Internal Use Only
非公開

07

TR－A－0065

線形予測分析によるフォルマント抽出精度の検討

002

加藤宏明　平原達也

1990. 1. 8

ATR視聴覚機構研究所

〒619-02 京都府相楽郡精華町乾谷　☎07749-5-1411

ATR Auditory and Visual Perception Research Laboratories

Inuidani, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1411
Facsimile: +81-7749-5-1408
Telex: 5452-516 ATR J

線形予測分析によるフォルマント抽出精度の検討

1. はじめに

本資料では、線形予測分析を行なう際に、各種パラメータがフォルマント抽出精度に与える影響について検討する。

２章では線形予測分析によるフォルマント抽出の原理について、３章では検討に用いた音声資料について、４章、５章においては各分析手法およびパラメータがフォルマント抽出に与える影響について調べたデータを示す。4.2以降のデータについての詳細な検討は、稿を改めて行なうこととし、データのみを示す。

2. 線形予測法によるフォルマント抽出の原理

この章では、まず線形予測分析の原理について、時間領域、周波数領域の両方から記述し、さらに分析の結果得られる線形予測係数からフォルマントを抽出する方法について述べる。なお、実際に用いた計算機プログラムのC言語によるソースリストを付録Bに示す。また、以下にこの章で用いる各記号、略号の一覧を示す。

$s(n)$	時刻nにおける音声波形の標本値
$x(n)$	s(n)に前処理（プリエンファシス、窓掛け）を施した値
$R(i)$	s(n)またはx(n)の自己相関関数
$r(i)$	s(n)またはx(n)の自己相関係数($\lvert r(i)\rvert \leqq 1$)
α_i , $i=1,2,,,p$	αーパラメータ
$\varepsilon(n)$	時刻nにおける予測残差
σ	ε(n)のRMS値
$\Theta = (\{\alpha_i\}, \sigma)$	線形予測モデルを記述するパラメータの組
$L(s \mid \Theta)$	対数尤度（s(n)がパラメータの組Θで表現される線形系から得られる事後確率）
$H(z)$	線形予測モデルの伝達関数
$P(\omega)$	音声信号s(n)のパワスペクトル
$T(\omega)$	線形予測モデルの全極型パワスペクトル
$\hat{F}_i, \hat{B}_i$	極推定法によるフォルマント周波数、フォルマント帯域幅の抽出値
$\tilde{F}_i, \tilde{B}_i$	ピーク抽出法によるフォルマント周波数、フォルマント帯域幅の抽出値

2.1 線形予測分析

本節では、音声の線形予測分析を時間領域で定式化する。

音声波形のある時刻nでの標本値s(n)は、図2.1に示すような隣接する過去のp個の標本値によって線形結合の形に予測できると仮定する。すなわち、

$$s(n) = \alpha_1 s(n-1)+\alpha_2 s(n-2)+...+\alpha_p s(n-p)+\varepsilon(n) \qquad (2.1)$$

ここでε(n)は、予測誤差を表わしている。このε(n)を最小にするような係数の組$\{\alpha_i\}$を求めるのが線形予測分析である。

(2.1)式を、$\alpha_0=1$とおき、さらに$-\alpha_i=\alpha_i$として書き直すとε(n)は次式のように表わせる。

$$\varepsilon(n) = \sum_{i=0}^{p}\alpha_i s(n-i) \qquad (2.2)$$

したがって、ある時間($0\leqq n<N$)の標本値を用いて、予測を行なった場合ε(n)の2乗平均は、

$$\overline{\varepsilon(n)^2} = 1/N\cdot\sum_{n=0}^{N-1}\varepsilon(n)^2 \qquad (2.3)$$

のように定義され、(2.2),(2.3)式より、

$$\overline{\varepsilon(n)^2} = \overline{(\sum_{i=0}^{p}\alpha_i s(n-i))^2} \qquad (2.4)$$

となる。(2.4)式に示されるように、$\varepsilon(n)^2$は各α_iについての2次式であり、かつ$\alpha_i{}^2$の係数は正であるので、各α_iに対して最小値となる$\varepsilon(n)^2$の極値が存在する。すなわち、予測誤差ε(n)を最小にするような$\{\alpha_i\}$は、(2.4)式の右辺の各α_iについての偏微分を、0とおくことにより得られる連立p元一次方程式の解として求められる。

(2.4)式の右辺を展開し、項別に平均値をとり、$\delta\varepsilon(n)^2/\delta\alpha_i = 0,\ i=1,2,,,p$とおくことによって、連立方程式、

$$\begin{pmatrix} R(0) & R(1) & \cdots & R(p-1) \\ R(1) & R(0) & & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & & \cdot & \cdot \\ R(p-1) & & & R(0) \end{pmatrix}\begin{pmatrix}\alpha_1\\ \alpha_2\\ \cdot\\ \cdot\\ \alpha_p\end{pmatrix} = -\begin{pmatrix}R(1)\\ R(2)\\ \cdot\\ \cdot\\ R(p)\end{pmatrix} \qquad (2.5)$$

$$R(0) = \overline{s(i)^2}$$
$$R(j) = \overline{s(i)s(i-j)} = \overline{s(i)s(i+j)}$$

