JAERI-Research
96-037

水平ダクト内気液並向流における界面波のウェーブレット解析

1996年7月

近藤昌也・久木田豊

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所研究情報部研究情報課（〒319-11 茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken, 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　いばらき印刷㈱

JAERI-Research 96-037

水平ダクト内気液並向流における界面波のウェーブレット解析

日本原子力研究所東海研究所原子炉安全工学部
近藤　昌也・久木田　豊

(1996年6月6日受理)

水平配管における気液二相流の流動様式遷移のうち、工学上特に重要とされる波状流からスラグ流への流動様式遷移に大きな影響を与える界面波の成長過程についてウェーブレット解析を行った。特に実験で観察された、界面波が突発的に急減衰及び急成長する現象に注目した。解析の結果、Morletのウェーブレットと Gabor関数が界面波の解析に十分な時間分解能及び周波数分解能を有していること、成長過程においても界面波を構成する周波数成分が各々固有の伝播速度で伝播していること、その伝播速度は深水波の位相速度の理論値とほぼ一致すること、界面波の急減衰及び急成長はこれら周波数成分の位相のずれや一致によって生じることを明らかにした。

東海研究所：〒319－11　茨城県那珂郡東海村白方白根2－4

JAERI-Research 96-037

Wavelet Analysis of Interfacial Waves in Cocurrent Two-phase Flow in Horizontal Duct

Masaya KONDO and Yutaka KUKITA

Department of Reactor Safety Research
Tokai Research Establishment
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received June 6, 1996)

Wavelet analysis was applied to spatially-growing interfacial waves in a cocurrent gas/liquid two-phase flow. The wave growth plays a key role in the transition from stratified-wavy to slug flow, which is an important phenomena in many engineering applications. Of particular interest to the present study was the quick growth or decay of particular waves which were observed in experiments together with the general growth of waves with distance in the flow direction. Among the several wavelet functions tested in the present study, the Morlet wavelet and the Gabor function were found to have spectral and spatial resolutions suitable to the analysis of interfacial wave data taken by the authors. The analysis revealed that 1) the spectral components composing the interfacial waves are propagating at different phase velocities which agree to the theoretical velocities of deep-water waves, 2) the group velocity of the waves also agrees to the deep-water theory, and 3) the quick growth and decay of particular waves occur as a result of the superposition of spectral components with different phase velocities.

Keywords: Wavelet Analysis, Interfacial Wave, Morlet Wavelet, Gabor Function, Deep-Water Wave

目　　次

Contents

1. 緒　　言

水平配管内の気液二相流は、気液の流量、管径、流体の物性値等によりFig. 1.1に示すような様々な流動様式を示す[1]。これらの流動様式は流量等の条件が変わると変化（流動様式遷移）することがある。なかでも気液が完全に分離した流れ（層状流）から、間欠的に液スラグにより流路全体が閉塞する流れ（スラグ流）への遷移は工学上特に重要とされている。

スラグ流への遷移条件付近の層状流は、気液と壁面との干渉や気液の速度差などの要因により界面波が顕著な流れ（波状流）になることが多い。この場合、波状流の界面波が成長することによる遷移が主となる。しかしながら、界面波の成長は一般に流路入口（発達開始点）からの距離や流路出入口条件などによって変化することに加え、波長の大きな波の頂部において気相の流路が狭められることによる界面波の不安定の影響も受ける。そのため、成長の過程は単調でない場合が多く、界面波の成長過程を把握することがスラグ流への遷移メカニズムの解明のために必要とされる。

本研究では界面波の成長過程を、高速度ビデオカメラ及び自走式ビデオカメラを用いて観察すると共に、流路に沿って設置した水位計の測定データを周波数解析することによる評価を試みた。周波数解析の手法としては、従来から広く用いられているフーリエ解析に加え、波の伝播及び成長をより微視的にとえる手段としてウェーブレット解析を用いた。ウェーブレット解析は時間的に急速に変化する信号を周波数解析する手段として様々な分野で注目されているものである。他方、ウェーブレット解析の解析結果の物理的意味については依然議論のあるところであるため、解析結果の評価の参考とするべく、ウェーブレット解析の特徴、性質について調査した結果についても報告する。

2. 実験装置

大型水平ダクト試験装置は、Fig. 2.1 に示すように矩形ダクト試験部、その両端に接続する入口及び出口容器、試験部に水／空気を供給する供給系から構成される。本試験装置は大気圧条件下の様々な流速の水／空気水平並行流の測定及び観察を目的としており、試験部内に水／空気いずれかの単相流か、成層流、波状流及びスラグ流のいずれかの流動様式の二相流を流すことが可能である。以下に本試験装置の構成と測定装置についての概略[2]を記す。

2.1 構成

試験部は流路高 700 mm、流路幅 100 mm、流路長 28.3 m の矩形ダクトであり、内部の流動様式を視認できるよう、その一部分が透明アクリルで作られている。入口容器は試験部入口における流動様式を流速の安定した成層流とするために設置されており、供給系から供給される水／空気の流量の変動の緩和及び整流を行う。出口容器は試験部終端より自由落下する水を受けとめるための容器である。すなわち、出口容器に水を自由落下させることにより、試験部内で生じた界面波の試験部終端での反射を防止する。ゆえに、試験部内の水位は下流に近付くにつれて緩やかに低下するが、本研究において評価の対象とした上流～中流域での水位低下は無視し得る程度に微小である。水供給系は 2 基の循環ポンプ（0 ～ 50 m^3/h、0 ～ 500 m^3/h；実験中はどちらか 1 基を使用）及びオリフィス流量計（測定可能最大流量 25, 50, 150, 500 m^3/h）より構成され、出口容器下部から吸水した水を入口容器下部に供給する。すなわち、本試験装置は特に外部より給水しない限り、実験中は同じ水を使い続ける構造である。一方、空気供給系はコンプレッサー（0 ～ 600 Nm^3/h）、ブロワー（0 ～ 1000 Nm^3/h）及びオリフィス流量計（測定可能最大流量 250, 600 m^3/h（コンプレッサーライン）、400, 1000 Nm^3/h（ブロワーライン））から構成され、外気を任意の流量で入口容器上部に供給する。なお、空気供給系から入口容器内に供給された空気は、試験部を通り出口容器上面より大気中に開放される。

2.2 測定装置

大型水平ダクト試験装置には測定装置として電気抵抗式水位計、オリフィス流量計、圧力計等が設置されている。

電気抵抗式水位計は、60 mm の間隔で鉛直に張った 2 本の白金被覆タングステン線（直径 0.090 mm）間の電気伝導度を動歪み計を用いて測定し、その歪みの大きさから水位を求める水位計である。この水位計の特徴としては、水質（水温）の変化に敏感であるために較正が欠かせないという欠点があるものの、時間分解能が高い（～ 数ミリ秒）、水位に対する直線性が優秀であるという利点を有する。なお、水位計は試験部に沿ってその 11 箇所（本報の評価の対象とした実験において；実験毎に増減）に設置されており、多チャンネルの記録装置と組み合わせることにより、各位置における水位の経時変化を同時に測定できる。

オリフィス流量計は、水供給系の大流量循環ラインと小流量循環ライン、空気供給系のコンプレッサーラインとブロワーラインにそれぞれ 1 基ずつ設置されている。このオリフィスを交換する

ことにより、供給流量を低流量から高流量まで精度良く測定することが可能である。付録Aにオリフィス流量計から流量を算出するために必要な各種定数及び流量算出式を示す。

目視では把握が困難な現象の観察にはビデオカメラを用いた。高速度ビデオカメラ（NAC200, NAC400）は界面波の瞬間的な形状の観察、界面波形と水位計による水位データとの対比とに用いた。また、自走式ビデオ画像撮影システム（ビデオカメラを搭載した台車を一定速度にて走行させ、相対座標系にて撮影を行うシステム）を界面波の伝播速度にあわせて走行させることにより、界面波の成長に伴う形状の変化についての観察を行った。

3. 観察結果

大型水平ダクト試験装置を用いて水／空気水平並行流の実験を行い、界面波の成長過程について観察した。実験は成長する界面波が見られる流量条件（すなわち、流動様式が波状流、もしくは波状流から間欠的に液スラグが発生する状態；水流速 1.0 ～ 1.1 m/s、空気流速 8 ～ 10 m/s、平均水位 500 ～ 550 mm）で行い、試験部入口における流動様式は成層流、試験部出口では水は自由落下、空気は大気解放するようにした。

以下に、試験部の透明アクリル部分において、目視、高速度ビデオカメラ及び自走式ビデオ画像撮影システムによって行った実験時の界面波及び流動様式の観察結果について記す。

3.1 界面波の発生

本試験装置試験部において観察される界面波の多くは、試験部入口から 10 ～ 50 cm 下流側に生じた水位の隆起より発生した。その概念図を自走式ビデオ画像撮影システムにて撮影した画像と共に Fig. 3.1 に示す。この水位の隆起は定常的に存在しており、その隆起の程度は流量条件によって変化するが最大で約 3 cm であった。実験では、隆起は垂直方向に間断なく小刻みに振動しており、その下流側には小さな波高（最大 1 cm 程度）の界面波が連続的に生じることが多かった。

このような水位の隆起は、試験部入口において一時的に加速された水が減速したことによって生じたと考えられる。すなわち、入口容器の流路断面が試験部流路断面より大幅に大きいため（およそ 50 倍）、入口容器内の水位は試験部内の水位より若干高くなる傾向にある。そのため、試験部流入時に水は一時的に加速し、流入直後の水位は低下する。しかし、水は壁面との摩擦などの影響により急激に減速するため、水位は上昇し、「盛り上がった」ような形状となる。また、このような水位の隆起は気相部流路を局所的に狭めることから、他の部分の界面と比べて気液の相互作用を強く受け、これが垂直方向の振動を引き起こすものと思われる。

3.2 界面波の成長

本試験装置試験部において観察された界面波を Figs. 3.2 ～ 3.4 に上流側から順に示す。このように界面波の大きさは試験部上流域と下流域とでは大きく異なる。そこで、界面波の成長過程の観察結果を試験部上流域における場合と、試験部下流域における場合の 2 通りの場合に分けて以下に示す。

試験部上流域における界面波の成長、すなわち、発生直後の界面波の成長は波数の急激な減少によって特徴付けられる。この場合、波数の減少は界面波の減衰及び消滅、あるいは他の界面波との合体によって生じる。この様子を自走式ビデオ画像撮影システムによって観察した結果を Fig 3.5 に示す。図は自走式ビデオ画像撮影システムにより 1/30 秒間隔で撮影された画像を縮小後、2 コマおきに時刻順に並べて表示したものである。図中の矢印は、撮影した画像を 1 コマずつ調べることにより関連づけた、画像間の界面波の対応を示している。0.3 秒間という僅かな時間であるが、界面波

の消失及び合体が見られる。

一方、試験部下流域における界面波の成長、すなわち、ある程度の大きさに成長した界面波については、波数は伝播に伴い緩やかに減少する。しかしながら、個々の界面波については活発に消滅及び発生を繰り返している。例えば、Fig. 3.6 において矢印で対応付けされている左側（下流側）の界面波は、最初は明瞭な界面波の形状を有しているが、0.6 秒間の間に界面波と識別しがたい程度まで減衰する。しかし、同じく図にあるように、減衰するの界面波の直後（上流側）に位置する界面波が急成長するため、巨視的には波数はほとんど変化しない。特筆すべきは、急減衰する界面波の直後に明瞭な形状を持った界面波が存在しなかったとしても、新しい界面波が出現し急成長することである。それゆえ、界面波が短い波群を構成している場合、波群の先頭に位置する界面波は伝播と共に順次減衰／消滅してゆくが、波群の後方に新しい界面波が出現するため、波群の規模が大きく変わることはない。

3.3 界面波の形状

波高、波長の相違を含め、界面波の形状については界面波の規模に関わらず、かなりの個体差が見られた。しかしながら、全体的な傾向としては波の谷に比べて峰の方が尖った形状を有しており、さらに、気相の影響により、波の峰が波の中央よりやや下流側にある場合が多い。また、波頭付近は気相の影響を特に強く受けるため、界面波の波長と比べて非常に短い波長（すなわち、大きな波数）の小波が重畳して見られた。

これらの観察は気液の流量を一定とした条件下で行ったが、流量条件によっては、界面波が連続的に見られる状態と界面波が全く見られない状態とが交互に現れることがあった。こうした現象は波状流と成層流の境界流量付近の流量条件の実験で多く見られ、気液の流量を増やすと連続した界面波の状態が長く続く傾向に、流量を減らすと界面波が見られない状態が長く続く傾向となった。なお、気液流量が一定の場合でも、界面波が連続してみられる状態、並びに界面波が全く見られない状態の区間の長さは共に一定していなかった。

4. ウェーブレット変換

4.1 基本原理

ウェーブレット変換は、任意の信号をウェーブレットと呼ばれる関数（以下、ウェーブレット関数）の加重和に分解する変換手法である。これは、フーリエ変換が信号を三角関数（調和振動 $\exp(i\omega t)$）の加重和に分解することに相当する[3-5]。しかしながら、三角関数が完全な周期関数であることに対して、ウェーブレット関数はある時刻の周辺を除き、ゼロに十分に近い値をとる（局在化（局所化）している）という点で大きく異なる。そのため、ウェーブレット変換により変換される信号は、特定の時刻の周辺の信号のみとなる。これはフーリエ変換が原理的には全時間域の信号を変換することと対照をなす。

こうしたことからウェーブレット変換の変換結果（以下、ウェーブレット変換係数）はフーリエ変換では失われる「時刻情報」を有する。すなわち、フーリエ変換による変換結果（フーリエ係数）は周波数という１つの引数を持つ関数として与えられるが、ウェーブレット変換係数は周期に相当するパラメータ（スケールパラメータ（以下、スケール））に加え、時刻を示すパラメータ（シフトパラメータ（以下、シフト量））という２つの引数を有する関数である[6]。

スケールはウェーブレット変換の定義式（次節の (4.2.5) 式）によって示されるように積分核であるウェーブレット関数の時間軸上の大きさを示す。ゆえに、スケールの値が小さい場合は高周波の現象に対して、大きい場合は低周波の現象に対して大きな感度を持つ変換結果が得られる。一方、シフト量はウェーブレット関数の時間軸上の位置、すなわち変換の対象となる信号の時刻を示す。なお、これら２つのパラメータの物理的性質はウェーブレット変換の定義式によって決定され、使用するウェーブレット関数の種類によらない。

このようにウェーブレット変換係数は時間情報と周波数情報の両方を同時に有するが、その分解能は時間と周波数との間に存在する不確定性関係によって制限される[3,5,6]。すなわち、時間分解能 Δt と周波数分解能 $\Delta\omega$ の積をある一定の値より小さくすることは、ウェーブレット変換のみならず、原理的に不可能である。実際、時間領域と周波数領域の双方においてデルタ関数となるような関数は存在しない（周波数領域においてデルタ関数となる三角関数は時間領域では無限に広がった形状となる）。

ウェーブレット変換の時間分解能及び周波数分解能は、時間と周波数との間に存在する不確定性関係による制約に加え、積分核であるウェーブレット関数の種類によっても大きく異なる。すなわち、分解能はウェーブレット関数の時間領域及び周波数領域における形状によって決定される。ゆえに、時間領域もしくは周波数領域において良好な分解能を得るためには、それぞれの領域において局在している関数をウェーブレット関数として用いる必要がある。

さらに、ウェーブレット変換においては、スケールの値が変化するとウェーブレット関数の時間領域及び周波数領域における形状が変化するため、スケールの値によって時間分解能及び周波数分解能が変化するという特徴がある[6,7]。Fig. 4.1 に時間－周波数平面におけるウェーブレット変換の分解能の概念図を、Fig. 4.2 にスケールを変化させた同一のウェーブレット関数を示す。スケールの

値が小さい場合はウェーブレット関数の時間軸上に占める大きさが小さくなるため時間分解能は高くなる。しかし、同時にそのウェーブレット関数の周波数領域における形状が広がったものとなるため、周波数分解能は低下する。同様に、スケールの値が大きい場合はこの逆となるため、時間分解能は相対的に低下し、周波数分解能が相対的に向上する。

4.2 数学的条件及び定義

以下に、実数全体を対象とする関数空間 $\mathbf{R} := (-\infty, \infty)$ の場合について、ウェーブレット関数の満たすべき条件とウェーブレット変換の定義式について示す[3,4,6]。

ウェーブレット関数 $\psi(t)$ は関数空間 $L^2(\mathbf{R})$ に属する必要がある。すなわち、

$$L^2(\mathbf{R}) = \int_{-\infty}^{+\infty} |\psi(t)|^2 dt < \infty \tag{4.2.1}$$

を満たす必要がある。フーリエ変換における積分核である三角関数は $t \to \pm\infty$ において零に減衰しないため、この条件を満たさない。

この条件に加え、ウェーブレット関数は次式で定義された許容条件 (admissibility condition) を満たす必要がある。

$$C_\psi := \int_{-\infty}^{+\infty} \left| \frac{\hat{\psi}(\omega)}{\omega} \right|^2 d\omega < \infty \tag{4.2.2}$$

ここで $\hat{\psi}(\omega)$ は $\psi(t)$ のフーリエ変換である。さて、(4.2.2) 式を満たすには、$\hat{\psi}(0) = 0$ である必要がある。これはすなわち、$\hat{\psi}(\omega)$ の定義

$$\hat{\psi}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \psi(t) \exp(-i\omega t) dt \tag{4.2.3}$$

より、$\psi(t)$ が

$$\int_{-\infty}^{+\infty} \psi(t) dt = 0 \tag{4.2.4}$$

を満たす必要があることを意味する。

(4.2.1) 式と (4.2.2) 式を満足するウェーブレット関数 $\psi(t)$ について、そのウェーブレット変換は関数空間 $L^2(\mathbf{R})$ 上において次のように定義される。

$$T_\psi(a,b)=\frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty}\overline{\psi\left(\frac{t-b}{a}\right)}f(t)dt \tag{4.2.5}$$

ここで、$T_\psi(a,b)$ はウェーブレット変換係数、a はスケール、b はシフト量、$f(t)$ は関数空間 $L^2(\mathbf{R})$ に属する被変換関数である。また、$\overline{\psi}$ は ψ の複素共役を示す。この定義式が示すように、ウェーブレット変換とフーリエ変換とは積分核が異なる他はほぼ同一の変換である。

4.3 ウェーブレット関数

ウェーブレット変換の積分核であるウェーブレット関数は、前節で示した数学的条件を満たす関数ならば自由に選択することが可能である。これはすなわち、変換結果であるウェーブレット変換係数の物理的意味もウェーブレット関数毎に異なることを意味する[8]。一例を挙げると、連続的に変化する信号を対象とする場合、ウェーブレット関数に偶関数を用いるならば信号の山や谷に、奇関数を用いるならば信号のエッジ（立ち上がりや下がり）に対して大きな感度を持つ結果が得られる。

実際には、時間分解能／周波数分解能、変換結果の物理的意味の解釈等の問題から、特定のウェーブレット関数が広く用いられている。以下に、そうしたウェーブレット関数についてその概略をまとめた。

4.3.1 Gauss 分布関数の高階導関数

Gauss 分布関数の高階導関数は、広く用いられている実数型ウェーブレット関数であり、以下の式によって表される[9]。

$$\psi_m(t)=(-1)^m\frac{d^m}{dt^m}\exp\left(-\frac{|t|^2}{2}\right) \tag{4.3.1}$$

$$\hat{\psi}_m(\omega)=m(i\omega)^m\exp\left(-\frac{|\omega|^2}{2}\right) \tag{4.3.2}$$

奇数階の導関数が奇関数、偶数階の導関数が偶関数となるが、いずれも時間領域においては $t=0$ の周りにおいてのみゼロでない値をとる。また、周波数領域においては 1 つの卓越周波数を有し、その周りにおいてのみゼロでない値をとる。

Gauss 分布関数のラプラシアンである 2 階導関数（$m=2$）は、これら Gauss 分布関数の高階導関数のなかでもとりわけ広く用いられており、一般に、Marr のウェーブレットまたは Mexican Hat 型ウェーブレットと呼ばれている。Fig. 4.2 に Mexican Hat 型ウェーブレットの時間領域及び周波数領域における形状を示す。

なお、Gauss 分布関数の高階導関数に類するものとして、異なる微分階数のGauss 分布関数の差 (D.O.G.：Differential of Gaussians) をウェーブレット関数として用いることもある。

4.3.2 Morlet のウェーブレット

Morlet のウェーブレットは代表的な複素数型ウェーブレット関数のひとつである。Fig. 4.3 に時間領域及び周波数領域におけるその形状を示す。このウェーブレット関数はガウス分布の包絡波を持つ波数ベクトル k_ψ の変調した正弦波であり、次式にて表される[9]。

$$\psi_{k_\psi}(t) = \exp\left(ik_\psi t\right)\exp\left(-\frac{|t|^2}{2}\right) \tag{4.3.3}$$

$$\hat{\psi}_{k_\psi}(\omega) = \frac{1}{\sqrt{2\pi}}\exp\left[-\frac{\left(k_\psi - \omega\right)^2}{2}\right] \qquad (\omega > 0) \tag{4.3.4}$$

$$\hat{\psi}_{k_\psi}(\omega) = 0 \qquad (\omega \le 0) \tag{4.3.5}$$

Morlet のウェーブレットは実数部が偶関数、虚数部が奇関数となっており、実数部は信号の山や谷に、虚数部は信号のエッジ（立ち上がりや下がり）に対して大きな感度を示す。ただし、このウェーブレット関数の時間積分は厳密にはゼロにならないため、必要に応じて補正する必要がある。なお、$k_\psi = 6$ とすると必要な補正量は計算誤差の範疇に収まるため特に補正の必要はない[9]。

4.3.3 Gabor 関数

Gabor 関数は時間と周波数に関する不確定性が最小の関数であり、時間領域及び周波数領域における局在性が良い[6,10,11]。Gabor 関数の時間領域及び周波数領域における形状を Fig. 4.4 に、定義式を以下に示す。

$$\psi(t) = \frac{1}{\pi^{1/4}}\sqrt{\frac{\omega_p}{\gamma}}\exp\left[-\frac{1}{2}\left(\frac{\omega_p t}{\gamma}\right)^2 + i\omega_p t\right] \tag{4.3.6}$$

$$\hat{\psi}(\omega) = \frac{\pi^{1/4}}{\sqrt{2\pi}}\sqrt{\frac{2\gamma}{\omega_p}}\exp\left[-\frac{\gamma^2}{2{\omega_p}^2}\left(\omega_p - \omega\right)^2\right] \tag{4.3.7}$$

ここで ω_p, γ は定数である。これらの式が示すように、Gabor 関数も Morlet 関数と同様、時間領域においてはガウス分布の包絡波を持つ正弦波である。また、周波数領域においてはピーク周波数を ω_p とするガウス分布をなす。ただし、Gabor 関数は $\hat{\psi}(0) \neq 0$ であるため、(4.2.2) 式の許容条件を満たさない。しかし、

$$\gamma = \pi\sqrt{\frac{2}{\ln 2}} \tag{4.3.8}$$

とすると、ピーク値 $\hat{\psi}(\omega_p)$ に比べて $\hat{\psi}(0)$ の値が十分小さくなるため、近似的に許容条件を満たす。

4.3.4 基底が完全正規直交系をなすウェーブレット

ウェーブレット変換はフーリエ変換と異なり、ごく限られたウェーブレット関数のみ完全正規直交系をなす基底を有する。それゆえ、それらごく限られたウェーブレット関数による変換結果を除き、一般に、ウェーブレット変換係数の二乗はフーリエ解析におけるパワースペクトル密度関数のようにエネルギーと見なすことは出来ない。これはウェーブレット変換が一般に直交基底による展開ではなくフレームによる展開であるからである。すなわち、ウェーブレット変換の基底 $\{\mathbf{e}_j\}$ は任意のベクトル $\mathbf{x}$ に対して次式を満たすものの、必ずしも直交しているわけではない[5,6]。

$$\|\mathbf{x}\|^2 = \sum_j \left|\langle \mathbf{x}, \mathbf{e}_j \rangle\right|^2 \tag{4.3.9}$$

さらに、直交化されたウェーブレットではスケールとシフト量は連続ではあり得ない。それゆえ、この場合のウェーブレット変換は離散ウェーブレット変換となり、離散的なスケールとシフト量の組み合わせにおいてのみ成り立つ[5,12]。

完全正規直交基底を有するウェーブレット関数の一つとして、Haar 関数（Haar 基底）と呼ばれる以下のような関数が知られている[4,12,13]。

$$\psi(t) = \begin{cases} 1 & (0 \le t < 1/2) \\ -1 & (1/2 \le t < 1) \\ 0 & (t < 0, t \ge 1) \end{cases} \tag{4.3.10}$$

この式が示すように Haar ウェーブレットは不連続関数であり、不連続関数に限れば、この他にも種々の完全正規直交基底を有するウェーブレット関数が存在する[14]。

一方、連続的なウェーブレット関数としては Meyer のウェーブレットと Daubechies のウェーブレットの２種のみが知られている。それぞれの時間領域における形状を Figs. 4.5[14], 4.6 に示す。これら２種類のウェーブレット関数はいずれも直交しており、遠方で急速に減衰する微分可能な関数である[4]。また、どちらのウェーブレットも一意に決まったものではなく、特定の条件を満たせばよいことからその形状には自由度が残されている。

なお、これらのウェーブレットは難解であり、計算機を用いて求める必要がある。Daubechies のウェーブレット（N = 8）に関しては付録 B に文献 [15, 16] に基づいて作成した算出プログラムを付す。

5. ウェーブレットによる計算

本研究では界面波のウェーブレット解析を行うにあたり、ガウス関数の2階導関数（以下、Mexican Hat 型ウェーブレット）、Morlet のウェーブレット及び Gabor 関数をウェーブレット関数として用いることとした。これらのウェーブレット関数はいずれも広く用いられており、「標準的な」ウェーブレット解析による結果が期待される。また、複数のウェーブレットを用いることにより、ウェーブレット関数の種類によってウェーブレット解析の結果がどのように変わるかを把握することができる。

Figs. 4.2 - 4.4 に示した関数形状から察せられるように、Mexican Hat 型ウェーブレットによる変換結果は他の2つのウェーブレットの変換結果と符号が逆になる。すなわち、凸型の非変換信号に対し、Morlet のウェーブレット及び Gabor 関数は正の変換係数を与えるが、Mexican Hat 型ウェーブレットによる変換は負の変換係数を与える。ゆえに、本報では容易に比較できるように Mexican Hat 型ウェーブレットによる変換結果についてはその符号は反転させることとした。

また、Morlet のウェーブレット及び Gabor 関数は複素関数であるが、実数型ウェーブレットである Maxican Hat 型ウェーブレットとの比較するために、その実数部のみを評価に用いることとした。さらに Morlet のウェーブレットは変数 k_ψ を、Gabor 関数は変数 ω_p を含むが、本研究では補正の必要性、対象とする周波数範囲を考慮して一律に $k_\psi = 6$、$\omega_p = 2\pi$ とした。

5.1 積分範囲

ウェーブレット変換の積分は $[-\infty, +\infty]$ の範囲で定義されているものの、数値計算上はウェーブレット関数（絶対値）が $t = 0$ から離れるにつれて急速にゼロに近づくため $t = 0$ 付近のみを計算すればよい。Mexican Hat 型ウェーブレットを例に取ると、このウェーブレットは時間領域において $t = 0$ のときに最大値 1 をとるが、それを離れるにつれて急減少する。すなわち、

$$\left|\psi\left(\pm\sqrt{3}\right)\right| = 2\exp\left(-\frac{3}{2}\right) \quad \sim 0.45 \quad \text{（極値）} \tag{5.1.1}$$

$$\left|\psi\left(\pm 2\sqrt{3}\right)\right| = 11\exp(-6) \quad \sim 2.73\times10^{-2} \tag{5.1.2}$$

$$\left|\psi\left(\pm 3\sqrt{3}\right)\right| = 26\exp\left(-\frac{27}{2}\right) \quad \sim 3.56\times10^{-5} \tag{5.1.3}$$

$$\left|\psi\left(\pm 4\sqrt{3}\right)\right| = 47\exp(-24) \quad \sim 1.80\times10^{-9} \tag{5.1.4}$$

となる。ここで $t = m\sqrt{3}$ としているのは、時間軸上の極値の位置を基準にとることにより、感覚的にわかりやすくするためである。この結果より、本研究では必要な計算精度などを考慮し、ピーク値 $\psi(0)$ の 10^{-4} 以下のオーダーは無視し得るものとして計算範囲を定めた。すなわち、Mexican Hat 型ウェーブレットの場合は、

$$-3\sqrt{3} \le t \le 3\sqrt{3} \tag{5.1.5}$$

とした。ゆえに、スケールが a、シフト量が b の場合の計算範囲は

$$b - 3\sqrt{3}a \le t \le b + 3\sqrt{3}a \tag{5.1.6}$$

となる。

同様にMorletのウェーブレットの場合は、$t = m\pi/6$（$k_\psi = 6$ の場合。m は整数）において極値をとるが、その極値がピーク値 $\psi(0)$ の 10^{-4} 以下のオーダーとなるように計算範囲を定めた。この条件を満たすのは $m = 8$ 以上の場合であるため、計算範囲を

$$b - \frac{8\pi}{6}a \le t \le b + \frac{8\pi}{6}a \tag{5.1.7}$$

とした。

Gabor関数の場合は $t = m\pi/\omega_p$ において極値をとるものの、極値のピーク値に対する比 $\psi(m\pi/\omega_p)/\psi(0)$ は変数 ω_p に無関係である。ゆえに、計算範囲をMorletのウェーブレットの場合と同様の条件により、

$$b - \frac{7\pi}{\omega_p}a \le t \le b + \frac{7\pi}{\omega_p}a \tag{5.1.8}$$

とした。

これらの積分範囲の妥当性に関しては5.3節にてスケール a の範囲と共に検討する。

5.2 スケールと周波数の関係

本報ではウェーブレット解析結果の比較検討のため、ウェーブレット固有の引数、すなわちスケールとシフト量を周期等の一般的な物理的指標に置き換えることを試みた。シフト量に関しては時刻に相当する物理的意味を持つことが明らかであるが、ウェーブレット関数の大きさを表すスケールに関しては、周期との関係を直裁的に示すことは難しい。そこで、ウェーブレット関数の周波数領域における卓越周波数を指標とし、スケールと周波数との関連付けを試みた。

評価に用いることとしたMexican Hat型ウェーブレット、Morletのウェーブレット及びGabor関数はいずれも周波数領域において、スケールの値によって変化する1つの卓越周波数を有する。信号をウェーブレット変換することは、その信号に含まれる種々の周波数成分のうち、ウェーブレット関数の卓越周波数付近の周波数成分を抽出することに相当する。すなわち、これを数式で示す

と以下のようになる。

任意の周波数成分 $f_\omega(t) = A\sin(\omega t)$ を考える。この周波数成分のウェーブレット変換は、

$$T_\psi(a,b) = \frac{A}{\sqrt{a}}\int_{-\infty}^{+\infty}\overline{\psi}\left(\frac{t-b}{a}\right)\sin(\omega t)dt \tag{5.2.1}$$

