーションプログラムから呼び出され、シミュレーション対象となるブルドーザに関わる入出力データファイルの新規作成や地盤等の掘削条件を設定するものである。図2-23に条件設定画面を示す。

入出力ファイルはそれぞれユニークな名称をつけることで、新規ファイルを作製することが可能とする。

データは、CSVフォーマット（各データをコンマとダブルクォーテーションで区切ったものでPC-DOSのテキストファイルとして処理可能）とする。これにより、表計算ソフトあるいは汎用のエディタでのデータの加工等を可能とする。

```
***** ブルドーザ オペレーション シミュレータ *****
----- 切 削 設 定 画 面 -----
設定値を変更する場合、その項目に該当するキーを押して
数値を入力してください（ESCキーで起動画面に戻ります）
*****

R : 掘削目標値          ( -0.100[m] )
U : 牽引係数            ( 0.900 )
T : 地盤のせん断強度    ( 10000.000[kg/m2] )
S : 地盤の滑り角        ( 45.000[度] )
K : 牽引比-滑り率 特性式の次数 ( 1 )
M : 移動平均処理の移動データ点数 ( 20 )
N : 移動平均処理の平均データ点数 ( 3 )
A : 走行抵抗            ( 5000.000[kgf] )
B : 地盤の単位面積当たりの粘着力 ( 2700.000[kg/m2] )
C : 地盤の内部摩擦角    ( 24.000[度] )
D : 換算係数            ( 0.600 )
E : 地盤の変形係数      ( 150000.000 )
F : file name of bulldozer data ( bulldata.csv )
I : file name of input ground   ( ingnd.csv )
O : file name of output ground  ( outgnd.csv )
ESC : Exit This Menu
```

図2-23 条件設定画面

## 3) シミュレーションの実行

ブルドーザオペレーションシミュレータを実行する。

実行中は、ジョイスティックとキーボードが入力装置となる。以下にそれらについて説明する。

## 4) ブレード操作レバーの操作

排土板はCフレームにつけられ、Cフレームはトラックフレームに取り付けられている。排土板の上げ下げは油圧式による油圧シリンダで行う。排土板は角度が変化できる構造になっている。油圧シリンダは油タンクの中よりギアポンプによって送油される。それを運転席のレバーによって、操作弁のスプールを動かして油の通路の切り替えを行う。

操作弁を上下位置にすると土工板は油圧シリンダの工程の長さだけ上下に動き、中立位置にすると土工板は静止する。

油圧シリンダの推力はシリンダの有効面積に比例し、ピストン速度は有効面積に反比例する。

ブレードの操作にあたっては、微少な操作レバーの操作でブレードが動く場合、操作がシビアになる。また、走行中の振動等より誤操作の危険性が高くなる。そこで、ブレード操作レバーの中立位置の近傍はスプールにより、「不感帯」を設け、この範囲で操作レバーを動かしてもブレードが動かないようにしている。また、不感帯からブレード上げに切り替える領域では、ブレードの微少な操作が可能となるよう

にブレードレバーの変位に対するブレードシリンダの伸縮速度を低いものになっている。

ジョイスティックはこのようなブレード操作レバーの機能を模擬したものであり、シミュレータの実行中、操作者は、このジョイスティックを操作しブレードを上下させることとする。

その他、シミュレータの実行中にキーボードから入力できる項目として、以下がある。

・速度段の変更

ギアダウン “>” キー

ギアアップ “<” キー

・シミュレーションの中止 “6” または, “ESC” キー

### 3. まとめ

研究結果をまとめると以下の通りである。

#### (1) 建設機械の制御手法の検討

AI, ファジィ, ニューロについて, 制御への応用の現状を調査し, ブルドーザへの適用について検討した。建設機械の中でもブルドーザは一台の機械で掘削や押土, 盛土, 均しなどの複数の作業を行う汎用土工機械であるので, その制御を考えるに当たっては, 作業全体の計画やそれぞれの作業単位に適した方法を考える必要がある。

#### (2) AIを用いたブルドーザの制御システムの検討

ブルドーザを対象として, ニューラルネットワークを応用した制御システムの開発プロセスを提案した。また, 開発を進める上での技術的課題について検討した。

本調査では, 操作に未熟練なオペレータでも容易に操作可能な建設機械のイージーオペレーションシステムの開発を将来的な構想として検討したものであるが, パターン認識やデータ収集方法など, 今後解決すべき技術的課題が明らかとなった。その中でも, 本調査においては, ニューラルネットワークの学習用データの収集等を効率的に実施するための開発支援システムとしてブルドーザ・オペレーション・シミュレータの開発に着目し, 検討した。

#### (3) ブルドーザ・オペレーション・シミュレータの製作

シミュレータに必要な基本的な機能について検討し, ブルドーザ本体と地盤との相互作用について簡略化した形でシミュレータに組み込むこととして試作機を製作した。今回製作したシミュレータでは, テラメカニクスの理論を活用することで, 建設機械の挙動を考える上で重要な地盤との相互作用について考慮することができた。

本研究では, 上記の検討を行い, AIを用いたブルドーザの制御について, その制御方法と開発プロセス, 及びオペレーションシミュレータについて検討を行い, それらについて提案した。今後の実用化を想定すると, 更なる技術的課題の検討を行うことが必要である。また, シミュレータについては, 実機の挙動との比較など, 実験等を通じた検討, 検証を進める必要がある。今後の主な技術的課題として, 以下が考えられる。

##### ① AIによる制御システムの実現に向けた課題

- a. 建設機械の動作分析等 (基本的な動作形態及び一連の作業の流れの整理)
- b. 機械による作業計画技術 (機械に自立的に作業計画を立案・実行させる)
- c. 機械によるパターン認識 (機械により実行中の作業状況・内容を認識させる)

##### ② シミュレータについての課題

- a. 実機の挙動との比較による, シミュレータの妥当性の検証・検討
- b. 操作インタフェースの違いによる, 操作感の差異についての把握
- c. シミュレータの改良 (機械-地盤系特性の更なる検討)

- ・走行による地盤の締め固め効果（土質の変化等）
- ・ブレード前方及び側方に滞留する掘削土とそれによる走行抵抗の算定
- ・地盤載荷実験及び掘削実験等を通じた、地盤特性と走行・掘削との関係の整理

## おわりに

本調査では、ブルドーザの制御システムにAIを応用することを検討した。また、オペレータの操作データを再現性の高い条件下で収集することを目的にブルドーザ・オペレーション・シミュレータを試作した。本資料が、今後の建設機械の自動化、イージーオペレーションシステム等の研究開発に役立つことを願う次第である。

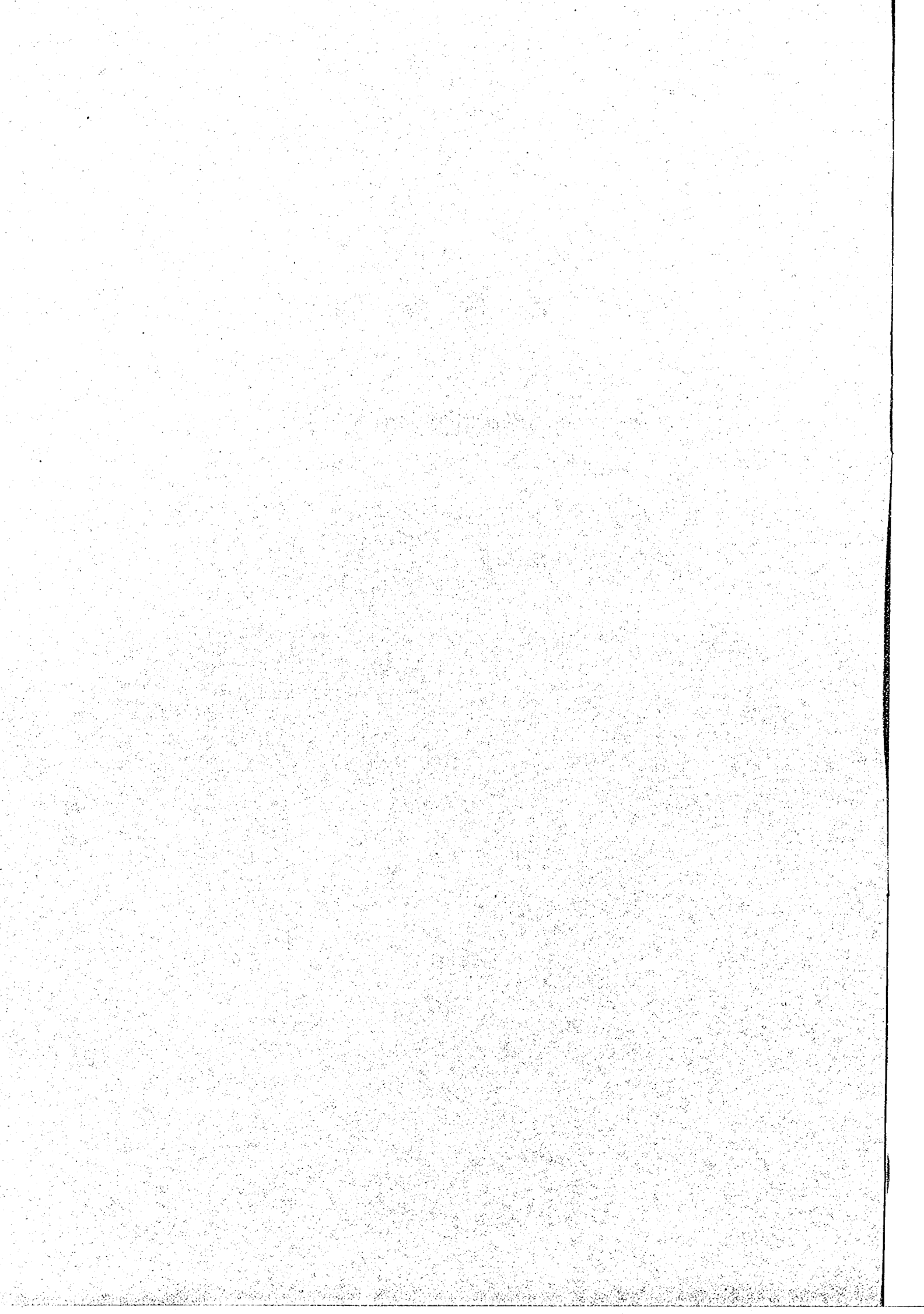
最後に、シミュレータの試作等にあたって、筑波大学油田信一教授をはじめ、日本建設機械化協会自動化委員会制御小委員会の皆様に多大なご指導、ご協力をいただいた。ここに記して感謝の意を表すものである。

## 【参考文献】

- 1) (財)計測自動制御学会：「ニューロ・ファジィ・AIハンドブック」，オーム社，1994. 5
- 2) 石坂誠一：「メカトロニクス 脚光をあびる複合技術」，通商産業調査会，1985. 10
- 3) 室達朗：テラメカニクスー走行力学ー，技報堂出版，1993

## 添 付 資 料

1. ブルドーザの挙動計測実験データ
2. ブルドーザ・オペレーション・シュミレータ  
プログラムリスト



## 1. ブルドーザの作業の計測データ



# 1. 実験方法

## 1) 実験対象建設機械

実験の対象とする建設機械として、移動と作業を同時に行う（動きながら作業を行う）ために操作が複雑で、熟練者と未熟練者の差異が大きいブルドーザを選定した。

実験には、土木研究所所有のブルドーザを用いた。表1-1に諸元を示す。

表1 実験に用いたブルドーザの諸元

【型 式】 D50A-15型（製造：1972年）

（車体SERIALNO.41064/ENG SERIALNO.56541/ダイレクトミッション）

【諸 元】

|         |      |        |                         |
|---------|------|--------|-------------------------|
| ・走行速度   | 回送時  | 1 速    | 0 ~ 2.5 (km/h)          |
|         |      | 2 速    | 0 ~ 3.5 (km/h)          |
|         |      | 3 速    | 0 ~ 5.5 (km/h)          |
|         | 作業時  | 4 速    | 0 ~ 9.4 (km/h)          |
|         |      | 1 速    | 0 ~ 3.2 (km/h)          |
|         |      | 2 速    | 0 ~ 5.2 (km/h)          |
|         | バック時 | 3 速    | 0 ~ 8.0 (km/h)          |
|         |      |        |                         |
|         |      |        |                         |
| ・形状・寸法他 | 車体   | 重量     | 1470 (mm)               |
|         |      | 全長     | 4700 (mm)               |
|         |      | 全幅     | 3550 (mm)               |
|         |      | 全高     | 2690 (mm) (排気管を含む)      |
|         | 履帯   | 接地長    | 2085 (mm) (ホイールのセンター間隔) |
|         |      | 幅      | 400 (mm)                |
|         |      | 全幅     | 2000 (mm) 左右の履帯の外幅      |
|         | ブレード | 幅      | 3350 (mm)               |
|         |      | 高さ     | 855 (mm)                |
|         |      | 刃先角度   | 55°                     |
|         | シリンダ | ストローク長 | 160 (mm)                |
|         |      | 内圧     | 140 (km/cm) (最大圧力)      |
|         |      | ストローク長 | 160 (mm)                |
|         |      | 伸縮速度   |                         |
|         | レバー  | ストローク長 | 33°                     |

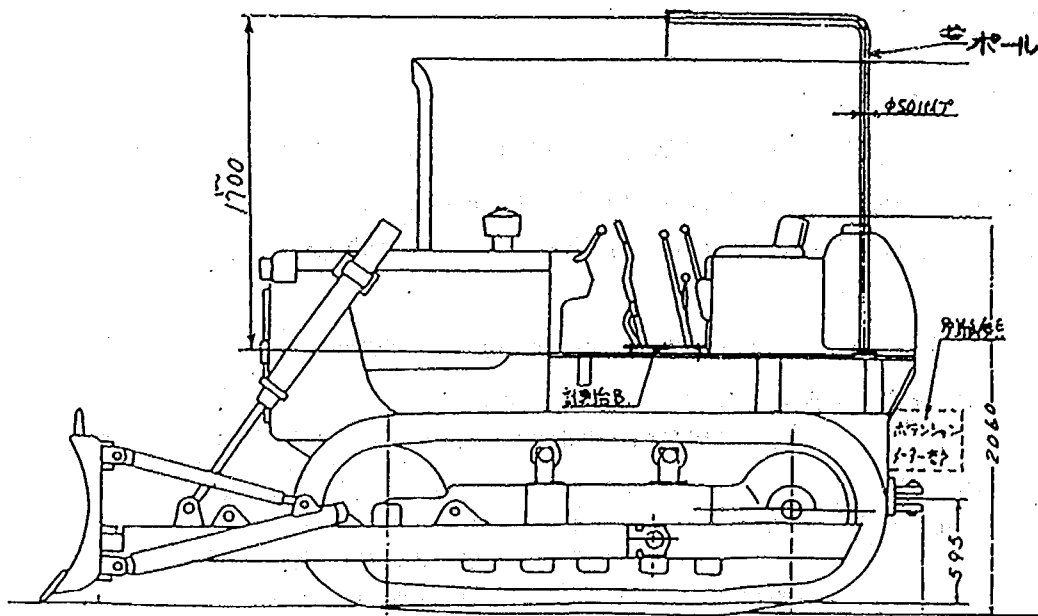


図1 ブルドーザの外観

## 2. 測定項目

実験における測定項目は、オペレータの操作及び注視野、ブルドーザの動き及び状態とした。

オペレータの操作を計測する方法としては、作業装置であるブルドーザの操作を行うためのブレードリフトレバーの操作量をポテンショメータで、前後進および変速レバー位置をリミットスイッチで測定するものとした。

オペレータがどこを見ながら運転しているかを調べる方法としては、アイ・カメラを用いた注視野と注視点（視線）の測定を行うものとした。

ブルドーザの動きおよび状態については、車体傾斜角、ブレードリフトシリンダ伸縮量、燃料消費量、車体加速度について測定するものとし、それぞれ、バーチカルジャイロ、ポテンショメータ、燃料流量計、三軸加速度計で計測するものとした。

また、ビデオ撮影（地上の定点およびオペレータの頭上より）及び実験前後の地形を計測することにより、これらを補完することとした。

さらに、試験の直前直後にオペレータに対し別表のインタビューを行い、作業中の状況判断の根拠などを調べた。

試験は、2人のオペレータが同一の作業を各2回行っている。各2回の繰り返し試験時の条件の均質性、試験の再現性を確認するために、試験実施の前と後に土質の計測・分析も実施した。

表2に今回測定する対象と測定器を示す。

表2 測定対象と測定器

| (計測対象)                | (計測項目)                        | (測定器)                        |
|-----------------------|-------------------------------|------------------------------|
| オペレーターの注視             | オペレーター注視野                     | アイ・カメラ                       |
| オペレーターの操作             | ブレードリフトレバー操作量<br>変速レバー位置      | ポテンシオメータ<br>リミットスイッチ         |
| 機械の動き, 状態             | 車体傾斜角                         | バーチカルジャイロ                    |
|                       | ブレードリフトシリンダ伸縮量                | ポテンシオメータ                     |
|                       | エンジン負荷                        | 燃料計                          |
|                       | 車体位置 (速度)<br>加速度 (左右, 前後, 上下) | ポテンシオメータ<br>加速度計             |
| オペレータの動作及び作業状態, 作業量   |                               | ビデオ撮影                        |
| ブレード操作レバー, 変速レバーなどの位置 |                               | (オペレータの操作状況及び<br>作業の全体状況を撮影) |
| ブルドーザの動作              |                               |                              |
| オペレータの状況判断            |                               | インタビュー (各作業終了直後に行う)          |
| 地形計測                  |                               | 光学式計距計角儀                     |
| 土質分析 (試験実施前後に実施する)    |                               |                              |

### 3. 測定方法

各測定器からの信号をデータレコーダに入力できるように接続し, 記録した。データの取り込み頻度は, オペレータおよびデータレコーダよりデータ解析用のパソコンへのデータ転送機能などから, 11.8 Hzとした。

また, 作業全体の状況を実験フィールド側面の定位置に接地したビデオ撮影機で, オペレータの操作状況を操作席真上に取り付けたビデオ撮影機でそれぞれ撮影した。

なお, 実験にあたっては, 稼働状況データとビデオ映像が時間的に照合でき, また各動作についてのデータを区分できるように, ビデオカメラとデータレコーダにそれぞれ内蔵している時計の時間を整合をとるようにした。

実験での測定系の構成を図-2に示す。

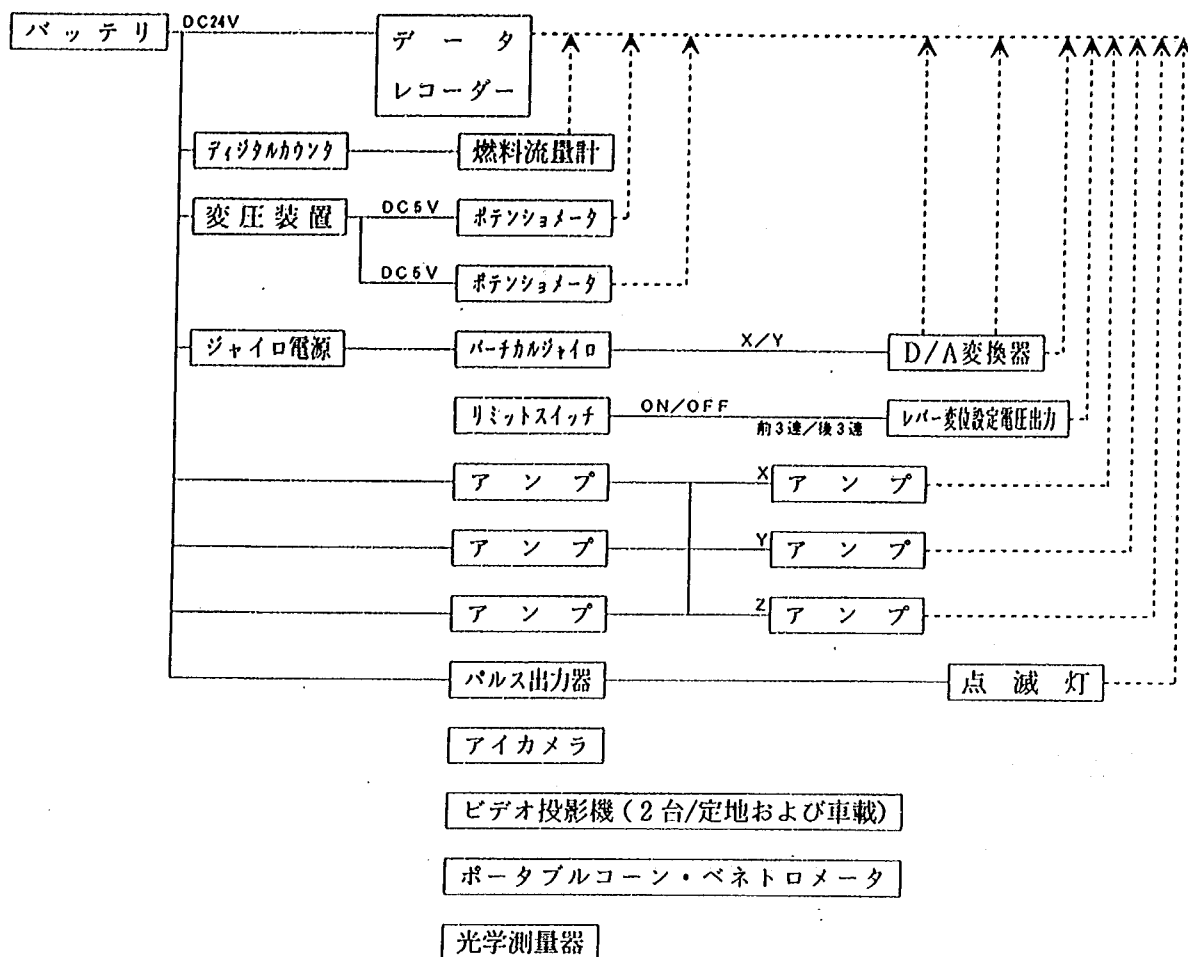


図2 測定系の構成

なお、実験フィールドの形状（地形）については、試験開始前後に表-3に示す機器で計測を行った。

表3 地形測定機器一覧

| No. | 機器名称    | 型式・仕様など                                                                    | 台数 |
|-----|---------|----------------------------------------------------------------------------|----|
| 1   | 光学式測距角儀 | SET2B (株ソキア)<br>測定精度 3mm + 2ppm・D 測定精度 H: 2 <sup>m</sup> V: 4 <sup>m</sup> | 1台 |
| 2   | 三脚      | SET2B用, バックサイト用                                                            | 2脚 |
| 3   | プリズム    | 測定用, バックサイト用                                                               | 2個 |
| 4   | ポール     | 測定プリズム                                                                     | 1本 |
| 5   | スチロンテープ | 30M用                                                                       | 1巻 |
| 6   | 消石灰     | 10kg                                                                       | 2袋 |
| 7   | その他     | ポリバケツ, 柄杓                                                                  |    |

測定の目的は、実験の前後に実験ヤードの地形測量を行い、ブルドーザが掘削・押し土した土量と形状を測定する事である。測量は表-4の手順によった。

表4 測量の手順

| No. | 手 順     | 作 業 内 容                                                         |
|-----|---------|-----------------------------------------------------------------|
| 1   | 測 量 準 備 | 基準となるX軸, Y軸線を設置する。25×25mの大きさに1mピッチのメッシュラインを石灰で地上にマーキングする。       |
| 2   | 実験前測量   | 1mあるいは2mのメッシュ交点上にプリズムを置き水準測量する。                                 |
| 3   | 実験前測量   | ブルドーザが掘削・押し土した範囲内のメッシュ交点上にプリズムを置き、座標測定を行い、メッシュ交点上にあるかXY座標で確認する。 |

#### 4. 計測器取り付け状況

実験時のブルドーザ上の衝撃や振動等による計測器や記録装置への影響については、防振材等を使用することで軽減させることとした。

それぞれの計測器や記録装置の取り付け位置を図-3に示す。

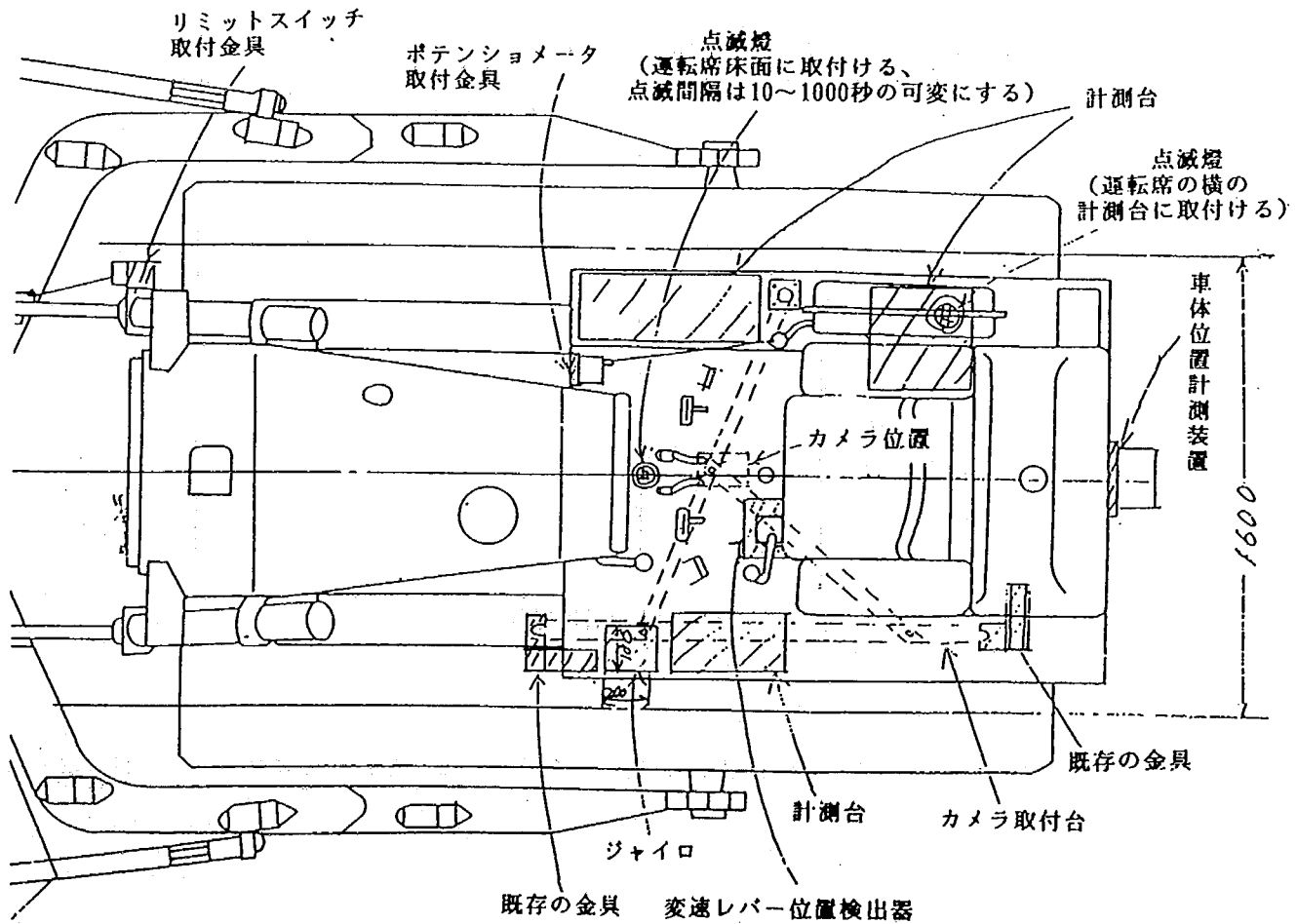


図3 計測器や記録装置の取り付け位置

## 5. 実験条件

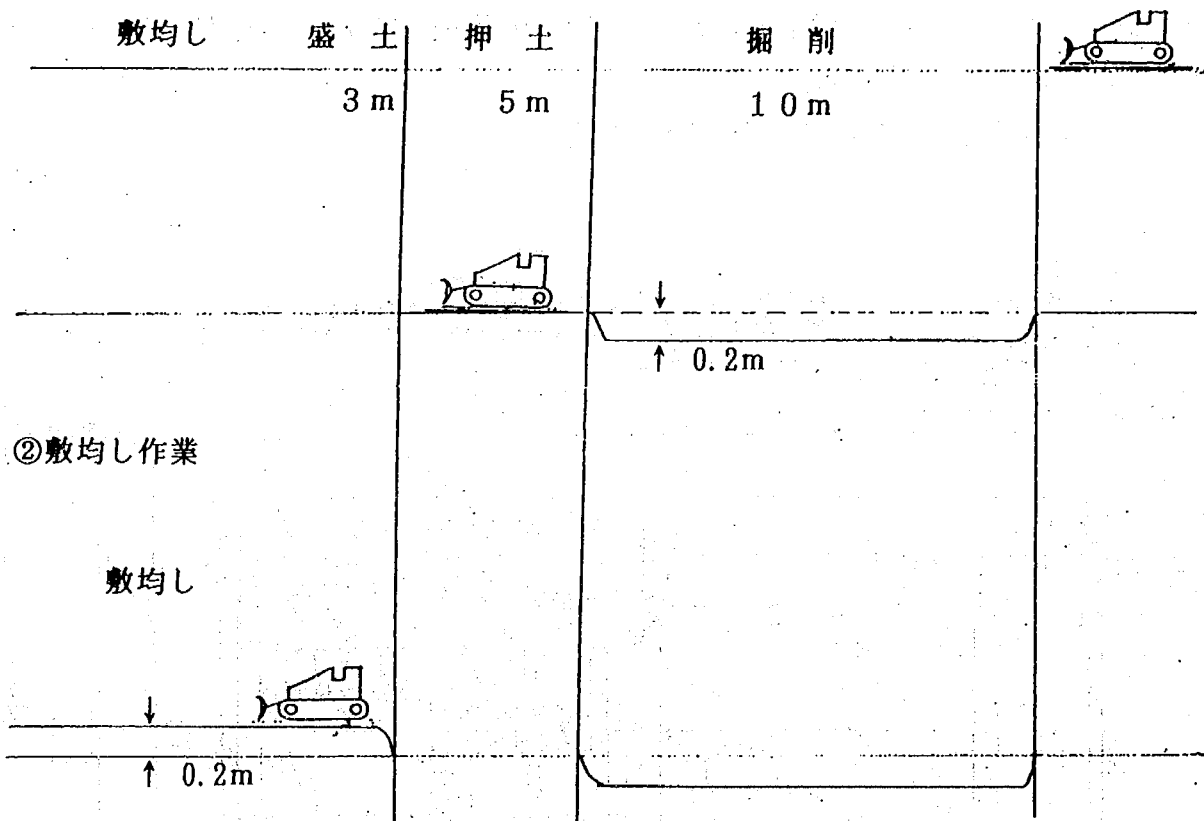
### 1) 計測対象作業

計測対象作業は、建設機械施工士試験を参考に次の通りとした。

- 掘削は最終掘削深さ0.2mとし、数回（標準4回とする）にわけて掘削を行う。
- 盛土は盛土領域3mの範囲に行い、高さ等の形状については指定しない。
- 敷均しは最終高さが0.2mとなるように行う。

概略を図-4に示す。

#### ① 掘削、盛土、押土



#### ② 敷均し作業

図4 計測対象動作

### 2) 実験場所

建設省土木研究所構内（建設機械屋外実験場）で行うものとした。なお、実験フィールドは、図-5の通りである。

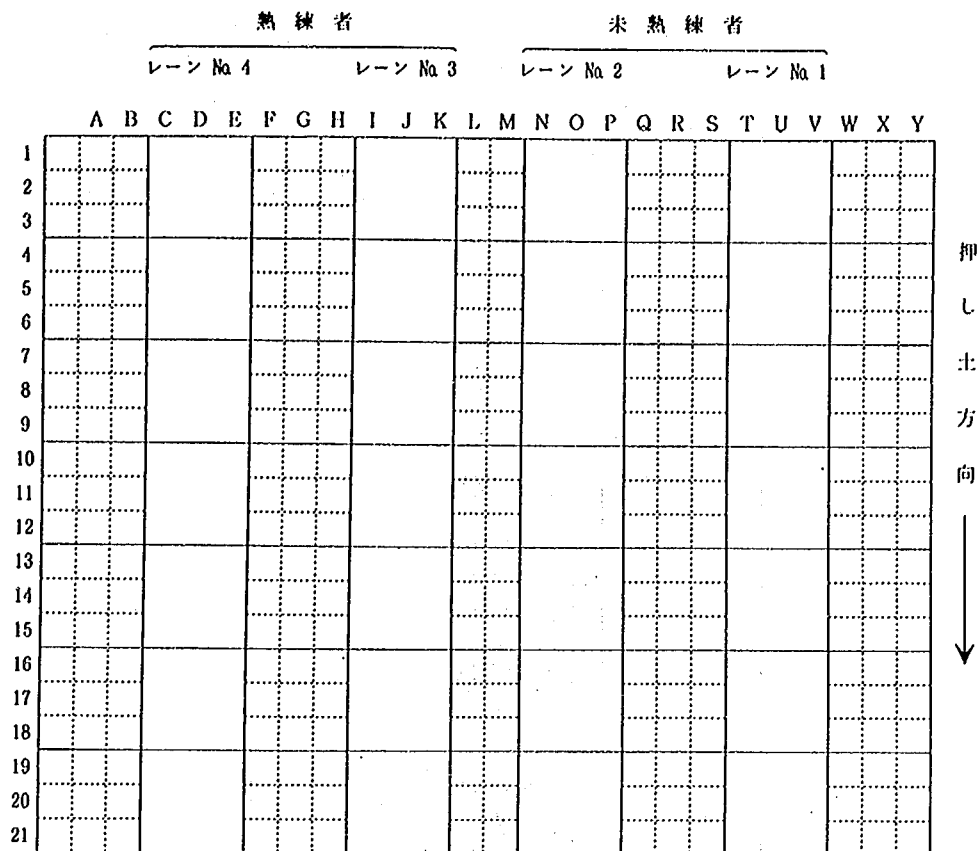


図5 実験フィールド

フィールドは全部で4レーンとした。後述の熟練、未熟練の2人のオペレータが各2回ずつ作業を行う事を想定したものである。

今回設定したレーンの幅員は6mとした。これは、実験に利用したブルドーザのブレード幅3.35mを前提として、レーン間の排土の干渉なしに十分に安定したデータが採取できる領域の確保を考慮したものである。

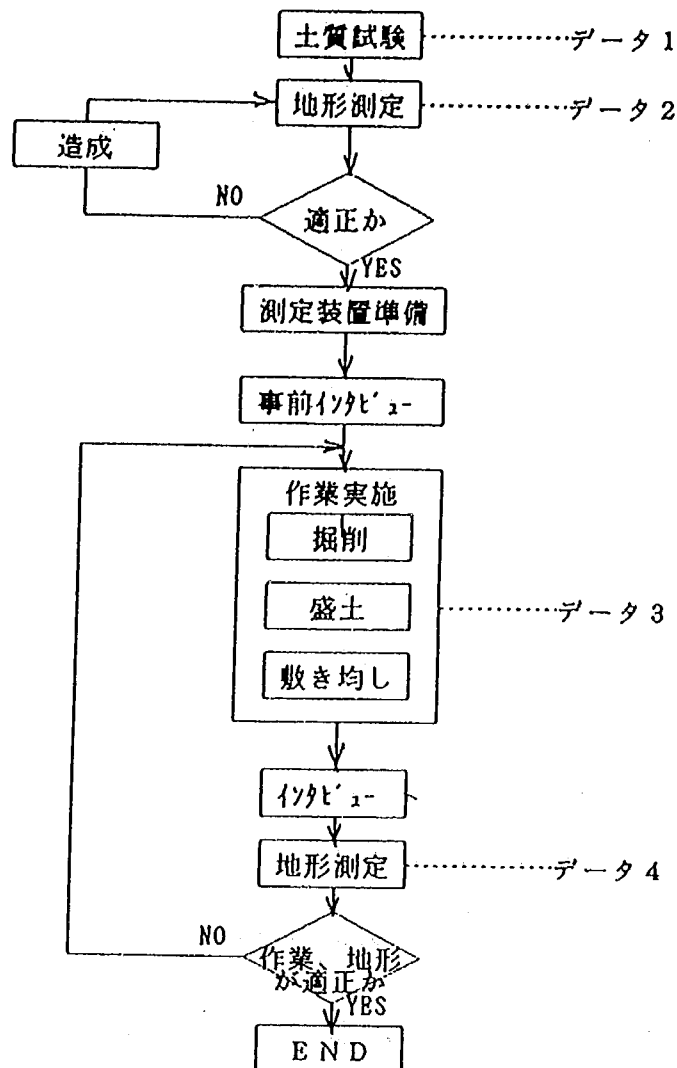
実験フィールドの長さについては、図-4の試験条件を十分に満たすものであった。

### 3) オペレータ

オペレータは、熟練したオペレータA（経験16年）と初心者のオペレータB（経験6カ月）をそれぞれ1名とした。

## 6. 実験手順

ブルドーザの作業測定に先だって、土質試験、地形測定とオペレータへの事前のインタビューをおこなった。作業測定終了後も、インタビューによるオペレータの計画（意図）と結果の差異などについての感想、地形測定による作業量の定量的な把握を行った。図-6に実験手順の全容を示した。



データ 1 : 含水率, 比重, 密度, 粒度分布

データ 1 : 地表高

データ 1 : オペレータの注視野, ブレードリフトレバー操作量, シリンダ伸縮量, 変速レバー位置,  
車体傾斜角, 燃料消費量, 車体加速度

データ 1 : 地表高

図 6 実験手順

## 7. キャリブレーション結果

データレコーダに記録したデータのキャリブレーションの結果を以下に示す。

なお, ブレードリフトシリンダ伸縮量, 車体傾斜角 (ピッチング), 車体振動加速度 (X, Y, Z), 車体傾斜角 (ローリング), ブレードリフトレバー操作量についてのデータは, 図-7 に示した取扱い (正負の方向) をするものとした。

### (I) 車体傾斜角 (ピッチング)

$$Y_1 = 12.8X_1 - 2.2$$

$Y_1$  : 角度 (°)

$X_1$  : 出力 (V)

(2) 車体傾斜角 (オーリング)

$$Y_2 = 12.8X_2$$

$Y_2$  : 角度 (°)

$X_2$  : 出力 (V)

(3) ブレードリフトレバー操作量

$$Y_3 = 53.1X_3 - 69.2$$

$Y_3$  : 角度 (°)

$X_3$  : 出力 (V)

(4) ブレードリフトシリンダ伸縮量

$$Y_4 = 27.3X_4 - 89$$

$Y_4$  : 伸縮量 (cm)

$X_4$  : 出力 (V)

(5) 燃料消費量

$$Y_5 = 20.0X_5$$

$Y_5$  : 燃料消費量 (L/HR)

$X_5$  : 出力 (V)

(6) 車体加速度

$$Y_6 = 20.0X_6$$

$Y_6$  : 車体加速度 (G)

$X_6$  : 出力 (V)

(7) 変速レバー位置

前進 1 速 : 4 (V)    前進 2 速 : 5 (V)    前進 3 速 : 6 (V)

後進 1 速 : 3 (V)    後進 2 速 : 2 (V)    後進 3 速 : 1 (V)