を得る。ここでR(i)はs(n)が時間的に定常であると仮定した場合の、s(n)の自己相関関数である。

この(2.5)式を解くことによって、線形予測係数$\{\alpha_i\}$が求められる。

2.2 最尤スペクトル推定法

本節では、周波数領域での線形予測分析法の解釈について述べる。

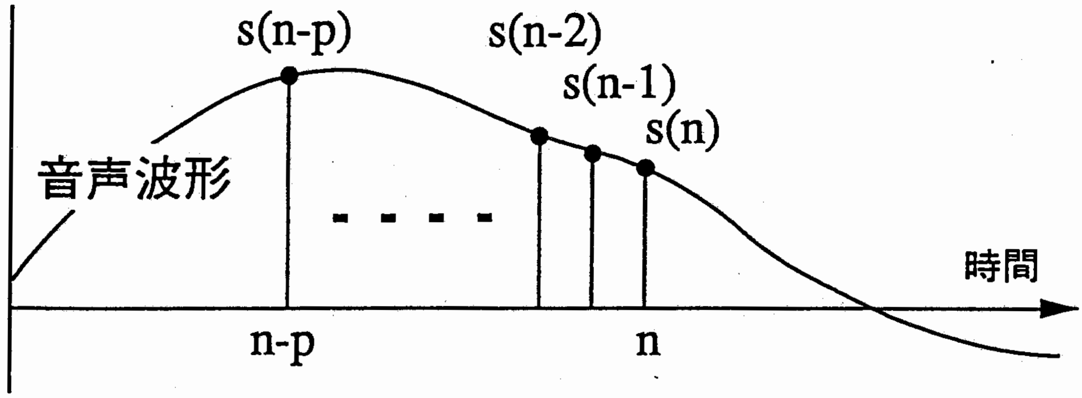


図2.1 音声波形とその標本値

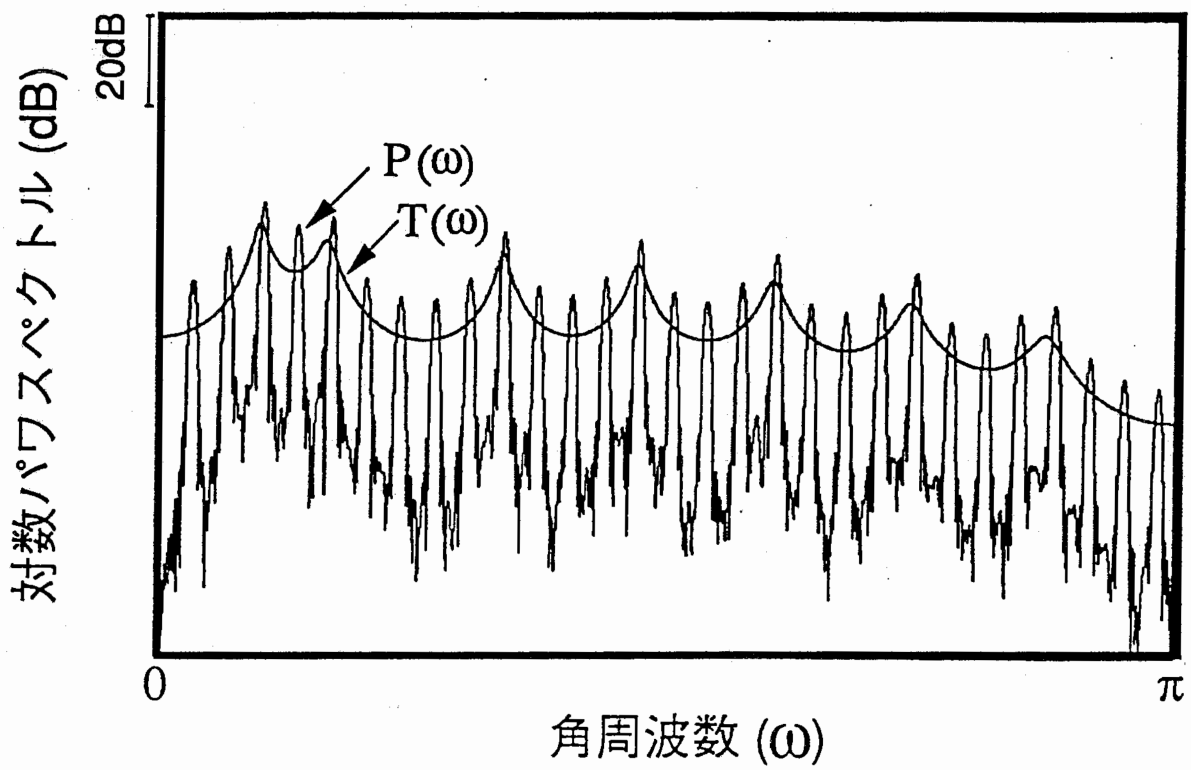


図2.2 DFTパワスペクトルと全極型パワスペクトル

まず音声波形に対して次の仮定をおく。
① 音声生成過程での伝達系である声道は、全極型のスペクトルを持つ系である。
② 音声波形は①の系に平均値0、分散σ^2の定常白色雑音入力が加えられた時の出力とみなせる。

この時、声道の伝達関数H(z)は、(2.2)式と同じ記号を用いて、

$$H(z) = \sigma / \left(\sum_{i=0}^{p} \alpha_i z^{-i}\right) \tag{2.6}$$

のように表わされる。さらに、このH(z)に対応する全極型パワスペクトルT(ω)は、

$$T(\omega) = |H(z)|^2 / 2\pi \tag{2.7}$$

であるから、(2.6),(2.7)式より、

$$T(\omega) = 1/2\pi \cdot \sigma^2 / \left|\sum_{i=0}^{p} \alpha_i z^{-i}\right|^2 \tag{2.8}$$

となる。ここで、zは離散系でフーリエ変換を施した場合の周波数軸を示し、z=exp(jω)である。

上記の仮定のもとでは、観測される音声信号s(n)を用いて、H(z)の特性をもった系からs(n)が出力として得られる尤度（事後確率）を計算することができる。
この尤度を最大にするようなH(z)を表わすパラメータの組$\Theta = (\{\alpha_i\}, \sigma)$を求めるのが、最尤スペクトル推定法である。
尤度の最大値を計算するために、付録A-1で導かれる対数尤度L(s|Θ)を用いる。

$$L(s|\Theta) = -N/2 \cdot \{\underbrace{2\log 2\pi}_{\text{第1項}} + 1/2\pi \cdot \int_{-\pi}^{\pi} (\underbrace{\log T(\omega)}_{\text{第2項}} + \underbrace{P(\omega)/T(\omega)}_{\text{第3項}})\, d\omega\} \tag{2.9}$$

ここで、Nは用いた音声波形の標本数、P(ω)は図2.2に示すような音声信号s(n)のDFTパワスペクトルである。
この対数尤度L(s|Θ)を表わす式は、確率密度関数のスケールファクタである第1項、全極モデルへの入力信号の分散を表わす第2項、そして、DFTパワスペクトルと全極型パワスペクトルの比を表わす第3項からなる。これらのうち、第1項は定数項なので、第2項、第3項のみがL(s|Θ)の変化に寄与する。付録A-1(2)で示されるように、第2項と第3項の和は、$\omega = [-\pi, \pi]$の範囲でP(ω)とT(ω)が常に等しいときに最大値をとり、P(ω)とT(ω)とが等しくなければ、それだけ小さい値をとる。すなわち、L(s|Θ)を最大化することは、P(ω)とT(ω)とのマッチングを行なった場合の誤差を最小化することに帰着される。
次に、L(s|Θ)を最大化するΘを導く。付録A-1(A1.9)式より対数尤度は、

$$L(s|\Theta) = -N/2 \cdot \{\log 2\pi\sigma^2 + 1/\sigma^2 \cdot \sum_{i=0}^{p}\sum_{j=0}^{p} \alpha_i R(i-j) \alpha_j\} \tag{2.10}$$

とも書ける。ここで、R(i)はs(n)の自己相関関数である。
まずL(s|Θ)をσ^2について最大化する。(2.10)式の右辺をσ^2で偏微分し、0とおくことにより、L(s|Θ)を最大化するσ^2の値$\hat{\sigma}^2$が求められる。

$$\hat{\sigma}^2 = \sum_{i=0}^{p}\sum_{j=0}^{p} \alpha_i R(i-j) \alpha_j \tag{2.11}$$

このとき最大化されたL(s|Θ)は、

$$L(s|\Theta) = -N/2\cdot(\log 2\pi\hat{\sigma}^2 + 1) \tag{2.12}$$

(2.12)式は$\hat{\sigma}^2$に関する単調減少関数なので、$L(s|\Theta)$の最大化は、$\hat{\sigma}^2$のα_jについての最小化に帰着される。したがって、(2.11)式の右辺を各α_jについて偏微分し、0とおくことにより、

$$\sum_{i=0}^{p}\alpha_i R(i-j) = 0,\ j = 1,2,,,p \tag{2.13}$$

が得られる。ここで、R(i)は自己相関関数なので、

$$R(i-j) = R(j-i) \tag{2.14}$$

である。さらに、α_0=1として(2.13)式を行列形式に書き改めると、次のp元連立1次方程式を得る。

$$\begin{pmatrix} R(0) & R(1) & \dots & R(p-1) \\ R(1) & R(0) & & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & & & \cdot \\ R(p-1) & & & R(0) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \cdot \\ \alpha_p \end{pmatrix} = -\begin{pmatrix} R(1) \\ R(2) \\ \cdot \\ \cdot \\ R(p) \end{pmatrix} \tag{2.15}$$

