

分置

炉心一プレナム相互作用に関する水流動試験(第2報)

—潜り込み深さの支配パラメータ評価—

1994年9月

動力炉・核燃料開発事業団
大洗工学センター

複製又はこの資料の入手については、下記にお問い合わせください。

〒311-13 茨城県東茨城郡大洗町成田町4002

動力炉・核燃料開発事業団

大洗工学センター システム開発推進部・技術管理室

Enquires about copyright and reproduction should be addressed to: Technology Management Section O-arai Engineering Center, Power Reactor and Nuclear Fuel Development Corporation 4002 Narita-cho, O-arai-machi, Higashi-Ibaraki, Ibaraki-ken, 311-13, Japan

動力炉・核燃料開発事業団 (Power Reactor and Nuclear Fuel Development Corporation)

炉心ープレナム相互作用に関する水流動試験（第2報）
－潜り込み深さの支配パラメータ評価－

小林 順*、大島宏之*、上出英樹*、家田芳明*

要旨

高速炉の崩壊熱除去系に浸漬型補助冷却器(Direct Reactor Auxiliary Cooling System : DRACS)を用いる場合、ホットプレナム内に浸漬された熱交換器によって冷却された冷却材がプランケット燃料集合体の出口部に到達し、集合体内に潜り込むことがある。この現象を炉心ープレナム相互作用と呼ぶが、炉心集合体の温度分布や本来の冷却材流れに影響を与える可能性があるため、炉心の冷却性という観点からもその解明が重要である。本研究の目的は、水流動試験によってこの炉心ープレナム相互作用現象を把握するとともに支配パラメータを導出することである。。

試験部は、ホットプレナム、熱交換器、炉心集合体の低出力部を模擬した、平板上プレナム、冷却壁及び垂直にプレナムに接続する矩形チャンネルからなる。矩形チャンネル内に発生する潜り込み現象は、超音波流速分布測定装置(Ultrasonic Velocity Profile monitor : UVP)¹⁾ およびT-タイプ熱電対を使用した流速場温度場同時計測によって把握した。

実験パラメータは入口流体温度及び流量である。試験結果及び考察より以下の結果を得た。

- (1) チャンネル内温度プロファイルの時間変化は流速のプロファイルのそれとよく一致した。これより温度データによって求めた潜り込み深さは流速データによって求めたものと一致することを確認した。
- (2) 潜り込み深さを決定する因子は浮力、慣性力、熱伝導及びエントレインメント効果と考えられる。次元解析によって潜り込み深さの支配パラメータとして Gr/Re^3 及び Pr 数を導出した。すなわち、潜り込み深さ z は以下のように表現できる。

$$z = f(Gr/Re^3, Pr)$$

本試験体系での潜り込み深さは以下のように整理できる（但し、 Pr 数は一定）。

$$z [m] = 220(Gr/Re^3)$$

*大洗工学センター 安全工学部 原子炉工学室

Core-Plenum Interaction Experiments

- Investigation on Onset Condition and Depth of Penetration Flow -

J. Kobayashi*, H. Ohshima*, H. Kamide*, Y. Ieda*

Abstract

In the case that the Direct Reactor Auxiliary Cooling System (DRACS) is adapted as a decay heat removal system of FBR, cold fluid coming from the heat exchangers immersed directly in the hot plenum may penetrate into some subassemblies under a certain natural circulation condition. This phenomenon is important from the viewpoint of the coolability of the core, because it affects the temperature distribution in the subassemblies and the natural circulation head in a primary cooling system. The objective of this study is to obtain the fundamental understanding of the penetration phenomenon with water experiments.

The test section is composed of a plenum, a cooled wall and a vertical channel, which simulate the hot plenum, the heat exchanger and lower power subassembly, respectively. The inlet temperature of water is kept constant during a run of experiments.

The velocity and temperature fields in the vertical channel are measured simultaneously with two instruments: Ultrasonic Velocity Profile monitor (UVP)¹⁾ for an axial profile of the vertical velocity components and T-type (copper-constantan) thermocouples for the temperature fields.

Inlet temperature and flow rate are selected as experimental parameters. Results of the tests are follows:

- (1) The transient axial profiles of velocity in the vertical channel were compared with those of temperature under the condition that the penetration flow was observed. The trend of temperature profiles agreed well with that of velocity profiles. It was, therefore, considered that the depth of the penetration flow could be estimated with the temperature data as well as the velocity data.

(2) The penetration depth is determined based on buoyancy, inertia, conduction and entrainment effects. With dimensional analysis, Gr/Re^3 and Pr number were derived as relevant parameters for the depth of penetration flow, z :

$$z = f(\text{Gr}/\text{Re}^3, \text{Pr}).$$

In our experiments, the depth of penetration flow could be correlated with Gr/Re^3 as follows:

$$z [\text{m}] = 220(\text{Gr}/\text{Re}^3).$$

*Reactor Engineering Section, Safety Engineering Division,
O-arai Engineering Center, PNC

目次

1. まえがき	1
2. 試験装置	2
2. 1 試験ループ	2
(1) 1次系	2
(2) 冷却系	2
2. 2 試験部	3
(1) 試験部形状	3
(2) 計測部	3
2. 3 計測システム	3
(1) 温度分布測定	3
(2) 流速分布測定	4
3. 実験方法	5
3. 1 試験パラメータと試験ケース	5
3. 2 チャンネル内温度分布測定	5
3. 3 チャンネル内流速分布測定	5
4. 実験結果	6
4. 1 温度分布測定結果	6
4. 2 流速分布測定結果	6
5. 考察	8
6. まとめ	9
7. 謝辞	10
8. 参考文献	11

付録

A 1. 超音波流速分布測定装置計測パラメータ	33
A 2. 超音波流速分布測定装置の性能	33
A 3. 超音波流速分布測定装置データファイルの内部構造	34
A 4. 超音波流速分布測定装置データ処理ツール	35
A 5. 潜り込み深さの支配パラメータ導出に至る経緯	36
(1) 地表付近の空気流れとのアナロジによる考察	36
(2) 流速プロファイルの時間変化から潜り込み流れの先端を探索する方法	37
(3) パワースペクトルによる現象把握	38
A 6. 水流动第2試験室・中央制御室の空調設備に関する問題点及び改善措置	50
A 7. UDP. EXEソースリスト	51

List of Tables

Table 1 Test Parameter Matrix of Report 1
Table 2 Test Parameter Matrix of Report 2

Appendix

Table A1 Comparison of Maximum Penetration Depth

List of Figures

Fig. 1	Flow Sheet
Fig. 2	Test Section
Fig. 3	Coordinate System
Fig. 4	Cooling Box
Fig. 5	Instrumentation
Fig. 6	Data Acquisition System
Fig. 7	Illustration of principle of UVP
Fig. 8	Trend of temperature profiles of penetration flow
Fig. 9	Instantaneous velocity profiles
Fig. 10	Trend of velocity fields of penetration flow
Fig. 11	Comparison of velocity field and temperature field
Fig. 12	Maximum Penetration Depth by velocity profiles data of Case-1
Fig. 13	Maximum Penetration Depth by velocity profiles data of Case-2
Fig. 14	Maximum Penetration Depth by velocity profiles data of Case-3
Fig. 15	Maximum Penetration Depth by velocity profiles data of Case-4
Fig. 16	Maximum Penetration Depth by temperature data
Fig. 17	Maximum Penetration Depth related to R_i and Re Number
Fig. 18	Physical Mechanism of Penetration Flow
Fig. 19	Maximum Penetration Depth related to Gr/Re^3

Appendix

Fig. A1	Physical Mechanism of Penetration Flow
Fig. A2	The Edge of Penetration Flow
Fig. A3(A)	Time series of the Edge of Penetration Flow for Case-3 (Left)
Fig. A3(B)	Time series of the Edge of Penetration Flow for Case-3 (Right)
Fig. A4(A)	Time series of the Edge of Penetration Flow for Case-4 (Left)
Fig. A4(B)	Time series of the Edge of Penetration Flow for Case-4 (Right)
Fig. A5	Power Spectrum Distribution of Penetration Flow for Case-1
Fig. A6	Power Spectrum Distribution of Penetration Flow for Case-2
Fig. A7	Power Spectrum Distribution of Penetration Flow for Case-3
Fig. A8	Power Spectrum Distribution of Penetration Flow for Case-4

1. まえがき

高速炉の安全性をより一層高める上で、スクラム後の崩壊熱を除去する専用システム(崩壊熱除去系)の信頼性の向上が最も重要なテーマの1つとなっている。動的機器によらない自然循環力を利用した崩壊熱除去は、信頼性を向上させる有力な手段と考えられている。

崩壊熱除去系の設計形式のうち、DRACS(Direct Reactor Auxiliary Cooling System)形式を採用した場合、上部プレナム内に浸漬された炉内冷却器により上部プレナム底面近傍の冷却材は低温となり、また自然循環除熱時には炉心を通過する循環流量が2～4%と小さいため、上部プレナムの冷却材温度が集合体出口温度よりかなり低いある条件下では、一部の集合体へ上部プレナムの低温流体が潜り込むあるいは逆流することが考えられる。この現象は炉心—プレナム相互作用と呼ばれ、集合体内と上部プレナムの冷却材の温度差によって現象が顕在化する。この現象すなわち、炉内冷却器によって冷却された低温流体が炉心集合体に潜り込むことで燃料ピンの冷却に寄与するという効果が期待されるが、一方で本来の下からの冷却材流れを妨げ、全体の自然循環流に悪影響を及ぼすことも考えられる。従って、この現象が自然循環による崩壊熱除去時の炉心の冷却性に与える影響を評価する必要がある。その第1段階として、現象把握を行い、潜り込み初生条件及び潜り込み深さを測定することによって、現象を支配するパラメータの同定を目的とする基礎試験を行った²⁾。

試験に用いた装置は、水を作動流体とした、集合体出口部を模擬した矩形チャンネルとその上部に接続する上部プレナム、上部プレナム内の補助冷却器を模擬した低温壁からなる。低温壁によって冷却された流体は、上部プレナム底面に成層流を形成し、矩形チャンネルに向かって流れる。ある条件下では、チャンネル出口においてプレナムからチャンネル内への潜り込み現象が発生する。

第1報では潜り込み初生条件については、 $Gr/Re^2 = Ri >$ (しきい値) として表せることを明らかにした。一方、潜り込み深さについてはRi数のみによっては整理しきれず、同じRi数であっても潜り込み深さにはらつきが見られるという結果となった。そこで、試験体を改造することにより、潜り込み流れの温度分布の時間変化と、超音波流速分布測定装置(Ultrasonic Velocity Profile monitor : UVP)¹⁾を導入することによって潜り込み流れの瞬間流速分布を同時に測定した。さらに、これらの測定を通して潜り込んだ低温流体が到達する最大潜り込み深さについて支配パラメータの同定、相関式の作成を行った。

2. 試験装置

試験装置のフローシートをFig. 1に示す。装置は、試験ループと試験部（テストセクション）から構成される。試験ループは1次系（高温部）、冷却系（低温部）からなる。

2. 1 試験ループ

(1) 1次系

1次系は試験部の入口チャンネルに高温の水を流量、温度共に安定して供給する役割を持つ。ポンプの脈動の影響を避けるため、ヘッドタンク方式を採用し、ヘッドタンクと試験部との高低差により試験部に水を供給する。1次系は、ヘッドタンク、加熱タンク、汲み上げ用ポンプ、大流量、小流量の2系統の流量計、流量調節弁からなる。

- (A) ヘッドタンクは試験部出口とのヘッド差が2370mmで、容量が 1m^3 、試験部に最大 $0.1\text{m}^3/\text{min}$ を安定供給することができる。試験部を出した水は加熱タンクに戻る。液面は、加熱タンクからポンプを通してヘッドタンクに水を汲み上げるラインとベルマウスを通して加熱タンクへ戻るオーバフローラインにより一定に保たれる。
- (B) 加熱タンクは 0.5m^3 の容量を持ち、8kWのヒータ及び攪拌プロペラを備えている。ヘッドタンクと合わせて 1.5m^3 の水を攪拌プロペラで混合し、PID制御のヒータで一定温度に保ち、テストセクションへの供給流体の温度変動を $\pm 0.1^\circ\text{C}$ 以内に抑えている。
- (C) 2つの流量計の測定レンジはそれぞれ最大 $0.1\text{m}^3/\text{min}$ と最大 $0.03\text{m}^3/\text{min}$ で、バルブを切り替えることによって、小流量から大流量の調節と測定を可能にしている。今回の試験では、大流量用のバルブは全閉状態で用いた。

(2) 冷却系

冷却系は試験部の低温壁を一定温度に保つため、低温壁の外側に低温の水を安定供給する役割を持つ。冷却系は冷却タンク、ポンプ、流量計、2次系ループから構成される。

- (A) 冷却タンクには2次系ループからの低温水を内側に通す熱交換器が備え付けられている。熱交換器によって一定の温度に保たれた水はポンプにより試験部の低温壁を通って冷却タンクに戻る。冷却タンクは 0.5m^3 の容量を持ち、攪拌器を用いることにより温度の変動を抑え、低温壁への供給冷水の温度は $\pm 0.1^\circ\text{C}$ 以内で一定に保つことができる。
- (B) 2次系ループはチラー、ポンプ、温度調節用ヒータから構成される。チラー及びヒータにより熱交換器への温度の変動を $\pm 0.1^\circ\text{C}$ 以内に制御している。

2. 2 試験部

(1) 試験部形状

試験部の形状をFig. 2に示す。試験部は集合体出口部を模擬した矩形の入口チャンネル、その上部に接続する平板上の上部プレナム、炉内冷却器を模擬した上部プレナム右側面に設けた低温壁からなる。入口チャンネルにはその上流に整流格子を設け、格子の出口においてフラットな流速分布を与えるようにした。測定を行うに当たって定義した座標系をFig. 3に示す。

冷却壁は、温度分布を均一にするため銅板で作られている。Fig. 4に示すように2次側の水との熱伝達を大きくするため2次側流路はジグザグ型とし流速を大きくしている。冷却壁の試験部側表面の温度を測定するため、6本の熱電対を表面に面位置となるよう埋め込んでいる。ヒートロスを抑えるためアクリルの厚さを5cmとし、更に5cm厚の発泡スチロールで断熱している。

(2) 計測部

プレナムから入口チャンネルへの潜り込み流れを詳細に捉えるため、入口チャンネル内部に7本の熱電対をz方向に並べて温度分布の時間変化を計測し、またUVPトランスデューサを入口チャンネル出口部付近に取り付け、潜り込み流れの流速分布を得られるように改造した。計測を同時に開始することにより、温度分布と流速分布の時間変化を同時に計測することができる。計測部をFig. 5に示す。

2. 3 計測システム

(1) 温度分布測定

直径1mmのC-C非接地型熱電対を用いて温度を測定した。測定システムをFig. 6に示す。熱電対の信号を温度補償型アンプ（アナログデバイス社製モデル3B37）を用いて増幅し、データロガー（日本電気三栄社製7V08）にてA/D変換し、ディジタル値をメモリ内に取り込む。アンプは温度補償型であるため計測室内の温度を空調システムにより一定に保った。データロガーは全点同時取り込み型であり、ディジタル変換精度は12bitである。温度の測定精度はアンプと組み合わせたA/Dの分解能が0.06°Cである。温度較正の方法は以下のように行った。

基準熱電対について、精密温度計と恒温槽を用いて5°C～50°Cの範囲でアンプ、データロガーを含めた較正式（2次式）を求めた。較正式の制度は±0.1°Cである。この基準熱電対を試験部に組み込み、4.8m³/hrで試験体内を強制循環させ、10°C～50°Cの範囲で等温状態を作つて他の熱電対の相対較正を行つた。このときの較正式は2次式で較正式の最大誤差は0.2°Cであった。

(2) 流速分布測定

MET-FLOW社製超音波流速分布測定装置(Ultrasonic Velocity Profile monitor : UVP)を用いて潜り込み流れの z 方向成分の瞬間流速分布を測定した。UVPの動作原理の説明図をFig. 7に示す。UVPは超音波の発信及び受信を单一のデバイスで行い、超音波パルスの発信時刻とエコーの検出時刻の時間間隔から測定対象の位置情報を128点に亘って得ることができる。その各点のエコーのドップラーシフト周波数より流速情報（ビーム方向の成分）を検出している。UVPについての詳細は付録に添付した。

3. 実験方法

ループの運転手順を以下に示す。入口チャンネル側温度、供給流量を設定し、低温壁2次側流量はバルブ全開($1.5\text{m}^3/\text{hr}$)とする。2次系ループのチラーは最大出力で運転し、低温壁の2次側入口温度は2次系ループのヒータで調節する。入口チャンネル側流体との干渉で温度変動があるので、設定した温度まで微調整を行う。試験流量は微少であるため、十分な安定時間をとる必要があり、設定後24時間定常待ちを行い、更に微調整の後24時間定常待ちを繰り返す。入口チャンネル供給流量の変更の場合は、変更後6時間定常待ちを行った後、データの収録を開始した。

3. 1 試験パラメータと試験ケース

第1報において、潜り込み深さについては Ri 数では整理しきれないばらつきが見られ、他の整理パラメータが必要となった。第1報で行った試験ケースをTable 1に示す。第1報の結果では、潜り込み深さに対して、温度差以上に流速が強く影響しているため、本試験では、第1報で行った試験ケースのうち、平均流速のみを変化させ、 Re 数の影響を調べた。今回行った試験の試験ケースをTable 2に示す。

3. 2 チャンネル内温度分布測定

熱電対が10mm間隔に7本設置されており、70mmまでの同時計測が可能となっている。以下に述べる超音波流速分布測定装置と計測開始時間と合わせることによって、流速分布と温度分布の時間変化を同時に得ることが可能となる。

3. 3 チャンネル内流速分布測定

超音波流速分布測定装置を用いて、流速の z 方向成分を測定した。潜り込みの発生する入口チャンネル壁面に沿って計測できるように固定し、熱電対付近の流速を計測する。測定の際に超音波の反射粒子として、直径 $80\mu\text{m}$ 程度の粉末を作動流体に溶かす必要がある。本試験では、エクスパンセルDU-551を用いた（粒子直径約 $80\mu\text{m}$ 比重はほぼ1.0）。装置はIBM互換機をコントローラとして使用しており、OSはDR-DOS 5.0 (MS-DOS互換) を用いている。データはBinary Dataとして記録される。データファイルの構造は付録に添付した。また、データ処理ツールを今回作成し、そのツールを用いて試験結果の出力を行った。ツールのリストは付録に収録した。

4. 実験結果

4. 1 温度分布測定結果

潜り込み流れの温度分布が時間の経過に伴って変化する様子をFig. 8に示す。試験条件は高温部50°C、低温部12.5°C、平均流速3cm/sec(13.5l/min)でTable 2のCase-3の場合である。縦軸にプレナム底面からの深さをとり、横軸に経過時間をとった。時間の経過と共に温度の低下と上昇を繰り返しており、潜り込み現象の不規則な様子を見ることができる。温度上昇の原因として、潜り込み流れの強制循環流からの熱伝導による効果と混合対流による効果とが考えられるが、後に示す流速分布の計測結果からわかるように流体の運動に伴った温度変動が生じており、対流混合による温度上昇の効果がより大きいと考えられる。このように潜り込み過程において、潜り込み流れの温度分布が時間の経過と共に変化し、負の浮力の減少が生じていることがわかる。

4. 2 流速分布測定結果

Fig. 9にUVPによって得られたある瞬間の潜り込み流れの流速分布を示す。縦軸にプレナム底面からの深さをとり、横軸は流速の z 方向成分である。トランスペューサに対して遠ざかる方向は正の速度として計測される。従って、正の流速は潜り込み流れを表し、負の流速は上昇流を表している。このように潜り込み流れはその内部で非常に乱れた流れとなっており、対流混合が生じている。更に、この瞬間流速分布の時間変化を表すため、各瞬間の流速の大きさを色分けし、それを時間経過の順に並べたものをFig. 10に示す。縦軸はプレナム底面からの深さで、横軸は経過時間を表している。チャンネルの深い領域では、同様な色のパターンが見られ、流速の小さな変動が見られるものの、ほぼ安定した状態となっている。また、各図においてチャンネルの下方から上方に向かう無数の斜めの線は、渦の移動であり、その勾配は渦の移動速度を表している。黒色となるのは、渦の移動速度と回転速度の方向が逆で、その速度が計測限界未満であるためである。一方、プレナム底面付近では、潜り込み流れによる下降流が不規則に発生しており、上昇流との混合により流速が複雑に変動しているのが確認される。平均流速が大きくなるにつれて潜り込み深さが減少するが、流速分布の測定結果でも平均流速が大きくなるにつれて、潜り込みによる複雑な流速が変動する領域が縮小されていく。

Fig. 11に入口平均流速が3.0[cm/sec] (Table2 Case-3) の条件での温度分布及び流速分布の同時計測の結果を示す。計測点の位置及び計測開始時間は厳密には一致していないが、プロファイルは非常に一致しており、流速の変動と共に温度も変化していることが明確にわかる。潜り込み流れの時間変化の様相は可視化試験によって得られていたが、流速と温度の定量的な把握が困難であった。今回の試験では、超音波流速分布測定装置の導入により、時間的に変動する流

れについても空間情報を得ることに成功した。

第1報では潜り込み深さを温度データから一定の規則に基づいて求めたが、流速分布から潜り込み深さを求めたものをFig. 12, 13, 14, 15に示す。これらの図は、正の流速と負の流速を別々に平均値と標準偏差を求め、横軸にプレナム底面からの深さをとり、縦軸に流速をとってプロットしたものである。流速が正の領域が潜り込みの発生している領域で、その最も深い位置が最大潜り込み深さと考えられる。プレナム底面からの深さが600mm付近において流速の乱れが観察されるが、この位置には固定された熱電対があり、UVPは非常によく位置情報を得ていることが確認される。このようにして流速分布から得られた潜り込み深さと、温度データから求めた潜り込み深さを比較すると、わずかの差であるが流速分布から求めた最大潜り込み深さの方がより大きいことがわかる。このことから潜り込み流れは温度差がなくなっても慣性のため更に深く潜り込んでいくと考えられる。しかし、この違いは、絶対的な深さの差だけであり、他のケースとの相対的な関係は変わらず、支配パラメータには無関係と考えられる。従って、第1報で行ったように温度データのみによって潜り込み深さを評価しても差し支えないといえる。

5. 考察

第1報で得られた温度データから求めた最大潜り込み深さを $Ri = Gr/Re^2$ を横軸にとって表したものとFig. 16に示す。第1報でも述べたように最大潜り込み深さは Ri 数のみでは整理できない。潜り込み深さに対する Re 数の影響を見るため、縦軸に Ri 数、横軸に Re 数をとり、潜り込み深さを色分けし、コンター表示したものをFig. 17に示す。このように潜り込み深さには Re 数依存性が見られる。また、ある深さでの Ri と $1/Re$ の関係は双曲線的であり、すなわちある深さでは $Ri \times 1/Re = \text{Const}$ の成立が推測される。そこで、潜り込み流れに対する物理モデルを以下のように仮定した。