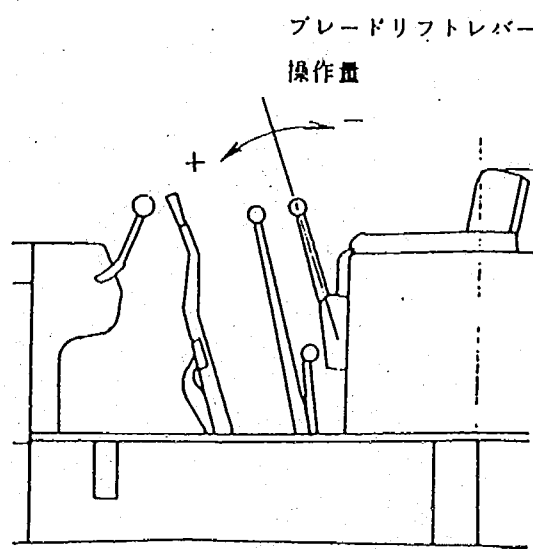
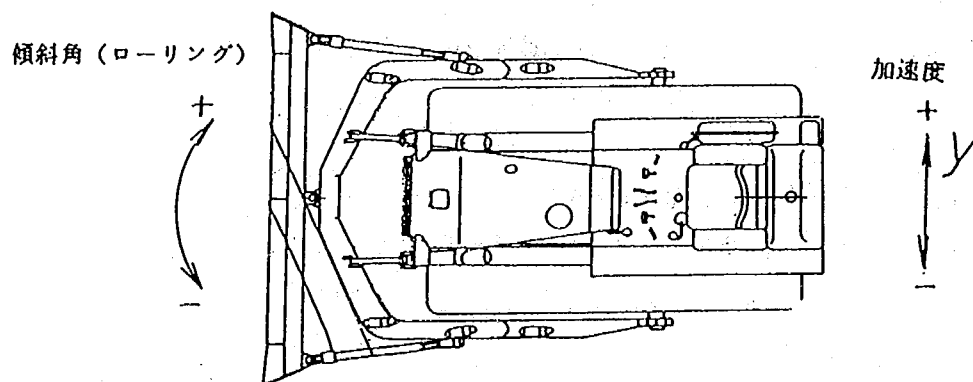
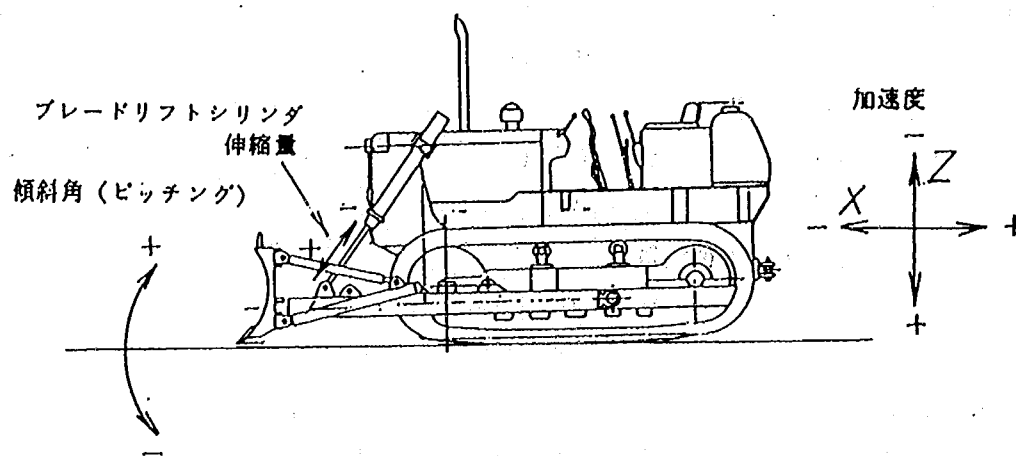


図7 データとその正負の定義

## 8. 実験結果及び考察

### 1) 掘削後の地形

地形測定の結果を表-5に示す。

熟練者は、例えば掘削すべき区間と押土すべき区間といった区分のように各区間の区分がはっきりしているのに対し、未熟練者においてはこの区分が曖昧な点がある。これは、各作業の開始地点で熟練者が的確に掘削・押土・盛土といった作業を適切な操作により、区分がされているものと思われる。

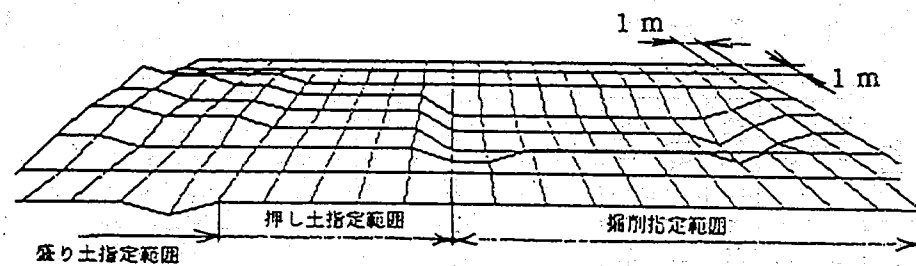
また、未熟練者においては、掘削面に段差がみられる。熟練者においても押土・盛土は適切におこなわれているが、掘削すべき区間において、掘削作業開始地点で若干の掘削し過ぎがみられる。このことを後に述べるブレードリフトシリンダ伸縮量の計測結果と比較してみると、掘削作業の開始のところで伸縮量が大きく伸びているのがよみとれる。これは、地表が硬かったためにブレードを下げすぎた等の理由によるものと考えられる。

(m)

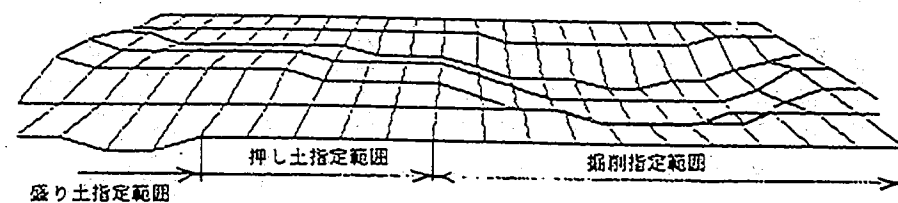
注) 計測地点区画状況

し  
土  
方  
向

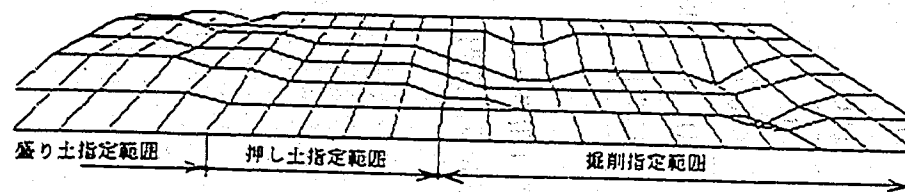
— 56 —



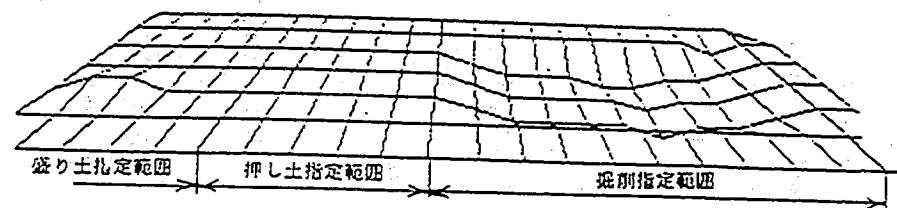
熟練者 (1 回目)



未熟練者 (1 回目)



熟練者 (2 回目)



未熟練者 (2 回目)

図7 測量結果

## 2) 作業に要した時間

操作の所定時間は、熟練者では、1回目は約578秒、2回目は約432秒、未熟練者では、1回目は約605秒、2回目は約516秒となっており、熟練者の方が作業に要した時間が短かった。

熟練者、未熟練者ともに作業手順としては、掘削・押土・盛土・後進からなるサイクルと敷均し・後進からなるサイクルに区分して作業を行った（作業条件）が、それぞれ個々の作業について、かかった時間の合計と平均を調べた結果を表-6に示す。なお、ここでは1サイクル当りの掘削等の作業量は、考慮していない。

トータルの作業時間はどの作業についても1回目より2回目が短くなっており、また熟練者と未熟練者の間には大きな違いは見られなかった。作業時間の平均値は掘削-押土-盛土-後進サイクル（1回目、2回目）における押土作業と敷均し-後進サイクル（1回目）における後進を除いて熟練者の方が未熟練者よりも数秒短い傾向がある。

また、未熟練者については個々の作業の合間において停止（動作無し）がみられた。

表6 作業時間の合計・平均

(単位：秒)

| 操 作 者   |                | 切削-押土-盛土-後進サイクル |     |     |      | 敷均し-後進サイクル |      | 無動作 |
|---------|----------------|-----------------|-----|-----|------|------------|------|-----|
|         |                | 掘 削             | 押 土 | 盛 土 | 後 進  | 敷均し        | 後 進  |     |
| 熟 練 者   | トータル作業時間       | 175             | 85  | 56  | 164  | 58         | 34   | 0   |
| (1 回 目) | 1サイクル当りの平均作業時間 | 13.5            | 6.5 | 4.3 | 13.7 | 14.5       | 11.3 |     |
| 熟 練 者   | トータル作業時間       | 140             | 62  | 34  | 109  | 48         | 34   | 0   |
| (2 回 目) | 1サイクル当りの平均作業時間 | 14.0            | 6.2 | 3.4 | 12.1 | 12.0       | 8.5  |     |
| 未熟練者    | トータル作業時間       | 150             | 59  | 50  | 177  | 73         | 51   | 19  |
| (1 回 目) | 1サイクル当りの平均作業時間 | 15.0            | 5.9 | 5.0 | 19.7 | 14.6       | 10.2 |     |
| 未熟練者    | トータル作業時間       | 141             | 46  | 32  | 154  | 55         | 38   | 43  |
| (2 回 目) | 1サイクル当りの平均作業時間 | 15.7            | 5.1 | 3.6 | 19.3 | 13.8       | 9.5  |     |

## 3) 車体に取り付けた計測器による計測データ

以下に、車体に取り付けた計測器による計測データの例を示す。

ブレードリフトレバー  
操作量(度)

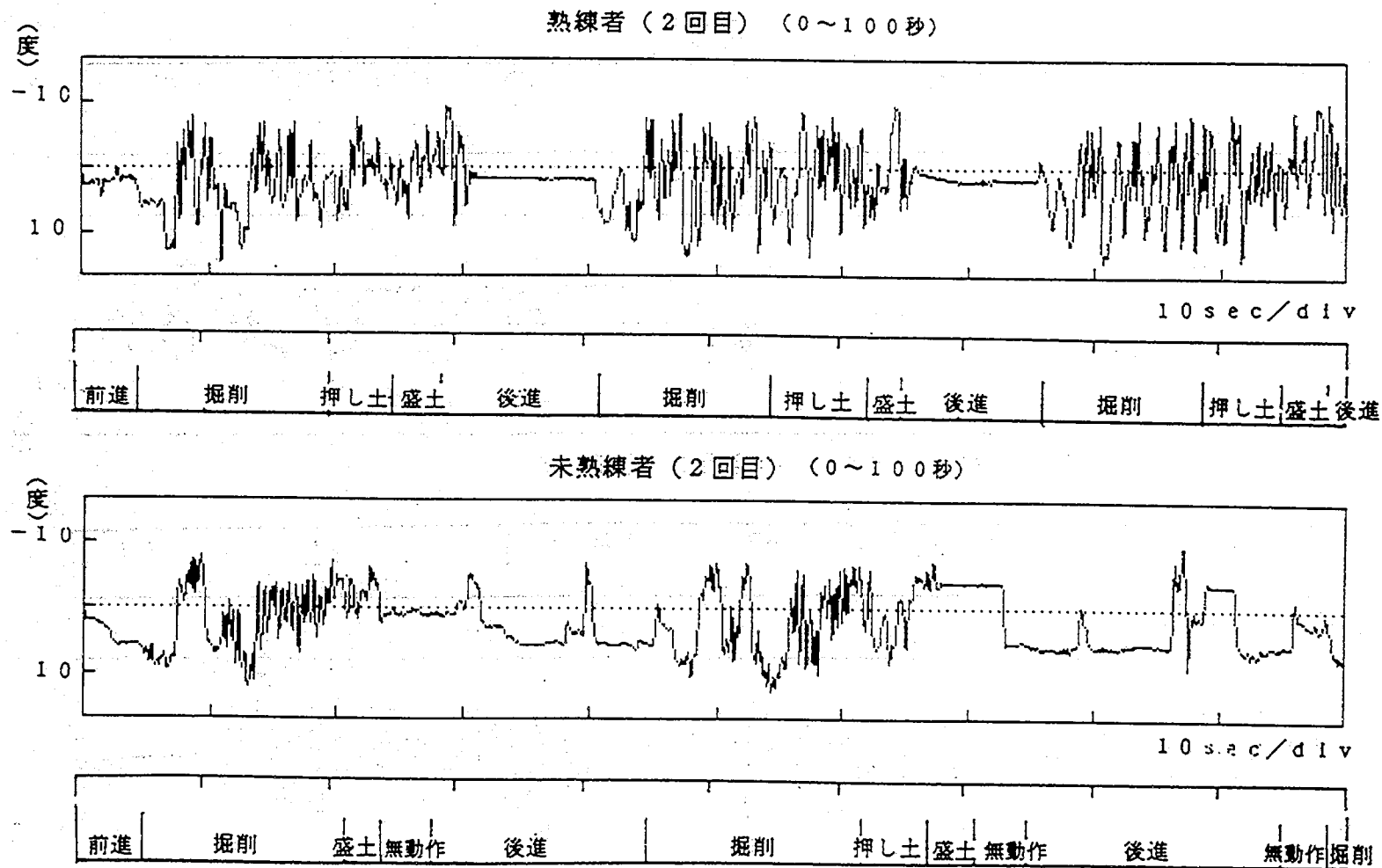
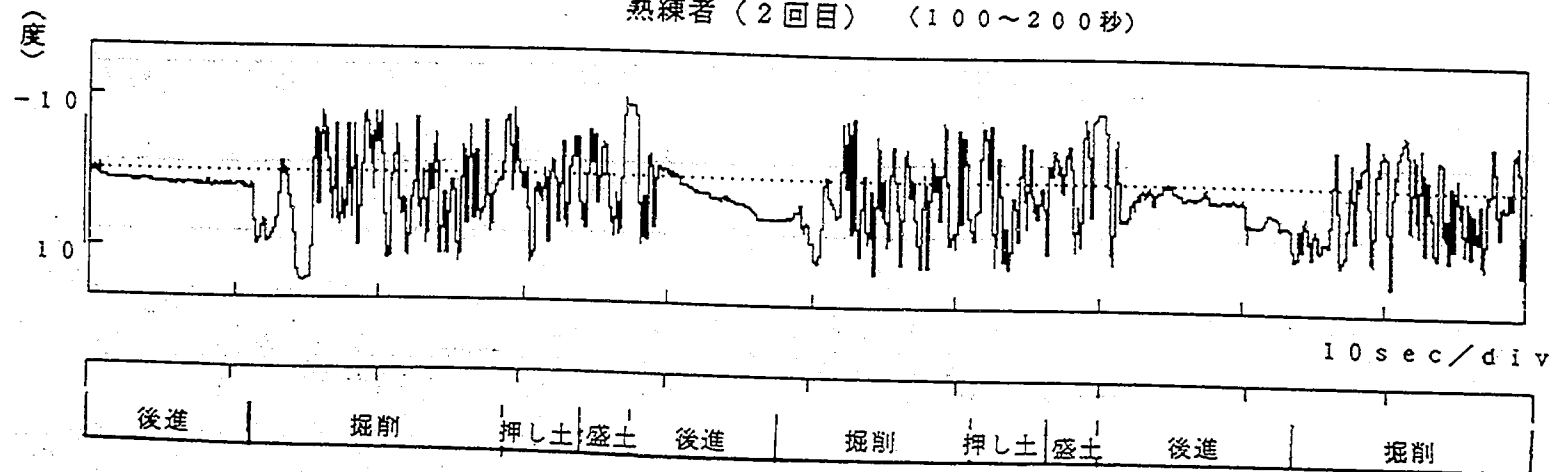


図8 ブレードリフトレバー操作量

ブレードリフトレバー  
操作量

熟練者 (2回目) (100~200秒)



未熟練者 (2回目) (100~200秒)

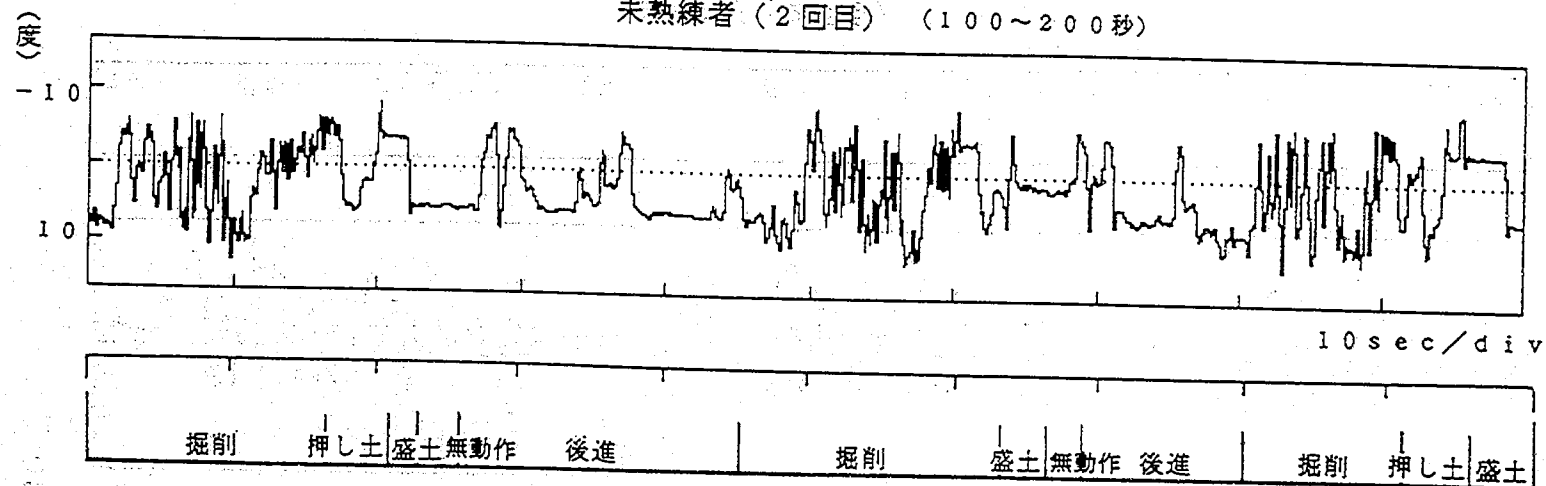


図9 ブレードリフトレバー操作量

ブレードリフトレバー  
操作量

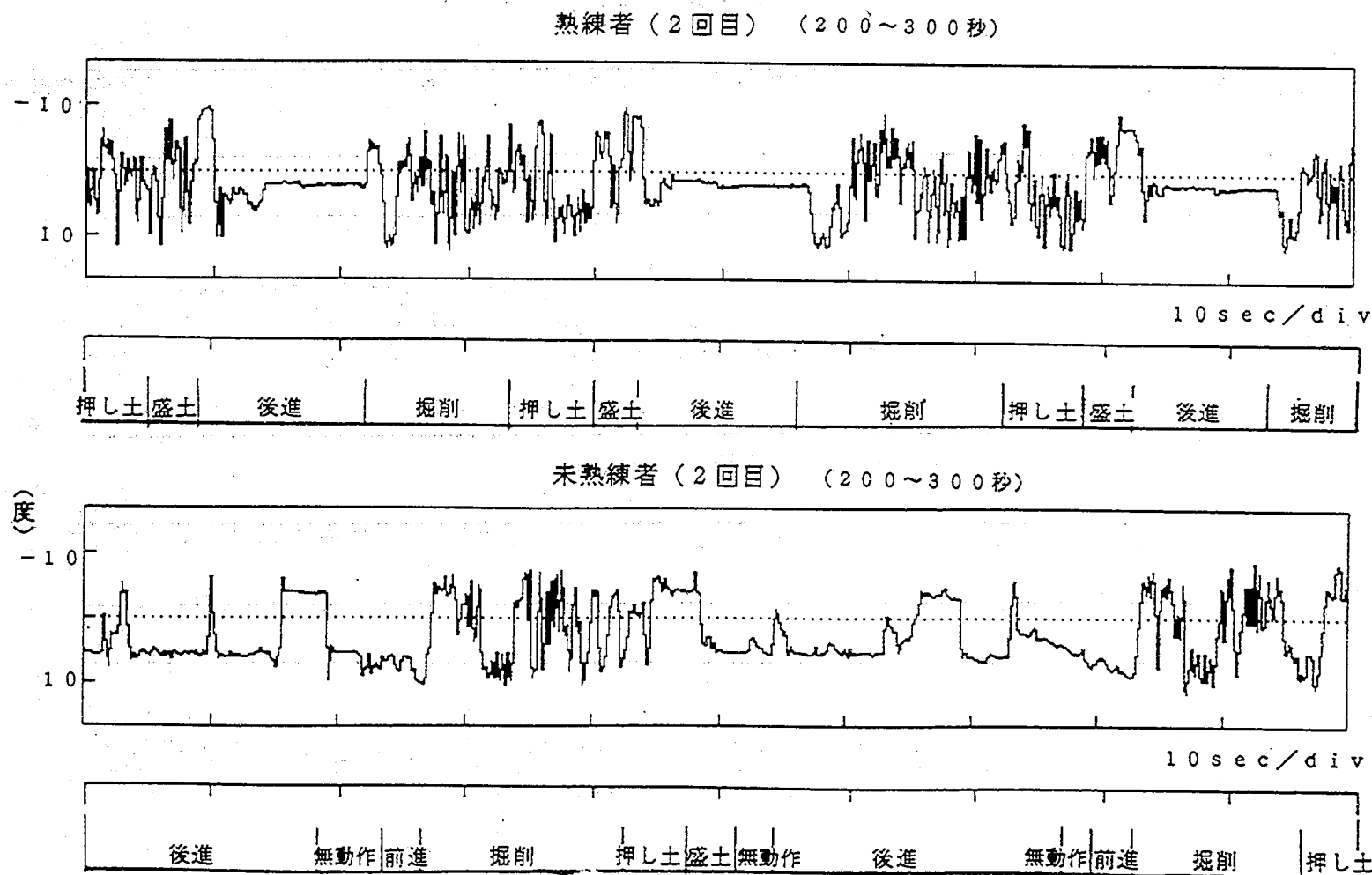
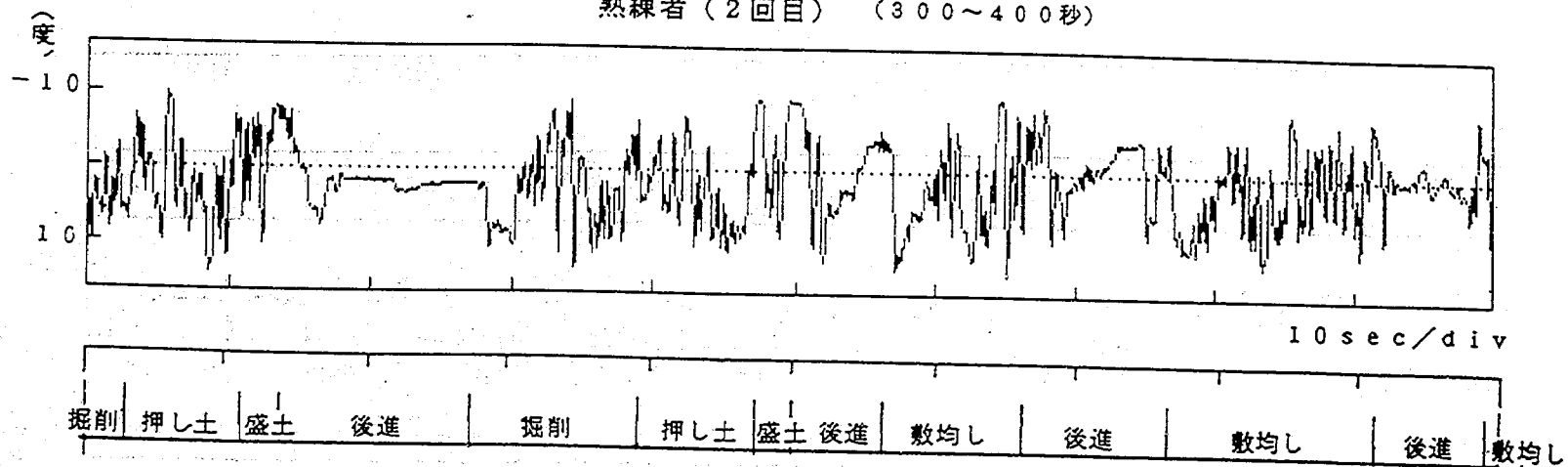


図10 ブレードリフトレバー操作量

ブレードリフトレバー  
操作量

熟練者 (2回目) (300~400秒)



未熟練者 (2回目) (300~400秒)

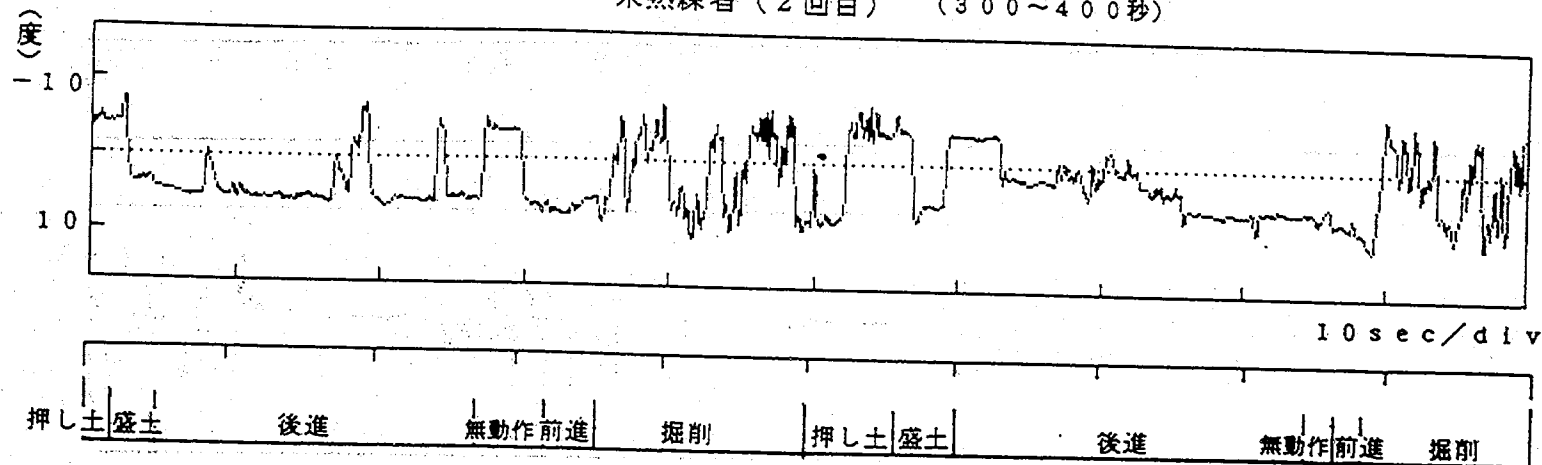
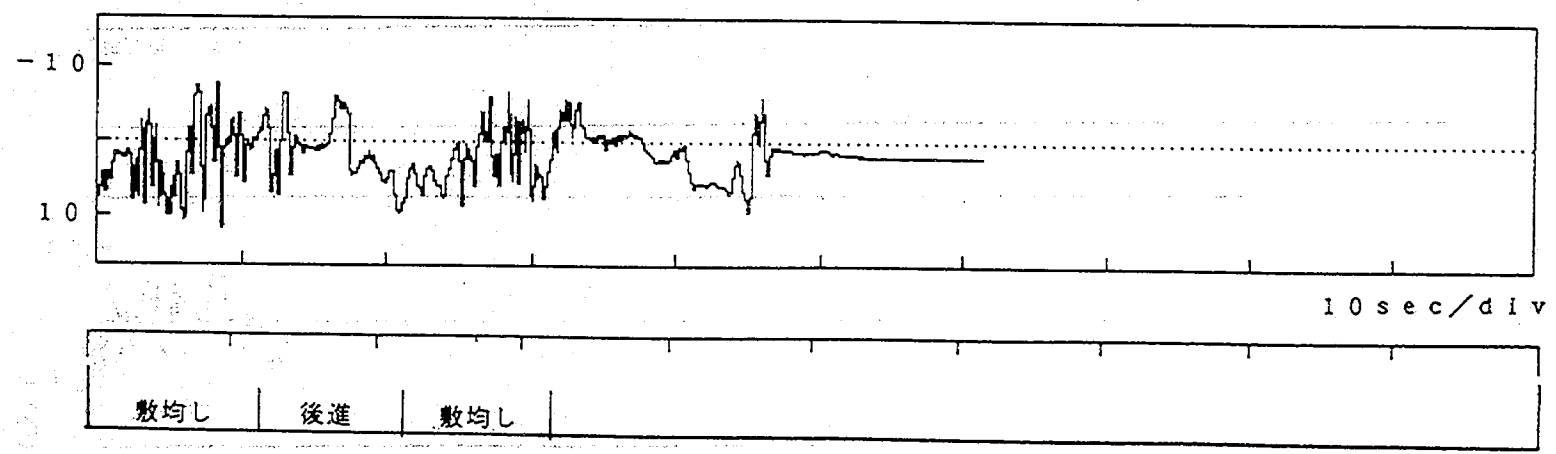


図11 ブレードリフトレバー操作量

ブレードリフトレバー  
操作量

熟練者 (2回目) (400~500秒)



未熟練者 (2回目) (400~500秒)

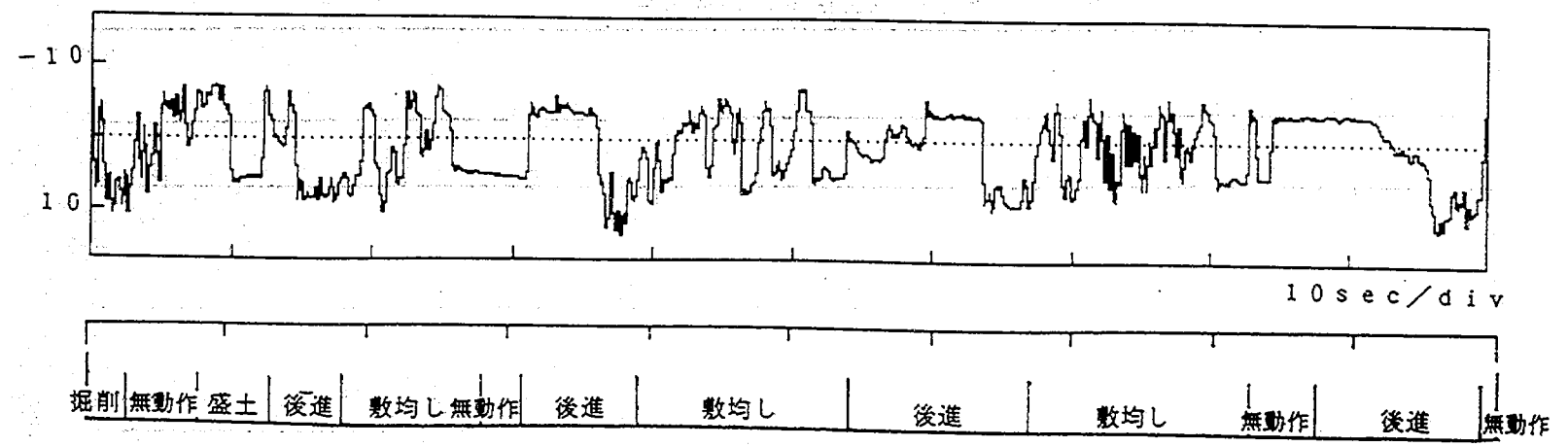
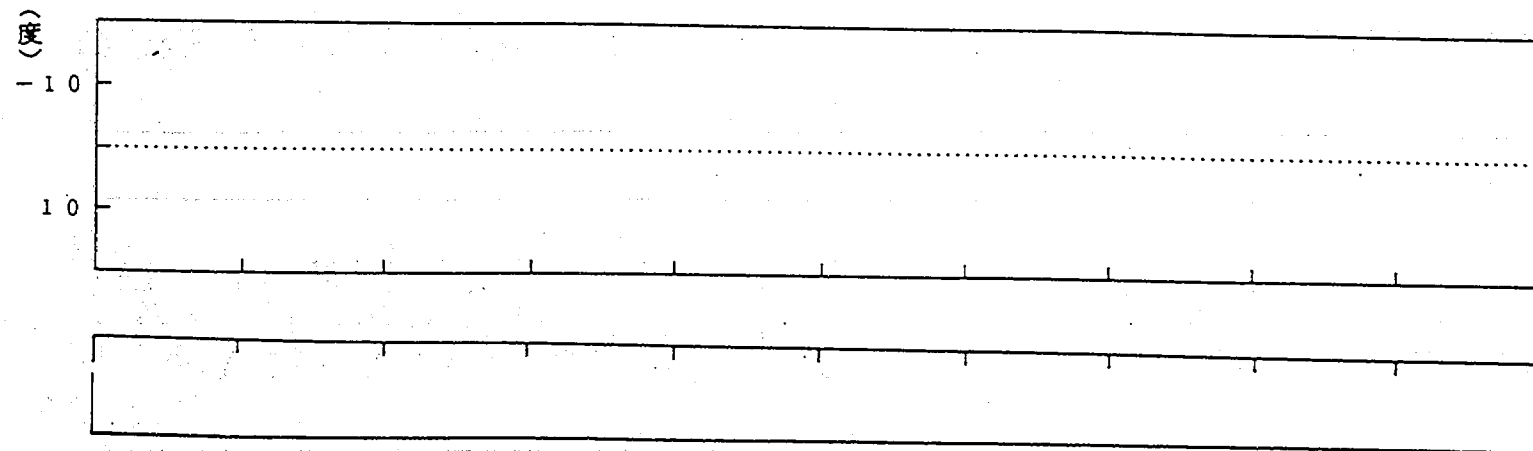


図12 ブレードリフトレバー操作量

熟練者（2回目）（500～600秒）



未熟練者（2回目）（500～600秒）

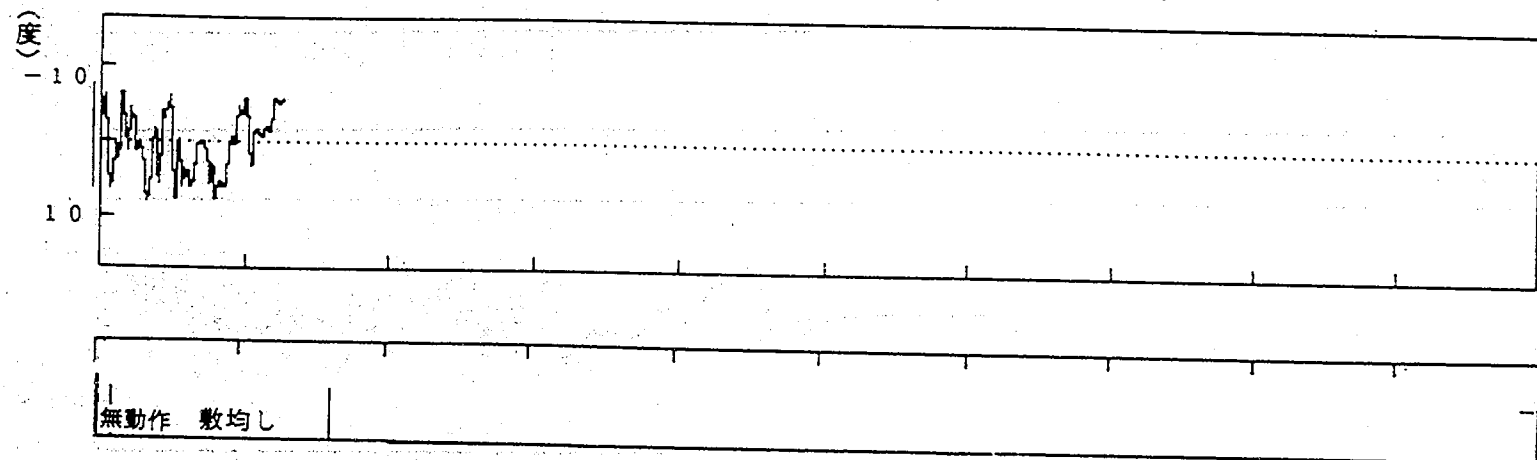


図13 ブレードリフトレバー操作量

ブレードリフトシリンダ  
伸縮量 (cc)