これは、前項で導かれた(2.5)式に全く等しい。すなわち、(2.15)式の解$\{\alpha_i\}$は、前項で求めた線形予測係数と同じものである。

2.3 フォルマントの抽出法

本節では、実際に音声波形の標本値からフォルマントを抽出する手順について述べる。

図2.3にフォルマント抽出の手順と、それらに対応する計算機プログラムのルーチン名を示す。

まず、標本値s(n)に適当な前処理を施し、次に、自己相関関数を計算し、連立1次方程式を解いて線形予測係数$\{\alpha_i\}$を求める。$\{\alpha_i\}$からフォルマントを抽出する方法は、次の2つに分けられる。1つは、$\{\alpha_i\}$によって記述される声道の伝達関数H(z)より、極を直接計算する方法であり、これをここでは極推定法と呼ぶ。もう1つは、$\{\alpha_i\}$より全極型パワスペクトル$T(\omega)$を計算し、そのピークの周波数と帯域幅を測定する方法である。これをピーク抽出法と呼ぶ。

以下に、フォルマント抽出の各段階について説明する。

① 前処理

前処理には、音声波形の周波数特性を線形予測法に適するように補正するプリエンファシスと、標本値に、スペクトル歪が小さくなるような重み付けを施す窓かけとの2つの段階がある。

線形予測法では、図2.4(a)に示すように、音源$\varepsilon(n)$として、インパルス、または白色雑音のような周波数特性が平坦なものを仮定している。しかし実際の音声では、特

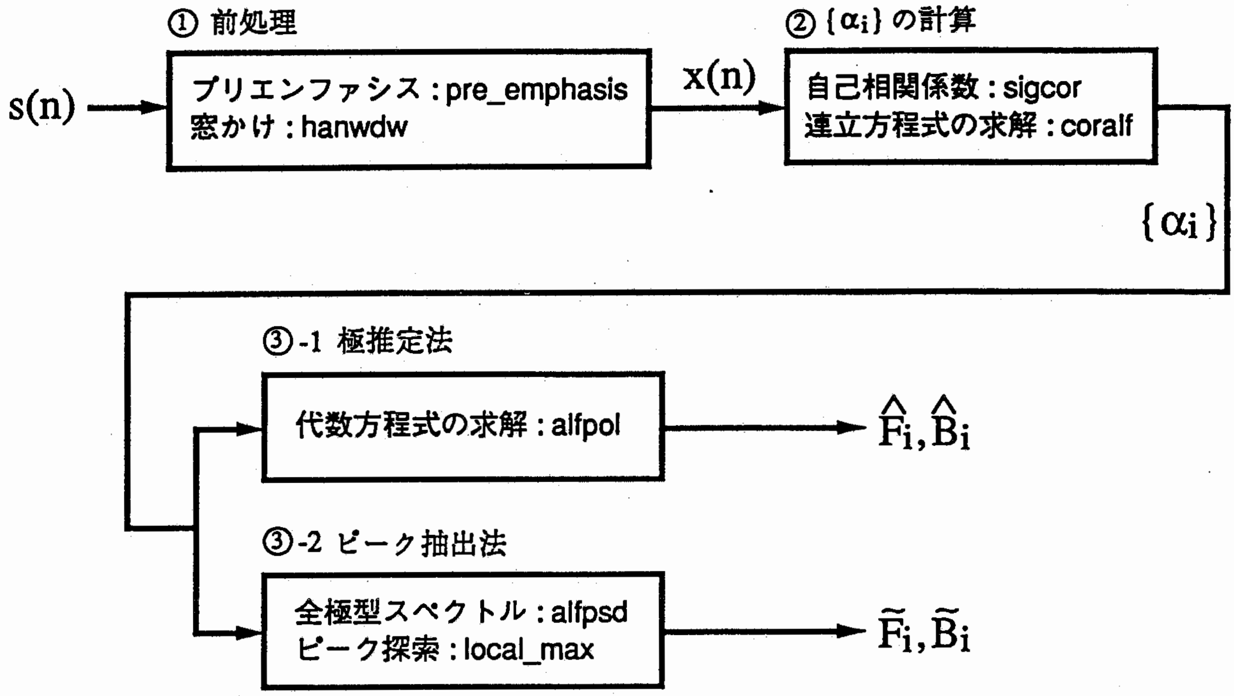


図2.3 フォルマント抽出の手順

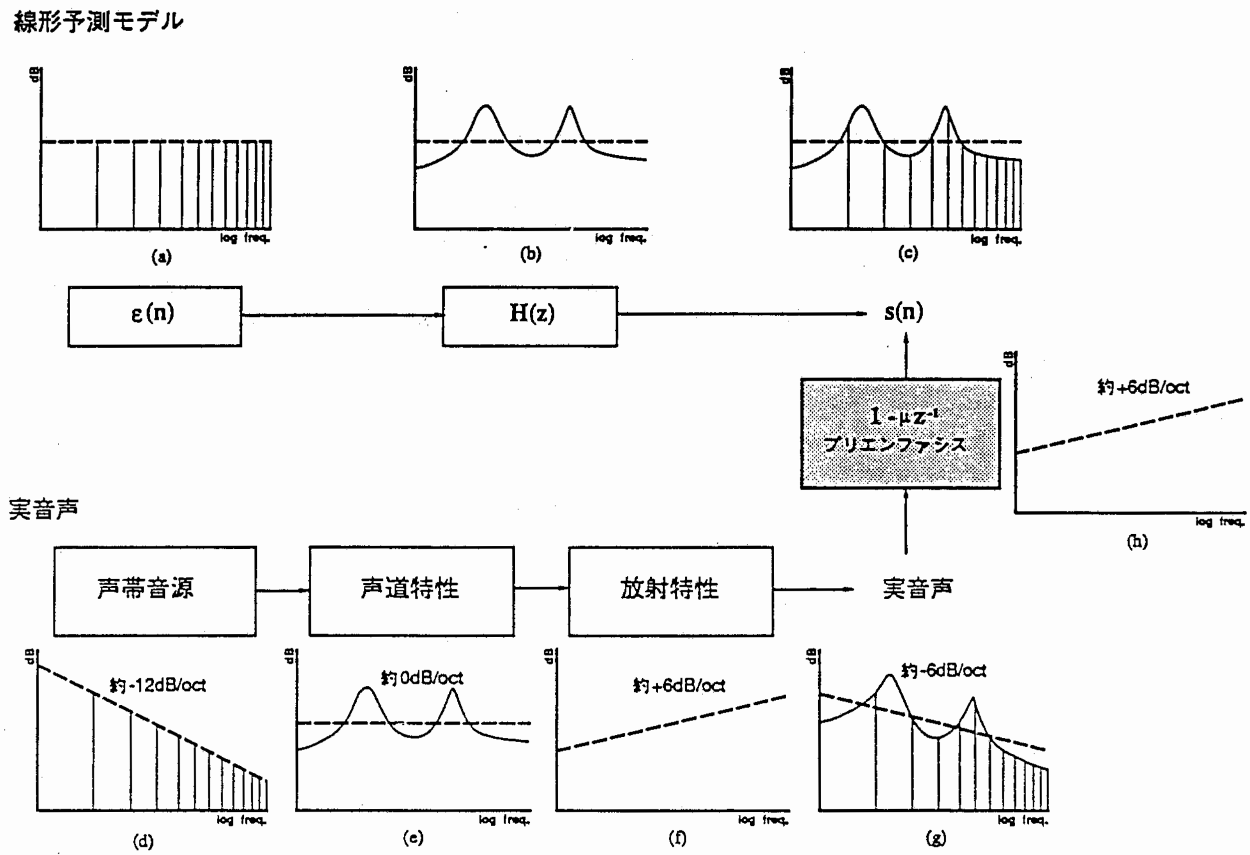


図2.4 声帯音源、放射の特性とプリエンファシス

に有声音の場合には、図2.4(d)に示されるように、声帯音源波の周波数特性は平坦ではなく、一般に約-12dB/octで高域減衰する特性を示す。また、声道から大気中へ放射される際に約+6dB/octの高域強調特性を与えられる(図2.4(f))。その結果、観測される音声の周波数特性は、一般に図2.4(g)のような約-6dB/octの傾きを持つ。したがって、声道での共振特性(図2.4(e))だけを抽出するためには、観測された音声波に、声道以外の特性を打ち消すような操作を加える必要がある。そこで、図2.4(h)に示すように$1-\mu z^{-1}$のフィルタにより高域強調(プリエンファシス)を施す。