可視化及び温度分布及び流速分布計測の結果から考えられる潜り込み流れの物理モデルをFig. 18に示す。潜り込み流れには、強制循環流の慣性力、プレナム底面と強制循環流との温度差に基づく負の浮力が働いている。更に潜り込みの過程において、強制循環流から潜り込み流れへの熱伝導によって潜り込み流れが温度上昇し、その結果、浮力が減少すること、強制循環流によるエントレインメントによって、潜り込み流れの一部が持ち去られることが考えられる。これらの現象に関係する物理量は、図に示したように潜り込み深さ z 、流速 v 、浮力 b ($\rho \beta \Delta T$)、粘性係数 μ 、密度 ρ 、熱伝導率 λ 、比熱 c_p と考えられる。これらの物理量に対して代表長さを矩形チャンネルの水力等価直径 De をとり、 π 定理に従って次元解析を行った。その結果は次のようになる。

$$\left(\frac{zp v}{\mu} \right)^{\pi_1} \left(\frac{\lambda}{C_p \mu} \right)^{\pi_2} \left(\frac{\mu \rho g \beta \Delta T}{\rho^2 \mu^3} \right)^{\pi_3}$$

$$\left(\frac{zp v}{\mu} \right)^{\pi_1} = (1/\Pr)^{\pi_2} (Gr/Re^3)^{\pi_3}$$

従って、 Gr/Re^3 が潜り込み深さに関係していることがわかる。更に左辺値が Re 数と同様な形となり、物理的な意味を持たせることができるように結果となるが、実験データをこの式に代入したところ、小さな Gr/Re^3 では一致しなくなる。理由として、小さな Gr/Re^3 域では潜り込みが深く、不確定さが大きくなるため誤差が増大することや、潜り込みの定義と物理モデルとの対応が困難であるなどが考えられる。しかし、潜り込み深さは上昇流の流速に大きく影響され、 $Ri \times 1/Re = \text{Const.}$ の成立が工学的にも矛盾がないため、潜り込み深さの支配パラメータは次元解析の結果を採用し、 Pr 数と Gr/Re^3 であると結論した。更に、左辺値を z/De として整理するとよく一致するため、実験相関式として以下の式を結論とした。

$$z/De \propto F(\Pr, Gr/Re^3)$$

縦軸に z/De をとり、横軸に Gr/Re^3 をとったものをFig. 19に示す。この当てはめの相関係数は 0.97 であり、非常によく一致している。この物理モデルは作動流体を水と限定せずに導いたものであるため、ナトリウムに適用しても成立すると考えられる。

6. まとめ

1. 瞬間流速分布及び温度分布の同時計測により、定量的な可視化手法を確立し、その手法を潜り込み流れに適用した。その結果、流速プロファイルの時間変化は温度プロファイルの時間変化とよく一致した。また、温度データによって求めた潜り込み深さは流速データによって求めたものと一致することを確認した。
2. 潜り込み流れの物理モデルから関係物理量を摘出し、次元解析を行うことにより、潜り込み深さの支配パラメータは、Pr数および Gr/Re^3 であることがわかった。すなわち、潜り込み深さ z は、

$$z = f(Gr/Re^3, Pr)$$

として表すことができ、本体系では、

$$z [m] = 220(Gr/Re^3)$$

と表される。

3. 炉工室にて実施しているCCTL-CFR試験において炉心プレナム相互作用に関する試験が予定されており、本水流動試験で得られた結果の妥当性を検討する予定である。

7. 謝辞

本試験の遂行にあたり適切なアドバイスを頂いた、安全工学部、相澤部長、前田部長代理、実験炉部、原部長代理、原子炉工学室、佐藤室長、山口主任研究員、試験の実施においてご指導を頂いた宮越研究員、試験に参加協力して頂いた水流動試験室員の常陽産業、伊藤真美氏、小沼英良氏に感謝の意を表する。また、P S I の武田靖博士には、超音波流速分布測定装置の使用法のみならず、試験データの評価方法等多大なるご指導を頂いた。ここに記して心から感謝する。最後に、原子炉工学室室員のC S K、浜田隆弘氏、天澤智子女史（現：機器・構造安全工学室）には、データの整理及び出力の協力を頂いた。ここに感謝の意を表する。

8. 参考文献

- (1) Y. Takeda, "Development of Ultrasonic Velocity Profile Monitor (UVP) and its Experience", NURETH-4 vol. 1. P418-423
- (2) J. Kobayashi, H. Kamide, Y. Ieda, "Core-Plenum Interaction Experiment -Onset Condition and Penetration Depth-", PNC ZN9410 92-267
- (3) MET-FLOW SA, Ultrasonic Velocity Profile monitor, Operational Manual

Table 1 Test Parameter Matrix of Report 1²⁾

Temperature at Channel Inlet	Temperature of feed water of Cooling box	Velocity at Channel Inlet (cm/sec)						
		2.5	2.75	3	4	5	6	7
50	12.5	XX	YY	VV	O	J	K	EE
45	-				QQ	RR	DD	
40	-			OO	PP	BB	CC	
35	-			NN	T	Z	AA	
30	-			MM	U	Y		
25	-			W	V			
20	-			X				
50	16.0				SS	FF		
-	20.0				TT	GG		
-	25.0				UU	HH		

* XX is a run name of each test case.

Table 2 Test Parameter Matrix of Report 2

	Temperature at Channel Inlet	Temperature of feed water of Cooling box	Velocity at Channel Inlet
	[°C]	[°C]	[cm/sec]
Case - 1	50	12.5	2.5
Case - 2	50	12.5	2.75
Case - 3	50	12.5	3.0
Case - 4	50	12.5	4.0

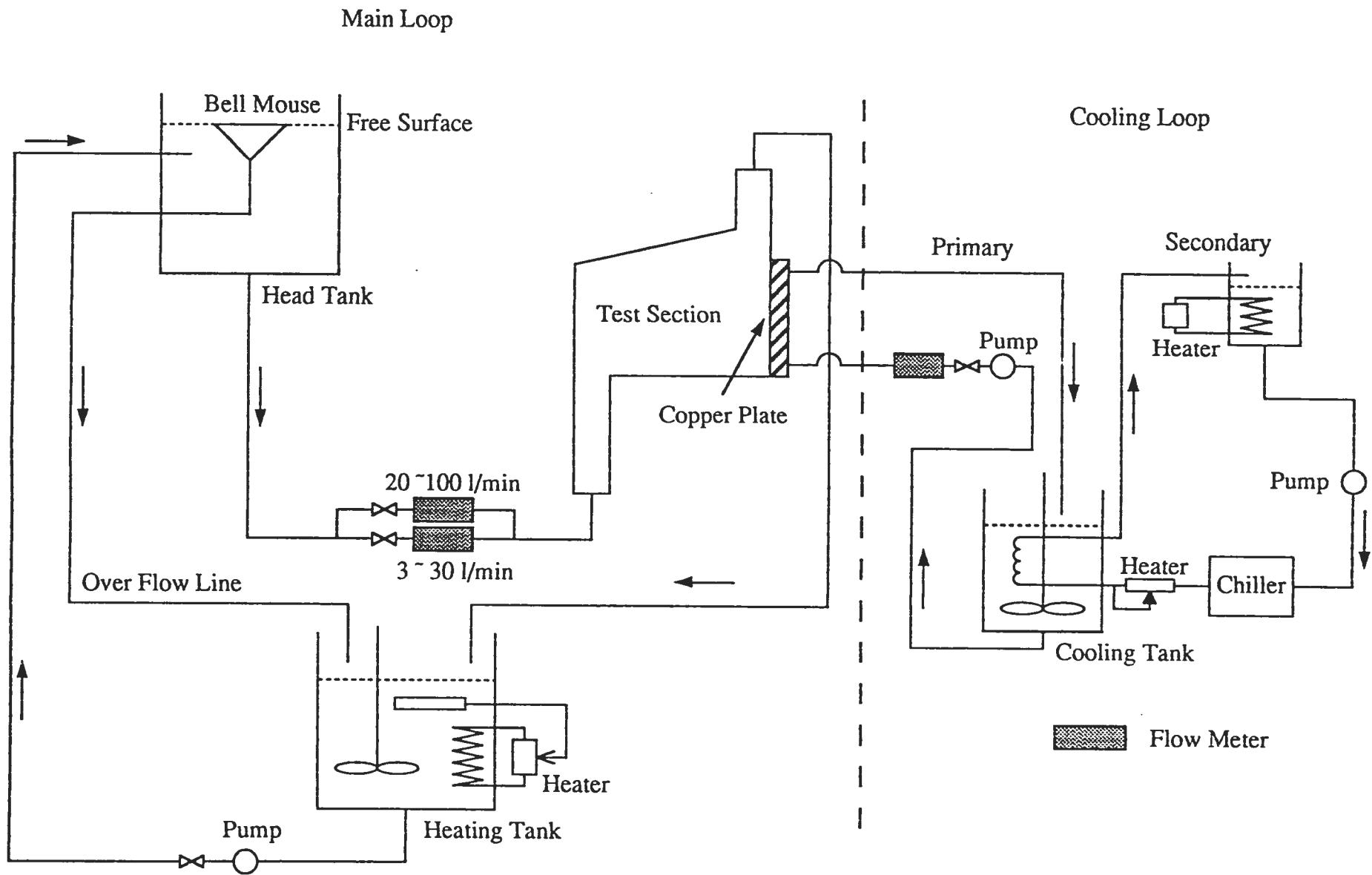


Fig. 1 Flow Sheet

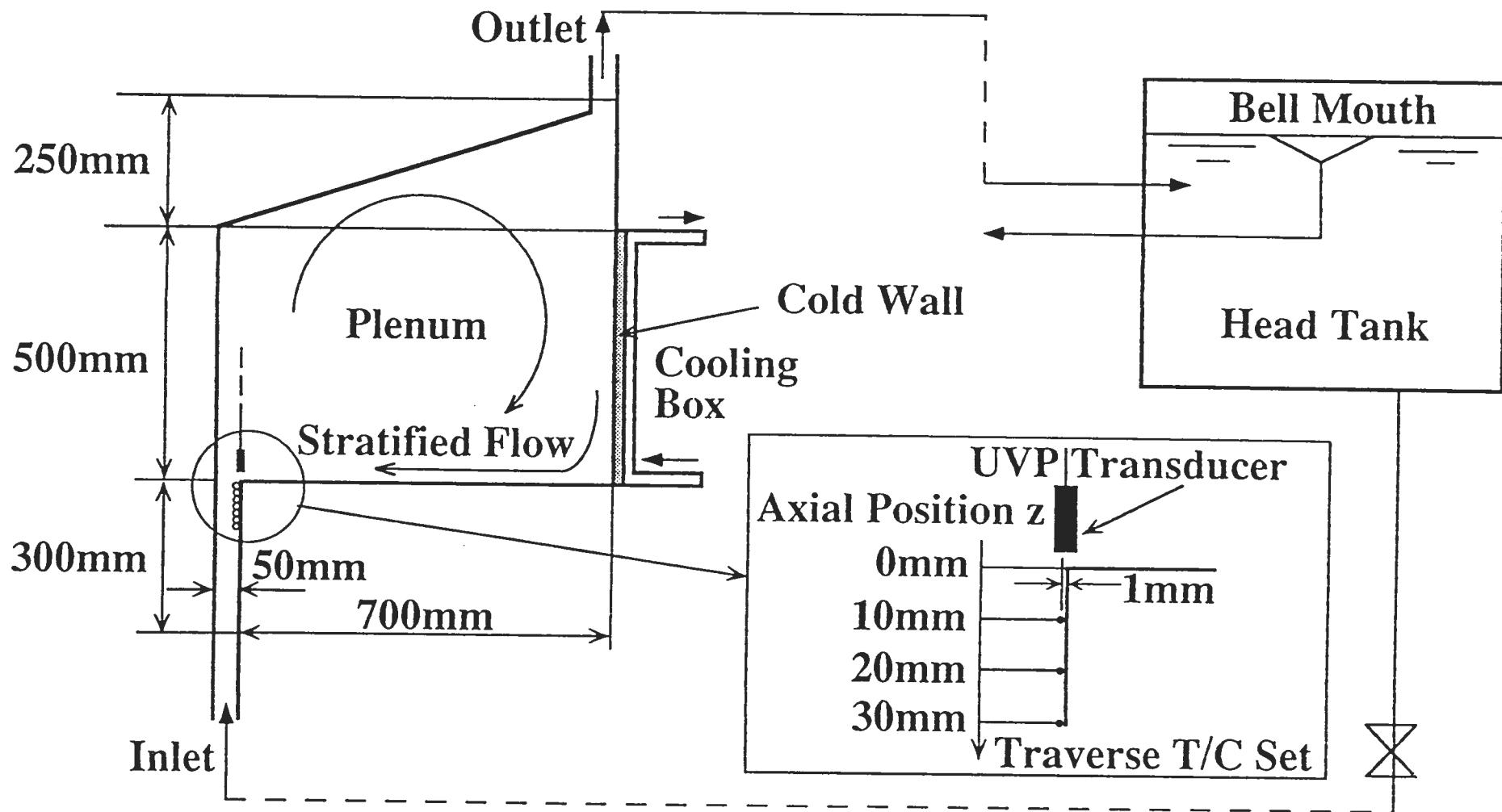


Fig. 2 Test Section

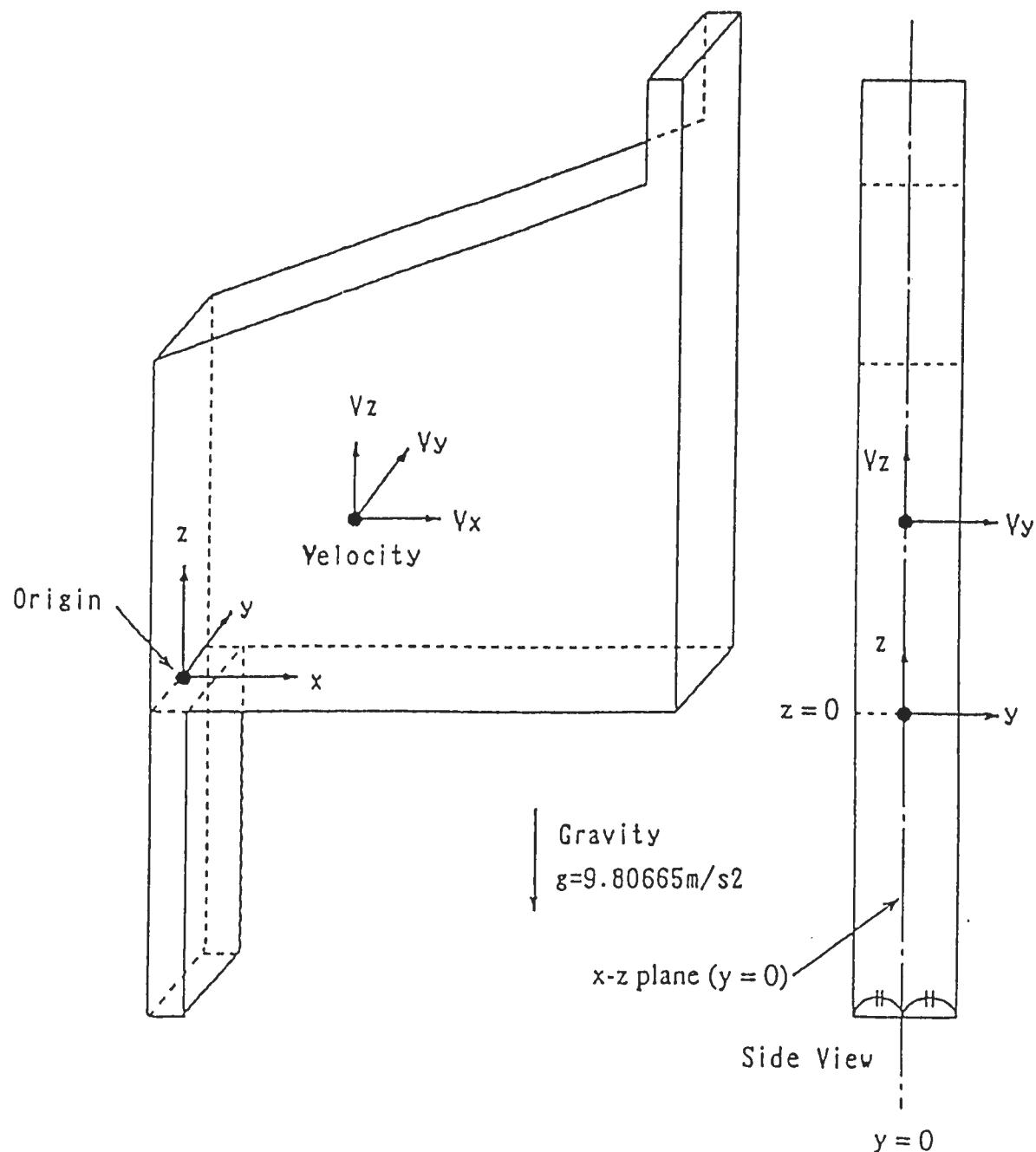


Fig. 3 Coordinate System

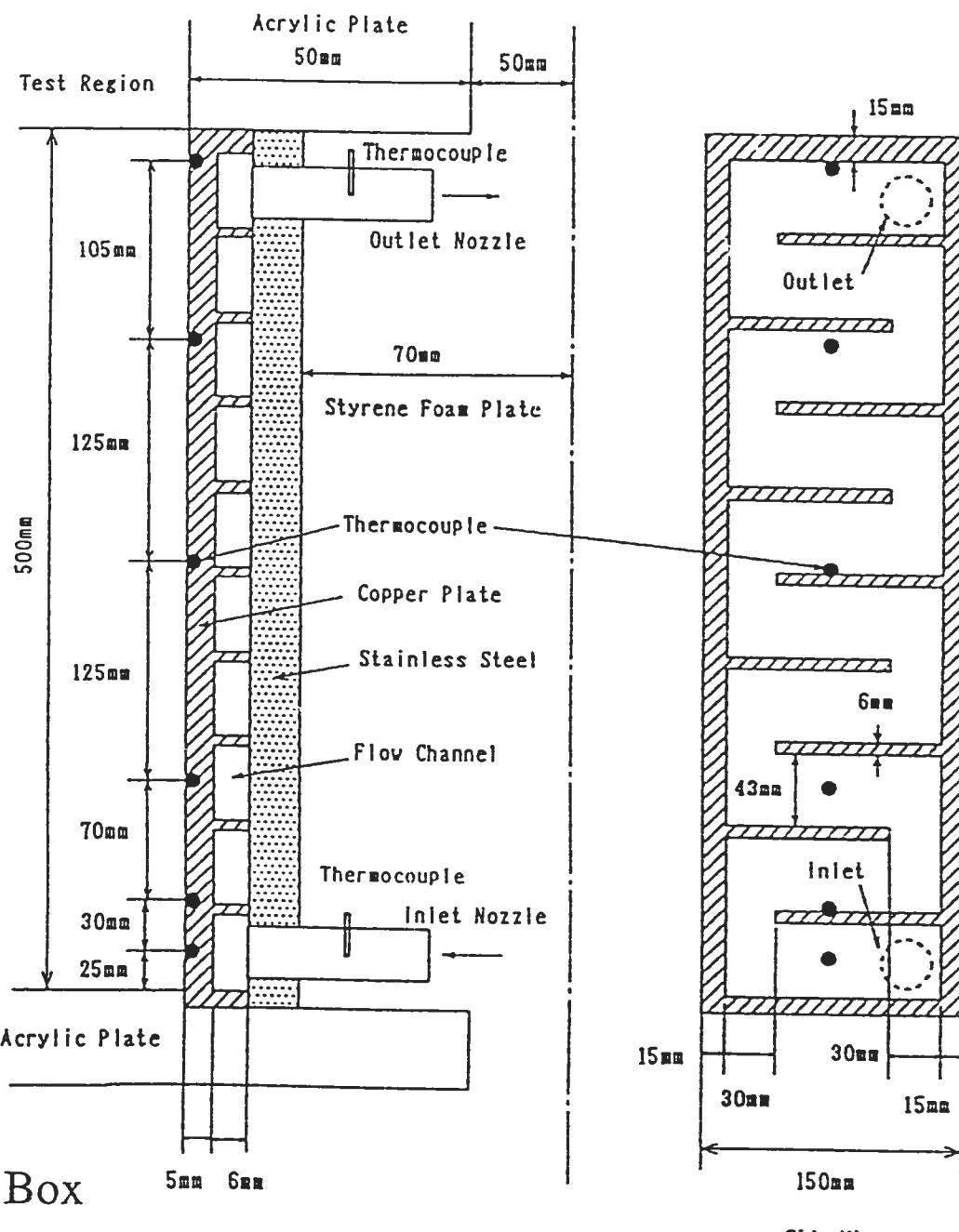
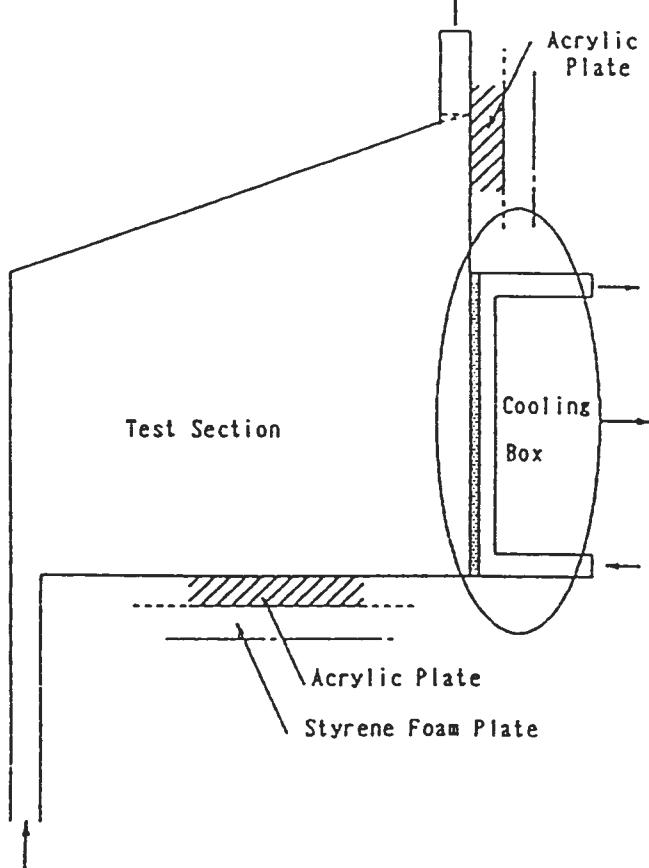