となる。ウェーブレット関数を偶関数と限定するならば、

$$T_\psi(a,b) = A\sqrt{a}\sin(\omega b)\int_{-\infty}^{+\infty}\overline{\psi}(t)\cos(\omega a t)dt \tag{5.2.2}$$

となり、これをフーリエ成分 $\hat{\psi}$ を用いて表すと、

$$T_\psi(a,b) = A\sqrt{a}\sin(\omega b)\,\mathrm{Re}\left[\hat{\psi}(\omega a)\right] \tag{5.2.3}$$

となる。さらに、ある時刻、すなわち、シフト量 b がある一定の値であると仮定すると (5.2.3) 式は、

$$T_\psi(a) = A_b'\sqrt{a}\,\mathrm{Re}\left[\hat{\psi}(\omega a)\right] \tag{5.2.4}$$

と簡略化できる。評価に用いることとしたMexican Hat型ウェーブレット、Morletのウェーブレット及びGabor関数は、周波数領域においていずれも尖った形状を有しており、その急峻さは $\sqrt{a}$ のそれよりもずっと大きい。ゆえに、積 ωa がフーリエ成分 $\hat{\psi}$ の卓越周波数となる場合、すなわち、$\hat{\psi}$ が極大となる場合に変換係数 $T_\psi(a)$ がほぼ極大になると見なせる。一方、積 ωa がフーリエ成分 $\hat{\psi}$ の卓越周波数から離れると、変換係数 $T_\psi(a)$ はゼロに近づく。それゆえ、ウェーブレット変換は、ウェーブレット関数という周波数フィルターを用いた、卓越周波数周りの周波数域に属する成分のみの抽出と見なせる。

スケール a、シフト量 b の場合のウェーブレット関数 $\psi\left(\frac{t-b}{a}\right)$ について考える。明らかなことではあるが、各ウェーブレット関数の卓越周波数はシフト量に関わらず一定である。また、いずれのウェーブレット関数も $\omega > 0$ の領域においては卓越周波数においてのみ極値をとる。ゆえに、その卓越周波数は $\frac{d}{d\omega}\hat{\psi}(\omega) = 0$ となる条件から次のようになる。

Mexican Hat型ウェーブレットの場合 $$\omega = \frac{\sqrt{2}}{a} \tag{5.2.5}$$

Morletのウェーブレット場合 $$\omega = \frac{k_\psi}{a} \tag{5.2.6}$$

Gabor関数の場合 $$\omega = \frac{\omega_p}{a} \tag{5.2.7}$$

これはすなわち、ウェーブレット変換において任意のスケール a が抽出する周波数成分は、上の関係式で与えられることを意味している。もっとも、周波数領域における各ウェーブレット関数は卓越周波数においてのみゼロでない値をとるデルタ関数ではないため、周波数分解能には限りがある。しかし、これらの関係式で表される周波数成分が最も良好に抽出されるという観点から、本報ではこのスケールと周波数との関係を用いてスケールを周波数に変換するものとした。

5.3 スケールの範囲

連続ウェーブレット変換を行う場合でも、数値計算を行うに際にはウェーブレット関数及び変換の対象となるデータを離散化する必要がある。しかしながら、時間軸上におけるウェーブレット関数の大きさはスケール a の値と共に変化するため、離散化の程度がスケールによって変化してしまうという問題がある。すなわち、スケールの値が小さい場合はウェーブレット関数が時間軸上で「縮まった」ものとなるため、Fig. 5.1 に示すように相対的にウェーブレット関数を粗く離散化することになる（メッシュ間隔はウェーブレット解析の対象となる実験データのサンプリングレートより決定されるため）。逆にスケールの値が大きい場合は、計算精度には影響しないものの必要以上に細かく計算されるため、計算量が増大するという弊害がある。

そこで、本研究では有効であるスケールの範囲と 5.1 節で扱った積分範囲の妥当性を確認するために、ウェーブレット関数がその関数の種類に関わらず満たすべき条件である、

$$\int_{-\infty}^{+\infty}\psi\left(\frac{t}{a}\right)dt = 0 \tag{5.3.1}$$

という条件（4.2 節参照）を、離散化したウェーブレット関数が満たすことを確認した。端的には、評価の対象とする 0.9 ～ 100 Hz の周波数帯に相当するスケールの範囲において、離散化したウェーブレット関数を 5.1 節で定めた積分範囲において時間積分し、その絶対値が無視できる範囲に収まることを確認した。すなわち、

$$\left|\sum_i \psi_i\left(\frac{t}{a}\right)\right| < 10^{-4} \tag{5.3.2}$$

という関係式が評価の対象とするスケールの範囲で常に成り立つことを確認した（計算はすべて倍精度で実行）。

5.4 基本特性

いくつかの例外を除き、ウェーブレット解析による解析結果の物理的意味は一意に与えられていない。また、フーリエ解析等と比較すると新しい手法であることもあり、その特性は広く知られているとは言い難い。そこで、解析結果の評価の指標とすべく、単純な関数を被変換関数とした場合のウェーブレット変換係数について計算した結果を付録 C にまとめた。

6. 界面波の周波数特性

大型水平ダクト試験装置における波状流の水位データの一例を Fig. 6.1 に示す（流速条件：水 1.0 m/s、空気12.9 m/s）。これらの水位データは 1 m 間隔で設置された 5 台の水位計によって同時に測定された水位の経時変化（収録周波数 1 kHz）であり、図ではそれを上流側から順に時刻を揃えて示している。このため、試験部内を上流から下流へと伝播する界面波は、図の左上（上流側水位データの早い時刻）から右下（下流側水位データの遅い時刻）へとその軌跡を残す。しかしながら、ほとんどの界面波は水位計の設置間隔（1 m）を伝播する間に多少なりとも変形してしまうため、各水位データに現れた界面波を相互に対応付けするにはある程度推測に頼らざるを得ず、水位データから個々の界面波の成長の評価を行うには限界がある。もっとも、全体的、統計的に見れば平均波高と平均周期とが伝播に従い増大する傾向を示しているのは明らかである。また、界面波の伝播速度が一定、もしくは伝播に従って増大していると仮定するならば、平均周期が増大していることから、平均波長も増大していることになる。

界面波形に関しては、L/D 等の条件が同一であったとしても同一の形状を示すことは少なく、波高と周期に関しても個体差が大きい。もっとも、波の山は波の谷よりも急峻になる傾向はほぼ共通に見られる。また、数分以上連続して採取した水位データを調べると、3 章で示した観察結果と同様、界面波のほとんど見られない時間領域と界面波が連続して見られる時間領域とが混在する様子が見られた。

なお、以下では特に断らない限り、Fig. 6.1 その一部を示した水位データ（1.5 ～ 60 秒間）を評価の対象とする。本水位データは気液の流量が一定の条件下で、界面波がほぼ連続的に連なっている波状流を測定したものである。また、スラグは発生していない。

6.1 界面波のパワースペクトル密度

界面波の平均波高及び平均周期の伝播による変化をより明確にするため、L/D = 3.6 ～ 10.8 の範囲に設置した 6 台の水位計のそれぞれについて、60 秒間の水位データのパワースペクトル密度関数を求めて比較した。計算は 60000 点（60 秒間×1000 Hz）の水位データを 1024 点ずつデータウィンド（Bingham ら[17]）を介して分割し、その各々にフーリエ変換を施してパワースペクトル密度関数を求め、その平均をとる（平滑化のため）という手順で行った。得られたパワースペクトル密度関数を Fig. 6.2 に、算出に用いたプログラムを付録 D に記す。

いずれの水位データからも、ひとつの卓越周波数とそこから離れるにつれてべき関数的に減衰するパワースペクトル密度関数が得られた。これは、卓越した周波数成分と、それより弱い多くの異なった周波数成分が重畳して界面波を構成していることを示している。また、その重畳している周波数成分の振幅は、卓越周波数より離れるにつれてべき関数的に小さくなる傾向にあることを示している。

パワースペクトル密度関数の伝播による変化からは、1）卓越周波数が伝播に従って低周波側へとシフトする、2）卓越周波数及びそれより低周波の成分の強度が伝播に従って増大する、3）

卓越周波数より高周波の成分の強度は伝播に関わらずほぼ一定である、4）卓越周波数より高周波成分については周波数に対するパワースペクトル密度関数の減少傾向は伝播に関わらずほぼ一定である、といった傾向が読みとれる。卓越周波数の低周波側へのシフトは界面波の平均周期の増大を、パワースペクトル密度関数の増大は平均波高の増大をそれぞれ周波数領域において定量的に示すものであり、観察結果と一致する。また、高周波成分のパワースペクトル密度関数がほとんど増加しなかったことに関しては、波長に対する波高（振幅）の大きさには限度があるためと思われる。

6.2 界面波のウェーブレット変換係数

L/D = 9.4 における水位データとそのウェーブレット変換係数（ウェーブレット関数：Mexican Hat 型ウェーブレット、Morlet のウェーブレット、Gabor 関数）を Figs. 6.3 ～ 6.5 に示す。これらの図では縦軸、すなわちスケールは 0.9 ～ 100 Hz（28 点）の周波数に相当する範囲を対数で、横軸は時刻及びシフト量を 0 ～ 1.5 秒（1500 点）の範囲で示している。また、ウェーブレット変換係数の大きさは等高線を含む明暗にて示している（等高線の部分を除き、明るくなるほど大きい）。参考のために、Fig. 6.6 に Fig. 6.5 のウェーブレット変換係数を 3 次元プロットの形態で表示した図を示す。

いずれのウェーブレット関数を用いて処理した場合においても、界面波の波頭が現れる時刻（シフト量）に強い正のピークが、また波の谷が現れる時刻に強い負のピークが現れる。これらの正負の強いピークが現れる周波数（スケール）は特定の周波数付近（約 5 ～ 10 Hz）に分布しているが、個々の界面波によって多少の変動が見られる。すなわち、個々の界面波の卓越周波数は完全には一致していないもののほぼ揃っており、フーリエ解析による卓越周波数（6 Hz；Fig. 6.2）はその平均を示していると捉えることができる。

Figs. 6.4, 6.5 からは、卓越周波数から外れた周波数の成分が卓越周波数の位相と独立した位相を有していることを示している。これは様々な周波数の成分が重畳して界面波を構成しており、かつ、それらが卓越周波数の位相と独立して存在していることを意味する。また、周波数が高くなるにつれて位相が徐々に不明瞭となる傾向が見られるが、これはパワースペクトル密度関数（Fig. 6.2）が示しているように成分の強度がべき関数的に減少しているためと思われる。ただ、波高の大きな界面波に関しては（図中において時刻 1.3 ～ 1.4 s）、低周波の成分から高周波の成分まで幅広い周波数域にわたって位相が揃っていることが示された。

3 者のウェーブレット関数による変換結果を比較すると、Morlet のウェーブレット及び Gabor 関数による変換の結果にはほとんど差異は見られないものの、Mexican Hat 型ウェーブレットによる変換の結果は他の 2 つの場合と大きく異なった。これは Mexican Hat 型ウェーブレットの周波数分解能が Morlet のウェーブレット及び Gabor 関数に比べて劣っているためと思われる。実際、付録 C において示した三角関数 $f(t) = \sin(120t)$（すなわち、周波数領域におけるデルタ関数）のウェーブレット変換係数についても、Mexican Hat 型ウェーブレットを使った場合は相対的に周波数方向に広がった結果が得られ、周波数分解能が他の 2 つのウェーブレットに比べて劣っていることが示されている。

同様のことは「ウェーブレットスペクトル[5]」からも検証できる。本報では、ウェーブレットスペクトル $T_\psi^2(a)$ を次のように定義する。

$$T_\psi^2(a) = \frac{1}{N}\sum_b T_\psi^2(a,b) \tag{6.2.1}$$

ここで N は右辺の項の数であり、十分大きいものとする。本報で評価に用いているウェーブレット関数はいずれも直交していないため、(6.2.1) 式で定義したウェーブレットスペクトルを用いてエネルギー等を評価することはできない。しかし、このウェーブレットスペクトルをパワースペクトル密度関数と比較することにより、その「ずれ」の程度を把握する目安とすることができると思われる。

各々のウェーブレット関数による変換係数についてウェーブレットスペクトルを求め、それをパワースペクトル密度関数と比較した結果を Fig. 6.7 に示す。ただし、ウェーブレットスペクトル及びパワースペクトルはピーク値を 1 に規格化してあり、その絶対値は意味を持たない。卓越周波数より高い周波数領域においては、Morlet のウェーブレット及び Gabor 関数によるウェーブレットスペクトルはパワースペクトル密度関数と良好に一致した。しかし、Mexican Hat 型ウェーブレットのウェーブレットスペクトルはパワースペクトル密度関数よりも大幅に大きな値をとった。これは卓越周波数等の相対的に強度の大きい周波数成分の影響が、Mexican Hat 型ウェーブレットの分解能の不足により高周波数領域に及んだ結果と思われる。逆に、Morlet のウェーブレット及び Gabor 関数に関してはそうした強度の大きい周波数成分の影響を受けていないことから、解析に十分な周波数分解能を有していると思われる。

6.3 界面波の周波数毎の伝播速度

水位データをウェーブレット変換した結果、界面波を構成する種々の成分をその位相を保持したまま分離することができた。それゆえ、近接する 2 点において同時刻に水位を測定し、それらをウェーブレット変換した後に、各周波数（スケール）毎に相互相関関数を求めるならば、各周波数成分の伝播に要する時間を明らかにすることが可能であると思われる。すなわち、水位 $x_1(t)$ と $x_2(t)$ のウェーブレット変換係数をそれぞれ $T_1(a,b)$、$T_2(a,b)$ とすると相互相関関数は、

$$C(a,\tau) = \overline{T_1(a,b)T_2(a,b+\tau)} \tag{6.3.1}$$

となり、相互相関関数 $C(a,\tau)$ は、ラグ（隔り時間：2 つの信号を時間軸をずらして比較する場合、そのずらす時間） τ がそれぞれの周波数成分の 2 点間の伝播に要する時間と一致する場合に極大になるものと推測される。

Fig. 6.8 ～ 6.10 は 1 m の間隔をおいて測定した 2 組の水位データ（L/D = 8.0 & 9.4、50 秒間）のウェーブレット変換係数の相互相関関数である。各スケールについてピーク（極値）が複数あるのは界面波が連続的な波であるため、その波自身との相関の他にその前後に位置する波との相関が生じ

るからである。また、波の山と谷の形状が大まかに水平線に対して線対称をなすことから負のピークも生じた。

界面波の伝播による、その波自身との相関を示すピーク（図中にて矢印で示す）は、最も強いピークではないものの、3 ～ 10 Hz という比較的広い周波数域にわたってはっきりと表れており、高周波成分ほど時間がかかる、すなわち位相速度が遅いという傾向を示した。これは成長過程にある界面波という特殊な状況であっても、界面波を構成する各成分は周波数毎に独立した位相を保持しつつ独立した位相速度にて伝播することを示している。これをパワースペクトル密度関数（Fig. 6.2）と比較すると、その周波数成分の強度が卓越周波数のそれの 10 分の 1 以下に至るまで明確な相関があることがわかる。周波数別に見ると、20 Hz 以上の高い周波数成分に関してはほとんど何の相関も得られなかった。また、3 Hz 以下の成分に関しては、界面波の伝播による相関と異なった傾向を有する弱い不明瞭な相関が存在しており、界面波の伝播以外の要因が支配的であることを示している。

Fig. 6.8 ～ 6.10 の相互相関関数によって示された周波数成分毎の位相速度の比較を行うために、その各々から位相速度を求め、周波数に対してプロットした図を Fig. 6.11 に示す。図には比較のためにフーリエ解析によって求めた位相速度 $U(\omega)$

$$U(\omega) = \frac{l\omega}{\tan^{-1}\left(\frac{Q(\omega)}{K(\omega)}\right)} \tag{6.3.2}$$

と、深水波の位相速度の理論値 V_{phase}[18]

$$V_{phase} = \frac{\omega}{k} = \sqrt{\frac{g}{k}} \tag{6.3.3}$$

を重ねて示す。ただし、$K(\omega)$ はコスペクトル（クロススペクトルの実部）、$Q(\omega)$ はクオドスペクトル（クロススペクトルの虚部）、l は 2 点間の距離である。また、図示する際に深水波の位相速度の理論値には水の流速 1.03 m/s（水が流路断面において均一の流速で流れていると仮定して算出）を加算した。フーリエ解析の対象とする水位データに関しては、ウェーブレット解析の場合と同一とすることが望ましかったが、位相角 $\tan^{-1}(Q(\omega)/K(\omega))$ は正接の逆関数、すなわち、$-\frac{\pi}{2} \sim \frac{\pi}{2}$ の範囲内の値で与えられるため、l が 1 m の場合は Fig. 6.12 に示すように位相角を的確に捉えることができなかった。そこで、フーリエ解析による位相速度の導出に関してのみ、l = 35 mm の 2 組の水位計（L/D = 9.4）のデータを用いて処理を行った（この場合の位相角を Fig. 6.13、コヒーレンスを Fig. 6.14 に示す）。

比較の結果、Morlet のウェーブレットによる位相速度と Gabor 関数による位相速度とは完全に一致したものの、深水波の理論値とは僅かな相違が見られた。とりわけ、3 Hz 以下の低周波成分に関しては大きな相違が見られたが、これは実験では界面波の伝播以外の要因が支配的であったためと思われる。Mexican Hat 型ウェーブレットによる位相速度は、卓越周波数である 6 Hz 付近では他の

ウェーブレットによる位相速度と一致したが、全体的に周波数に影響されにくい傾向を示した。これはMexican Hat型ウェーブレットの周波数分解能が相対的に悪いため、卓越周波数の影響が他の周波数領域に及んだためと思われる。なお、一連のウェーブレット解析の計算に用いたプログラムを付録Dに付す。

一方、フーリエ解析から求めた位相速度も全体的に周波数に影響されにくい傾向を示した。これは実験装置の制約上2組の水位計の間隔が35 mmと極端に短かったため、周波数成分毎の差異がはっきりと表れなかったことにも一因があると思われる。また、低周波側で大きく他と異なった傾向を示した理由として、平滑化のためにデータを約1秒の幅のウィンドーをかけて切り分けたことが考えられる。すなわち、数Hzという低周波成分を精度良く求めるには数分間以上の時間的に長いデータをフーリエ変換する必要があったのではないかと思われる。ウェーブレット変換で明確な相関が得られなかった20 Hz以上の高周波の成分に関しては、深水波の位相速度の傾向と定性的に異なる傾向が得られた。また、水位計の間隔が35 mmと短かったにも関わらず、コヒーレンスは20 Hzを境に急減した。こうしたことから20 Hz以上の高周波成分は、3 Hz以下の低周波成分と同様に界面波の伝播以外の要因が支配的な周波数領域であると思われる。

Gabor関数を用いて求めたウェーブレット変換係数の相互相関関数の場合、周波数が5.3 Hz、ラグが0.630 ms、0.722 msにおいて最も強い負及び正のピークが見られるが、これは包絡波の伝播によって生じた相関である。このラグから包絡波の速度、すなわち、群速度を求めると1.4 ～ 1.6 m/sとなり、その理論値

$$V_{group} + V_{liqiud} = \frac{\omega}{2k_0} + V_{liqiud} = \frac{1}{2}\sqrt{\frac{g}{k_0}} + V_{liqiud} \qquad \sim 1.4 \text{ (m/s)} \qquad (6.3.4)$$

とほぼ一致した。ただし、水の流速 V_{liqiud} を1.03 m/s（水が流路断面において均一の流速で流れていると仮定して算出）、波数 k_0 を19.1 m^{-1}（水の流速を考慮した場合の周波数5.3 Hzの波に相当）とした。僅かではあるが実験値が理論値よりやや大きめの値を示したのは、水の表層付近が気液の相互作用により加速されているためと思われる。

これらの結果を踏まえてFig. 6.4及び6.5に着目すると、シフト量1.0 ～ 1.3 sの範囲にまたがって存在する界面波は、有意な波高がほとんど見られないものの、この水位計を通過する直前までは波高の大きな界面波であたっと思われる。すなわち、この界面波はシフト量1.3 ～ 1.4 sの範囲に存在する界面波のように各周波数成分の位相が揃った界面波であったが、各々の成分が独立した位相速度で伝播したため、明瞭な形状を持たなくなった。しかし、各々の周波数成分は消滅したわけでないためウェーブレット変換を行うと、周波数が低い成分ほど早い時刻に水位計を通過したという履歴が現れる。こうしたことから、界面波の急成長、急減衰は種々の周波数成分の位相が揃うこと、ずれることにより生じることが明らかになった。

7. 結　　論

大型水平ダクト試験装置において観測された成長過程にある界面波の突然の急減衰／急成長についてフーリエ解析とウェーブレット解析を用いて評価を行った。その結果、

1）界面波は1つの卓越周波数とそこからべき関数的に減衰するパワースペクトル密度関数を有することを示した。また、パワースペクトル密度関数は伝播に従って、卓越周波数が低周波側にシフトすること、卓越周波数におけるスペクトル密度関数の値が増大すること、卓越周波数より高い周波数成分のスペクトル密度関数の値はほとんど変化しないことを明らかにした。

2）波状流の水位データに関し、Morletのウェーブレットによる変換係数とGabor関数による変換係数とがほぼ一致することを明らかにした。また、Mexican Hat型ウェーブレットによる変換係数は、周波数分解能が相対的に劣るために卓越周波数の影響を強く受けることを明らかにした。

3）Morletのウェーブレット及びGabor関数によるウェーブレット解析により、卓越周波数以外の周波数成分も各々固有の位相で伝播していることを明らかにした。また、波高の大きい界面波については、各周波数成分の位相が広い周波数域において揃っていることを明らかにした。

4）Morletのウェーブレット及びGabor関数によるウェーブレット解析により、界面波を構成する3～10 Hzの周波数成分は、深水波の位相速度並びに群速度の理論値とほぼ一致した速度で伝播することを明らかにした。また、この周波数域より高い周波数成分はコヒーレントでなく、低い周波数成分は界面波に関係ない他の要因が支配的であることを明らかにした。

5）界面波の突然の急減衰／急成長は界面波を構成する種々の周波数成分の位相がずれること及び重なることにより生じることを明らかにした。

謝　　辞

ウェーブレット解析に関する多くの助言を下さった茨城大学工学部の神永文人教授に感謝致します。また、大型水平ダクト試験装置を用いた実験を行うにあたり、種々の協力を頂いた熱水力安全研究室の中村秀夫氏、装置を運転・整備して頂いた（株）原子力エンジニアリングの方々に感謝致します。実験データ処理には（株）アイ・ティ・ジェイの方々のご協力を頂きました。

7. 結　　論

大型水平ダクト試験装置において観測された成長過程にある界面波の突然の急減衰／急成長についてフーリエ解析とウェーブレット解析を用いて評価を行った。その結果、

1）界面波は1つの卓越周波数とそこからべき関数的に減衰するパワースペクトル密度関数を有することを示した。また、パワースペクトル密度関数は伝播に従って、卓越周波数が低周波側にシフトすること、卓越周波数におけるスペクトル密度関数の値が増大すること、卓越周波数より高い周波数成分のスペクトル密度関数の値はほとんど変化しないことを明らかにした。

2）波状流の水位データに関し、Morlet のウェーブレットによる変換係数と Gabor 関数による変換係数とがほぼ一致することを明らかにした。また、Mexican Hat 型ウェーブレットによる変換係数は、周波数分解能が相対的に劣るために卓越周波数の影響を強く受けることを明らかにした。

3）Morlet のウェーブレット及び Gabor 関数によるウェーブレット解析により、卓越周波数以外の周波数成分も各々固有の位相で伝播していることを明らかにした。また、波高の大きい界面波については、各周波数成分の位相が広い周波数域において揃っていることを明らかにした。

4）Morlet のウェーブレット及び Gabor 関数によるウェーブレット解析により、界面波を構成する3～10 Hz の周波数成分は、深水波の位相速度並びに群速度の理論値とほぼ一致した速度で伝播することを明らかにした。また、この周波数域より高い周波数成分はコヒーレントでなく、低い周波数成分は界面波に関係ない他の要因が支配的であることを明らかにした。

5）界面波の突然の急減衰／急成長は界面波を構成する種々の周波数成分の位相がずれること及び重なることにより生じることを明らかにした。

謝　　辞

ウェーブレット解析に関する多くの助言を下さった茨城大学工学部の神永文人教授に感謝致します。また、大型水平ダクト試験装置を用いた実験を行うにあたり、種々の協力を頂いた熱水力安全研究室の中村秀夫氏、装置を運転・整備して頂いた（株）原子力エンジニアリングの方々に感謝致します。実験データ処理には（株）アイ・ティ・ジェイの方々のご協力を頂きました。

参 考 文 献

1. 植田 辰洋："気液二相流", 養賢堂 (1981).
2. 近藤 昌也, 中村 秀夫, 安濃田 良成, 久木田 豊："大型水平ダクト実験による分離流からスラグ流への二相流流動様式遷移条件の評価", JAERI-M 92-016 (1992).
3. Charles K. Chui："ウェーブレット入門", 東京電機大学出版局 (1993).
4. 山田 道夫："ウェーブレット変換とは何か", 数理科学, Vol. 12, pp. 11-17 (1992).
5. 山口 昌哉, 山田 道夫："ウェーブレット解析", 科学, Vol. 60, No. 6, pp. 398-405 (1990).
6. 佐藤 雅昭："ウェーブレット理論の数学的基礎　第I部", 日本音響学会誌, Vol. 47, No. 6, pp. 405-415 (1991).
7. 菊池 久和："異常燃焼のウェーブレット解析", 数理科学, Vol. 12, pp. 44-51 (1992).
8. 太田黒 俊夫："乱流への応用", 数理科学, Vol. 12, pp. 18-23 (1992).
9. Marie Farge："Wavelet Transforms and Their Applications to Turbulence", Annu. Rev. Fluid Mech., Vol. 24, pp. 395-457 (1992).
10. 神永 文人, 柴田 裕一, 金澤 隆弘："対向二相流における気液界面速度のウェーブレット変換による解析", 第32回日本伝熱シンポジウム講演論文集, pp. 483-484 (1995).
11. 井上裕嗣, 岸本 喜久雄, 中西 智明, 渋谷 壽一："ウェーブレット変換による分散性応力波の時間-周波数解析", 日本機械学会論文集A編, Vol. 61, No. 581, pp. 153-160 (1995).
12. 佐藤 雅昭："ウェーブレット理論の数学的基礎　第II部", 日本音響学会誌, Vol. 47, No. 6, pp. 416-423 (1991).
13. 山口 昌哉, 守本 晃："ウェーブレットとその応用 [I]", 計測と制御, Vol. 31, No. 8, pp. 879-886 (1992).
14. 山口 昌哉："カオス・フラクタル・ウェーブレット", 数理科学, Vol. 12, pp. 5-10 (1992).
15. 守本 晃："ウェーブレットを用いた数値計算について", 数理科学, Vol. 12, pp. 36-43 (1992).
16. 山口 昌哉, 守本 晃："ウェーブレットとその応用 [II]", 計測と制御, Vol. 31, No. 10, pp. 1066-1074 (1992).
17. 日野 幹雄："スペクトル解析", 朝倉書店 (1977).
18. 日本流体力学会："流体における波動", 朝倉書店 (1989).