このフィルタリングは、$\mu=1$の場合、ナイキスト周波数までの範囲であれば、近似的に+6dB/octの特性を与えるが、$\mu<1$である方がフィルタは安定するので、本稿の分析では$\mu=0.98$に設定した。

次に、プリエンファシスを施された標本値を$s'(n)$として、$s'(n)$に自己相関関数を求めるための窓かけを行なう。

自己相関関数は、

$$R(i) = \sum_{n=-\infty}^{\infty} s'(n)s'(n-i) \tag{2.16}$$

によって定義されるが、実際にはnを$-\infty$から∞まで変化させることは不可能である。そこで、音声標本値に、ある範囲($0 \leqq n \leqq N-1$)の外側では0であるような有限長の窓(ハニング窓など)を乗じた値$x(n)$から、自己相関関数を計算する。すなわち、

$$x(n) = w(n)s'(n) \tag{2.17}$$

ここで、$w(n)$は窓関数の重み係数である。本稿では、窓の両端がなめらかに0になり、したがってスペクトルの雑音レベルが低く抑えられるハニング窓を用いた。

② $\{\alpha_i\}$の計算

本節では、前処理を施された標本値$x(n)$から、(2.5),(2.15)式の連立方程式の係数$R(i)$を求め、さらに、連立方程式を解いて$\{\alpha_i\}$を求める方法について述べる。

$x(n)$は$n<0, n \geqq N$の範囲では0なので、自己相関関数は(2.16),(2.17)式より、

$$R(i) = \sum_{n=0}^{N-1} x(n)x(n-i) \tag{2.18}$$

のように計算できる。本稿の分析では、さらに(2.18)式の$R(i)$を$R(0)$で除し、絶対値が1以下になるように正規化した自己相関係数$r(i)$を用いる((2.19)式)。

$$r(i) = R(i)/R(0) \ , \ 0 \leqq i \leqq N-1 \tag{2.19}$$

次に、(2.5),(2.15)式のp元連立方程式の解法を示す。

(2.5),(2.15)式の係数行列は、自己相関関数の性質より、対角方向の成分が皆等しく、かつ対角軸に対して対称である。このような特別な形(Töplitz型)の係数行列を持つ連立一次方程式では、付録A-3に示すように、逆行列を直接計算することなしに、以下のような逐次計算で解を求めることができる。ここで、$\alpha_i^{(j)}$、$\sigma^2_{(j)}$はj回目のループで求められたα_i、σ^2を示している。k_iは$\alpha_i^{(j)}$の値を一時的に保持するために定義されている。

(1) $\alpha_0 = 1$

$$\kappa_1 = -r(1), \quad \alpha_1^{(1)} = \kappa_1, \quad \sigma^2_{(1)} = 1 - \kappa_1^2, \tag{2.20a}$$

$j = 2$

(2) $\kappa_j = -1/\sigma^2_{(j-1)} \cdot \{r(j) + \sum_{i=1}^{j-1} \alpha_i^{(j-1)} r(j-i)\}$ (2.20b)

(3) $\alpha_j^{(j)} = \kappa_j$ (2.20c)
$\alpha_i^{(j)} = \alpha_i^{(j-1)} + \kappa_j \alpha_{j-i}^{(j-1)}, \ (1 \leqq i \leqq j-1)$ (2.20d)

(4) $\sigma^2_{(j)} = (1 - \kappa_j^2)\sigma^2_{(j-1)}$ (2.20e)
ここで、j＞pならばj=j+1として(2)(3)(4)を繰り返す。j=pならば、

(5) $\alpha_i = \alpha_i^{(j)}, \ (1 \leqq i \leqq j)$
$\sigma^2 = \sigma^2_{(j)}$ (2.20f)
(2.20g)

とすることで、全ての$\{\alpha_i\}$が求められ、計算は終了する。

③-1 極推定法

本節では、$\{\alpha_i\}$から極推定法によりフォルマントを推定する手順を述べる。線形予測法によって推定される声道の伝達関数は(2.6)式に示されるように、

$$H(z) = \sigma / \sum_{i=0}^{p} \alpha_i z^{-i}, \quad \alpha_0 = 1$$

で表わされる。したがってH(z)の極は、代数方程式

$$\sum_{i=0}^{p} \alpha_i z^{-i} = 0 \qquad (2.21)$$

をzについて解いたときの根として求められる。(2.21)式は実係数の方程式なので、実根または共役複素根を持つ。それらの根を

$$z = \gamma_i \cdot \exp(\pm j \lambda_i) \qquad (2.22)$$

とおく。一方、フォルマント周波数およびフォルマント帯域幅の推定値$\hat{F}_i$,$\hat{B}_i$はs平面で、

$$s = -\pi \hat{B}_i \pm j2\pi \hat{F}_i \qquad (2.23)$$

と表わされるので、(2.23)式とs平面からz平面への変換式$z=\exp(s\tau)$より、

$$\begin{aligned} z &= \exp(-\pi \hat{B}_i \tau \pm j2\pi \hat{F}_i \tau) \\ &= \exp(-\pi \hat{B}_i \tau) \cdot \exp(\pm j2\pi \hat{F}_i \tau) \end{aligned} \qquad (2.24)$$

ただし、τは音声波形のサンプリング周期である。
(2.22),(2.24)式の虚部より、

$$\pm \hat{F}_i = \pm \lambda_i / 2\pi\tau \qquad (2.25)$$

$\hat{F}_i$は周波数なので、$\hat{F}_i \geqq 0$とすると、

$$\hat{F}_i = |\lambda_i| / 2\pi\tau \qquad (2.26)$$

が得られる。また、(2.22),(2.24)式の実部より、

$$\hat{B}_i = -\log\gamma_i/\pi\tau \qquad (2.27)$$

が得られる。

③-2 ピーク抽出法

本節では、$\{\alpha_i\}$からフォルマントを抽出するもう1つの方法、ピーク抽出法の手順を述べる。

声道の共振特性を推定した全極型パワスペクトル$T(\omega)$は、(2.8)式に示されるように、$z = \exp(j\omega)$とおいて、

$$T(\omega) = 1/2\pi \cdot \sigma^2/|\sum_{i=0}^{p}\alpha_i\exp(ji\omega)|^2 \qquad (2.28)$$

と表わせる。$\{\alpha_i\}$のフーリエ変換を$A(\exp(j\omega))$とおくと、離散フーリエ変換の定義より、

$$A(\exp(j\omega)) = \sum_{i=-\infty}^{\infty}\alpha_i\exp(ji\omega) \qquad (2.29)$$