Fig. 4 Cooling Box

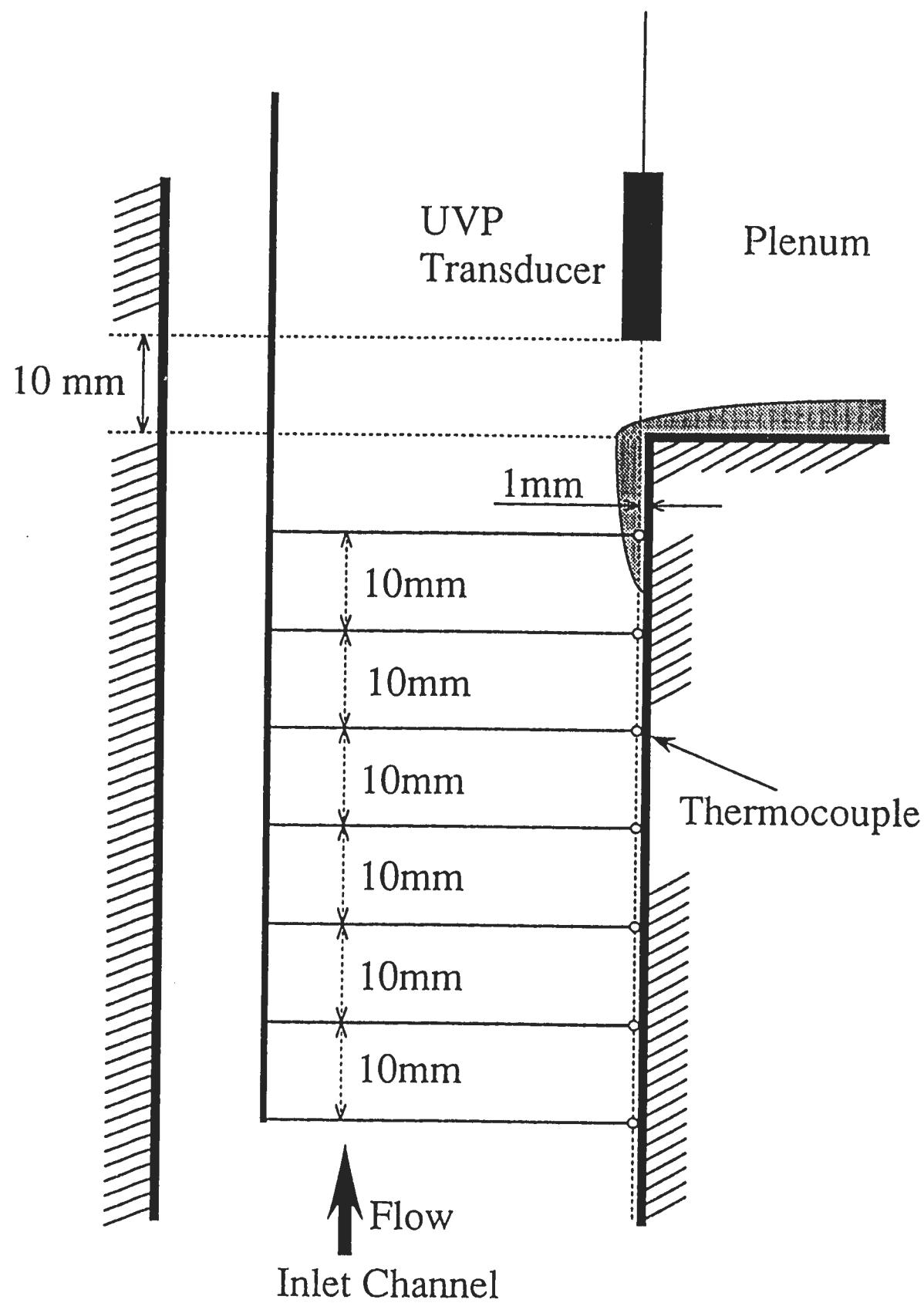


Fig. 5 Instrumentation

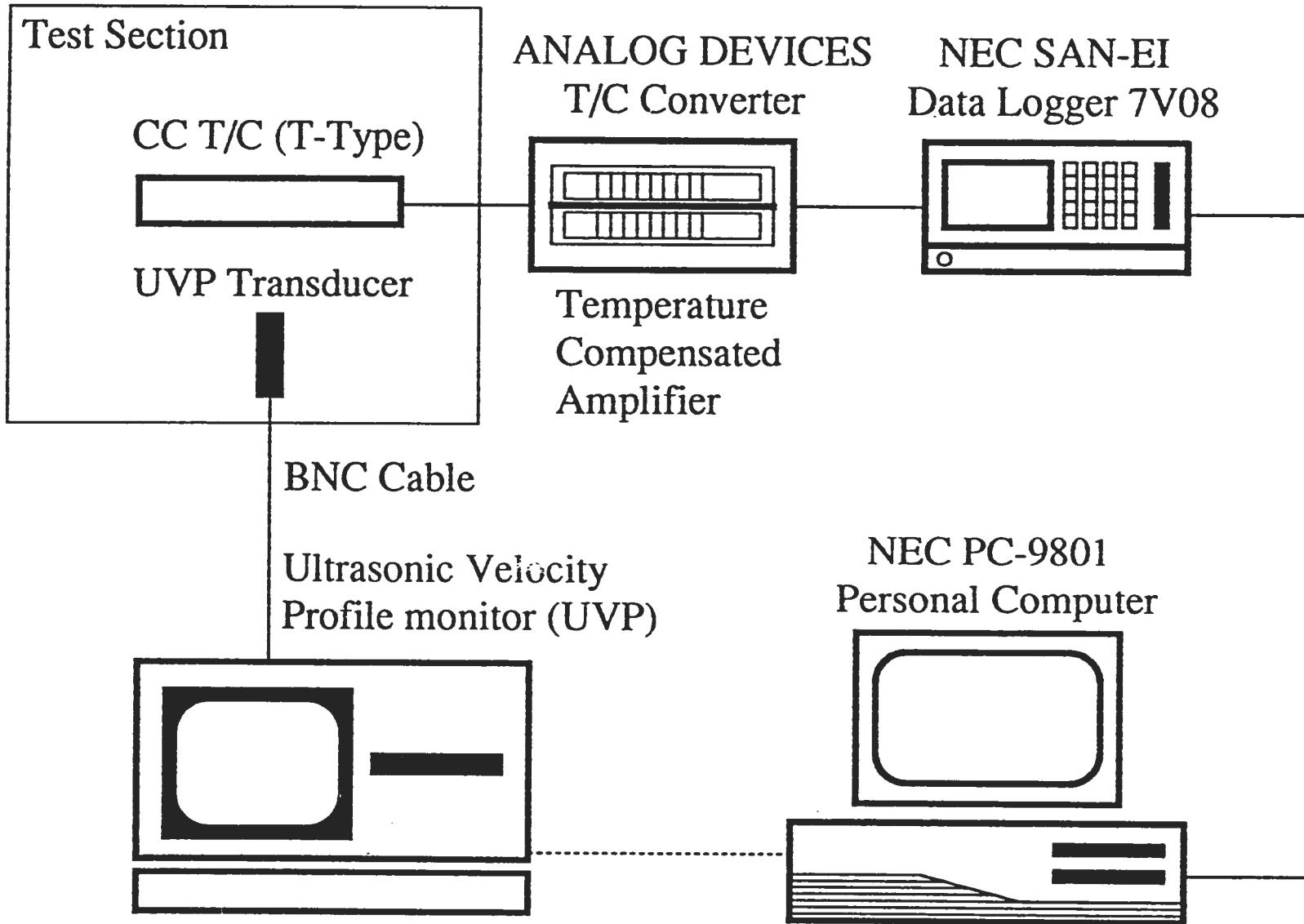


Fig. 6 Data Acquisition System

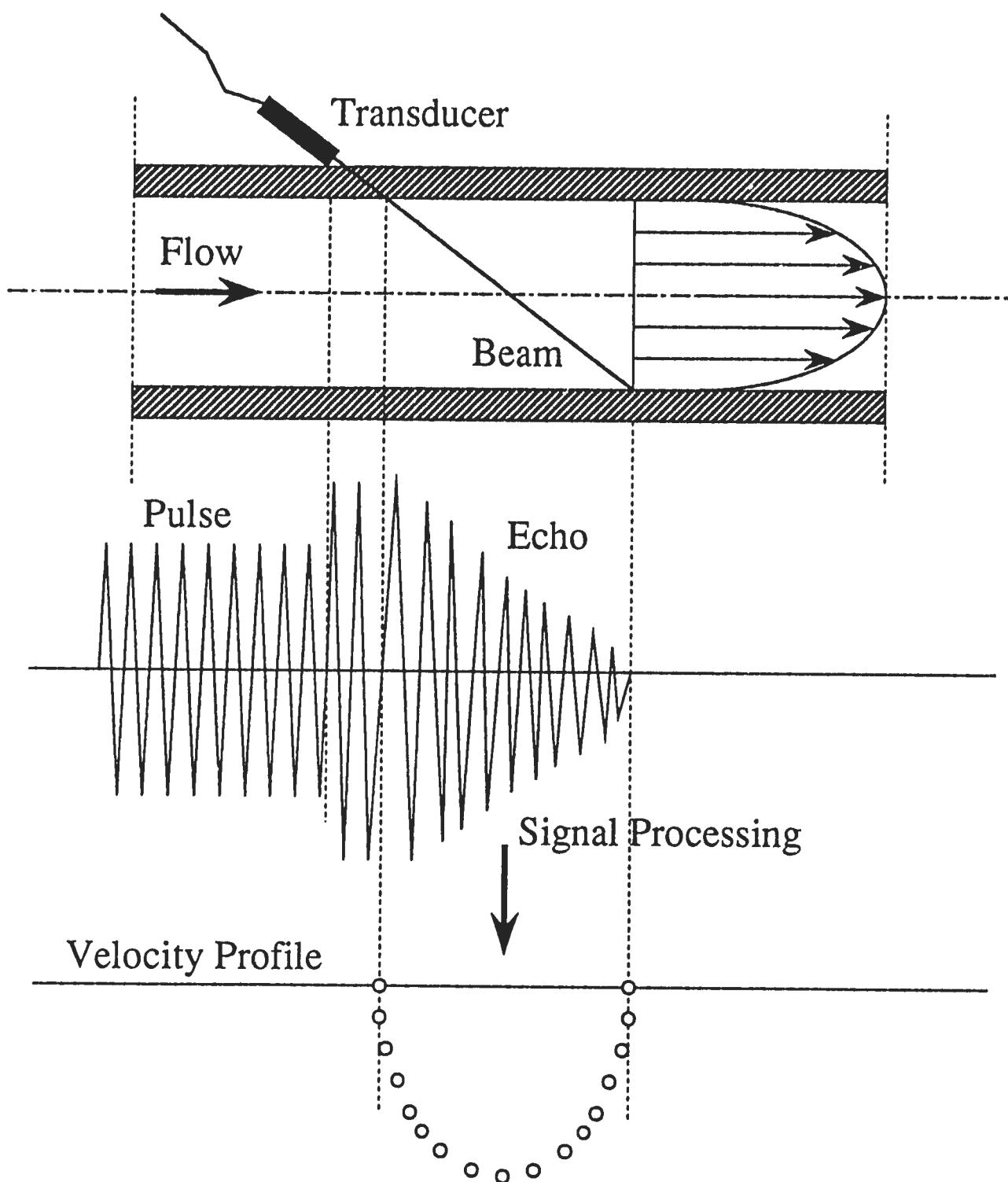


Fig. 7 Illustration of principle of UVP

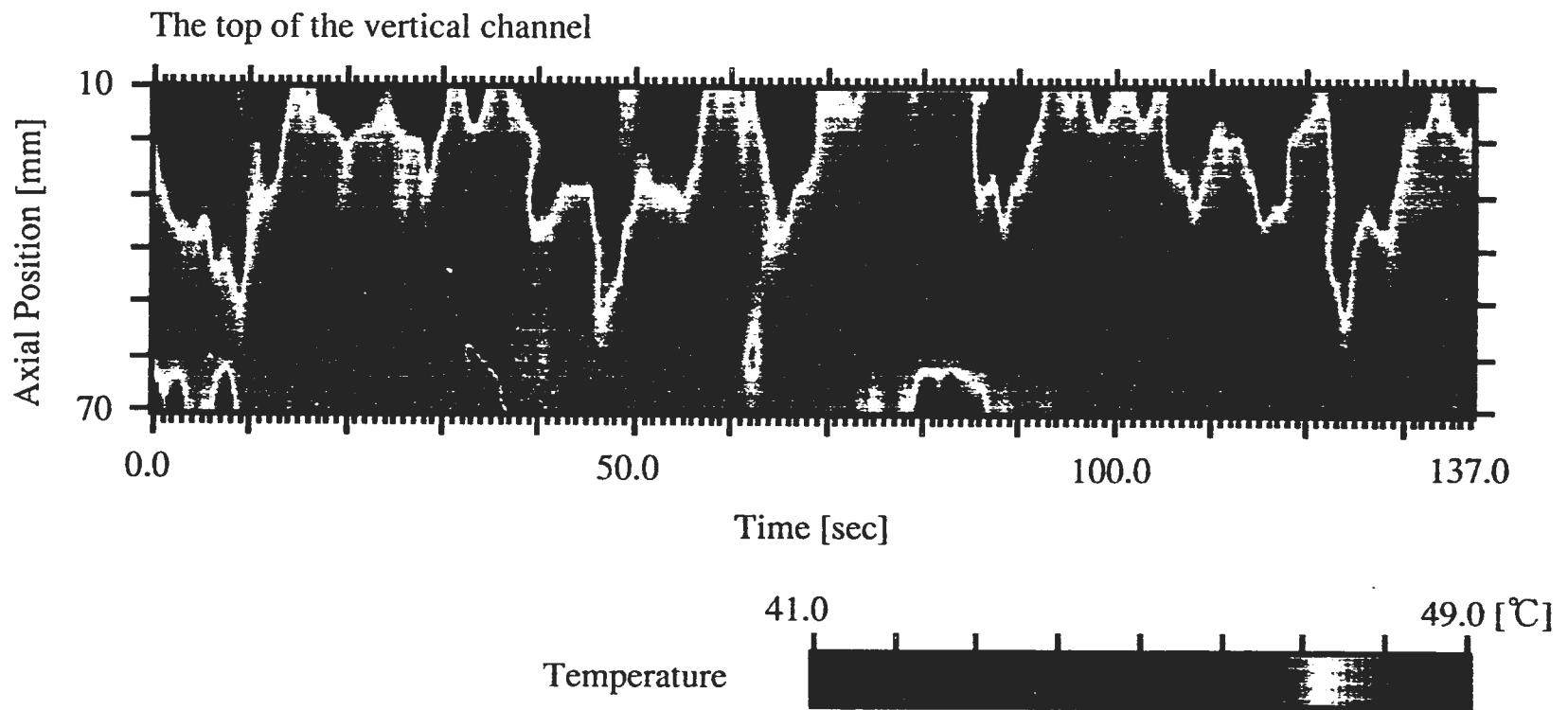


Fig. 8 Trend of temperature profiles of penetration flow

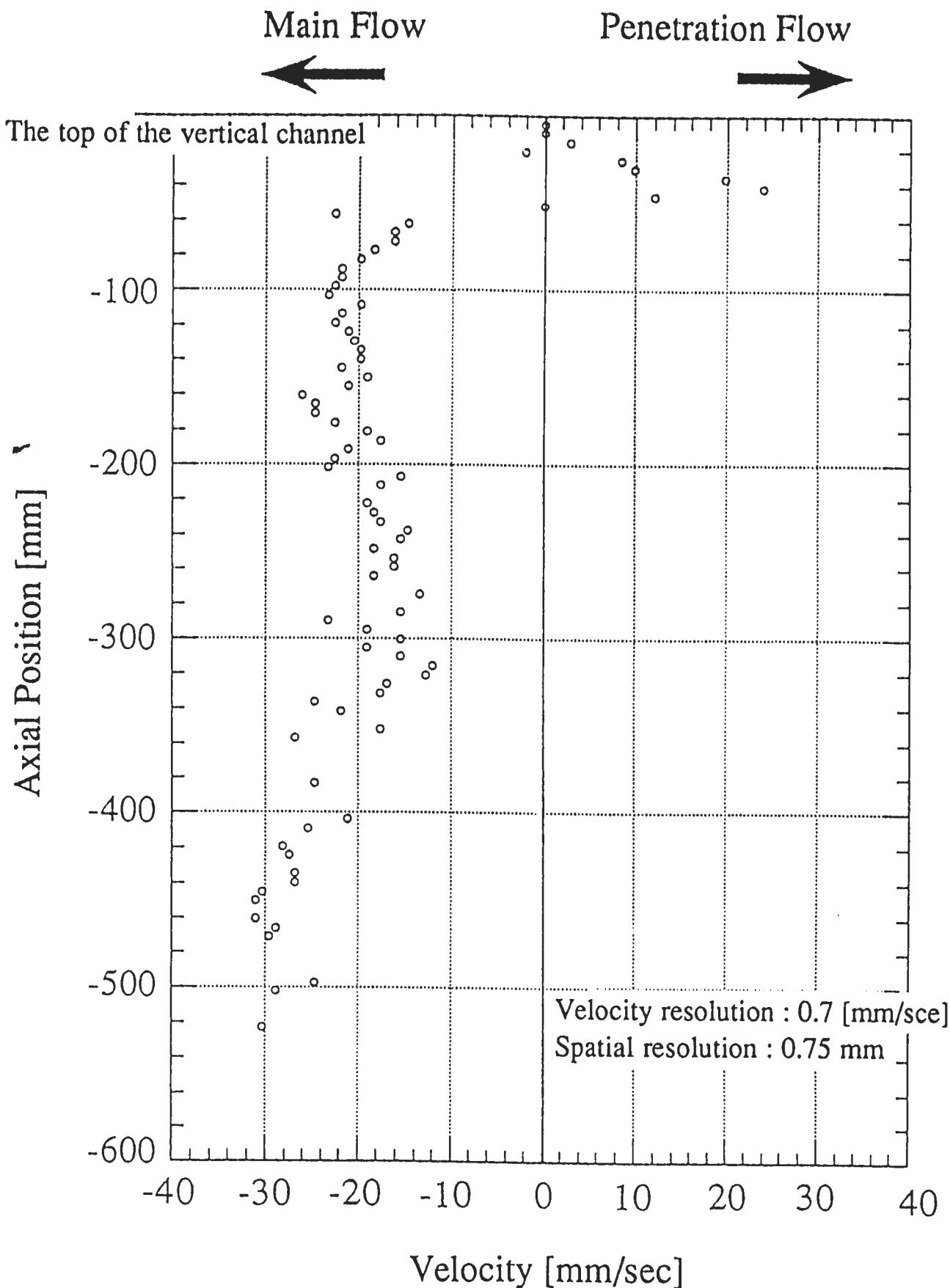


Fig. 9 Instantaneous velocity profiles

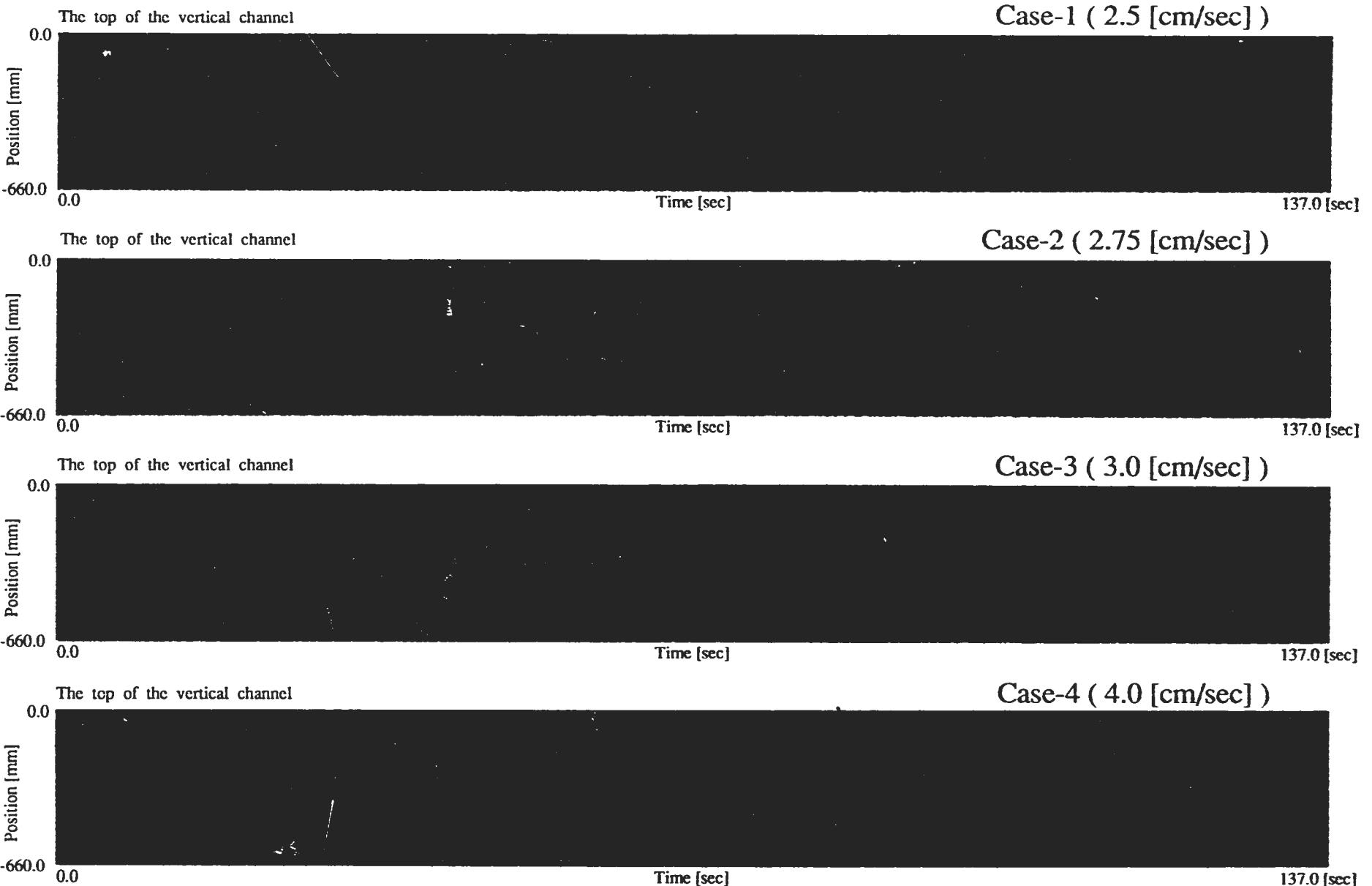


Fig. 10 Trend of velocity fields of penetration flow

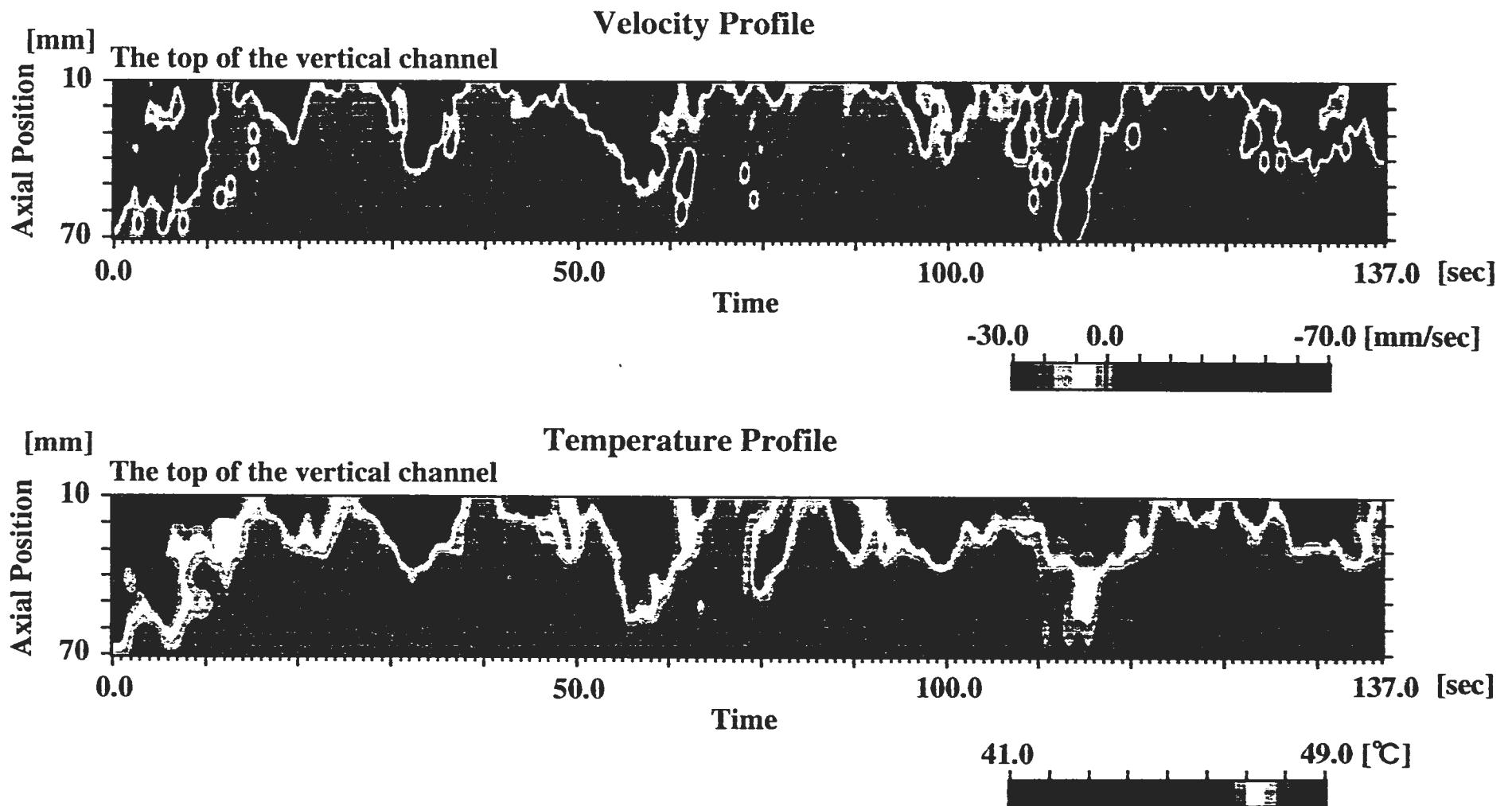


Fig. 11 Comparison of velocity field and temperature field

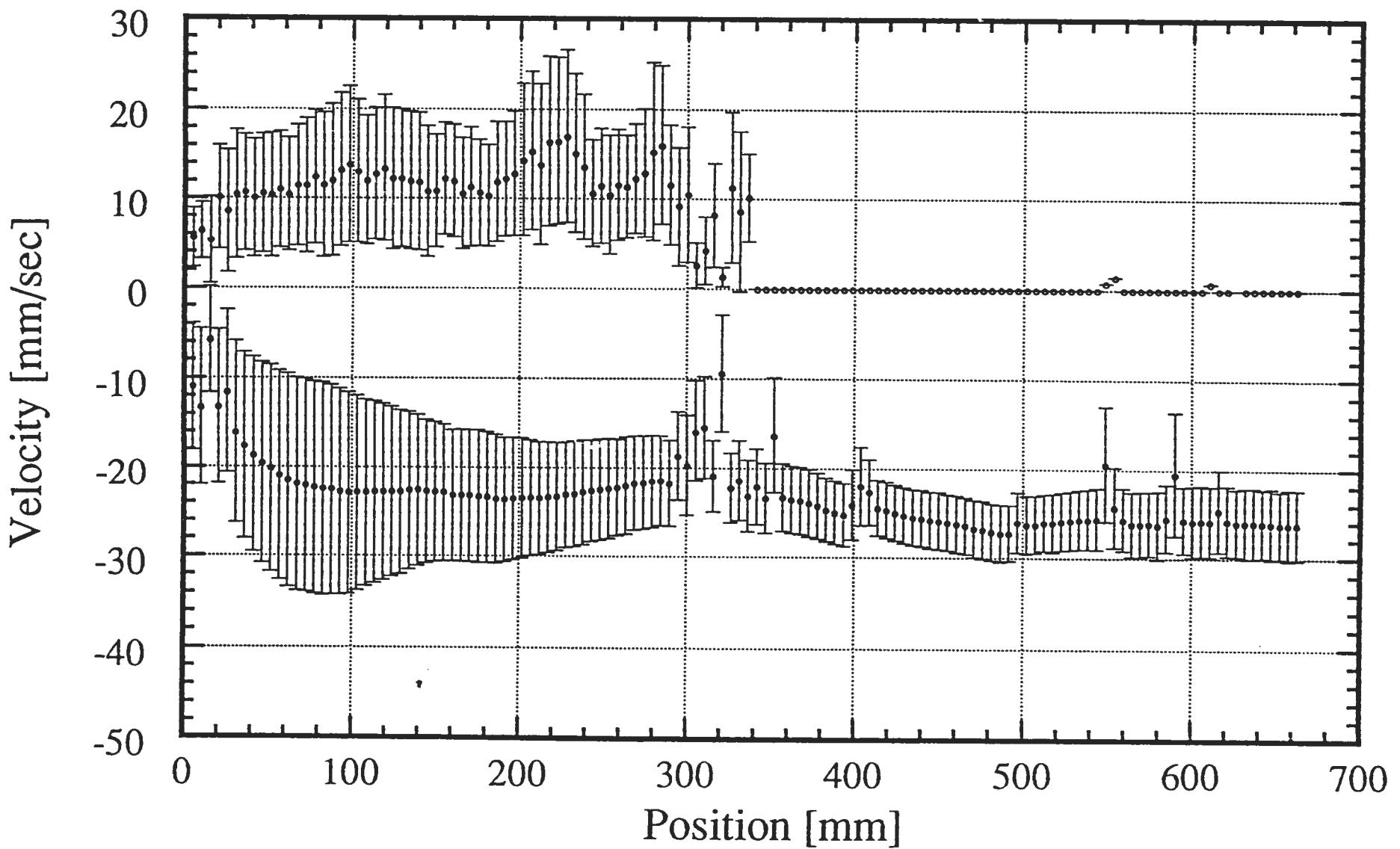


Fig. 12 Maximum Penetration Depth
by velocity profile data of Case-1

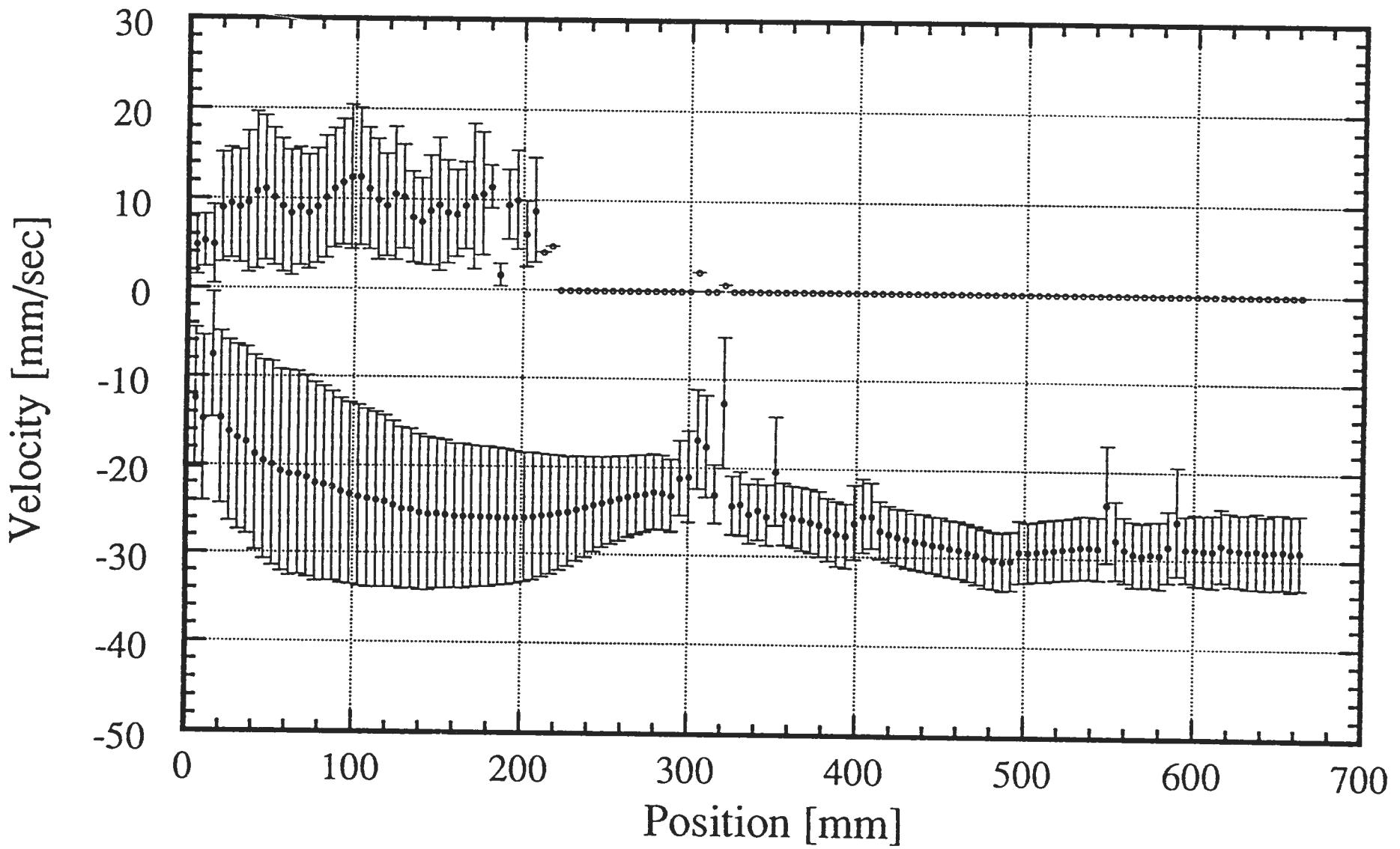


Fig. 13 Maximum Penetration Depth
by velocity profiles data of Case-2

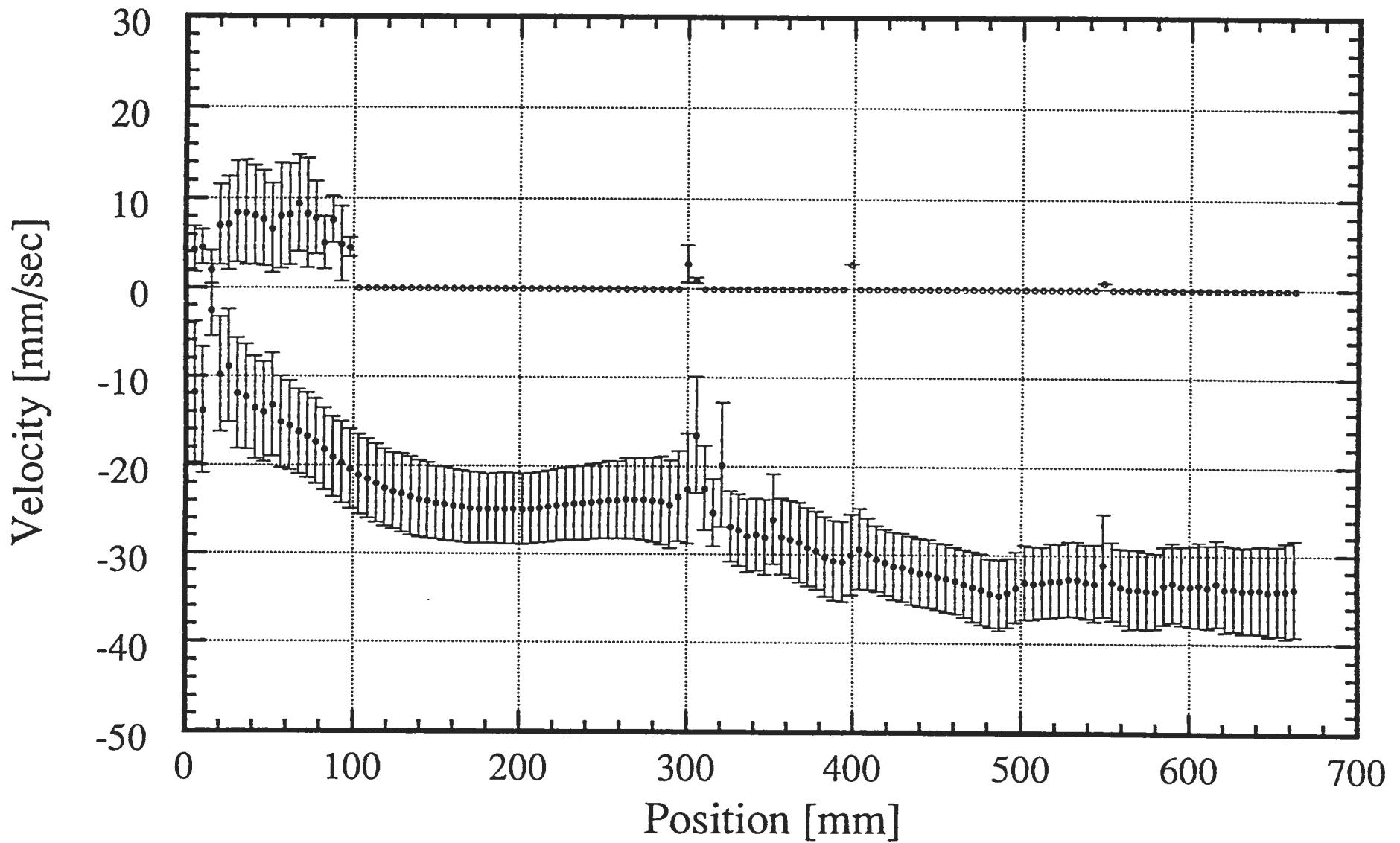


Fig. 14 Maximum Penetration Depth
by velocity profiles data of Case-3

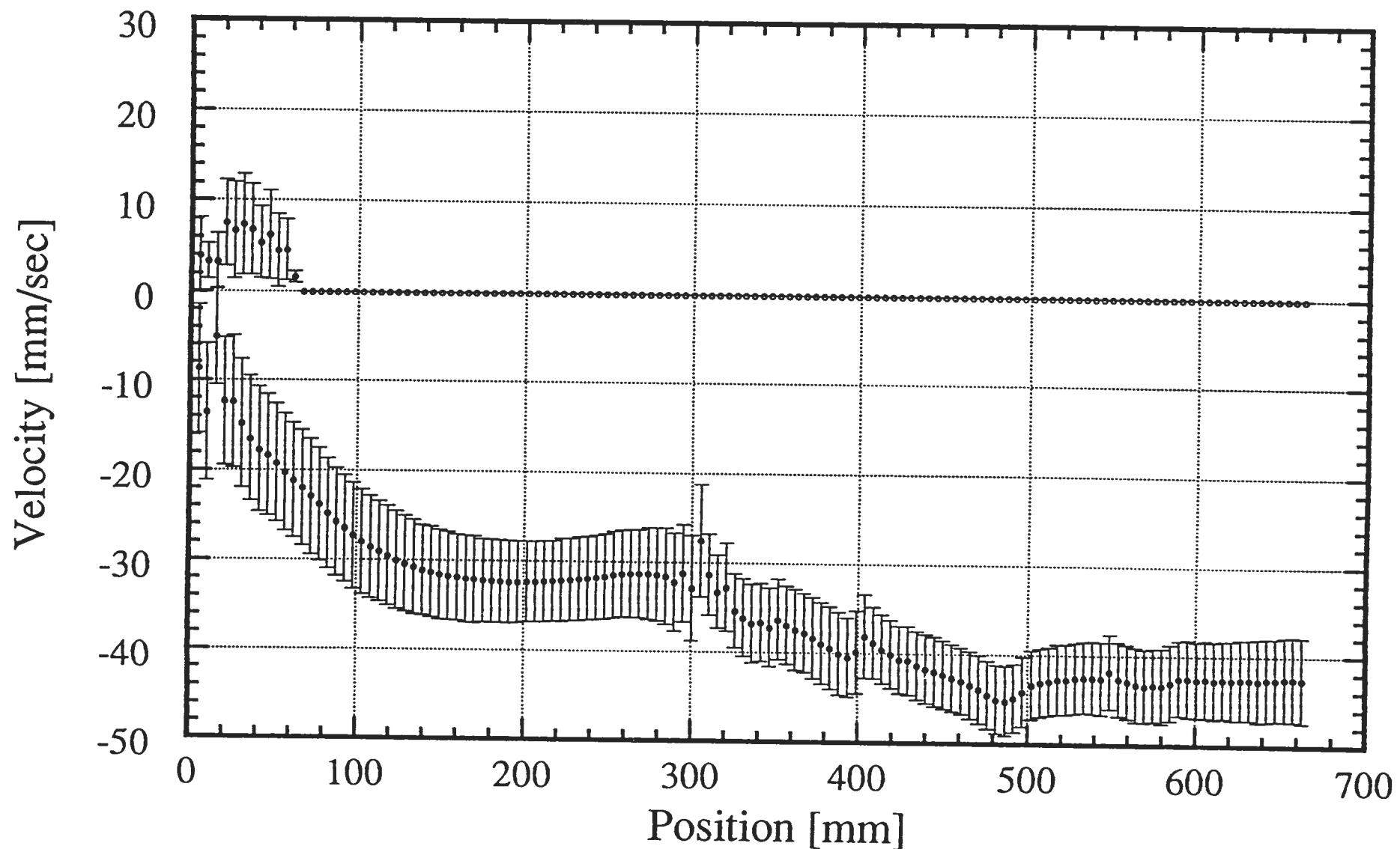


Fig. 15 Maximum Penetration Depth
by velocity profiles data of Case-4

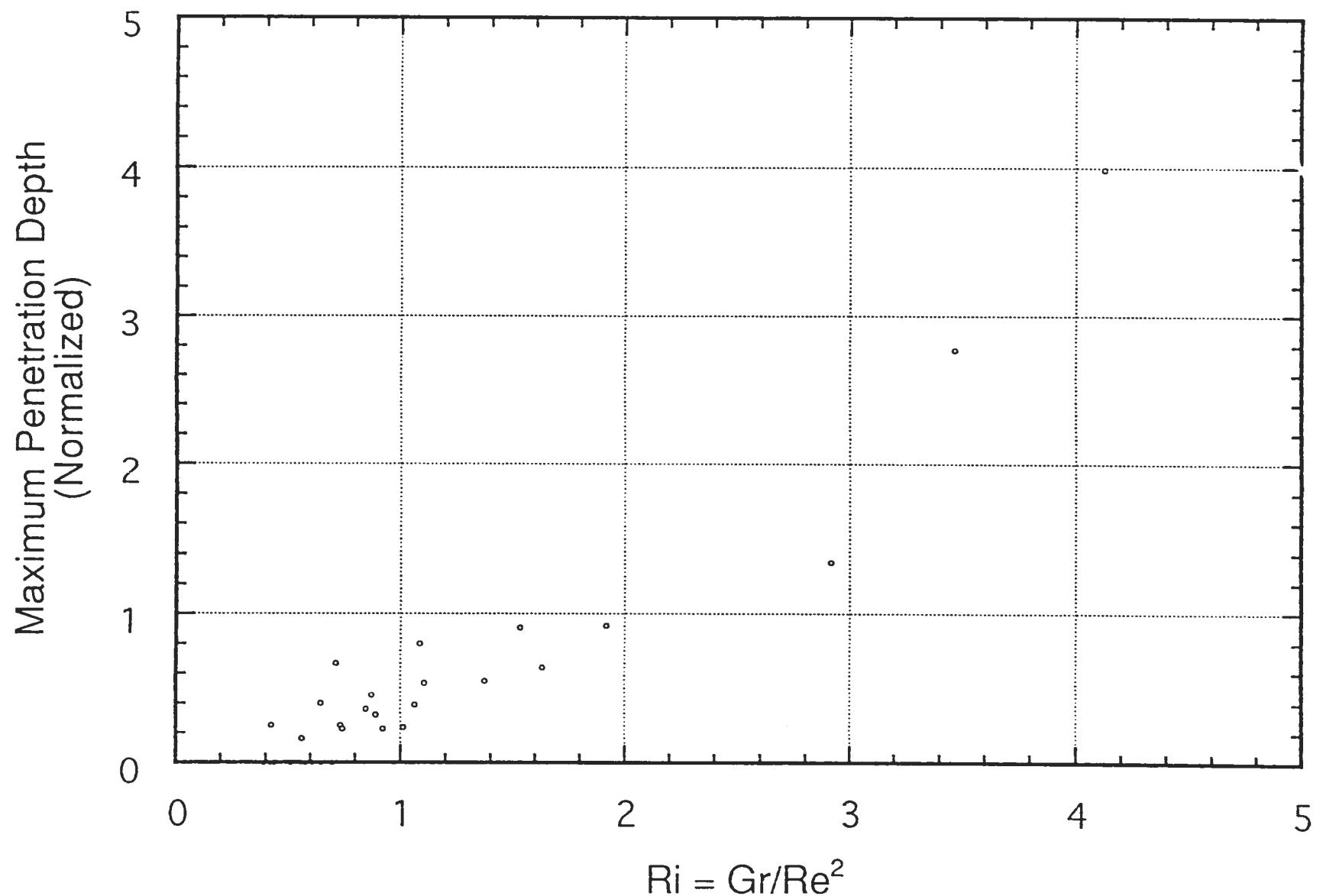


Fig. 16 Maximum Penetration Depth by temperature data

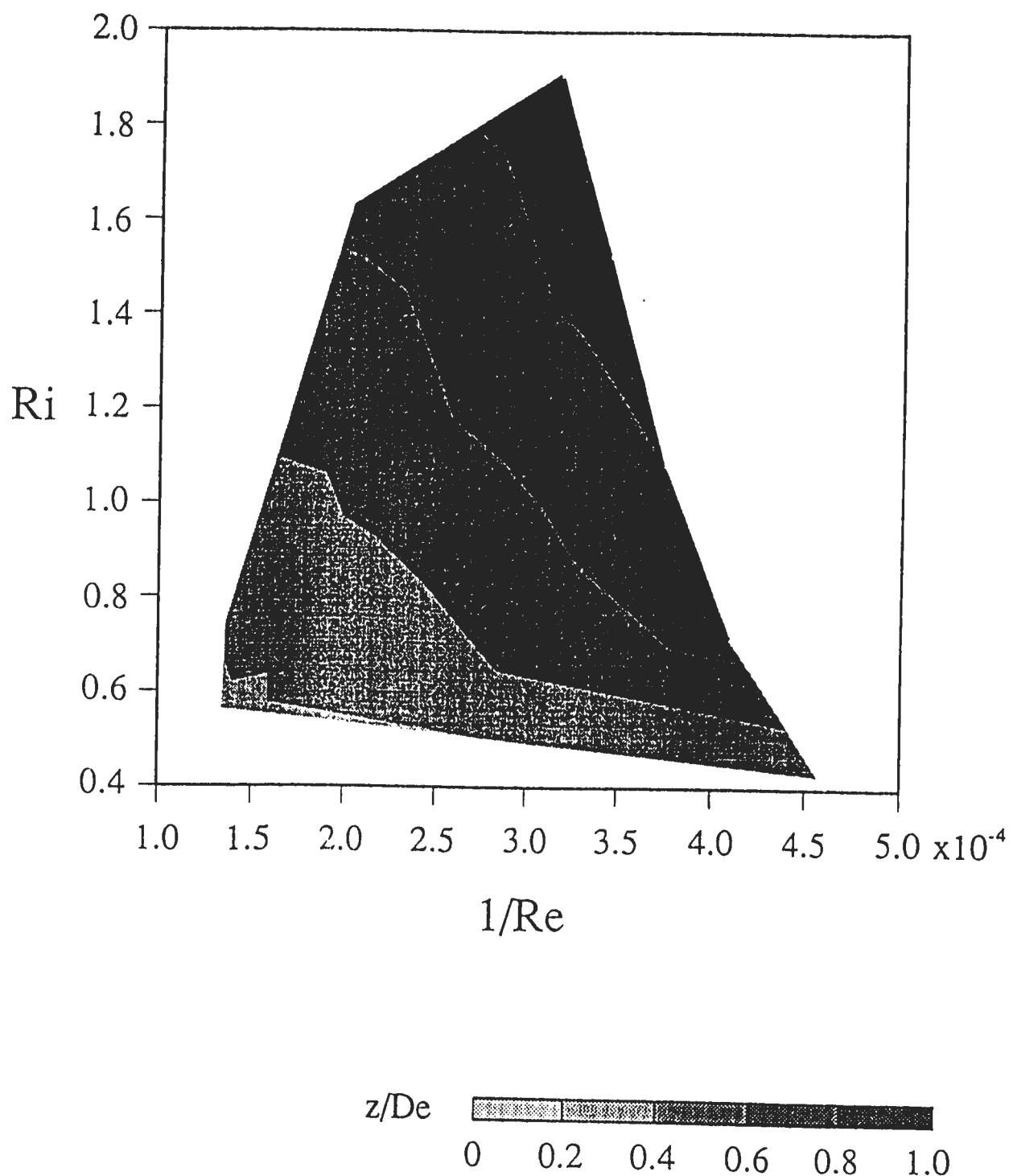


Fig. 17 Maximum Penetration Depth related to Ri and Re Number

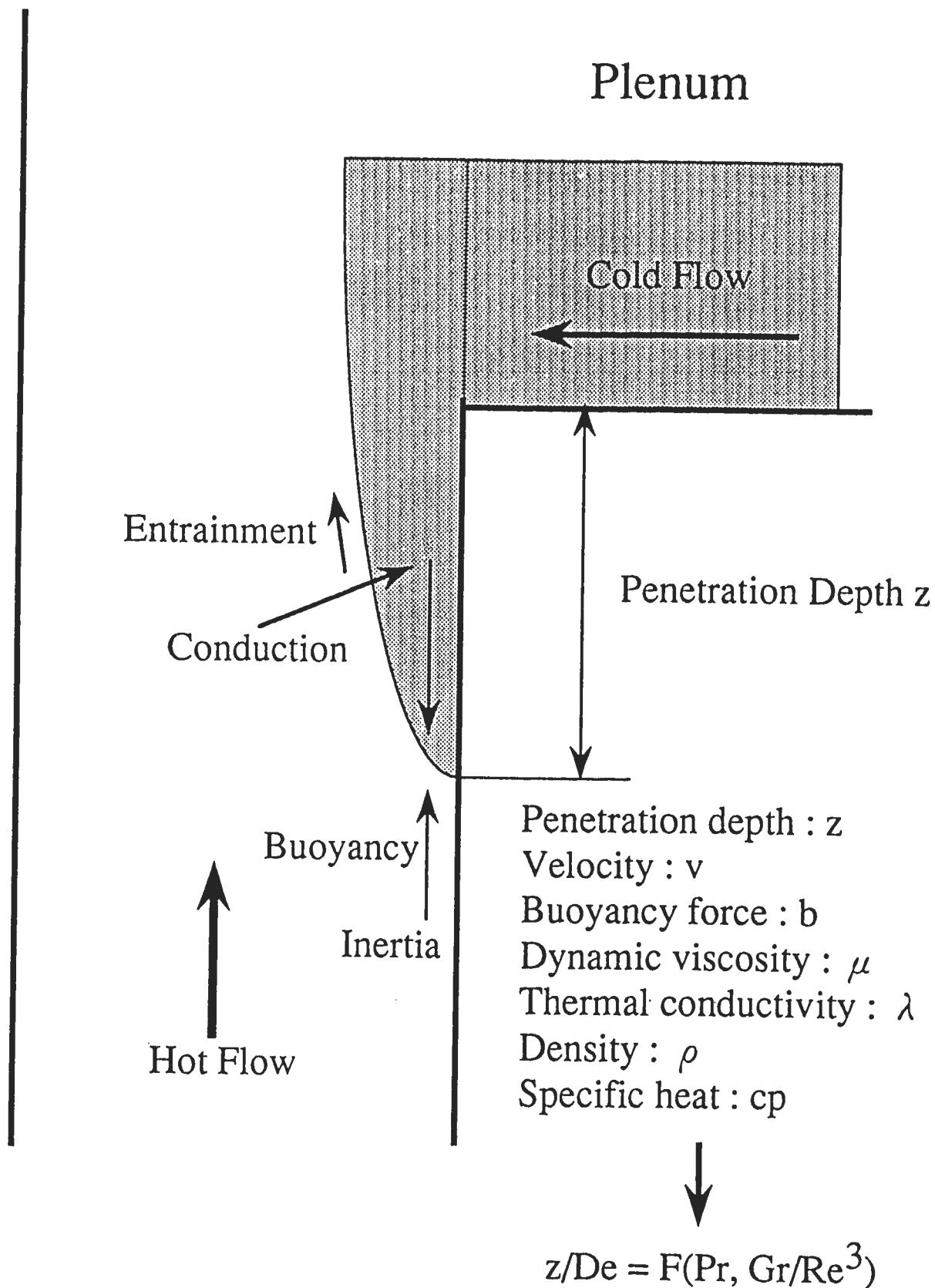


Fig. 18 Physical Mechanism of Penetration Flow

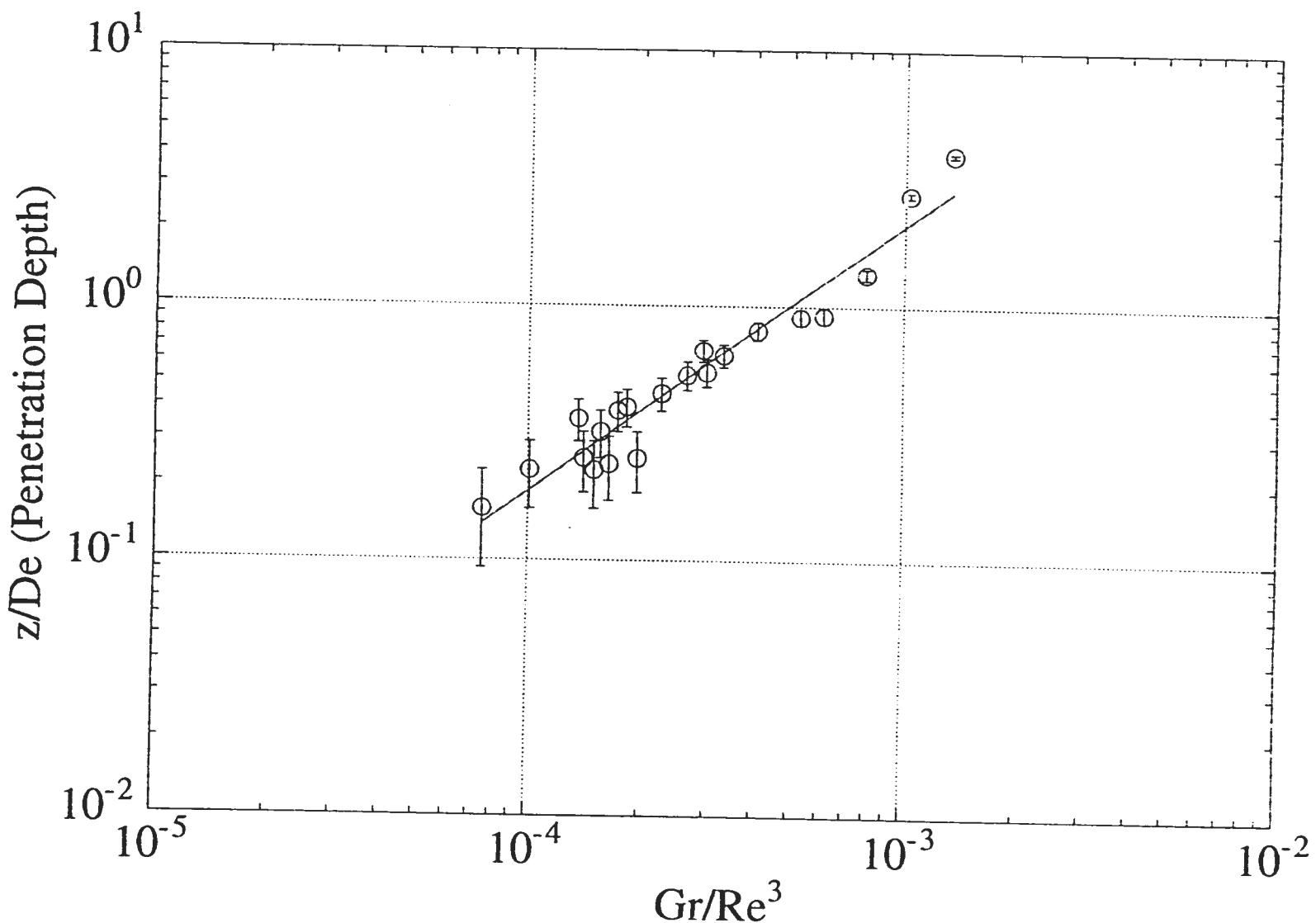


Fig. 19 Maximum Penetration Depth related to Gr/Re^3

付録

- A 1. 超音波流速分布測定装置計測パラメータ
- A 2. 超音波流速分布測定装置の性能
- A 3. 超音波流速分布測定装置データファイルの内部構造
- A 4. 超音波流速分布測定装置データ処理ツール (U D P. E X E)
- A 5. 潜り込み深さの支配パラメータ導出に至る経緯
 - (1) 地表付近の空気流れとのアナロジによる考察
 - (2) 流速プロファイルの時間変化から潜り込み流れの先端を探索する方法
 - (3) パワースペクトルによる現象把握
- A 6. 水流動第2試験室・中央制御室の空調設備に関する問題点及び改善措置
- A 7. U D P. E X E ソースリスト

A 1. 超音波流速分布測定装置計測パラメータ

(1) Starting depth

トランスデューサ表面と計測する最初の点との距離。計測開始位置を表す。壁面やトランスデューサを設置することによって流れが乱される領域をこの値を調整することによって排除することができる。

(2) Channel distance

チャンネル間距離。計測点の距離を表す。この値を修正して、計測レンジを変える。

(3) Sound velocity

流体の種類による。音速は一般的に温度等に若干影響されるが、特定の1つの値を与えている。この値は他のパラメータを計算する最の基本的な数値となる。水と異なる流体を使用する場合には適切な値を与える必要がある。水の場合は通常1480 [m/s] にセットする。

(4) Maximum depth

計測線上の最大計測長さ。超音波バースト信号のパルス反復周波数に直接関係している。流動状況による必要な計測幅に応じて変化させる。

(5) Storing interval

流速分布データの格納インターバル。1つの流速分布データの計測時間は固定であるが、メモリに格納されるデータ数は可変である。全計測時間を変化させることができる。

(6) RF gain factor

水と異なる流体を計測する場合、超音波の減衰係数も一般的には異なる。増幅の利得分布の勾配を、最も適切な値に修正することができる。

(7) US emission voltage

トランスデューサに与える電圧値。減衰の強度と流れ場の反射体の濃度に応じて、最適なデータを得られるように修正する。

(8) Channel width

チャンネル幅は空間分解能と等しく、固定であり、水の場合は0.75mmである。この値は音速に依存し以下の式で与えられる。

$$w = c * 1(\mu \text{ sec}) / 2$$

A 2. 超音波流速分布測定装置の性能

(1) 空間分解能と計測レンジ