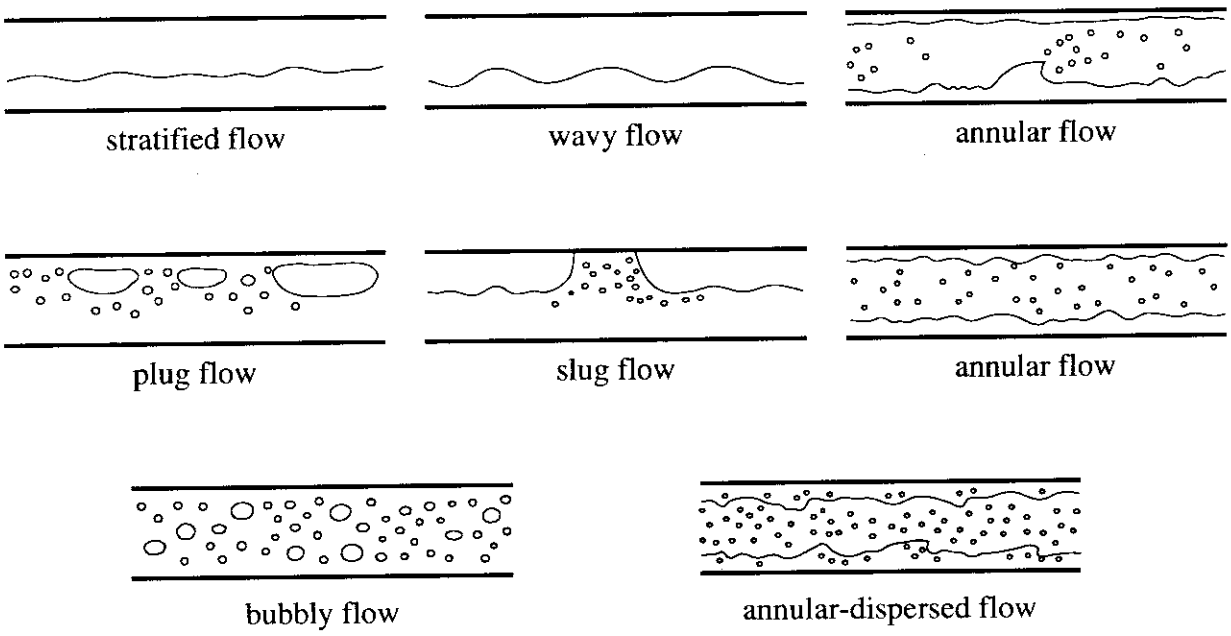


Fig. 1.1 Flow Pattern in Horizontal Piping

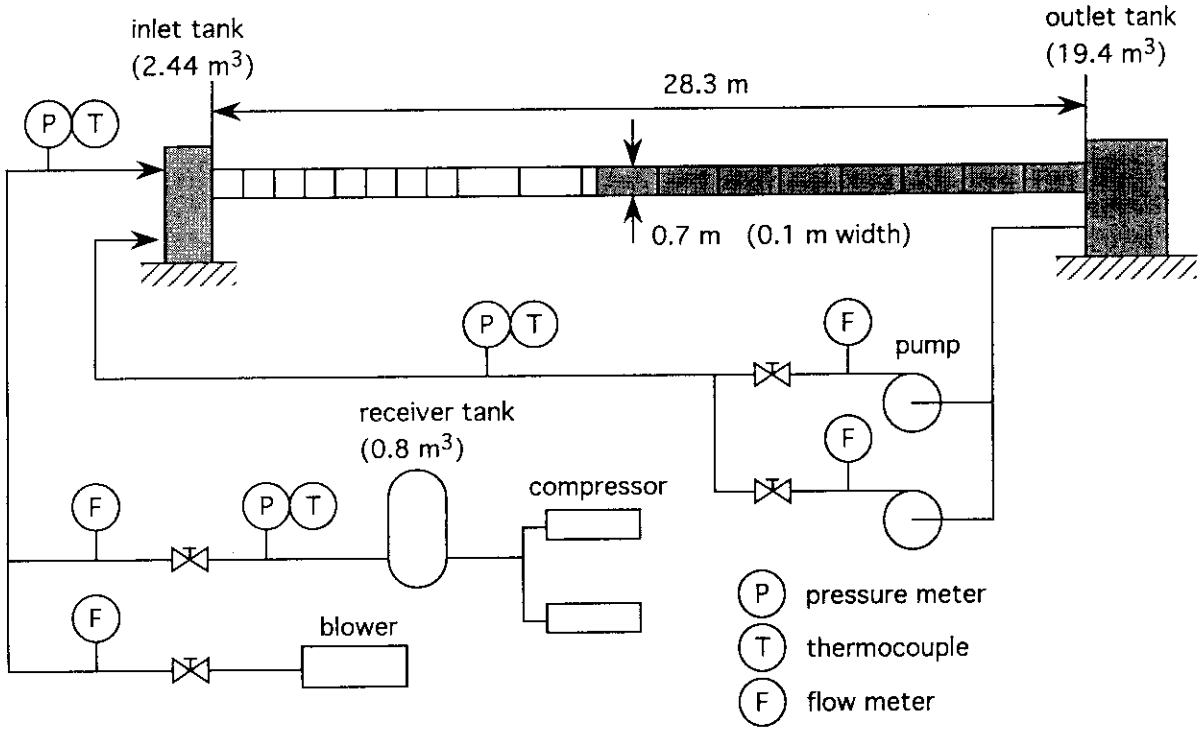
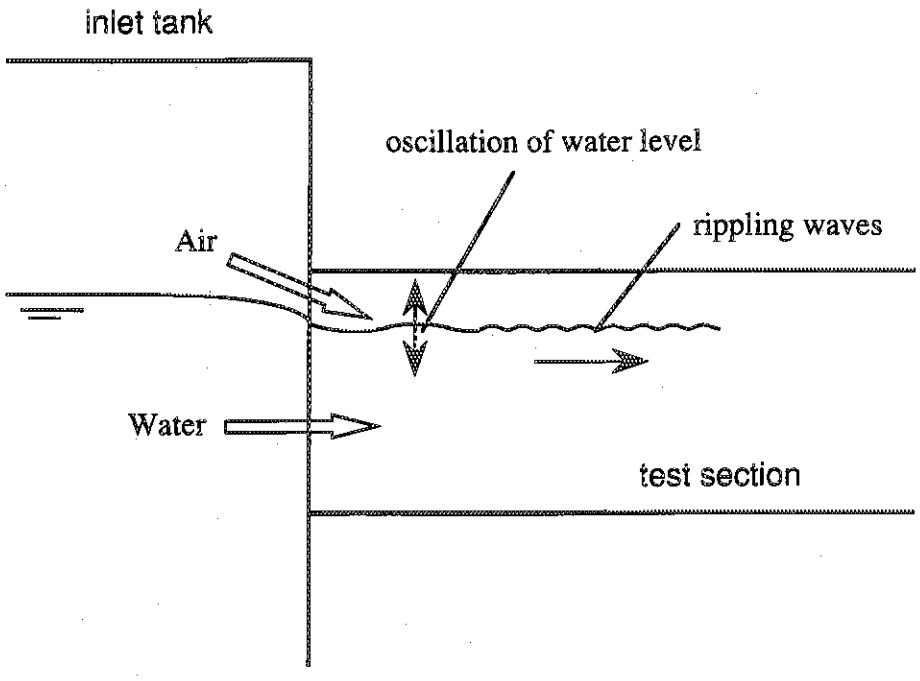


Fig. 2.1 Test Facility



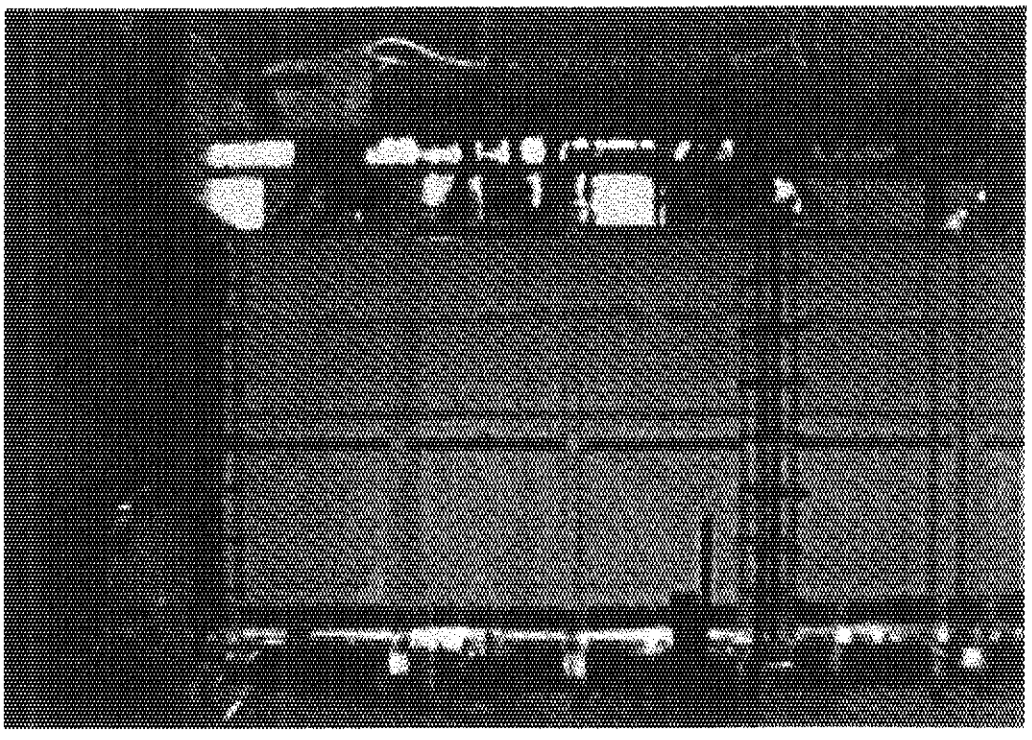


Fig. 3.1 Water Level Rise at Test Section Inlet

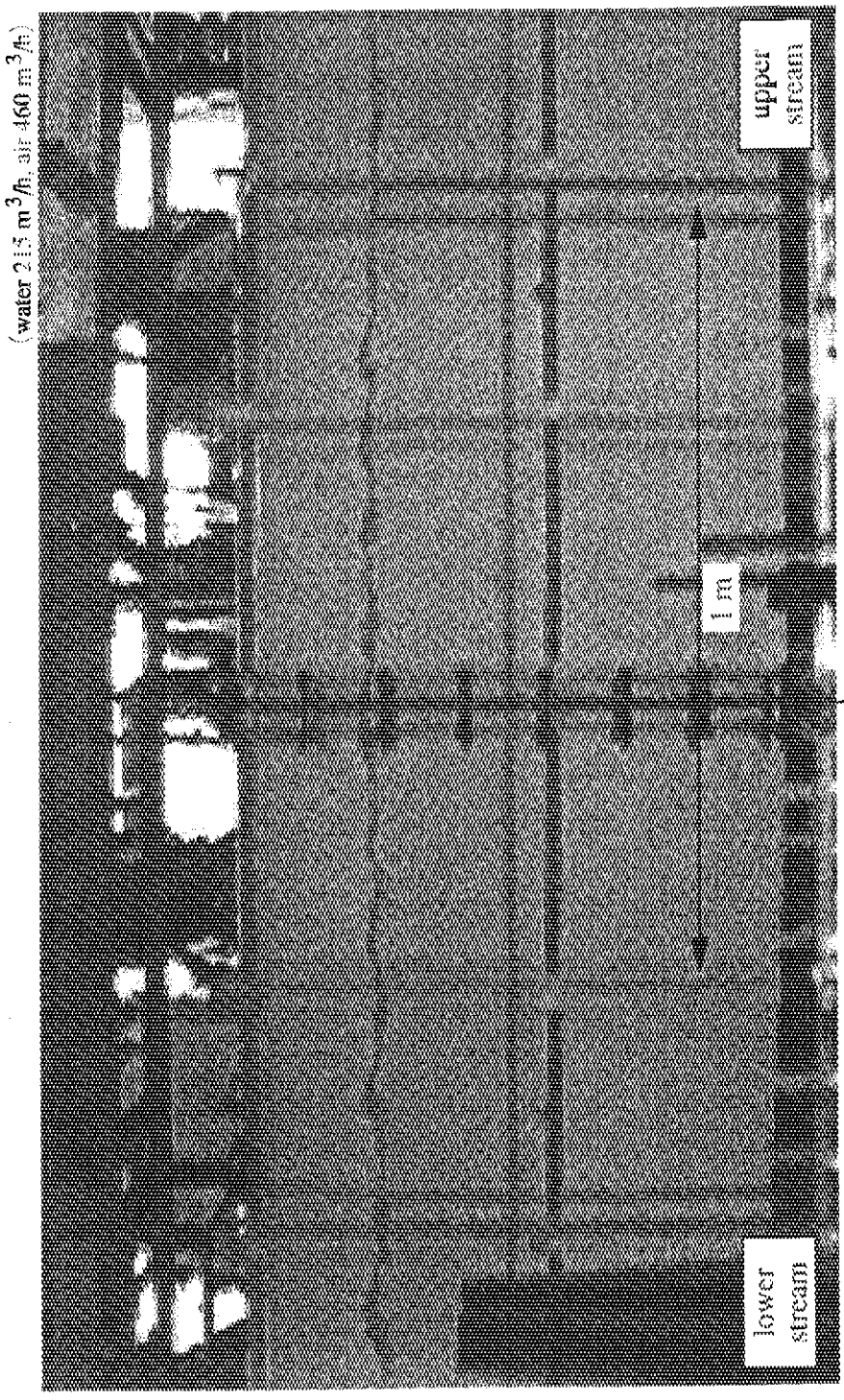


Fig. 3.2 Flow Pattern in Test Section Upper Stream (L/D ~1.4)

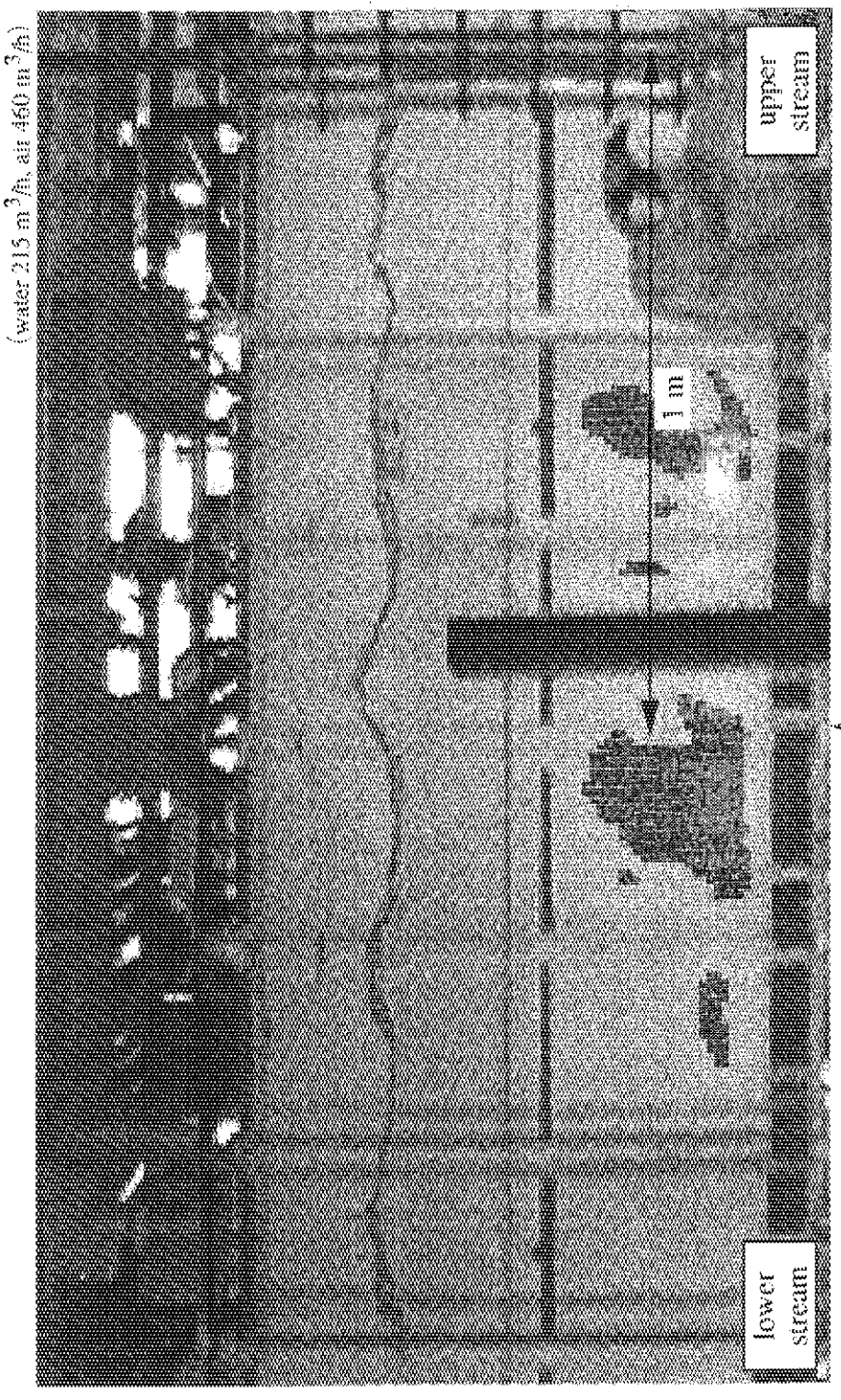


Fig. 3.3 Flow Pattern in Test Section Middle Reaches (L/D ~7.1)

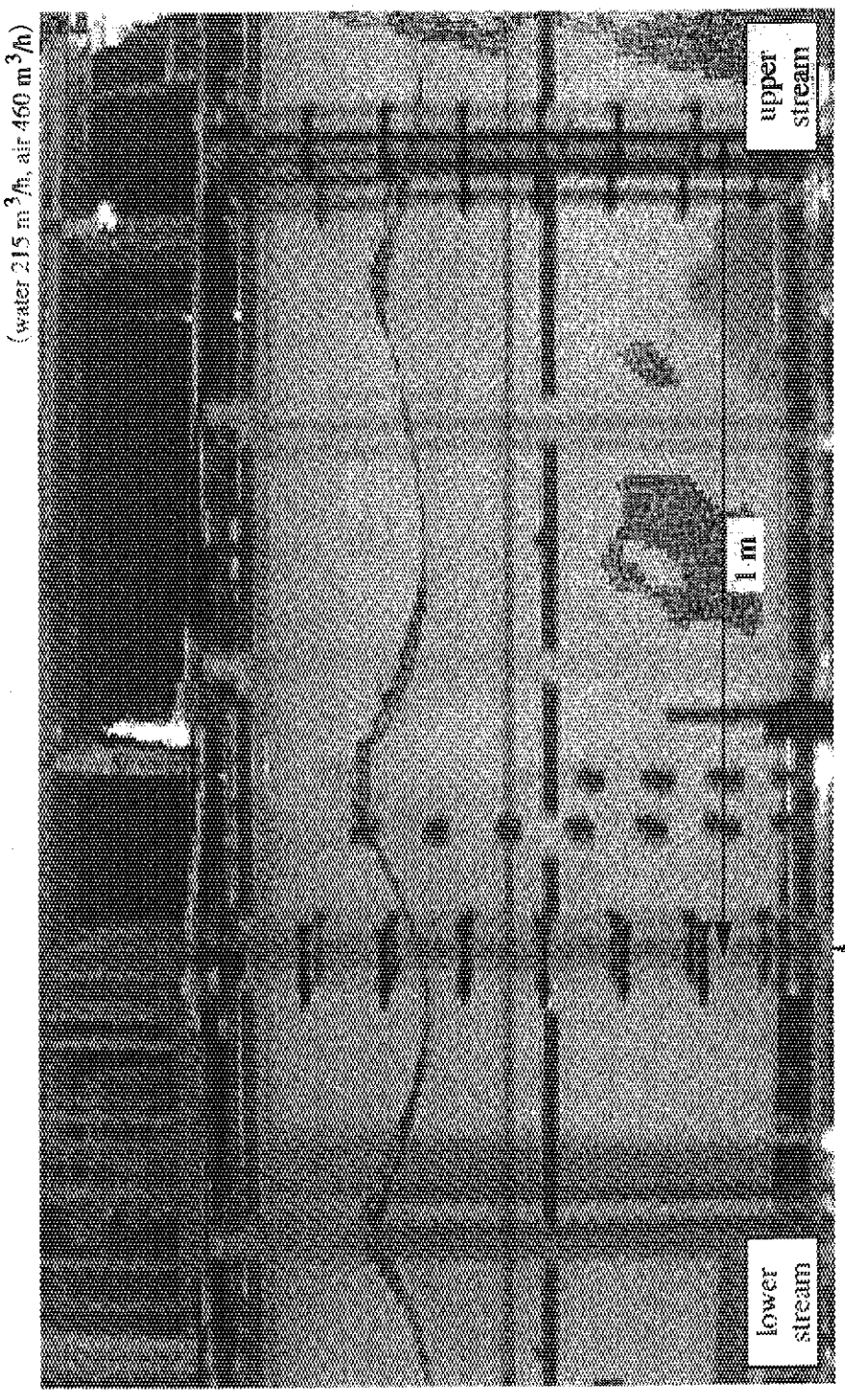


Fig. 3.4 Flow Pattern in Test Section Lower Stream (L/D ~15.7)

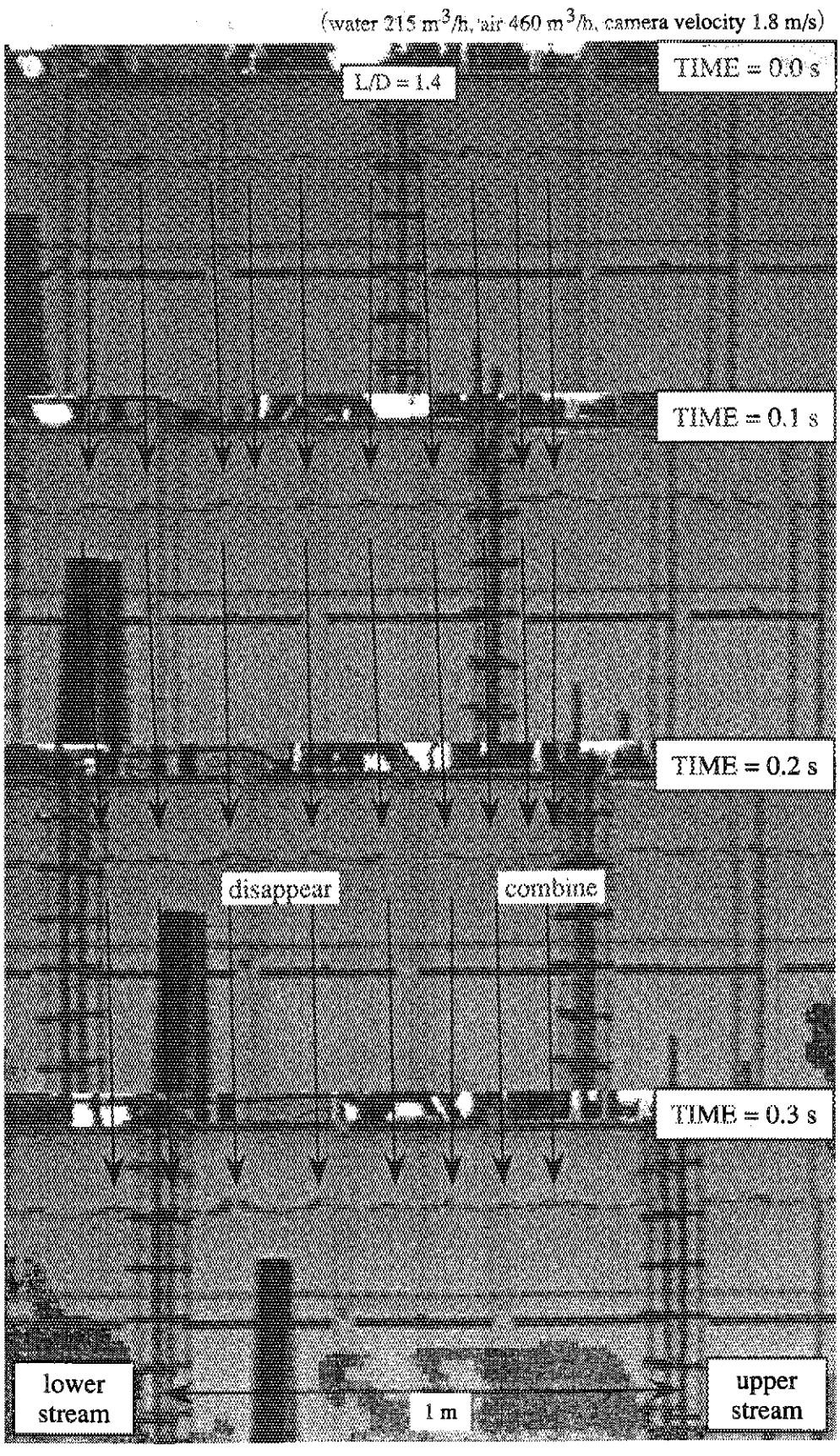


Fig 3.5 Rippling Wave Growth

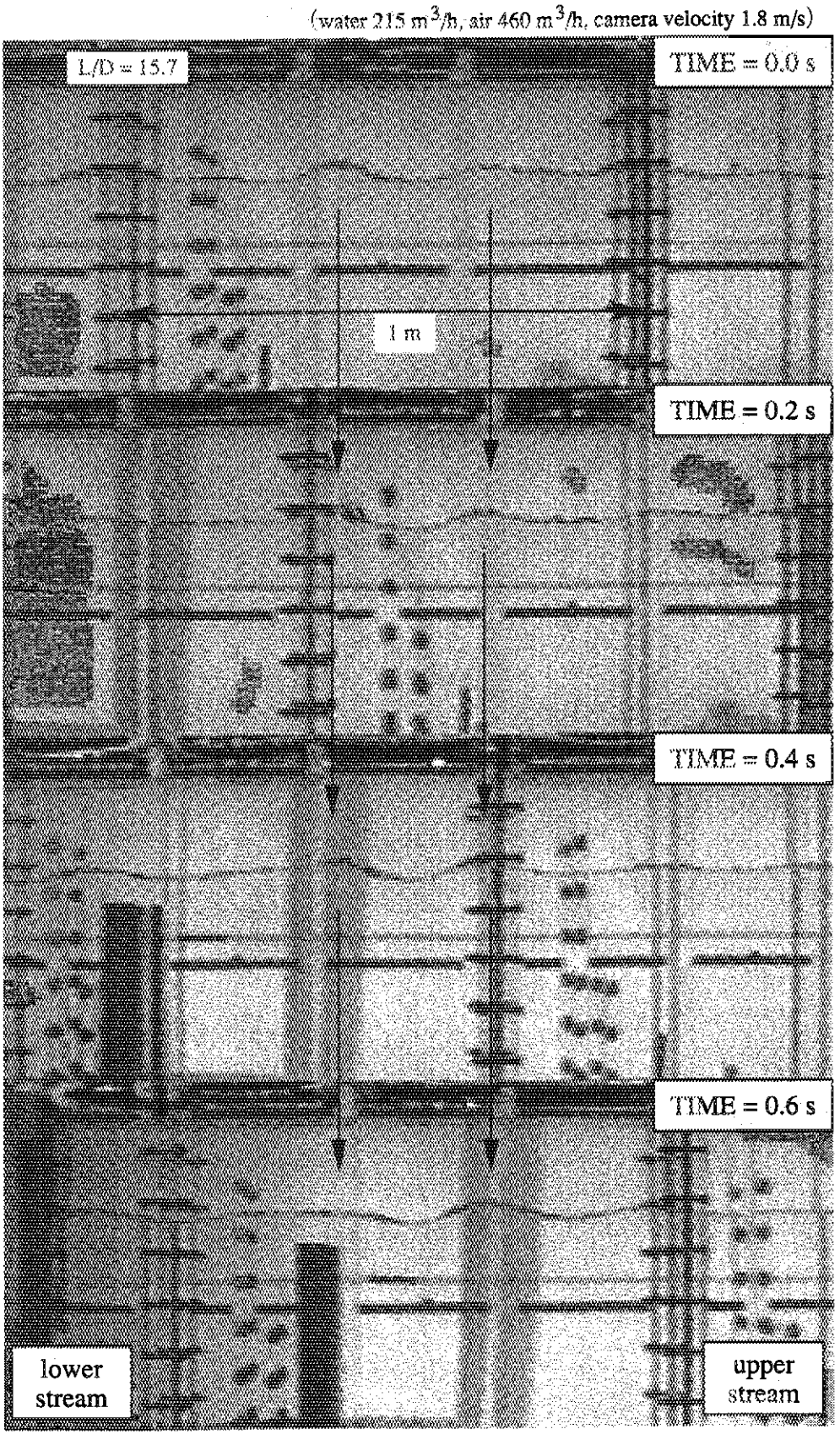


Fig 3.6 Sudden Wave Disappear and Growth

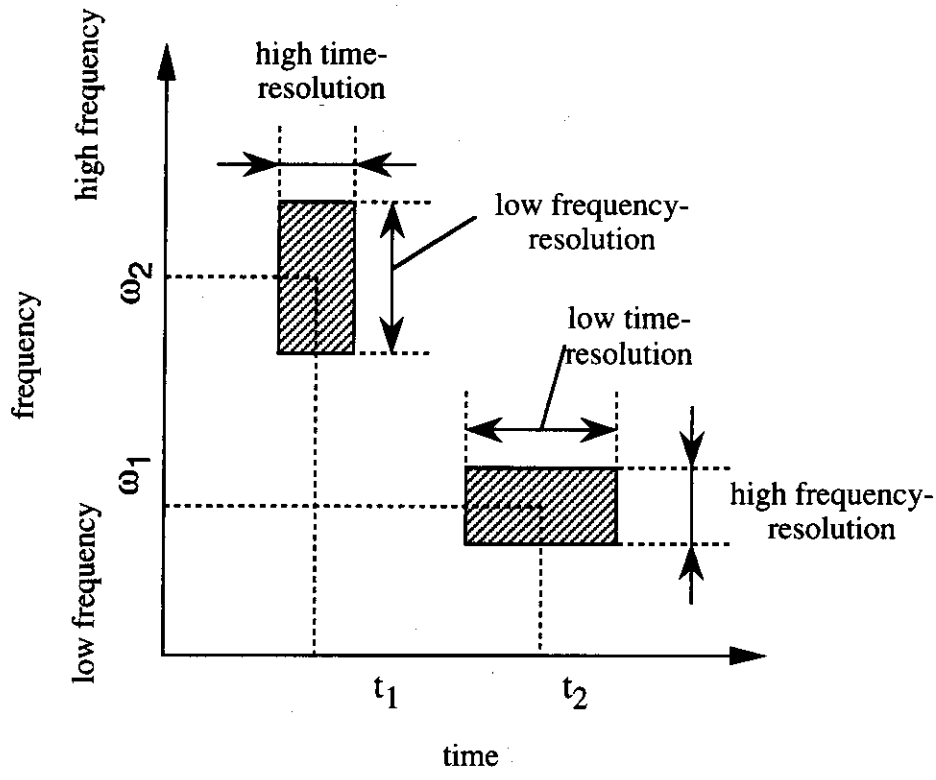
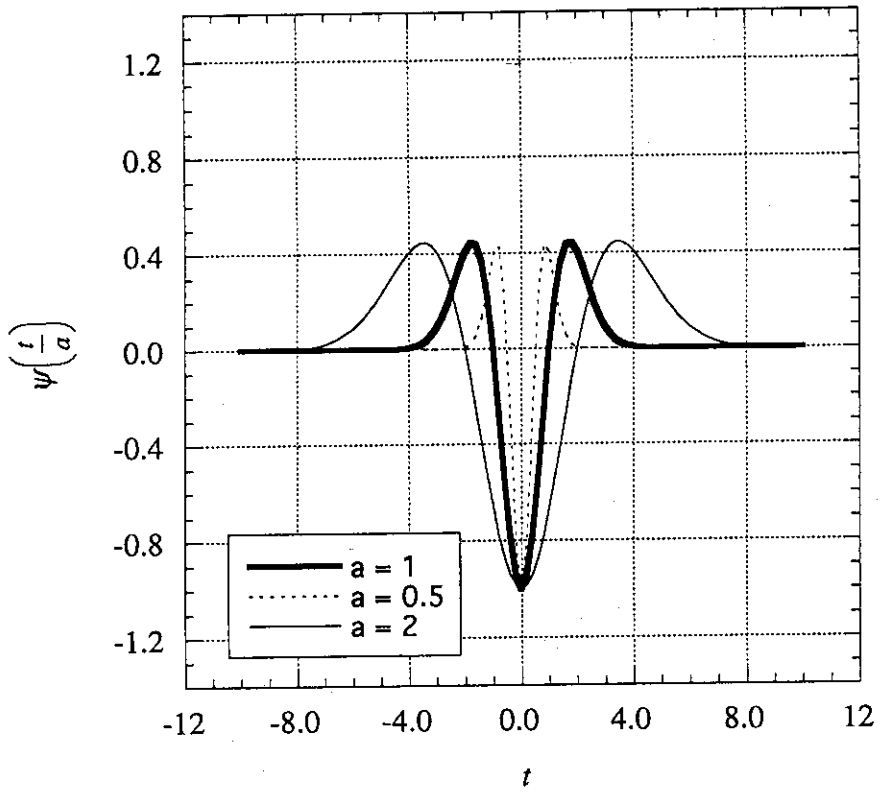