$\{\alpha_i\}$は$0\leqq i\leqq p$の範囲で定義されているので、(2.29)式は、

$$A(\exp(j\omega)) = \sum_{i=0}^{p}\alpha_i\exp(ji\omega) \qquad (2.30)$$

となり、したがって(2.28),(2.30)式より、$T(\omega)$は、

$$T(\omega) = 1/2\pi \cdot \sigma^2/|A(\exp(j\omega))|^2 \qquad (2.31)$$

と書ける。すなわち、$T(\omega)$は$\{\alpha_i\}$のパワスペクトルの逆数に、音声波形の平均パワσ^2を乗じたものとして求められる。

フォルマント周波数およびフォルマント帯域幅の抽出値F_i,B_iは、付録A-4に示すような方法で、$T(\omega)$のピークを探索することで得られる。

参考文献

(1) 城戸健一、ディジタル信号処理入門、丸善(1985)
(2) 東倉洋一、偏自己相関係数を用いた音声分析合成系における音声品質向上の研究、東京大学学位論文(1980)
(3) 斎藤収三、中田和男、音声情報処理の基礎、オーム社(1981)
(4) J.Markel, A.Gray Jr.原著、鈴木久喜訳、音声の線形予測、コロナ社(1980)
(5) 渡辺力他編、数値解析とFORTRAN、丸善(1983)

3. 分析資料

本章では、分析実験に用いた、合成母音資料の合成パラメータについて述べる。
資料は、Klatt型のカスケードフォルマント合成器によって合成された時間長500 msec.の母音資料である。
各々の資料においては、基本周波数、フォルマント周波数、フォルマント帯域幅は時間的に定常である。

基本周波数は、表3.1に示すように、0.8 Bark～4.2 Barkまでを1/40 Bark～1/10 Barkごとの間隔にとった63通りの値を用いた。表3.1は、左の列から順に、合成器に与えたF0のBark単位表示および、Hz単位表示、そして音源パルス励起間隔の逆数のHz単位表示、およびそのBark単位表示を示している。本稿の分析では、この最終列の値を基本周波数F0と定義する。但し、合成時の時間分解能は50μsec.なので、音源パルス励起間隔は50μsec.の単位に切り上げられた値となる。

フォルマント周波数に関しては、資料は２つのグループに分けられる。１つは、フォルマント周波数が、リニアな周波数軸上で等間隔にとられる、中性母音のグループで、もう１つは、日本語５母音のフォルマント構造を模した、５母音のグループである。
中性母音のグループでは、フォルマント周波数Fiは、

$$Fi = (2i-1)F1$$

を満足する。ここで、Fiの単位はHzである。
本稿では、F1については3Bark(308.5Hz)，5Bark(530.5Hz)，8Bark(922.2Hz)の３種類、フォルマントの個数Nfについては5,7,9の３種類の値を選んだ。但し、F1=8 Barkのものでは、F6=10144.2Hzとなり、F6以上はナイキスト周波数を超える値になるので、Nfは5だけである。
５母音のグループでは、Nfは7だけとした。F1,F2は中性母音のグループのいずれかのものに一致しており、F3～F7は母音の種類によらず一定である。
図3.1に中性母音資料、５母音資料のF1,F2の位置関係を示す。

フォルマント帯域幅は、F1=5 Barkの中性母音資料についてのみ、50Hz、100Hz、200Hz、300Hzの４通りを用いており、それ以外は、１つのフォルマント周波数のセットに対しては１種類だけである。
表3.2に中性母音資料のフォルマント周波数、フォルマント帯域幅を示す。
表3.3に５母音資料のフォルマント周波数、フォルマント帯域幅を示す。

これらの組合せでできる15種類のフォルマントのセットに対して、前述の63種のF0で合成した、合計945個の合成母音資料を本稿の分析では用いる。

表3.1 母音資料の基本周波数

Given value Bark	Given value Hz	Real value Hz	Real value Bark	Given value Bark	Given value Hz	Real value Hz	Real value Bark
0.800	81.03	80.97	0.799	1.800	183.14	181.81	1.787
0.825	83.57	83.33	0.823	1.850	188.29	186.91	1.837
0.850	86.11	85.83	0.847	1.900	193.45	192.30	1.889
0.875	88.65	88.49	0.874	1.950	198.61	198.02	1.944
0.900	91.19	90.90	0.897	2.000	203.77	202.02	1.983
0.925	93.73	93.45	0.922	2.050	208.95	208.33	2.044
0.950	96.27	96.15	0.949	2.100	214.12	212.76	2.087
0.975	98.81	98.52	0.972	2.150	219.31	217.39	2.132
1.000	101.35	101.01	0.997	2.200	224.50	222.22	2.178
1.025	103.89	103.62	1.022	2.250	229.69	227.27	2.227
1.050	106.44	106.38	1.049	2.300	234.89	232.55	2.278
1.075	108.98	108.69	1.072	2.350	240.10	238.09	2.331
1.100	111.52	111.11	1.096	2.400	245.32	243.90	2.386
1.125	114.07	113.63	1.121	2.500	255.77	253.16	2.475
1.150	116.62	116.27	1.147	2.600	266.26	263.15	2.570
1.175	119.16	119.04	1.174	2.700	276.77	273.97	2.673
1.200	121.71	121.21	1.195	2.800	287.32	285.71	2.785
1.225	124.26	124.22	1.225	2.900	297.90	294.11	2.864
1.250	126.81	126.58	1.248	3.000	308.52	307.69	2.992
1.275	129.36	129.03	1.272	3.100	319.17	317.46	3.084
1.300	131.91	131.57	1.297	3.200	329.86	327.86	3.181
1.325	134.46	134.22	1.323	3.300	340.59	338.98	3.285
1.350	137.01	136.98	1.350	3.400	351.36	350.87	3.395
1.375	139.57	138.88	1.368	3.500	362.18	357.14	3.453
1.400	142.12	141.84	1.397	3.600	373.03	370.37	3.575
1.450	147.23	147.05	1.448	3.700	383.94	377.35	3.640
1.500	152.35	151.51	1.492	3.800	394.89	392.15	3.775
1.550	157.47	156.25	1.538	3.900	405.90	400.00	3.852
1.600	162.62	162.60	1.600	4.000	416.93	416.66	3.998
1.650	167.73	166.66	1.640	4.100	428.03	425.53	4.078
1.700	172.86	172.41	1.696	4.200	439.19	434.78	4.161
1.750	178.00	176.99	1.740				

表3.2 中性母音資料のフォルマント周波数,フォルマント帯域幅

F1 B1	F2 B2	F3 B3	F4 B4	F5 B5	F6 B6	F7 B7	F8 B8	F9 B9
3.00 *308.5*	8.02 *925.5*	11.39 *1542.5*	13.60 *2159.5*	15.14 *2776.5*	16.32 *3393.5*	17.27 *4010.5*	18.09 *4627.5*	18.82 *5244.5*
0.940 *100.0*	0.673 *100.0*	0.435 *100.0*	0.294 *100.0*	0.215 *100.0*	0.170 *100.0*	0.142 *100.0*	0.124 *100.0*	0.111 *100.0*
5.00 *530.5*	11.60 *1591.5*	14.87 *2652.5*	16.84 *3713.5*	18.27 *4774.9*	19.44 *5835.5*	20.43 *6896.5*	21.25 *7957.5*	21.92 *9018.5*
0.856 *100.0*	0.421 *100.0*	0.228 *100.0*	0.154 *100.0*	0.121 *100.0*	0.101 *100.0*	0.085 *100.0*	0.070 *100.0*	0.057 *100.0*
8.00 *922.2*	15.12 *2766.6*	18.07 *4611.0*	20.04 *6455.4*	21.48 *8299.8*	--------	--------	--------	--------
0.674 *100.0*	0.216 *100.0*	0.125 *100.0*	0.091 *100.0*	0.066 *100.0*	--------	--------	--------	--------