空間分解能はチャンネル幅に等しく、以下の式によって決定される。

$$T_{sp} = c * 1(\mu \text{ sec}) / 2$$

これはチャンネル距離には直接関係していない。チャンネル距離は2つの計測

点の距離である。計測領域の長さはチャンネル距離を通して制御されるが、空間分解能は音速を変えない限り常に一定である。

(2) 時間分解能

1つの流速分布を計測する際の時間分解能は、データの取得時間によって決定され、それは F_{prf} (超音波バースト信号のパルス反復周波数) に依存する。以下の表に示すように、計測最大深さに対して変化する。

Maximum Depth	Time Resolution
95mm	16.4msec
189mm	32.8msec
375mm	65.5msec
758mm	131.1msec

ただし、最小の P_{max} (最大計測深さ、最速の F_{prf}) に対しては、前半の流速分布 (0 - 63 チャンネル) が上記の時間分解能で分析され、その後後半が分析される。従って、最大深さの実時間分解能は上記の値であるが、前半と後半のデータの間に瞬間計測の時間差が存在する。

(3) 計測時間

計測された複数の流速分布の時間差は原理的には上記の時間分解能と等しい。しかし、実時間でデータ解析とデータ表示を行なうためには、データの分析のための処理の時間が必要である。そのため、計測時間は以下の表に示すように、データの取得とデータ分析の時間によって定義する。データの分析に必要な時間はデータの内容にわずかに依存している。しかし、それぞれの時間分解能に対するこの時間変動はせいぜい 0.1 %未満である。

Maximum Depth	Average measuring Time
95mm	37.2msec
189mm	37.2msec
375mm	69.8msec
758mm	135.3msec

(4) サンプリングスピード

装置は流速分布データの格納をスキップすることができ (Storing interval parameter ; m) 、格納された流速分布データの時間間隔はサンプリングスピードと等価である。従って、記録されるデータのサンプリングスピードは次式によつて計算される。

$$T_{samp} = (m + 1) * \text{Measuring Time}$$

A 3. 超音波流速分布測定装置データファイルの内部構造

すべてのデータはバイナリデータであるため、バイナリモードでオープンする必要がある。

1. Parameter Block 00 00 00

..... (512 bytes) [parameter data]

2. Data Block 00 00 00
 (1024 * 128 bytes) [profile data]
 (Total 131,584 Bytes)

(1) パラメータブロック

パラメータデータは以下の形式で与えられる。それぞれのデータは2バイト占有する。

00-01	Version number of the machine software
02-03	Channel distance (micron)
04-05	Position of the first channel
06-07	Window-end position
08-09	Sound velocity (m/sec)
0A-0B	Storing interval
0C-0D	0 = OFF, 1 = ON [Trigger mode]
0E-0F	0 = OFF, 1 = Floppy, 2 = Sefial
10-11	(internal use)
12-13	(internal use)
14-15	(internal use)
16-17	(internal use)
18-19	Basic frequency of ultrasound (kHz)
1A-1B	Fprf (Hz)
1C-1D	US emission voltage (V)
1E-1F	(internal use)
20-21	(internal use)
22-23	(internal use)
24-25	(internal use)
26-27	Screen scale factor

(2) データブロック

プロファイルデータは1点のデータあたり1バイト占有する。データの並びは、データブロックの先頭が計測開始直後のプロファイルとなる。それ以降は計測順に格納されている。

A 4. 超音波流速分布測定装置データ処理ツール (U D P. E X E)

このソフトウェアツールは、U V Pによって得られたデータファイルを読み込み、流速データをテキストファイルに書き込み、別のアプリケーションプログラムで利用できるような形式で出力する機能のほか、平均値処理、周波数解析が行える機能を持つツールである。この全リストをA 7. U D P. E X Eソースリストに添付した。上記のデータファイルの構造に従い、メインモジュールのU D P. Cのopen_profile()でパラメータを読み込み、read_file()でメモリにデータを読み込んだ後各処理を行っている。

A 5. 潜り込み深さの支配パラメータ導出に至る経緯

本報告書では、潜り込み深さの支配パラメータは Gr/Re^3 および Pr 数としているが、この結論を得るまでに他の潜り込みに対する考察も行ったので、付録とする。

(1) 地表付近の空気流れとのアナロジによる考察

参考にした実験研究はコロラド州立大学にある気象研究所で行われた風洞実験である^{脚注1) 2)}。現象を支配する物理法則は、乱流における慣性力、密度勾配による浮力、壁面に及ぼすせん断力と考える。この物理モデルをFig. A1に示す。壁面に近い流れでは、壁面に近いところほど、流れの平均水平速度は小さくなる。流体の小さいエレメントが壁面と直角に移動すると、移動した新しい環境で自己の運動量を変化し、周囲のエレメントと同じ速度になろうとする。この現象では、水平方向の運動と鉛直方向の運動は互いに独立に考えなければならないので、水平方向と鉛直方向の2つの代表長さが必要となる。流体のエレメントの質量を m 、加速度を a 、水平方向の長さを L 、潜り込み深さを z 、流体の密度を ρ 、水平方向に距離 L 離れた場所における平均鉛直速度の差 Δu 、エレメントが水平方向に距離 L 移動するのに要する時間を t とし、すべて代表値である。エレメントの水平移動による鉛直方向の運動量変化を慣性力の代表値で表すと