Fig. 4.1 Wavelet Transform Resolution on Time-Frequency Domain



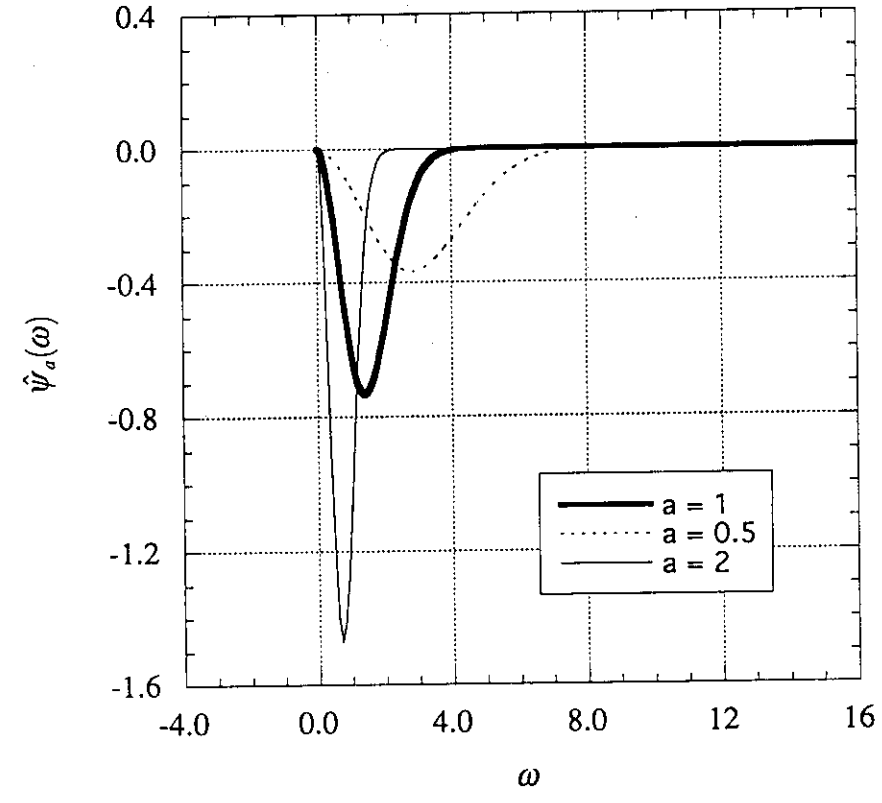


Fig. 4.2 Wavelet Function Shape with Various Scaling Parameter (Mexican Hat Wavelet)

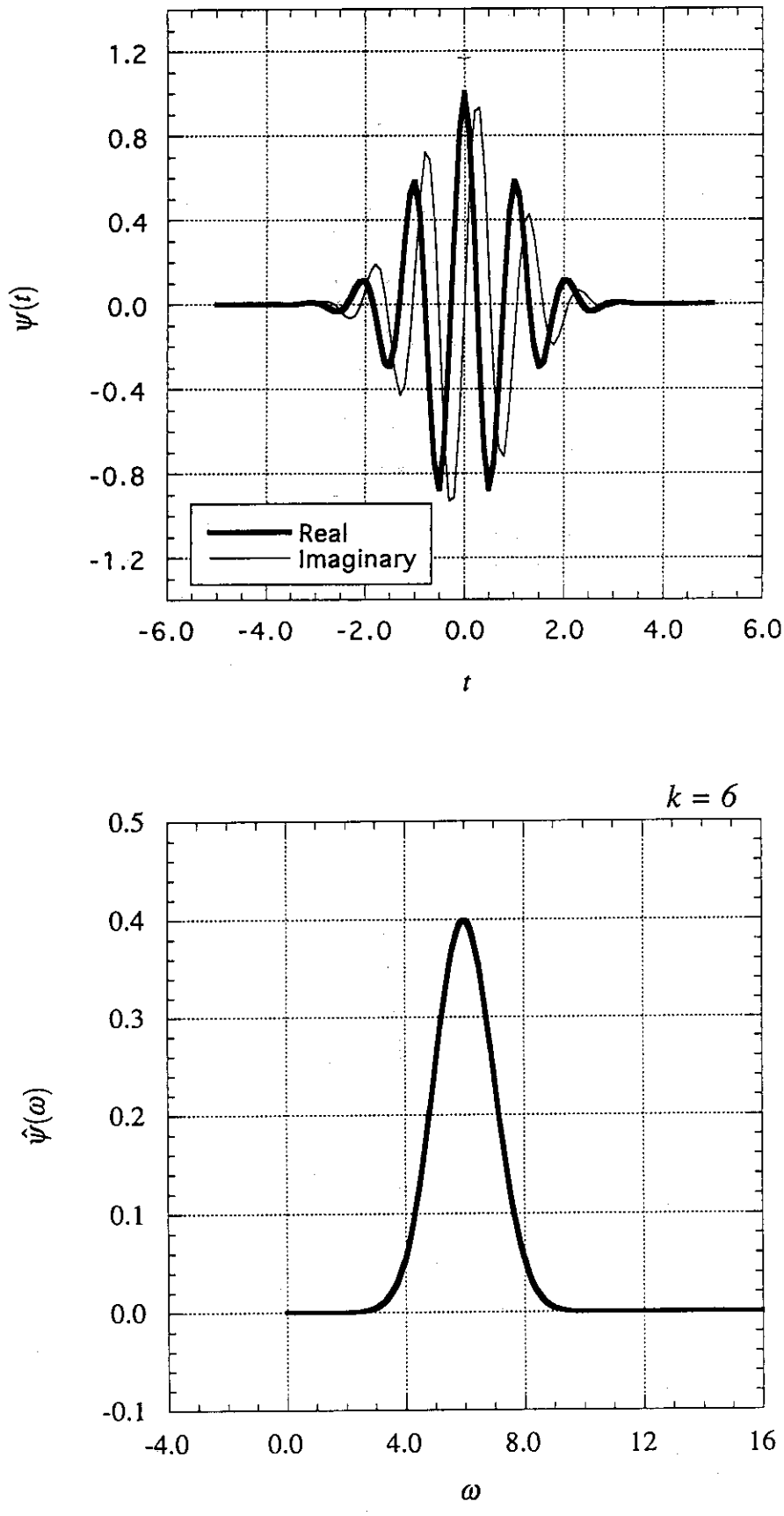
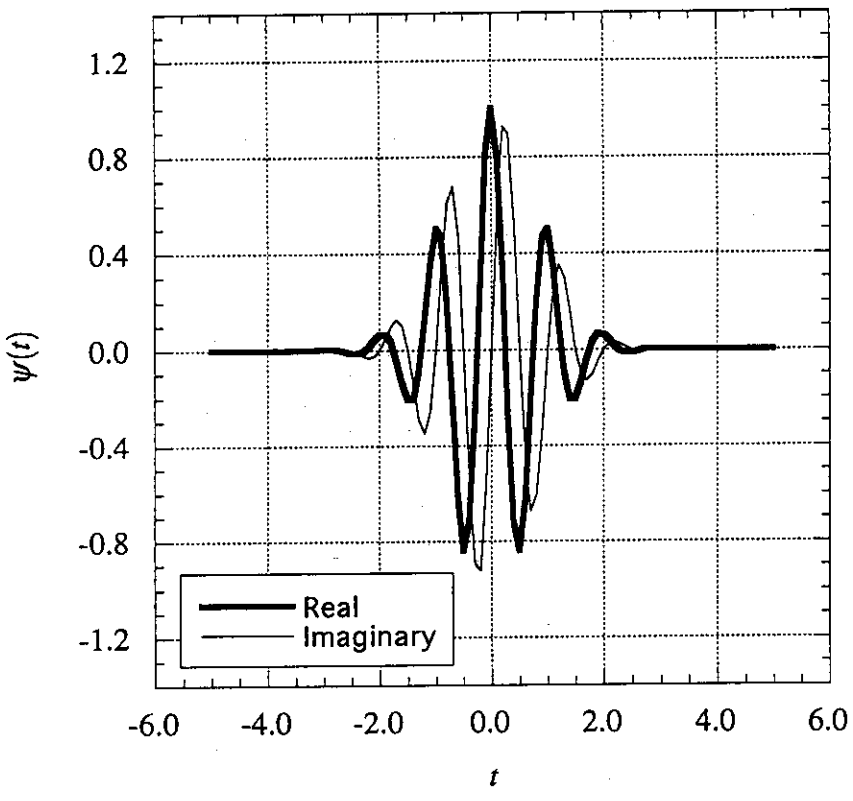


Fig. 4.3 Morlet Wavelet



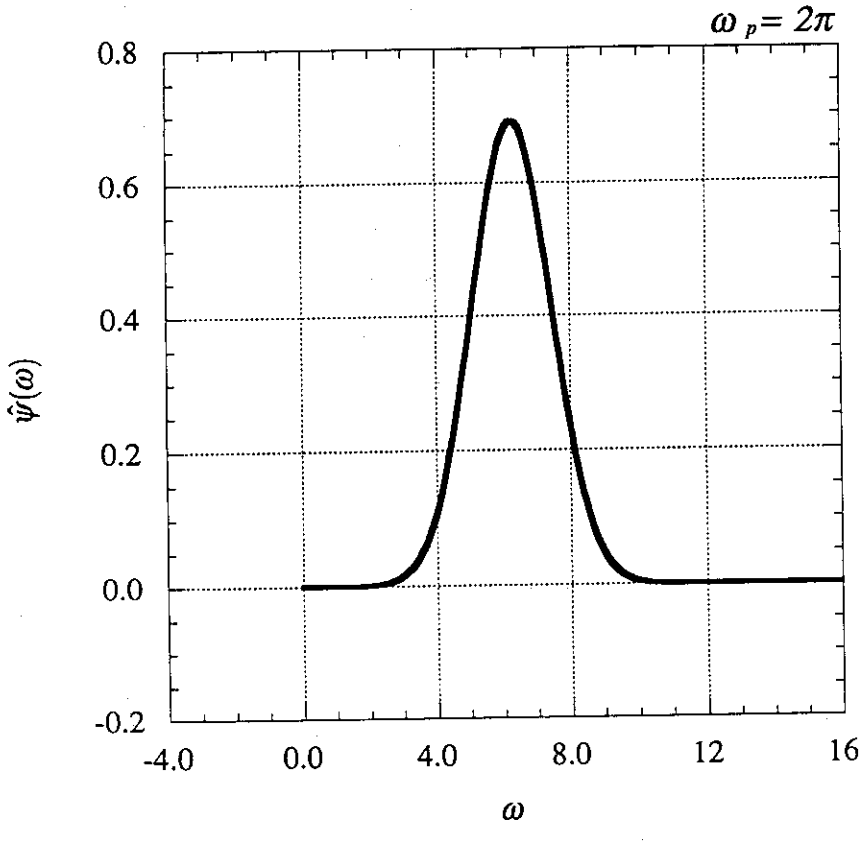


Fig. 4.4 Gabor Function

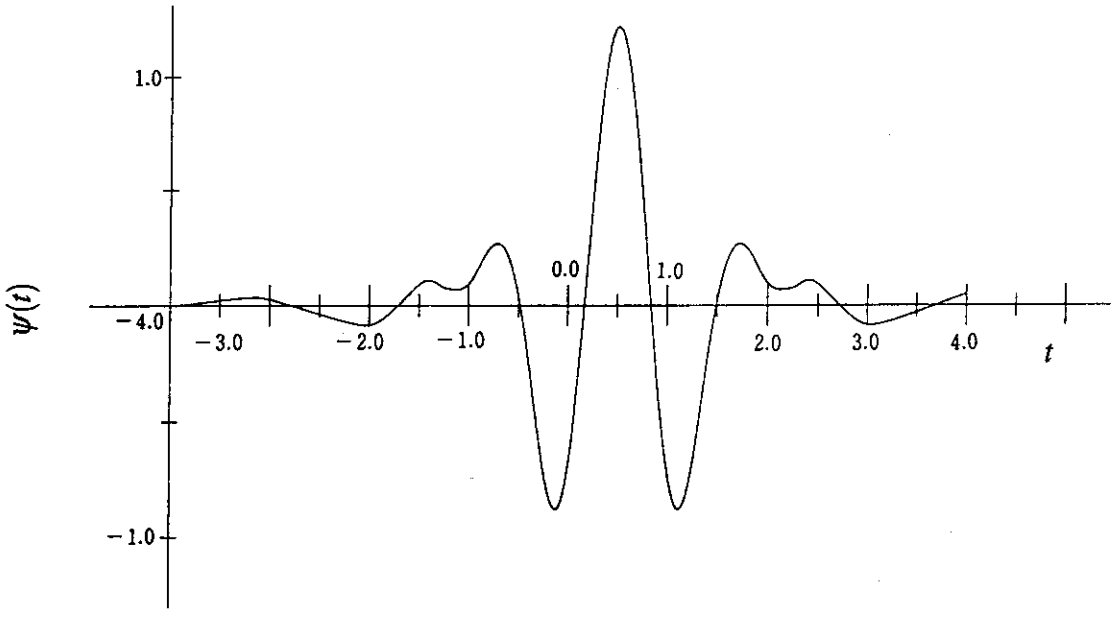


Fig. 4.5 Meyer Wavelet

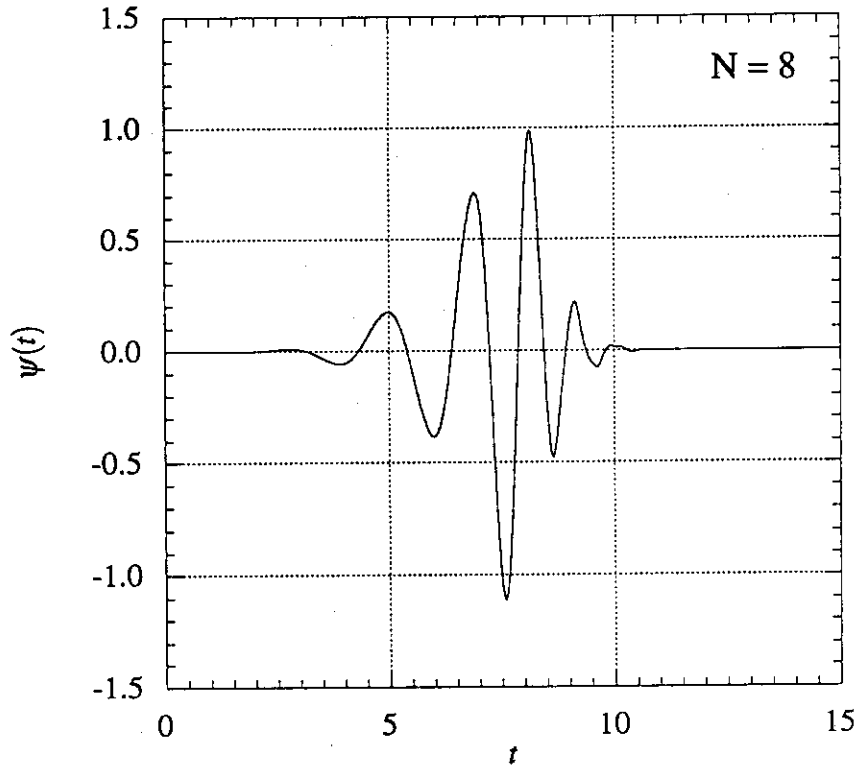


Fig. 4.6 Daubechies Wavelet (N = 8)

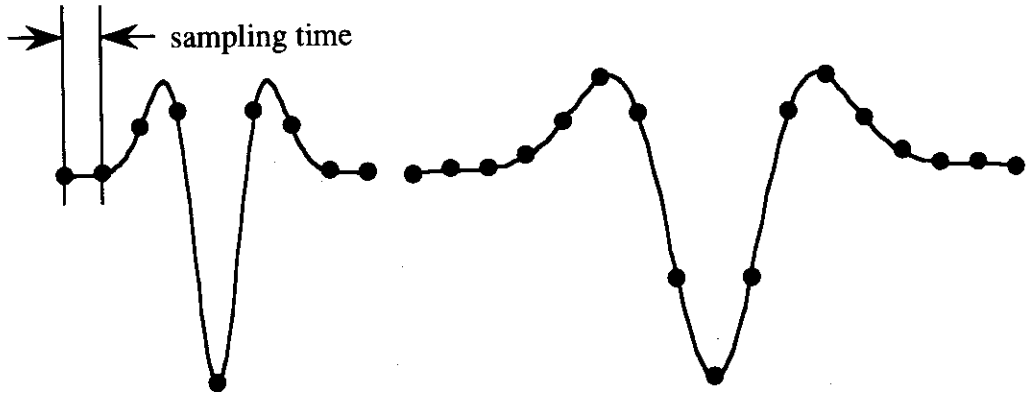


Fig. 5.1 Wavelet Function Dispersion

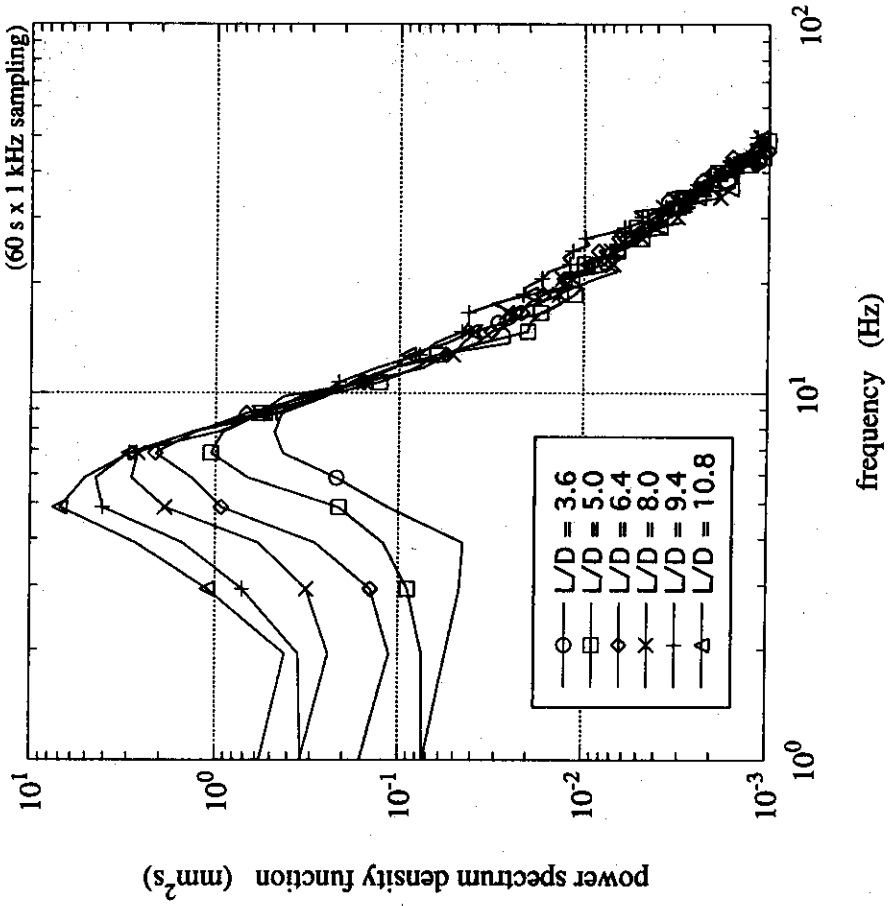


Fig. 6.2 Power Spectrum Density Function of Water Levels

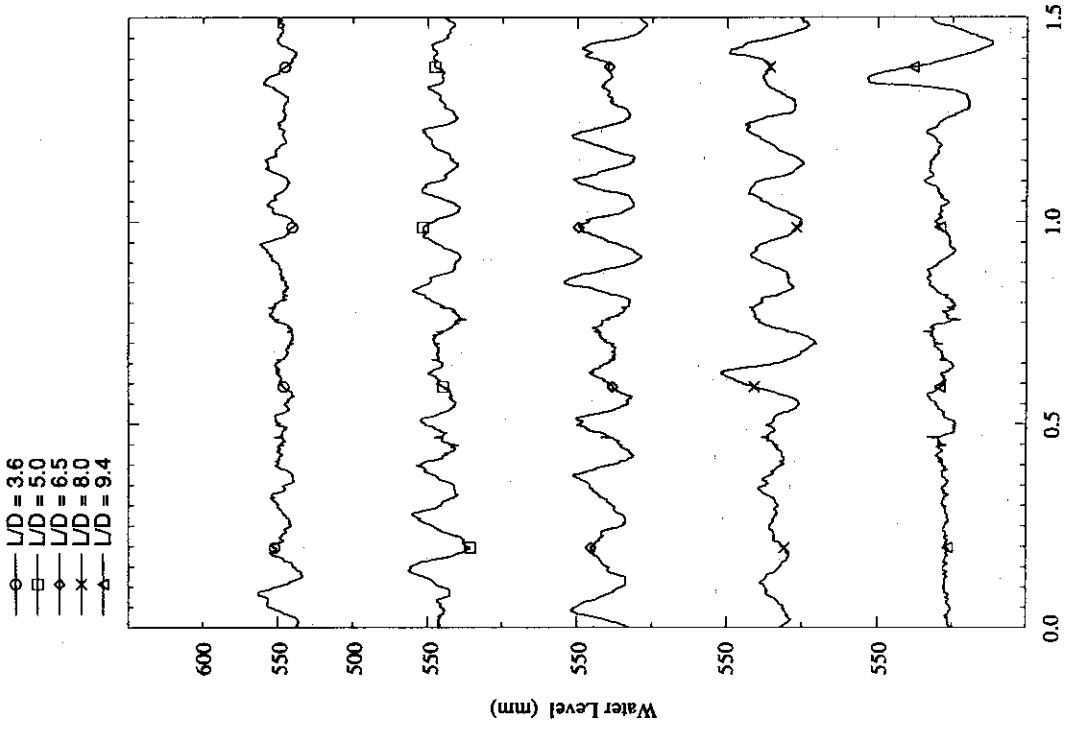


Fig. 6.1 Water Level Data (Wavy Flow)

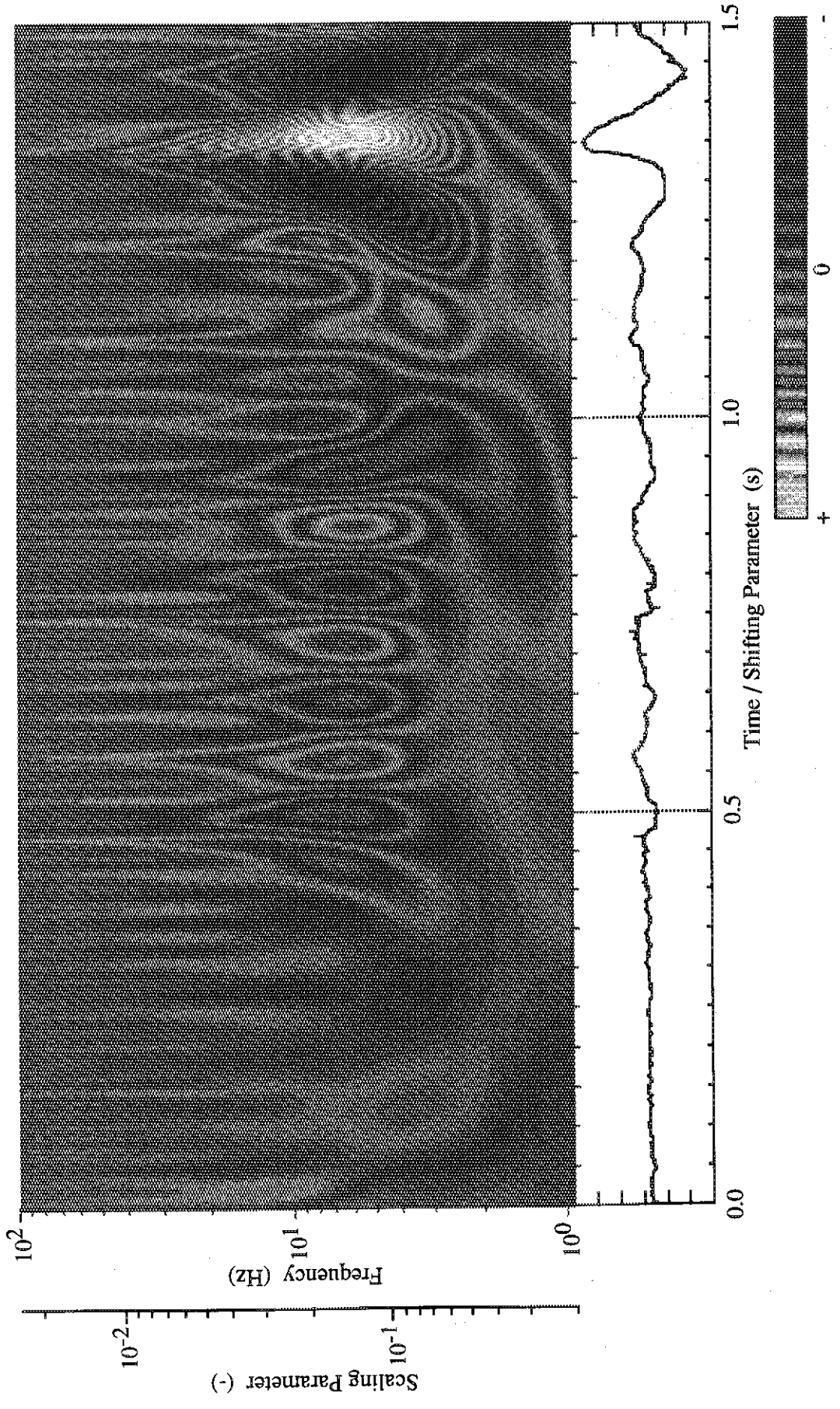


Fig. 6.3 Water Level and Wavelet Transform Coefficients (Mexican Hat Wavelet)

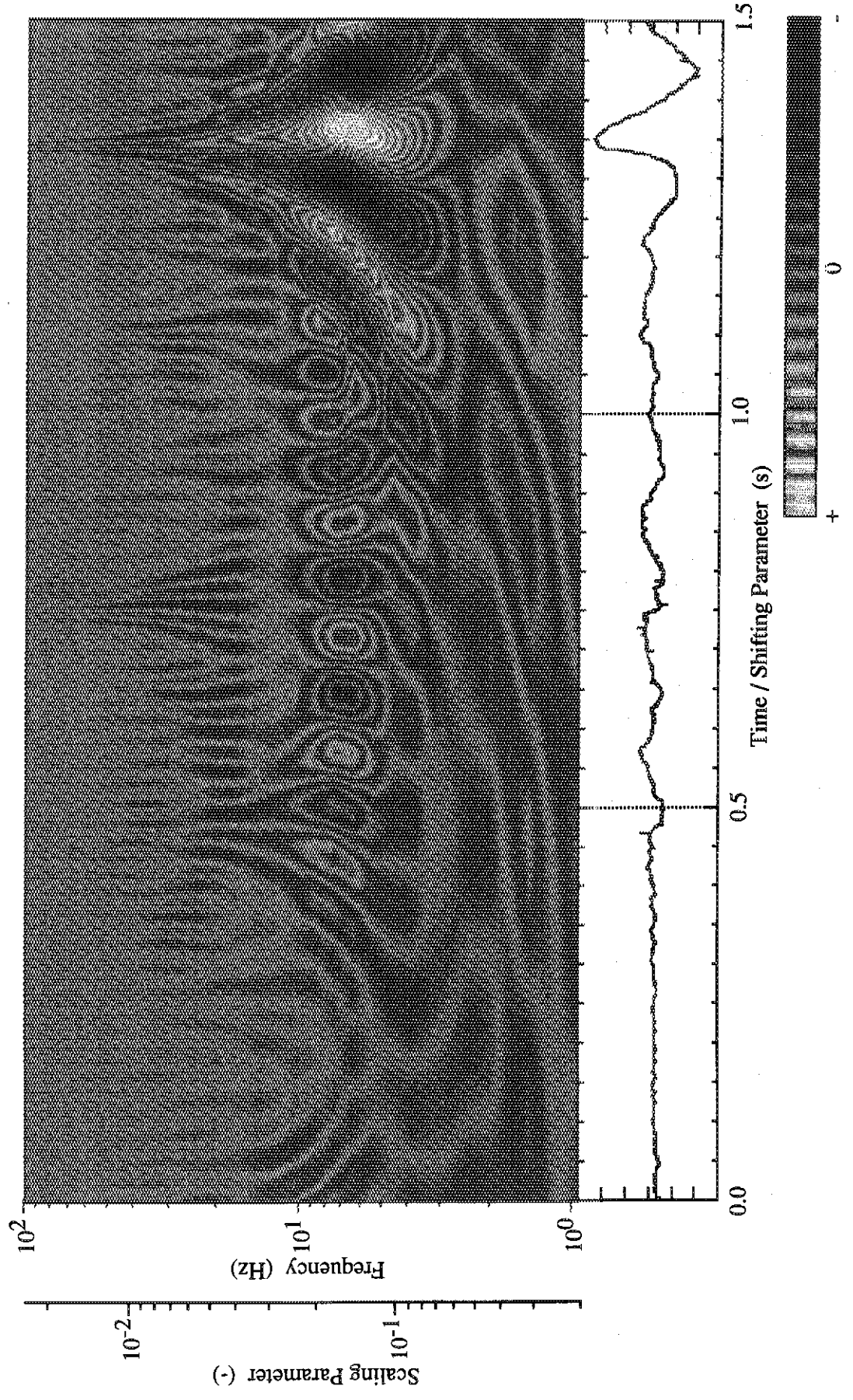


Fig. 6.4 Water Level and Wavelet Transform Coefficients (Morlet Wavelet)

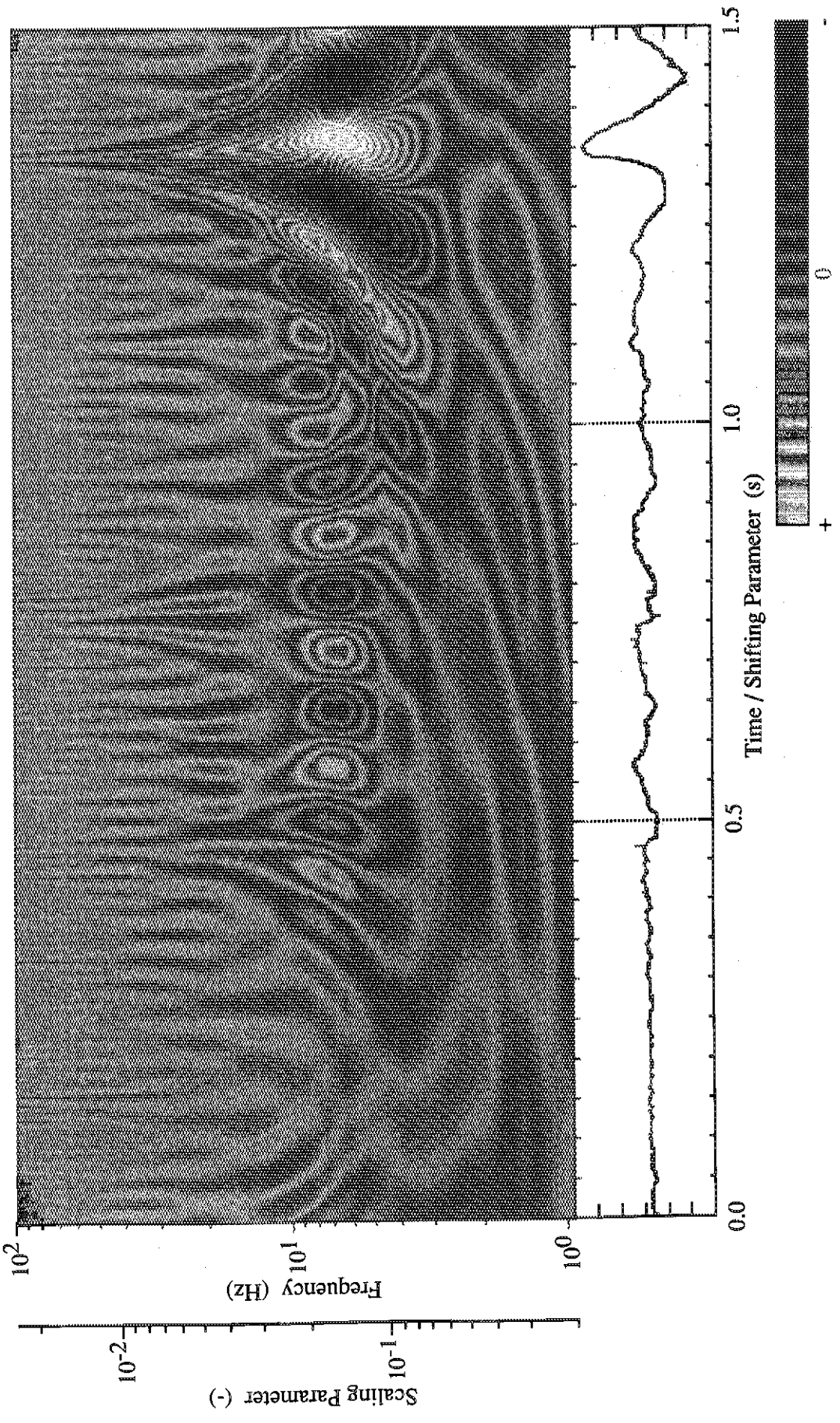


Fig. 6.5 Water Level and Wavelet Transform Coefficients (Gabor Function)