立字体:Bark単位
斜字体：Hz単位

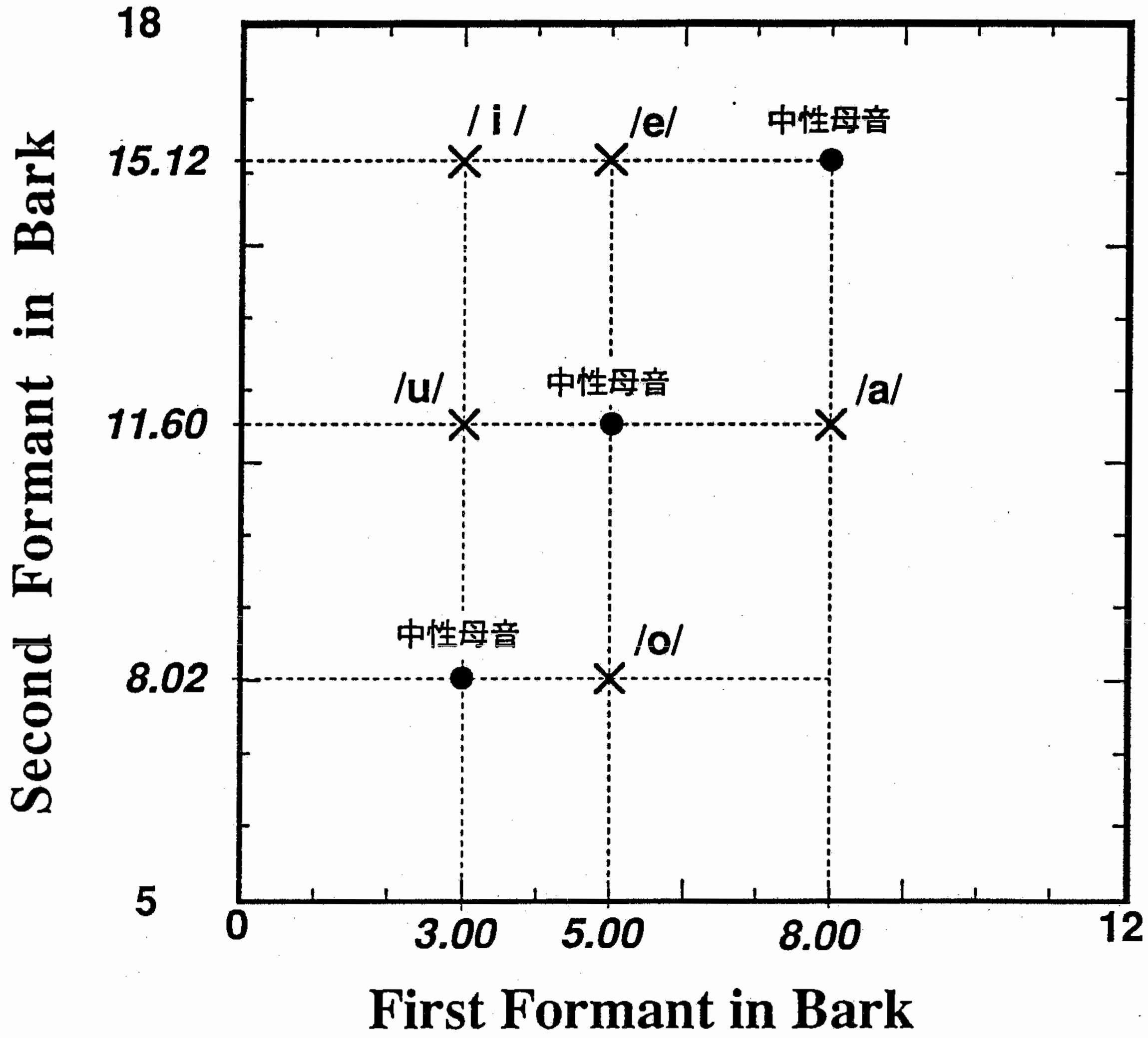


図3.1 資料の第１、第２フォルマント周波数

表3.3 5母音資料のフォルマント周波数,フォルマント帯域幅

	F1 B1	F2 B2	F3 B3	F4 B4	F5 B5	F6 B6	F7 B7
/i/	3.00 *308.5*	15.12 *2766.6*	16.23 *3343.0*	18.16 *4680.2*	19.62 *6017.4*	20.80 *7354.6*	21.73 *8691.8*
	0.470 *50.0*	0.216 *100.0*	0.259 *150.0*	0.184 *150.0*	0.147 *150.0*	0.117 *150.0*	0.091 *150.0*
/e/	5.00 *530.5*	15.12 *2766.6*	16.23 *3343.0*	18.16 *4680.2*	19.62 *6017.4*	20.80 *7354.6*	21.73 *8691.8*
	0.428 *50.0*	0.216 *100.0*	0.259 *150.0*	0.184 *150.0*	0.147 *150.0*	0.117 *150.0*	0.091 *150.0*
/a/	8.00 *922.2*	11.60 *1591.5*	16.23 *3343.0*	18.16 *4680.2*	19.62 *6017.4*	20.80 *7354.6*	21.73 *8691.8*
	0.674 *100.0*	0.421 *100.0*	0.259 *150.0*	0.184 *150.0*	0.147 *150.0*	0.117 *150.0*	0.091 *150.0*
/o/	5.00 *530.5*	8.02 *925.5*	16.23 *3343.0*	18.16 *4680.2*	19.62 *6017.4*	20.80 *7354.6*	21.73 *8691.8*
	0.428 *50.0*	0.673 *100.0*	0.259 *150.0*	0.184 *150.0*	0.147 *150.0*	0.117 *150.0*	0.091 *150.0*
/u/	3.00 *308.5*	11.60 *1591.5*	16.23 *3343.0*	18.16 *4680.2*	19.62 *6017.4*	20.80 *7354.6*	21.73 *8691.8*
	0.470 *50.0*	0.421 *100.0*	0.259 *150.0*	0.184 *150.0*	0.147 *150.0*	0.117 *150.0*	0.091 *150.0*

立字体:Bark単位
斜字体：Hz単位

4. 中性母音の分析結果

本章では、合成中性母音を資料として、分析を行なった結果を示す。

4.1 極推定の結果

本節では、極推定法を用いてフォルマント周波数、フォルマント帯域幅を推定する際に、分析パラメータと資料の合成パラメータとの関係が、推定値にどの様な影響を及ぼすかを調べた結果を示す。

4.1.1 分析窓の影響

本項では、用いる分析窓の長さおよび位置が、推定値にどの様な影響を及ぼすかを検討する。ここで、分析窓は窓長Wlのハニング窓とし、窓位置は図4.1.1のように、合成中性母音の中心近くにある波形の最大値を与える位置を0とした、ハニング窓の中心の位置Wpで表わす。

(i) 窓長

分析窓長がフォルマント推定精度に及ぼす影響を調べるため、3種類のフォルマント構造と4種類の基本周波数F0を持つ12種類の合成中性母音に対して、いくつかの異なった長さの分析窓を用いてフォルマント周波数とフォルマント帯域幅を推定し、それらの推定値の誤差が窓長によってどの様に変化するかを測定した。ここで、分析の次数(Np)は、合成中性母音の3種類のフォルマント構造のうち、第1フォルマントF1が3Barkあるいは5Barkのものに対してはNp=23に、F1が8Barkのものに対しては、Np=14に設定した。窓長は、Wp=0msec.に固定した。