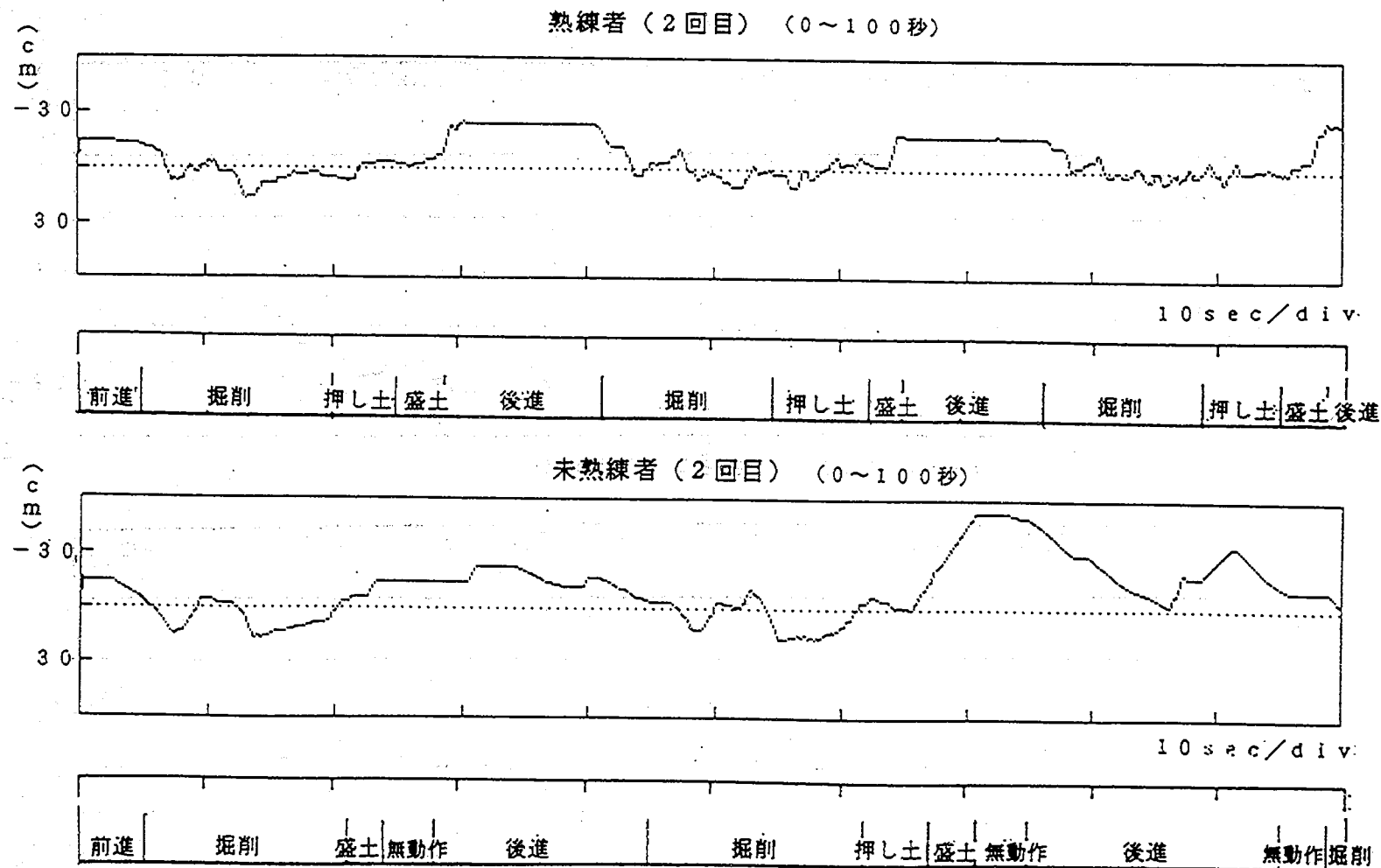


図14 ブレードリフトシリンダ伸縮量

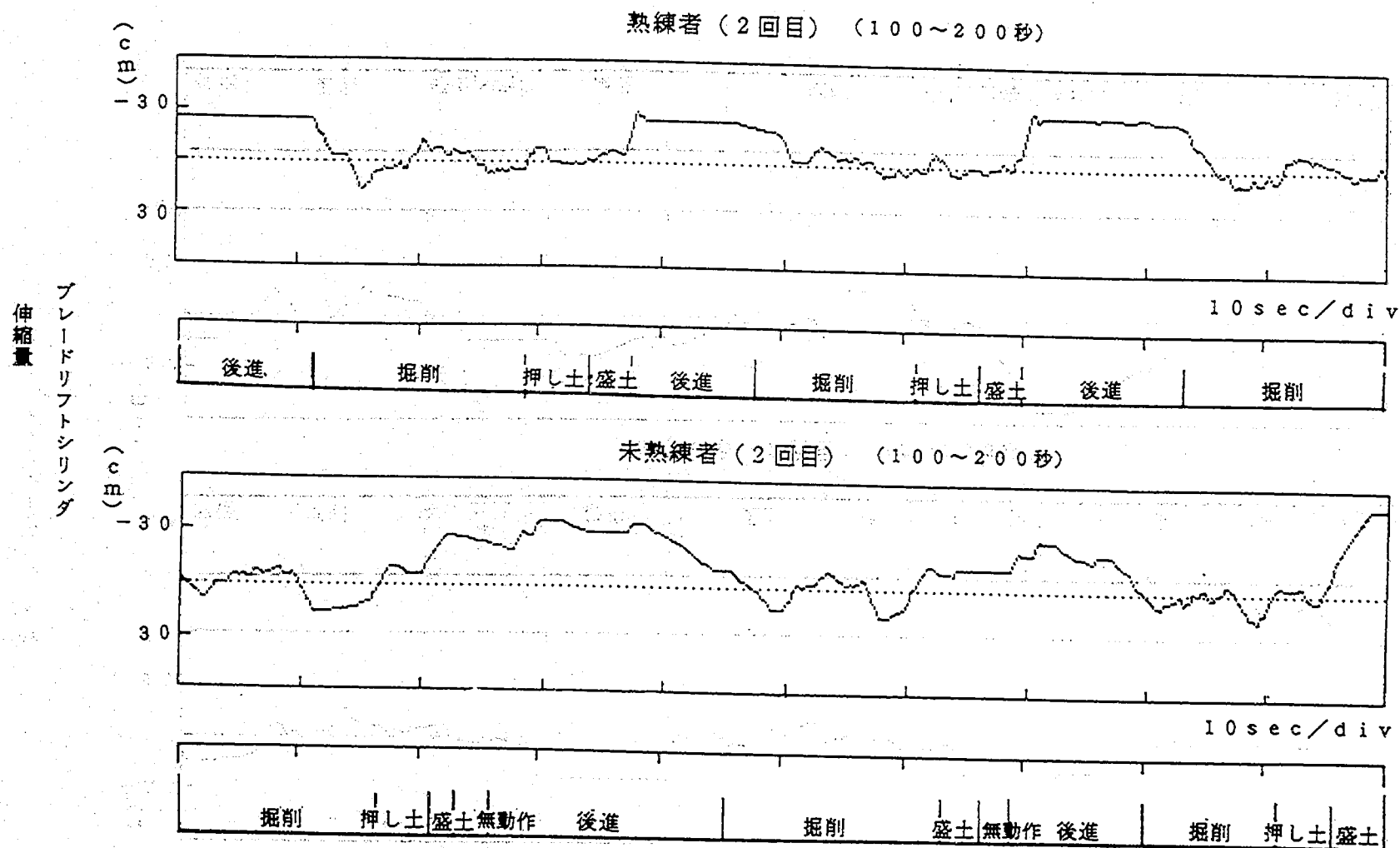


図15 ブレードリフトシリンダ伸縮量

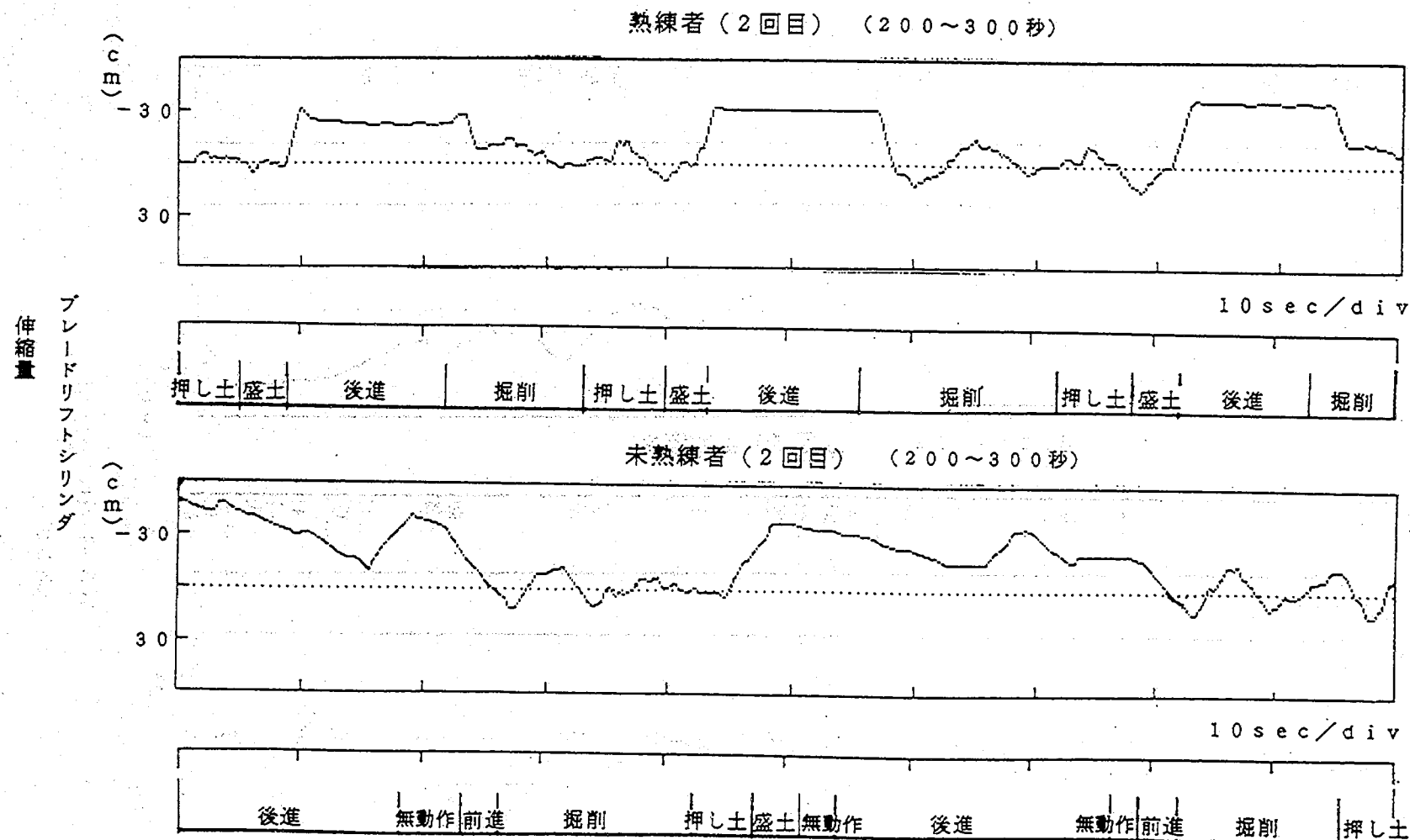


図16 ブレードリフトシリンダ伸縮量

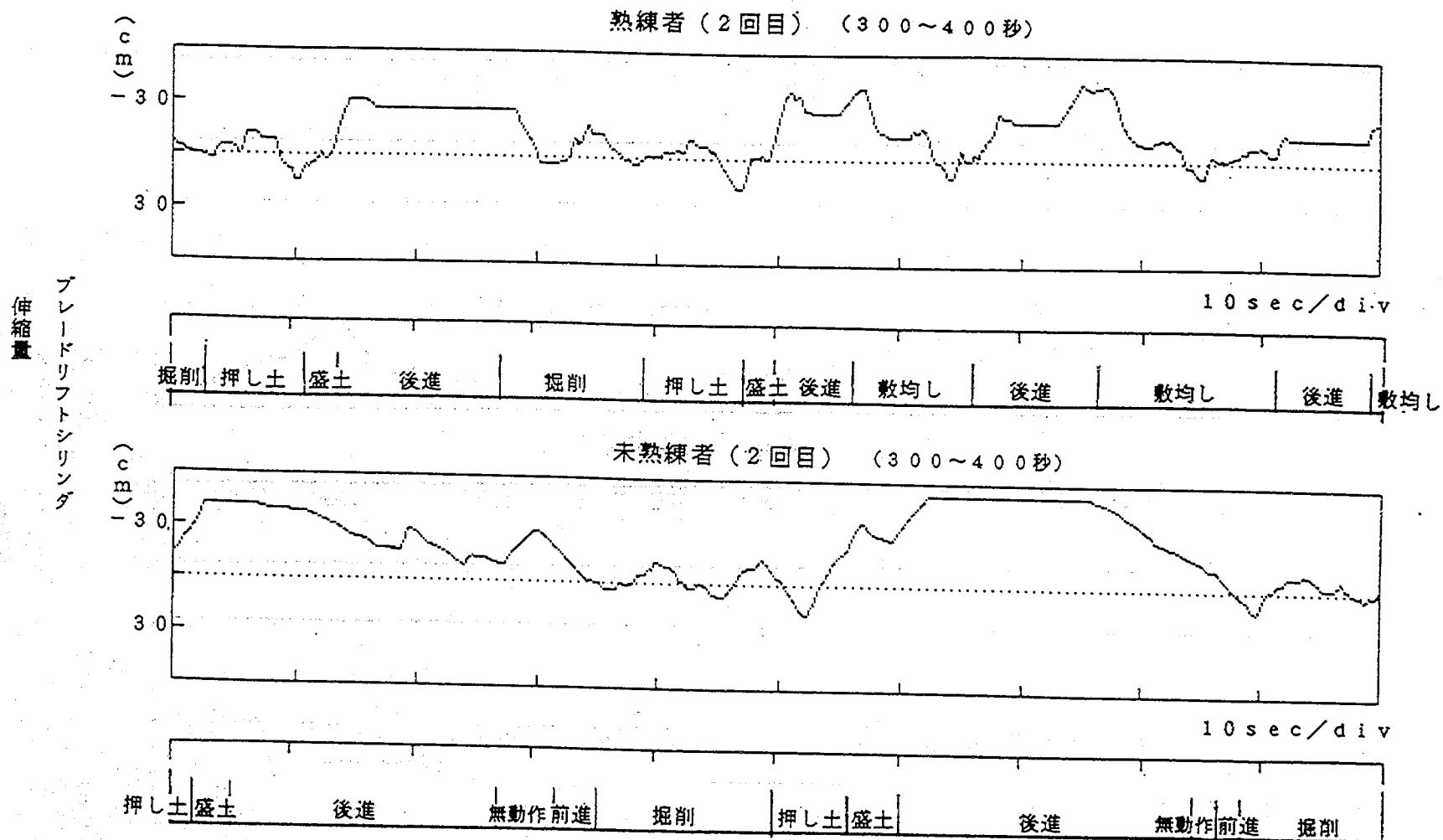


図17 ブレードリフトシリンダ伸縮量

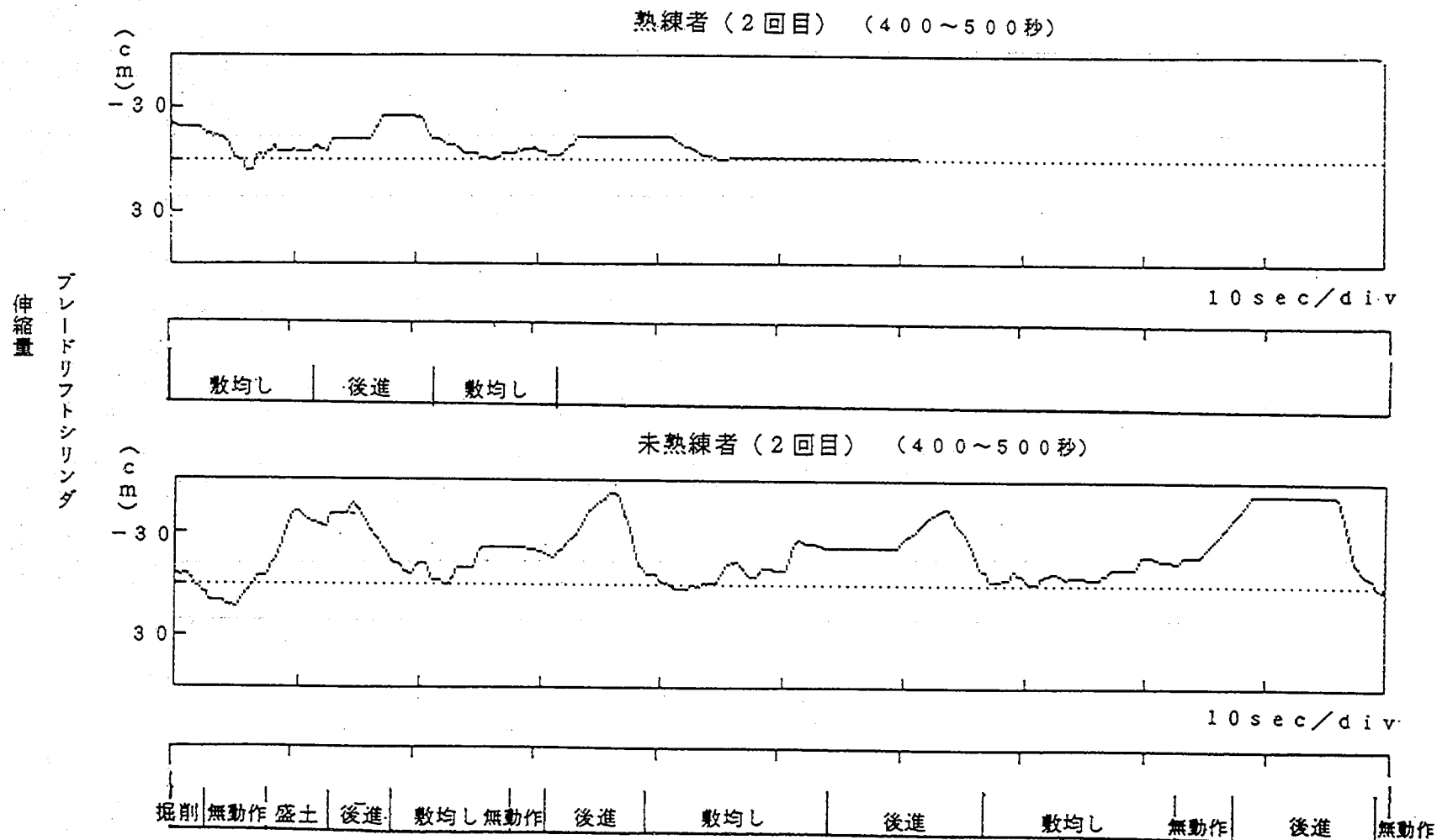
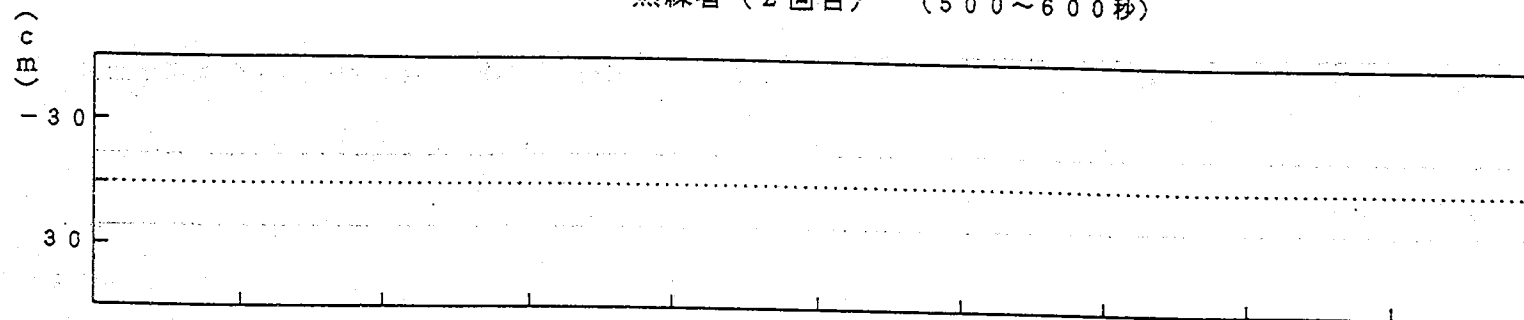
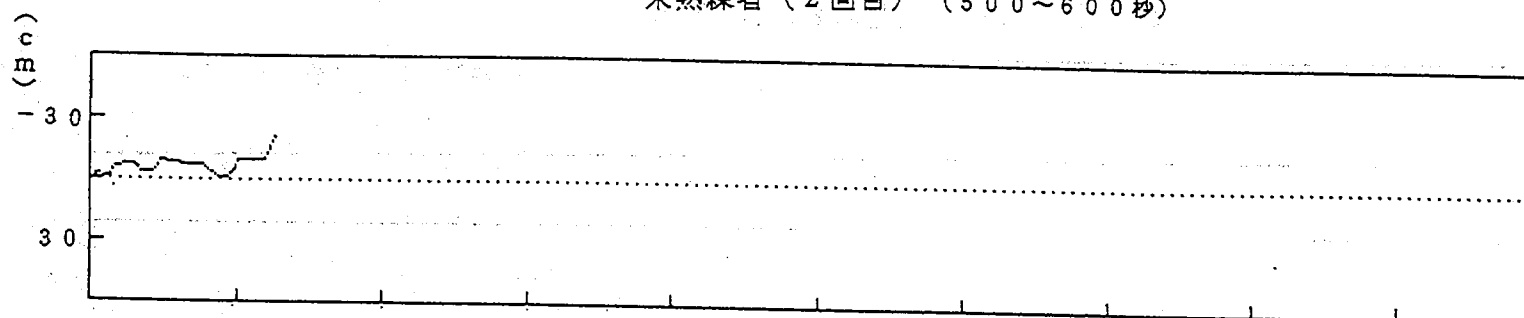


図18 ブレードリフトシリンダ伸縮量

熟練者 (2回目) (500~600秒)



未熟練者 (2回目) (500~600秒)

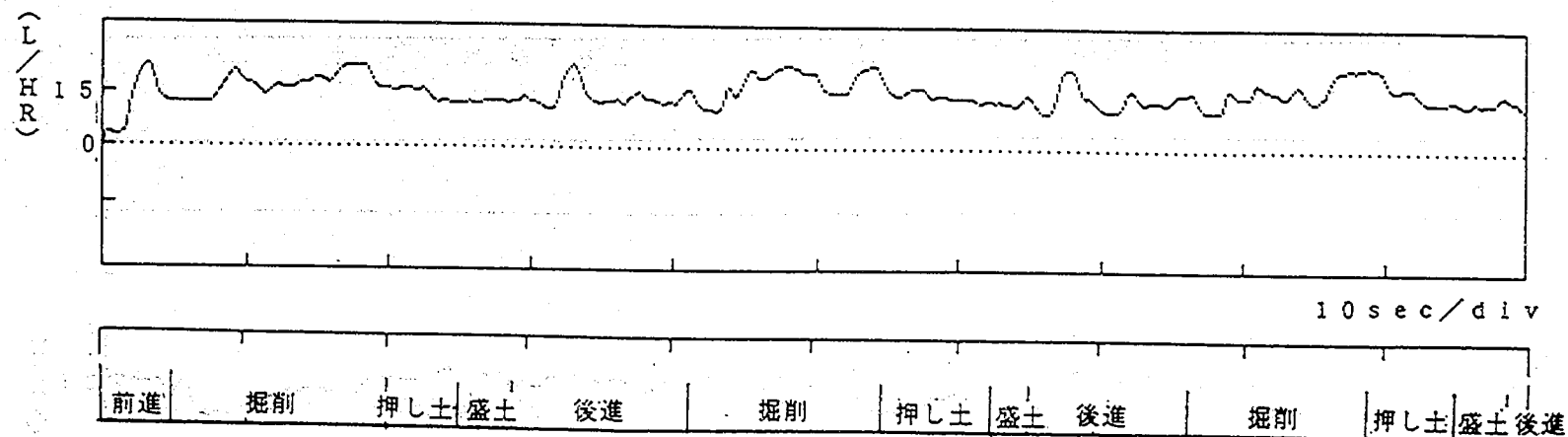


無動作 数均し

図19 ブレードリフトシリンダ伸縮量

燃料消費量

熟練者 (2回目) (0~100秒)



未熟練者 (2回目) (0~100秒)

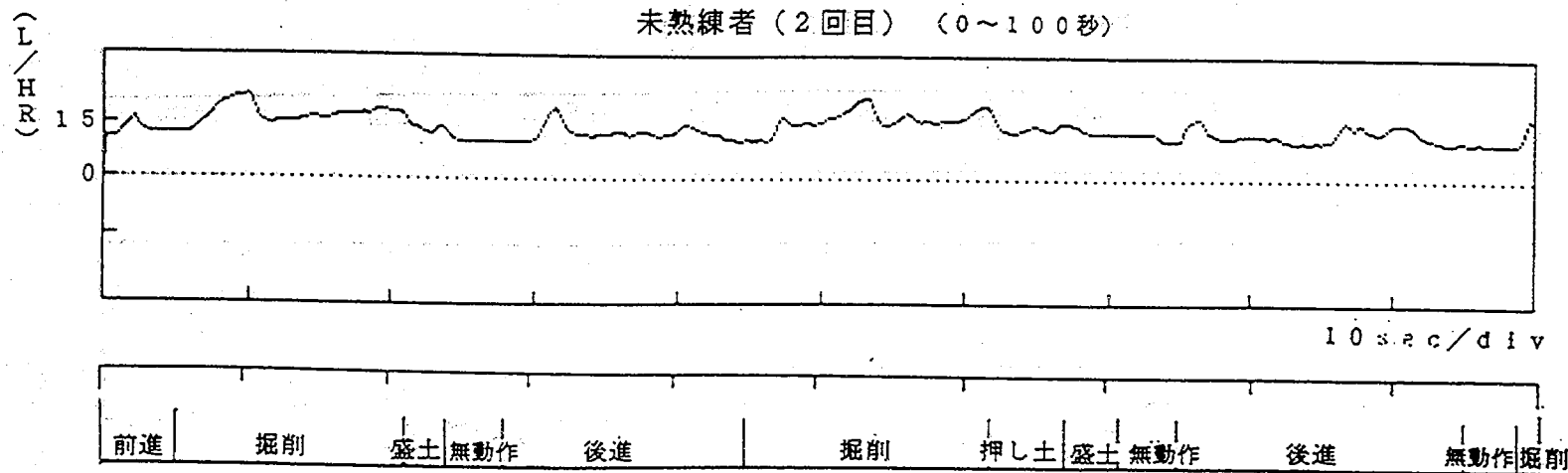
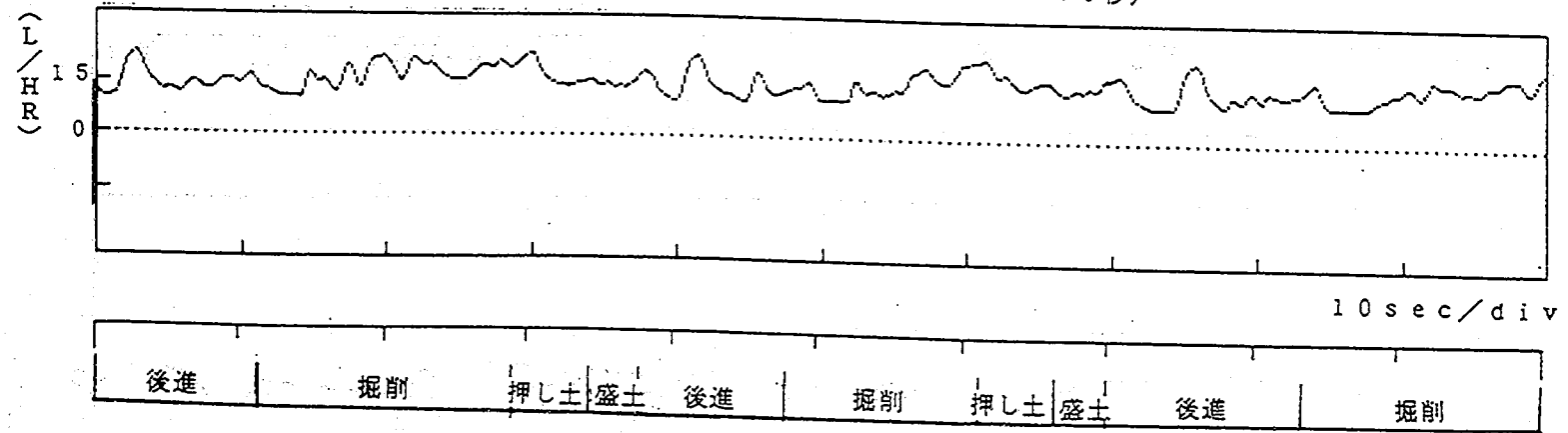


図20 燃料消費量

熟練者 (2回目) (100~200秒)



未熟練者 (2回目) (100~200秒)

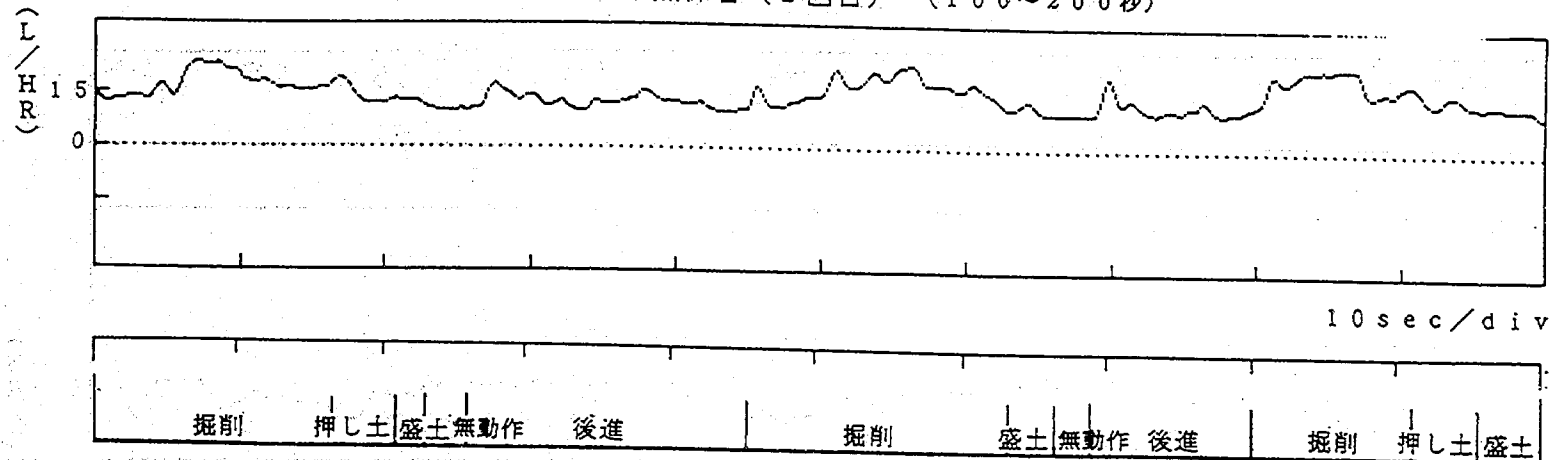
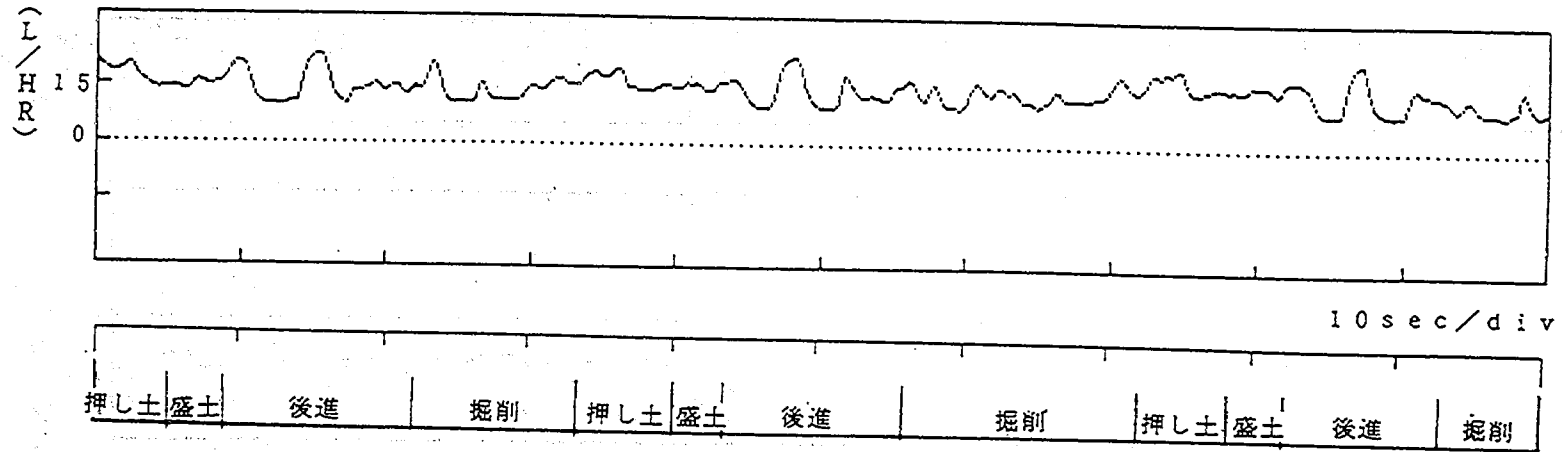


図21 燃料消費量

熟練者 (2回目) (200~300秒)



未熟練者 (2回目) (200~300秒)

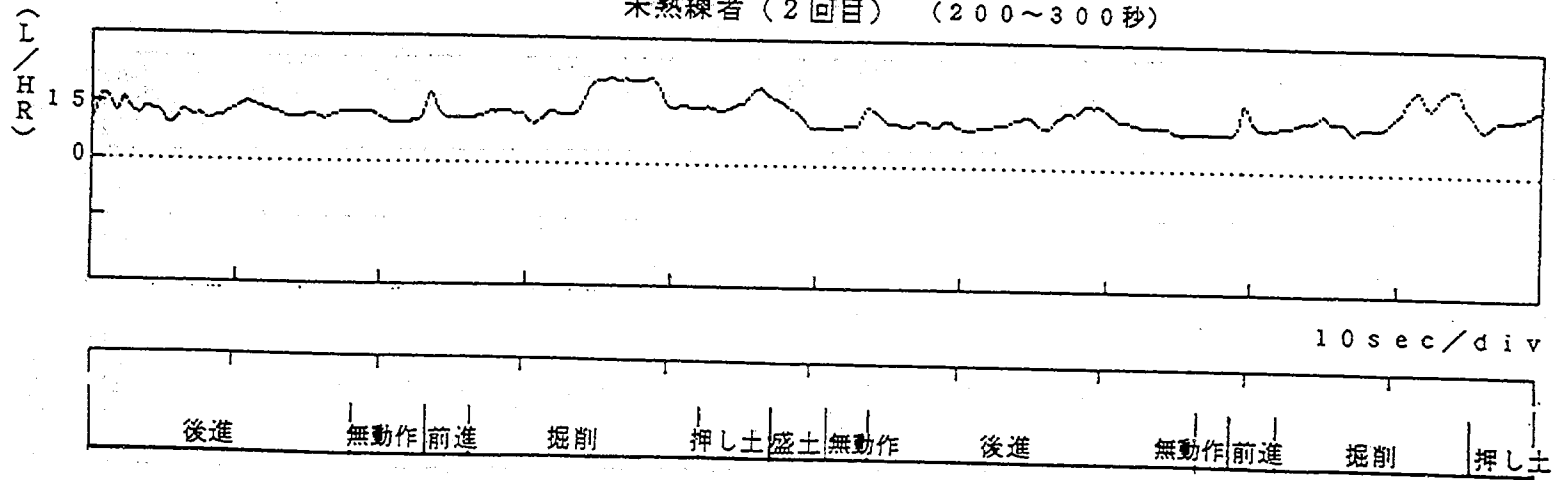


図22 燃料消費量

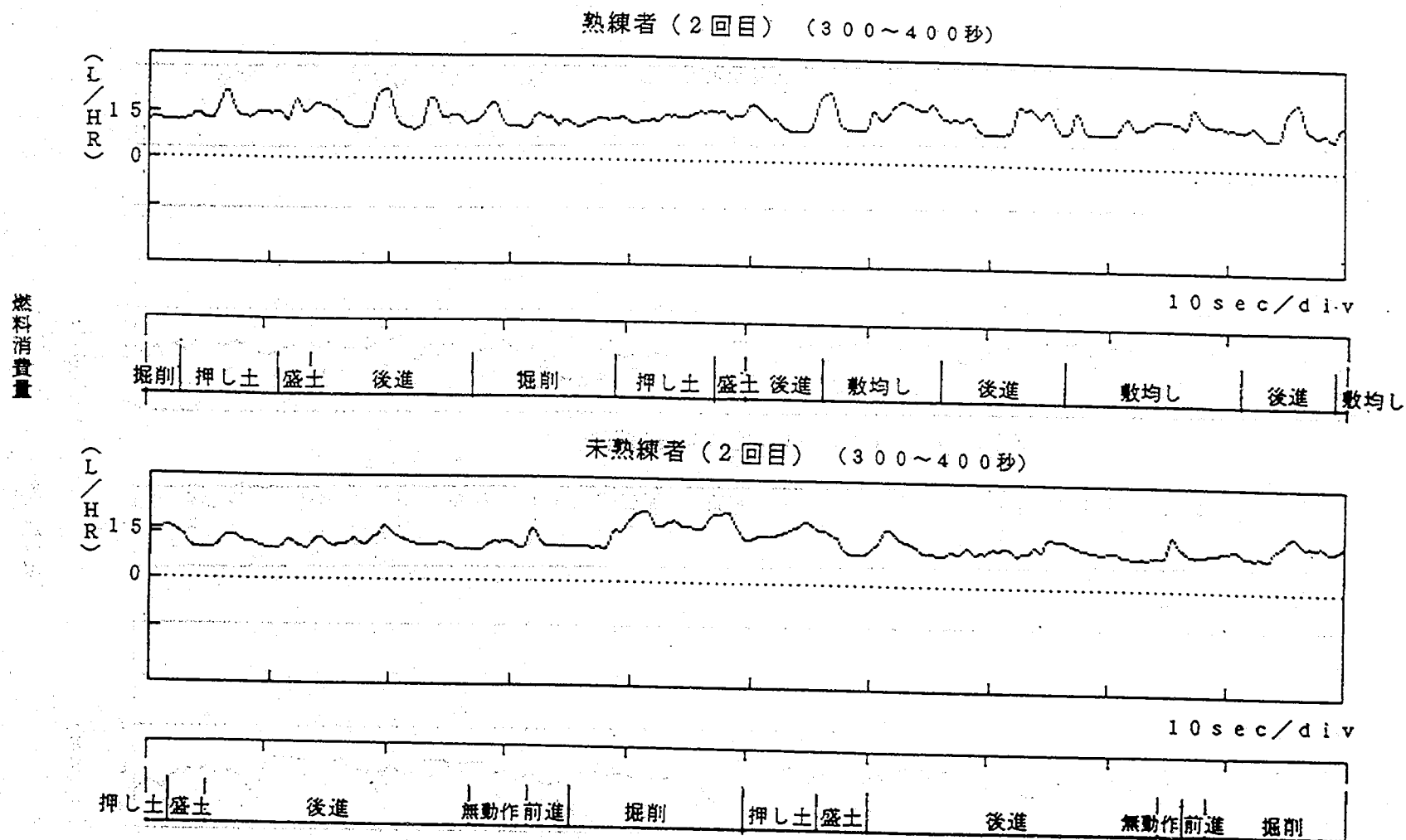


図23 燃料消費量

燃料消費量

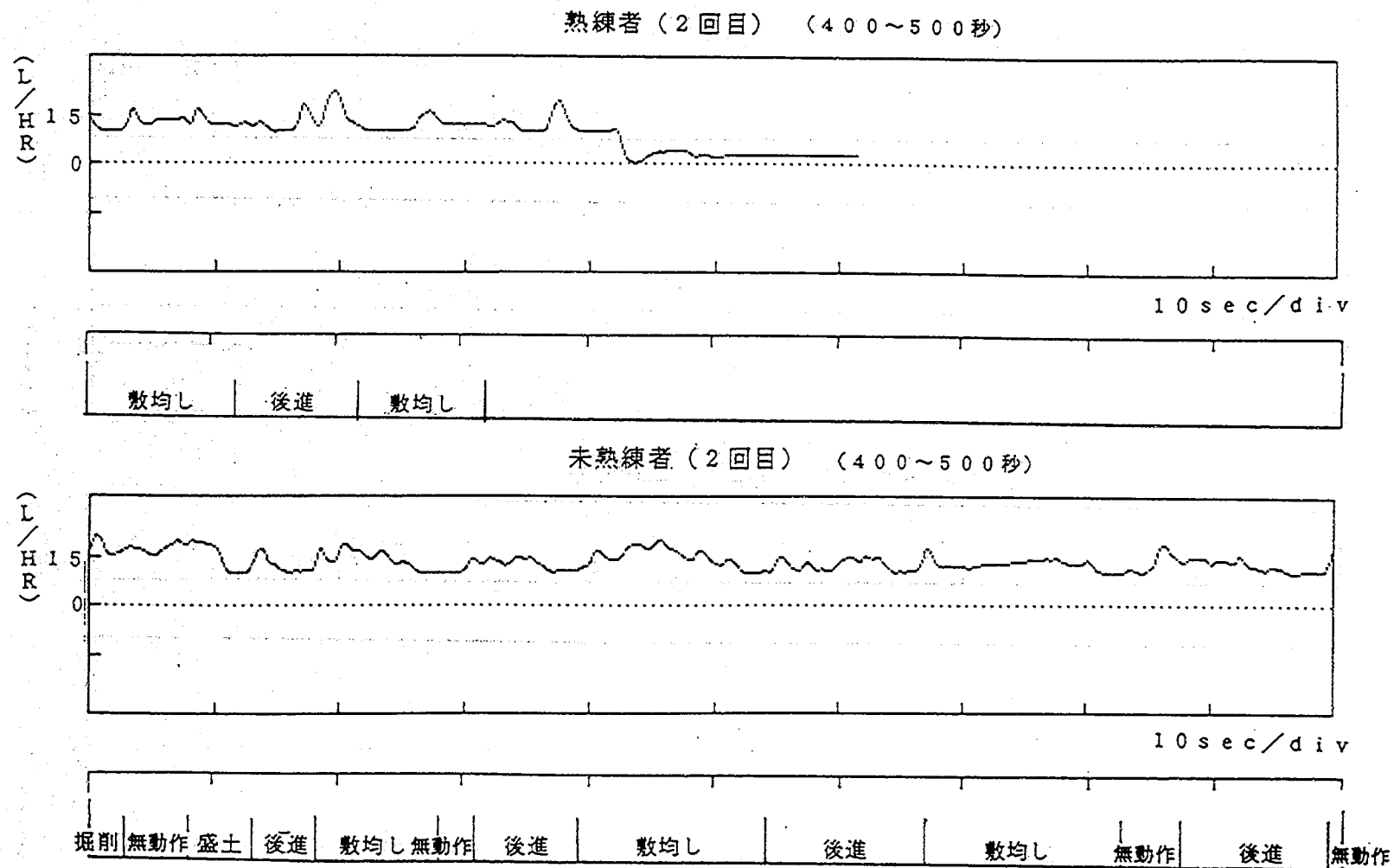
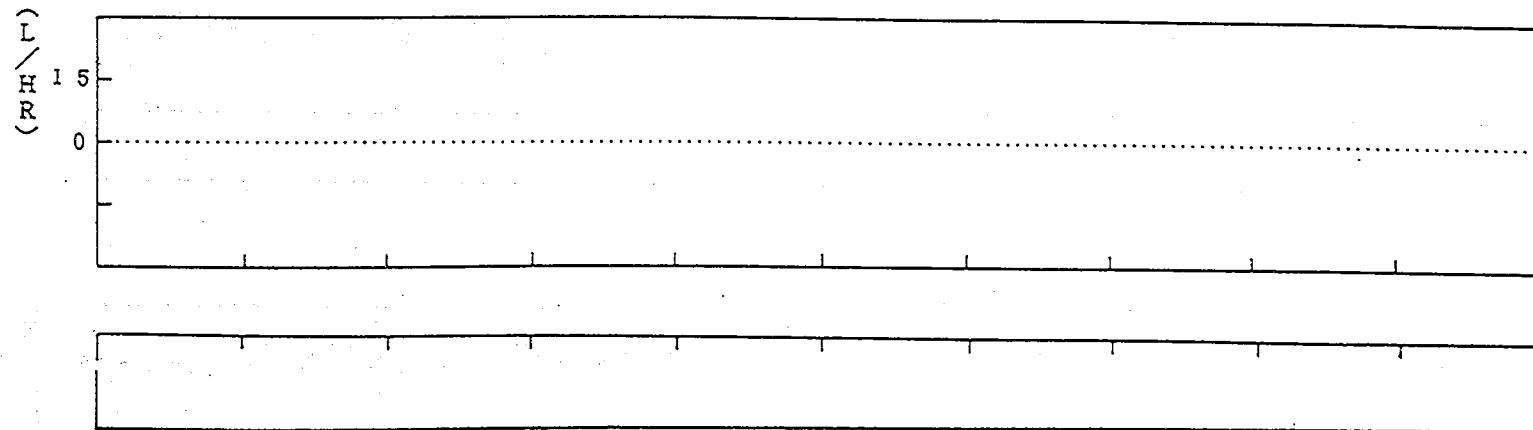