$$\begin{aligned} F_i &= ma \\ &= \rho z^2 L a \\ &= \rho z^2 L \frac{\Delta u}{t} \end{aligned}$$

また、

$$\Delta u = \frac{z}{t}$$

$$\Delta u = \frac{\partial u}{\partial L} L$$

であるから、これらの関係をまとめて慣性力 F_i を書き直すと

$$F_i = \rho z^2 L^2 \left(\frac{\partial u}{\partial L} \right)^2$$

参考文献

- 1) 江守一郎著 模型実験の理論と応用 P211-214 (株) 技報堂出版
- 2) H. Chuang and J. E. Cermak, "Similarity of thermally stratified shear flows in the laboratory and atmosphere," The Physics of Fluids, Supplement Pt. 2, Vol. 10, 1967, pp. S255-S258

一方、周囲の密度とエレメントの密度の差は

$$\Delta \rho = \frac{\partial \rho}{\partial L} L$$

と書けるので、浮力 F_b は

$$\begin{aligned} F_b &= \Delta \rho z^2 L g \\ &= g z^2 L^2 \frac{\partial \rho}{\partial L} \end{aligned}$$

更に壁面との間に働く粘性力 F_v は

$$F_v = \mu z u$$

慣性力 = 浮力の場合は初生条件を表しており、

$$Ri = g \frac{\partial \rho}{\partial L} / \rho \left(\frac{\partial u}{\partial L} \right)^2 = \text{Const.}$$

となって第1報の結果と一致する。

また、潜り込み深さはこれらの3者のバランス（慣性力 = 浮力 + 粘性力）より

$$\left| \rho L^2 \left(\frac{\partial u}{\partial L} \right)^2 - g L^2 \frac{\partial \rho}{\partial L} \right| z^2 - \mu u z = 0$$

が成り立ち、これを z について解くと

$$z = \frac{\mu u}{\left\{ \rho \left(\frac{\partial u}{\partial L} \right)^2 - g \frac{\partial \rho}{\partial L} \right\} L^2}$$

となる。しかし、本試験では密度の水平方向の勾配を求めることができず、流速の水平方向の勾配を求めるのも困難であるため、この方法によっては潜り込み深さを求めていない。

(2) 流速プロファイルの時間変化から潜り込み流れの先端を探索する方法

UV P によって得られる瞬間の流速分布は、潜り込みを捉えた場合には Fig. A2 に示すように上昇流から下降流に変化する位置が存在する。この位置をある一定の規則に基づいてもとめ（0 または負の流速で、その位置より浅い位置において 3 点以上正の流速が連続している点）、その次系列を追うことによって潜り込みの先端の動きを探索する方法である。チャンネル入口平均流速が 3.0 [cm/sec]、および 4.0 [cm/sec] のケースについての結果を Fig. A3(A), A3(B), A4(A), A4(B) に示す。Left はチャンネルの浅い方から探索したもの、Right は深い方から

探索したものである。潜り込みの先端が最も深いものを最大潜り込み深さとして、両者（RightとLeft）を比較すると、Rightの方が深く潜り込んでいるが、この理由は計測線上において下降流の生じている部分が2カ所以上ある場合があるためである。従って、最も深く潜り込んでいる位置はRightの方に現れる。Rightから求めた深さと、本文中で用いた正の流速と負の流速を別々に平均値をとる方法と比較するとほぼ同じ結果となった（Table A1）。この方法は、物理的な考察を行う場合、潜り込み深さの定義の物理的な意味を考えやすい利点があるが、すべてのケースの実験が行われていないため、この方法からは結論が得られない。

（3）パワースペクトルによる現象把握

UV Pによって得られる流速分布から、各位置の流速の時間変化をフーリエ変換することによってパワースペクトルを求めた。潜り込みが頻繁に発生し、そして深く潜り込むケースでは、プレナム底面付近で大きなピークが見られるが、計測領域のパワーが一定であるため、潜り込みの頻度が少ないケースではピーク値が減少すると共に乱流の幅広いスペクトルがチャンネル入口に向かって広く分布してくる。結果をFig. A5, A6, A7, A8に示す。このように定性的には潜り込み現象を把握することができるが、定量的な把握や、潜り込み深さに結びつけることが困難であるため、考察を中断した。

Table A1 Maximum Penetration Depth

	From Averaging Data	From Time Series Data
Case - 1	342 [mm]	340 [mm]
Case - 2	223 [mm]	220 [mm]
Case - 3	103 [mm]	94 [mm]
Case - 4	67 [mm]	62 [mm]