図4.1.2～図4.1.4はフォルマント周波数Fiとフォルマント帯域幅Biの推定誤差の絶対値E(Fi)とE(Bi),(i=1,2,3,4)を分析窓長Wlを横軸にして描いたものである。図4.1.2はF1=3Bark、図4.1.3はF1=5Bark、図4.1.4はF1=8Barkの合成中性母音に対する結果である。各図とも左側の列がフォルマント周波数の推定誤差の絶対値を、右側の列がフォルマント帯域幅の推定誤差の絶対値を描いたものであり、左右の列とも上から順にF0が低いものから高いものへと並んでいる。すなわち、(a)(e)はF0=0.897Bark、(b)(f)はF0=1.79Bark、(c)(g)はF0=2.79Bark、(d)(h)はF0=4.00Barkの場合をそれぞれ示す。

まず、F1=3Barkの合成中性母音に対するフォルマント周波数の推定誤差について述べる。

F0が0.897Barkと低い場合には、図4.1.2(a)に示されるように、Wlを5msec～50msecの範囲で変化させてもE(Fi)(i=1,2,3,4)は常に0.07Bark以下で、推定誤差の絶対値は窓長の影響を受けない。しかし、F0が高くなると窓長の影響が現われる。

例えば、F0=1.79Barkの場合(図4.1.2(b))、Wl=10msec～50msec(E(F1)に対しては20msec～50msec)では推定誤差の絶対値は常に0.07Bark以下(E(F1)は0.22Bark)で、それらの変動幅も小さいが、Wlが5msecと短い場合には、いずれのE(Fi)の値も大きくなる。また、F0=2.79Barkの場合(図4.1.2(c))でも、Wlが10msec～50msecの範囲で

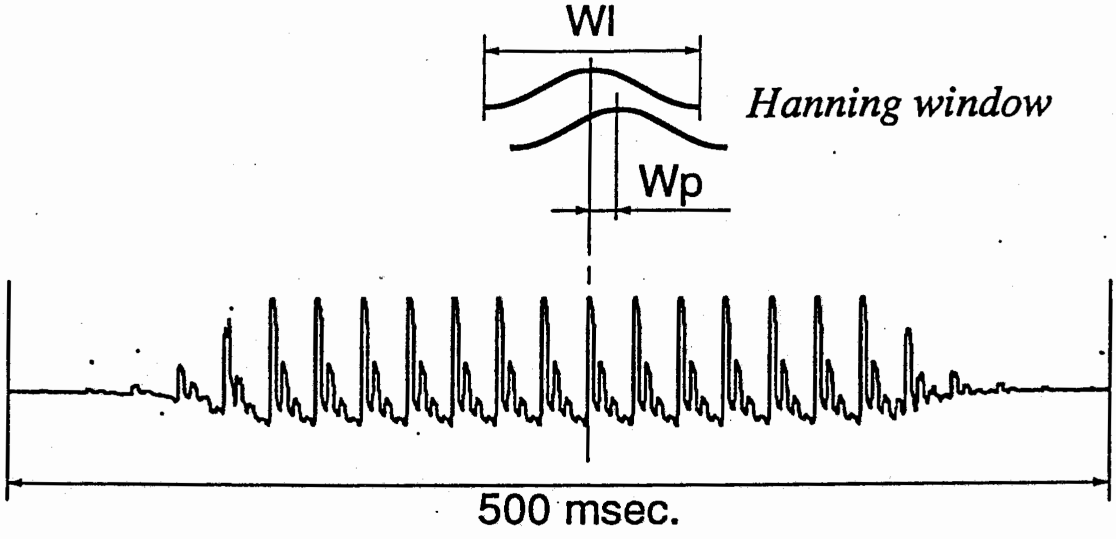


図 4.1.1　音声波形と分析窓長(Wl),分析窓位置(Wp)の関係

あれば、各E(Fi)は、絶対値はそれぞれに異なるが、変動幅はいずれも0.03Bark以下で、推定誤差は窓長の影響を受けていない。しかし、Wlが5msecと短くなると、第2フォルマントに対する結果(E(F2))を除いて、やはりいずれのE(Fi)も大きくなる。さらに、F0=4.00Barkの場合(図4.1.2(d))においても、Wl=10msec～50msecでは、E(Fi)の大きさが各々異なっている(E(F1)は1Bark以上にもなる)にもかかわらず、変動幅はいずれのE(Fi)でも0.03Barkと小さい。しかし、Wl=5msecに対しては、E(F1),E(F2),E(F4)はWlの長い場合よりも小さくなり、E(F3)は逆に大きくなっている。

このように、分析窓の長さが10msec～50msecと長い場合には、窓長にかかわらず、各F0に対して安定したフォルマント周波数の推定値が得られる。しかし、窓長が5msecと短い場合には、高いF0に対してのフォルマント周波数の推定値は、窓長が長い場合の安定した値とは異なり、誤差の絶対値が増加する場合も減少する場合もある。これは、窓かけによって切り出した分析区間と入力波形の周期との位相関係の影響が現われたためと考えられる。

次に、F1=3Barkの合成中性母音に対するフォルマント帯域幅の推定誤差について述べる。フォルマント帯域幅の推定においても、周波数の推定の場合と同様のことが言える。

F0が0.897Barkの場合には、図4.1.2(e)に示されるように、Wlを20msec～50msecの範囲で変化させてもE(Bi)(i=1,2,3,4)は常に0.14Bark以下で、その変動幅はいずれも0.05Bark以下と推定誤差の絶対値は窓長の影響を受けない。しかし、Wlが10msecさらに5msecと短くなるにつれて、E(Bi)はいずれも大きくなる。F0=1.79Bark～4.00Barkの場合(図4.1.2(f)～(h))でも、wlが20msec～50msecの範囲ではE(Bi)は常に0.5Bark以下で、その変動幅は小さいが、F0=0.897Barkの場合と同様にWlが短くなるにつれてE(Bi)は増加する。ただし、Wl=10msecに対してのみ、F0=1.79BarkのE(B2)、F0=1.79BarkのE(B2)、F0=2.79BarkのE(B1)、F0=4.00BarkのE(B1)は逆に減少している。

すなわち、フォルマント帯域幅の推定を行なう場合にも、分析窓の長さが20msec～50msecと長い場合には、窓長にかかわらず、各F0に対して安定した推定値が得られる。しかし、窓長が5msec,10msecと短い場合には、いずれのF0に対してもフォルマント帯域幅の推定値は窓長が長い場合の安定した値とは異なり、いくつかの例外を除いて推定値の絶対値は増加する。そして、この窓長が短い場合のフォルマント帯域幅の絶対値はF0に依存して大きさが異なる。このことは、フォルマント帯域幅の推定もフォルマント周波数の推定と同様に、分析区間と入力波形との位相関係に影響されることを示している。

このように、F1=3Barkの合成中性母音に対して極推定法でフォルマント周波数とフォルマント帯域幅を推定する場合、分析窓長はその推定値に影響を与えるが、窓長を20msecから50msecの範囲に設定すれば、各F0に対して安定した推定値が得られる。

同様にして、F1=5Bark,F1=8Barkの合成中性母音に対しては、図4.1.3、図4.1.4に示されるように、分析窓長が30msec～50msecの範囲であれば、各F0に対して安定したフォルマント周波数、フォルマント帯域幅の推定ができることがわかる。