図24 燃料消費量

熟練者 (2 回目) (500 ~ 600 秒)



未熟練者 (2 回目) (500 ~ 600 秒)

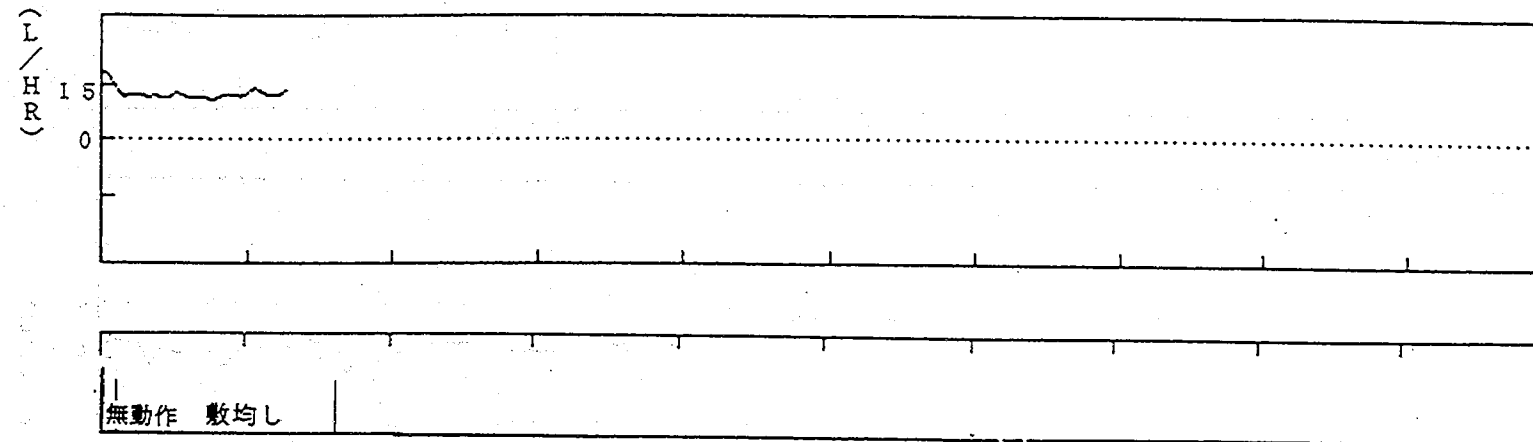


図25 燃料消費量

燃料消費量

車体傾斜角  
ピッチング

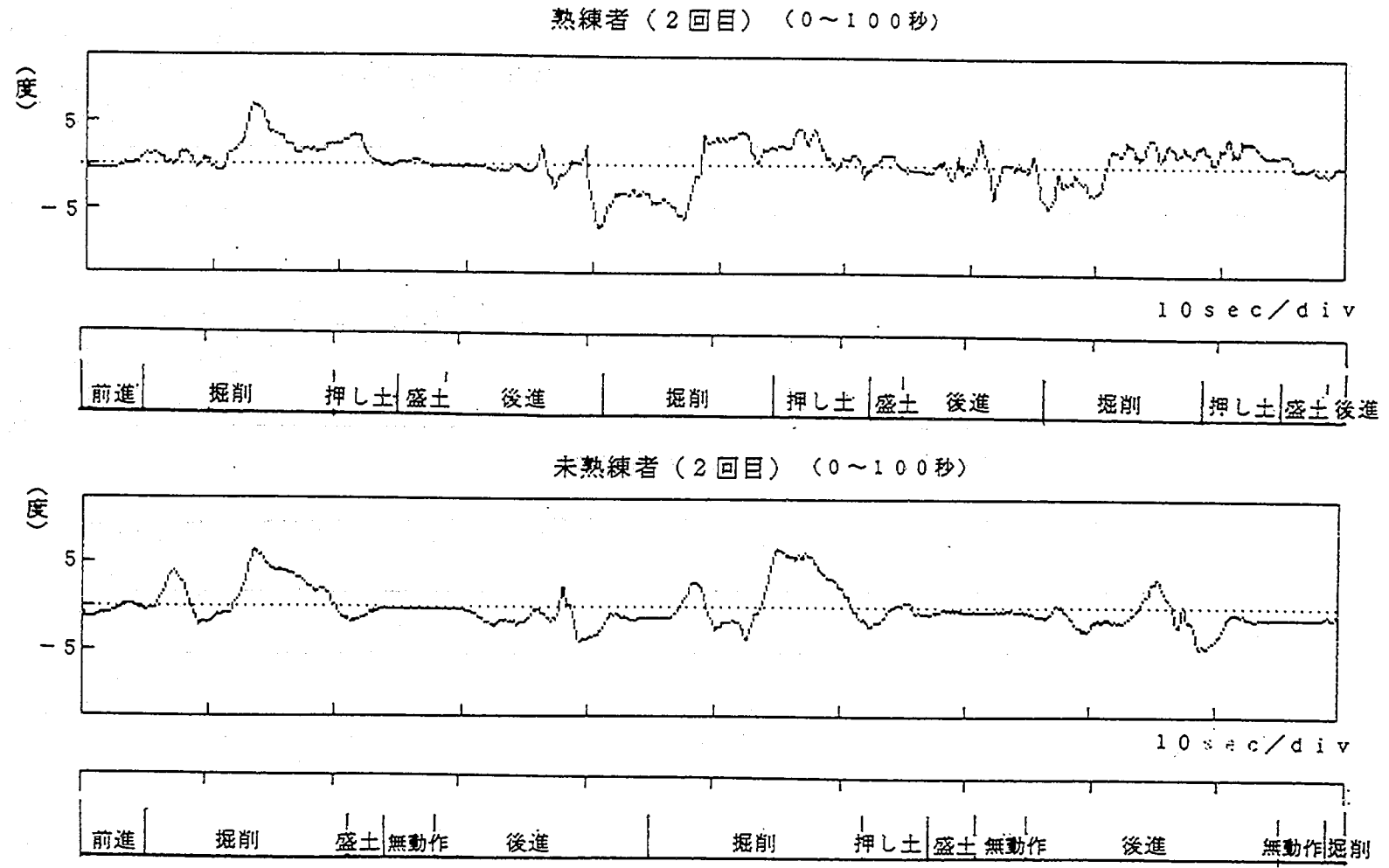
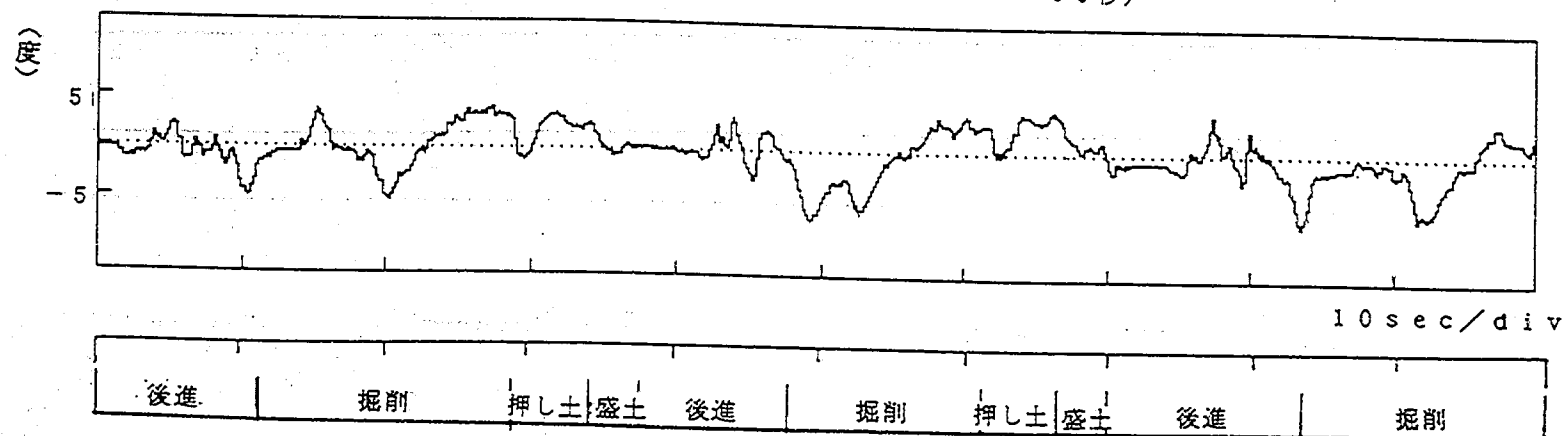


図26 車体傾斜角 (ピッチング)

車体傾斜角  
ピッチング

熟練者 (2回目) (100~200秒)



未熟練者 (2回目) (100~200秒)

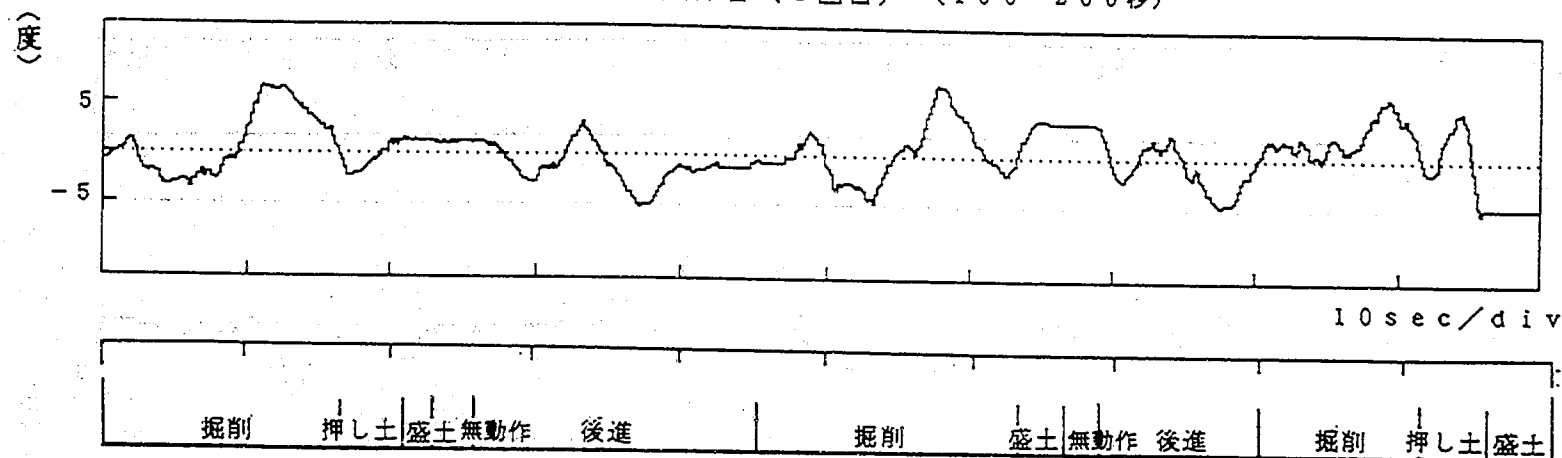
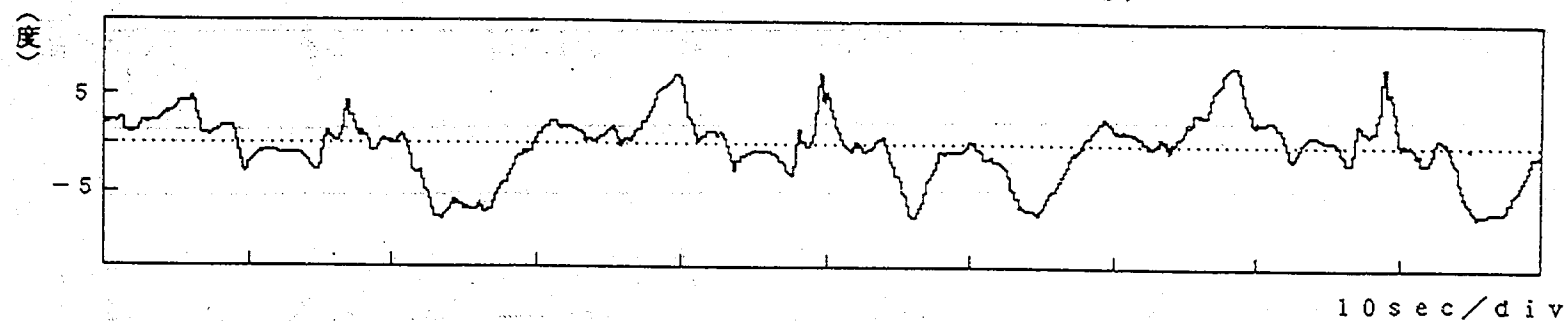


図27 車体傾斜角 (ピッチング)

熟練者 (2回目) (200~300秒)



未熟練者 (2回目) (200~300秒)

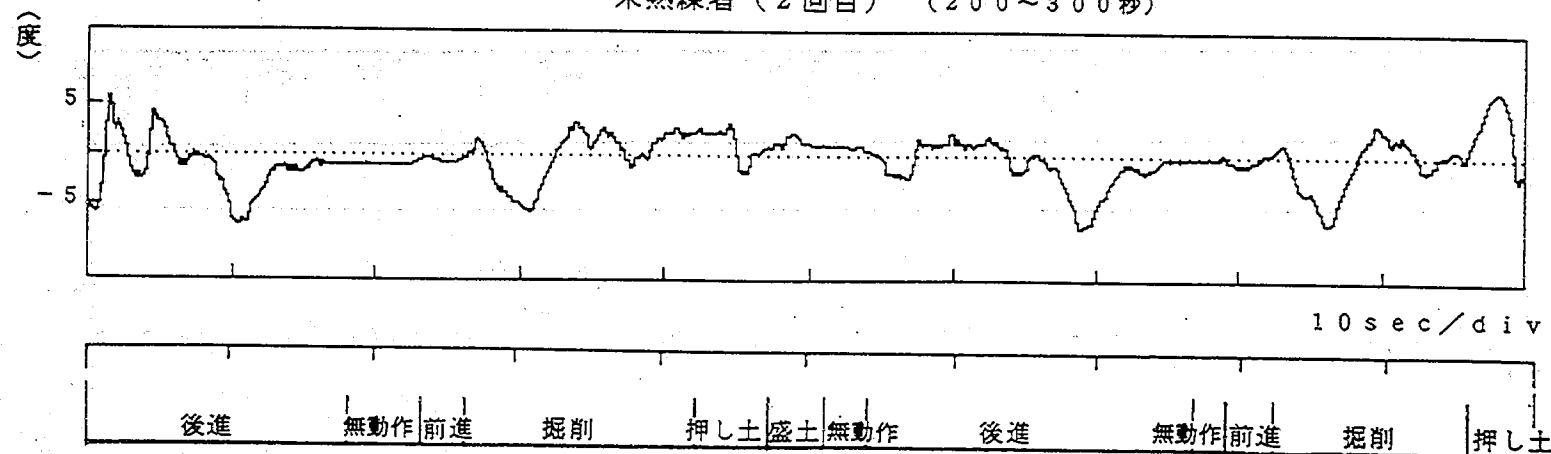
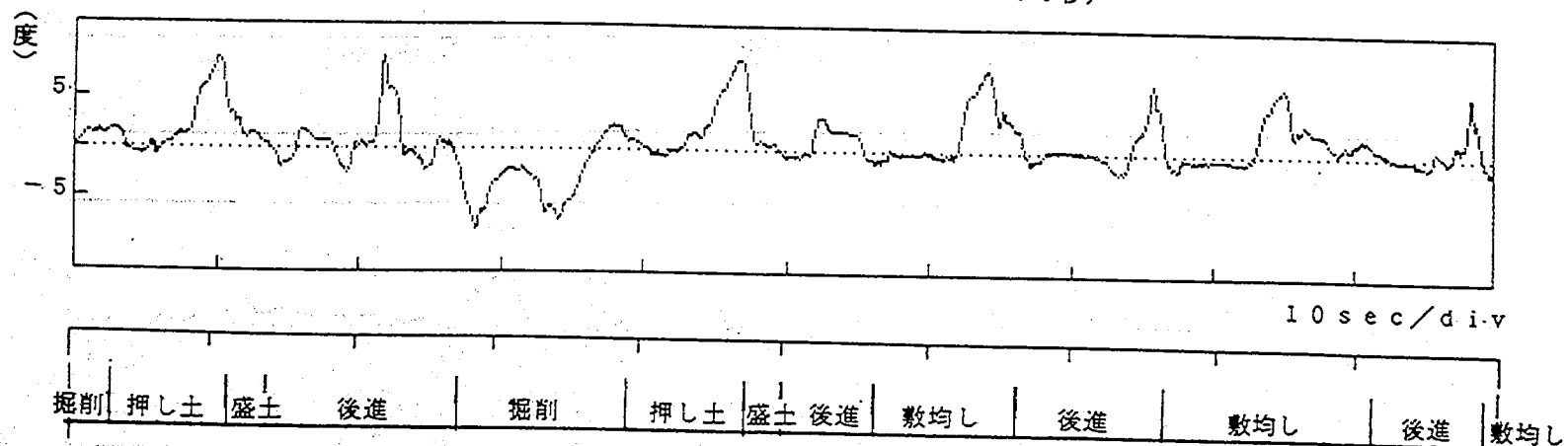


図28 車体傾斜角 (pitching)

車体傾斜角  
ピッチング

熟練者 (2回目) (300~400秒)



未熟練者 (2回目) (300~400秒)

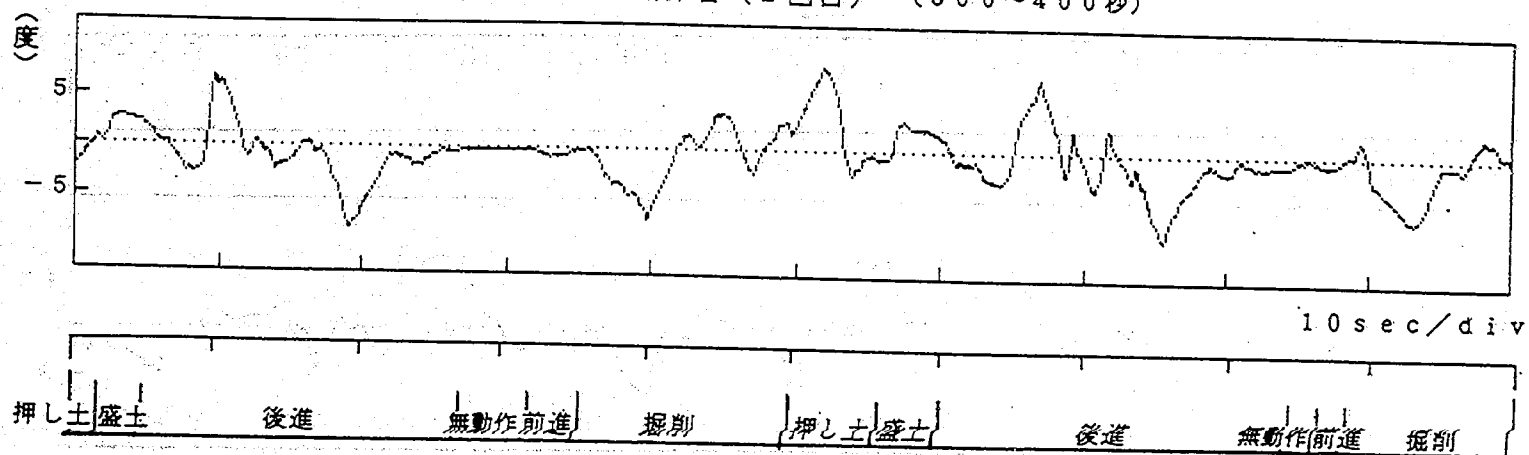


図29 車体傾斜角 (ピッチング)

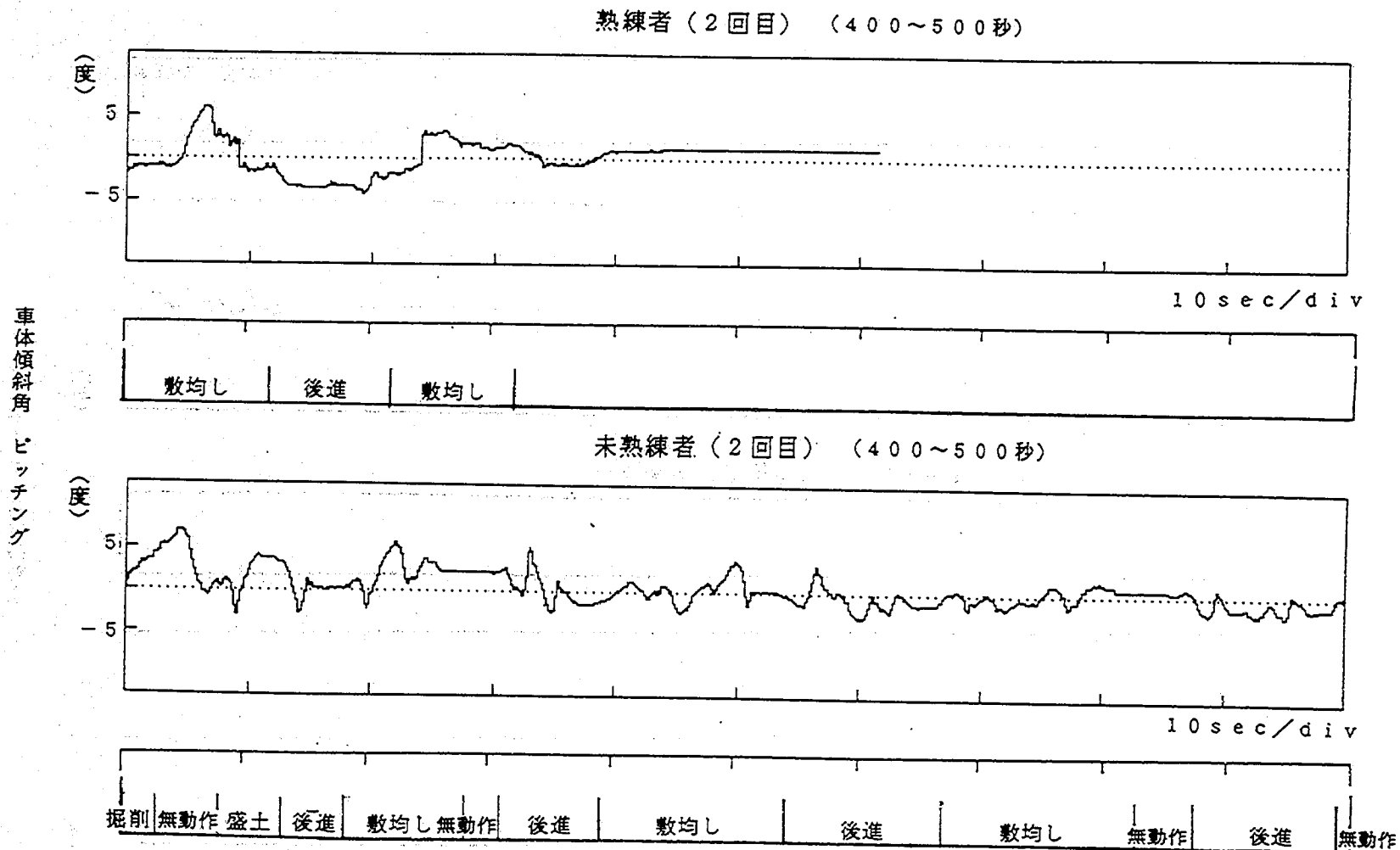


図30 車体傾斜角 (ピッチング)

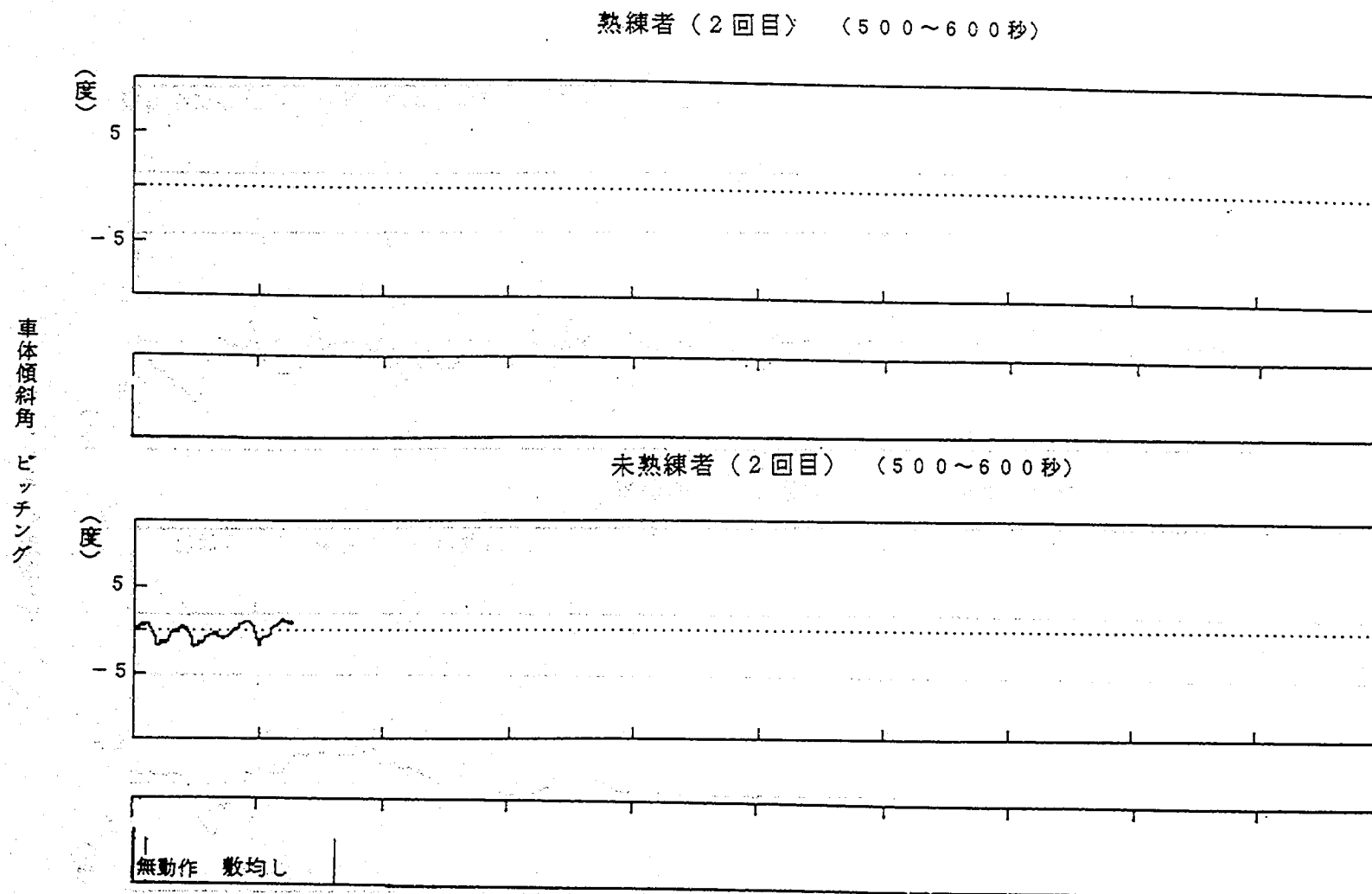


図31 車体傾斜角 (ピッチング)

車体傾斜角  
ローリング

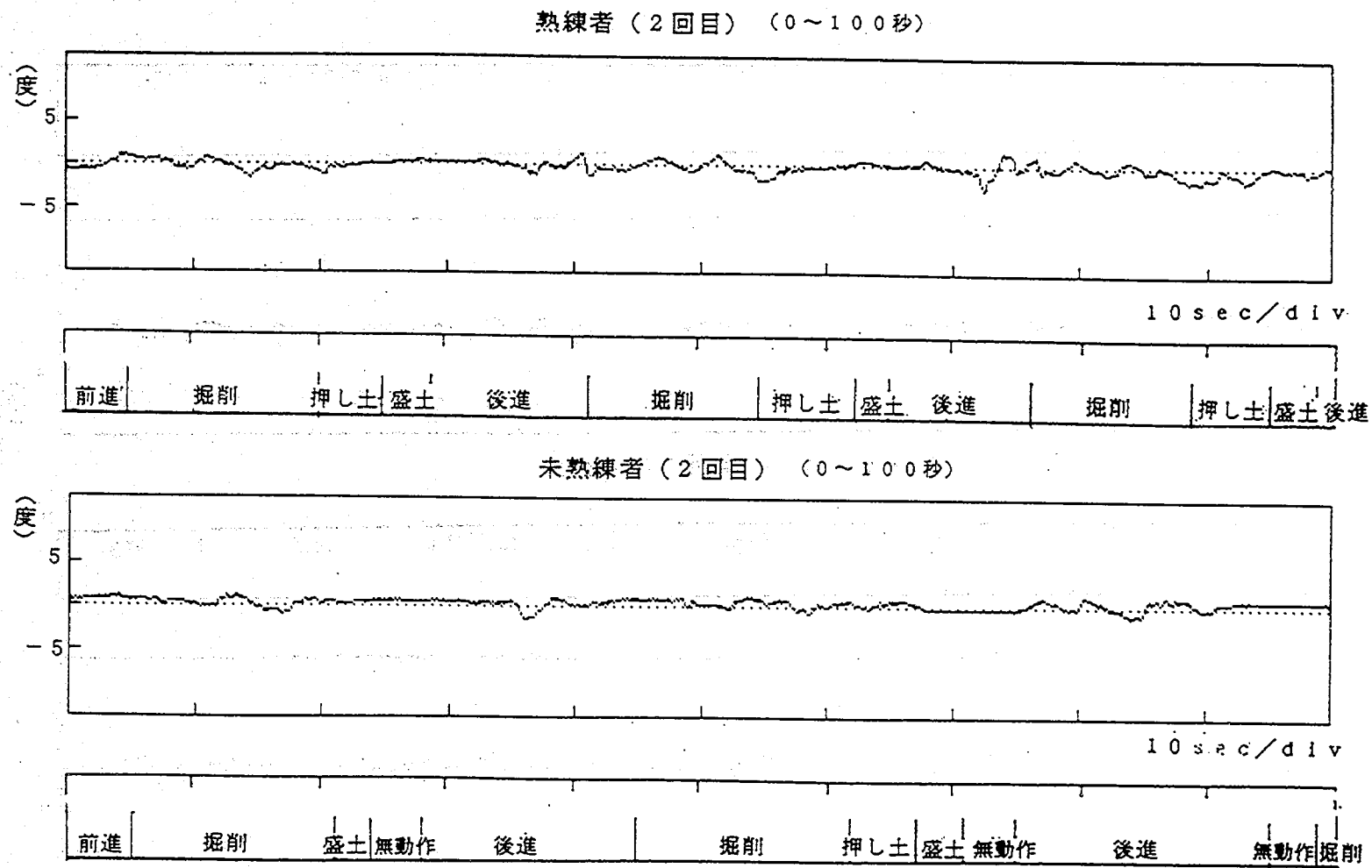
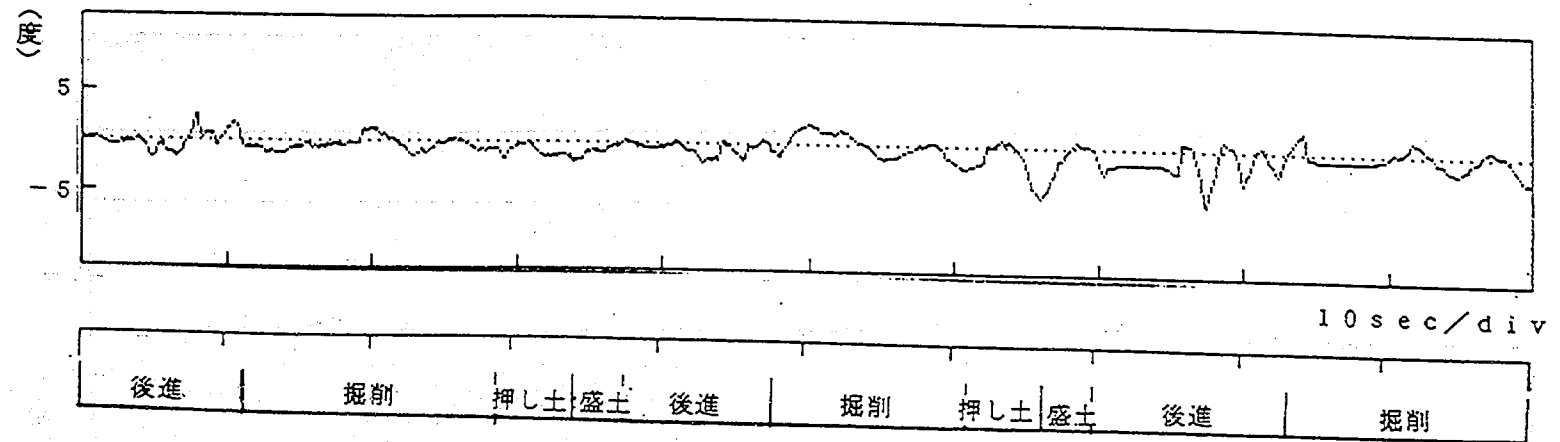


図32 車体傾斜角 (ローリング)

熟練者 (2 回目) (100~200 秒)



未熟練者 (2 回目) (100~200 秒)

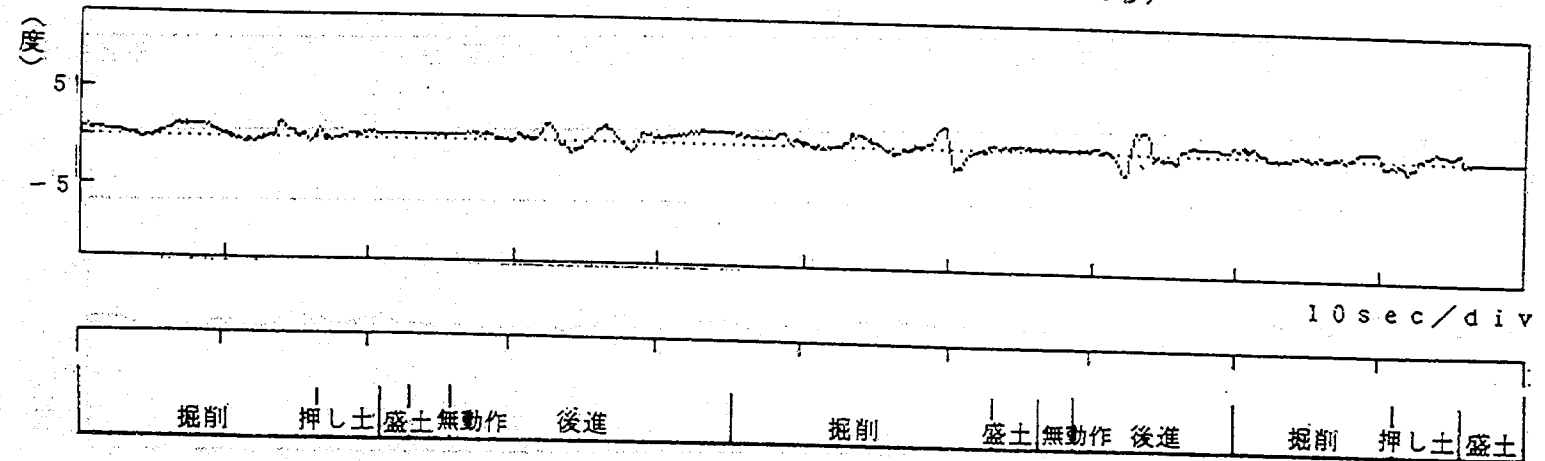
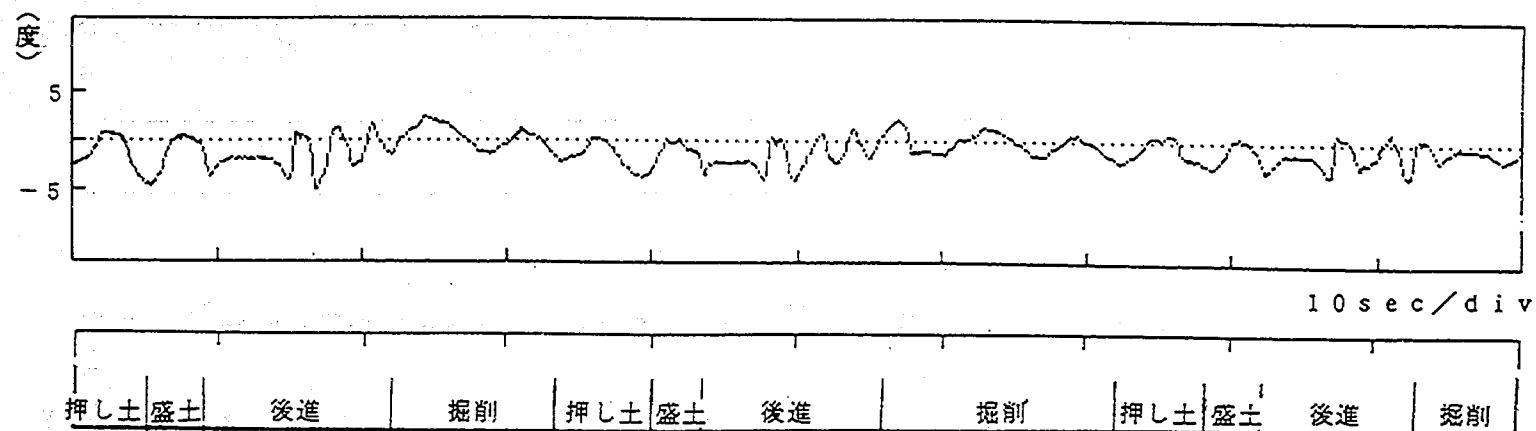


図33 車体傾斜角 (ローリング)

車体傾斜角  
ローリング

熟練者 (2回目) (200~300秒)



未熟練者 (2回目) (200~300秒)