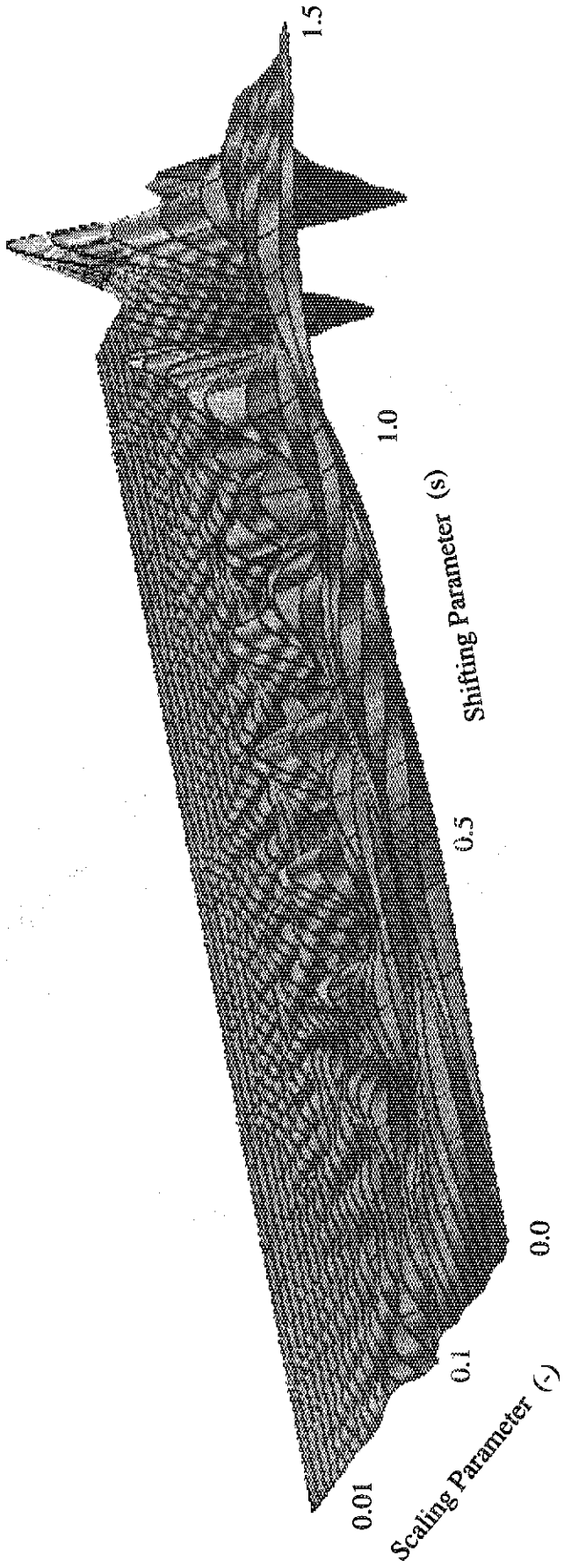


Fig. 6.6 Wavelet Transform Coefficients of Water Level (Gabor Function)

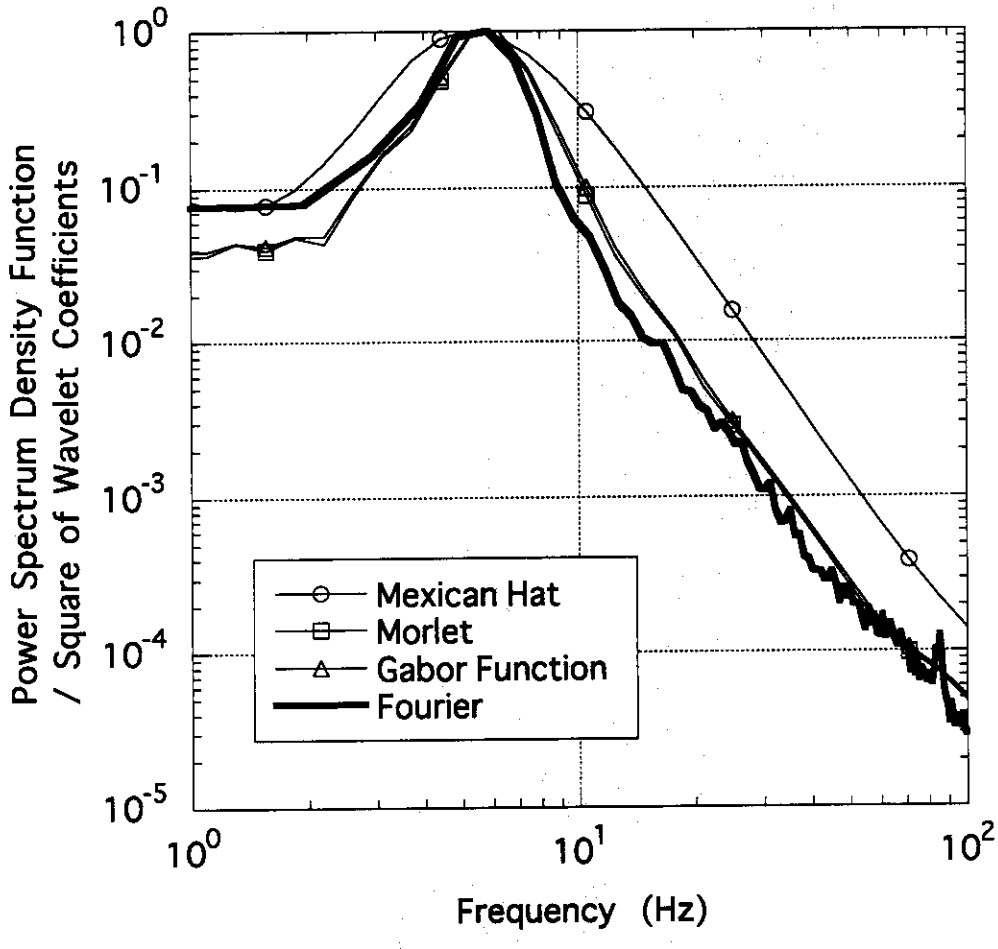


Fig.6.7 Comparison of Power Spectrum Density Function with square of Wavelet Coefficients

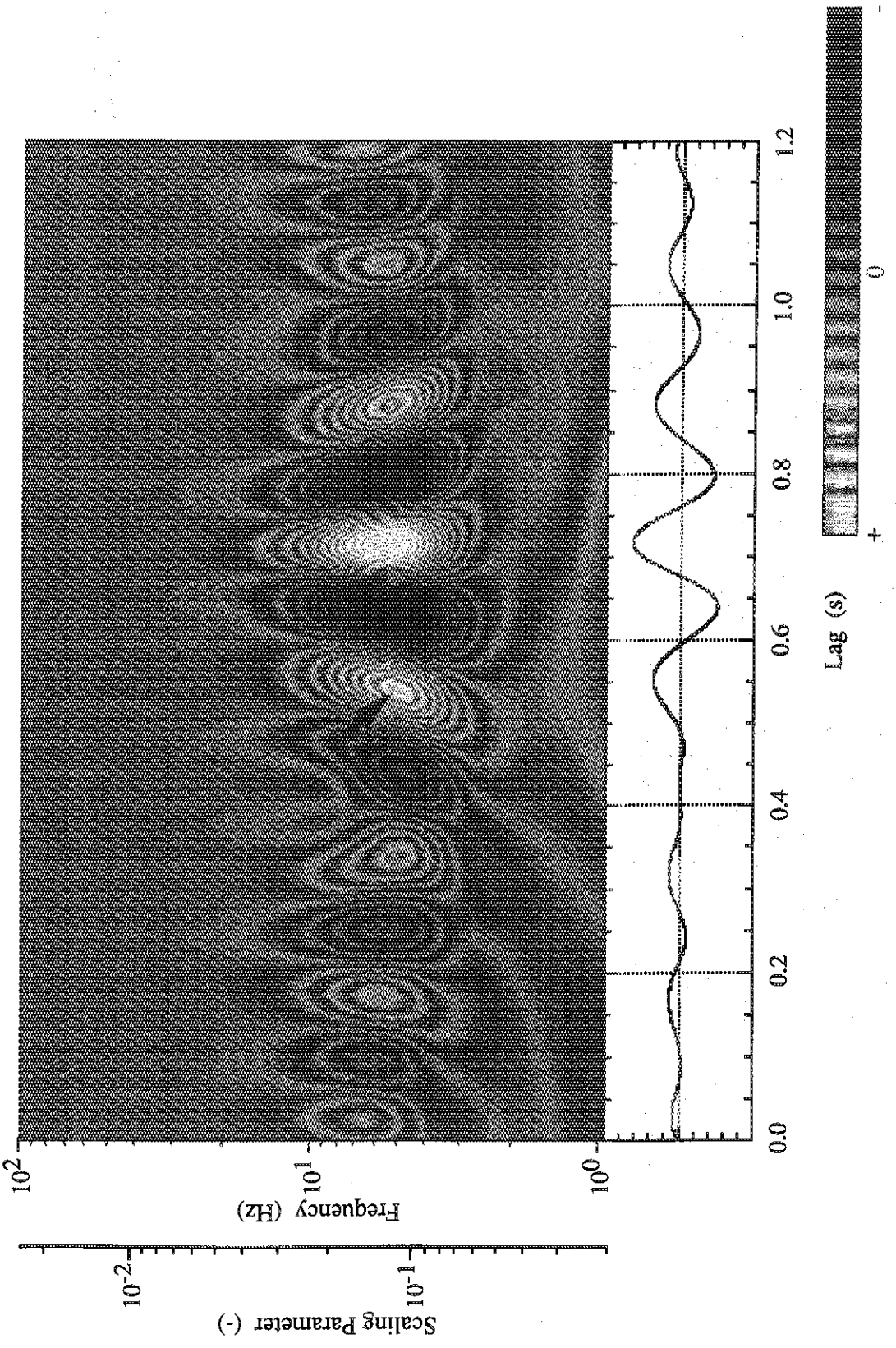


Fig. 6.8 Cross-Correlation of Wavelet Coefficients (Mexican Hat Wavelet)

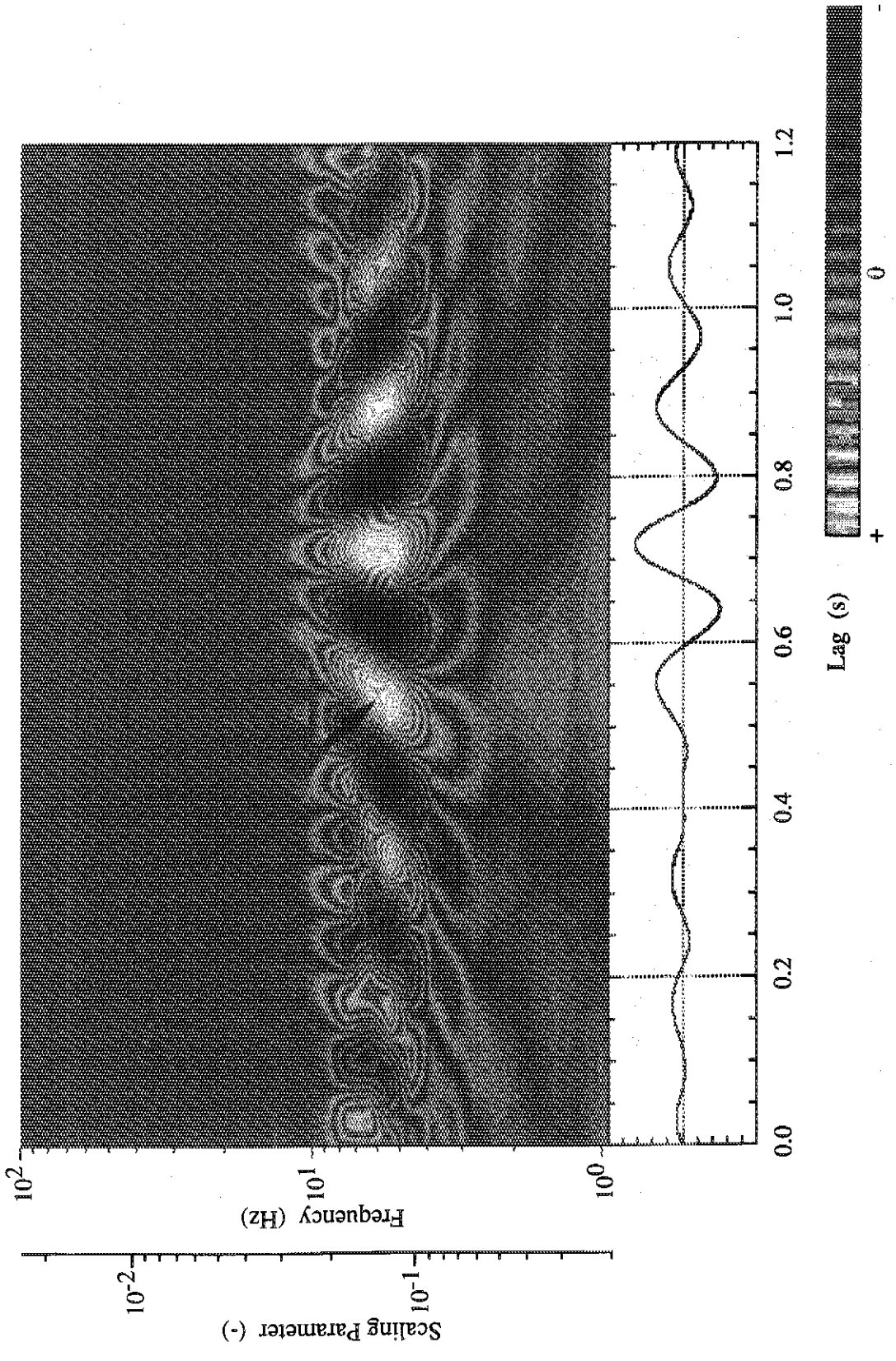


Fig. 6.9 Cross-Correlation of Wavelet Coefficients (Morlet Wavelet)

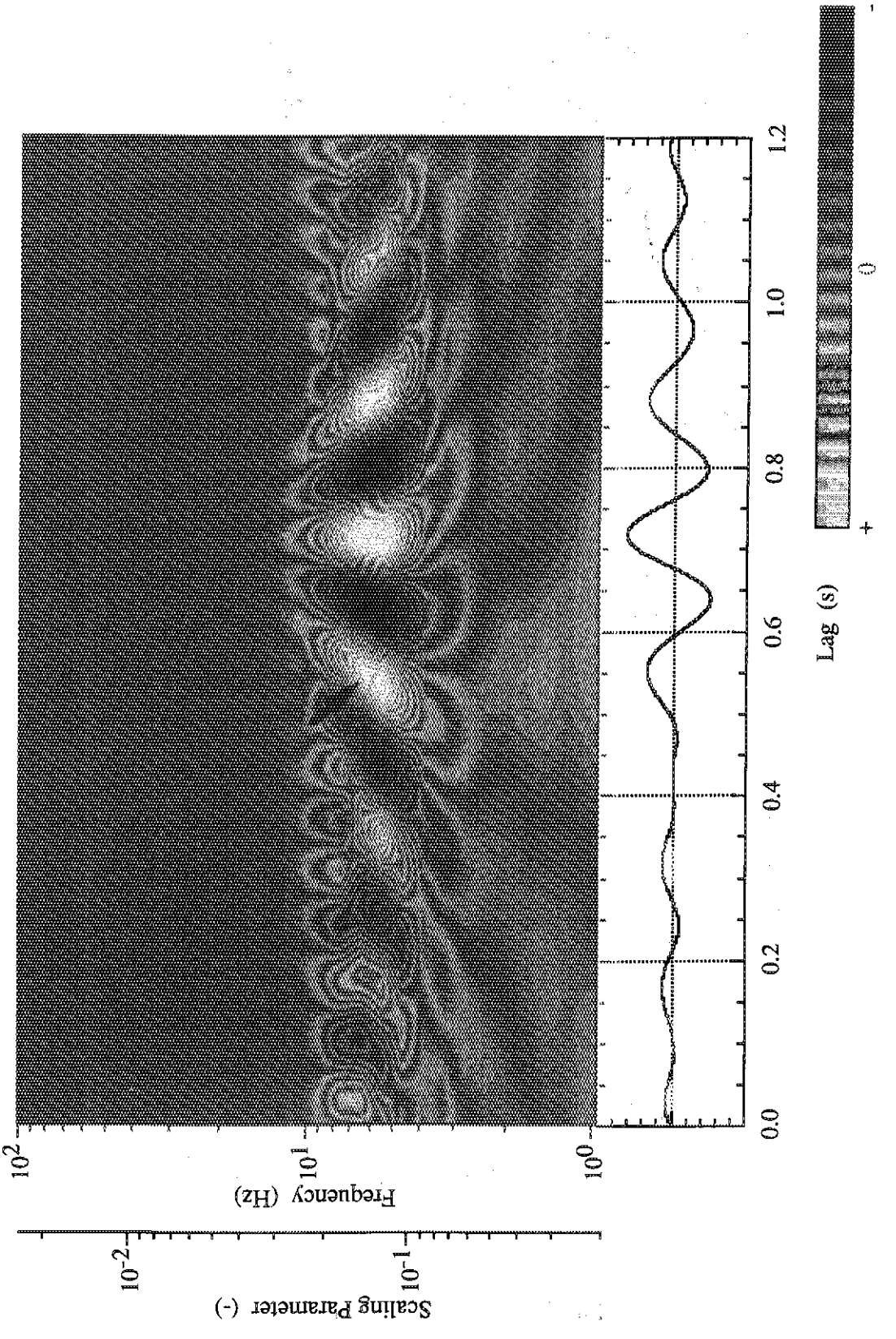


Fig. 6.10 Cross-Correlation of Wavelet Coefficients (Gabor Function)

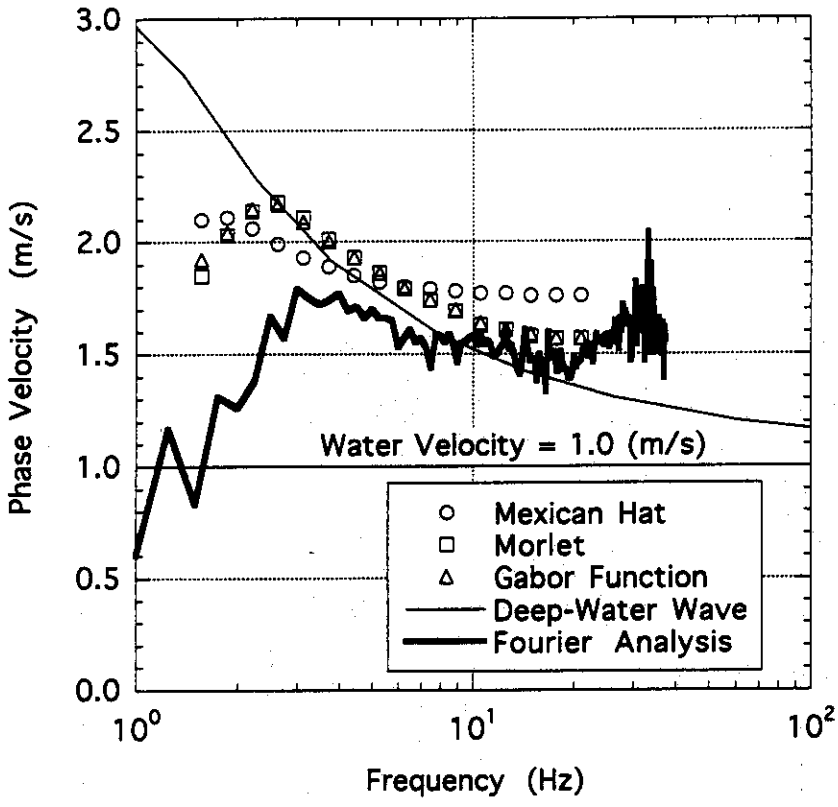


Fig. 6.11 Phase Velocity vs. Frequency

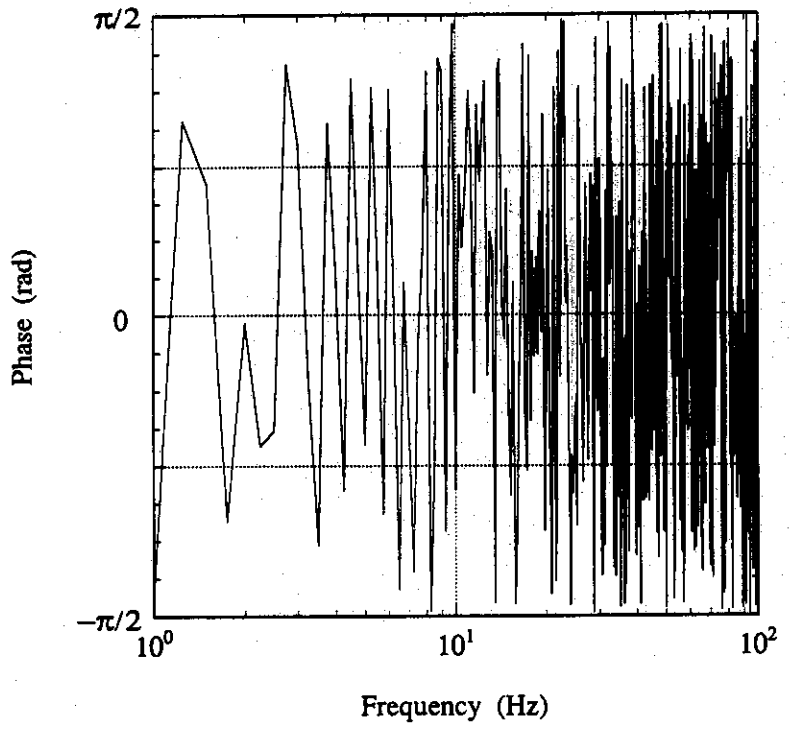


Fig. 6.12 Phase (l = 1 m)

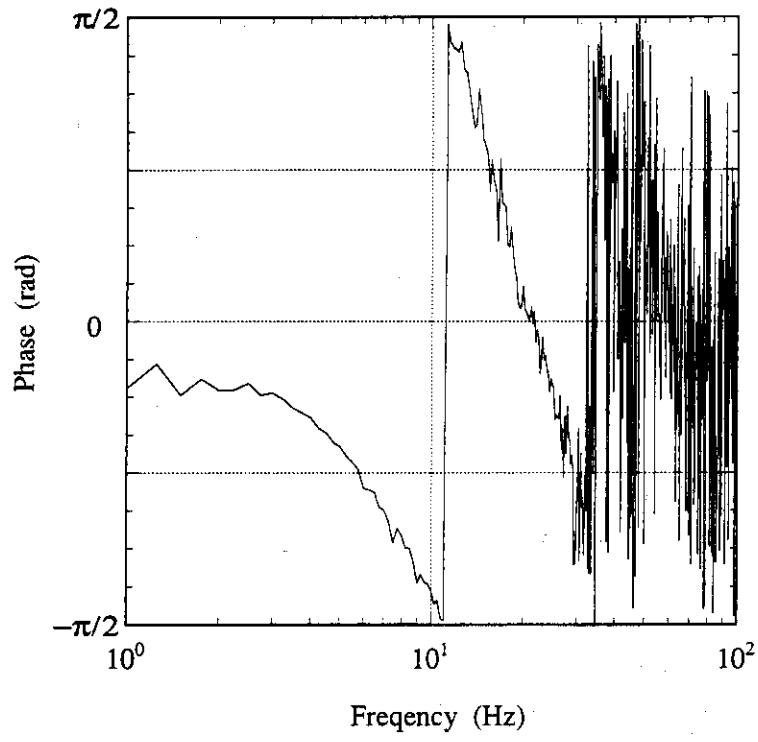


Fig. 6.13 Phase (l = 35 mm)

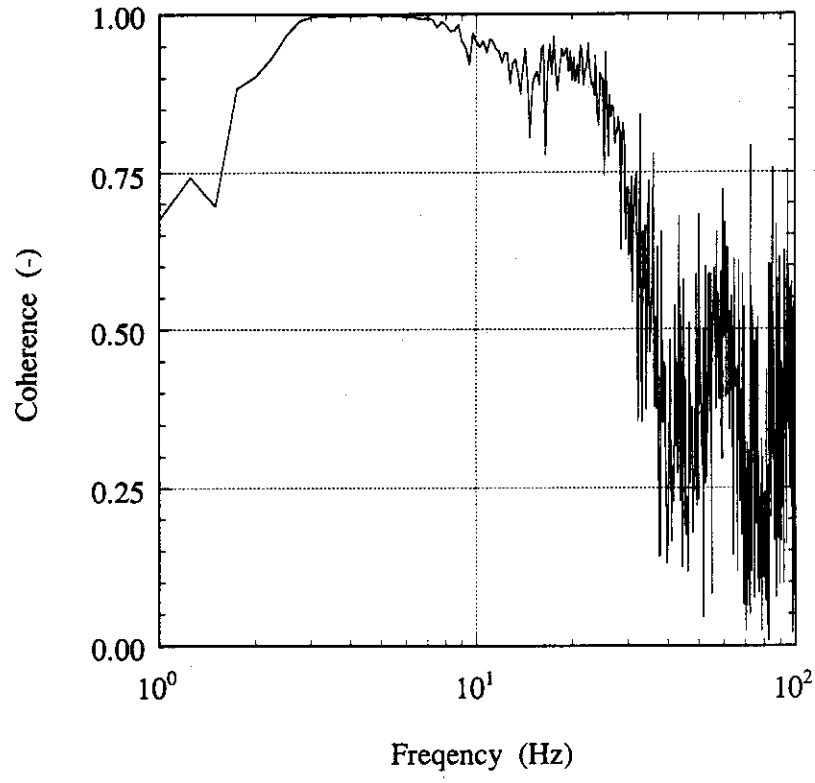


Fig. 6.14 Coherence (l = 35 mm)

付録 A　大型水平ダクト試験装置における水／空気流量の算出

大型水平ダクト試験装置に用いられているオリフィス流量計の出力電圧と流量との関係は次式にて与えられる。

$$W = 0.012522\,\alpha\,\varepsilon\,Fa\,d_{20}^2\sqrt{\gamma_f\,\Delta P} \qquad (A.1)$$

この式において各変数はそれぞれ以下の物理量を示すものであり、各オリフィス固有の変数に関してはその値を Table A.1 に示す。

W	Weight Flow Rate
ΔP	Differential Pressure
α	Flow Coefficient
ε	Expansion Factor
Fa	Thermal Experimental Factor
d_{20}	Orifice Diameter at 20 ℃
γ_f	Fluid Density

ここで差圧 ΔP は、

$$\Delta P = \frac{V - V_0}{V_{max} - V_0}\Delta P_{max} \qquad (A.2)$$

であり、各変数は、

ΔP_{max}	Differential Pressure (Maximum Value)
V	Output Voltage
V_0	Output Voltage ($\Delta P = 0$)
V_{max}	Output Voltage ($\Delta P = \Delta P_{max}$)

である。

Table A.1

Fluid Name		WATER	WATER	WATER	WATER
Full Scale	(m3/h)	500	150	50	25
Line		water pump 1	water pump 1	water pump 2	water pump 2
α	(-)	0.7202	0.6149	0.6598	0.6179
ϵ	(-)	1.0000	1.0000	1.0000	1.0000
Fa	(-)	1.0007	1.0007	1.0007	1.0007
d20	(mm)	157.1	93.13	51.91	37.93
Δ Pmax	(mmH2O)	5000.0	5000.0	5000.0	5000.0
Flow Range	(%)	17.2 - 100	10.6 - 100	30.2 - 100	23.6 - 100

Fluid Name		Air	Air	Air	Air
Full Scale	(m3/h)	600	250	1000	400
Line		compressor	compressor	blower	blower
α	(-)	0.6546	0.6109	0.6284	0.6040
ϵ	(-)	0.9936	0.9941	0.9858	0.9865
Fa	(-)	1.0009	1.0009	1.0004	1.0004
d20	(mm)	41.36	27.64	67.82	43.74
Δ Pmax	(mmH2O)	1000.0	1000.0	1000.0	1000.0
Flow Range	(%)	29.7 - 100	25.1 - 100	21.6 - 100	18.6 - 100