Plenum

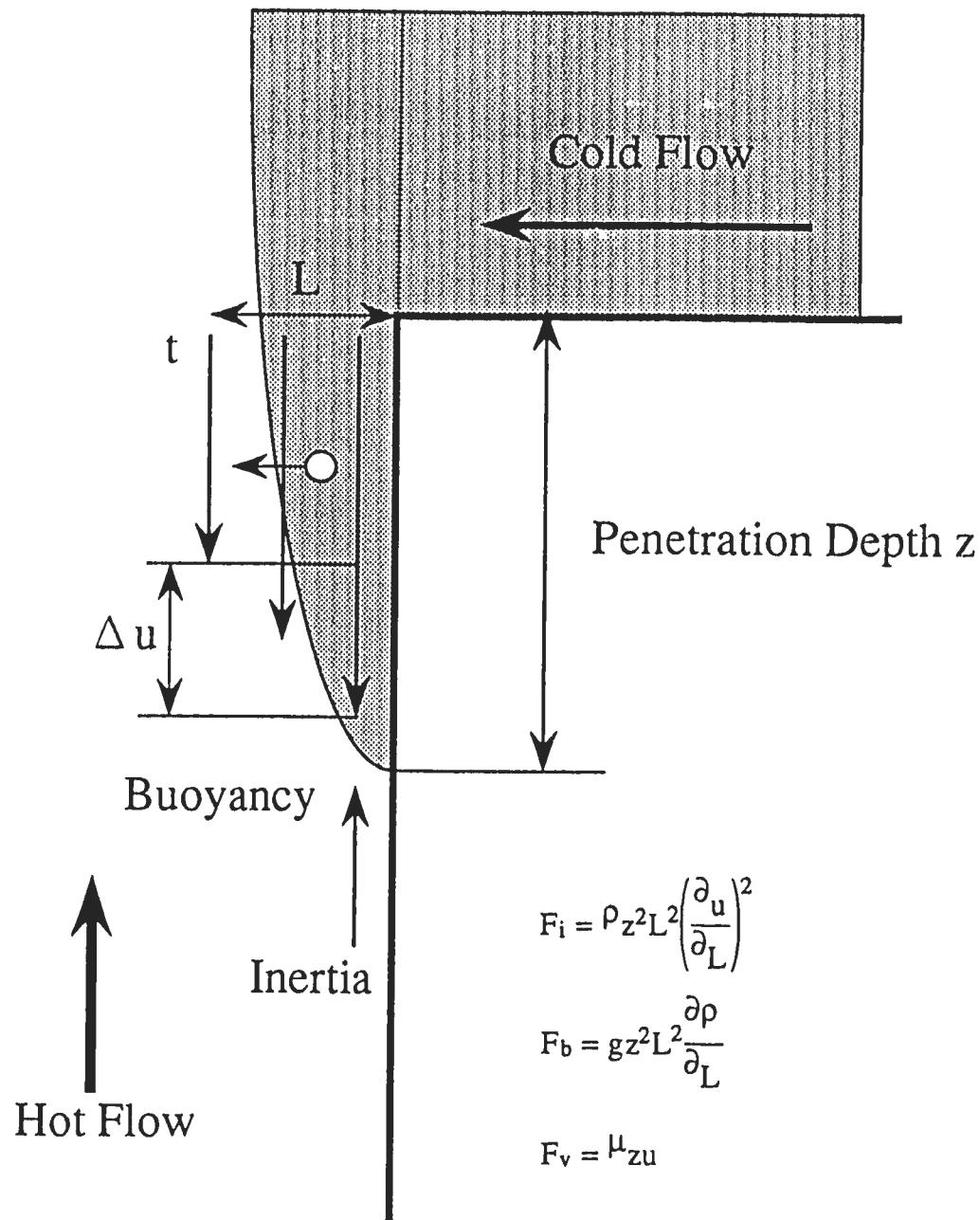


Fig. A1 Physical Mechanism of Penetration Flow

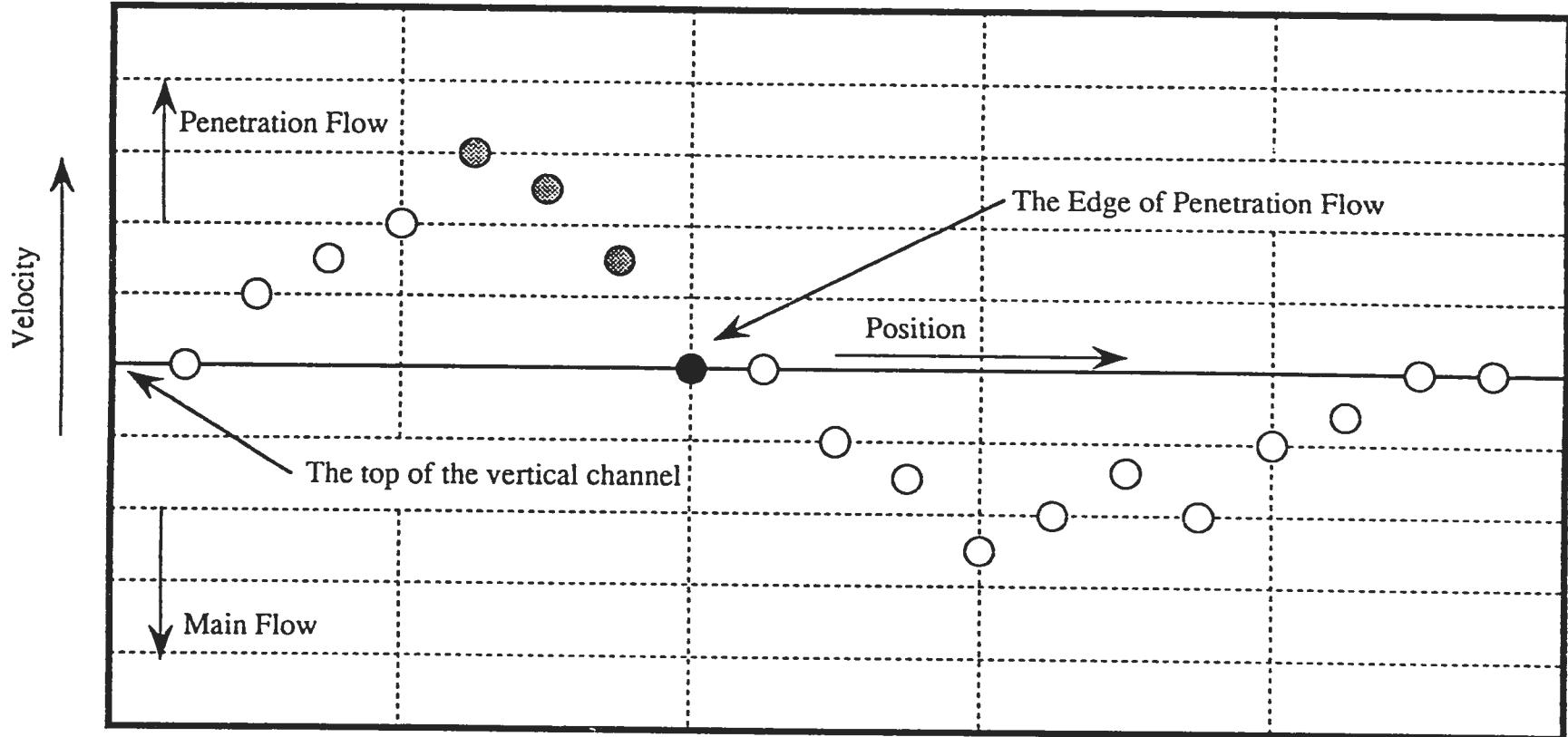


Fig. A2 The Edge of Penetration Flow

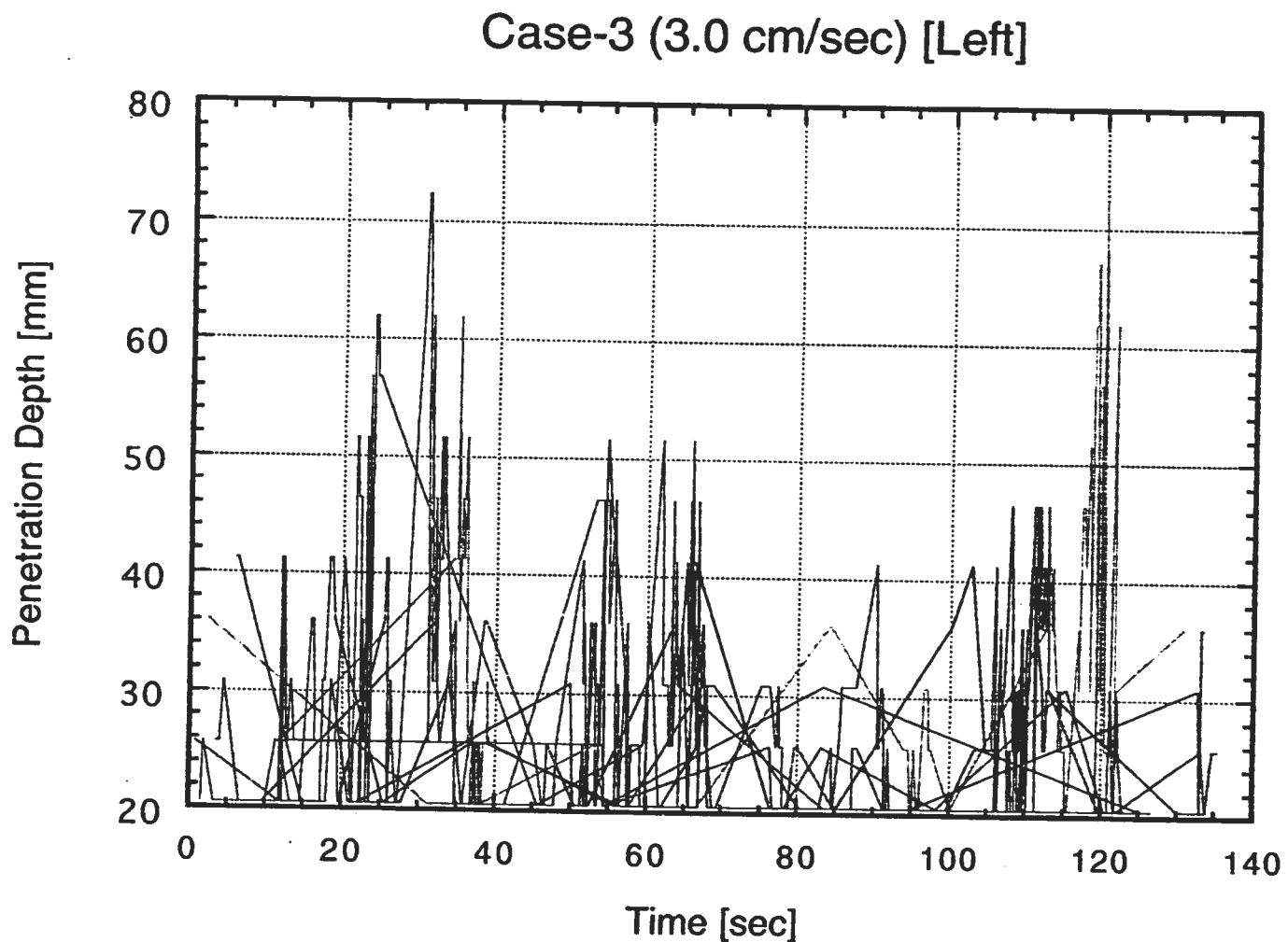


Fig. A3(A) Time series of the Edge of Penetration Flow for Case-3 (Left)

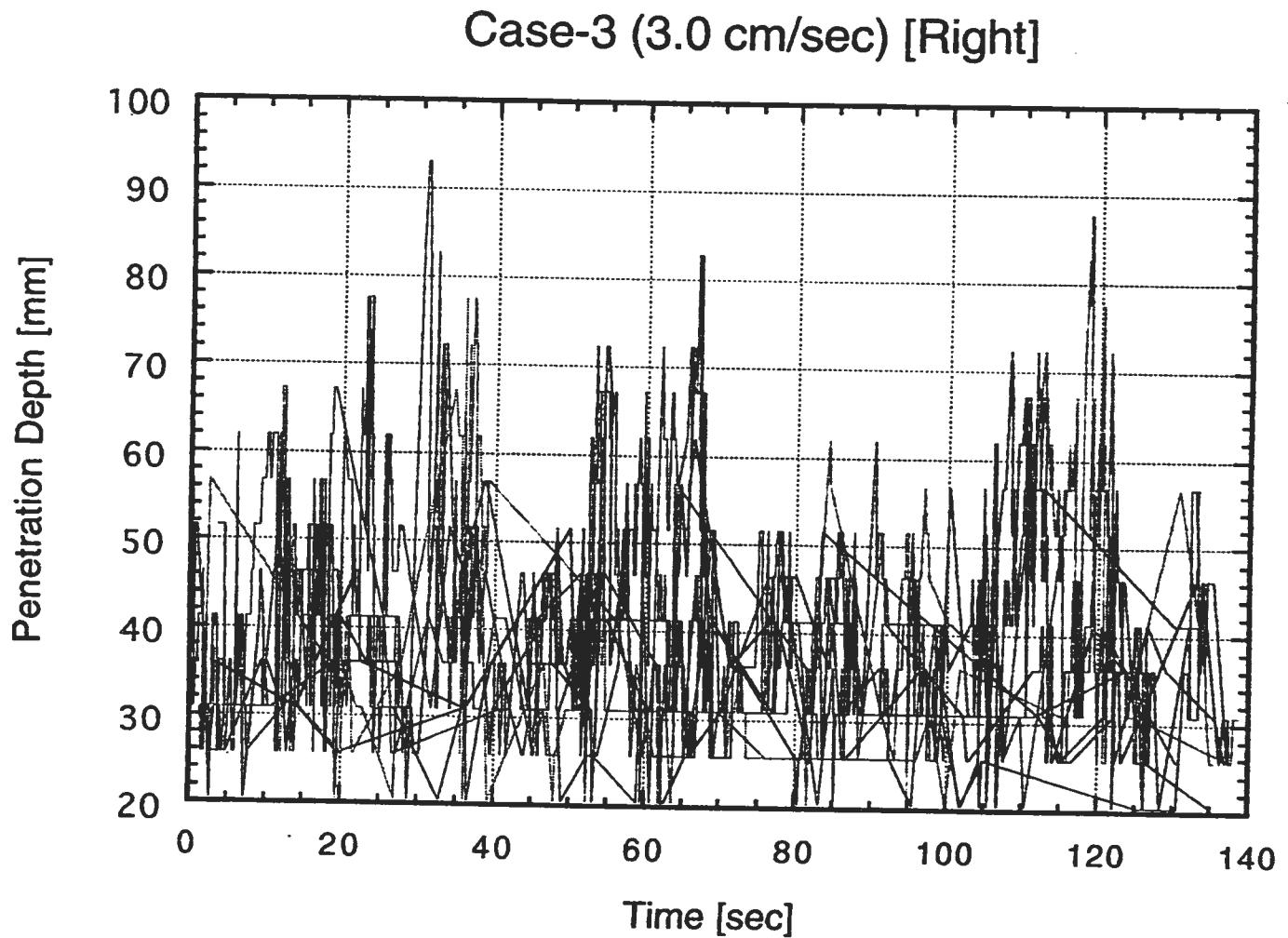


Fig. A3(B) Time series of the Edge of Penetration Flow for Case-3 (Right)

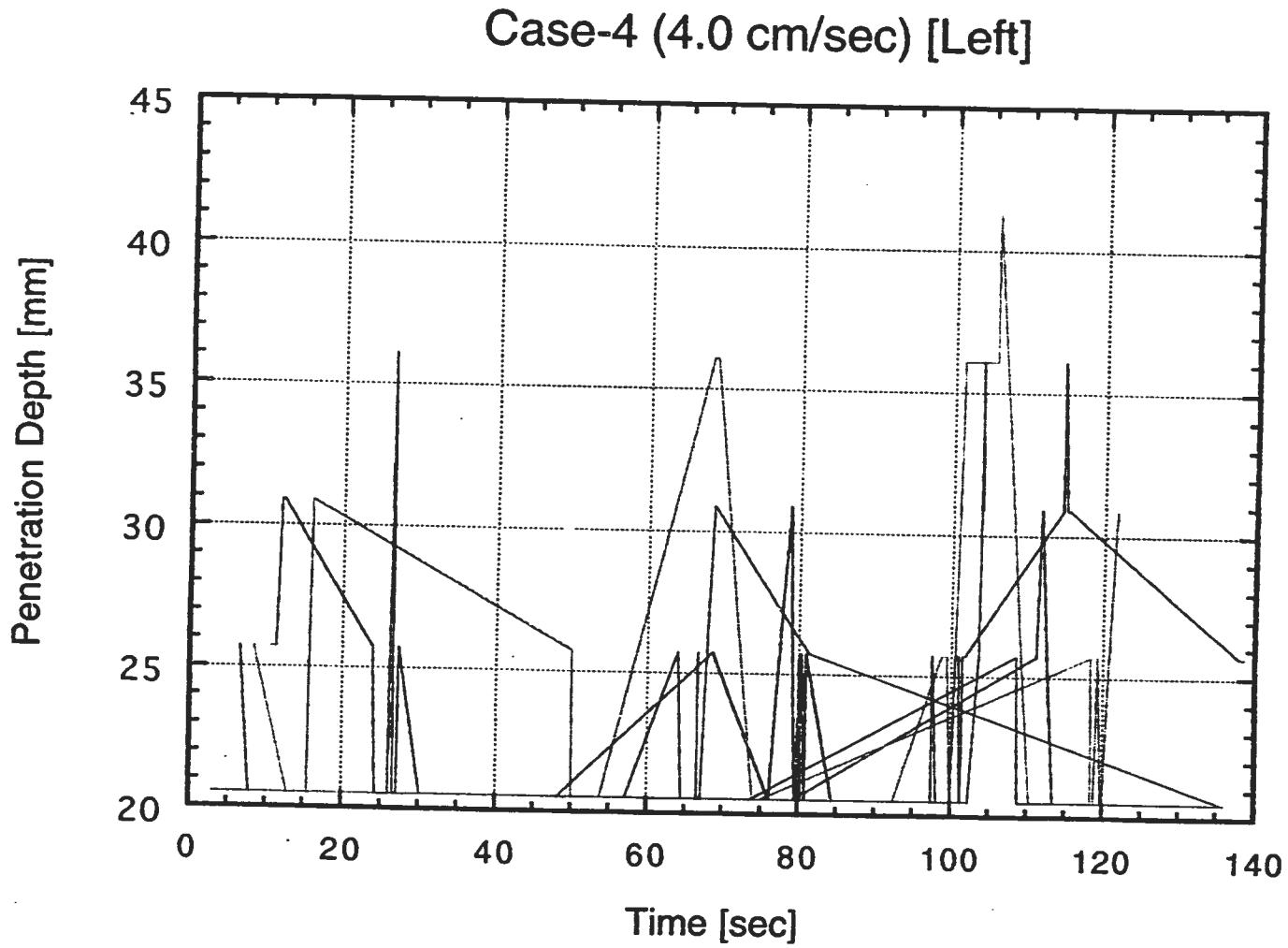


Fig. A4(A) Time series of the Edge of Penetration Flow for Case-4 (Left)

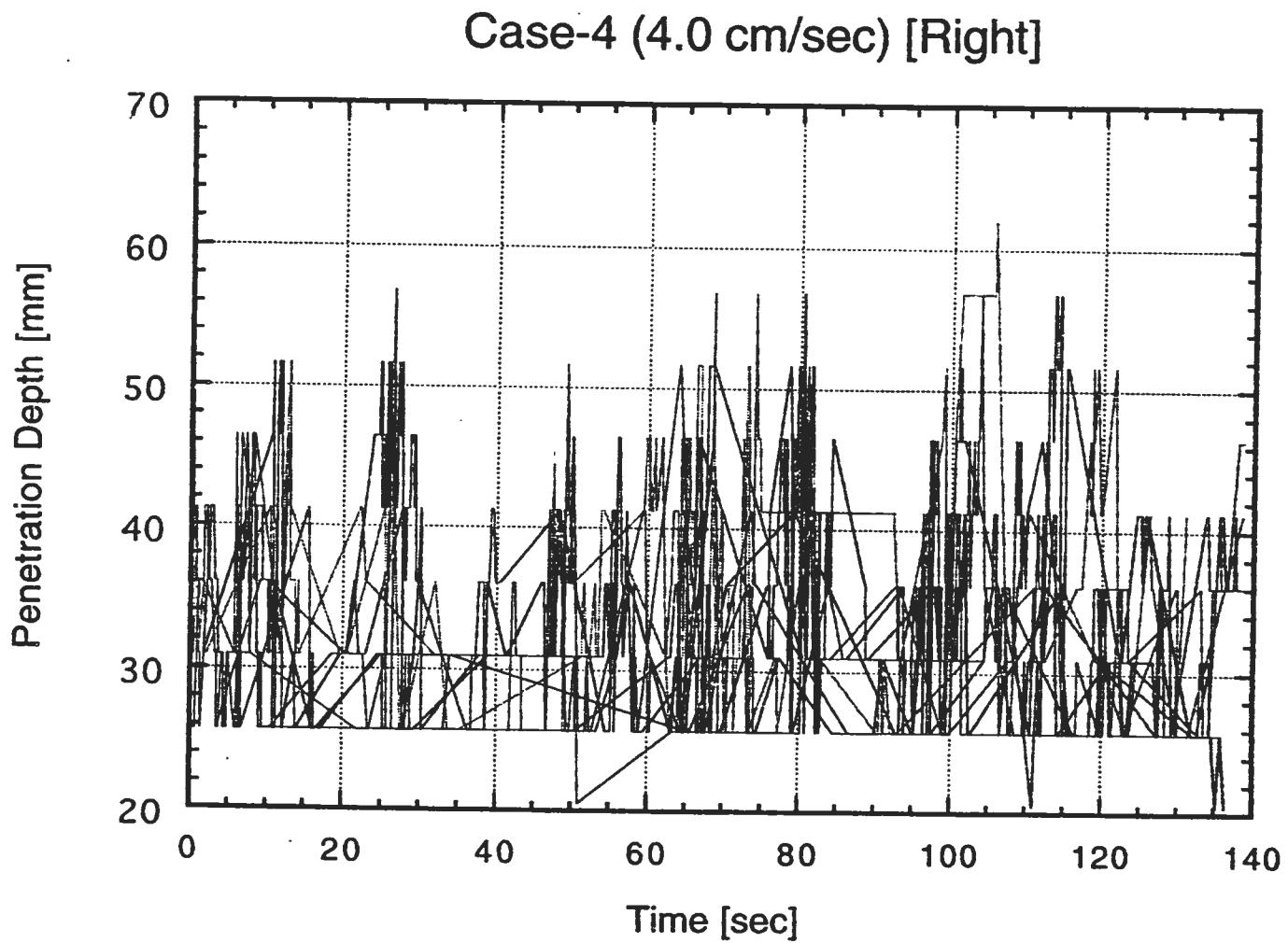


Fig. A4(B) Time series of the Edge of Penetration Flow for Case-4 (Right)