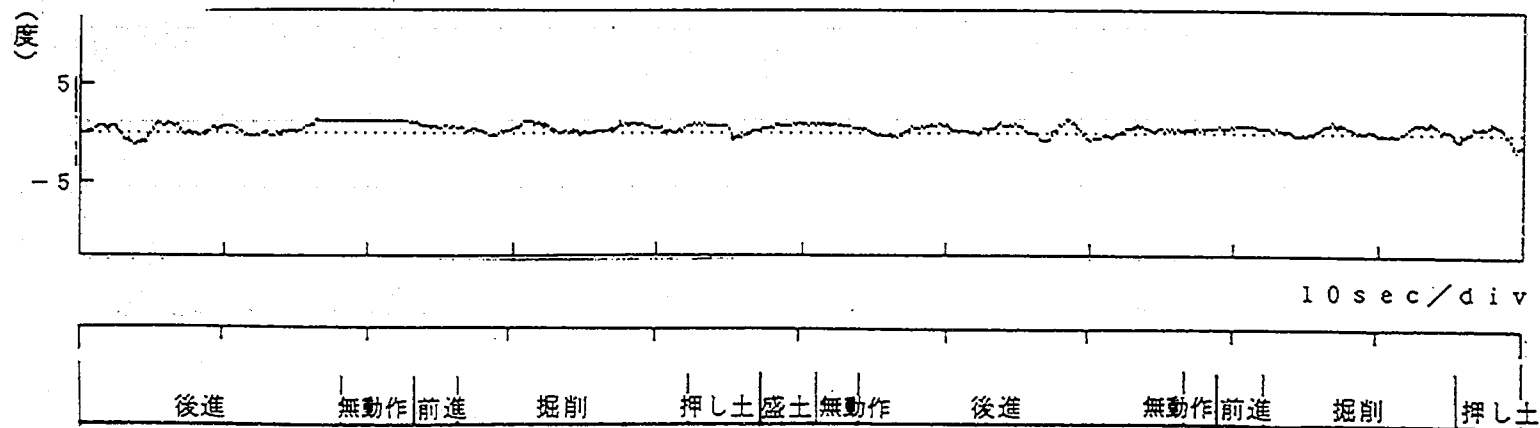
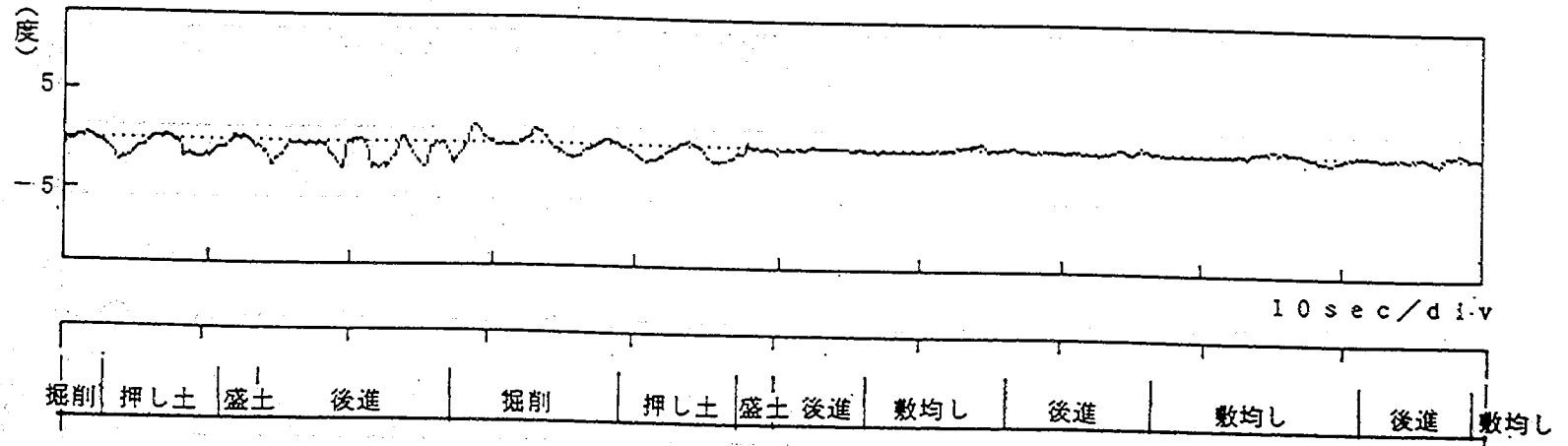


図34 車体傾斜角 (ローリング)

車体傾斜角  
ローリング

熟練者 (2 回目) (300~400 秒)



未熟練者 (2 回目) (300~400 秒)

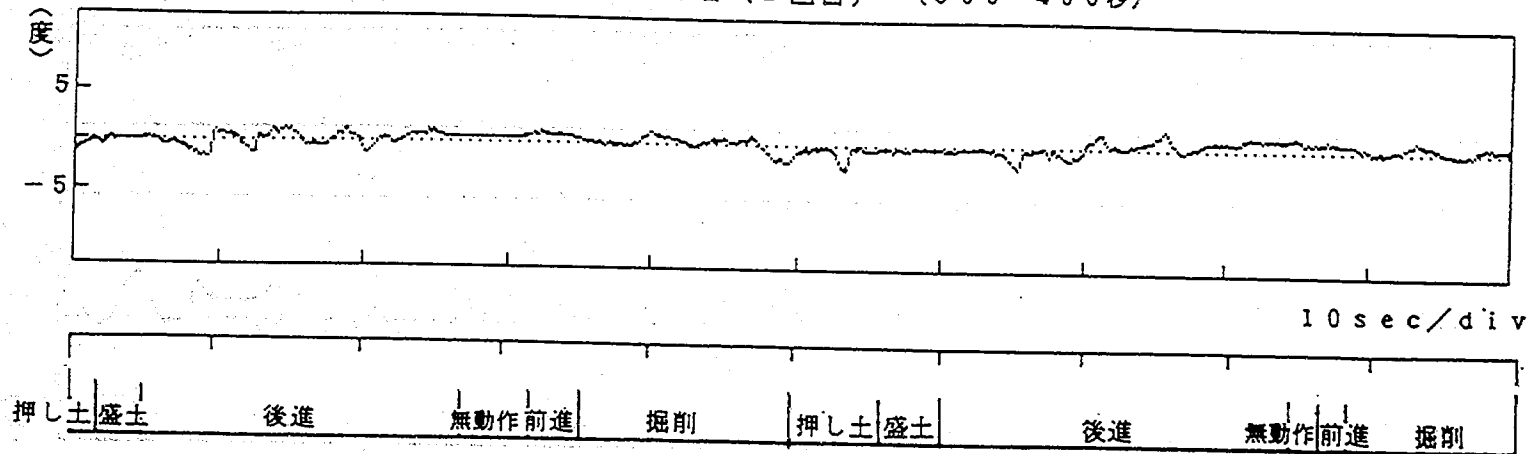


図35 車体傾斜角 (ローリング)

車体傾斜角  
ローリング

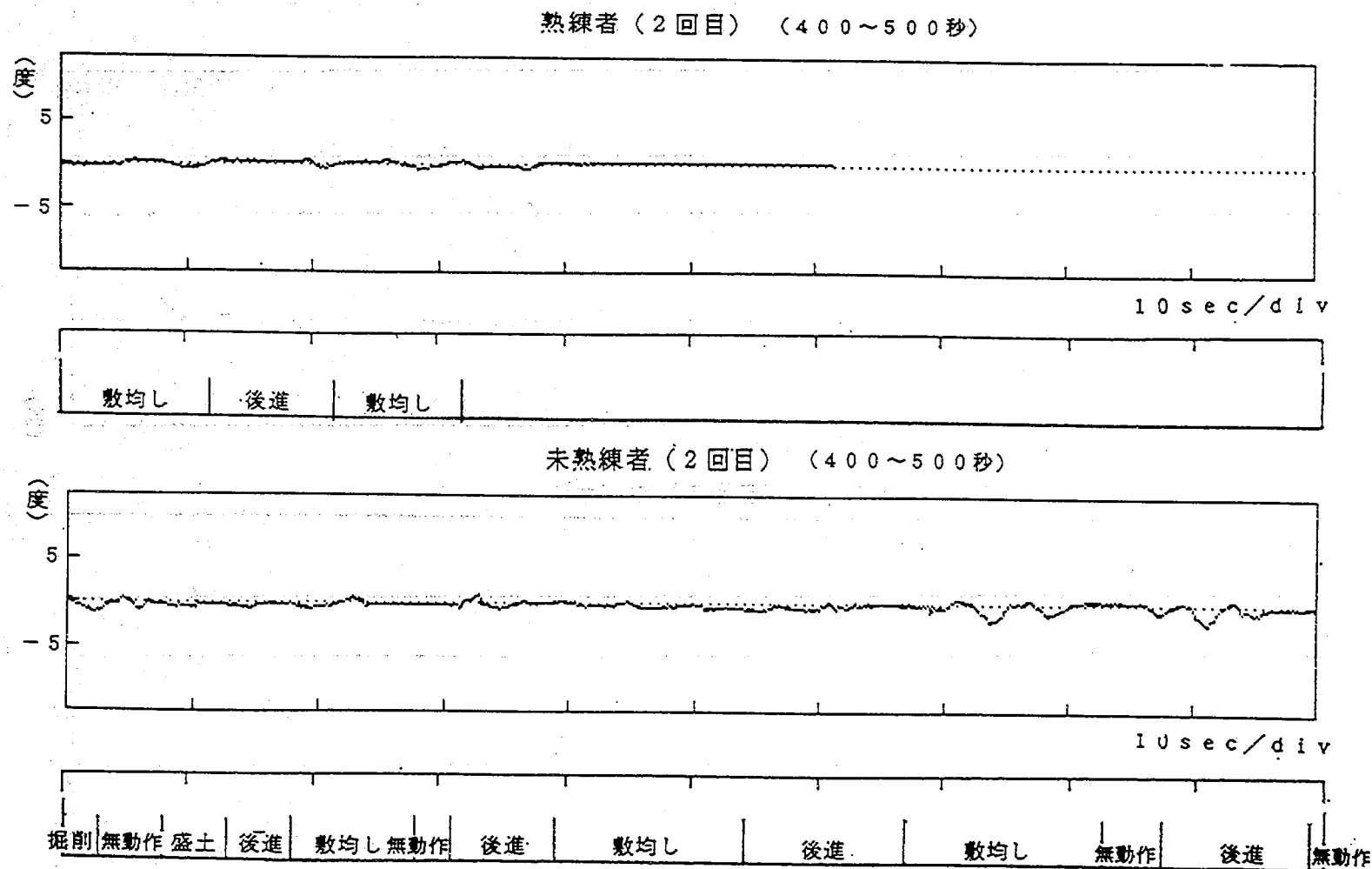
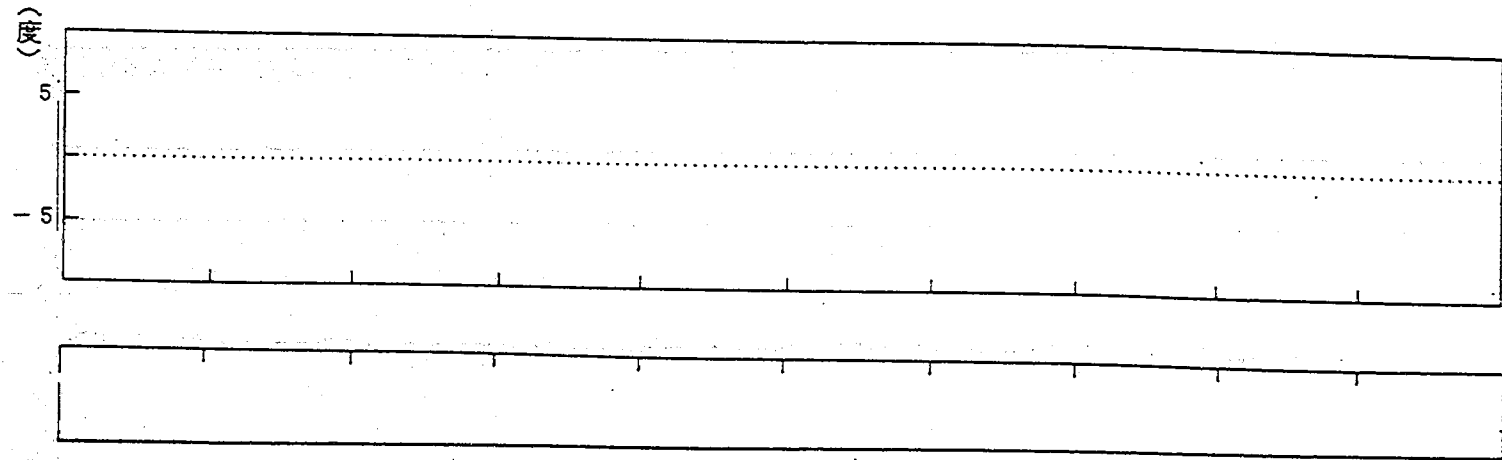
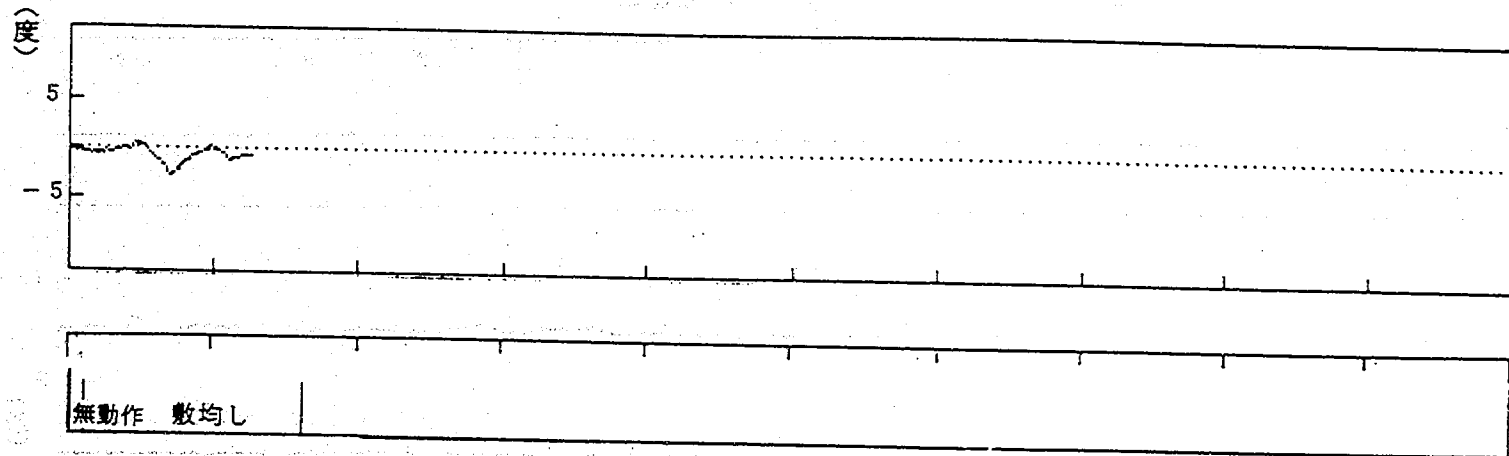


図36 車体傾斜角 (ローリング)

熟練者 (2 回目) (500 ~ 600 秒)



未熟練者 (2 回目) (500 ~ 600 秒)



無動作 敷均し

図37 車体傾斜角 (ローリング)

## 5) 土質試験結果

実験フィールドの土質については、その状態によって作業の能率等が大きく異なることから含水比、比重、密度、粒度分布を調べるため試験を実施した。実施結果を表8～表10に示す

表8 ポータブルコーンペネトロメータ貫入試験結果（実験前）

（5～19m）

| コーン<br>No. | 測定<br>深度 | 測定位置（ポールよりの距離） |             |                  |        |             |                  |        |             |                  |
|------------|----------|----------------|-------------|------------------|--------|-------------|------------------|--------|-------------|------------------|
|            |          | 5 m            |             |                  | 10 m   |             |                  | 15 m   |             |                  |
|            |          | 計測値            | 荷重<br>(kgf) | コーン抵抗<br>(kgf/d) | 計測値    | 荷重<br>(kgf) | コーン抵抗<br>(kgf/d) | 計測値    | 荷重<br>(kgf) | コーン抵抗<br>(kgf/d) |
| 1          | 地表上      | 90             | 40.7        | 12.6             | 100    | 45.1        | 14.0             | 90     | 40.7        | 12.6             |
|            | 5cm      | 140            | 62.9        | 19.5             | *5.0cm |             |                  | *2.0cm |             |                  |
|            | 10cm     | 130            | 58.5        | 18.1             |        |             |                  |        |             |                  |
|            | 20cm     | *18.5cm        |             |                  |        |             |                  |        |             |                  |
| 2          | 地表上      | 70             | 31.8        | 9.8              | 100    | 45.1        | 14.0             | 130    | 58.5        | 18.1             |
|            | 5cm      | 90             | 40.7        | 12.6             | *1.0cm |             |                  | *0cm   |             |                  |
|            | 10cm     | 120            | 54.0        | 16.7             |        |             |                  |        |             |                  |
|            | 20cm     | *11.5cm        |             |                  |        |             |                  |        |             |                  |
| 3          | 地表上      | 130            | 58.5        | 18.1             | 110    | 49.6        | 15.4             | 90     | 40.7        | 12.6             |
|            | 5cm      | *0.5cm         |             |                  | *1.0cm |             |                  | *1.5cm |             |                  |
|            | 地表上      | 110            | 49.6        | 15.4             | 80     | 36.2        | 11.2             | 130    | 58.5        | 18.1             |
|            | 5cm      | 90             | 40.7        | 12.6             | 150    | 67.4        | 20.9             | *1.0cm |             |                  |
| 4          | 10cm     | 120            | 54.0        | 16.7             | *7.0cm |             |                  |        |             |                  |
|            | 20cm     | 140            | 62.9        | 19.5             |        |             |                  |        |             |                  |
|            | 30cm     | 100            | 45.1        | 14.0             |        |             |                  |        |             |                  |
|            | 40cm     | 100            | 45.1        | 14.0             |        |             |                  |        |             |                  |
|            | 50cm     | 70             | 31.8        | 9.8              |        |             |                  |        |             |                  |

（20～30m）

| コーン<br>No. | 測定<br>深度 | 測定位置（ポールよりの距離） |             |                  |        |             |                  |        |             |                  |
|------------|----------|----------------|-------------|------------------|--------|-------------|------------------|--------|-------------|------------------|
|            |          | 20 m           |             |                  | 25 m   |             |                  | 30 m   |             |                  |
|            |          | 計測値            | 荷重<br>(kgf) | コーン抵抗<br>(kgf/d) | 計測値    | 荷重<br>(kgf) | コーン抵抗<br>(kgf/d) | 計測値    | 荷重<br>(kgf) | コーン抵抗<br>(kgf/d) |
| 1          | 地表上      | 130            | 58.5        | 18.1             | 130    | 58.5        | 18.1             | 120    | 54.0        | 16.7             |
|            | 5cm      | *1.0cm         |             |                  | *0cm   |             |                  | *1.5cm |             |                  |
|            | 10cm     |                |             |                  |        |             |                  |        |             |                  |
|            | 20cm     |                |             |                  |        |             |                  |        |             |                  |
| 2          | 地表上      |                |             |                  |        |             |                  |        |             |                  |
|            | 5cm      | 120            | 54.0        | 16.7             | 130    | 58.5        | 18.1             | 120    | 54.0        | 16.7             |
|            | 10cm     | *0.5cm         |             |                  | *0.5cm |             |                  | *1.0cm |             |                  |
|            | 20cm     |                |             |                  |        |             |                  |        |             |                  |
| 3          | 地表上      | 140            | 62.9        | 19.5             | 100    | 45.1        | 14.0             | 110    | 49.6        | 15.4             |
|            | 5cm      | *0cm           |             |                  | *1.5cm |             |                  | *1.0cm |             |                  |
|            | 10cm     | 140            | 62.9        | 19.5             | 170    | 76.3        | 23.6             | 120    | 54.0        | 16.7             |
|            | 20cm     | *1.0cm         |             |                  | *1.0cm |             |                  | *0.5cm |             |                  |

\*印は貫入不可の深さ

表9 ポータブルコーンペネトロメータ貫入試験結果 (実験後)

| コーン<br>No. | 測定<br>深度 | 測定位置 (ボールよりの距離) |             |                  |        |             |                  |         |             |                  |
|------------|----------|-----------------|-------------|------------------|--------|-------------|------------------|---------|-------------|------------------|
|            |          | 5 m             |             |                  | 10 m   |             |                  | 20 m    |             |                  |
|            |          | 計測値             | 荷重<br>(kgf) | コーン支効<br>(kgf/d) | 計測値    | 荷重<br>(kgf) | コーン支効<br>(kgf/d) | 計測値     | 荷重<br>(kgf) | コーン支効<br>(kgf/d) |
| 1          | 地表上      | 115             | 51.8        | 16.0             | 70     | 31.8        | 9.8              | 80      | 36.2        | 11.2             |
|            | 5cm      | *0.5cm          |             |                  | 105    | 47.4        | 14.7             | 120     | 54.0        | 16.7             |
|            | 10cm     |                 |             |                  | 90     | 40.7        | 12.6             | 130     | 58.5        | 18.1             |
|            | 20cm     |                 |             |                  | 80     | 36.2        | 11.2             | 130     | 58.5        | 18.1             |
|            | 30cm     |                 |             |                  | 60     | 27.4        | 8.5              | *24.0cm |             |                  |
|            | 40cm     |                 |             |                  | 110    | 49.6        | 15.4             |         |             |                  |
|            | 50cm     |                 |             |                  | 70     | 31.8        | 9.8              |         |             |                  |
| 2          | 地表上      | 60              | 27.4        | 8.5              | 55     | 25.1        | 7.8              | 35      | 16.2        | 5.0              |
|            | 5cm      | 80              | 36.2        | 11.2             | 110    | 49.6        | 15.4             | 75      | 34.0        | 10.5             |
|            | 10cm     | 90              | 40.7        | 12.6             | *9.0cm |             |                  | 70      | 31.8        | 9.8              |
|            | 20cm     | 120             | 54.0        | 16.7             |        |             |                  | 85      | 38.5        | 11.9             |
|            | 30cm     | *21.0cm         |             |                  |        |             |                  | 110     | 49.6        | 15.4             |
|            | 40cm     |                 |             |                  |        |             |                  | 110     | 49.6        | 15.4             |
|            | 50cm     |                 |             |                  |        |             |                  | 120     | 54.0        | 16.7             |
| 3          | 地表上      | 45              | 20.7        | 6.4              | 75     | 34.0        | 10.5             | 65      | 29.6        | 9.2              |
|            | 5cm      | 60              | 27.4        | 8.5              | 70     | 31.8        | 9.8              | 110     | 49.6        | 15.4             |
|            | 10cm     | 35              | 16.2        | 5.0              | 60     | 27.4        | 8.5              | 90      | 40.7        | 12.6             |
|            | 20cm     | 40              | 18.5        | 5.7              | 60     | 27.4        | 8.5              | 75      | 34.0        | 10.5             |
|            | 30cm     | 40              | 18.5        | 5.7              | 40     | 18.5        | 5.7              | 60      | 27.4        | 8.5              |
|            | 40cm     | 45              | 20.7        | 6.4              | 50     | 22.9        | 7.1              | 70      | 31.8        | 9.8              |
|            | 50cm     | 45              | 20.7        | 6.4              | 40     | 18.5        | 5.7              | 50      | 22.9        | 7.1              |
| 4          | 地表上      | 140             | 62.9        | 19.5             | 90     | 40.7        | 12.6             | 110     | 49.6        | 15.4             |
|            | 5cm      | 120             | 54.0        | 16.7             | 120    | 54.0        | 16.7             | 120     | 54.0        | 16.7             |
|            | 10cm     | 120             | 54.0        | 16.7             | 120    | 54.0        | 16.7             | 105     | 47.4        | 14.7             |
|            | 20cm     | 110             | 49.6        | 15.4             | 110    | 49.6        | 15.4             | 100     | 45.1        | 14.0             |
|            | 30cm     | 90              | 40.7        | 12.6             | 115    | 51.8        | 16.0             | 90      | 40.7        | 12.6             |
|            | 40cm     | 60              | 27.4        | 8.5              | 130    | 58.5        | 18.1             | 90      | 40.7        | 12.6             |
|            | 50cm     | 50              | 22.9        | 7.1              | 90     | 40.7        | 12.6             | 70      | 31.8        | 9.8              |

\*印は貫入不可の深さ

表10 土質試験結果一覧表 (基礎地盤)

| 資料番号 (深さ) m~m |          |                                   |
|---------------|----------|-----------------------------------|
| 一般            | 湿潤密度     | $\rho_1$ g/cm <sup>3</sup>        |
|               | 乾燥密度     | $\rho_1$ g/cm <sup>3</sup>        |
|               | 土粒子の密度   | $\rho_s$ g/cm <sup>3</sup>        |
|               | 自然含水比    | $w_n$ %                           |
|               | 間隙比      | $e$                               |
|               | 飽和度      | $S_r$ %                           |
|               | 礫分       | 2~75mm%                           |
|               | 砂分       | 75 $\mu$ m~2mm %                  |
|               | シルト分     | 5~75 $\mu$ m%                     |
|               | 粘土分      | 5 $\mu$ m未満%                      |
| 粒度            | 均等係数     | $U_c$                             |
|               | 曲率係数     | $U_c$                             |
|               | 最大粒径     | mm                                |
|               | 液性限界     | $w_L$ %                           |
|               | 塑性限界     | $w_P$ %                           |
| 特性            | 塑性指数     | $I_P$ %                           |
|               | 分類名      |                                   |
| 分類            | 分類記号     | (VH2)                             |
|               | 一軸圧縮強さ   | $q_u$ kgf/cm <sup>2</sup>         |
| 一軸圧縮          | 試験条件     |                                   |
|               | 全応力      | $c$ kg/cm <sup>2</sup>            |
|               | $\phi$ 度 |                                   |
| 三軸圧縮          | 有効応力     | $c$ kg/cm <sup>2</sup>            |
|               | $\phi$ 度 |                                   |
| 圧密            | 圧縮指数     |                                   |
|               | 圧密降伏応力   | $P_{ck}$ kg/cm <sup>2</sup> 3. 実験 |

# 6) インタビュー結果

実験の被験者であるオペレータ2名にそれぞれの以下の規則についてインタビューを行った。その結果を以下に示す。

## ● 掘削前インタビュー

日 時： 1992年10月23日

操作者： (19歳, 運転経験 5 ヶ月)

| 分 類  | 設 問                            | 回 答                                        |
|------|--------------------------------|--------------------------------------------|
| 掘削関連 | 1 何往復位で所定の深さまで掘削できると思うか？       | 土の硬さによるから答えるのは難しい。10回位か？                   |
|      | 2 掘削時のブレードの下げのタイミングはどのようにとるのか？ | ブレードのわきを確認する。                              |
|      | 3 掘削時のブレードの深さはどのように調整するのか？     | ブレードのわきを見、盤から下がったかをチェックする。                 |
|      | 4 掘削時のブレードの上げのタイミングはどのようにとるのか？ | 盤のわきを見る。                                   |
| 盛土関係 | 5 押土時のブレードの高さはどのように調整するのか？     | まわりを見ながら、また、エンジン負荷のかかり具合、履帯のすべり状況により、調整する。 |
|      | 6 何往復位で所定の盛土ができそうか？            | 掘削と同様、土の状況によるから答えにくい。                      |
|      | 7 盛土高さはどのように判断するのか？            | 基本的にはトンボ、丁張りによる。これらがなければ勘。                 |
|      | 8 盛土の勾配の角度はどのように判断するのか？        | 本格的に行うのであれば丁張りによる。                         |
| 作業全体 | 9 作業時間はどの位かかりそうか？              | 15分位か？                                     |
| その他  |                                | 試験ブルドーザの履帯が丸くなっており、これが掘削作業に影響することが考えられる。   |

● 掘削後インタビュー

日 時： 1992年10月23日

操作者： (19歳, 運転経験 5 ヶ月)

1. 掘削は、自分のイメージしたようにできたか。

うまくいかなかった。

土が硬いこともあるが、ブルドーザの運転に慣れていないことによると思う。

2. ブルドーザをこのように改良したら運転がやりやすくなるという希望はあるか。

思いつかない。

● 掘削後インタビュー

日 時： 1992年10月23日

操作者： (46歳, 運転経験16年)

| 分 類  | 設 問                            | 回 答                   |
|------|--------------------------------|-----------------------|
| 掘削関連 | 1 何往復位で所定の深さまで掘削できると思うか？       | 硬さもあるが、3～4回か。         |
|      | 2 掘削時のブレードの下げのタイミングはどのようにとるのか？ | ブレードとラインを見て。          |
|      | 3 掘削時のブレードの深さはどのように調整するのか？     | ブレードの刺さった感じで。         |
|      | 4 掘削時のブレードの上げのタイミングはどのようにとるのか？ | ブレードとラインを見て。          |
| 盛土関係 | 5 押土時のブレードの高さはどのように調整するのか？     | 負荷のかかり具合。板の高さを見ながら。   |
|      | 6 何往復位で所定の盛土ができそうか？            | 5～6回。                 |
|      | 7 盛土高さはどのように判断するのか？            | 丁張りがあればこれを見ながら。無ければ勘。 |
|      | 8 盛土の勾配の角度はどのように判断するのか？        | 感覚。慣れが必要と思う。          |
| 作業全体 | 9 作業時間はどの位かかりそうか？              | 10分もかからないと思う。         |

● 掘削後インタビュー

日 時： 1992年10月23日

操作者： (46歳, 運転経験16年)

1. 掘削は、自分のイメージしたようにできたか。

地盤が硬いという点では、だいたい想像どおりであった。

なお、普段、使用しているブルドーザは、チルト、デセルが付いているものであり、試験用ブルドーザはこの面で操作がやりにくかった。また、これに加えて、試験方法が1方向の方押しということで、通常の掘削・整地が広い場所を縦横に走行して行うのと異なることから、掘削の表面の整地ができなかったことを付け加える。

2. ブルドーザをこのように改良したら運転がやりやすくなるという希望（例えば自動化）はあるか。

思い浮かばない。掘削などおおきく土を動かす場合は人間の判断が必要であろうし、仕上げではトンボを確認すればよい。

近年の戦車のようにサスペンションにより車体の傾きや高さを調製する機構があれば、掘削などがやりにくくなるかも知れない。

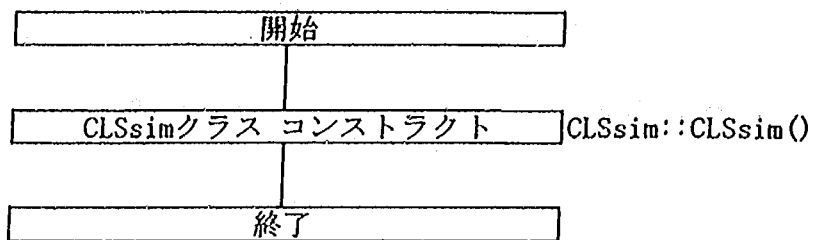


## 2. ブルドーザ・オペレーション・シミュレータ・プログラムリスト

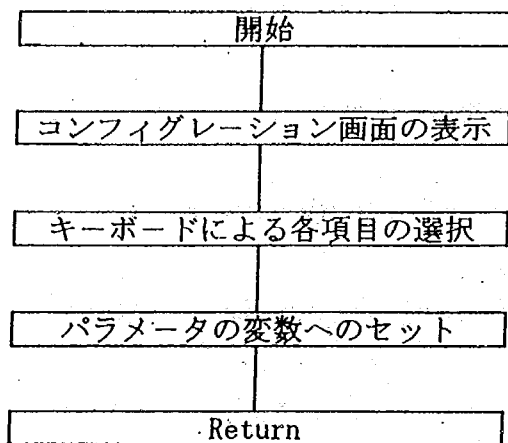


## フローチャート

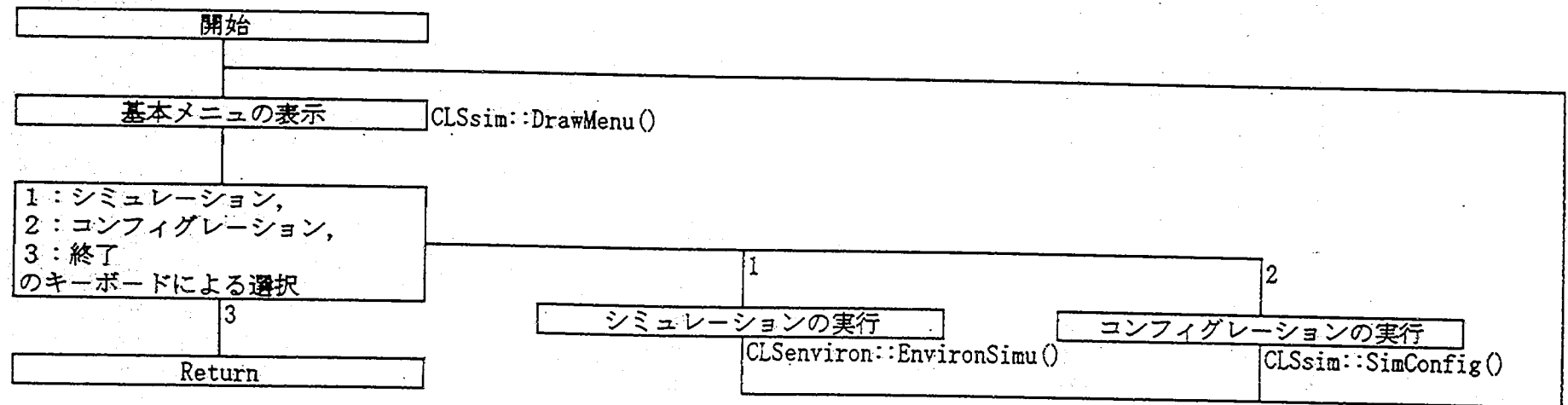
ブルドーザシミュレータフローチャート main ()

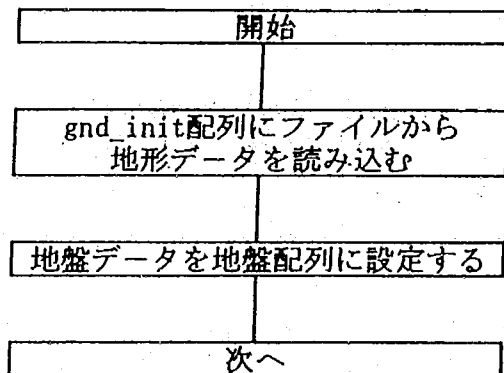
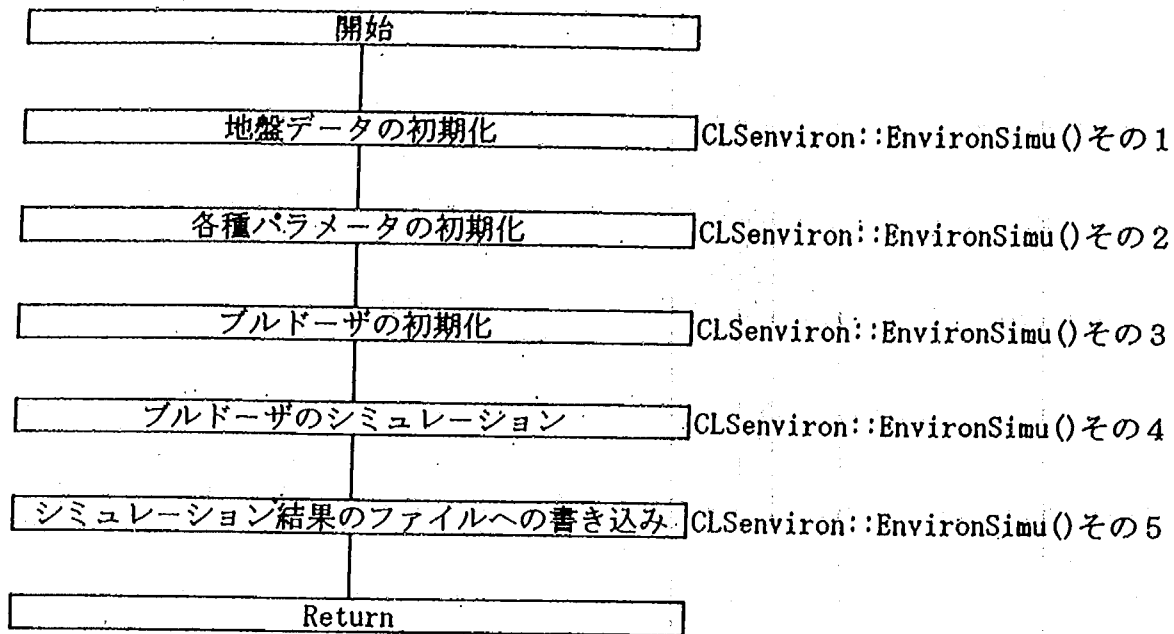


コンフィグレーション CLSSim::SimConfig ()

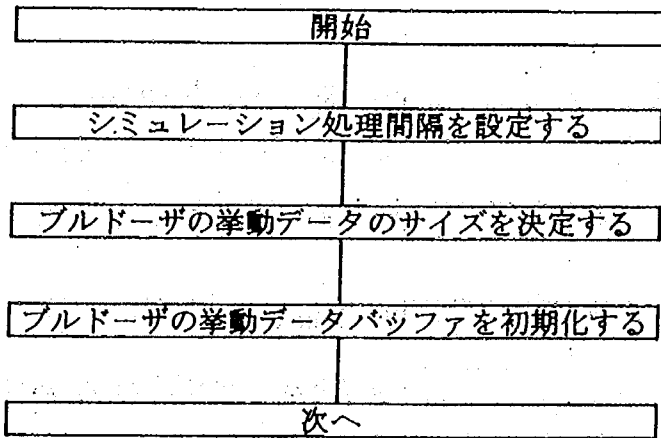


CLSSim クラス コンストラクタ CLSsim::CLSsim ()