付録 B Daubechies Wavelet の算出プログラム

```
/*
        N8.c ( Calculation of Daubechies Wavelet (N=8) )

        Produced by M.Kondo

        Ver. 950927
*/

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

#define         nmax            100
#define         M               15

FILE    *fopen(), *fp;

main()
{
        double A0[M][M], A1[M][M], z[nmax][M],
               phi[M*nmax], psi[M*nmax], x[nmax], alpha[M+1];

        srand( ( unsigned int) clock() );

        make_matrix( A0, A1, alpha );
        make_z0( A0, z );
        calc_z( A0, A1, z, x );
        calc_phi_psi( z, x, phi, psi, alpha );

        output( phi, psi );
}



calc_phi_psi( z, x, phi, psi, alpha )
double z[nmax][M], x[nmax], phi[M*nmax], psi[M*nmax], alpha[M+1];
{
        int     n, k;
        double a, function_psi();

        for( n = 0, a = 0; n < M*nmax; n++, a += 1./nmax ) {
                if( n/100 == (float)n/100 ) printf( "%3d¥n", n );
                for( k = 0; k < M; k++ )
                        phi[n] += pow( -1.0, k )*alpha[M-k]*sqrt(2)
                                *function_psi( z, x, 2*a-k );
                psi[n] = function_psi( z, x, a );
        }
}



calc_z(  A0, A1, z, x )
double A0[M][M], A1[M][M], z[nmax][M], x[nmax];
{
        int     k, n;
        double a[M], r[M];
```

```
        printf( "*** calculating f & x ***¥n" );

        x[0] = 0;

        for( n = 1; n < nmax; n++ ) {
                for( k = 0; k < M; k++ ) a[k] = z[n-1][k];
                if( rand() < 32767/2 ) {
                        x[n] = x[n-1]/2;
                        calc_Ab( A0, a, r );
                }
                else {
                        x[n] = ( x[n-1] + 1 )/2;
                        calc_Ab( A1, a, r );
                }
                for( k = 0; k < M; k++ )
                        z[n][k] = r[k];
        }
}


make_z0( A0, z )
double A0[M][M], z[nmax][M];
{
        int     i, ix, iy, itr, itrmax = 50;
        double A[M][M], R[M][M], a[M], r[M], sum = 0;


        printf( "*** calculating A*A*...A ***¥n" );

        copy_A_B( A0, A );

        for( itr = 0; itr < itrmax; itr++ ) {
                calc_AB( A, A0, R );
                copy_A_B( R, A );
        }


        printf( "*** calculating Ab ***¥n" );

        for( iy = 0; iy < M; iy++ ) a[iy] = 1.0;

        calc_Ab( A, a, r );

        for( iy = 0; iy < M; iy++ ) sum += r[iy];
        for( iy = 0; iy < M; iy++ ) z[0][iy] = r[iy] / sum;
}


make_matrix( A0, A1, alpha )
double A0[M][M], A1[M][M], alpha[M+1];
{
        int     ix, iy;

        alpha[ 0] = 0.054415842243;  alpha[ 1] = 0.312871590914;
        alpha[ 2] = 0.675630736297;  alpha[ 3] = 0.585354683654;
        alpha[ 4] =-0.015829105256;  alpha[ 5] =-0.284015542962;
```

```
alpha[ 6] = 0.000472484574;  alpha[ 7] = 0.128747426620;
alpha[ 8] =-0.017369301002;  alpha[ 9] =-0.044088253931;
alpha[10] = 0.013981027917;  alpha[11] = 0.008746094047;
alpha[12] =-0.004870352993;  alpha[13] =-0.000391740373;
alpha[14] = 0.000675449406;  alpha[15] =-0.000117476784;

clear_A( A0 );
clear_A( A1 );

A0[ 0][ 0] = alpha[0];

A0[ 0][ 1] = alpha[2];       A0[ 1][ 1] = alpha[1];       A0[ 2][ 1] = alpha[0];

A0[ 0][ 2] = alpha[4];       A0[ 1][ 2] = alpha[3];       A0[ 2][ 2] = alpha[2];
A0[ 3][ 2] = alpha[1];       A0[ 4][ 2] = alpha[0];

A0[ 0][ 3] = alpha[6];       A0[ 1][ 3] = alpha[5];       A0[ 2][ 3] = alpha[4];
A0[ 3][ 3] = alpha[3];       A0[ 4][ 3] = alpha[2];       A0[ 5][ 3] = alpha[1];
A0[ 6][ 3] = alpha[0];

A0[ 0][ 4] = alpha[8];       A0[ 1][ 4] = alpha[7];       A0[ 2][ 4] = alpha[6];
A0[ 3][ 4] = alpha[5];       A0[ 4][ 4] = alpha[4];       A0[ 5][ 4] = alpha[3];
A0[ 6][ 4] = alpha[2];       A0[ 7][ 4] = alpha[1];       A0[ 8][ 4] = alpha[0];

A0[ 0][ 5] = alpha[10];      A0[ 1][ 5] = alpha[9];       A0[ 2][ 5] = alpha[8];
A0[ 3][ 5] = alpha[7];       A0[ 4][ 5] = alpha[6];       A0[ 5][ 5] = alpha[5];
A0[ 6][ 5] = alpha[4];       A0[ 7][ 5] = alpha[3];       A0[ 8][ 5] = alpha[2];
A0[ 9][ 5] = alpha[1];       A0[10][ 5] = alpha[0];

A0[ 0][ 6] = alpha[12];      A0[ 1][ 6] = alpha[11];      A0[ 2][ 6] = alpha[10];
A0[ 3][ 6] = alpha[9];       A0[ 4][ 6] = alpha[8];       A0[ 5][ 6] = alpha[7];
A0[ 6][ 6] = alpha[6];       A0[ 7][ 6] = alpha[5];       A0[ 8][ 6] = alpha[4];
A0[ 9][ 6] = alpha[3];       A0[10][ 6] = alpha[2];       A0[11][ 6] = alpha[1];
A0[12][ 6] = alpha[0];

A0[ 0][ 7] = alpha[14];      A0[ 1][ 7] = alpha[13];      A0[ 2][ 7] = alpha[12];
A0[ 3][ 7] = alpha[11];      A0[ 4][ 7] = alpha[10];      A0[ 5][ 7] = alpha[9];
A0[ 6][ 7] = alpha[8];       A0[ 7][ 7] = alpha[7];       A0[ 8][ 7] = alpha[6];
A0[ 9][ 7] = alpha[5];       A0[10][ 7] = alpha[4];       A0[11][ 7] = alpha[3];
A0[12][ 7] = alpha[2];       A0[13][ 7] = alpha[1];       A0[14][ 7] = alpha[0];

A0[ 1][ 8] = alpha[15];      A0[ 2][ 8] = alpha[14];      A0[ 3][ 8] = alpha[13];
A0[ 4][ 8] = alpha[12];      A0[ 5][ 8] = alpha[11];      A0[ 6][ 8] = alpha[10];
A0[ 7][ 8] = alpha[9];       A0[ 8][ 8] = alpha[8];       A0[ 9][ 8] = alpha[7];
A0[10][ 8] = alpha[6];       A0[11][ 8] = alpha[5];       A0[12][ 8] = alpha[4];
A0[13][ 8] = alpha[3];       A0[14][ 8] = alpha[2];

A0[ 3][ 9] = alpha[15];      A0[ 4][ 9] = alpha[14];      A0[ 5][ 9] = alpha[13];
A0[ 6][ 9] = alpha[12];      A0[ 7][ 9] = alpha[11];      A0[ 8][ 9] = alpha[10];
A0[ 9][ 9] = alpha[9];       A0[10][ 9] = alpha[8];       A0[11][ 9] = alpha[7];
A0[12][ 9] = alpha[6];       A0[13][ 9] = alpha[5];       A0[14][ 9] = alpha[4];

A0[ 5][10] = alpha[15];      A0[ 6][10] = alpha[14];      A0[ 7][10] = alpha[13];
A0[ 8][10] = alpha[12];      A0[ 9][10] = alpha[11];      A0[10][10] = alpha[10];
A0[11][10] = alpha[9];       A0[12][10] = alpha[8];       A0[13][10] = alpha[7];
```

```
A0[14][10] = alpha[6];

A0[ 7][11] = alpha[15];     A0[ 8][11] = alpha[14];     A0[ 9][11] = alpha[13];
A0[10][11] = alpha[12];     A0[11][11] = alpha[11];     A0[12][11] = alpha[10];
A0[13][11] = alpha[9];      A0[14][11] = alpha[8];

A0[ 9][12] = alpha[15];     A0[10][12] = alpha[14];     A0[11][12] = alpha[13];
A0[12][12] = alpha[12];     A0[13][12] = alpha[11];     A0[14][12] = alpha[10];

A0[11][13] = alpha[15];     A0[12][13] = alpha[14];     A0[13][13] = alpha[13];
A0[14][13] = alpha[12];

A0[13][14] = alpha[15];     A0[14][14] = alpha[14];

A1[ 0][ 0] = alpha[1];      A1[ 1][ 0] = alpha[0];

A1[ 0][ 1] = alpha[3];      A1[ 1][ 1] = alpha[2];      A1[ 2][ 1] = alpha[1];
A1[ 3][ 1] = alpha[0];

A1[ 0][ 2] = alpha[5];      A1[ 1][ 2] = alpha[4];      A1[ 2][ 2] = alpha[3];
A1[ 3][ 2] = alpha[2];      A1[ 4][ 2] = alpha[1];      A1[ 5][ 2] = alpha[0];

A1[ 0][ 3] = alpha[7];      A1[ 1][ 3] = alpha[6];      A1[ 2][ 3] = alpha[5];
A1[ 3][ 3] = alpha[4];      A1[ 4][ 3] = alpha[3];      A1[ 5][ 3] = alpha[2];
A1[ 6][ 3] = alpha[1];      A1[ 7][ 3] = alpha[0];

A1[ 0][ 4] = alpha[9];      A1[ 1][ 4] = alpha[8];      A1[ 2][ 4] = alpha[7];
A1[ 3][ 4] = alpha[6];      A1[ 4][ 4] = alpha[5];      A1[ 5][ 4] = alpha[4];
A1[ 6][ 4] = alpha[3];      A1[ 7][ 4] = alpha[2];      A1[ 8][ 4] = alpha[1];
A1[ 9][ 4] = alpha[0];

A1[ 0][ 5] = alpha[11];     A1[ 1][ 5] = alpha[10];     A1[ 2][ 5] = alpha[9];
A1[ 3][ 5] = alpha[8];      A1[ 4][ 5] = alpha[7];      A1[ 5][ 5] = alpha[6];
A1[ 6][ 5] = alpha[5];      A1[ 7][ 5] = alpha[4];      A1[ 8][ 5] = alpha[3];
A1[ 9][ 5] = alpha[2];      A1[10][ 5] = alpha[1];      A1[11][ 5] = alpha[0];

A1[ 0][ 6] = alpha[13];     A1[ 1][ 6] = alpha[12];     A1[ 2][ 6] = alpha[11];
A1[ 3][ 6] = alpha[10];     A1[ 4][ 6] = alpha[9];      A1[ 5][ 6] = alpha[8];
A1[ 6][ 6] = alpha[7];      A1[ 7][ 6] = alpha[6];      A1[ 8][ 6] = alpha[5];
A1[ 9][ 6] = alpha[4];      A1[10][ 6] = alpha[3];      A1[11][ 6] = alpha[2];
A1[12][ 6] = alpha[1];      A1[13][ 6] = alpha[0];

A1[ 0][ 7] = alpha[15];     A1[ 1][ 7] = alpha[14];     A1[ 2][ 7] = alpha[13];
A1[ 3][ 7] = alpha[12];     A1[ 4][ 7] = alpha[11];     A1[ 5][ 7] = alpha[10];
A1[ 6][ 7] = alpha[9];      A1[ 7][ 7] = alpha[8];      A1[ 8][ 7] = alpha[7];
A1[ 9][ 7] = alpha[6];      A1[10][ 7] = alpha[5];      A1[11][ 7] = alpha[4];
A1[12][ 7] = alpha[3];      A1[13][ 7] = alpha[2];      A1[14][ 7] = alpha[1];

A1[ 2][ 8] = alpha[15];     A1[ 3][ 8] = alpha[14];     A1[ 4][ 8] = alpha[13];
A1[ 5][ 8] = alpha[12];     A1[ 6][ 8] = alpha[11];     A1[ 7][ 8] = alpha[10];
A1[ 8][ 8] = alpha[9];      A1[ 9][ 8] = alpha[8];      A1[10][ 8] = alpha[7];
A1[11][ 8] = alpha[6];      A1[12][ 8] = alpha[5];      A1[13][ 8] = alpha[4];
A1[14][ 8] = alpha[3];

A1[ 4][ 9] = alpha[15];     A1[ 5][ 9] = alpha[14];     A1[ 6][ 9] = alpha[13];
```

```
        A1[ 7][ 9] = alpha[12];     A1[ 8][ 9] = alpha[11];     A1[ 9][ 9] = alpha[10];
        A1[10][ 9] = alpha[9];      A1[11][ 9] = alpha[8];      A1[12][ 9] = alpha[7];
        A1[13][ 9] = alpha[6];      A1[14][ 9] = alpha[5];

        A1[ 6][10] = alpha[15];     A1[ 7][10] = alpha[14];     A1[ 8][10] = alpha[13];
        A1[ 9][10] = alpha[12];     A1[10][10] = alpha[11];     A1[11][10] = alpha[10];
        A1[12][10] = alpha[9];      A1[13][10] = alpha[8];      A1[14][10] = alpha[7];

        A1[ 8][11] = alpha[15];     A1[ 9][11] = alpha[14];     A1[10][11] = alpha[13];
        A1[11][11] = alpha[12];     A1[12][11] = alpha[11];     A1[13][11] = alpha[10];
        A1[14][11] = alpha[9];

        A1[10][12] = alpha[15];     A1[11][12] = alpha[14];     A1[12][12] = alpha[13];
        A1[13][12] = alpha[12];     A1[14][12] = alpha[11];

        A1[12][13] = alpha[15];     A1[13][13] = alpha[14];     A1[14][13] = alpha[13];

        A1[14][14] = alpha[15];

        for( iy = 0; iy < M; iy++ ) {
                for( ix = 0; ix < M; ix++ ) {
                        A0[ix][iy] *= sqrt(2);
                        A1[ix][iy] *= sqrt(2);
                }
        }
}


calc_AB( A, B, R )
double A[M][M], B[M][M], R[M][M];
{

/*              calculate  A*B                  */

        int     i, ix, iy;

        for( iy = 0; iy < M; iy++ ) {
                for( ix = 0; ix < M; ix++ ) {
                        R[ix][iy] = 0;
                        for( i = 0; i < M; i++ )
                                R[ix][iy] += ( A[i][iy] * B[ix][i] );
                }
        }
}


calc_Ab( A, b, r )
double A[M][M], b[M], r[M];
{

/*              calculate  A*b                  */

        int     i, iy;

        for( iy = 0; iy < M; iy++ ) {
                r[iy] = 0;
                for( i = 0; i < M; i++ )
                        r[iy] += ( A[i][iy] * b[i] );
```

```
        }
}


clear_A( A )
double A[M][M];
{

/*              A <= 0                  */

        int     ix, iy;

        for( iy = 0; iy < M; iy++ )
                for( ix = 0; ix < M; ix++ )
                        A[ix][iy] = 0;
}


copy_A_B( A, B )
double A[M][M], B[M][M];
{

/*              copy from A to B                */

        int     ix, iy;

        for( iy = 0; iy < M; iy++ )
                for( ix = 0; ix < M; ix++ )
                        B[ix][iy] = A[ix][iy];
}




output( phi, psi )
double phi[M*nmax], psi[M*nmax];
{
        int     n;
        double a;

        fp = fopen( "N8.dat", "w" );
                for( n = 0, a = 0; n < M*nmax; n++, a+= 1./nmax )
                        fprintf( fp, "%5.3f, %5.3f, %5.3f¥n", a, phi[n], psi[n] );
        fclose( fp );
}


double function_psi( z, x, a )
double z[nmax][M], x[nmax], a;
{
        int     k, n, f = 0;
        double xl, xh, pl, ph, ans;

        xl = -1000.0;
        xh =  1000.0;
        if(( a < 0 ) || ( a > M )) ans = 0;
```

```
        else {
                for( k = 0; k < M; k++ ) {
                        for( n = 0; n < nmax; n++ ) {
                                if( x[n] + k == a ) {
                                        ans = z[n][k];
                                        f = 1;
                                }
                                if(( x[n] + k < a ) && ( x[n] + k > xl )) {
                                        xl = x[n] + k;
                                        pl = z[n][k];
                                }
                                if(( x[n] + k > a ) && ( x[n] + k < xh )) {
                                        xh = x[n] + k;
                                        ph = z[n][k];
                                }
                        }
                }
                if( f == 0 ) ans = ( ph - pl )/( xh - xl )*( a - xl ) + pl;
        }

        return ans;
}
```

付録 C　　単純な関数のウェーブレット変換

C.1 被変換関数が一定値の場合

被変換関数が $f(t)=A$ の場合（A は任意の定数）、そのウェーブレット変換は (4.2.4) 式より、

$$T_\psi(a,b)=\frac{A}{\sqrt{a}}\int_{-\infty}^{\infty}\overline{\psi\left(\frac{t-b}{a}\right)}dt=\frac{Aa}{\sqrt{a}}\int_{-\infty}^{\infty}\overline{\psi(t)}dt=0 \tag{C.1}$$

となり、スケール及びシフト量の値に関わらずゼロとなる。

C.2 被変換関数が一次関数の場合

被変換関数が $f(t)=At+B$ の場合、そのウェーブレット変換は、

$$T_\psi(a,b)=\frac{A}{\sqrt{a}}\int_{-\infty}^{\infty}\overline{\psi\left(\frac{t-b}{a}\right)}tdt=\frac{Aa^2}{\sqrt{a}}\int_{-\infty}^{\infty}\overline{\psi(t)}tdt \tag{C.2}$$

となる。この場合、ウェーブレット関数が偶関数、すなわち、Mexican Hat 型ウェーブレットあるいは Morlet のウェーブレット／Gabor 関数の実数部といった偶関数であるならば変換結果はゼロとなる。一方、Morlet のウェーブレット及び Gabor 関数の虚数部といった奇関数の場合は、

$$T_\psi(a)=\frac{2Aa^2}{\sqrt{a}}\int_{0}^{\infty}\overline{\psi(t)}tdt \tag{C.3}$$

とシフト量 b によらない一定値となる。

C.3 被変換関数が三角関数の場合

被変換関数が三角関数（$f(t)=A\sin(\omega t)$）の場合、そのウェーブレット変換は、

$$\begin{aligned}T_\psi(a,b)&=\frac{A}{\sqrt{a}}\int_{-\infty}^{\infty}\overline{\psi\left(\frac{t-b}{a}\right)}\sin(\omega t)dt\\&=\frac{Aa}{\sqrt{a}}\left[\sin(\omega b)\int_{-\infty}^{\infty}\overline{\psi(t)}\cos(\omega at)dt+\cos(\omega b)\int_{-\infty}^{\infty}\overline{\psi(t)}\sin(\omega at)dt\right]\end{aligned} \tag{C.4}$$

となる。

この式をウェーブレット関数が偶関数と奇関数の場合にわけて整理すると、偶関数の場合は、

$$T_{\psi}(a,b)=\frac{2A\sin(\omega b)}{\sqrt{a}}\int_0^{\infty}\overline{\psi(t)}\cos(\omega a t)dt \tag{C.5}$$

となり、スケール a を一定値とするとシフト量が

$$b=\frac{1}{\omega}\left(\frac{\pi}{2}+m\pi\right) \qquad (m \text{ は整数}) \tag{C.6}$$

の場合にピーク値をとる。これはすなわち、被変換関数である三角関数（$f(t)=A\sin(\omega t)$）がピーク値をとる時刻（シフト量）に、そのウェーブレット変換係数もピーク値をとることを意味する。一方、シフト量 b を一定とすると、これはウェーブレット関数をフーリエ変換することに他ならず、5.2 節にて示したスケールの値において極大値をとる。

同様にウェーブレット関数が奇関数の場合、そのウェーブレット変換は

$$T_{\psi}(a,b)=\frac{2A\cos(\omega b)}{\sqrt{a}}\int_0^{\infty}\overline{\psi(t)}\sin(\omega a t)dt \tag{C.7}$$

となり、

$$b=\frac{m\pi}{\omega} \qquad (m \text{ は整数}) \tag{C.8}$$

の場合にピーク値をとる。これは被変換関数である三角関数が単調に増加ないしは減少する時刻に、ウェーブレット変換係数がピーク値をとることを示す。

例として、三角関数 $f(t)=\sin(120t)$ を Mexican Hat 型ウェーブレット、Morlet のウェーブレット及び Gabor 関数にてウェーブレット変換した結果を Fig. C.1 ～ C.3 に示す。図ではウェーブレット変換係数の縦軸にスケールを対数でとっており、これは周波数に換算すると 0.9 ～ 100 Hz の範囲に相当する。同様に、横軸にはシフト量をとっており、その範囲は 0 ～ 1.5 秒（1500 点）である。

C.4 被変換関数が三角波／矩形波の場合

三角波、矩形波は、エッジとそれ以外の直線部分とに分けられる。時間軸上でウェーブレット関数に比べて直線部分が十分に長い場合、直線部分に関しては先の C.1 節及び C.2 節における取り扱いと同様に取り扱うことが可能である。一方、エッジに関しては、三角波のように山／谷の形状を持つエッジの場合は偶関数のウェーブレットが、矩形波のように急増／急減するエッジの場合は奇関数のウェーブレットがそれぞれ大きな値をとる。また、三角波、矩形波に関わらず、これらの周期に比べてウェーブレット関数が時間軸上で十分に長い場合は、直線を変換することに相当するため、変換係数はゼロに近い値となる。例として三角波のウェーブレット変換係数を Fig. C.4 ～ C.6 に、矩形波のウェーブレット変換係数を Fig. C.7 ～ C.9 に示す。