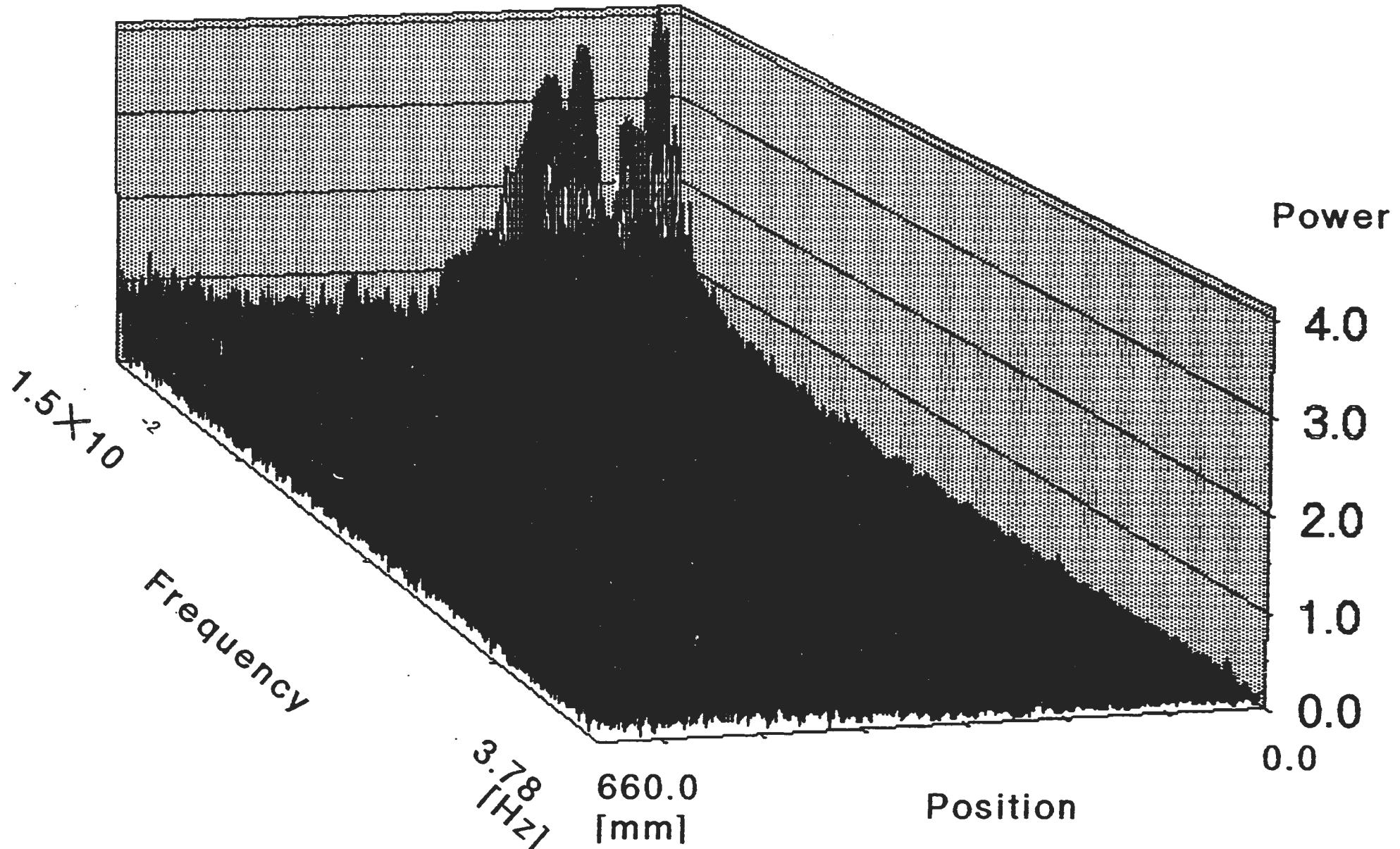


Fig. A5

Power Spectrum Distribution of Penetration Flow for Case-1

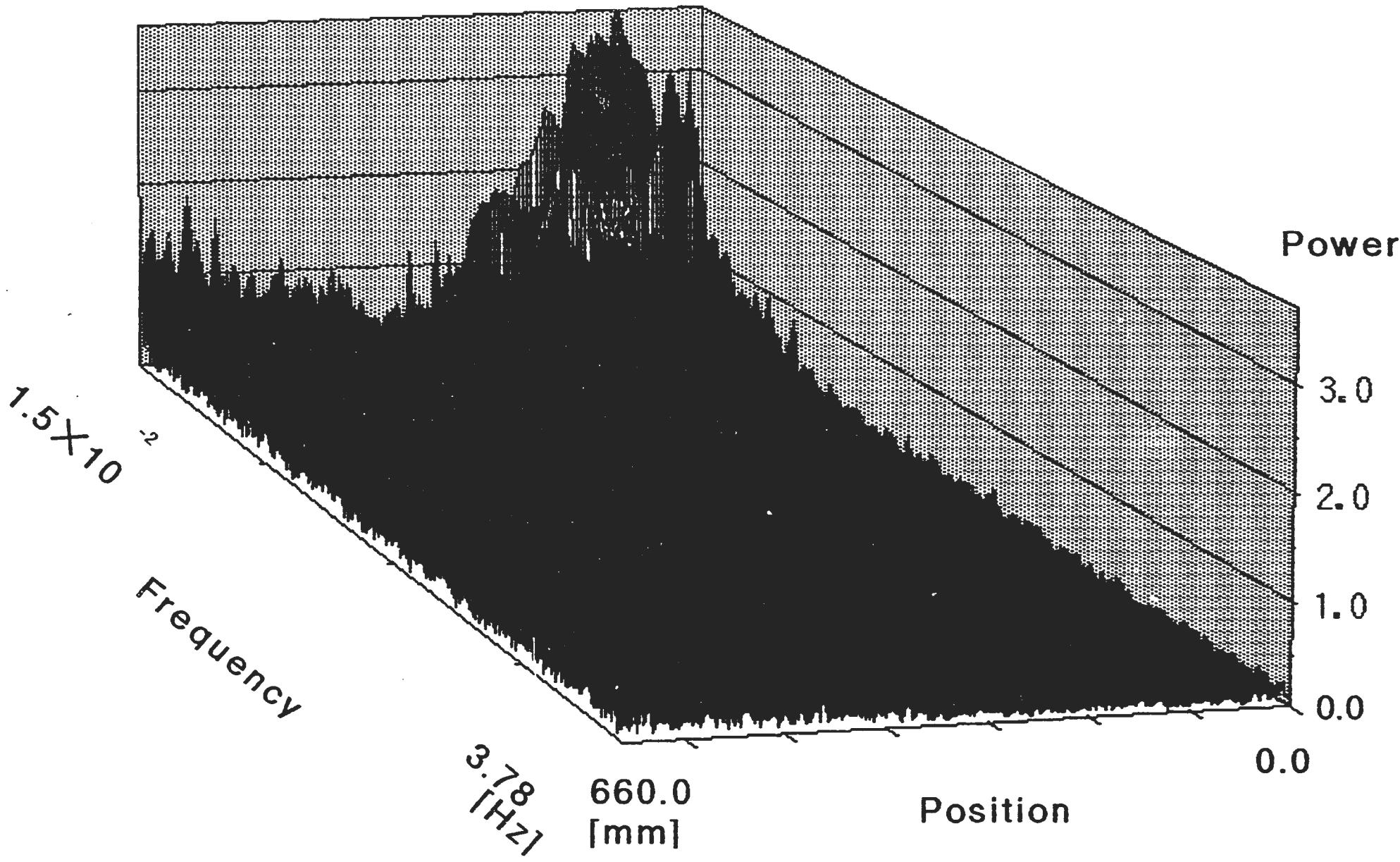


Fig. A6

Power Spectrum Distribution of Penetration Flow for Case-2

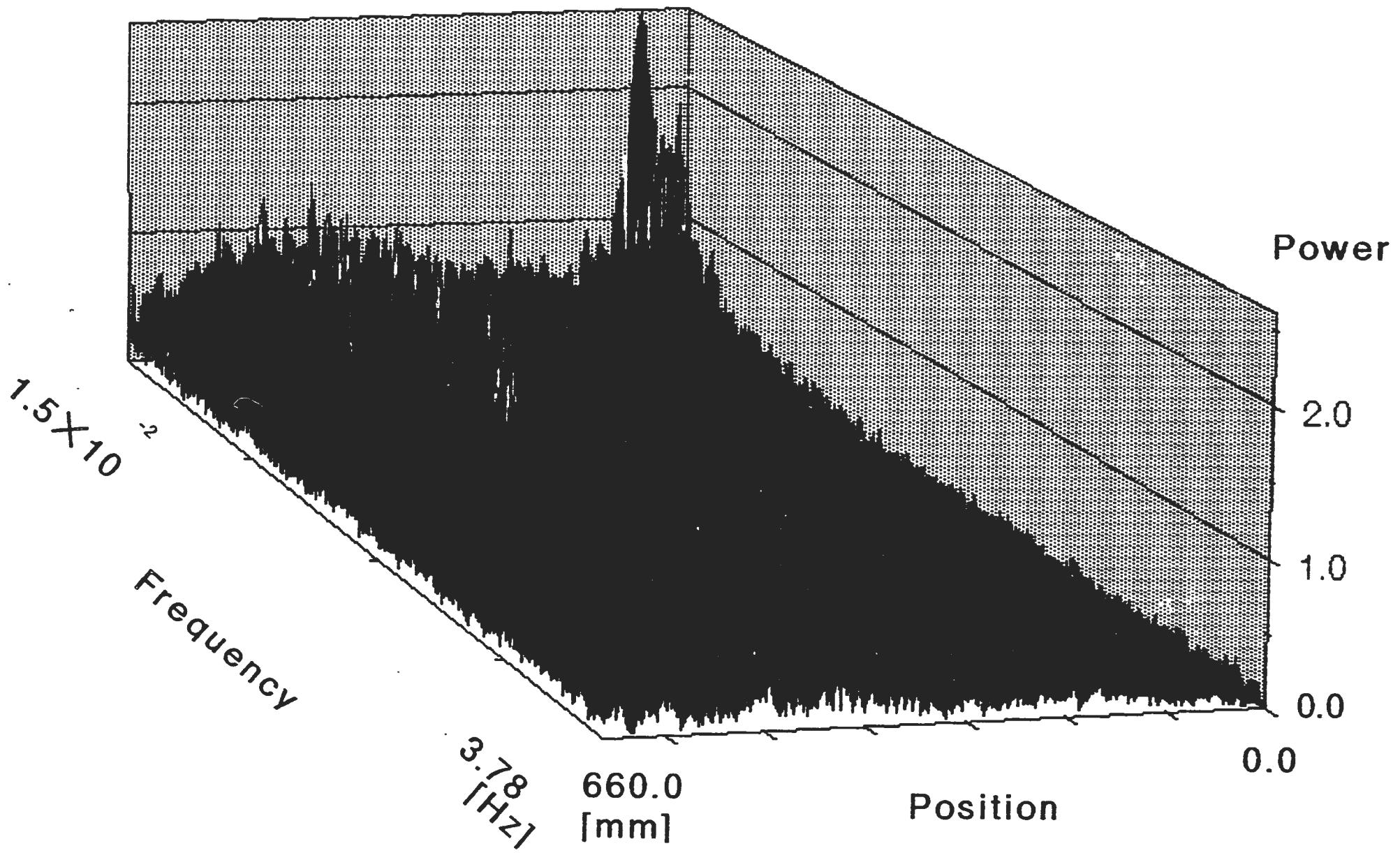


Fig. A7

Power Spectrum Distribution of Penetration Flow for Case-3

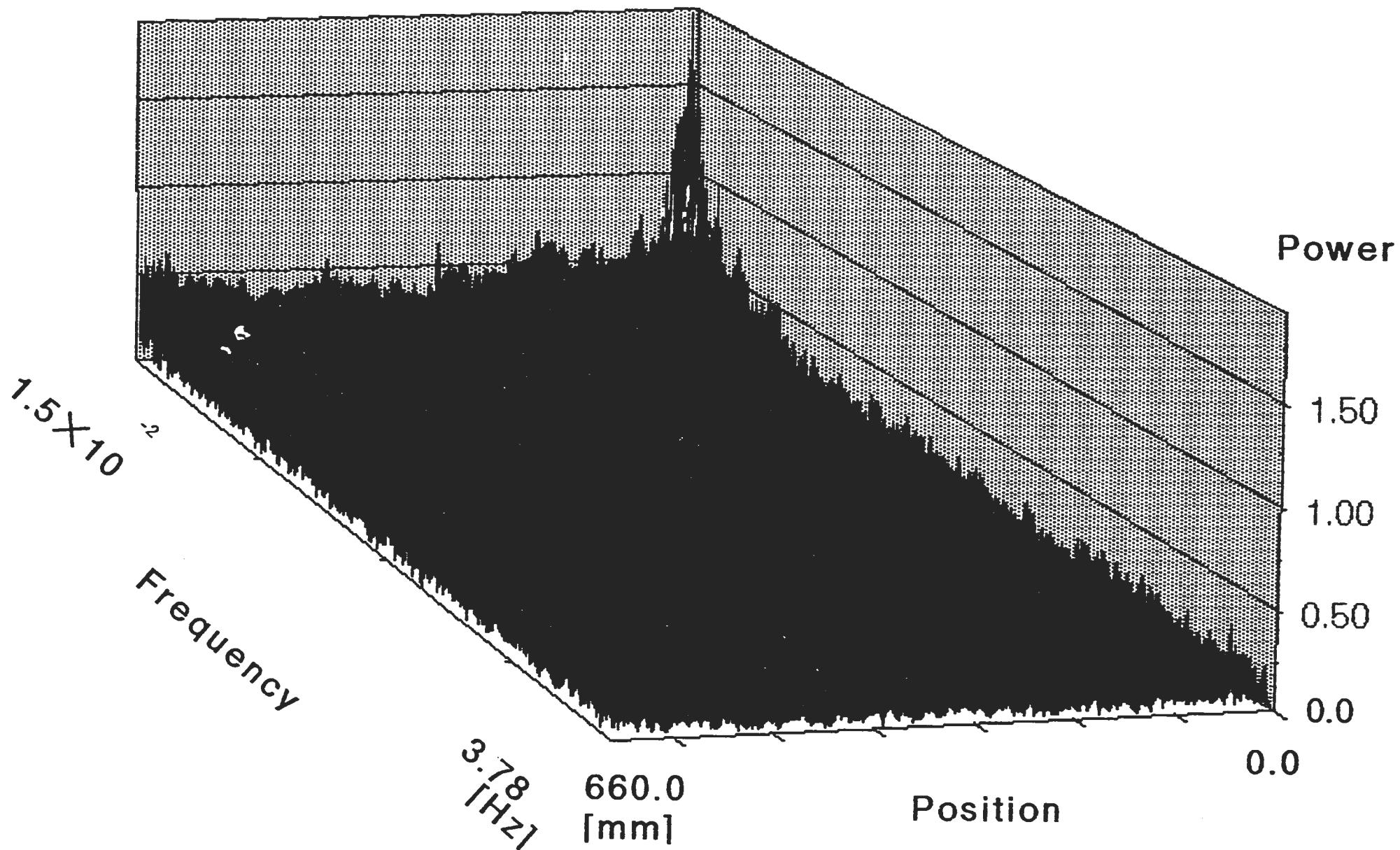


Fig. A8 Power Spectrum Distribution of Penetration Flow for Case-4

A 6. 原子炉工学室・水流動第2試験室・中央制御室の 空調設備に関する問題点及び改善措置

本実験は原子炉工学室内水流動第2試験室において行われ、温度データは同中央制御室にて収録された。その際に異常な温度指示変動が見られたため、その原因究明にあたった。その結果をここに報告する。

[要旨] 温度データの収録にアナログデバイス社3B37シグナル・コンディショニングI/Oサブシステムを使用した際に温度指示チャンネルに異常な変動が見られたためにその原因を究明した。

[症状] 試験データに約30分間隔で最大±0.5°C程度の連続的な変動が不定期に確認された。更に24時間周期で±0.2°C程度の変動も確認された。

[変動原因の特定]

1. 電圧変動に関しては8:30、17:00に確認されたが、今回の指示変動は周期的なものであるから無関係であると結論した。
2. 電源波形は正常であった。
3. エアコンによる室温変化と指示変動には同調性が見られた。

[変動原因] シグナルコンディショナ零接点補償用温度センサがエアコンによる室温変動の影響を受けたためで、インバータ無しのON/OFF制御のため周期変動が温度データに現れたものと見られる。

[改善措置] 温度センサへの影響を軽減させるためラック全体をビニールで覆い、ラック下部に吸気、上部に排気口を設け、ダクトによりラックない空気を制御室外へ排出させた。

[確認] 改善後シグナルコンディショナへの影響が改善されたことを60時間に亘りデータを収録し、周期変動のないことを確認した。

A 7. UDP. EXE ソースリスト

超音波流速分布測定装置データ処理ツールの使用方法及びソースリストを示す。機能の選択はコマンドラインでのオプション指定で行なう。各オプションの意味はソースリスト中で行なっている。また使用方法は以下に使用例を示す。

(使用例)

REM 1. データ表示

```
UDP/P COPIES.001 > B:\UDPY_P.DAT
UDP/I COPIES.001 B:\UDPY_I1.DAT B:\UDPY_I2.DAT
UDP/D COPIES.001 > B:\UDPY_D.DAT
UDP/TV 64 COPIES.001 > B:\UDPY_TV.DAT
UDP/PV 512 COPIES.001 > B:\UDPY_PV.DAT
```

REM 2. 平均値処理

```
UDP/A COPIES.001 > B:\UDPY_A.DAT
UDP/Z COPIES.001 > B:\UDPY_Z.DAT
UDP/O COPIES.001 > B:\UDPY_O.DAT
```

REM 3. 周波数解析

```
UDP/T 64 COPIES.001 > B:\UDPY_T.DAT
UDP/TM 64 COPIES.001 > B:\UDPY_TM.DAT
UDP/TN 64 COPIES.001 > B:\UDPY_TN.DAT
UDP/S 512 COPIES.001 > B:\UDPY_S.DAT
UDP/SM 512 COPIES.001 > B:\UDPY_SM.DAT
UDP/SN 512 COPIES.001 > B:\UDPY_SN.DAT
UDP/T 128 COPIES.001 > B:\UDPY_T128.DAT
UDP/S 1024 COPIES.001 > B:\UDPY_S1024.DAT
```

```
*****
*      超音波流速分布測定装置データ処理ツール
*      --- UVPI Data Processing tool ---
*      UDP.C Ver 1.0
*.
* 1. データ表示
*(1)/P [INPUT] -> CON : パラメータファイル出力
*(2)/I [INPUT] [OUTPUT1] [OUTPUT2] : 整数データ出力
*(3)/D [INPUT] -> CON : 流速データ表示
*(4)/TV (0..127) [INPUT] -> CON : 任意位置の流速時間変化出力
*(5)/PV (0..1023) [INPUT] -> CON : 任意時間の瞬間流速分布出力
* 2. 平均値処理
*(1)/A [INPUT] -> CON : ALL Data
*(2)/Z [INPUT] -> CON : Zero Supressed
*(3)/O [INPUT] -> CON : Only +- (Zero Supressed)
* 3. 周波数解析
*(1)/T (0..127) [INPUT] -> CON : パワースペクトル・矩形窓・位置
*(2)/TM (0..127) [INPUT] -> CON : ハミング窓
*(3)/TN (0..127) [INPUT] -> CON : ハニング窓
*(4)/S (0..1023) [INPUT] -> CON : エネルギスペクトル・矩形窓・時間
*(5)/SM (0..1023) [INPUT] -> CON : ハミング窓
*(6)/SN (0..1023) [INPUT] -> CON : ハニング窓
*(7)/Option (128, 1024) [INPUT] -> CON : ALL Data
*.
* 参考文献：Cによる科学技術計算（基底2の周波数間引型FFT関数）
*.
* MAIN MODULE : UDP.C
* SUB1    : DISP.C
* SUB2    : AVE.C
* SUB3    : FFT.C
* INCLUDE FILE : FFT.H
* ERROR.FILE : ERROR.DAT
*.
* CL.EXE /AC /Od /W2 /F 2000 UDP.C DISP.C AVE.C FFT.C
******/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <malloc.h>
#include <string.h>

extern int f_disp0(struct parameter *);
extern int f_disp1(char *, char *);
extern int f_disp2(struct parameter *);
extern int f_disp3(struct parameter *, char *, int);
extern int f_ave(struct parameter *, char *);
extern int f_fft(struct parameter *, char *, int);
```

```

struct parameter {
    char sprof[40];      /* Data File Name */
    double ch_distance;   /* Channel Distance */
    double start_ch;     /* Position of 1st Channel */
    double end_ch;       /* Window End Position */
    int fprf;            /* Fprf */
    int s_velocity;      /* Sound Velocity */
    int s_interval;      /* Storing Interval */
    char t_mode[4];      /* Trigger Mode */
    char ar_mode[7];     /* Auto Record Mode */
    int e_volt;          /* US Emission Voltage */
    int rf_gain;          /* RF Gain Start Value */
    double d_coef;        /* Doppler Coefficient */
    double s_coef;        /* Speed Coefficient */
    double vel_conv;      /* Velocity Conv. */
};

int huge *prof; /* Pointer for profile data */
FILE *fp;      /* Profile Data File */

int p_error(void)
{
    FILE *fo;
    int c;

    if (NULL == (fo = fopen("ERROR.DAT", "r"))) {
        fputs("ERROR.DAT NOT EXIST.", stderr);
        exit(1);
    }

    while (EOF != (c = fgetc(fo))) {
        putc(c, stderr);
    }

    fclose(fo); return 0;
}

int open_profile(struct parameter *prm)
{
    int buf[256];
    int wk1, wk2;

    if (NULL == (fp = fopen(prm->sprof, "rb"))) {
        fprintf(stderr, "%s Open Failed.\n", prm->sprof);
        exit(1);
    }

    fread(buf, 1, 512, fp);
    wk1 = buf[1]; /* win_int */
}

```

```

wk2 = buf[4]; /* vit_son */
prm->ch_distance = (double)(wk1 + 1)*(double)wk2*0.001/2.0;

wk1 = buf[2]; /* win_dep */
prm->start_ch = (double)wk1;

wk1 = buf[3]; /* win_fin */
prm->end_ch = (double)wk1;

wk1 = buf[13]; /* frq_prf */
prm->frq_prf = wk1;

prm->s_velocity = wk2;

wk1 = buf[5]; /* sto_int */
prm->s_interval = wk1;

wk1 = buf[6]; /* trig_mod */
if(0 == wk1)
    strcpy(prm->t_mode, "OFF");
if(1 == wk1)
    strcpy(prm->t_mode, "ON");

wk1 = buf[7]; /* rec_mod */
if(0 == wk1)
    strcpy(prm->ar_mode, "OFF");
if(1 == wk1)
    strcpy(prm->ar_mode, "DISK");
if(2 == wk1)
    strcpy(prm->ar_mode, "SERIAL");

wk1 = buf[14]; /* val_tens */
prm->e_volt = wk1;

wk1 = buf[11]; /* val_tgc */
prm->rf_gain = wk1;

wk1 = buf[8] & 0x00FF; /* val_prf */
if (wk1 == 0) wk2 = 7812;
if (wk1 == 0x40) wk2 = 3915;
if (wk1 == 0x80) wk2 = 1953;
if (wk1 == 0xC0) wk2 = 976;

prm->d_coef = (double)wk2/(2.0*128.0);
prm->s_coef = (double)prm->s_velocity*1000.0/(2.0*4.0E+06);
prm->vel_conv = prm->d_coef*prm->s_coef;

/* Display Parameter */
fprintf(stderr, " DATA FILE --- %s --- \n", prm->fprof);
fprintf(stderr, " Channel Distance = %6.2f [mm]\n", prm->ch_distance);

```

```

        sprintf(stderr, " Position of 1st Channel = %6.2f [mm]\n", prm->start_ch);
        sprintf(stderr, " Window End Position   = %6.2f [mm]\n", prm->end_ch);
        sprintf(stderr, " Fprf           = %6d [Hz]\n", prm->fprf);
        sprintf(stderr, " Sound Velocity     = %6d [m/s]\n", prm->s_velocity);
        sprintf(stderr, " Storing Interval    = %6d\n", prm->s_interval);
        sprintf(stderr, " Trigger Mode       = %6s\n", prm->t_mode);
        sprintf(stderr, " Auto Record Mode    = %6s\n", prm->ar_mode);
        sprintf(stderr, " US Emission Voltage = %6d [V]\n", prm->e_volt);
        sprintf(stderr, " RF Gain Start Value = %6d\n", prm->rf_gain);

        sprintf(stderr, " Doppler Coefficient  = %6.2f\n", prm->d_coef);
        sprintf(stderr, " Speed Coefficient    = %6.2f\n", prm->s_coef);
        sprintf(stderr, " Velocity Conv.      = %6.2f\n", prm->vel_conv);

        return 0;
    }

    int read_file()
    {
        int huge *pind;
        char dat[128];
        int i, j;

        if (NULL == (prof = (int huge *)malloc(128L*1024L, sizeof(int)))) {
            fputs("Memory Allocation Failed.", stderr);
            exit(1);
        }

        for (i = 0; i < 1024; i++) {
            fread(dat, 1, 128, fp);
            for (j = 0; j < 128; j++) {
                pind = prof + 128L*i + j;
                *pind = (int)dat[j];
            }
        }
        return 0;
    }

    int main(int argc, char *argv[])
    {
        struct parameter prm;
        char opt[3]; /* Option */
        char pn;      /* Process Number */
        int ch;

        if (argc == 1 || argc == 2) {
            p_error(); exit(1);
        }

        if (argc > 5) {