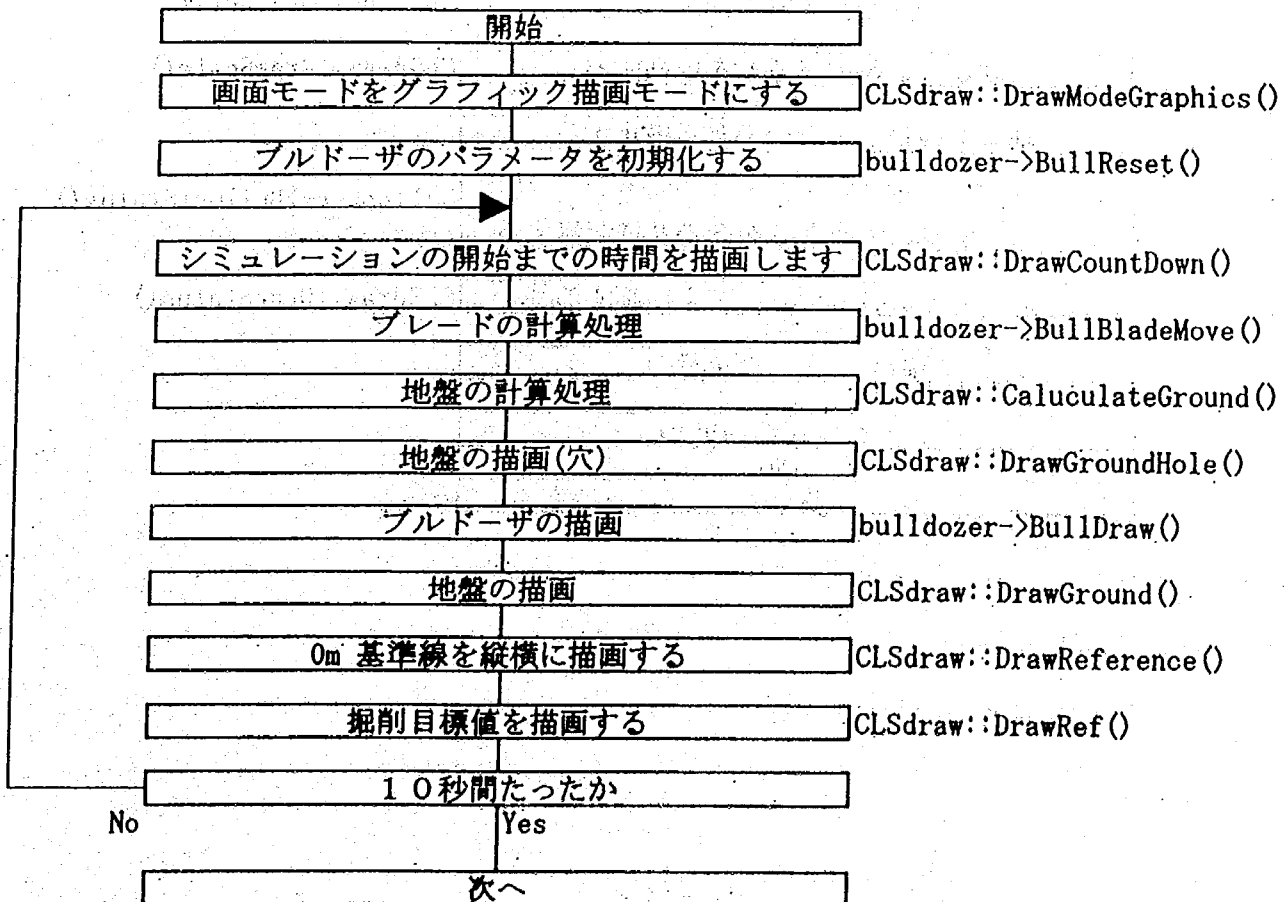


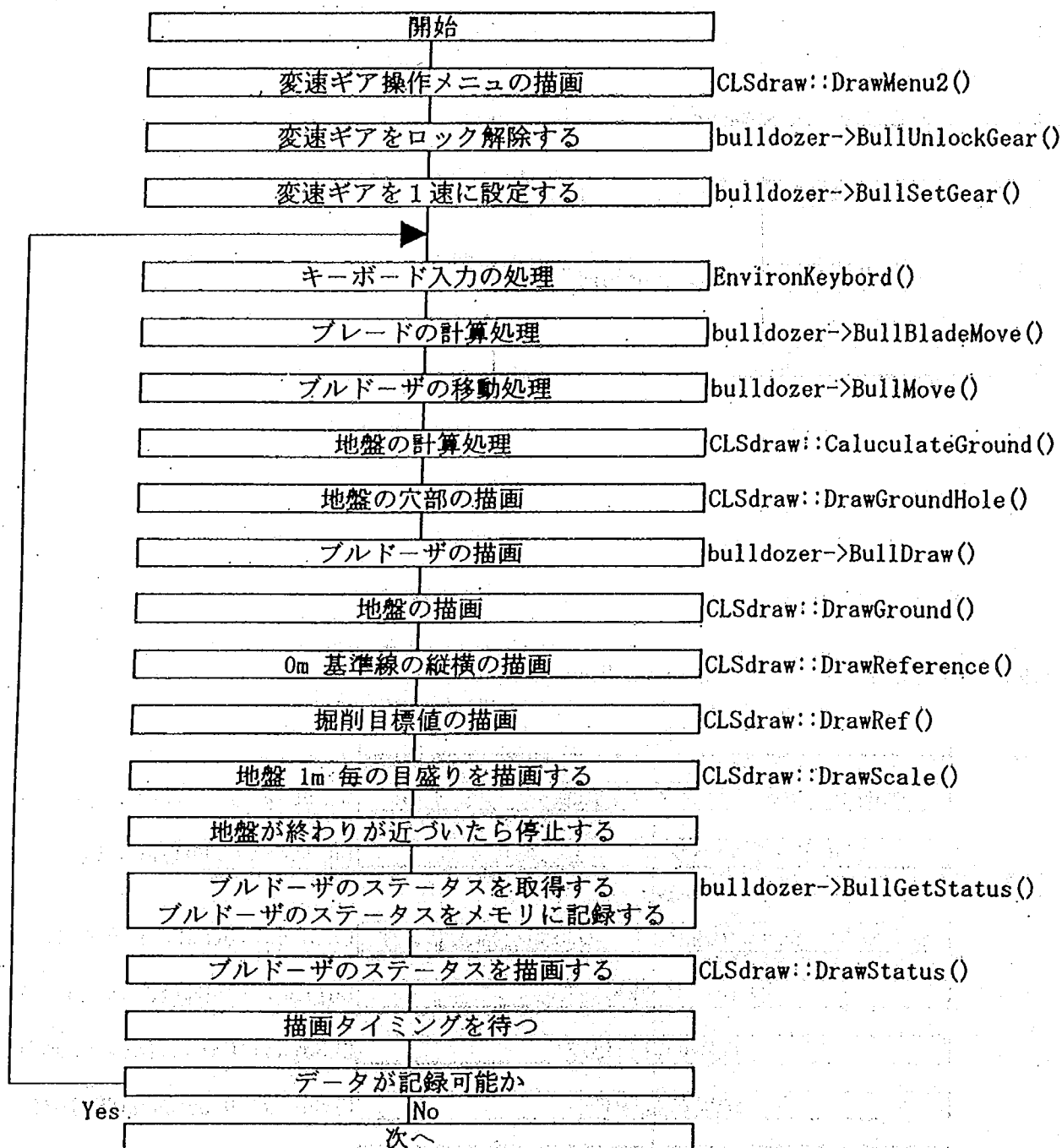


各種パラメータの初期化 CLSenvirom::EnvironSimu () その2

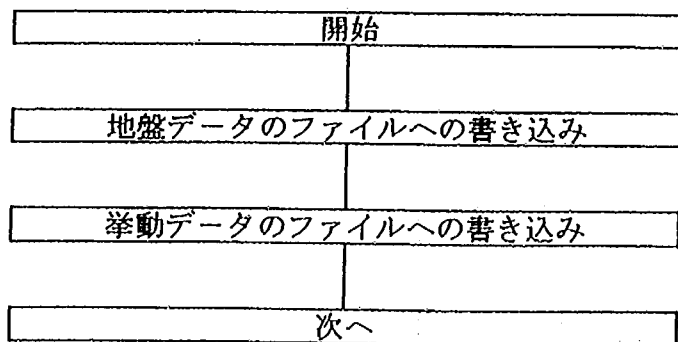


ブルドーザの初期化 CLSenvirom::EnvironSimu () その3

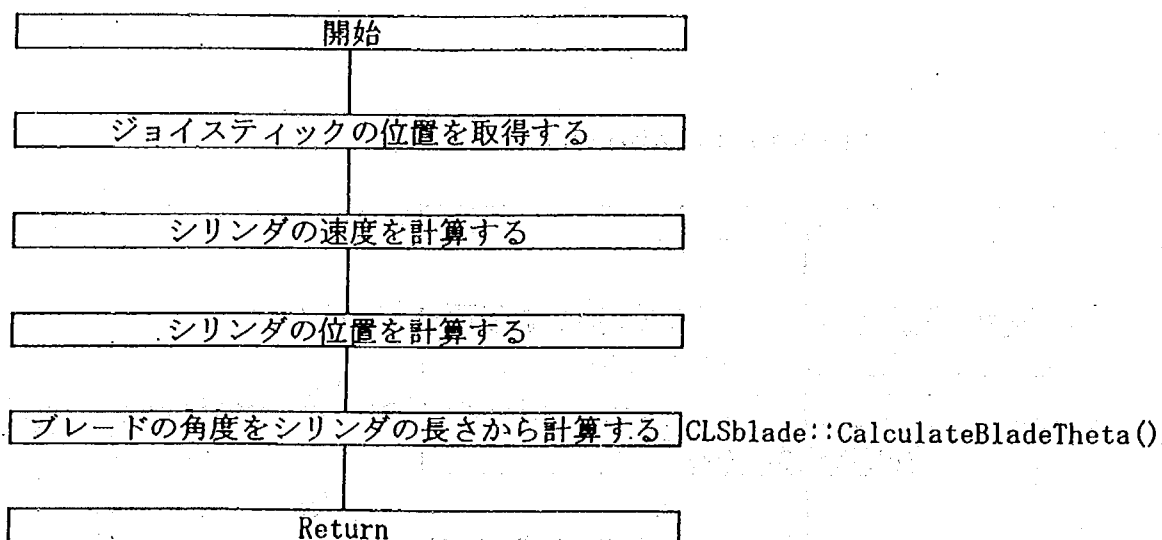




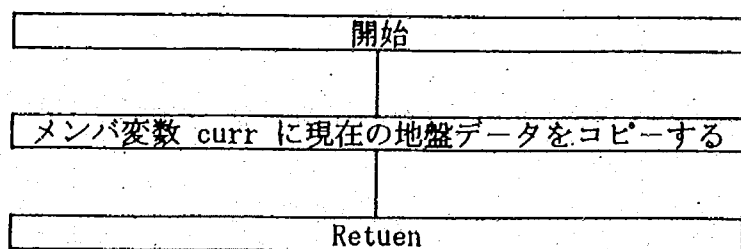
シミュレーション結果のファイルへの書き込み CLSenvirom::EnvironSimu () その5



ブレードの計算処理 CLSblade::BullBladeMove ()

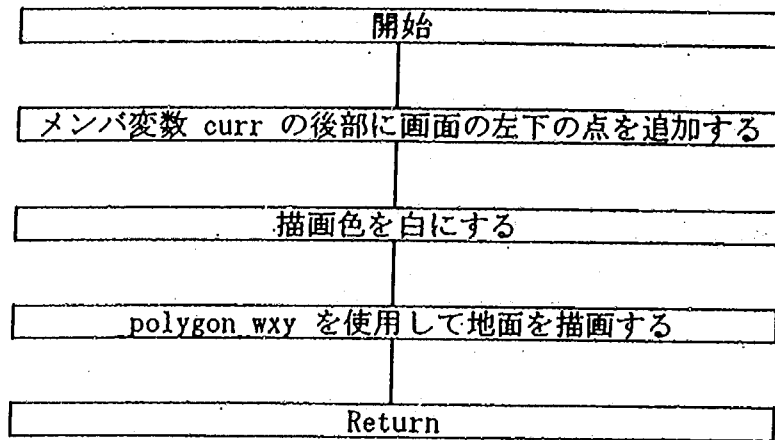


地盤の計算処理 CLSdraw::CaluculateGround ()



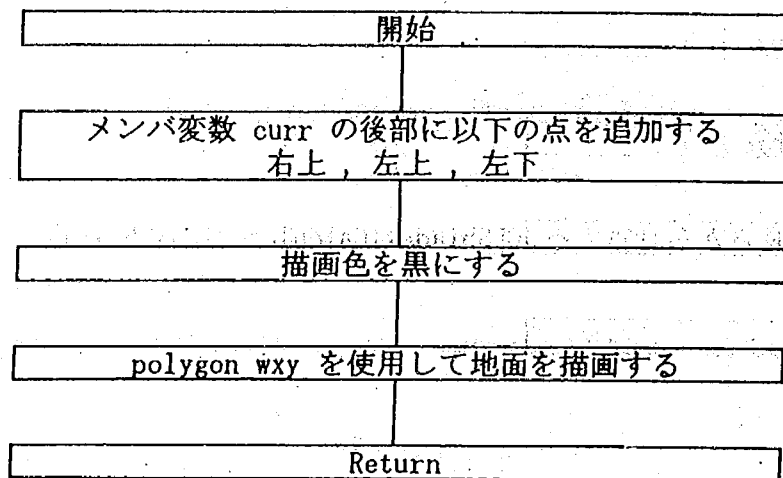
地盤の描画

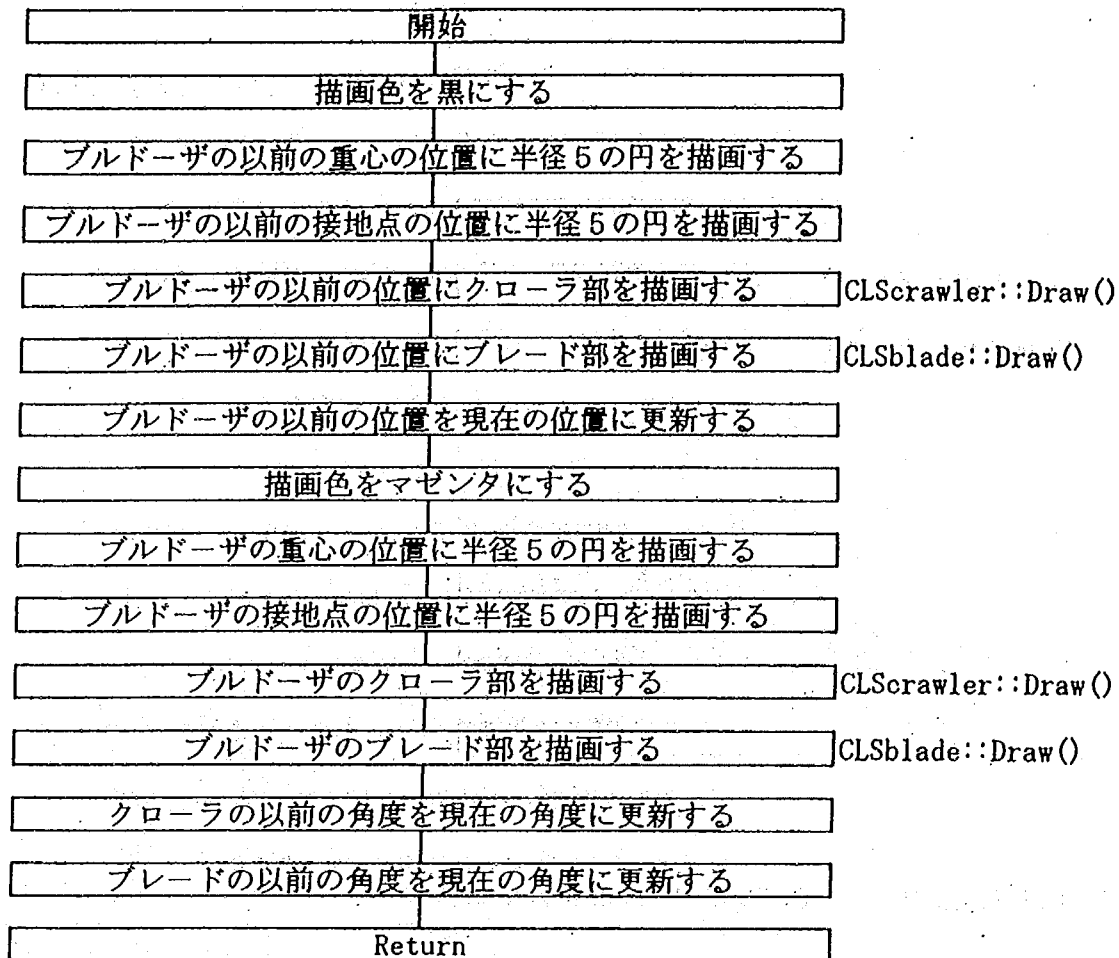
CLSdraw::DrawGround ()

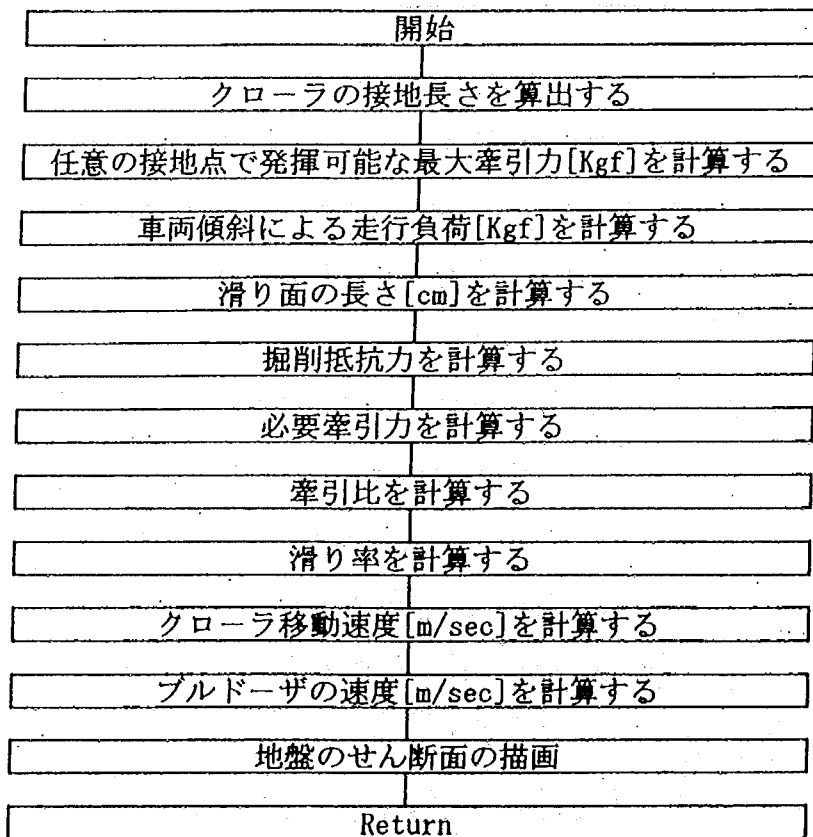
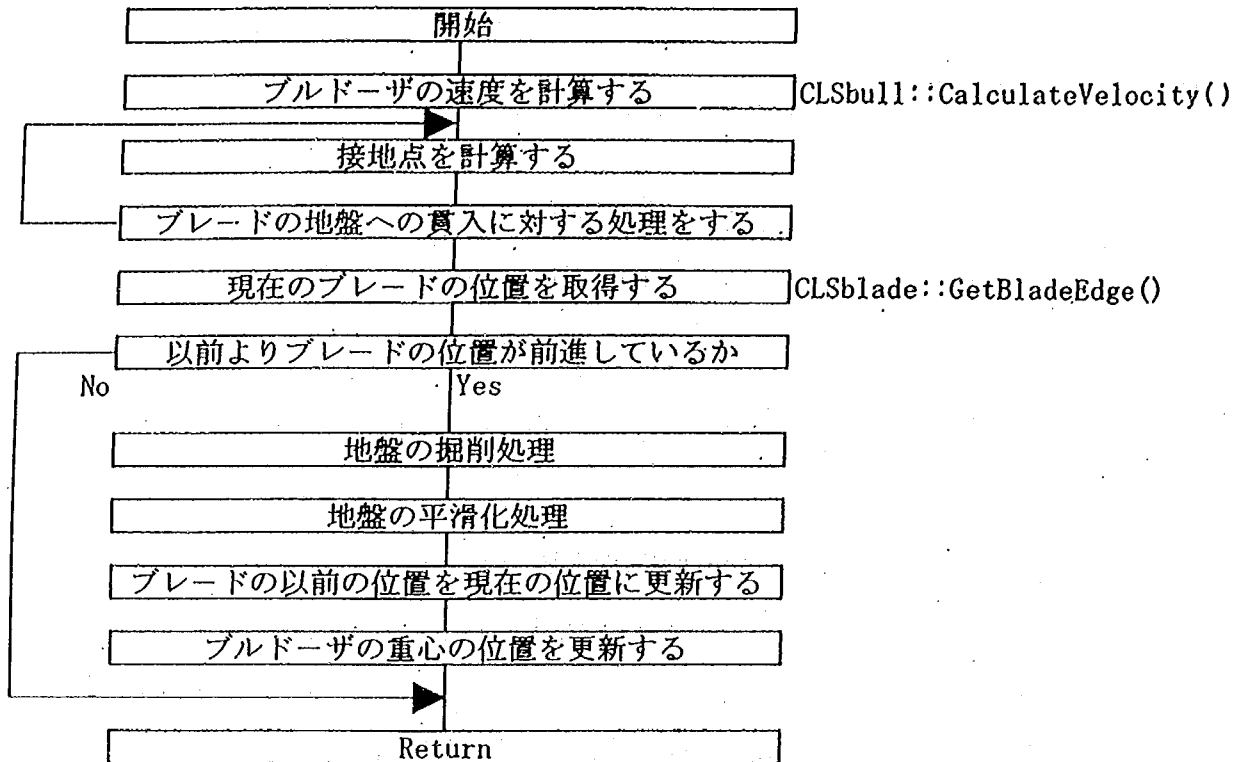


地盤の描画 (穴部)

CLSdraw::DrawGroundHole ()







## ソースリスト

```

/*-----
プログラム名  blade.cc
処理対象      :   ブレードシミュレーター   ブレードの製作
-----*/

```

```

#include <stdio.h>
#include <graph.h>
#include <math.h>

```

```

#include "etch"
#include "blade.h"
#include "console.h"
#include "magic.h"

```

```

/*-----
機能      :   ブレードの初期化
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :   ptBlade[]関数について 0点,1線,2丸
起り点    :
注意      :
-----*/

```

```

CLSblade::CLSblade(void)
{

```

```

    int n=0;
    ptBlade[n++].set(0,159, 44);      //blade center position
    ptBlade[n++].set(1,494, 0);      //blade edge position

```

```

// ブレードの支柱メイン

```

```

    ptBlade[n++].set(0,143,30);
    ptBlade[n++].set(1,146,60);
    ptBlade[n++].set(1,454,30);
    ptBlade[n++].set(1,454,0);
    ptBlade[n++].set(4,0,0);

```

```

// ブレードの支柱補助

```

```

    ptBlade[n++].set(0,454,110);
    ptBlade[n++].set(1,300,45);
    ptBlade[n++].set(1,356,40);
    ptBlade[n++].set(1,454,80);
    ptBlade[n++].set(4,0,0);

```

```

// ブレードの歯

```

```

    ptBlade[n++].set(0,454,0);
    ptBlade[n++].set(1,494,0);
    ptBlade[n++].set(1,484,10);
    ptBlade[n++].set(1,484,120);
    ptBlade[n++].set(1,454,120);
    ptBlade[n++].set(4,0,0);

```

```

    dbBladeWidth=IDD_BladeWidth;      // ブレード幅[m]
    dbBladeAtsusi=IDD_BladeAtsusi;    // ブレードの厚さ[m]
    bBladeDown=1;                     // 1:地盤にブレードが潜り込める 0:潜り込まない

```

```

// constants for cylinder length to blade angle calculation

```

```

    double c1, c2;                    // unit (mm)
    c1=(3370. - 1500.)*(3370. - 1500.)+(1840. - 440.)*(1840. - 440.);
    c2=((4940. - 1500.)*(4940. - 1500.)+(750./2.)*(750./2.));
    syl_to_bl_c1=c1+c2;                // mm^2
    syl_to_bl_c2=2.*sqrt(c1*c2);       // mm^2
    syl_to_bl_c3=atan(1840./3370. - 1500.)*atan(750./2/(4940. - 1500.)); // radian

```

```

    ResetCylinderLength();              // シリンダの長さの初期化
    CalculateBladeTheta();              // ブレードの角度をシリンダの長さから計算する
}

```

```

/*-----
機能      :   ブレードのデストラクタ
-----*/

```

引数 : なし  
 作成日 : 1995年10月23日  
 戻り値 : なし  
 補足 :  
 起動環境  
 注意 :

```

.....*/
CLSblade::~CLSblade(void)
{
}

```

```

/*-----
機能 : ブレードの先端の座標を取得する
引数 : int cx 本体の横位置[cm]
      int cy 本体の縦位置[cm]
      double dbCrawlerSin 本体の傾きの sin 値
      double dbCrawlerCos 本体の傾きの cos 値
      double dbBladeSin ブレードの傾きの sin 値
      double dbBladeCos ブレードの傾きの cos 値
作成日 : 1995年10月23日
戻り値 : ブレードの先端の座標
補足 :
起動環境
注意 :
.....*/

```

```

CLSpoint CLSblade::GetBladeEdge(int cx,int cy,double dbCrawlerSin,double dbCrawlerCos,double dbBladeSin,double dbBladeCos)
{
    CLSpoint ptCenter=ptBlade[0].Change1(cx,cy,dbCrawlerSin,dbCrawlerCos);
    CLSpoint ptEdge;
    ptEdge=ptBlade[1].Change1(cx,cy,dbCrawlerSin,dbCrawlerCos);
    ptEdge=ptEdge-ptCenter;
    ptEdge=ptEdge.Change1(ptCenter.x,ptCenter.y,dbBladeSin,dbBladeCos);
    return (ptEdge);
}

```

```

/*-----
機能 : ブレードの先端の座標を取得する
引数 : int cx 本体の横位置[cm]
      int cy 本体の縦位置[cm]
      double dbCrawlerTheta 本体の傾きの角度[rad]
      double dbBladeTheta ブレードの傾きの角度[rad]
作成日 : 1995年10月23日
戻り値 : ブレードの先端の座標
補足 :
起動環境
注意 :
.....*/

```

```

CLSpoint CLSblade::GetBladeEdge(int cx,int cy,double dbCrawlerTheta,double dbBladeTheta)
{
    CLSpoint ptCenter=ptBlade[0].Change1(cx,cy,dbCrawlerTheta);
    CLSpoint ptEdge;
    ptEdge=ptBlade[1].Change1(cx,cy,dbCrawlerTheta);
    ptEdge=ptEdge-ptCenter;
    ptEdge=ptEdge.Change1(ptCenter.x,ptCenter.y,dbBladeTheta);
    return (ptEdge);
}

```

```

/*-----
機能 : ブレードの各ノードの座標を表示する
引数 : int nNumber 表示するノードの番号
作成日 : 1995年10月23日
戻り値 : なし
補足 : テスト用の関数
起動環境
注意 :
.....*/

```

```

void CLSblade::print(int nNumber)
{

```

```

        printf("%d %d %d\n", ptBlade[nNumber].nFunction, ptBlade[nNumber].x, ptBlade[nNumber].y);
    }
}

```

```

/*-----
機能      :   ブレードを描画する
引数      :   int      cx          本体の横位置[cm]
               int      cy          本体の縦位置[cm]
               double dbCrawlerSin  本体の傾きの sin 値
               double dbCrawlerCos  本体の傾きの cos 値
               double dbBladeSin    ブレードの傾きの sin 値
               double dbBladeCos    ブレードの傾きの cos 値

作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境  :
注意      :
-----*/

```

```

void CLSblade::Draw(int cx,int cy,double dbCrawlerSin,double dbCrawlerCos,double dbBladeSin,double dbBladeCos)
{

```

```

    int i;
    CLSpoint ptDraw;
    CLSpoint ptCenter;
    int nLinePoint=0;
    struct _wxycoord *xyNode=new _wxycoord[MaxBladePoint];

    ptCenter=ptBlade[0].Change1(cx,cy,dbCrawlerSin,dbCrawlerCos);
    for (i=2;i<MaxBladePoint;i++)
    {
        ptDraw=ptBlade[i].Change1(cx,cy,dbCrawlerSin,dbCrawlerCos);
        ptDraw=ptDraw-ptCenter;
        ptDraw=ptDraw.Change1(ptCenter.x,ptCenter.y,dbBladeSin,dbBladeCos);
        switch (ptBlade[i].nFunction)
        {
            case 0:          // 点 {0,x,y}
            {
                xyNode[0].wx=ptDraw.x;
                xyNode[0].wy=ptDraw.y;
                nLinePoint=1;
                break;
            }
            case 1:          // 線 {1,x,y}
            {
                xyNode[nLinePoint].wx=ptDraw.x;
                xyNode[nLinePoint++].wy=ptDraw.y;
                break;
            }
            case 4:          // 描画
            {
                _polygon_wxy(_GBORDER,nLinePoint,xyNode);
                break;
            }
        }
    }
}

```

```

/*-----
機能      :   ブレードを描画する
引数      :   int      cx          本体の横位置[cm]
               int      cy          本体の縦位置[cm]
               double dbCrawlerTheta 本体の傾きの角度[rad]
               double dbBladeTheta   ブレードの傾きの角度[rad]

作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境  :
注意      :
-----*/

```

```

void CLSblade::Draw(int cx,int cy,double dbCrawlerTheta,double dbBladeTheta)
{

```

```

int i;
CLSpoint ptDraw;
CLSpoint ptCenter;
int nLinePoint=0;
struct _wxycoord *xyNode=new _wxycoord[MaxBladePoint];

ptCenter=ptBlade[0].Change1(cx,cy,dbCrawler/Theta);
for (i=2;i<MaxBladePoint;i++)
{
    ptDraw=ptBlade[i].Change1(cx,cy,dbCrawler/Theta);
    ptDraw=ptDraw-ptCenter;
    ptDraw=ptDraw.Change1(ptCenter.x,ptCenter.y,dbBlade/Theta);
    switch (ptBlade[i].nFunction)
    {
        case 0: // 点 (0,x,y)
        {
            xyNode[0].wx=ptDraw.x;
            xyNode[0].wy=ptDraw.y;
            nLinePoint=1;
            break;
        }
        case 1: // 線 (1,x,y)
        {
            xyNode[nLinePoint].wx=ptDraw.x;
            xyNode[nLinePoint++].wy=ptDraw.y;
            break;
        }
        case 4:
        {
            _polygon_wxy_GBORDER, nLinePoint, xyNode);
            break;
        }
    }
}
}

```

```

/*
機能 : ブレードの動作
引数 : double tim 移動時間[sec]
作成日 : 1995年10月23日
戻り値 : なし
補足 :
起滅原因
注意 :
*/

```

```

void CLSblade::BullBladeMove(double tim)
{
    double cylinder_vel=0; // [mm/sec]

    // calculate speed;
    lever_position = Joystick_Y0 * 140; // -140 mm < lever displacement < 140 mm

    if (lever_position >= 0)
    {
        if (lever_position < 65)
            cylinder_vel = 0;
        else if (lever_position < 70)
            cylinder_vel = (25. * (lever_position - 65)) / (70. - 65);
        else if (lever_position < 80)
            cylinder_vel = (25. + (75. - 25) * (lever_position - 70) / (80. - 70));
        else if (lever_position < 120)
            cylinder_vel = (75. + (350. - 75) * (lever_position - 80) / (120. - 80));
        else
            cylinder_vel = 350;
        cylinder_vel = -cylinder_vel;
    }
    else
    {
        if (dbBladeDown) // 地盤にブレードが潜り込めるなら
        {

```

```

        lever_position = -lever_position;
        if (lever_position < 25)
            cylinder_vel = 0;
        else if (lever_position < 100)
            cylinder_vel = (500 * (lever_position - 25)) / (100 - 25);
        else
            cylinder_vel = 500;
        lever_position = -lever_position;
    }
    cylinder_jeng += cylinder_vel * tim; //mm
    if (cylinder_jeng > 1745) cylinder_jeng = 1745;
    if (cylinder_jeng < 710) cylinder_jeng = 710;

    CalculateBladeTheta(); // ブレードの角度をシリンダの長さから計算する
}

```

```

/*-----*/
機能 : ブレードの角度をシリンダの長さから計算する
引数 : なし
作成日 : 1995年10月23日
戻り値 : なし
補足 :
起動環境
注意 :
/*-----*/

```

```

void CLSblade::CalculateBladeTheta(void)
{
    double alpha = acos((syl_to_bl_c1 - cylinder_jeng * cylinder_jeng) / syl_to_bl_c2);
    theta_blade = syl_to_bl_c3 * alpha; // ブレードの角度[rad]
    sin_blade = sin(theta_blade); // ブレードの角度の[sin]
    cos_blade = cos(theta_blade); // ブレードの角度の[cos]
}

```

```

/*-----*/
機能 : ブレードのシリンダの長さを初期値にする
引数 : なし
作成日 : 1995年10月23日
戻り値 : なし
補足 :
起動環境
注意 :
/*-----*/

```

```

void CLSblade::ResetSylinderLength(void)
{
    cylinder_jeng = IDD_SylinderLength; // シリンダの長さ[mm]
    CalculateBladeTheta(); // ブレードの角度をシリンダの長さから計算する
}

```

```

/*-----*/
機能 : ブレードのシリンダの長さの取得
引数 : なし
作成日 : 1995年10月23日
戻り値 : ブレードのシリンダの長さ[mm]
補足 :
起動環境
注意 :
/*-----*/

```

```

double CLSblade::GetSylinderLength(void)
{
    return (cylinder_jeng);
}

```

```

/*-----*/
機能 : ジョイスティックの位置の取得
引数 : なし
作成日 : 1995年10月23日

```

戻り値 : ジョイスティックの位置[mm]  
補足 :  
起動環境  
注意 :

.....\*/

double CLSblade::GetLeverPosition(void)

{  
    return (lever\_position);  
}

/\*=====||End of File||=====\*/

□

```

/*-----
プログラム名  bull.cc
処理概要      :   ブルドーザシミュレーター   ブルドーザ全体の動作
*/-----

```

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graph.h>

```

```

#include "magic.h"
#include "etc.h"
#include "crawler.h"
#include "blade.h"
#include "console.h"
#include "bull.h"
#include "draw.h"

```

```

/*-----
機能      :   ブルドーザの初期化
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境
注意      :
*/-----

```

```

CLSbull::CLSbull(void)
{
    grav_accel = IDD_GravityAccel;           // 重力加速度[m/s^2]
    gravity_x   = IDD_BullGravityX;          // 重心の位置 x[m]      all length unit is meter.
    gravity_y   = IDD_BullGravityY;          // 重心の位置 y[m]
    dbBullWeight = IDD_BullWeight;           // 本体重量[kg]
    dbKeninKeisu = IDD_KeninKeisu;           // 牽引係数
    dbTauMax     = IDD_TauMax;                // 地盤のせん断強度[Kg/m^2]
    dbAlpha      = IDD_Alpha;                // 地盤のせん断摩擦角[度]
    dbGroundTheta = IDD_GroundTheta;         // 地盤の滑り角[度]
    dbk = IDD_K;                             // 牽引比滑り率 特性式の次数
    nTsutiHeikin1 = IDN_TsutiHeikin1;        // 土の移動平均平滑化処理の長さ[cm]
    nTsutiHeikin2 = IDN_TsutiHeikin2;        // 土の移動平均平滑化処理の点数
    dbRunResistance = IDD_RunResistance;     // 走行抵抗[Kg]
    dbAdhesion     = IDD_Adhesion;           // 地盤の単位面積当たりの粘着力[Kg/m^2]
    dbNue = IDD_Nue;                         // 換算係数
    dbZibunHenkei = IDD_ZibunHenkei;        // 地盤の変形係数

    BullReset();
}

```

```

/*-----
機能      :   ブルドーザのデストラクタ
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境
注意      :
*/-----

```

```

CLSbull::~CLSbull(void)
{
}

```

```

/*-----
機能      :   ブルドーザの初期化(変数)
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境

```

注意 :

```

void CLSbull::BullReset(void)
{
    velocity = 0.0;           // 速度[m/sec]
    BullSetGear(0);           // ギア

    fulldown_flag = NOT_FALL; // 落下フラグ
    exe_flag = 0;
    gx = gravity_x * 100;     // 重心の位置 x[cm]
    gy = gravity_y * 100;     // 重心の位置 y[cm]
    prev_gravx = 0;           // 重心の位置 x[cm](以前の)
    prev_gravy = 0;           // 重心の位置 y[cm](以前の)
    supx1 = supx2 = supy1 = supy2 = 0; // 接地点
    prev_supx1 = prev_supx2 = prev_supy1 = prev_supy2 = 0; // 接地点(以前の)
    y_trans = 0;              // 車体の高さ

    tht_revol = 0.0;          // クローラの角度[rad]
    sin_revol = 0.0;          // クローラの角度の[sin]
    tan_revol = 0.0;          // クローラの角度の[tan]
    cos_revol = 1.0;          // クローラの角度の[cos]
    angle_vel = 0.0;          // ブルドーザの落下時に変わる角度の速度[rad/s]
    CLSblade::ResetSylinderLength(); // ブレードのシリンダの長さを初期値にする
    CLSpoint.pt = GetBladeExlge(0, y_trans, sin_revol, cos_revol, sin_blade, cos_blade);
    prev_x_blade = pt.x;       // ブレードの以前の位置[m]
    prev_y_blade = pt.y;       // ブレードの以前の位置[m]
    dbSumMoveDist = 0;         // ブルドーザの移動距離の半端[cm]
}

```

機能 : ブルドーザのギアをロックする。ギアを変化させなくする  
 引数 : なし  
 作成日 : 1995年10月23日  
 戻り値 : なし  
 補足 : nGearLock=1-ロック, 0-アンロック  
 起動環境  
 注意 :

```

void CLSbull::BullLockGear(void)
{
    nGearLock = 1;
}

```

機能 : ブルドーザのギアをアンロックする。ギアを変化出来るようにする  
 引数 : なし  
 作成日 : 1995年10月23日  
 戻り値 : なし  
 補足 : nGearLock=1-ロック, 0-アンロック  
 起動環境  
 注意 :

```

void CLSbull::BullUnlockGear(void)
{
    nGearLock = 0;
}

```

機能 : ブルドーザのギアをセットする  
 引数 : int nn 変更後のギア番号  
 作成日 : 1995年10月23日  
 戻り値 : なし  
 補足 : ギアがロックされていると変化しない  
 起動環境  
 注意 :

```

void CLSbull::BullSetGear(int nn)
{

```

```

        if(!nGearLock)
            nGear = nn;
    }

    /*-----
    機能      :   ブルドーザのギアを変更する
    引数      :   int    nd   ギアの変更量 +1up -1down
    作成日    :   1995年10月23日
    戻り値    :   なし
    補足      :   0から3の間のみ動く
    起動/環境 :
    注意      :
    -----*/

```

```

void CLSbull::BullChangeGear(int nd)

```

```

{
    int ng = nGear + nd;
    if(ng < 0)
        ng = 0;
    if(ng > 3)
        ng = 3;
    BullSetGear(ng);
}

```

```

    /*-----
    機能      :   ブルドーザの移動
    引数      :   double t       移動時間[sec]
                int          *gnd 現在の地面を指すポインタ
    作成日    :   1995年10月23日
    戻り値    :   移動した距離[m]
    補足      :   掘削および地形データの平滑化をも行う
    起動/環境 :
    注意      :
    -----*/

```

```

double CLSbull::BullMove(double t, int *gnd)

```

```

{
    int *gnd;
    int offset_index;
    double moved_dist = CalculateVelocity(gnd) * t;
    dbSumMoveDist += (moved_dist * 100);
    if(dbSumMoveDist < 1.0)
    {
        offset_index = 0;
    }
    else
    {
        offset_index = (int)dbSumMoveDist;
        dbSumMoveDist = dbSumMoveDist - offset_index;
    }
    // 移動距離 [m]
    // 移動距離 [cm]
    // ブルドーザの移動距離 実数 [cm]
    // ブルドーザの移動距離の半端の処理
    // 半端だから動かない
    // 半端がまったから動く

```

```

//Search 接地点を探す

```

```

    int f, b;
    int flase, blase;
    int start_index;
    // double dbGap_test = 0;
    double dbGap = 0;
    int count;

```

```

// 交差距離その2の再計算の終位置補正変数 [cm]

```

```

//Initialize

```

```

    count = 0;
    start_index = gx + offset_index;
    gnd1 = &(gnd[offset_index]);
    f = flase = start_index + 1;

```

```

label1:

```

```

// 交差距離その2の再計算の戻り場所

```

```

    b = blase = start_index;

```

```

    for (;
    {

```

```

        //Search
    }

```

```

double ux, uy, vx, vy;

f = flbase;
vx = flbase - blbase;
vy = grnd(flbase) - grnd(blbase);
do
{
    f++;
    if (wheel_base < f) break;
    ux = f - blbase;
    uy = grnd(f) - grnd(blbase);
} while (ux * vy - uy * vx > 0);
if (f <= wheel_base)
    flbase = f;
if (f <= wheel_base)
{
    //Search forward
    ux = flbase - blbase;
    uy = grnd(flbase) - grnd(blbase);
    do
    {
        f++;
        if (wheel_base < f) break;
        vx = ux;
        vy = uy;
        ux = f - blbase;
        uy = grnd(f) - grnd(blbase);
    } while (ux * vy - uy * vx <= 0);
    flbase = f - 1;
}

//Skip backward
b = blbase;
vx = blbase - flbase;
vy = grnd(blbase) - (grnd(flbase) + dlGap);
do
{
    --b;
    if (b < 0) break;
    ux = b - flbase;
    uy = grnd(b) - (grnd(flbase) + dlGap);
} while (ux * vy - uy * vx < 0);
if (0 <= b)
    blbase = b;
if (0 <= b)
{
    //Search backward
    ux = blbase - flbase;
    uy = grnd(blbase) - (grnd(flbase) + dlGap);
    do
    {
        --b;
        if (b < 0) break;
        vx = ux;
        vy = uy;
        ux = b - flbase;
        uy = grnd(b) - (grnd(flbase) + dlGap);
    } while (ux * vy - uy * vx >= 0);
    blbase = b + 1;
}

//quit
if (0 <= 0 && wheel_base <= f)
    break;

//end of loop
}

{
double a, b, c;
double tan_temp, tht_temp;
a = (grnd(flbase) + dlGap) - grnd(blbase);
b = flbase - blbase;
c = sqrt(a*a + b*b);
tan_temp = a / b;
supx1 = blbase;
supx2 = flbase;

```