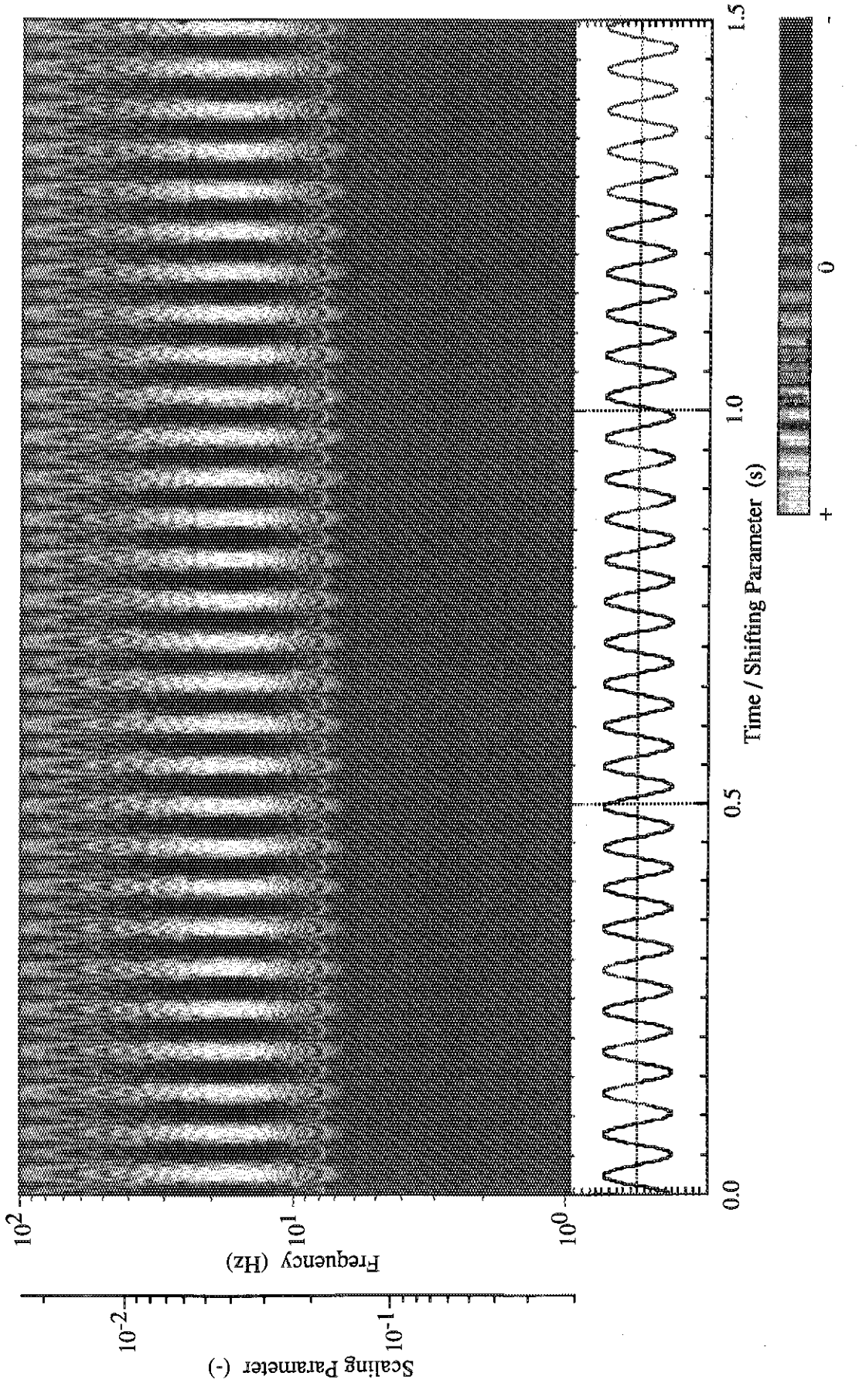


Fig. C.1 三角関数のウェーブレット変換係数（Mexican Hat 型ウェーブレット）

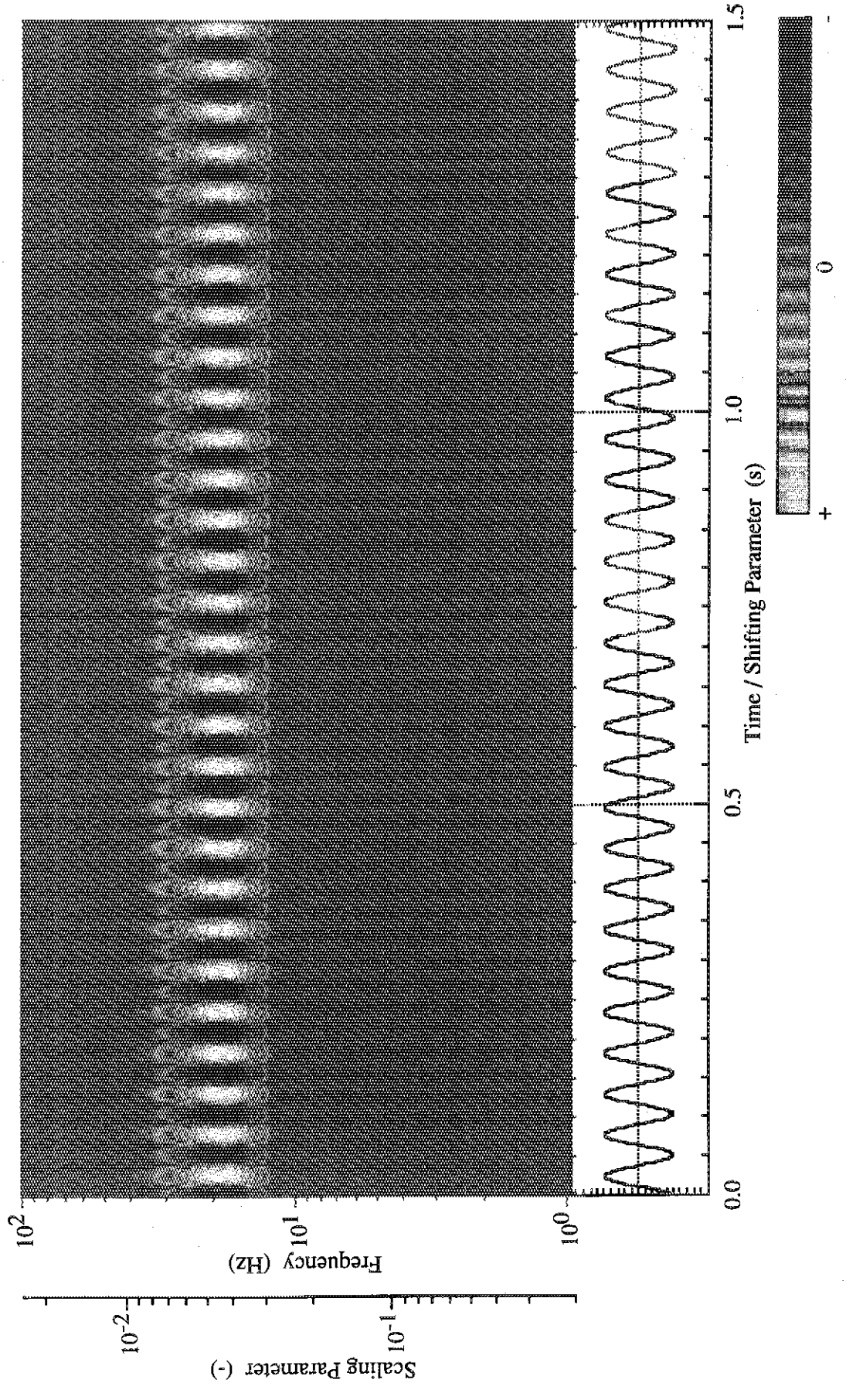


Fig. C.2 三角関数のウェーブレット変換係数（Morlet のウェーブレット）

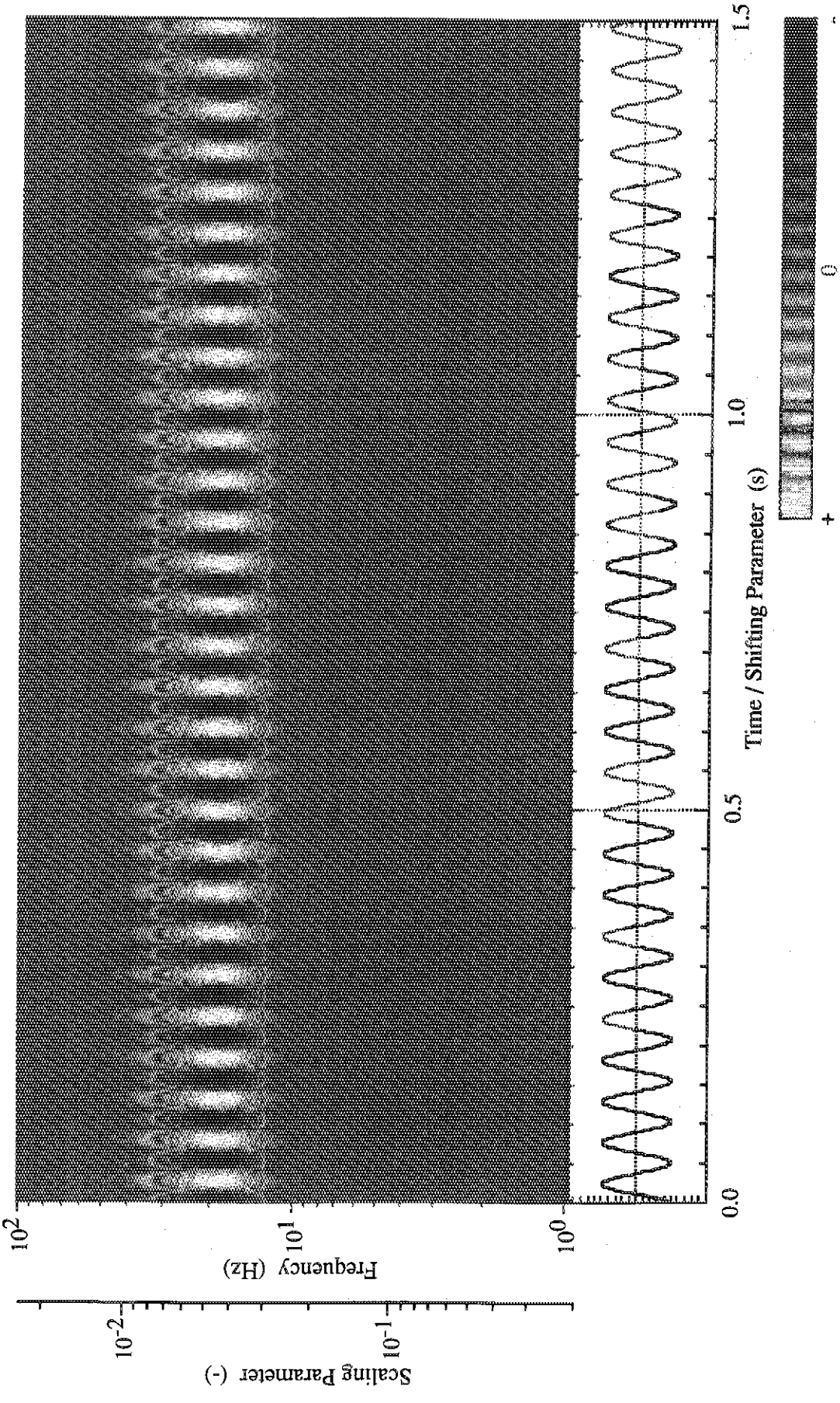


Fig. C.3 三角関数のウェーブレット変換係数（Gabor 関数）

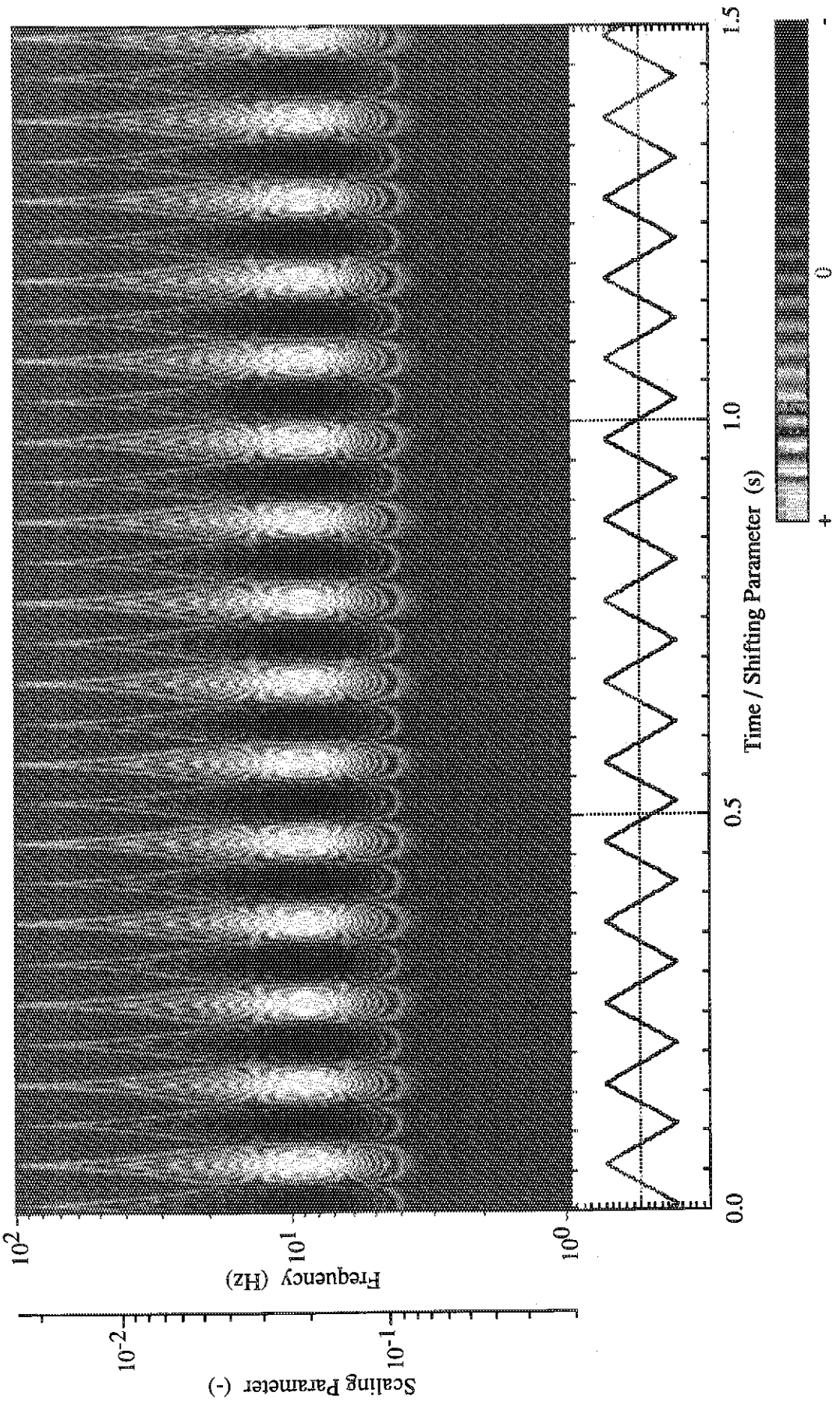


Fig. C.4 三角波のウェーブレット変換係数（Mexican Hat 型ウェーブレット）

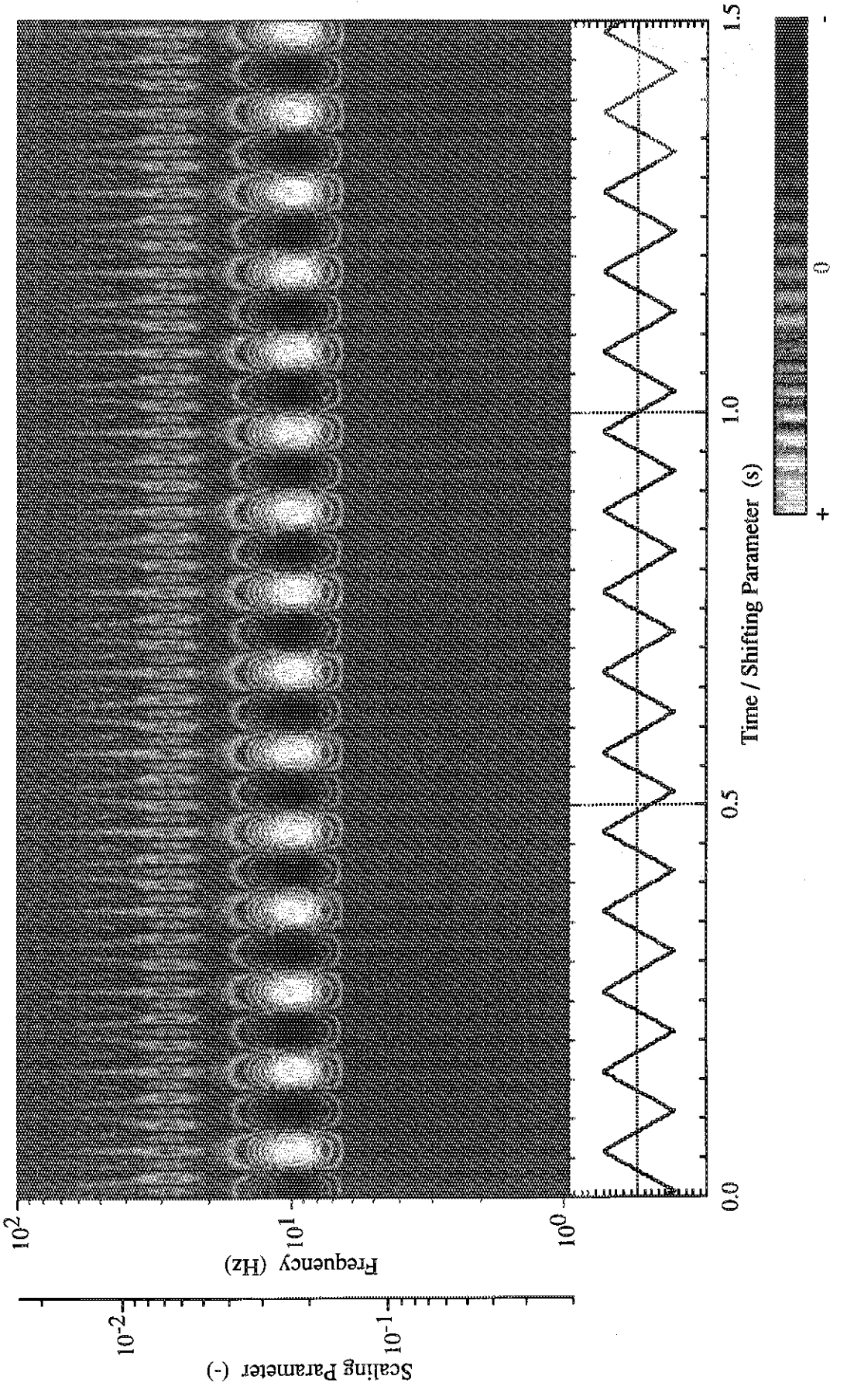


Fig. C.5 三角波のウェーブレット変換係数（Morlet のウェーブレット）

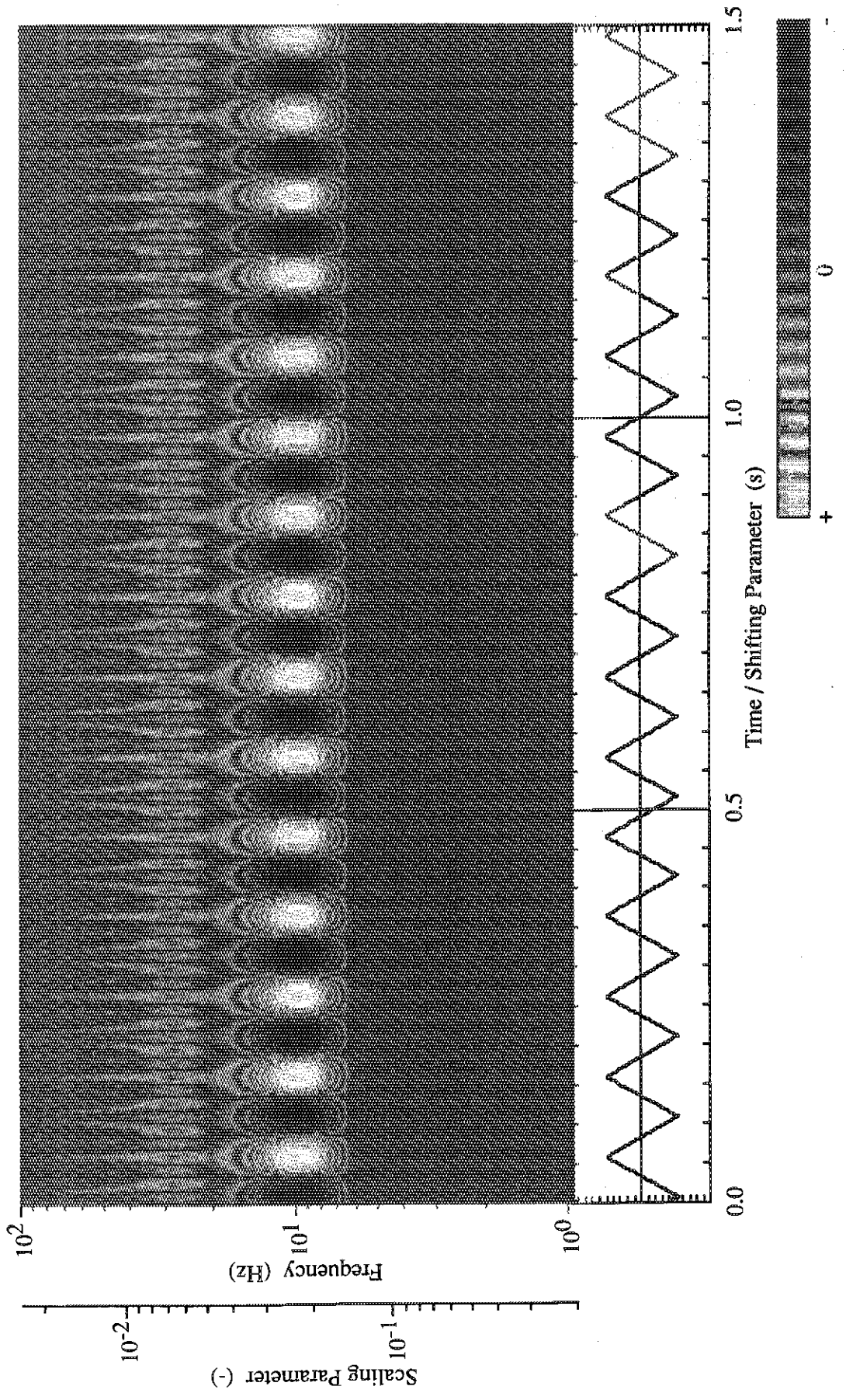


Fig. C.6 三角波のウェーブレット変換係数（Gabor 関数）

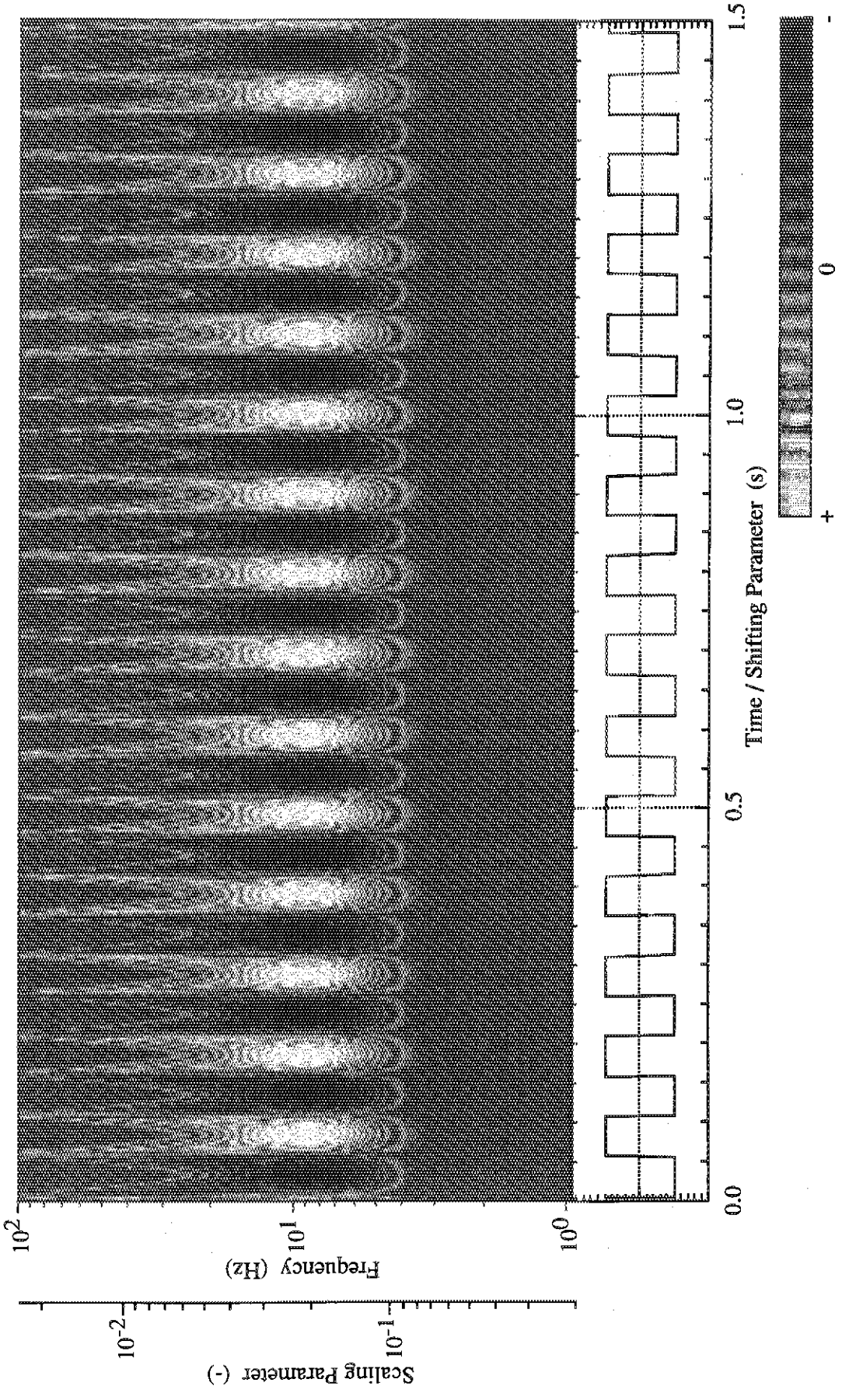


Fig. C.7 矩形波のウェーブレット変換係数（Mexican Hat 型ウェーブレット）

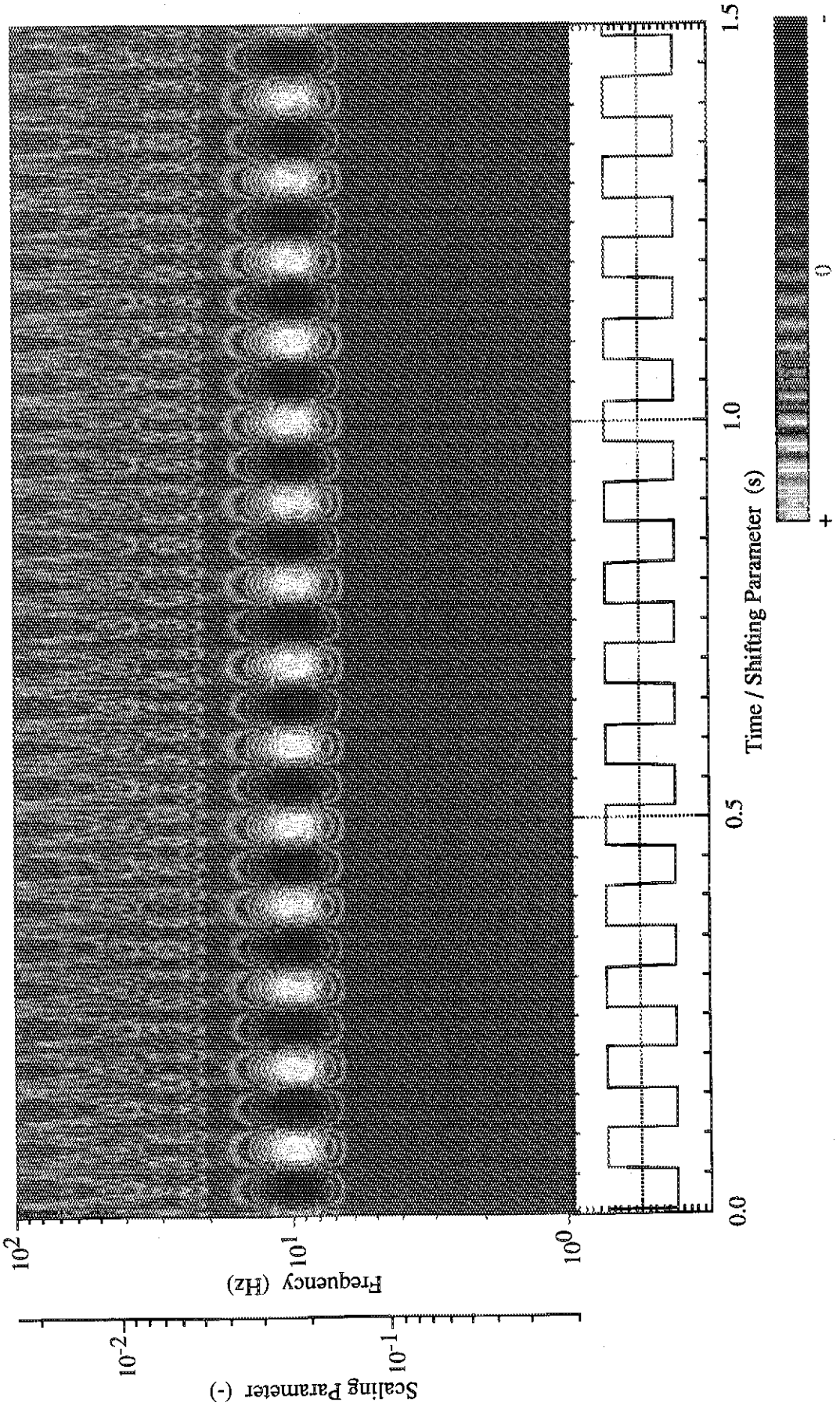


Fig. C.8 矩形波のウェーブレット変換係数 (Morlet のウェーブレット)