```

```

    p_error(); exit(1);
}

strcpy(opt, argv[1]); /* Option */
if (0 == strcmp(opt, "/P")) {
    if (3 != argc) {
        p_error(); exit(1);
    }
    pn = 0;
} else if (0 == strcmp(opt, "/I")) {
    if (5 != argc) {
        p_error(); exit(1);
    }
    pn = 1;
} else if (0 == strcmp(opt, "/D")) {
    if (3 != argc) {
        p_error(); exit(1);
    }
    pn = 2;
} else if (0 == strcmp(opt, "/TV") || 0 == strcmp(opt, "/PV")) {
    if (4 != argc) {
        p_error(); exit(1);
    }
    pn = 3;
} else if (0 == strcmp(opt, "/A") || 0 == strcmp(opt, "/Z") || 0 == strcmp(opt, "/O")) {
    if (3 != argc) {
        p_error(); exit(1);
    }
    pn = 4;
} else if (0 == strcmp(opt, "/T") || 0 == strcmp(opt, "/TM")) {
    if (4 != argc) {
        p_error(); exit(1);
    }
    pn = 5;
} else if (0 == strcmp(opt, "/TN") || 0 == strcmp(opt, "/S")) {
    if (4 != argc) {
        p_error(); exit(1);
    }
    pn = 5;
} else if (0 == strcmp(opt, "/SM") || 0 == strcmp(opt, "/SN")) {
    if (4 != argc) {
        p_error(); exit(1);
    }
    pn = 5;
} else {
    fputs("オプションの指定に誤りがあります。", stderr);
    exit(1);
}

/* Input File */

```

```
if (3 == pn || 5 == pn) {
    strcpy(prm.sprof, argv[3]); /* Input File Name */
    ch = atoi(argv[2]);        /* Channel */
} else {
    strcpy(prm.sprof, argv[2]); /* Input File Name */
}

/* Open Data File and Read Parameter Block */
open_profile(&prm);

switch (pn) {
    case 0:
        f_disp0(&prm);
        break;
    case 1:
        read_file();
        f_disp1(argv[3], argv[4]); /* No Need Parameter Value */
        break;
    case 2:
        read_file();
        f_disp2(&prm);
    case 3:
        read_file();
        f_disp3(&prm, opt, ch);
        break;
    case 4:
        read_file();
        f_ave(&prm, opt);
        break;
    case 5:
        read_file();
        f_fft(&prm, opt, ch);
        break;
}
hfree((char *)prof); fclose(fp); return 0;
}
```

```
*****
* 超音波流速分布測定装置データ処理ツール *
*   1. データ表示処理ルーチン          *
*   f_disp0(), f_disp1(), f_disp2()    *
*   DISP.C                            *
*****
```

```
#include <stdio.h>

struct parameter {
    char fprof[40];      /* Data File Name */
    double ch_distance;  /* Channel Distance */
    double start_ch;     /* Position of 1st Channel */
    double end_ch;       /* Window End Position */
    int fprf;            /* Fprf */
    int s_velocity;      /* Sound Velocity */
    int s_interval;      /* Storing Interval */
    char t_mode[4];      /* Trigger Mode */
    char ar_mode[7];     /* Auto Record Mode */
    int e_volt;          /* US Emission Voltage */
    int rf_gain;         /* RF Gain Start Value */
    double d_coef;       /* Doppler Coefficient */
    double s_coef;        /* Speed Coefficient */
    double vel_conv;     /* Velocity Conv. */
};

extern int huge *prof;
extern struct parameter *prm;

/* Parameter for "/P" */
int f_disp0(struct parameter *prm)
{
    /* Display Parameter */
    sprintf(stdout, " DATA FILE ... %s ... \n", prm->fprof);
    sprintf(stdout, " Channel Distance = %6.2f [mm]\n", prm->ch_distance);
    sprintf(stdout, " Position of 1st Channel = %6.2f [mm]\n", prm->start_ch);
    sprintf(stdout, " Window End Position = %6.2f [mm]\n", prm->end_ch);
    sprintf(stdout, " Fprf = %6d [Hz]\n", prm->fprf);
    sprintf(stdout, " Sound Velocity = %6d [m/s]\n", prm->s_velocity);
    sprintf(stdout, " Storing Interval = %6d\n", prm->s_interval);
    sprintf(stdout, " Trigger Mode = %6s\n", prm->t_mode);
    sprintf(stdout, " Auto Record Mode = %6s\n", prm->ar_mode);
    sprintf(stdout, " US Emission Voltage = %6d [V]\n", prm->e_volt);
    sprintf(stdout, " RF Gain Start Value = %6d\n", prm->rf_gain);

    sprintf(stdout, " Doppler Coefficient = %6.2f\n", prm->d_coef);
    sprintf(stdout, " Speed Coefficient = %6.2f\n", prm->s_coef);
    sprintf(stdout, " Velocity Conv. = %6.2f\n", prm->vel_conv);
}
```

```

    return 0;
}

/* Parameter for "/I" */
int f_disp1(char *p3, char *p4)
{
    FILE *fo;
    int i, j;
    int huge *pind;

    /* Output File VOL.1 */
    if (NULL == (fo = fopen(p3, "w"))) {
        fprintf(stderr, "%s Open Failed.\n", *p3);
        exit(1);
    }

    for (i = 0; i < 128; i++) {
        for (j = 0; j < 512; j++) {
            pind = prof + i + 128L*j;
            if (*pind != 0) fprintf(fo, "%4d\n", *pind);
            else      fprintf(fo, "%4d\n", 256);
        }
        fprintf(fo, "\n");
    }
    fclose(fo);

    /* Output File VOL.2 */
    if (NULL == (fo = fopen(p4, "w"))) {
        fprintf(stderr, "%s Open Failed.\n", *p4);
        exit(1);
    }

    for (i = 0; i < 128; i++) {
        for (j = 512; j < 1024; j++) {
            pind = prof + i + 128L*j;
            if (*pind != 0) fprintf(fo, "%4d\n", *pind);
            else      fprintf(fo, "%4d\n", 256);
        }
        fprintf(fo, "\n");
    }
    fclose(fo);
    return 0;
}

/* Parameter for "/D" (Velocity Data Output) */
int f_disp2(struct parameter *prm)
{
    int i, j;
    int huge *pind;
}

```

```

int idat;
double dat;

for (i = 0; i < 1024; i++) {
    for (j = 0; j < 128; j++) {
        pind = prof + 128L*i + j;
        idat = (int)(*pind);
        dat = (double)idat*(prm->vel_conv);
        sprintf(stdout, "%8.4f\n", dat);
    }
    fprintf(stdout, "\n");
}
return 0;
}

/* Parameter for "/TV" and "/PV" */
int f_disp3(struct parameter *prm, char *opt, int ch)
{
    int i;
    int huge *pind;
    int idat;
    double dat;
    double position;
    double time;

    if (0 == strcmp(opt, "/TV")) { /* Time - Velocity */
        if (ch < 0 || ch > 127) {
            fputs("位置の指定が誤っています。", stderr);
            exit(1);
        }
        if (7812 == prm->fprf) time = 37.2E-03;
        else if (3906 == prm->fprf) time = 37.2E-03;
        else if (1953 == prm->fprf) time = 69.8E-03;
        else if (976 == prm->fprf) time = 135.3E-03;
        else {
            fputs("Value of Fprf is incorrect.", stderr);
            exit(1);
        }
        sprintf(stdout, " Ch%t Time%t Velocity%t\n");
        for (i = 0; i < 1024; i++) {
            pind = prof + 128L*(long)i + (long)ch;
            idat = (int)(*pind);
            dat = (double)idat*prm->vel_conv;
            sprintf(stdout, "%6d%t%9.4f%t%9.4f\n", i, (double)i*time, dat);
        }
    } else { /* Position - Velocity */
        if (ch < 0 || ch > 1023) {
            fputs("時間の指定が誤っています。", stderr);
            exit(1);
        }
    }
}

```

```
fprintf(stdout, " Ch%t Position%t Velocity%t\n");
for (i = 0; i < 128; i++) {
    pind = prof + (long)i + 128L*(long)ch;
    idat = (int)(*pind);
    dat = (double)idat*prm->vel_conv;
    position = (prm->start_ch) + (double)i*(prm->ch_distance);
    sprintf(stdout, "%6d%t%9.4f%t%9.4f\n", i, position, dat);
}
}
return 0;
}
```

```

*****+
* 超音波流速分布測定装置データ処理ツール *
*   2. 平均値処理ルーチン *
*     f_ave()
*     AVE.C
*****/

```

```

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

struct parameter {
    char fprof[40];      /* Data File Name */
    double ch_distance;  /* Channel Distance */
    double start_ch;     /* Position of 1st Channel */
    double end_ch;       /* Window End Position */
    int fprf;            /* Fprf */
    int s_velocity;      /* Sound Velocity */
    int s_interval;      /* Storing Interval */
    char t_mode[4];      /* Trigger Mode */
    char ar_mode[7];     /* Auto Record Mode */
    int e_volt;          /* US Emission Voltage */
    int rf_gain;          /* RF Gain Start Value */
    double d_coef;        /* Doppler Coefficient */
    double s_coef;        /* Speed Coefficient */
    double vel_conv;      /* Velocity Conv. */
};

extern int huge *prof;
extern struct parameter *prm;

int f_state(double *dat, double *ave, double *sd, int n)
{
    int i;
    double wk1, wk2;
    double *p;

    p = dat;
    wk1 = 0.0;
    for (i = 0; i < n; i++) {
        wk1 = wk1 + *p;
        p++;
    }
    *ave = wk1/(double)n;

    p = dat;
    wk2 = 0.0;
    for (i = 0; i < n; i++) {

```

```

    wk1 = *p - *ave;
    wk1 = wk1*wk1;
    wk2 = wk2 + wk1;
    p++;
}
*sd = sqrt(wk2/(double)(n - 1));

return 0;
}

int f_ave(struct parameter *prm, char *opt)
{
    int huge *pind;
    int i, j;
    double position;

    int idat;
    double p_ave[128], p_sd[128]; /* for (+) Data */
    double *dat;
    double *p;
    double ave, sd;
    int ct;

    double m_ave[128], m_sd[128]; /* for (-) Data */
    double *dat2;
    double *p2;
    int ct2;

    if (NULL == (dat = (double *)malloc(1024*sizeof(double)))) {
        fputs("Memory Allocation Failed.", stderr);
        exit(1);
    }

    if (0 == strcmp(opt, "/A")) {
        for (i = 0; i < 128; i++) {
            p = dat;
            for (j = 0; j < 1024; j++) {
                pind = prof + (long)i + 128L*(long)j;
                idat = (int)(*pind);
                *p++ = (double)idat*(prm->vel_conv);
            }
            f_state(dat, &ave, &sd, 1024);
            p_ave[i] = ave;
            p_sd[i] = sd;
        }
        free((char *)dat);
    } else if (0 == strcmp(opt, "/Z")) {
        for (i = 0; i < 128; i++) {
            p = dat;
            ct = 0;

```

```

for (j = 0; j < 1024; j++) {
    pind = prof + (long)i + 128L*(long)j;
    idat = (int)(*pind);
    if (idat != 0) {
        *p++ = (double)idat*(prm->vel_conv);
        ct++;
    }
}
if (0 == ct) {
    p_ave[i] = 0.0;
    p_sd[i] = 0.0;
} else {
    f_state(dat, &ave, &sd, ct);
    p_ave[i] = ave;
    p_sd[i] = sd;
}
free((char *)dat);
} else if (0 == strcmp(opt, "/O")) {
    if (NULL == (dat2 = (double *)malloc(1024*sizeof(double)))) {
        fputs("Memory Allocation Failed.", stderr);
        exit(1);
    }

    for (i = 0; i < 128; i++) {
        p = dat; ct = 0;
        p2 = dat2; ct2 = 0;
        for (j = 0; j < 1024; j++) {
            pind = prof + (long)i + 128L*(long)j;
            idat = (int)(*pind);
            if (idat > 0) {
                *p++ = (double)idat*(prm->vel_conv);
                ct++;
            } else if (idat < 0) {
                *p2++ = (double)idat*(prm->vel_conv);
                ct2++;
            }
        }
        if (0 == ct) {
            p_ave[i] = 0.0;
            p_sd[i] = 0.0;
        } else {
            f_state(dat, &ave, &sd, ct);
            p_ave[i] = ave;
            p_sd[i] = sd;
        }
        if (0 == ct2) {
            m_ave[i] = 0.0;
            m_sd[i] = 0.0;
        } else {
    }
}

```

```

    f_state(dat, &ave, &sd, ct2);
    m_ave[i] = ave;
    m_sd[i] = sd;
}
}

free((char *)dat); free((char *)dat2);
}

/* Output */
if (0 == strcmp(opt, "/A") || 0 == strcmp(opt, "/Z")) {
    sprintf(stdout, " Ch\n");
    fprintf(stdout, " Position\n");
    fprintf(stdout, " Average\n");
    fprintf(stdout, " S.D\n");
    for (i = 0; i < 128; i++) {
        position = (prm->start_ch) + (prm->ch_distance)*(double)i;
        fprintf(stdout, "%4d\n", i);
        fprintf(stdout, "%9.4f\n", position);
        fprintf(stdout, "%9.4f\n", p_ave[i]);
        fprintf(stdout, "%9.4f\n", p_sd[i]);
    }
} else {
    sprintf(stdout, " Ch\n");
    fprintf(stdout, " Position\n");
    fprintf(stdout, " Average(+)\n");
    fprintf(stdout, " S.D(+)\n");
    fprintf(stdout, " Average(-)\n");
    fprintf(stdout, " S.D(-)\n");
    for (i = 0; i < 128; i++) {
        position = (prm->start_ch) + (prm->ch_distance)*(double)i;
        fprintf(stdout, "%4d\n", i);
        fprintf(stdout, "%11.4f\n", position);
        fprintf(stdout, "%11.4f\n", p_ave[i]);
        fprintf(stdout, "%11.4f\n", p_sd[i]);
        fprintf(stdout, "%11.4f\n", m_ave[i]);
        fprintf(stdout, "%11.4f\n", m_sd[i]);
    }
}
return 0;
}

```

```

*****
* 超音波流速分布測定装置データ処理ツール *
*   3. 周波数解析ルーチン
*   f_fft()
* zero_y(), ipow2(), inv2pow(), fft()
*     hann(), hamm();
*     FFT.C
*****
#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include "fft.h"

struct parameter {
    char sprof[40];      /* Data File Name */
    double ch_distance;  /* Channel Distance */
    double start_ch;     /* Position of 1st Channel */
    double end_ch;       /* Window End Position */
    int sprof;           /* Fprf */
    int s_velocity;      /* Sound Velocity */
    int s_interval;      /* Storing Interval */
    char t_mode[4];      /* Trigger Mode */
    char ar_mode[7];     /* Auto Record Mode */
    int e_volt;          /* US Emission Voltage */
    int rf_gain;          /* RF Gain Start Value */
    double d_coef;        /* Doppler Coefficient */
    double s_coef;        /* Speed Coefficient */
    double vel_conv;      /* Velocity Conv. */
};

extern int huge *prof;
extern struct parameter *prm;

int f_fft(struct parameter *prm, char *opt, int ch)
{
    int i, j;
    int huge *pind;
    int idat;
    double *x, *y;      /* x : Real Part, y : Imaginary Part */
    double *px, *py;
    double dat;
    double time;
    double b_frequency; /* Basic Frequency */
    double w_number;    /* Wave Number */
    double wkx, wky;

    if (0 == strcmp(opt, "/T") || 0 == strcmp(opt, "/TM") || 0 == strcmp(opt, "/TN")) {

```

```

if (ch < 0 || ch > 128) {
    fputs("位置の指定が誤っています。", stderr);
    exit(1);
}

if (NULL == (x = (double *)malloc(1024*sizeof(double)))) {
    fputs("Memory Allocation Failed.", stderr);
    exit(1);
}

if (NULL == (y = (double *)malloc(1024*sizeof(double)))) {
    fputs("Memory Allocation Failed.", stderr);
    exit(1);
}

if (128 != ch) {
    px = x;
    for (i = 0; i < 1024; i++) {
        pind = prof + 128L*(long)i + (long)ch;
        idat = (int)(*pind);
        *px++ = (double)idat*prm->vel_conv;
    }
}

if (0 == strcmp(opt, "/TM"))
    hamm(x, 1024);
else if (0 == strcmp(opt, "/TN"))
    hann(x, 1024);

zero_y(y, 1024);
fft(x, y, 10, -1.0);

if (7812 == prm->fprf)    time = 37.2E-03;
else if (3906 == prm->fprf) time = 37.2E-03;
else if (1953 == prm->fprf) time = 69.8E-03;
else if (976 == prm->fprf) time = 135.3E-03;
else {
    fputs("Value of Fprf is incorrect.", stderr);
    exit(1);
}

px = x; py = y;
b_frequency = 1.0/(1024.0*time*(prm->s_interval + 1));
fprintf(stdout, " No.%tFrequency%t Power%n");
for (i = 0; i < 512; i++) {
    fprintf(stdout, "%4d%t%9.4f%t", i, b_frequency*(double)i);
    wkx = *px; wky = *py;
    dat = sqrt(wkx*wkx + wky*wky);
    fprintf(stdout, "%9.4f%t", dat);
    px++; py++;
}
} else {

```

```

for (j = 0; j < 128; j++) {
    px = x;
    for (i = 0; i < 1024; i++) {
        pind = prof + 128L*(long)i + (long)j;
        idat = (int)(*pind);
        *px++ = (double)idat*prm->vel_conv;
    }

    if (0 == strcmp(opt, "/TM"))
        hamm(x, 1024);
    else if (0 == strcmp(opt, "/TN"))
        hann(x, 1024);

    zero_y(y, 1024);
    fft(x, y, 10, -1.0);

    px = x; py = y;
    for (i = 0; i < 512; i++) {
        wkx = *px; wky = *py;
        dat = sqrt(wkx*wkx + wky*wky);
        fprintf(stdout, "%9.4f\n", dat);
        px++; py++;
    }
    fprintf(stdout, "\n");
}
}

free((char *)x); free((char *)y);
} else /* opt == "/S" || opt == "/SM" || opt == "SN" */ {
if (ch < 0 || ch > 1024) {
    fputs("時間の指定が誤っています。", stderr);
    exit(1);
}

if (NULL == (x = (double *)malloc(128*sizeof(double))))) {
    fputs("Memory Allocation Failed.", stderr);
    exit(1);
}
if (NULL == (y = (double *)malloc(128*sizeof(double))))) {
    fputs("Memory Allocation Failed.", stderr);
    exit(1);
}

if (1024 != ch) {
    px = x;
    for (i = 0; i < 128; i++) {
        pind = prof + (long)i + 128L*(long)ch;
        idat = (int)(*pind);
        *px++ = (double)idat*prm->vel_conv;
    }
}

```

```

if (0 == strcmp(opt, "/SM"))
    hamm(x, 128);
else if (0 == strcmp(opt, "/SN"))
    hann(x, 128);

zero_y(y, 128);
fft(x, y, 7, -1.0);

px = x; py = y;
w_number = 1.0/(128.0*prm->ch_distance*1.0E-03);
sprintf(stdout, " No.\tWave Number\t Energy\n");
for (i = 0; i < 64; i++) {
    sprintf(stdout, "%4d\t%11.4f\t", i, w_number*(double)i);
    wkx = *px; wky = *py;
    dat = sqrt(wkx*wkx + wky*wky);
    fprintf(stdout, "%9.4f\n", dat);
    px++; py++;
}
} else {
    for (j = 0; j < 1024; j++) {
        px = x;
        for (i = 0; i < 128; i++) {
            pind = prof + (long)i + 128L*(long)ch;
            idat = (int)(*pind);
            *px++ = (double)idat*prm->vel_conv;
        }

        if (0 == strcmp(opt, "/SM"))
            hamm(x, 128);
        else if (0 == strcmp(opt, "/SN"))
            hann(x, 128);

        zero_y(y, 128);
        fft(x, y, 7, -1.0);

        px = x; py = y;
        for (i = 0; i < 64; i++) {
            wkx = *px; wky = *py;
            dat = sqrt(wkx*wkx + wky*wky);
            fprintf(stdout, "%9.4f\t", dat);
            px++; py++;
        }
        fprintf(stdout, "\n");
    }
    free((char *)x); free((char *)y);
}
return 0;
}

```

```

/*
 * 超音波流速分布測定装置データ処理ツール
 * 周波数解析ルーチン用ヘッダファイル
 * zero_y(), ipow2(), inv2pow(), fft()
 * FFT.H
 */
#include <math.h>

#define PI 3.1415926536

int zero_y(double *y, int n)
{
    while (n != 0) {
        *y++ = 0.0;
        n--;
    }
    return 0;
}

int ipow2(int l)
{
    int n = 1;
    while (l != 0) {
        n *= 2;
        l--;
    }
    return n;
}

int inv2pow(int n)
{
    int l = -1;
    while (n != 0) {
        n /= 2;
        l--;
    }
    return l;
}

int hann(double *x, int n)
{
    int i;
    double sc;

    sc = 2.0*PI/(double)n;
    for (i = 0; i < n; i++)
        x[i] = x[i]*(0.50 - 0.50*cos(sc*(double)i));
}

```

```

    return 0;
}

int hamm(double *x, int n)
{
    int i;
    double sc;

    sc = 2.0*PI/(double)n;
    for (i = 0; i < n; i++)
        x[i] = x[i]*(0.54 - 0.46*cos(sc*(double)i));

    return 0;
}

int fft(double *x, double *y, int l, double f)
{
    int i, ix, j, j1, j2, l0, li, lm, mx, n, k;
    int ipow2();
    double s, c, sc, t, tx, ty, arg;

    n = ipow2(l);
    mx = n;
    sc = 2*PI/(double)n;

    for (l0 = 0; l0 < l; l0++) {
        ix = mx - 1;
        mx /= 2;
        arg = 0.0;
        for (lm = 0; lm < mx; lm++) {
            c = cos(arg);
            s = sin(f*arg);
            arg += sc;
            for (li = ix; li < n; li += (ix + 1)) {
                j1 = li - ix + lm;
                j2 = j1 + mx;
                tx = x[j1] - x[j2];    ty = y[j1] - y[j2];
                x[j1] = x[j1] + x[j2]; y[j1] = y[j1] + y[j2];
                x[j2] = c*tx + s*ty;   y[j2] = c*ty - s*tx;
            }
        }
        sc *= 2.0;
    }

    if (f < 0.0) {
        for (i = 0; i < n; i++) {
            x[i] /= (double)n;
            y[i] /= (double)n;
        }
    }
}

```

```
}

j = 0;
for (i = 0; i < n - 1; i++) {
    if (i <= j) {
        t = x[i]; x[i] = x[j]; x[j] = t;
        t = y[i]; y[i] = y[j]; y[j] = t;
    }
    k = n/2;
    while (k <= j) {
        j = j - k;
        k /= 2;
    }
    j = j + k;
}
return 0;
}
```