```

supy1=grnd[lbase];
supy2=grnd[lbase]+dlGap;

falldown_flag=NOT_FALL;
if(tan_temp<tan_revol) //calculate falling down
{
double l;
double aa, bb;
tht_temp=atan(tan_revol);
supx2:=movex_dist*100;
// aa=(gravx-supx1);
bb=(gravy-supy1);
l=sqrt(aa*aa+bb*bb); //cm->m
angle_vel+=grav_xcel/l*cos_revol;
tht_revol:=angle_vel*t;
if(tht_temp>tht_revol) //fall down
{
falldown_flag=FAIL;
tan_revol=tan(tht_revol);
sin_revol=sin(tht_revol);
cos_revol=cos(tht_revol);
y_trans=supy1-tan_revol*supx1;
}
else
{
falldown_flag=NOT_FALL;
angle_vel=0;
}
}

if(falldown_flag==NOT_FALL)
{
angle_vel=0;
tan_revol=a/b;
sin_revol=a/c;
cos_revol=b/c;
tht_revol=atan(tan_revol);
y_trans=grnd[lbase]-tan_revol*blase;
//here lbase and blase are the supporting points 以下カ接地点
supx1=blase;
supx2=lbase;
supy1=grnd[lbase];
if(exc_flag==1)
{
CLSpoin ptBladeEdge;
ptBladeEdge=GetBladeEdge(0,grnd[lbase]-tan(tht_revol)*blase,sin(tht_revol),cos(tht_revol),sin_blade,cos_blade);
supy2=ptBladeEdge.y;
}
else
{
supy2=grnd[lbase];
}
}
}

// 交差検出その2 (11/25 追加) ブレードの地盤への貫入に対する処理
{
CLSpoin ptBladeEdge;
ptBladeEdge=GetBladeEdge(0,grnd[lbase]-tan(tht_revol)*blase,sin(tht_revol),cos(tht_revol),sin_blade,cos_blade);
//ptBladeEdge.x=ptBladeEdge.x*cos_revol;
if((ptBladeEdge.y<prev_y_blade)&&((int)ptBladeEdge.y<grnd[(int)ptBladeEdge.x]))
{
double s;
exc_flag=1;
falldown_flag=NOT_FALL;
angle_vel=0; // ブレード貫入量(m)
//count=count+1;
//if(count>=30) goto label2;
//if((gx-blase)==0) goto label2;
if((int)prev_y_blade<grnd[(int)ptBladeEdge.x])
s=(prev_y_blade-ptBladeEdge.y)*0.01;
}
}

```

```

else
    s = (grnd[(int)ptBladeEdge.x] - (int)ptBladeEdge.y) * 0.01;
    double p = -GetZAxis1Lenkei() * s; // 地盤の圧力[kg/m2]
    //
    double test_gx = gx * cos_revol - gy * sin_revol;
    double m1 = dbBladeWeight * (gx - blade) * 0.01; // モーメント(ブレードの重心)
    double pb = (dbBladeWidth * (dbBladeAtsuen * p; // 地盤の支持力[kg]
    double m2 = pb * (ptBladeEdge.x - blade) * 0.01; // モーメント(ブレード)
    //
    if ((test_gx - blade) == 0) m1 = 0;
    //
    if (s < 1) s = 0;
    if ((m1 + m2) < 0)
    {
        f = blade = (int)ptBladeEdge.x; // blade の値を再計算させるための前処理
        // dbCap_test = dbCap;
        //
        dbCap = 0;
        dbCap = (y_trans + (ptBladeEdge.x) * tan_revol - grnd[(int)ptBladeEdge.x] + 0.5;
        //
        if (((dbCap + grnd[(blade)] - (dbCap_test + grnd[(blade)])) < 1) dbCap = dbCap + 1;
        goto label1;
    }
    }
    exc_flag = 0;
// label2:
//
// tan_revol = tan(tht_revol);
//
// sin_revol = sin(tht_revol);
//
// cos_revol = cos(tht_revol);
//
// y_trans = grnd[(blade)] - tan_revol * blade;
//
}

// calculate a position of the blade
{
    CLSpoint ptBladeEdge = GetBladeEdge(0, y_trans, sin_revol, cos_revol, sin_blade, cos_blade);
    prev_x_blade = offset_index; // 以前にブレードの先端があった場所を補正する
    if (ptBladeEdge.x > prev_x_blade) // ブレードが前進した場合
    {
        int i, j;
        double lim;
// 掘削
        for (i = prev_x_blade; i <= ptBladeEdge.x; i++)
        {
            lim = prev_y_blade + (ptBladeEdge.y - prev_y_blade) / (ptBladeEdge.x - prev_x_blade) * (i - prev_x_blade);
            if (grnd[i] > lim)
                grnd[i] = lim; // 掘削
        }
    }

// 移動平均の地形データの平滑化
{
    int nMaxGnd = int(X_COORD - X_OFFSET); // 画面にうつっているぶん
    int *nGnd = new int[nMaxGnd]; // 地盤のマップ
    int nTsutiSum;
    int nWidth;
    int nHeikinIndex = (prev_x_blade + GetWheelBase() * cos_revol) / 2;
    int k = 0;
    for (i = 0; i < nMaxGnd; i++)
        nGnd[i] = grnd[i];
    for (j = -(nTsutiLeikin1 / 2); j <= nTsutiLeikin1 / 2; j++)
    {
        nTsutiSum = 0;
        nWidth = 0;
        for (i = -(nTsutiLeikin2 / 2); i <= nTsutiLeikin2 / 2; i++)
        {
            nTsutiSum += grnd[nHeikinIndex + i + j];
            nWidth++;
            k++;
        }
        nGnd[nHeikinIndex + j] = nTsutiSum / nWidth;
    }
    for (i = 0; i < nMaxGnd; i++)
        grnd[i] = nGnd[i];
    delete [] nGnd;
}

```

```

    printf("%d",k);exit(0);
    }
    prev_x_blade=ptBladeEdge.x;
    prev_y_blade=ptBladeEdge.y;
    }

    if((ptBladeEdge.x <= prev_x_blade) && (ptBladeEdge.y <= prev_y_blade))
    {
        if(gnd[(int)prev_x_blade]>ptBladeEdge.y)
        {
            gnd[(int)ptBladeEdge.x]=ptBladeEdge.y;
            prev_x_blade=ptBladeEdge.x;
            prev_y_blade=ptBladeEdge.y;
        }
    }
}

```

//calculate the coordinate of center of gravity. 重心

```

    gravx =      gx*cos_revol-gy*sin_revol;
    gravity=y_trans+gx*sin_revol+gy*cos_revol;

```

```

    return moved_dist;
}

```

```

/*-----
機能 :   ブルドーザを描画する
引数 :   なし
作成日 :   1995年10月23日
戻り値 :   なし
補足 :
起り/変更
注意 :
-----*/

```

void CLSBull::BullDraw(void)

```

{
    _setcolor(0);
    _ellipse_w(GBORDER, prev_gravx-5, prev_gravy-5, prev_gravx+5, prev_gravy+5); // 重心
    _ellipse_w(GBORDER, prev_supx1-5, prev_supy1-5, prev_supx1+5, prev_supy1+5); // 接地点前
    _ellipse_w(GBORDER, prev_supx2-5, prev_supy2-5, prev_supx2+5, prev_supy2+5); // 接地点後

    CLScrawler::Draw(0, dbOkly_trans, dbOkCrawlerSin, dbOkCrawlerCos);
    CLSblade::Draw(0, dbOkly_trans, dbOkCrawlerSin, dbOkCrawlerCos, dbOkBladeSin, dbOkBladeCos);

    prev_supx1=supx1;
    prev_supy1=supy1;
    prev_supx2=supx2;
    prev_supy2=supy2;
    prev_gravx=gravx;
    prev_gravy=gravity;

    _setcolor(3);
    _ellipse_w(GBORDER, gravx-5, gravity-5, gravx+5, gravity+5);
    _ellipse_w(GBORDER, supx1-5, supy1-5, supx1+5, supy1+5);
    _ellipse_w(GBORDER, supx2-5, supy2-5, supx2+5, supy2+5);

    CLScrawler::Draw(0, y_trans, sin_revol, cos_revol);
    dbOkCrawlerSin=sin_revol;
    dbOkCrawlerCos=cos_revol;
    dbOkly_trans=y_trans;
    CLSblade::Draw(0, y_trans, sin_revol, cos_revol, sin_blade, cos_blade);
    dbOkBladeSin=sin_blade;
    dbOkBladeCos=cos_blade;
}

```

```

/*-----
機能 :   ブルドーザのステータスを取得する
引数 :   CLSBullStatus *st   ステータスを戻すクラスへのポインタ

```

作成日 : 1995年10月23日  
 戻り値 : なし  
 補足 : 履帯は履帯の1マスからの位置  
 起動環境  
 注意 :

-----\*/

void CLSBull::BullGetStatus(CLSBullStatus \*st)

```
{
    st->velocity = velocity;
    st->bull_X = 0; //cm->m
    st->bull_Y = y_trans/100 + GetWheelDiameter()/100; //cm->m
    st->bull_THI = tht_revol;
    st->lever_pos = GetLeverPosition();
    st->sylinder_jeng = GetSylinderLength();
    st->blade_X = prev_x_blade/100; //cm->m
    st->blade_Y = prev_y_blade/100; //cm->m
    st->gear = nGear; // 変速ギア番号
}
```

-----\*/

機能 : ブルドーザの速度を計算する  
 引数 : int \*gnd 現在の地面を指すポイント  
 作成日 : 1995年10月23日  
 戻り値 : ブルドーザの速度  
 補足 :  
 起動環境  
 注意 :

-----\*/

double CLSBull::CalculateVelocity(int \*gnd)

```
{
    int i;
    double lg = GetGrossLength() * 100; // グローラの長さ[cm]

    double la = 0; // 接地長さ[cm]
    {
        double si; // クローラと地面の距離[cm]
        double yi;
        for (i = 0; i < wheel_base; i++) // クローラの接地長さ[cm]を求める
        {
            yi = y_trans + i * tan_revol;
            si = gnd[i] + lg - yi;
            if (si >= 0)
                la++;
        }
    }

    double tc = dbNuc * (2 * dbCrawlerWidth * dbAdhesion * la / 100 + dbBullWeight * tan(dbAlpha * 3.14 / 180)); // 任意の接地点で発生可能な最大牽引(Kgf)

    double tm = dbBullWeight * sin_revol; // 車重と地形による走行負荷(Kgf)

    {
        double dbSuberiKaku = dbGround[theta * 3.14 / 180; // 土の滑り角 [rad]
        double si; // 滑り面の高さ[cm]
        double le; // 滑り面の長さ[cm]
        double xe; // 滑り面の横長さ[cm]
        CLSpoint ptBladeEdge = GetBladeEdge(0, y_trans, sin_revol, cos_revol, sin_blade, cos_blade);
        for (i = 1, xe = 0; i < X_COORD - ptBladeEdge.x; i++, xe++)
        {
            si = ptBladeEdge.y + i * tan(dbSuberiKaku + tht_revol);
            if (si > gnd[i + int(ptBladeEdge.x)])
                break;
        }
        le = xe * 1 / cos(dbSuberiKaku + tht_revol); // 滑り面の長さ[cm]の算出

        double te = dbTauMax * le / 100 * dbBladeWidth; // 滑り抵抗(Kgf)
        double tt = tm + te + dbRunResistance; // 必要牽引(Kgf)
        double ht = tt / tc; // 牽引比
        double dbi; // 滑り率
        if (ht < 0)
            dbi = 0; // 滑り率の算出 0 以下 の場合
    }
}
```

```

else if (it < 1)
    dbi = pow(it, dbk);
else
    dbi = 1;
double vc;
switch (nGear)
{
    double v0, t0;
    case 1:
        v0 = IDD_Gear1V0/3.6;
        t0 = IDD_Gear1T0/1000;
        vc = v0 - (v0/t0) * (t/1000);
        break;
    case 2:
        v0 = IDD_Gear2V0/3.6;
        t0 = IDD_Gear2T0/1000;
        vc = v0 - (v0/t0) * (t/1000);
        break;
    case 3:
        v0 = IDD_Gear3V0/3.6;
        t0 = IDD_Gear3T0/1000;
        vc = v0 - (v0/t0) * (t/1000);
        break;
    default:
        vc = 0;
        break;
}
velocity = (1 - dbi) * vc;

if (xe)
{
    _setcolor(0);
    _move_to_w(ptBladeEdge.x, ptBladeEdge.y);
    _line_to_w(xe + ptBladeEdge.x, si - 1);
}
return (velocity);
}

```

```

/*-----*/
機能      :   牽引係数をセットする
引数      :   double db   牽引係数
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境
注意      :
/*-----*/

```

```

void CLSbull::SetKeninKeisuu(double db)
{
    dbKeninKeisuu = db;
}

```

```

/*-----*/
機能      :   牽引係数を取得する
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   牽引係数
補足      :
起動環境
注意      :
/*-----*/

```

```

double CLSbull::GetKeninKeisuu(void)
{
    return (dbKeninKeisuu);
}

```

```

/*-----*/

```

機能 : 地盤のせん断強度[ $\text{Kg}/\text{m}^2$ ]をセットする  
 引数 : double db 地盤のせん断強度[ $\text{Kg}/\text{m}^2$ ]  
 作成日 : 1995年10月23日  
 戻り値 : なし  
 補足 :  
 起動環境  
 注意 :

.....\*/  
 void CLSbull::SetTauMax(double db)

{  
     dbTauMax=db;  
 }

/\*.....\*/  
 機能 : 地盤のせん断強度[ $\text{Kg}/\text{m}^2$ ]を取得する  
 引数 : なし  
 作成日 : 1995年10月23日  
 戻り値 : 地盤のせん断強度[ $\text{Kg}/\text{m}^2$ ]  
 補足 :  
 起動環境  
 注意 :

.....\*/  
 double CLSbull::GetTauMax(void)

{  
     return (dbTauMax);  
 }

/\*.....\*/  
 機能 : 地盤の滑り角[度]をセットする  
 引数 : double db 地盤の滑り角[度]  
 作成日 : 1995年10月23日  
 戻り値 : なし  
 補足 :  
 起動環境  
 注意 :

.....\*/  
 void CLSbull::SetGroundTheta(double db)

{  
     dbGroundTheta=db;  
 }

/\*.....\*/  
 機能 : 地盤の滑り角[度]を取得する  
 引数 : なし  
 作成日 : 1995年10月23日  
 戻り値 : 地盤の滑り角[度]  
 補足 :  
 起動環境  
 注意 :

.....\*/  
 double CLSbull::GetGroundTheta(void)

{  
     return (dbGroundTheta);  
 }

/\*.....\*/  
 機能 : 赤井此滑り率 特許式の次数をセットする  
 引数 : double db 赤井此滑り率 特許式の次数  
 作成日 : 1995年10月23日  
 戻り値 : なし  
 補足 :  
 起動環境  
 注意 :

.....\*/  
 void CLSbull::SetK(double db)

{  
     dbk=db;  
 }

}

/\*-----\*/  
機能 : 本引比滑り率 特許式の次数を取得する  
引数 : なし  
作成日 : 1995年10月23日  
戻り値 : 本引比滑り率 特許式の次数  
補足 :  
起動環境  
注意 :

double CLSbull::GetK(void)  
{  
return (ilk);  
}

/\*-----\*/  
機能 : 土の移動平均平滑化処理の広さ[cm]をセットする  
引数 : int nn 本引比滑り率 特許式の次数  
作成日 : 1995年10月23日  
戻り値 : なし  
補足 :  
起動環境  
注意 :

void CLSbull::SetM(int nn)  
{  
nTsutilHeikin1=nn;  
}

/\*-----\*/  
機能 : 土の移動平均平滑化処理の広さ[cm]を取得する  
引数 : なし  
作成日 : 1995年10月23日  
戻り値 : 土の移動平均平滑化処理の広さ[cm]  
補足 :  
起動環境  
注意 :

int CLSbull::GetM(void)  
{  
return (nTsutilHeikin1);  
}

/\*-----\*/  
機能 : 土の移動平均平滑化処理の点数をセットする  
引数 : int nn 本引比滑り率 特許式の次数  
作成日 : 1995年10月23日  
戻り値 : なし  
補足 :  
起動環境  
注意 :

void CLSbull::SetN(int nn)  
{  
nTsutilHeikin2=nn;  
}

/\*-----\*/  
機能 : 土の移動平均平滑化処理の点数を取得する  
引数 : なし  
作成日 : 1995年10月23日  
戻り値 : 土の移動平均平滑化処理の点数  
補足 :  
起動環境  
注意 :

```

.....*/
int CLSbull::GetN(void)
{
    return (nTsutilIcikin2);
}

```

```

/*-----
機能      :  走行抵抗(Kg)をセットする
引数      :  なし
作成日    :  1995年10月23日
戻り値    :  なし
補足      :
起動環境
注意      :
.....*/

```

```

void CLSbull::SetRunResistance(double db)
{
    dbRunResistance=db;
}

```

```

/*-----
機能      :  走行抵抗(Kg)を取得する
引数      :  なし
作成日    :  1995年10月23日
戻り値    :  走行抵抗(Kg)
補足      :
起動環境
注意      :
.....*/

```

```

double CLSbull::GetRunResistance(void)
{
    return (dbRunResistance);
}

```

```

/*-----
機能      :  地盤の内摩擦角(度)をセットする
引数      :  なし
作成日    :  1995年10月23日
戻り値    :  なし
補足      :
起動環境
注意      :
.....*/

```

```

void CLSbull::SetAlpha(double db)
{
    dbAlpha=db;
}

```

```

/*-----
機能      :  地盤の内摩擦角(度)を取得する
引数      :  なし
作成日    :  1995年10月23日
戻り値    :  地盤の内摩擦角(度)
補足      :
起動環境
注意      :
.....*/

```

```

double CLSbull::GetAlpha(void)
{
    return (dbAlpha);
}

```

```

/*-----
機能      :  地盤の単位面積当たりの粘着(Kg0m^2)をセットする
引数      :  なし
作成日    :  1995年10月23日

```

戻り値 : なし  
 補足 :  
 起動環境 :  
 注意 :

.....\*/

void CLSbull::SetAdhesion(double db)

```
{
    dbAdhesion = db;
}
```

/\*.....

機能 : 地盤の単位面積当たりの粘着力[ $\text{Kg}/\text{m}^2$ ]を取得する  
 引数 : なし  
 作成日 : 1995年10月23日  
 戻り値 : 地盤の単位面積当たりの粘着力[ $\text{Kg}/\text{m}^2$ ]  
 補足 :  
 起動環境 :  
 注意 :

.....\*/

double CLSbull::GetAdhesion(void)

```
{
    return (dbAdhesion);
}
```

/\*.....

機能 : 換算係数をセットする  
 引数 : なし  
 作成日 : 1995年10月23日  
 戻り値 : なし  
 補足 :  
 起動環境 :  
 注意 :

.....\*/

void CLSbull::SetNuc(double db)

```
{
    dbNuc = db;
}
```

/\*.....

機能 : 換算係数を取得する  
 引数 : なし  
 作成日 : 1995年10月23日  
 戻り値 : 換算係数  
 補足 :  
 起動環境 :  
 注意 :

.....\*/

double CLSbull::GetNuc(void)

```
{
    return (dbNuc);
}
```

/\*.....

機能 : 地盤の交換係数をセットする  
 引数 : なし  
 作成日 : 1995年11月24日  
 戻り値 : なし  
 補足 :  
 起動環境 :  
 注意 :

.....\*/

void CLSbull::SetZibun1 lenkei(double db)

```
{
    dbZibun1 lenkei = db;
}
```

```

/*-----
機能  : 地盤の変形係数を取得する
引数  : なし
作成日 : 1995年11月24日
戻り値 : 換算係数
補足  :
起切要否
注意  :
-----*/

```

```

double ClShall::GetZibunIkenkei(void)
{
    return (dbZibunIkenkei);
}

```

```

/*=====||End of File||=====*/
0

```

```

/*
プログラム名:  console.c
処理概要   :   ブロードキャストシミュレーター   キーボード、ジョイスティック関係
*/

```

```

#include <stdio.h> //debug
#include <conio.h>
#include <time.h>

#include "console.h"

#define KEY_DATA_PORT 0x00
#define KEY_STATE_PORT 0x04
#define KEY_STATE_TX_FULL 1

#define BiosBuffOffset 0x001c

#define GAMEPORT 0x0201
#define BUTTON1 0x0010
#define BUTTON2 0x0020
#define JSTICKX 0x0001
#define JSTICKY 0x0002

static int keypressed = 0;
static char key_input = 0;
static short* NextBiosChar = (short*)0x41a; //dos port
static short* LastBiosChar = (short*)0x41c; //dos port
static void (__interrupt__ far *OldKeyHandler);

//Keyboard Handler: this is called when an interrupt occurs
static void __interrupt__ far KeyHandler()
{
    if (inp(KEY_STATE_PORT) & KEY_STATE_TX_FULL)
    {
        key_input = (char)inp(KEY_DATA_PORT);
        keypressed = 1;
    }
    (*OldKeyHandler);
    *NextBiosChar = BiosBuffOffset;
    *LastBiosChar = BiosBuffOffset;
}

void Keyboard_Init(void)
{
    OldKeyHandler = _dos_getvect(9);
    _dos_setvect(9, KeyHandler);
};

void Keyboard_close(void)
{
    _dos_setvect(9, OldKeyHandler);
}

int Keyboard_input(void)
{
    if (keypressed)
    {
        keypressed = 0;
        return (0x00ff & key_input);
    }
    else
        return (-1);
}

/*
 * JoyStick
 */
static int jy_max, jy_min, jy_center;

```

```

/*-----
機能      :   ジョイスティックの初期化
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起用環境  :
注意      :
-----*/

```

```

void Joystick_init(void)
{
    int clk;
    int by;
    printf("Yn 操作レックの位置補正を行います。両面表示に従って操作してください Yn");
    printf("Yn ジョイスティックを一番前に倒してボタンを押して下さい Yn");
    b=~inp(GAMEPORT);
    while ((b & BUTTON))
        b=~inp(GAMEPORT);
    outp(GAMEPORT, 0);
    for (jy_min=0; inp(GAMEPORT) & JSTICKY; jy_min++);
    for (clk=clock0; clock0 - clk < 50);

    printf("Yn ジョイスティックを一番後ろに倒してボタンを押して下さい Yn");
    b=~inp(GAMEPORT);
    while ((b & BUTTON))
        b=~inp(GAMEPORT);
    outp(GAMEPORT, 0);
    for (jy_max=0; inp(GAMEPORT) & JSTICKY; jy_max++);
    for (clk=clock0; clock0 - clk < 50);

    printf("Yn ジョイスティックを中央にしてボタンを押して下さい Yn");
    b=~inp(GAMEPORT);
    while ((b & BUTTON))
        b=~inp(GAMEPORT);
    outp(GAMEPORT, 0);
    for (jy_center=0; inp(GAMEPORT) & JSTICKY; jy_center++);
    printf("Yn 操作レック位置の補正が終了しました Yn");
    for (clk=clock0; clock0 - clk < 50);
}

```

```

/*-----
機能      :   ジョイスティックの縦方向の位置の取得
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   ジョイスティックの縦方向の位置
補足      :
起用環境  :
注意      :
-----*/

```

```

double Joystick_Y()
{
    int jy;
    outp(GAMEPORT, 0);
    for (jy=0; inp(GAMEPORT) & JSTICKY; jy++);
    if (jy > jy_center)
        return (jy - jy_center) / double(jy_max - jy_center);
    else
        return (jy_center - jy) / double(jy_min - jy_center);
}

```

```

/*-----[End of File]-----*/

```

```

/*-----
プログラム名: crawler.cc          ブルドーザシミュレーター      クローラと本体の動作
処理概要      :
/*-----

```

```

#include <stdio.h>
#include <graph.h>
#include <math.h>

```

```

#include "magic.h"
#include "etc.h"
#include "crawler.h"

```

```

/*-----
機能      :      クローラの初期化
引数      :      なし
作成日    :      1995年10月23日
戻り値    :      なし
補足      :      ptBody[n].nFunction について 0:点, 1:線, 2:丸
起り/発見
注意      :
/*-----

```

```

CLScrawler::CLScrawler(void)
{
    int nNum=0;          //
    double rad;          // 角度変数
    int i;
    dbGrossLength=IDD_GrossLength;          // グローサの長さ[m]
    dbCrawlerWidth=IDD_CrawlerWidth;        // クローラ幅[m]

    wheel_base=IDD_WheelBase*100;          // length unit is [cm]
    wheel_diameter=IDD_WheelDiameter*100;
    double grossa=dbGrossLength*100;        // グローサの長さ[cm]
    double dbr=5;          // 支柱の幅?
    double r45=sqrt(2.0)/2;
    double t=200;          // 全高[cm]

    ptBody[nNum++].set(0, 0, 0);          // center

// キヤタピラ内部
    for (i=0, rad=90*3.1415/180; rad<=450*3.1416/180; i=1, rad+=IDD_CrawlerPolygon*3.1415/180)
    {
        ptBody[nNum].nFunction=i;
        ptBody[nNum].x=wheel_diameter*cos(rad);
        ptBody[nNum++].y=wheel_diameter+wheel_diameter*sin(rad);
    }
    ptBody[nNum++].set(4,0,0);
    for (i=0, rad=90*3.1415/180; rad<=450*3.1416/180; i=1, rad+=IDD_CrawlerPolygon*3.1415/180)
    {
        ptBody[nNum].nFunction=i;
        ptBody[nNum].x=wheel_base+wheel_diameter*cos(rad);
        ptBody[nNum++].y=wheel_diameter+wheel_diameter*sin(rad);
    }
    ptBody[nNum++].set(4,0,0);
    ptBody[nNum++].set(0, 0, 0);
    ptBody[nNum++].set(1, wheel_base, 0);
    ptBody[nNum++].set(4,0,0);
    ptBody[nNum++].set(0, wheel_base, wheel_diameter*2);
    ptBody[nNum++].set(1, 0, wheel_diameter*2);
    ptBody[nNum++].set(4,0,0);

// キヤタピラ外部
    for (i=0, rad=90*3.1415/180; rad<=270*3.1416/180; i=1, rad+=IDD_CrawlerPolygon*3.1415/180)
    {
        ptBody[nNum].nFunction=i;
        ptBody[nNum].x=(wheel_diameter+grossa)*cos(rad);
        ptBody[nNum++].y=wheel_diameter+(wheel_diameter+grossa)*sin(rad);
    }
    for (i=1, rad=270*3.1415/180; rad<=450*3.1416/180; i=1, rad+=IDD_CrawlerPolygon*3.1415/180)
    {

```

```

        ptBody[nNum].nFunction=i;
        ptBody[nNum].x=wheel_base+(wheel_diameter+gross)*cos(nu0);
        ptBody[nNum++].y=wheel_diameter+(wheel_diameter+gross)*sin(nu0);
    }
    ptBody[nNum++].set(4,0,0);

// キ+クピラ内部の支柱
    ptBody[nNum++].set(0,0-dbr,wheel_diameter-dbr*8);
    ptBody[nNum++].set(1,wheel_base+dbr,wheel_diameter-dbr*8);
    ptBody[nNum++].set(1,wheel_base+dbr,wheel_diameter+dbr);
    ptBody[nNum++].set(1,0-dbr,wheel_diameter+dbr);
//
    ptBody[nNum++].set(1,0-dbr,wheel_diameter-dbr);
    ptBody[nNum++].set(4,0,0);

// 本体上部
    ptBody[nNum++].set(0,(-wheel_diameter-gross)*r45,wheel_diameter+(wheel_diameter+gross)*r45);
    ptBody[nNum++].set(1,(-wheel_diameter-gross)*r45,0);
    ptBody[nNum++].set(1,115,0);
    ptBody[nNum++].set(1,115,t-60);
    ptBody[nNum++].set(1,159,t-60);
    ptBody[nNum++].set(1,159,0);
    ptBody[nNum++].set(1,wheel_base+(wheel_diameter+gross)*r45,0);
    ptBody[nNum++].set(1,wheel_base+(wheel_diameter+gross)*r45,wheel_diameter+(wheel_diameter+gross)*r45);
    ptBody[nNum++].set(4,0,0);
    ptBody[nNum++].set(0,60,0);
    ptBody[nNum++].set(1,60,t+60);          // 支柱
    ptBody[nNum++].set(2,0,0);
    ptBody[nNum++].set(0,60,t+60);
    ptBody[nNum++].set(1,160,t+60);        // 屋根
    ptBody[nNum++].set(2,0,0);

//
    printf("%dYn",nNum);
}

```

```

/*-----
機能      : クローラのデストラクタ
引数      : なし
作成日    : 1995年10月23日
戻り値    : なし
補足      :
起動環境
注意      :
-----*/

```

CLScrawler::~CLScrawler(void)

```

{
}

```

```

/*-----
機能      : クローラの各ノードの座標を表示する
引数      : int nNumber      表示するノードの番号
作成日    : 1995年10月23日
戻り値    : なし
補足      : テスト用の回数
起動環境
注意      :
-----*/

```

void CLScrawler::print(int nNumber)

```

{
    printf("%d %d %dYn",ptBody[nNumber].nFunction,ptBody[nNumber].x,ptBody[nNumber].y);
}

```

```

/*-----
機能      : クローラを描画する
引数      : int cx      本体の横位置[cm]
              int cy      本体の縦位置[cm]
              double dbSin 本体の傾きの sin 値
              double dbCos 本体の傾きの cos 値
作成日    : 1995年10月23日

```

戻り値 : なし  
補足 :  
起り環境  
注意 :

-----\*/

void CLScrawler::Draw(int cx,int cy,double dbSin,double dbCos)

```
{
    int i;
    CLSpoint ptIDraw;
    int nLinePoint=0;
    struct _wxycoord *xyNode=new _wxycoord[MaxCrawlerPoint];

    for (i=1;i<MaxCrawlerPoint;i++)
    {
        ptIDraw=ptBody[i].Change1(cx,cy,dbSin,dbCos);
        switch (ptBody[i].nFunction)
        {
            case 0: // 点 (0,x,y)
            {
                xyNode[0].wx=ptIDraw.x;
                xyNode[0].wy=ptIDraw.y;
                nLinePoint=1;
                break;
            }
            case 1: // 線 (1,x,y)
            {
                xyNode[nLinePoint].wx=ptIDraw.x;
                xyNode[nLinePoint++].wy=ptIDraw.y;
                break;
            }
            case 2:
            {
                _moveTo_w(xyNode[0].wx,xyNode[0].wy);
                _lineTo_w(xyNode[1].wx,xyNode[1].wy);
                break;
            }
            case 4: // 描画
            {
                _polygon_wxy(_GBORDER,nLinePoint,xyNode);
                break;
            }
        }
    }
}
```

-----\*/

機能 : クローラを描画する  
引数 : int cx 本体の横位置[cm]  
int cy 本体の縦位置[cm]  
double dbTheta 本体の傾きの角度[rad]  
作成日 : 1995年10月23日  
戻り値 : なし  
補足 :  
起り環境  
注意 :

-----\*/

void CLScrawler::Draw(int cx,int cy,double dbTheta)

```
{
    int i;
    CLSpoint ptIDraw;
    int nLinePoint=0;
    struct _wxycoord *xyNode=new _wxycoord[MaxCrawlerPoint];

    for (i=1;i<MaxCrawlerPoint;i++)
    {
        ptIDraw=ptBody[i].Change1(cx,cy,dbTheta);
        switch (ptBody[i].nFunction)
        {
            case 0: // 点 (0,x,y)
            {
```

```

        xyNode[0].wx = ptD.mw.x;
        xyNode[0].wy = ptD.mw.y;
        nLinePoint = 1;
        break;
    }
    case 1:                // 線 {1, x, y}
    {
        xyNode[nLinePoint].wx = ptD.mw.x;
        xyNode[nLinePoint++].wy = ptD.mw.y;
        break;
    }
    case 2:
    {
        _moveto_w(xyNode[0].wx, xyNode[0].wy);
        _lineto_w(xyNode[1].wx, xyNode[1].wy);
        break;
    }
    case 4:                // 描画
    {
        _polygon_wxy(GBORDER, nLinePoint, xyNode);
        break;
    }
}
}
}

```

```

/*-----
機能 : クローラの基本位置の座標を取得する
引数 : int cx 本体の横位置[cm]
      int cy 本体の縦位置[cm]
      double dbTheta ブレードの傾きの角度[rad]
作成日 : 1995年10月23日
戻り値 : ブレードの先端の座標
補足 :
起動環境
注意 :
-----*/

```

```

CLSpint.CLSawler::Center(int cx, int cy, double dbTheta)
{
    return (ptBody[0].Change1(cx, cy, dbTheta));
}

```

```

/*-----
機能 : クローラの基本位置の座標を取得する
引数 : int cx 本体の横位置[cm]
      int cy 本体の縦位置[cm]
      double dbSin ブレードの傾きの sin 値
      double dbCos ブレードの傾きの cos 値
作成日 : 1995年10月23日
戻り値 : ブレードの先端の座標
補足 :
起動環境
注意 :
-----*/

```

```

CLSpint.CLSawler::Center(int cx, int cy, double dbSin, double dbCos)
{
    return (ptBody[0].Change1(cx, cy, dbSin, dbCos));
}

```

```

/*-----
機能 : クローラの全長の取得
引数 : なし
作成日 : 1995年10月23日
戻り値 : クローラの全長[m]
補足 :
起動環境
注意 :
-----*/

```

```

double CLSawler::GetWheelBase(void)

```

```
{
    return (wheel_diam);
}
```

```
/*-----
機能 : クローラの半径の取得
引数 : なし
作成日 : 1995年10月23日
戻り値 : クローラの半径[m]
補足 :
起用環境
注意 :
-----*/
```

```
double CLScrewler::GetWheelDiameter(void)
{
    return (wheel_diameter);
}
```

```
/*-----
機能 : グローサの長さの取得
引数 : なし
作成日 : 1995年10月23日
戻り値 : グローサの長さ[m]
補足 :
起用環境
注意 :
-----*/
```

```
double CLScrewler::GetGrossLength(void)
{
    return (lbGrossLength);
}
```

```
/*-----[End of File]-----*/
```

```

*-----
プログラム名:  drnw.cc
処理概要   :   ブルドーザシミュレーター   描画関係
*-----

```

```

#include <stdio.h> //necessary for sprintf()
#include <math.h>

```

```

#include "etch"
#include "crawler.h"
#include "blade.h"
#include "bull.h"
#include "magic.h"
#include "drnw.h"

```

```

*-----
機能      :   描画クラスの初期化
リ数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境  :
注意      :
*-----

```

```

CLSDraw::CLSDraw(void)
{
    int i;

    // this->DrawModeGraphics();

    for (i=0; i<GND_POLYGON; i++)
    {
        curr.gnd[i].wx=i*10+X_OFFSET;
        curr.gnd[i].wy=0;
    }

    curr.gnd[GND_POLYGON-2].wy=Y_BOTTOM;
    curr.gnd[GND_POLYGON-1].wy=Y_BOTTOM;
    curr.gnd[GND_POLYGON-1].wx=0;

    gnd_num=0;
}

```

```

*-----
機能      :   描画クラスのデストラクタ
リ数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境  :
注意      :
*-----

```

```

CLSDraw::~CLSDraw(void)
{
    _clearscreen(GWINDOW);
    _setvideomode(DEFAULTMODE);
}

```

```

*-----
機能      :   グラフィック描画モードを xor にする
リ数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境  :
注意      :
*-----

```

```

void CLSDraw::DrawSetXor(void)
{

```

```

        _setpixmap(_GXOR);
    }

```

```

/*-----
機能      :      描画モードをグラフィックモードにする
リ数      :      なし
作成日    :      1995年10月23日
戻り値    :      なし
補足      :
起り環境 :
注意      :
-----*/

```

```

void CLSdraw::DrawModeGraphics(void)
{
    _setvideomode(_VRES16COLOR);
    _setwindow(1, X_OFFSET, Y_OFFSET, X_COORD, Y_COORD);
    _clearscreen(_GWINDOW);
}

```

```

/*-----
機能      :      描画モードをテキストモードにする
リ数      :      なし
作成日    :      1995年10月23日
戻り値    :      なし
補足      :
起り環境 :
注意      :
-----*/

```

```

void CLSdraw::DrawModeText(void)
{
    _setvideomode(_TEXT80);
}

```

```

/*-----
機能      :      地面描画ポリゴン表の作成をする
リ数      :      int      *ground      現在の地面の表へのポインタ
               int      nGround      残りの地盤の数(cm)
作成日    :      1995年10月23日
戻り値    :      なし
補足      :
起り環境 :
注意      :
-----*/

```

```

void CLSdraw::CalculateGround(int *ground, int nGround)
{
    double dbMaxHeight=0;      // 穴の描画の最も上の高さ 地上 dbMaxHeight[cm]より下を消す
    nGround=nGround/10;
    if(nGround>(GND_POLYGON-2))
        gnd_num=GND_POLYGON-2;
    else
        gnd_num=nGround;

    //update curr.gnd
    for(int i=0; i<gnd_num; i++)
    {
        int j=0;
        curr.gnd[i].wx=i*10+X_OFFSET;
        curr.gnd[i].wy=0;
        for(j=0; j<10; j++)
            curr.gnd[i].wy+=ground[i*10+j];
        curr.gnd[i].wy=curr.gnd[i].wy/10;
        if(dbMaxHeight<curr.gnd[i].wy)
            dbMaxHeight=curr.gnd[i].wy;
    }
    curr.gnd[gnd_num].wx=i*10+X_OFFSET;
    curr.gnd[gnd_num+2].wx=X_COORD;      // 穴部分の右上
    curr.gnd[gnd_num+2].wy=dbMaxHeight;  // 穴部分の右上
    curr.gnd[gnd_num+3].wx=X_OFFSET;      // 穴部分の左上
}

```

```

curr.gnd[gnd_num+3].wy=dbMaxI leight;          // 穴部分の左上
}

/*-----
機能  :  地面剖面ポリゴンを描画する(地面部分)
引数  :  なし
作成日 :  1995年10月23日
戻り値 :  なし
補足  :
起動環境
注意  :
-----*/
void CLSdraw::DrawGround(void)
{
    curr.gnd[gnd_num].wy = Y_BOTTOM;
    curr.gnd[gnd_num+1].wx=X_OFFSET;
    curr.gnd[gnd_num+1].wy=Y_BOTTOM;
    _setcolor(7);
    _polygon_wxy(_GILLINTERIOR, gnd_num+2, curr.gnd);    // 地面部分の左下
}

/*-----
機能  :  地面剖面ポリゴンを描画する(穴部分)
引数  :  なし
作成日 :  1995年10月23日
戻り値 :  なし
補足  :
起動環境
注意  :
-----*/
void CLSdraw::DrawGroundHole(void)
{
    curr.gnd[gnd_num].wy = Y_BOTTOM;
    curr.gnd[gnd_num+1].wx=X_COORD;
    curr.gnd[gnd_num+1].wy=Y_BOTTOM;
    _setcolor(0);
    _polygon_wxy(_GILLINTERIOR, gnd_num+4, curr.gnd);    // 穴部分の右下
}

/*-----
機能  :  0m 基準線を縦横に描画する
引数  :  なし
作成日 :  1995年10月23日
戻り値 :  なし
補足  :
起動環境
注意  :
-----*/
void CLSdraw::DrawReference(void)
{
    _setcolor(3);
    _move_to_w(X_OFFSET, 0);
    _line_to_w(X_COORD+X_OFFSET, 0);
    _move_to_w(0, Y_OFFSET);
    _line_to_w(0, Y_COORD+Y_OFFSET);
}

/*-----
機能  :  掘削目標値を描画する
引数  :  double r      掘削目標値[m]
               int      nindex 掘削目標地点[cm]
作成日 :  1995年10月23日
戻り値 :  なし
補足  :
起動環境
注意  :
-----*/

```