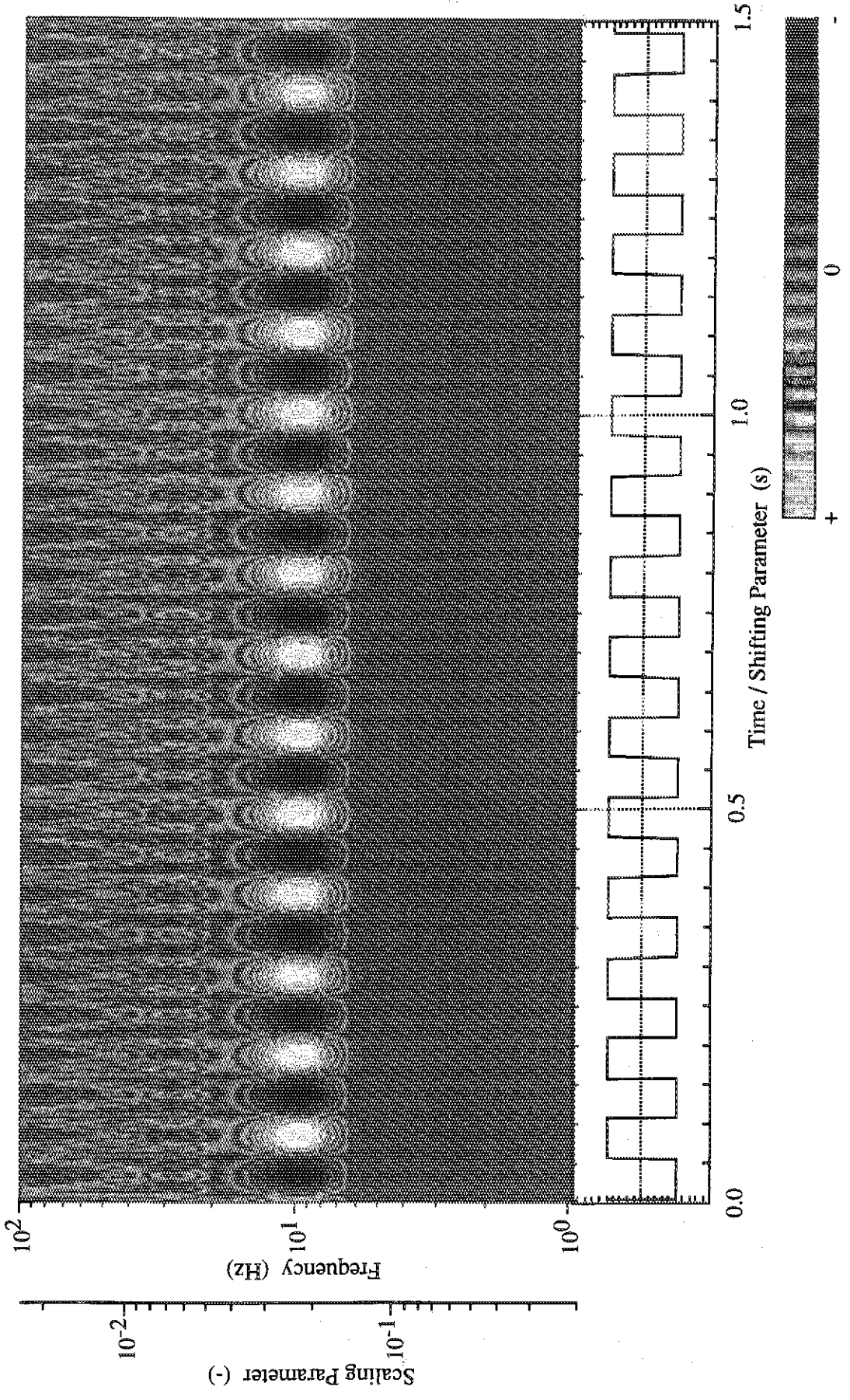


Fig. C.9 矩形波のウェーブレット変換係数（Gabor 関数）

付録 D Fourier 変換プログラム

```
/*
        Fourier2.c ( Fourier Transfer for EDX1000AS data )

        for EDX-1000AS

        Produced by M.Kondo

        Ver. 960529
*/

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

#define       FTMAXSIZE     40000
#define       PI            3.1416

FILE    *fopen(), *fp;

main( argc, argv )
char    *argv[];
int     argc;
{
        char    output_filename[20];
        int     ch1, ch2, n, nmax, nf, nf_all, input_int();
        long    nf0, nf0_step, header[20], input_long();
        double  sampling, freq[FTMAXSIZE/2],
                coh[FTMAXSIZE/2], phase[FTMAXSIZE/2],
                ps1[FTMAXSIZE/2], ps2[FTMAXSIZE/2],
                csr[FTMAXSIZE/2], csi[FTMAXSIZE/2],
                fr1[FTMAXSIZE], fi1[FTMAXSIZE],
                fr2[FTMAXSIZE], fi2[FTMAXSIZE],
                ch_info[32][6];

        EDX1000AS_read_header( argv[1], header, ch_info );
        EDX1000AS_printf_header( header, ch_info );

        ch1 = header[5] = atol( argv[2] );  /*      ch1             */
        ch2 = header[6] = atol( argv[3] );  /*      ch2             */
        nf_all = header[7] = atol( argv[4] );       /*      data size       */
        nf0     = header[8] = atol( argv[5] );      /*      start point     */
        nf      = atoi( argv[6] );    /*      processing data size  */
        sampling = 1./header[3];

        nmax = 0;
        while( nmax * nf < nf_all ) nmax++;
        nf0_step = nf - ( (long)nmax * nf - nf_all )/( nmax - 1 );

        zero( ps1, nf/2 );
        zero( ps2, nf/2 );
        zero( csr, nf/2 );
        zero( csi, nf/2 );

        if( ch2 > 0 ) {
        for( n = 0; n < nmax; n++ ) {
                printf( "%7ld to %7ld¥n", nf0, nf0 + nf );
                EDX1000AS_read_data( fr1, ch1, nf, nf0, argv[1], header, ch_info );
                EDX1000AS_read_data( fr2, ch2, nf, nf0, argv[1], header, ch_info );
                pre_calculation( fr1, nf, header[9], header[10] );
                pre_calculation( fr2, nf, header[11], header[12] );
                zero( fi1, nf );
                zero( fi2, nf );
```

```
            Fourier( fr1, fi1, nf );
            Fourier( fr2, fi2, nf );
            PSD( sampling, fr1, fi1, ps1, nf );
            PSD( sampling, fr2, fi2, ps2, nf );
            CSD( sampling, fr1, fi1, fr2, fi2, csr, csi, nf );
            nf0 += nf0_step;
        }
        frequency( sampling, freq, nf/2 );
        post_calculation( nmax, ps1, nf/2 );
        post_calculation( nmax, ps2, nf/2 );
        post_calculation( nmax, csr, nf/2 );
        post_calculation( nmax, csi, nf/2 );

        COH( ps1, ps2, csr, csi, coh, phase, nf/2 );

        make_filename( argv[1], output_filename, "F", ch1, ch2 );
        EDX1000AS_fprintf_header( output_filename, header, ch_info );
        CSV_output( freq, ps1, ps2, coh, phase, csr, csi, nf/2, output_filename );
        }
        else {
        for( n = 0; n < nmax; n++ ) {
            printf( "%7ld to %7ld¥n", nf0, nf0 + nf );
            EDX1000AS_read_data( fr1, ch1, nf, nf0, argv[1], header, ch_info );
            pre_calculation( fr1, nf, header[9], header[10] );
            zero( fi1, nf );
            Fourier( fr1, fi1, nf );
            PSD( sampling, fr1, fi1, ps1, nf );
            nf0 += nf0_step;
        }
        frequency( sampling, freq, nf/2 );
        post_calculation( nmax, ps1, nf/2 );
        }

        make_filename( argv[1], output_filename, "F", ch1, ch2 );
        EDX1000AS_fprintf_header( output_filename, header, ch_info );
        CSV_output2( freq, ps1, nf/2, output_filename );
}


check_sinwave( x, nx )
int     nx;
double *x;
{
        int     n;

        for( n = 0; n < nx; n++ )
            *x++ = 5.*sin( 2*PI*n/143 );
}



check_sinwave2( x, nx )
int     nx;
double *x;
{
        int     n;

        for( n = 0; n < nx; n++ )
            *x++ = 2.*sin( 2*PI*n/20 ) + 4.* sin( 2*PI*n/25 );
}



wlcnv( x, nx, ch )
```

```
int             ch, nx;
double *x;
{
        double a, b;

/*  These water level convert coefficients are for g129.  */
        switch( ch ) {
                case  1 : a = -132.8; b = 700.3; break;
                case  2 : a = -135.3; b = 700.6; break;
                case  3 : a = -131.6; b = 700.3; break;
                case  4 : a = -130.3; b = 700.4; break;
                case  5 : a = -129.0; b = 700.4; break;
                case  6 : a = -130.0; b = 700.3; break;
                case  7 : a = -145.4; b = 700.4; break;
                case  8 : a = -131.2; b = 700.4; break;
                case  9 : a = -133.7; b = 700.4; break;
                case 10 : a = -133.4; b = 700.4; break;
                case 11 : a = -139.1; b = 700.3; break;
                case 12 : a = -240.1; b = 700.4; break;
        }
        while( nx-- ) {
                *x = *x * a + b;
                *x++;
        }
}



pre_calculation( x, nx, average, stdev )
int     nx;
long    average, stdev;
double *x;
{
        int             n;
        double sum, sum2;

/* average calculation */

        sum = sum2 = 0;

        for( n = 0; n < nx; n++ ) {
                sum  += *x;
                sum2 += (x[n] * x[n]);
        }
        average = sum / nx;
        stdev   = sqrt( sum2/nx - average*average );

        for( n = 0; n < nx; n++ ) x[n] -= average;

        average *= 1000000;
        stdev   *= 1000000;


/* window */

        for( n = 0; n < nx/10; n++ ){
                x[n]      *= sin( PI/2 * n/(nx/10) );
                x[nx-1-n] *= sin( PI/2 * n/(nx/10) );
        }
}



post_calculation( nmax, x, nx )
```

```
int     nmax, nx;
double *x;
{
        while( nx-- ) *x++ /= (nmax/0.875);
}



zero( x, nx )
int             nx;
double *x;
{
        while( nx-- ) *x++ = 0;
}



frequency( sampling, x, nx )
int     nx;
double sampling, *x;
{
        int     n;

        for( n = 0; n < nx; n++ )
                *x++ = n / ( sampling * 2*nx );
}



CSV_output( x1, x2, x3, x4, x5, x6, x7, nx, filename )
char    *filename;
int             nx;
double *x1, *x2, *x3, *x4, *x5, *x6, *x7;
{
        int             n;

        fp = fopen( filename, "a" );
                fprintf( fp, "freqency,  PSD1, PSD2, COH, phase¥n" );
                for( n = 0; n < nx; n++ )
                        fprintf( fp, "%12e,  %12e, %12e, %12e, %12e¥n",
                                *x1++, *x2++, *x3++, *x4++, *x5++, *x6++, *x7++ );
        fclose( fp );
}



CSV_output2( x1, x2, nx, filename )
char    *filename;
int             nx;
double *x1, *x2;
{
        int     n;

        fp = fopen( filename, "a" );
                fprintf( fp, "freqency,  PSD¥n" );
                for( n = 0; n < nx; n++ )
                        fprintf( fp, "%12e,  %12e¥n", *x1++, *x2++ );
        fclose( fp );
}



make_filename( filename, filename2, c, i1, i2 )
```

```
char    *filename, *filename2, *c;
int             i1, i2;
{
        while( *filename ) *filename2++ = *filename++;
        *filename2++ = *c;
        *filename2++ = ( 0x30 + i1 );
        *filename2++ = ( 0x30 + i2 );
        *filename2   = 0x00;
}



int input_int( s )
char    s[];
{
        int     i;

        printf( "%s = ", s );
        scanf( "%d", &i );

        return i;
}



long input_long( s )
char    s[];
{
        long    l;

        printf( "%s = ", s );
        scanf( "%ld", &l );

        return l;
}



PSD( sampling, xr, xi, psd, nx )
int     nx;
double sampling, *xr, *xi, *psd;
{
        int     n;

        for( n = 0; n < nx/2; n++ )
                psd[n] += ( sampling/nx * ( xr[n]*xr[n]+xi[n]*xi[n] ));
}



CSD( sampling, xr1, xi1, xr2, xi2, csr, csi, nx )
int     nx;
double sampling, *xr1, *xi1, *xr2, *xi2, *csr, *csi;
{
        int     n;

        for( n = 0; n < nx/2; n++ ) {
                csr[n] += ( sampling/nx * ( xr1[n]*xr2[n] + xi1[n]*xi2[n] ));
                csi[n] += ( sampling/nx * ( xi1[n]*xr2[n] - xr1[n]*xi2[n] ));
        }
}


COH( ps1, ps2, csr, csi, coh, phase, nx )
```

```
int     nx;
double *ps1, *ps2, *csr, *csi, *coh, *phase;
{
        int     n;

        for( n = 0; n < nx; n++ ) {
                coh[n]   = sqrt( csr[n]*csr[n] + csi[n]*csi[n] )
                         /sqrt( ps1[n]*ps2[n] );
                phase[n] = atan( csi[n] / csr[n] );
        }
}



Fourier( xr, xi, nx )
int     nx;
double *xr, *xi;
{
        int     j, k, n;
        double x[FTMAXSIZE], s[FTMAXSIZE];

        for( n = 0; n < nx; n++ ) x[n] = xr[n];
        for( n = 0; n <= nx/4; n++ ) s[n] = sin(2.*PI*n/nx);
        for( n = 0; n <= nx/4; n++ ) {
                s[n+nx/4]   =  s[nx/4-n];
                s[n+nx/2]   = -s[n];
                s[n+3*nx/4] = -s[nx/4-n];
                s[n+nx]     =  s[n];
        }

        for( k = 0; k < nx/2; k++ ) {
                xr[k] = 0;
                xi[k] = 0;
                for( j = 0; j < nx; j++ ) {
                        n = (k*j)%nx;
                        xr[k] += (x[j]*s[n+nx/4]);
                        xi[k] += (x[j]*s[n]);
                }
        }
}



EDX1000AS_read_header( filename, header, ch_info )
char    *filename;
long    header[20];
double ch_info[32][6];
{
        int     ch;
        long    sample, read_bytes(), size_of_file();
        double read_real();

        fp = fopen( filename, "rb" );
                header[0] = read_bytes( 631, 2 );   /* Top Channel   */
                header[1] = read_bytes( 651, 2 );   /* Last Channel  */
                header[2] = read_bytes( 671, 2 );   /* Channel Size  */
                sample    = read_bytes( 691, 1 );   /* Sample Rate   */

                switch( sample ) {
                        case  6 : header[3] = 10e3; break;
                        case  7 : header[3] =  5e3; break;
                        case  8 : header[3] =  2e3; break;
                        case  9 : header[3] =  1e3; break;
                        case 10 : header[3] =  500; break;
```

```
                case 11 : header[3] =  200; break;
                case 12 : header[3] =  100; break;
                case 13 : header[3] =   50; break;
                case 14 : header[3] =   20; break;
                case 15 : header[3] =   10; break;
                case 16 : header[3] =    5; break;
                case 17 : header[3] =    2; break;
                case 18 : header[3] =    1; break;
                case 19 : header[3] =  0.5;
                printf( "*** Use Manual Sample Rate ! *** ); break;
                case 20 : header[3] =  0.2;
                printf( "*** Use Manual Sample Rate ! *** ); break;
                case 21 : header[3] =  0.1;
                printf( "*** Use Manual Sample Rate ! *** ); break;
        }

        for( ch = 0; ch < 32; ch++ ) {
                ch_info[ch][0] = read_bytes( 42*ch +  961, 1 );    /*N/A */
                ch_info[ch][1] = 1;                               /* A  */
                ch_info[ch][2] = 0;                               /* B  */
                ch_info[ch][3] = read_real(  42*ch + 1241 );      /* FS */
                ch_info[ch][4] = read_bytes( 42*ch + 1301, 2 );   /* Z  */
                ch_info[ch][5] = read_real(  42*ch + 1321 );      /* CF */
        }
    fclose( fp );

    /* Number of Points */
#if SOF
    header[4] = ( size_of_file( filename ) - 128 * 22 ) / header[2] / 2;
#else
    header[4] = 0;
#endif
}



EDX1000AS_read_data( x, ch, nx, nx0, filename, header, c )
char    *filename;
int             ch, nx;
long    header[20], nx0;
double *x, c[32][6];
{
    int             idat;
    long    n, ofs, read_bytes();

    fp = fopen( filename, "rb" );
        for( n = nx0; n < nx0 + nx; n++ ) {
                ofs  = ( n * header[2] + ch - 1 ) * 21 + 128 * 221;
                idat = (short int) read_bytes( ofs, 2 ) / 16;
                *x++ = c[ch][3]/2000.*(idat-c[ch][4])/c[ch][5]*c[ch][1]
                        + c[ch][2];
        }
    fclose( fp );
}



EDX1000AS_printf_header( header, ch_info )
long    header[20];
double ch_info[32][6];
{
    int             ch;

    printf( "Top Channel  =¥t%3ld¥n", header[0] );
```

```
        printf( "Last Channel =¥t%3ld¥n", header[1] );
        printf( "Channel Size =¥t%3ld¥n", header[2] );
        printf( "Sample Frequency =¥t%3ld¥n", header[3] );
        printf( "Number of Points =¥t%6ld¥n", header[4] );
/*
        for( ch = 0; ch < 32; ch++ ) {
                printf( "%3d:F = %5.3f, A = %5.3f, B = %5.3f, ",
                        ch, ch_info[ch][0], ch_info[ch][1], ch_info[ch][2] );
                printf( "FS = %5.3f, Z = %5.3f, CF = %5.3f¥n",
                        ch_info[ch][3], ch_info[ch][4], ch_info[ch][5] );
        }
*/
}



EDX1000AS_fprintf_header( filename, header, ch_info )
char   *filename;
long   header[20];
double ch_info[32][6];
{
        int     ch;

        fp = fopen( filename, "w" );
                fprintf( fp, "Top Channel =¥t%3ld¥n", header[0] );
                fprintf( fp, "Last Channel=¥t%3ld¥n", header[1] );
                fprintf( fp, "Channel Size=¥t%3ld¥n", header[2] );
                fprintf( fp, "Sample Frequency =¥t%3ld¥n", header[3] );
                fprintf( fp, "Number of Points =¥t%6ld¥n", header[4] );
                fprintf( fp, "¥n" );
                fprintf( fp, "Estimate Channel1=¥t%6ld¥n", header[5] );
                fprintf( fp, "Estimate Channel2=¥t%6ld¥n", header[6] );
                fprintf( fp, "Estimate Points  =¥t%6ld¥n", header[7] );
                fprintf( fp, "Start Point      =¥t%6ld¥n", header[8] );
                fprintf( fp, "Average =¥t%7.5f¥n", (double)header[9] /1000000 );
                fprintf( fp, "St. Dev.=¥t%7.5f¥n", (double)header[10]/1000000 );
                fprintf( fp, "¥n" );

        for( ch = 0; ch < 32; ch++ ) {
                fprintf( fp, "%3d:F = %5.3f, A = %5.3f, B = %5.3f, ",
                        ch, ch_info[ch][0], ch_info[ch][1], ch_info[ch][2] );
                fprintf( fp, "FS = %5.3f, Z = %5.3f, CF = %5.3f¥n",
                        ch_info[ch][3], ch_info[ch][4], ch_info[ch][5] );
        }
        fclose( fp );
}



long size_of_file( filename )
char    *filename;
{
        long    filesize = 0;

        fp = fopen( filename, "rb" );
                while( getc( fp ) != EOF ) filesize++;
        fclose( fp );

        return filesize;
}
```

```
double read_real( ofs )
long    ofs;
{
        int     f = 1, i = 5, i_exp;
        double sum = 0;

        fseek( fp, ofs, SEEK_SET );

        if( ( i_exp = getc( fp ) ) != 0 ) {
                while( i-- )
                        sum += (double) getc( fp ) * pow( 2, -8*(i+1) + 1 );
                if( sum >= 1.0 ) {
                        sum -= 1.0;
                        f = -1;
                }
                sum = f * pow( 2, i_exp - 129 ) * ( sum + 1.0 );
        }
        else    sum = 0;

        return sum;
}



long read_bytes( ofs, byte )
int     byte;
long    ofs;
{
        int     i = byte;
        long    sum = 0;

        fseek( fp, ofs, SEEK_SET );
        while( i-- ) sum += getc( fp ) * pow( 256, byte - i - 1 );

        return sum;
}



long lpow( x, a )
int     a, x;
{
         long    sum = 1;

        if( a > 0 ) while( a-- ) sum *= x;
        else if( a < 0 ) while( a++ ) sum /= x;

         return sum;
}
```

付録 E Wavelet 変換プログラム

```
/*
        WTC.c (Based on MHC.c)

        MexicanHat Wavelet Transform Program

        Produced by M.Kondo

        for EDX-1000AS

        Ver. 960604
*/


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

#define         DATAMAX 100000
#define         ENLARGE 8000
#define         NA      28
#define         PI      3.14159265
#define         SOF     0
#define         TAUMAX  1200

FILE    *fopen(), *fp;

main( argc, argv )
char    *argv[];
int     argc;
{
        char    filename_mh1[15], filename_mh2[15],
                filename_gb1[15], filename_gb2[15],
                filename_ml1[15], filename_ml2[15],
                filename_mh1p[15], filename_ml1p[15],
                filename_gb1p[15], filename_dat1p[15],
                filename_dat1[15], filename_dat2[15],
                filename_mh_cxy[15], filename_gb_cxy[15],
                filename_ml_cxy[15], filename_dat_cxy[15];
        int     ch, ia, ich, input_int();
        long    n_dat, n_wc, start_add, header[19], input_long();
        double a, sampling, ch_info[32][6], cxy[TAUMAX],
                mh_cxy[TAUMAX], gb_cxy[TAUMAX], ml_cxy[TAUMAX],
                dat1[DATAMAX], dat2[DATAMAX],
                mh1[DATAMAX], mh2[DATAMAX], gb1[DATAMAX], gb2[DATAMAX],
                ml1[DATAMAX], ml2[DATAMAX];

/*      Reading Experimental & Calculating Condition      */

        EDX1000AS_read_header( argv[1], header, ch_info );
        EDX1000AS_printf_header( header, ch_info );

        ch          = header[5] = atoi(argv[2]);
        n_wc        = header[7] = atol(argv[3]);
        start_add   = header[8] = atol(argv[4]);

        n_dat = n_wc + ENLARGE*2;
        header[6] = ch + 1;
        sampling = 1./header[3];


/*      Preparing Data      */
```

```
        EDX1000AS_read_data( dat1, ch, n_dat, start_add - ENLARGE,
                             argv[1], header, ch_info );
        EDX1000AS_read_data( dat2, ch + 1, n_dat, start_add - ENLARGE,
                             argv[1], header, ch_info );
        wlcnv( dat1, n_dat, ch );
        wlcnv( dat2, n_dat, ch + 1 );


/*      Making Output File      */

        make_filename( argv[1], filename_gb1,  "g", ch );
        make_filename( argv[1], filename_gb2,  "g", ch + 1 );
        make_filename( filename_gb1, filename_gb_cxy, "c", ch + 1 );
        make_filename( filename_gb1, filename_gb1p, "p", 0 );
        EDX1000AS_fprintf_header( filename_gb1,  header, ch_info );
        EDX1000AS_fprintf_header( filename_gb2,  header, ch_info );
        EDX1000AS_fprintf_header( filename_gb_cxy, header, ch_info );
        EDX1000AS_fprintf_header( filename_gb1p, header, ch_info );
/*
        make_filename( argv[1], filename_mh1,  "m", ch );
        make_filename( argv[1], filename_mh2,  "m", ch + 1 );
        make_filename( filename_mh1, filename_mh_cxy, "c", ch + 1 );
        make_filename( filename_mh1, filename_mh1p, "p", 0 );
        EDX1000AS_fprintf_header( filename_mh1,  header, ch_info );
        EDX1000AS_fprintf_header( filename_mh2,  header, ch_info );
        EDX1000AS_fprintf_header( filename_mh1p, header, ch_info );
        EDX1000AS_fprintf_header( filename_mh_cxy, header, ch_info );

        make_filename( argv[1], filename_ml1,  "l", ch );
        make_filename( argv[1], filename_ml2,  "l", ch + 1 );
        make_filename( filename_ml1, filename_ml_cxy, "c", ch + 1 );
        make_filename( filename_ml1, filename_ml1p, "p", 0 );
        EDX1000AS_fprintf_header( filename_ml1,  header, ch_info );
        EDX1000AS_fprintf_header( filename_ml2,  header, ch_info );
        EDX1000AS_fprintf_header( filename_ml_cxy, header, ch_info );
        EDX1000AS_fprintf_header( filename_ml1p, header, ch_info );
*/
        make_filename( argv[1], filename_dat1, "d", ch );
        make_filename( argv[1], filename_dat2, "d", ch + 1 );
        make_filename( filename_dat1, filename_dat_cxy, "c", ch + 1 );
        make_filename( filename_dat1, filename_dat1p, "p", 0 );
        EDX1000AS_fprintf_header( filename_dat1, header, ch_info );
        EDX1000AS_fprintf_header( filename_dat2, header, ch_info );
        EDX1000AS_fprintf_header( filename_dat_cxy, header, ch_info );
        EDX1000AS_fprintf_header( filename_dat1p, header, ch_info );


/*      Main Loop       */

        for( ia = 0; ia < NA; ia++ ) {
                a = sampling*10.*exp((double)ia/4*log(2));
                printf( "%3d¥n", ia );
                Gabor( dat1, gb1, a, n_wc, sampling );
                Gabor( dat2, gb2, a, n_wc, sampling );
                CSV_write_line( gb1, n_wc, filename_gb1 );
                CSV_write_line( gb2, n_wc, filename_gb2 );
                Cxy( gb1, gb2, n_wc, gb_cxy, filename_gb1p );
                CSV_write_line( gb_cxy, TAUMAX, filename_gb_cxy );
/*
                a = sqrt(2.)/(2*PI)*sampling*10.*exp((double)ia/4*log(2));
                printf( "%3d¥n", ia );
                MexicanHat( dat1, mh1, a, n_wc, sampling );
                MexicanHat( dat2, mh2, a, n_wc, sampling );
```

```
                CSV_write_line( mh1, n_wc, filename_mh1 );
                CSV_write_line( mh2, n_wc, filename_mh2 );
                Cxy( mh1, mh2, n_wc, mh_cxy, filename_mh1p );
                CSV_write_line( mh_cxy, TAUMAX, filename_mh_cxy );

                a = 6./(2*PI)*sampling*10.*exp((double)ia/4*log(2));
                printf( "%3d¥n", ia );
                Morlet( dat1, ml1, a, n_wc, sampling );
                Morlet( dat2, ml2, a, n_wc, sampling );
                CSV_write_line( ml1, n_wc, filename_ml1 );
                CSV_write_line( ml2, n_wc, filename_ml2 );
                Cxy( ml1, ml2, n_wc, ml_cxy, filename_ml1p );
                CSV_write_line( ml_cxy, TAUMAX, filename_ml_cxy );
*/
        }
        CSV_write_data( dat1, n_dat, start_add - ENLARGE, filename_dat1 );
        CSV_write_data( dat2, n_dat, start_add - ENLARGE, filename_dat2 );
        Cxy( dat1, dat2, n_dat, cxy, filename_dat1p );
        CSV_write_data( cxy, TAUMAX, 0, filename_dat_cxy );

        printf( "*** Done ! ***¥n" );
}



Cxy( x1, x2, nx, cxy, filename )
char    *filename;
int     nx;
double  *x1, *x2, *cxy;
{
        int     tau, n;
        double  x1_sum, x2_sum;

        x1_sum = x2_sum = 0.0;

        for( n = 0; n < nx; n++ ) {
                x1_sum += x1[n];
                x2_sum += x2[n];
        }
        x1_sum /= (float) nx;
        x2_sum /= (float) nx;

        for( n = 0; n < nx; n++ ) {
                x1[n] -= x1_sum;
                x2[n] -= x2_sum;
        }


        x1_sum = x2_sum = 0.0;

        for( n = 0; n < nx; n++ ) {
                x1_sum += (x1[n]*x1[n]);
                x2_sum += (x2[n]*x2[n]);
        }
        x1_sum /= (float) nx;
        x2_sum /= (float) nx;

        fp = fopen( filename, "a" );
                fprintf( fp, "%12e, %12e¥n", x1_sum, x2_sum );
        fclose( fp );


        for( tau = 0; tau < TAUMAX; tau++ ) {
```

```
            cxy[tau] = 0.0;
            for( n = 0; n < nx - tau; n++ ) cxy[tau] += (x1[n]*x2[n+tau]);
            cxy[tau] /= (float)( nx - tau );
        }
}


MexicanHat( dat, wc, a, size, sampling )
long    size;
double  *dat, *wc, a, sampling;
{
        int     i, ib, it, half_size;
        double  b, t, psi;

        half_size = (int) ( 3*sqrt(3)*a/sampling );
        printf( "a = %5.3e¥tSize =%4d¥n", a, 2*half_size + 1 );

       for( i = 0; i < size; i++ ) wc[i] = 0.0;

       for( i = 0; i < size; i++ ) {
              ib = i + ENLARGE;
              b  = (double) ib * sampling;
              for( it = ib - half_size; it <= ib + half_size; it++) {
                     t = (double) it * sampling;
                     psi = -((t - b)*(t - b)/( a*a ) - 1)
                         * exp( -(t - b)*(t - b)/( 2.*a*a ));
                     wc[i] += ( 1./sqrt(a) * dat[it] * psi * sampling );
               }
        }
}


Gabor( dat, wc, a, size, sampling )
long    size;
double  *dat, *wc, a, sampling;
{
        int     i, ib, it, half_size;
        double  b, r, t, psi, wp;

       r = PI*sqrt( 2./log(2) );
       wp = 2.*PI;
        half_size = (int) ( 7.*PI*a/wp/sampling );

       for( i = 0; i < size; i++ ) wc[i] = 0.0;

        printf( "a = %5.3e¥tSize =%4d¥n", a, 2*half_size + 1 );

       for( i = 0; i < size; i++ ) {
              ib = i + ENLARGE;
              b  = (double) ib * sampling;
              for( it = ib - half_size; it <= ib + half_size; it++) {
                     t = (double) it * sampling;
                     psi = 1./sqrt(sqrt(PI))*sqrt(wp/r)
                         * exp( -(t - b)/a*wp/r*(t - b)/a*wp/r/2. )
                         * cos( (t - b)/a*wp );
                     wc[i] += ( 1./sqrt(a) * dat[it] * psi * sampling );
               }
        }
 }
```

```
Morlet( dat, wc, a, size, sampling )
long   size;
double  *dat, *wc, a, sampling;
{
        int     i, ib, it, half_size;
        double  b, k, t, psi;

       k = 6;
        half_size = (int) ( 8./k*PI*a/sampling );
        printf( "a = %5.3e¥tSize =%4d¥n", a, 2*half_size + 1 );

       for( i = 0; i < size; i++ ) wc[i] = 0.0;

       for( i = 0; i < size; i++ ) {
              ib = i + ENLARGE;
              b  = (double) ib * sampling;
              for( it = ib - half_size; it <= ib + half_size; it++) {
                     t = (double) it * sampling;
                     psi = cos( k*(t - b)/a )
                         * exp( -(t - b)*(t - b)/( 2.*a*a ));
                     wc[i] += ( 1./sqrt(a) * dat[it] * psi * sampling );
               }
        }
}



wlcnv( x, nx, ch )
int    ch;
long   nx;
double *x;
{
       double a, b;

/*  These conversion data are for q4189-92.  */

       switch( ch ) {
              case  1 : a = -70.90; b = 686.8; break;
              case  2 : a = -68.96; b = 687.3; break;
              case  3 : a = -69.59; b = 686.3; break;
              case  4 : a = -69.33; b = 685.8; break;
              case  5 : a = -70.02; b = 686.8; break;
              case  6 : a = -67.67; b = 687.0; break;
              case  7 : a = -68.71; b = 686.2; break;
              case  8 : a = -68.72; b = 687.2; break;
              case  9 : a = -67.66; b = 686.4; break;
              case 10 : a = -68.16; b = 686.1; break;
              case 11 : a = -69.42; b = 686.2; break;
              case 12 : a = -70.92; b = 686.4; break;
       }
       while( nx-- ) {
              *x = *x * a + b;
              *x++;
       }
}



make_filename( filename, filename2, c, i )
char   *filename, *filename2, *c;
int    i;
{
       while( *filename ) *filename2++ = *filename++;
```

```
        *filename2++ = *c;
        if( i > 9 ) i += 7;
        *filename2++ = ( 0x30 + i );
        *filename2   = 0x00;
}


CSV_write_line( x, nx, filename )
char    *filename;
long    nx;
double  *x;
{
        fp = fopen( filename, "a" );
                while(nx--) fprintf( fp, "%12e, ", *x++ );
                fprintf( fp, "¥n" );
        fclose( fp );
}


CSVLOG_write_line( x, nx, filename )
char    *filename;
long    nx;
double  *x;
{
        fp = fopen( filename, "a" );
                while(nx--) fprintf( fp, "%12e, ", log(*x++) );
                fprintf( fp, "¥n" );
        fclose( fp );
}


CSV_write_data( x, nx, nx0, filename )
char    *filename;
long    nx;
long    nx0;
double  *x;
{
        long    l;

        fp = fopen( filename, "a" );
                for( l = 0; l < nx; l++ )
                        fprintf( fp, "%7ld,  %12e¥n", l + nx0, *x++ );
        fclose( fp );
}


EDX1000AS_read_header( filename, header, ch_info )
char    *filename;
long    header[20];
double ch_info[32][6];
{
        int     ch;
        long    sample, read_bytes(), size_of_file();
        double read_real();

        fp = fopen( filename, "rb" );
               header[0] = read_bytes( 631, 2 );    /* Top Channel   */
               header[1] = read_bytes( 651, 2 );    /* Last Channel  */
               header[2] = read_bytes( 671, 2 );    /* Channel Size  */
```

```
        sample    = read_bytes( 691, 1 );   /* Sample Rate   */

        switch( sample ) {
                case  6 : header[3] = 10e3; break;
                case  7 : header[3] =  5e3; break;
                case  8 : header[3] =  2e3; break;
                case  9 : header[3] =  1e3; break;
                case 10 : header[3] =  500; break;
                case 11 : header[3] =  200; break;
                case 12 : header[3] =  100; break;
                case 13 : header[3] =   50; break;
                case 14 : header[3] =   20; break;
                case 15 : header[3] =   10; break;
                case 16 : header[3] =    5; break;
                case 17 : header[3] =    2; break;
                case 18 : header[3] =    1; break;
                case 19 : header[3] =  0.5;
                printf( "*** Use Manual Sample Rate ! *** ); break;
                case 20 : header[3] =  0.2;
                printf( "*** Use Manual Sample Rate ! *** ); break;
                case 21 : header[3] =  0.1;
                printf( "*** Use Manual Sample Rate ! *** ); break;
        }

        for( ch = 0; ch < 32; ch++ ) {
                ch_info[ch][0] = read_bytes( 42*ch +  961, 1 );   /*N/A */
                ch_info[ch][1] = 1;                              /* A  */
                ch_info[ch][2] = 0;                              /* B  */
                ch_info[ch][3] = read_real(  42*ch + 1241 );      /* FS */
                ch_info[ch][4] = read_bytes( 42*ch + 1301, 2 );   /* Z  */
                ch_info[ch][5] = read_real(  42*ch + 1321 );      /* CF */
        }
    fclose( fp );

    /* Number of Points */
#if SOF
    header[4] = ( size_of_file( filename ) - 128 * 22 ) / header[2] / 2;
#else
    header[4] = 0;
#endif
}



EDX1000AS_read_data( x, ch, nx, nx0, filename, header, c )
char   *filename;
int    ch, nx;
long   header[20], nx0;
double *x, c[32][6];
{
    int           idat;
    long   n, ofs, read_bytes();
    double average, stdev, sum, sum2;

    sum = sum2 = 0;

    fp = fopen( filename, "rb" );
        for( n = nx0; n < nx0 + nx; n++ ) {
                ofs  = ( n * header[2] + ch - 1 ) * 21 + 128 * 221;
                idat = (short int) read_bytes( ofs, 2 ) / 16;
                *x = c[ch][3]/2000.*(idat-c[ch][4])/c[ch][5]*c[ch][1] + c[ch][2];
                sum  += *x;
                sum2 += (*x * *x);
                x++;
```

```
            }
        fclose( fp );

        average = sum/nx;
        stdev   = sqrt( sum2/nx - average*average );
        header[9]  = average * 100000;
        header[10] = stdev   * 100000;
}


EDX1000AS_printf_header( header, ch_info )
long   header[20];
double ch_info[32][6];
{
        int             ch;

        printf( "Top Channel  =¥t%3ld¥n", header[0] );
        printf( "Last Channel =¥t%3ld¥n", header[1] );
        printf( "Channel Size =¥t%3ld¥n", header[2] );
        printf( "Sample Rate  =¥t%3ld¥n", header[3] );
        printf( "Number of Points =¥t%6ld¥n", header[4] );
/*
        for( ch = 0; ch < 32; ch++ ) {
                printf( "%3d:F = %5.3f, A = %5.3f, B = %5.3f, ",
                        ch, ch_info[ch][0], ch_info[ch][1], ch_info[ch][2] );
                printf( "FS = %5.3f, Z = %5.3f, CF = %5.3f¥n",
                        ch_info[ch][3], ch_info[ch][4], ch_info[ch][5] );
        }
*/
}


EDX1000AS_fprintf_header( filename, header, ch_info )
char   *filename;
long   header[20];
double ch_info[32][6];
{
        int             ch;

        fp = fopen( filename, "w" );
                fprintf( fp, "Top Channel =¥t%3ld¥n", header[0] );
                fprintf( fp, "Last Channel=¥t%3ld¥n", header[1] );
                fprintf( fp, "Channel Size=¥t%3ld¥n", header[2] );
                fprintf( fp, "Sample Rate =¥t%3ld¥n", header[3] );
                fprintf( fp, "Number of Points=¥t%6ld¥n", header[4] );
                fprintf( fp, "¥n" );
                fprintf( fp, "Estimate Channel1=¥t%6ld¥n", header[5] );
                fprintf( fp, "Estimate Channel2=¥t%6ld¥n", header[6] );
                fprintf( fp, "Estimate Points  =¥t%6ld¥n", header[7] );
                fprintf( fp, "Start Point      =¥t%6ld¥n", header[8] );
                fprintf( fp, "Average =¥t%7.5f¥n", (double)header[9] /1000000 );
                fprintf( fp, "St. Dev.=¥t%7.5f¥n", (double)header[10]/1000000 );
                fprintf( fp, "¥n" );

        for( ch = 0; ch < 32; ch++ ) {
                fprintf( fp, "%3d:F = %5.3f, A = %5.3f, B = %5.3f, ",
                        ch, ch_info[ch][0], ch_info[ch][1], ch_info[ch][2] );
                fprintf( fp, "FS = %5.3f, Z = %5.3f, CF = %5.3f¥n",
                        ch_info[ch][3], ch_info[ch][4], ch_info[ch][5] );
        }
```

```
        fclose( fp );
}



long size_of_file( filename )
char    *filename;
{
        long    filesize = 0;

        fp = fopen( filename, "rb" );
                while( getc( fp ) != EOF ) filesize++;
        fclose( fp );

        return filesize;
}



double read_real( ofs )
long    ofs;
{
        int     f = 1, i = 5, i_exp;
        double sum = 0;

        fseek( fp, ofs, SEEK_SET );

        if( ( i_exp = getc( fp ) ) != 0 ) {
                while( i-- )
                        sum += (double) getc( fp ) * pow( 2, -8*(i+1) + 1 );
                if( sum >= 1.0 ) {
                        sum -= 1.0;
                        f = -1;
                }
                sum = f * pow( 2, i_exp - 129 ) * ( sum + 1.0 );
        }
        else    sum = 0;

        return sum;
}



long read_bytes( ofs, byte )
int     byte;
long    ofs;
{
        int     i = byte;
        long    sum = 0;

        fseek( fp, ofs, SEEK_SET );
        while( i-- ) sum += getc( fp ) * pow( 256, byte - i - 1 );

        return sum;
}



int input_int( s )
char    s[];
{
        int     i;

        printf( "%s = ", s );
```

```
        scanf( "%d", &i );

        return i;
}



long input_long( s )
char    s[];
{
        long    l;

        printf( "%s = ", s );
        scanf( "%d", &l );

        return l;
}



Daubechies( dat, size, sampling, phi, filename )
char    *filename;
long    size;
double *dat, *phi, sampling;
{
        int     i, ib, it, iphi, half_size, n;
        double a, b, b0, bc, t, psi, wc[DATAMAX];

        for( n = -6; n < 2; n++ ) {
                a = pow( 2, n );
                half_size = (int) ( 8.*a/sampling );
                printf( "a = %5.3e¥tSize =%4d¥n", a, 2*half_size + 1 );

                for( i = 0; i < size; i++ ) wc[i] = 0.0;

                b0 = bc = pow( 2, n );
                for( i = 0; i < size; i++ ) {
                        ib = i + ENLARGE;
                        bc += sampling;
                        if( bc >= b0 ) {
                                b = (double) ib * sampling;
                                bc -= b0;
                                for( it=ib-half_size; it<=ib+half_size; it++) {
                                        t = (double) it * sampling;
                                        iphi = (int)((t - b)*D8_N/a);
                                        if( abs(iphi) > D8_M*D8_N/2 - 2 )
                                                psi = 0;
                                        else
                                                psi = phi[iphi+D8_M*D8_N/2];
                                        wc[i] += ( 1./sqrt(a)*dat[it]*psi*sampling );
                                }
                        }
                        else wc[i] = wc[i-1];
                }
                CSV_write_line( wc, size, filename );
        }
}
```