```
void CLSdraw::DrawItef(double r,int nIndex)
```

```
{
    double X_STPOINT=600;
    double X_MVPOINT=200;
    double rr=r*100; //m->cm
    _setcolor(0);
    _moveo_w(X_OFFSET,0);
    _lineo_w(X_STPOINT+X_OFFSET-nIndex,0);
    _ellipse_w(GBORDER,X_STPOINT+X_OFFSET-nIndex+2,0,X_STPOINT+X_OFFSET-nIndex-2,-0);
    _lineo_w(X_MVPOINT+X_STPOINT+X_OFFSET-nIndex,rr);
    _lineo_w(X_COORD+X_OFFSET,rr);
}
```

```
/*-----
機能 : 地盤 1m 毎の目盛りを描画する
引数 : int nIndex;
作成日 : 1995年11月01日
戻り値 : なし
補足 :
起動環境
注意 :
```

```
-----*/
```

```
void CLSdraw::DrawScale(int nIndex)
```

```
{
    char szStr[MAXSTRING];
    _setcolor(3);
    for (int i=nGndIndex+X_OFFSET;i<nGndIndex+X_COORD;i++)
    {
        if((i%100))
        {
            _moveo_w(i-nGndIndex,Y_BOTTOM);
            _lineo_w(i-nGndIndex,Y_BOTTOM+10);
            sprintf(szStr,"%03d",i/100);
            _moveo_w(i-nGndIndex+3,Y_BOTTOM+10);
            _outgtext(szStr);
        }
    }
}
```

```
/*-----
機能 : 起動画面(メインメニュー)の描画
引数 : なし
作成日 : 1995年10月23日
戻り値 : なし
補足 :
起動環境
注意 :
```

```
-----*/
```

```
void CLSdraw::DrawMenu(void)
```

```
{
    printf("\n\n\n\n");
    printf("***** ブルドザ オペレーション シミュレータ*****\n");
    printf("----- 起 動 画 面 ----- \n");
    printf("\n\n 以下の1-3を選択してキー入力して下さい\n");
    printf("1: シミュレーションを開始します\n");
    printf("2: 設定画面(初期設定を変更したい場合に選択してください)\n");
    printf("3: プログラムを終了します\n");
}
```

```
/*-----
機能 : ギアおよび金切終了のメニューの描画
引数 : なし
作成日 : 1995年10月23日
戻り値 : なし
補足 :
起動環境
注意 :
```

```
-----*/
```

```

void CLSdraw::DrawMenu2(void)
{
    char str[MAXSTRING];
    sprintf(str, "[<]=[Gear Down] [>]=[Gear Up] [G]=[END]");
    _settextposition(34, 0);
    _outtext(str);
}

```

```

/*-----
機能      :      ブルドーザのステータスの描画
引数      :      double      time      現在の時間(シミュレーション開始時から)(sec)
               CLSBullStatus *st      ブルドーザのステータスクラスへのポインタ
               int      dat      出力地形データの大きさ(line)
作成日    :      1995年10月23日
戻り値    :      なし
補足      :
起動環境  :
注意      :
-----*/

```

```

void CLSdraw::DrawStatus(double time, CLSBullStatus *st, int dat)
{
    char s[MAXSTRING];

    sprintf(s, "Time      %6.2f sec", time);
    _settextposition(1, 25);
    _outtext(s);

    sprintf(s, "Velocity %6.2f [m/sec] Gear %d", st->velocity, st->gear);
    _settextposition(2, 25);
    _outtext(s);

    sprintf(s, "Bulldozer (%6.2f [m], %6.2f [m], %6.2f [deg])", st->bull_X, st->bull_Y, st->bull_TH * 180./3.14);
    _settextposition(3, 25);
    _outtext(s);

    sprintf(s, "Lever Pos %6.2f [mm] ", st->lever_pos);
    _settextposition(4, 25);
    _outtext(s);

    sprintf(s, "Cylinder Length %6.2f [mm]", st->cylinder_len);
    _settextposition(5, 25);
    _outtext(s);

    sprintf(s, "Blade      (%6.2f [m], %6.2f [m])", st->blade_X, st->blade_Y);
    _settextposition(6, 25);
    _outtext(s);

    sprintf(s, "Data Size  %d", dat);
    _settextposition(7, 25);
    _outtext(s);
}

```

```

/*-----
機能      :      シミュレーションの開始までの時間の描画
引数      :      double  dblTime 時間(sec)
作成日    :      1995年10月23日
戻り値    :      なし
補足      :
起動環境  :
注意      :
-----*/

```

```

void CLSdraw::DrawCountDown(double dblTime)
{
    char sStr[MAXSTRING];
    sprintf(sStr, "Count Down %5.2f", dblTime);
    _settextposition(34, 0);
    _outtext(sStr);
}

```

-----[End of File]-----

```

/*-----
プログラム名   enviro.nce
処理概要       :   ブルドーザシミュレーター   ブルドーザの付帯部分の動作
-----*/

```

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

```

```

#include "etc.h"
#include "crawler.h"
#include "blade.h"
#include "console.h"
#include "enviro.n.h"
#include "magic.h"

```

```

#include "process.h"

```

```

/*-----
機能       :   ブルドーザの付帯部分の初期化
引数       :   なし
作成日     :   1995年10月23日
戻り値     :   なし
補足       :
起動環境   :
注意       :
-----*/

```

```

CLSEnviro.n::CLSEnviro.n(void)
{
    param.time_interval = IDD_DeltaT;           // シミュレーション処理間隔(sec)
    param.field_length = IDD_DefaultFieldLength; // 地盤の長さ[m]
    param.reference = IDD_DefaultReference;      // 掘削目標高さ[m]

    bulldozer = new CLSbulldozer;
    simtime = new CLSsime;

    ground = new int[(int)param.field_length * 100]; // 1cm step

    Enviro.nReset(); // ブルドーザの付帯部分のリセット
}

```

```

/*-----
機能       :   ブルドーザの付帯部分のデストラクタ
引数       :   なし
作成日     :   1995年10月23日
戻り値     :   なし
補足       :
起動環境   :
注意       :
-----*/

```

```

CLSEnviro.n::~CLSEnviro.n(void)
{
    delete bulldozer;
}

```

```

/*-----
機能       :   ブルドーザの付帯部分のリセット
引数       :   なし
作成日     :   1995年10月23日
戻り値     :   なし
補足       :   クラス変数の初期化
起動環境   :
注意       :
-----*/

```

```

void CLSEnviro.n::Enviro.nReset(void)
{
    simtime->SimeReset();
}

```

```

    gnd_index=0;
}

/*-----
機能      :   ブルドーザのシミュレータ
引数      :   char *i_gnd      入力地盤データファイル名
              char *o_gnd      出力地盤データファイル名
              char *o_bul      出力シミュレーション結果データファイル名
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起用環境  :
注意      :
-----*/
void CLSenviro::EnviroSimu(char *i_gnd,char *o_gnd,char *o_bul)
{
    CLSBullStatus *bstat;                // ブルドーザの挙動データ
    int clk;                             // 時刻計算の回数
    int datasize;                         // ブルドーザの挙動データのサイズ
    int wait_time;

// 地盤データの初期化
{
    CLSGround_init *gnd_init;            // 地盤データ一次読み込みバッファへのポインタ
    gnd_init=new CLSGround_init(int((param.fekl_length+1)*10));
    FILE *ingnd_fp;                      // 入力地盤データファイルのファイルポインタ
    double gx;                           // 地盤の位置X[cm]
    double gy;                           // 地盤の高さ[cm]
    int i;
    int n=param.fekl_length*100;
    if((ingnd_fp=fopen(i_gnd,"r"))==NULL)
        exit(EXIT_FAILURE);
    for(i=0;i<n/10;i++)                  // ファイルから地盤データを読み込む
    {
        fscanf(ingnd_fp,"%lf%lf\n",&gx,&gy);
        gnd_init[i].nValue=gy*100; // meter -> centimeter
    }
    fclose(ingnd_fp);

    for(i=0;i<n;i++)                     // 地盤データを地盤配列に設定する
        ground[i]=gnd_init[i/10].nValue/10+(gnd_init[(i/10)+1].nValue-gnd_init[i/10].nValue)*(i%10)/10;
    delete [] gnd_init;                  // 地盤データ一次読み込みバッファの破棄
}

// 各種パラメータの初期化
wait_time=param.time_interval*100;      // シミュレーション処理間隔[msec]
datasize=param.fekl_length/param.time_interval/IDD_HeikinVelocity; //IDD_HeikinVelocity[m/sec] で走行した場合に指定距離を全て記録するための数
bstat=new CLSBullStatus(datasize);        // ブルドーザ挙動データのバッファの作成

// Init Bulkcozer
// CLSdraw::DrawSetXor();
CLSdraw::DrawModeGraphics();
bulkcozer->BullReset();
gnd_index += bulkcozer->BullMove(param.time_interval, &(ground[0]))*100;
{
    int i;
    for(i=0;i<int(5/param.time_interval);i++) // wait 10seconds
    {
        clk=clk();
        CLSdraw::DrawCountDown(double(5/param.time_interval-i)/10); // シミュレーションの開始までの時間の描画
        while((clk()-clk)<param.time_interval*100)
        {
            bulkcozer->BullBladeMove(param.time_interval); // ブレードの計算処理
            CLSdraw::CaluculateGround(&(ground[0]),(int)param.fekl_length*100); // 地盤の計算処理
            CLSdraw::DrawGroundHole(); // 地盤の描画(穴)
            bulkcozer->BullDraw(); // ブルドーザの描画
            CLSdraw::DrawGround(); // 地盤の描画
            CLSdraw::DrawReference(); // 0m 基準線を縦線に描画する
        }
    }
}

```

```

        CLSdraw::DrawRef(param.reference, 0);
        CLSdraw::DrawScale(0);
    }
}

//Start Bulldozer

CLSdraw::DrawMenu20;
bulldozer->BullUnlockGear0;
bulldozer->BullSetGear(1);

//start

for (int idx = 0; idx < datasize; idx++)
{
    clk = clock();
    if (!EnvironKeyboard())
        break;

    simtime->STimeAdd(param.time_interval);

    bulldozer->BullBladeMove(param.time_interval);
    gnd_index += bulldozer->BullMove(param.time_interval, &(ground[gnd_index]) * 100;
    int n;
    n = (int)param.field_length * 100 - gnd_index;
    CLSdraw::CalculateGround(&(ground[gnd_index]), n);
    CLSdraw::DrawGroundHole0;
    bulldozer->BullDraw0;
    CLSdraw::DrawGround0;
    CLSdraw::DrawReference0;
    CLSdraw::DrawRef(param.reference, gnd_index);
    CLSdraw::DrawScale(gnd_index);
    if (n < 600)
    {
        bulldozer->BullSetGear(0);
        bulldozer->BullLockGear0;
    }
    bulldozer->BullGetStatus(&(bstat[idx]));

    bstat[idx].bull_X += gnd_index / 100;
    bstat[idx].blade_X += gnd_index / 100;
    CLSdraw::DrawStatus(simtime->STimeGet0, &(bstat[idx]), gnd_index);
    while ((clock() - clk) < wait_time); // 1 ticks = 10 msec wait
}

// シミュレーション結果のファイルへの書き込み
{
    FILE *ognd_fp;
    FILE *obull_fp;
    int n = param.field_length * 100;
    int i;
    if ((ognd_fp = fopen(o_gnd, "w")) == NULL)
        exit(EXIT_FAILURE);
    for (i = 0; i < n; i++)
        fprintf(ognd_fp, "%f%%f\n", i / 100, ground[i] / 100);
    fclose(ognd_fp);

    if ((obull_fp = fopen(o_bull, "w")) == NULL)
        exit(EXIT_FAILURE);
    fprintf(obull_fp, "Bull X,Y [m] Velocity [m/sec] Lever [mm] Syllinder [mm] Blade X,Y [m] Gear Yn");
    for (i = 0; i < datasize; i++)
    {
        fprintf(obull_fp, "%f%%f%%f%%f%%f%%f%%f%%f%%f%%f\n",
            bstat[i].bull_X,
            bstat[i].bull_Y,
            bstat[i].bull_TH,
            bstat[i].velocity,
            bstat[i].lever_pos,
            bstat[i].syllinder_keng,
            bstat[i].blade_X,
            bstat[i].blade_Y,
        );
    }
}

```

```

        bstat[i]gear);
    }
    fclose(obull_f);
}
delete[] bstat;
}

/*-----
機能      :      掘削目標値の取得
引数      :      なし
作成日    :      1995年10月23日
戻り値    :      なし
補足      :
起動環境
注意      :
-----*/
double CLSenviro::EnvironGetParamRef(void)
{
    return param.reference;
}

/*-----
機能      :      掘削目標値のセット
引数      :      double r      掘削目標値
作成日    :      1995年10月23日
戻り値    :      なし
補足      :
起動環境
注意      :
-----*/
void CLSenviro::EnvironSetParamRef(double r)
{
    param.reference = r;
}

/*-----
機能      :      シミュレーション中のキーボード処理
引数      :      なし
作成日    :      1995年10月23日
戻り値    :      1-通常処理終了, 0-シミュレーション終了せよ
補足      :
起動環境
注意      :
-----*/
int CLSenviro::EnvironKeyboard(void)
{
    if (kbhit())          // Check a key input and set flags
    {
        switch (getch())
        {
            case '!':
            case '<':
                bulldozer->BullChangeGear(-1);    // 変速ギアダウン
                break;
            case '!':
            case '>':
                bulldozer->BullChangeGear(1);      // 変速ギアアップ
                break;
            case 'G':
            case 'Yx1b':
                return (0);                        // シミュレーション終了
        }
    }
    return (1);
}

/*-----[End of file]-----*/

```

```

/*
    プログラム名  :  cfcce
    処理概要    :  ブルドーザシミュレーター  描画ノード等 (独自のクラスライブラリ)
*/

```

```

#include <stdio.h>
#include <math.h>

```

```

#include "etc.h"

```

```

/*
    機能      :  描画ノードの初期化
    引数      :  なし
    作成日    :  1995年10月23日
    戻り値    :  なし
    補足      :
    起動環境  :
    注意      :
*/

```

```

CLSpoin::CLSpoin(void)

```

```

{
    nFunction=0,
    x=0,
    y=0,
}

```

```

/*
    機能      :  描画ノードのデストラクタ
    引数      :  なし
    作成日    :  1995年10月23日
    戻り値    :  なし
    補足      :
    起動環境  :
    注意      :
*/

```

```

CLSpoin::~CLSpoin(void)

```

```

{
}

```

```

/*
    機能      :  描画ノードの値セット
    引数      :  double nFunction  0点,1線,2丸
                  double x        横位置
                  double y        縦位置
    作成日    :  1995年10月23日
    戻り値    :  セットされた値
    補足      :  点,線,丸 くわしくは CLSblade::Draw を参照のこと
    起動環境  :
    注意      :
*/

```

```

CLSpoin CLSpoin::set(int nFunction,double x,double y)

```

```

{
    this->nFunction = nFunction;
    this->x = x;
    this->y = y;
    return (*this);
}

```

```

/*
    機能      :  描画ノードを一次変換で回転させます
    引数      :  double cx        横方向へのオフセット
                  double cy        縦方向へのオフセット
                  double dltTheta[回転角度rad]
    作成日    :  1995年10月23日
    戻り値    :  なし
    補足      :  回転アルゴリズムには一次変換を使用しています
    起動環境  :
*/

```

注意 :

```
.....*/
CLSpint.CLSpint::Change1(double cx, double cy, double dl1Theta)
{
    double dlSin=sin(dl1Theta);
    double dlCos=cos(dl1Theta);
    return (Change1(cx, cy, dlSin, dlCos));
}
```

```
/*.....
機能 : 描画ノードを一変換で回転させます
引数 : double cx 横方向へのオフセット
       double cy 縦方向へのオフセット
       double dlSin 回転角の sin 値
       double dlCos 回転角の cos 値
作成日 : 1995年10月23日
戻り値 : なし
補足 : 回転アルゴリズムには一変換を使用しています
起期開発
注意 :
```

```
.....*/
CLSpint.CLSpint::Change1(double cx, double cy, double dlSin, double dlCos)
{
    CLSpint pt;
    pt.nFunction=this->nFunction;
    pt.x=cx+this->x*dlCos-this->y*dlSin;
    pt.y=cy+this->x*dlSin+this->y*dlCos;
    return (pt);
}
```

```
/*.....
機能 : ノードの座標を表示する
引数 : なし
作成日 : 1995年10月23日
戻り値 : なし
補足 : テスト用の関数
起期開発
注意 :
```

```
.....*/
void CLSpint::print(void)
{
    printf("%d %d %d\n", this->nFunction, this->x, this->y);
}
```

```
/*.....
機能 : ノード間の差を求める
引数 : CLSpint pt 引くノード
作成日 : 1995年10月23日
戻り値 : ノード間の差
補足 :
起期開発
注意 :
```

```
.....*/
CLSpint.CLSpint::operator-(CLSpint pt)
{
    pt.nFunction=this->nFunction;
    pt.x=this->x-pt.x;
    pt.y=this->y-pt.y;
    return (pt);
}
```

===== [End of File] =====

```

/*-----*/
プログラム名  simulat.cc
処理概要      :   ブルドーザシミュレーター   シミュレータ起動, パラメータ設定
/*-----*/

```

```

#include <stdio.h>
#include <string.h>

#include "magic.h"
#include "etc.h"
#include "crawler.h"
#include "blade.h"
#include "simulat.h"
#include "console.h"

```

```

/*-----*/
機能      :   シミュレータ起動, パラメータ設定の初期化
リ数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :   キーボード入力により 0.シミュレーション起動, 1.パラメータ設定, 2.シミュレータ終了
起動環境  :
注意      :
/*-----*/

```

```

CLSsim::CLSsim(void)
{
    strcpy(o_bull_file, IDS_BullDataDefaultFileName);
    strcpy(i_gnd_file , IDS_InputGroundDefaultFileName);
    strcpy(o_gnd_file , IDS_OutputGroundDefaultFileName);
    DrawMenu();
    for (;;)
    {
        while (kbhit() == 0);
        switch (getch())
        {
            case '1':
                CLSenviro::EnvironReset();
                CLSenviro::EnvironSimu(i_gnd_file, o_gnd_file, o_bull_file);
                DrawMenu();
                break;
                //Simulation

            case '2':
                this->SimConfig();
                DrawMenu();
                break;
                //Configuration

            case '3':
                return;
                //End
        }
    }
}

```

```

/*-----*/
機能      :   シミュレータ起動, パラメータ設定のデストラクタ
リ数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境  :
注意      :
/*-----*/

```

```

CLSsim::~CLSsim(void)
{
}

```

```

/*-----*/
機能      :   パラメータ設定
リ数      :   なし
作成日    :   1995年10月23日

```

戻り値 : なし  
 補足 : ファイル名は 128 文字まで  
 起動環境  
 注意 :

```
void CLSim::SimConfig(vokl)
```

```
{
    CLSenviro::DmwmockText();
    printf("***** ブルドザ オペレーション シミュレータ *****Yn");
    printf("..... 初期設定画面 ..... Yn");
    printf("設定値を変更する場合,その項目に該当するキーを押してYn");
    printf("数値を入力してください(ESC キーで起動画面に戻ります)Yn");
    printf("*****YnYn");
    printf("R: 掘削目標値 (%10.3lf[m] )Yn", CLSenviro::EnvironGetParamRef());
    printf("U: 牽力係数 (%10.3lf )Yn", bulldozer->GetKeninKeisuu());
    printf("T: 地盤のせん断強度 (%10.3lf[Kg/m^2])Yn", bulldozer->GetTauMax());
    printf("S: 地盤の滑り角 (%10.3lf[度] )Yn", bulldozer->GetGroundTheta());
    printf("K: 牽力比滑り率 材料式の次数 (%10.1f )Yn", bulldozer->Get(K));
    printf("M: 移動平均処理の移動データ点数 (%10.1 )Yn", bulldozer->Get(M));
    printf("N: 移動平均処理の平均データ点数 (%10.1 )Yn", bulldozer->Get(N));
    printf("A: 走行抵抗力 (%10.3lf[Kgf] )Yn", bulldozer->GetRunResistance());
    printf("B: 地盤の単位面積当たりの粘着力 (%10.3lf[Kg/m^2])Yn", bulldozer->GetAdhesion());
    printf("C: 地盤の内部摩擦角 (%10.3lf[度] )Yn", bulldozer->GetAlpha());
    printf("D: 換算係数 (%10.3lf )Yn", bulldozer->GetNuc());
    printf("E: 地盤の変形係数 (%1f )Yn", bulldozer->GetZibanlenkei());
    printf("F: file name of bulldozer data (%-14s )Yn", o_bull_file);
    printf("I: file name of the input ground (%-14s )Yn", i_gnd_file);
    printf("O: file name of the output ground (%-14s )Yn", o_gnd_file);
    printf("ESC: Exit This MenuYnYn");

    for (;)
        switch (getch())
        {
            case 'x' || b:
                return;
            case 'r':
            case 'R':
                {
                    double r;
                    printf("Input Reference: [m] ");
                    scanf("%lf", &r);
                    CLSenviro::EnvironSetParamRef(r);
                    printf("Reference is set to %10.3lf [m]YnYn", CLSenviro::EnvironGetParamRef());
                    break;
                }
            case 'u':
            case 'U':
                {
                    double dl;
                    printf("Input Kenin Keisuu: ");
                    scanf("%lf", &dl);
                    bulldozer->SetKeninKeisuu(dl);
                    printf("Kenin Keisuu is set to %10.3lfYnYn", bulldozer->GetKeninKeisuu());
                    break;
                }
            case 't':
            case 'T':
                {
                    double dl;
                    printf("Input TauMax: ");
                    scanf("%lf", &dl);
                    bulldozer->SetTauMax(dl);
                    printf("TauMax is set to %10.3lfYnYn", bulldozer->GetTauMax());
                    break;
                }
            case 's':
            case 'S':
                {
                    double dl;
                    printf("Input Ground Theta: ");
                    scanf("%lf", &dl);
                }
        }
}
```

```

        bulldozer->SetGroundItheta(11);
        printf("GroundItheta is set to %10.3lf YnYn", bulldozer->GetGroundItheta());
        break;
    }
case 'k':
case 'K':
    {
        double db;
        printf("Input K: ");
        scanf("%lf", &db);
        bulldozer->SetK(db);
        printf("K is set to %10.3lf YnYn", bulldozer->GetK());
        break;
    }
case 'm':
case 'M':
    {
        int n;
        printf("Input M: ");
        scanf("%d", &n);
        bulldozer->SetM(n);
        printf("M is set to %10d YnYn", bulldozer->GetM());
        break;
    }
case 'n':
case 'N':
    {
        int n;
        printf("Input N: ");
        scanf("%d", &n);
        bulldozer->SetN(n);
        printf("N is set to %10d YnYn", bulldozer->GetN());
        break;
    }

case 'a':
case 'A':
    {
        double db;
        printf("Input RunResistance: ");
        scanf("%lf", &db);
        bulldozer->SetRunResistance(db);
        printf("RunResistance is set to %10lf YnYn", bulldozer->GetRunResistance());
        break;
    }
case 'v':
case 'B':
    {
        double db;
        printf("Input Adhesion: ");
        scanf("%lf", &db);
        bulldozer->SetAdhesion(db);
        printf("Adhesion is set to %10lf YnYn", bulldozer->GetAdhesion());
        break;
    }
case 't':
case 'C':
    {
        double db;
        printf("Input Alpha: ");
        scanf("%lf", &db);
        bulldozer->SetAlpha(db);
        printf("Alpha is set to %10lf YnYn", bulldozer->GetAlpha());
        break;
    }
case 'd':
case 'D':
    {
        double db;
        printf("Input Nuc: ");
        scanf("%lf", &db);

```

```

        bulkcozer->SetNuc(11);
        printf("Nuc is set to %10lf YnYn", bulkcozer->GetNuc());
        break;
    }
case 'c':
case 'C':
    {
        double dl;
        printf("Input Zibani lenkei:");
        scanf("%lf",&dl);
        bulkcozer->SetZibani lenkei(dl);
        printf("Zibani lenkei is set to %lf YnYn", bulkcozer->GetZibani lenkei());
        break;
    }

case 'f':
case 'F':
    {
        if(o_bull_file[0]!='\0')
            printf("Current file name: %sYn", o_bull_file);
        printf("Output file of bulkcozer data:");
        scanf("%s", o_bull_file);
        printf("Input file of bulkcozer data is set to %sYnYn", o_bull_file);
        break;
    }

case 'g':
case 'G':
    {
        if(i_gnd_file[0]!='\0')
            printf("Current file name: %sYn", i_gnd_file);
        printf("Input file of ground data:");
        scanf("%s", i_gnd_file);
        printf("Input file of ground data is set to %sYnYn", i_gnd_file);
        break;
    }

case 'O':
case 'o':
    {
        if(o_gnd_file[0]!='\0')
            printf("Current file name: %sYn", o_gnd_file);
        printf("Output file of ground data:");
        scanf("%s", o_gnd_file);
        printf("Output file of ground data is set to %sYnYn", o_gnd_file);
        break;
    }
}
}

```

```

/*-----*/
機能      :   メニューの描画
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境  :
注意      :
/*-----*/

```

```

void CLSim::DrawMenu(void)
{
    CLSenviro::DrawModeText();
    CLSenviro::DrawMenu();
}

```

```

/*-----[End of File]-----*/

```

```

/*=====
プログラム名:  simece
処理概要      :   ブルドーザシミュレーター      時間管理
=====*/

```

```
#include "sime.h"
```

```

/*=====
機能      :   時間の初期化
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境
注意      :
=====*/

```

```
CLSstime::CLSstime(void)
```

```

{
    sim_time=0.0; //sec
}

```

```

/*=====
機能      :   時間の設定
引数      :   double init_t
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境
注意      :
=====*/

```

```
CLSstime::CLSstime(double init_t)
```

```

{
    sim_time=init_t; //sec
}

```

```

/*=====
機能      :   時間管理のデストラクタ
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :
起動環境
注意      :
=====*/

```

```
CLSstime::~CLSstime(void)
```

```

{
    //nothing to do
}

```

```

/*=====
機能      :   時間の初期化
引数      :   なし
作成日    :   1995年10月23日
戻り値    :   なし
補足      :   単位(sec)
起動環境
注意      :
=====*/

```

```
void CLSstime::STimeReset(void)
```

```

{
    sim_time=0.0;
}

```

```

/*=====
機能      :   時間を足す
引数      :   double d      経過時間(sec)

```

作成日 : 1995年10月23日  
戻り値 : なし  
補足 :  
起動環境 :  
注意 :

.....\*/  
void CJSstime:STimeAdd(double d)  
{  
 sim\_time += d;  
}

/\*.....  
機能 : 時間を取得する  
引数 : なし  
作成日 : 1995年10月23日  
戻り値 : 経過時間[sec]  
補足 :  
起動環境 :  
注意 :

.....\*/  
double CJSstime:STimeGet(void)  
{  
 return sim\_time;  
}

/\*===== [End of File] =====\*/

```
#include <stdio.h>
#include <graph.h>
#include <math.h>
```

```
#include "magic.h"
#include "etc.h"
#include "crawler.h"
```

```
void main()
{
    CLSrawler *cls;
    cls = new CLSrawler;
}
```

```

/*=====
プログラム名:  main.cc
処理概要   :   ブルドーザシミュレーター   メイン関数
*/=====

```

```
#include <stdio.h>
```

```
#include "etc.h"
```

```
#include "crawler.h"
```

```
#include "block.h"
```

```
#include "console.h"
```

```
#include "simulat.h"
```

```

/*=====
機能   :   ブルドーザシミュレータのメイン関数
引数   :   なし
作成日 :   1995年10月23日
戻り値 :   0正常, その他異常
補足   :
起動環境 :   wmake makefile
注意   :
*/=====

```

```
int main(void)
```

```
{
/*   FILE *fp1; */
```

```
Joystick_init();
```

```
/*   if((fp1=fopen("debug1.kk", "w"))==0) */
/*       return (0); */
```

```
/* Keyboard_init(); */
```

```
CLSSim *sim=new CLSSim;
delete sim;
```

```
/* Keyboard_close(); */
```

```
/*   fclose(fp1); */
/*   return (0);
```

```
}
```

```
/*===== [End of File] =====*/
```

/\*=====

プログラム名 magich  
処理概要 : ブルドーザシミュレータ 各種定数値の定義

/\*=====

```
#define X_COORD (800.0) /* 画面座標のX座標[cm] */
#define Y_COORD (400.0) /* 画面座標のY座標[cm] */
#define X_OFFSET (0.0) /* 画面座標のXオフセット[cm] */
#define Y_OFFSET (-200.0) /* 画面座標のYオフセット[cm] */
#define Y_BOTTOM (Y_OFFSET+80) /* 画面座標のY座標の下端[cm] */

#define IDD_DeltaT (0.10) /* シミュレーション処理時間間隔[sec] */
#define IDD_DefaultReference (-0.1) /* 掘削目標高さ[m] */
#define IDD_DefaultFieldLength (50) /* 地盤の長さ[m] */
#define IDD_HeikinVelocity (2) /* 予想されるブルドーザの平均速度[m/sec] */

#define IDS_BullDataDefaultFileName "bulldata.csv" /* ブルドーザ操作データファイル名 */
#define IDS_InputGroundDefaultFileName "ingnd.csv" /* 入地地形データファイル名 */
#define IDS_OutputGroundDefaultFileName "outgnd.csv" /* 出地地形データファイル名 */

#define IDD_GravityAccel (9.8) /* 重力加速度[m/s^2] */
#define IDD_BullGravityX (1.670) /* ブルドーザの重心の位置 X[m] */
#define IDD_BullGravityY (0.820) /* ブルドーザの重心の位置 Y[m] */
#define IDD_BullWeight (18400) /* ブルドーザの体重[kg] */
#define IDD_KeninKeisuu (0.90) /* 牽引係数 */
#define IDD_TauMax (10000.0) /* 地盤せん断耐力[kg/m^2] */
#define IDD_Alpha (24.0) /* 地盤のせん断摩擦角[度] */
#define IDD_GroundTheta (45.0) /* 地盤の滑り角[度] */
#define IDD_K (4.0) /* 牽引比滑り率特性式の係数 */
#define IDD_WheelBase (3.14) /* 車体の長さ[m] */
#define IDD_WheelDiameter (0.60) /* 車体の高さ[m] */
#define IDD_GrossLength (0.11) /* グローサの長さ[m] */
#define IDD_CrawlerWidth (0.95) /* クローラ幅[m] */
#define IDD_BladeWidth (3.97) /* ブレード幅[m] */
#define IDD_BladeAtsusi (0.05) /* ブレードの厚さ[m] */
#define IDD_SylinderLength (1500.0) /* シリンダの長さの初期値[mm] */
#define IDD_CrawlerPolygon (30.0) /* クローラの多角形の角数[度] */

#define IDD_Gear1V0 (3.0) /* 変速ギア1 V0[Km/h] */
#define IDD_Gear1T0 (33000.0) /* 変速ギア1 T0[Ton] */
#define IDD_Gear2V0 (6.3) /* 変速ギア2 V0[Km/h] */
#define IDD_Gear2T0 (18000.0) /* 変速ギア2 T0[Ton] */
#define IDD_Gear3V0 (10.0) /* 変速ギア3 V0[Km/h] */
#define IDD_Gear3T0 (10500.0) /* 変速ギア3 T0[Ton] */

#define IDN_TsutiHeikin1 (20) /* 土の移動平均平滑化処理の長さ[cm] */
#define IDN_TsutiHeikin2 (3) /* 土の移動平均平滑化処理の点数 */
#define IDD_RunResistance (6000.0) /* 走行抵抗[kg] */
#define IDD_Adhesion (2700.0) /* 地盤の単位面積当たりの粘着力[kg/m^2] */
#define IDD_Nue (0.5) /* 換算係数 */
#define IDD_ZikanHenkei (150000.0) /* 地盤の変形係数 */

#define MAXSTRING (128) /* 通常の auto 文字列変数の大きさ */
```

/\*===== [End of File] =====

```

/*=====
プログラム名  blade.h
処理概要      :   ブルドーザシミュレーター   ブレードの動作のヘッダ
*/

#define MaxBladePoint 18                                     // ブレードのノードの数

/*-----
機能      :   ブレードのクラス定義
作成日    :   1995年10月23日
*/
class CLSblade
{
private:
    double cylinder_leng;                                     // ブレードを動かすシリンダの長さ[mm]
    double lever_position;                                    // JoyStick のレバの位置
    double syl_to_bl_c1, syl_to_bl_c2, syl_to_bl_c3;         // シリンダ関係
protected:
    CLSpoint ptBlade[MaxBladePoint];                         // ブレードの描画データ
    double theta_blade;                                       // ブレードの角度[rad]
    double sin_blade;                                         // ブレードの角度の[sin]
    double cos_blade;                                         // ブレードの角度の[cos]
    double dbBladeWidth;                                       // ブレードの幅[m]
    double dbBladeAtsusa;                                       // ブレードの厚さ[m]
    int bBladeDown;                                           // 1:地面にブレードが滑り込める 0:滑り込めない
public:
    CLSblade(void);
    ~CLSblade(void);
    void CLSblade::Draw(int cx, int cy, double dbCrawlerSin, double dbCrawlerCos, double dbBladeSin, double dbBladeCos);
    void CLSblade::Draw(int cx, int cy, double dbCrawlerTheta, double dbBladeTheta);
    void print(int nNumber);
    CLSpoint CLSblade::GetBladeEdge(int cx, int cy, double dbCrawlerSin, double dbCrawlerCos, double dbBladeSin, double dbBladeCos);
    CLSpoint CLSblade::GetBladeEdge(int cx, int cy, double dbCrawlerTheta, double dbBladeTheta);
    void CLSblade::BullBladeMove(double tim);
    void CLSblade::CalculateBladeTheta(void);
    void CLSblade::ResetSylinderLength(void);
    double CLSblade::GetSylinderLength(void);
    double CLSblade::GetLeverPosition(void);
};

/*=====
[End of File]
*/

```

```

/*-----
プログラム名: console.h
処理概要: ブロード-ザシミュレーター キーボードジョイスティック関係のヘッダ
-----*/

```

```

#include <conio.h>
#include <dos.h>

```

```

// Scan Codes

```

```

// Function keys

```

```

#define KEY_F1 59
#define KEY_F2 60
#define KEY_F3 61
#define KEY_F4 62
#define KEY_F5 63
#define KEY_F6 64
#define KEY_F7 65

```

```

// Number (ten) keys

```

```

#define KEY_HOME : 71
#define KEY_UP 72
#define KEY_PgUP 73
#define KEY_GMINUS 74
#define KEY_LEFT 75
#define KEY_CENTER 76
#define KEY_RIGHT 77
#define KEY_GPLUS 78
#define KEY_END 79
#define KEY_DOWN 80
#define KEY_PgDn 81
#define KEY_INS 82
#define KEY_DEL 83

```

```

void Keyboard_init(void);
void Keyboard_close(void);
int Keyboard_input(void);

```

```

void Joystick_init(void);
double Joystick_Y(void);

```

```

/*-----[End of File]-----*/

```

```

/*=====
プログラム名  clrw.h
処理数値      :      ブルドーザシミュレーター      描画関係のクラス定義
*/=====

```

```

#include <graph.h>
#include "magic.h"
#define GND_POLYGON ((int)X_COORD)/10+2)      /* 画面に表示される地盤の点の総人数 穴部はこれに +2 */

```

```

/*-----
機能      :      描画関係のクラス定義
作成日    :      1995年10月31日
*/-----

```

```

class CLSclrw
{
private:
    struct gnd_data                                /* Shape of ground */
    {
        struct waycoord gnd[GND_POLYGON+2];      // 地盤
        } curr;
        int gnd_num;                                // 計算後の地盤
                                                    // No. of available data
public:
    CLSdraw::CLSdraw(void);
    CLSdraw::~CLSdraw(void);
    void CLSdraw::DrawSetXor(void);
    void CLSdraw::DrawMenu(void);
    void CLSdraw::DrawMenu2(void);
    void CLSdraw::DrawModeGraphics(void);
    void CLSdraw::DrawModeText(void);
    void CLSdraw::DrawCountDown(double dt/Time);
    void CLSdraw::CaluculateGround(int *ground, int nGround);
    void CLSdraw::DrawGround(void);
    void CLSdraw::DrawGroundHole(void);
    void CLSdraw::DrawReference(void);
    void CLSdraw::DrawStatus(double time, CLSBullStatus *st, int n_dat);
    void CLSdraw::DrawRef(double r, int nIndex);
    void CLSdraw::DrawScale(int nGndIndex);
};

```

```

/*===== [End of File] =====*/

```

```

/*=====
プログラム名   environ.h
処理概要       :   ブルドーザシミュレーター       ブルドーザの付帯部分の動作のクラス定義
*/=====

#include "bull.h"
#include "draw.h"
#include "stime.h"

/*=====
機能       :   地盤定数の構造体の定義
作成日     :   1995年10月23日
*/=====
typedef struct tagTypConParams
{
    double time_interval;          /* シミュレーション処理間隔[sec] */
    double field_length;           /* 地盤の長さ[m] */
    double reference;              /* 掘削目標高さ[m] */
}TypConParams;                  /* 地盤定数型 */

/*=====
機能       :   ブルドーザの付帯部分の動作のクラス定義
作成日     :   1995年10月23日
*/=====
class CLEnviron : public CLSdraw
{
private:
    CLSstime * simtime;
    TypConParams param;           /* 地盤定数 */
    int * ground;                 /* 地盤[cm] */
    int gnd_index;                /* 位置[cm] */
protected:
    CLSbull * bulldozer;
public:
    CLEnviron(void);
    ~CLEnviron(void);
    void EnvironSimu(char * i_gnd, char * o_gnd, char * o_bul);
    void EnvironReset(void);
    double EnvironGetParamRef(void);          /* 掘削目標値の取得[m] */
    void EnvironSetParamRef(double r);        /* 掘削目標値のセット[m] */
    int CLEnviron::EnvironKeyboard(void);     /* シミュレーション中のキーボード処理 */
};

/*=====
[End of File]
*/=====

```

```

/*=====
プログラム名: simulat.h
処理概要: ブルドーザシミュレーター シミュレータ起動, パラメータ設定のクラス定義
*/

#include "environ.h"

/*=====
機能: シミュレータ起動, パラメータ設定のクラス定義
作成日: 1995年10月23日
*/
class CLSim: public CLSenvron
{
private:
    char i_gnd_file[128];
    char o_gnd_file[128];
    char o_bull_file[128];
public:
    CLSim(void);
    ~CLSim(void);
    void SimConfig(void);
    void DrawMenu(void);
};

/*===== [End of File] =====*/

```

```

/*=====
プログラム名  stime.h
処理概要      :   ブルドーザシミュレーター   時刻管理のクラス定義
*/=====

```

```

/*-----
機能      :   時刻管理のクラス定義
作成日    :   1995年10月23日
*/-----

```

```

class CLSstime
{
private:
    double sim_time;           // シミュレーション時間[sec]
public:
    CLSstime(double init_t);
    CLSstime(void);
    ~CLSstime();
    void STimeReset(void);      //sec
    void STimeAdd(double t);    //sec
    double STimeGet(void);      //sec
};

```

```

/*===== [End of file] =====*/

```

```

/*
  プログラム名:  cmwkr.h
  処理対象   :   フルドーザシミュレーター   クローラと本体の動作のヘッダ
*/

```

```

#define MaxCrawlerPoint      71                // クローラのノードの数

```

```

/*
  機能   :   クローラのクラス定義
  作成日 :   1995年10月23日
*/

```

```

class CLSrawler
{
protected:
    CLSpoin ptlBody(MaxCrawlerPoint); // クローラと本体の描画データ
    double wheel_Lbase;                // wheel base[m]
    double wheel_diameter;             // wheel diameter[m]
    double dlGrossLength;              // グローサの長さ[m]
    double dlCrawlerWidth;             // クローラ幅[m]
public:
    CLSrawler(void);
    ~CLSrawler(void);
    void Draw(int cx, int cy, double dlSin, double dlCos);
    void Draw(int cx, int cy, double dlItheta);
    void print(int nNumber);
    CLSpoin Center(int cx, int cy, double dlSin, double dlCos);
    CLSpoin Center(int cx, int cy, double dlItheta);
    double GetWheelBase(void);
    double GetWheelDiameter(void);
    double GetGrossLength(void);
};

```

```

/*-----[End of File]-----*/

```