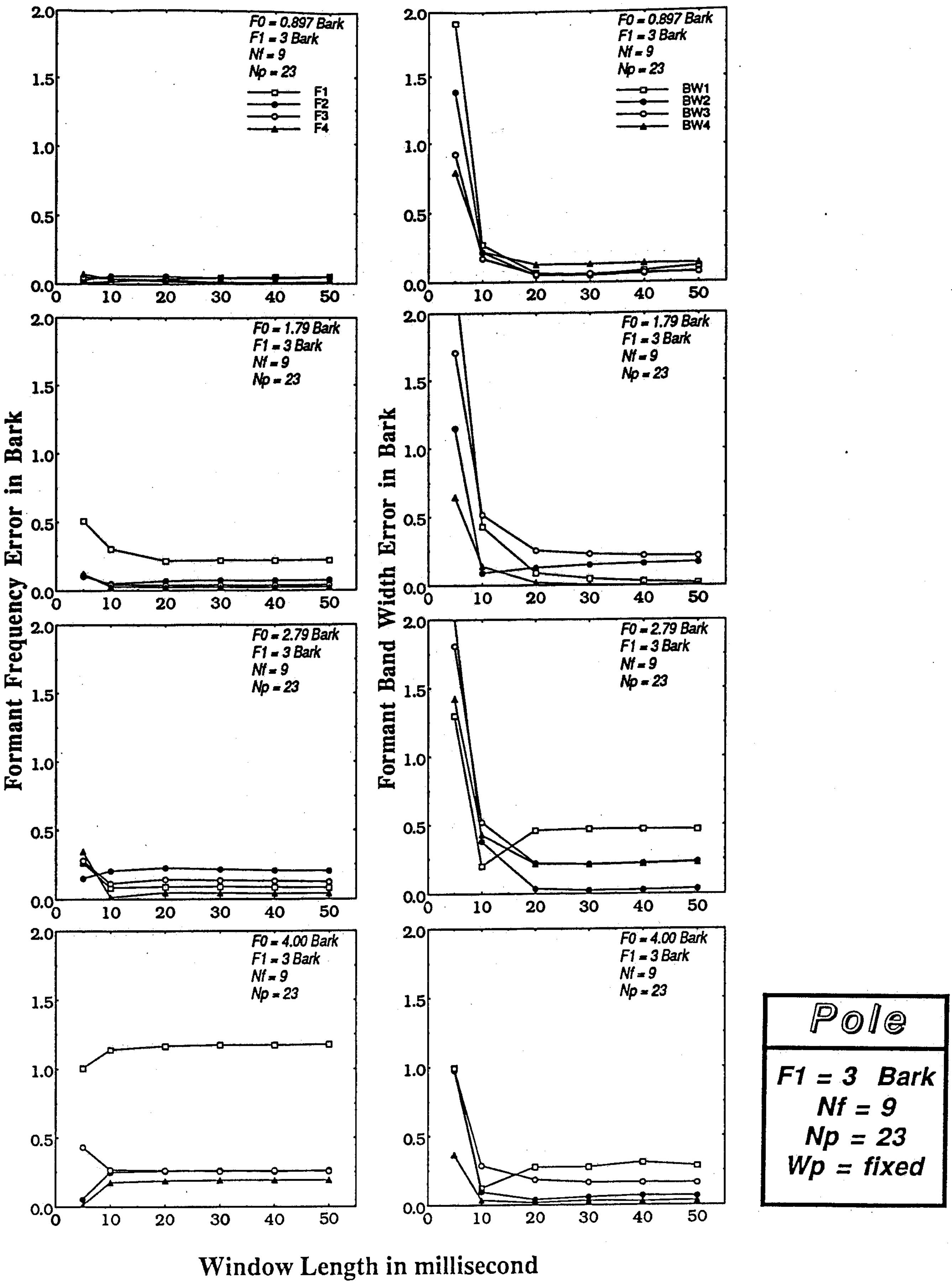


図4.1.2 フォルマント推定誤差の絶対値と分析窓長との関係

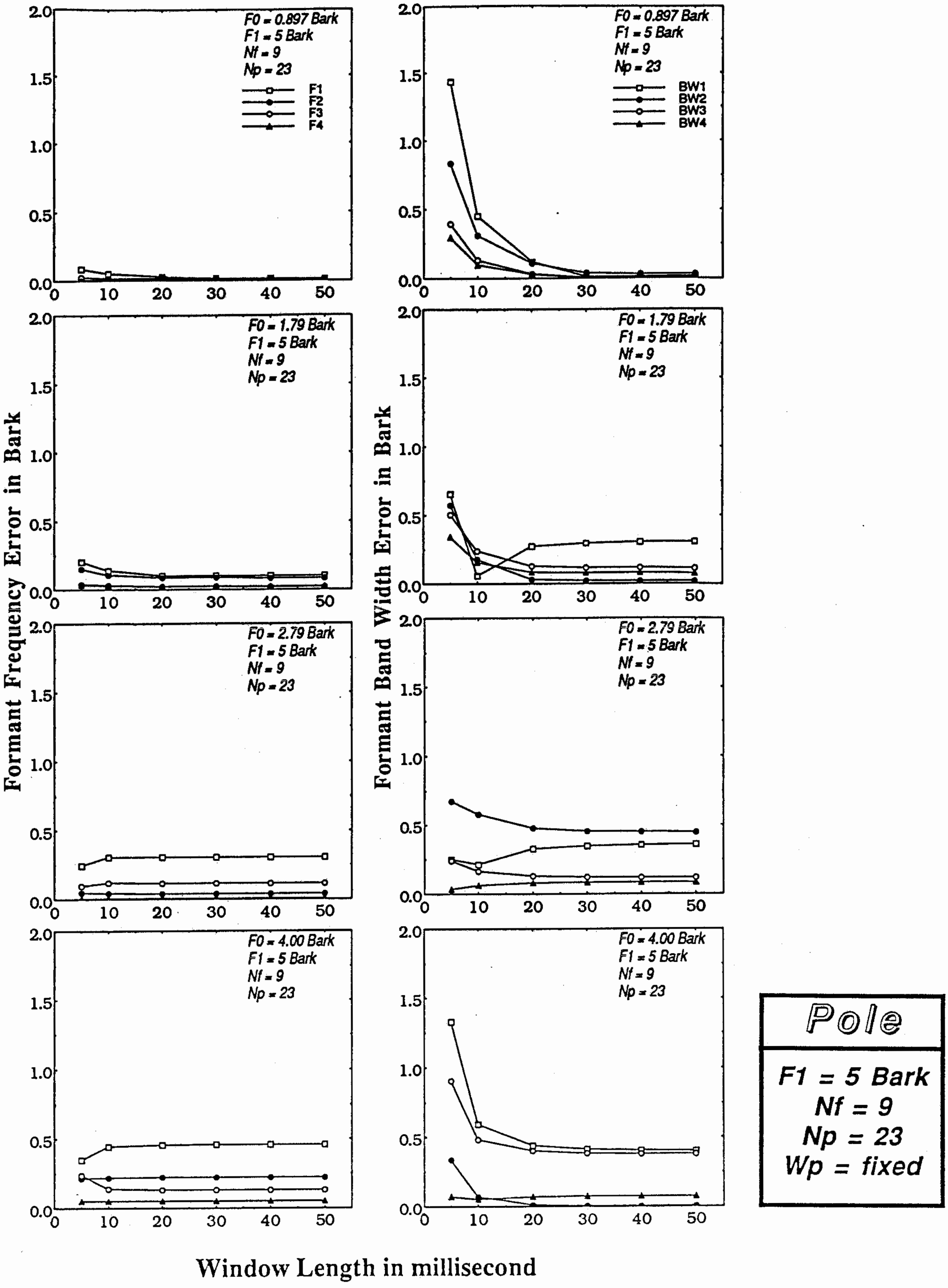


図4.1.3 フォルマント推定誤差の絶対値と分析窓長との関係

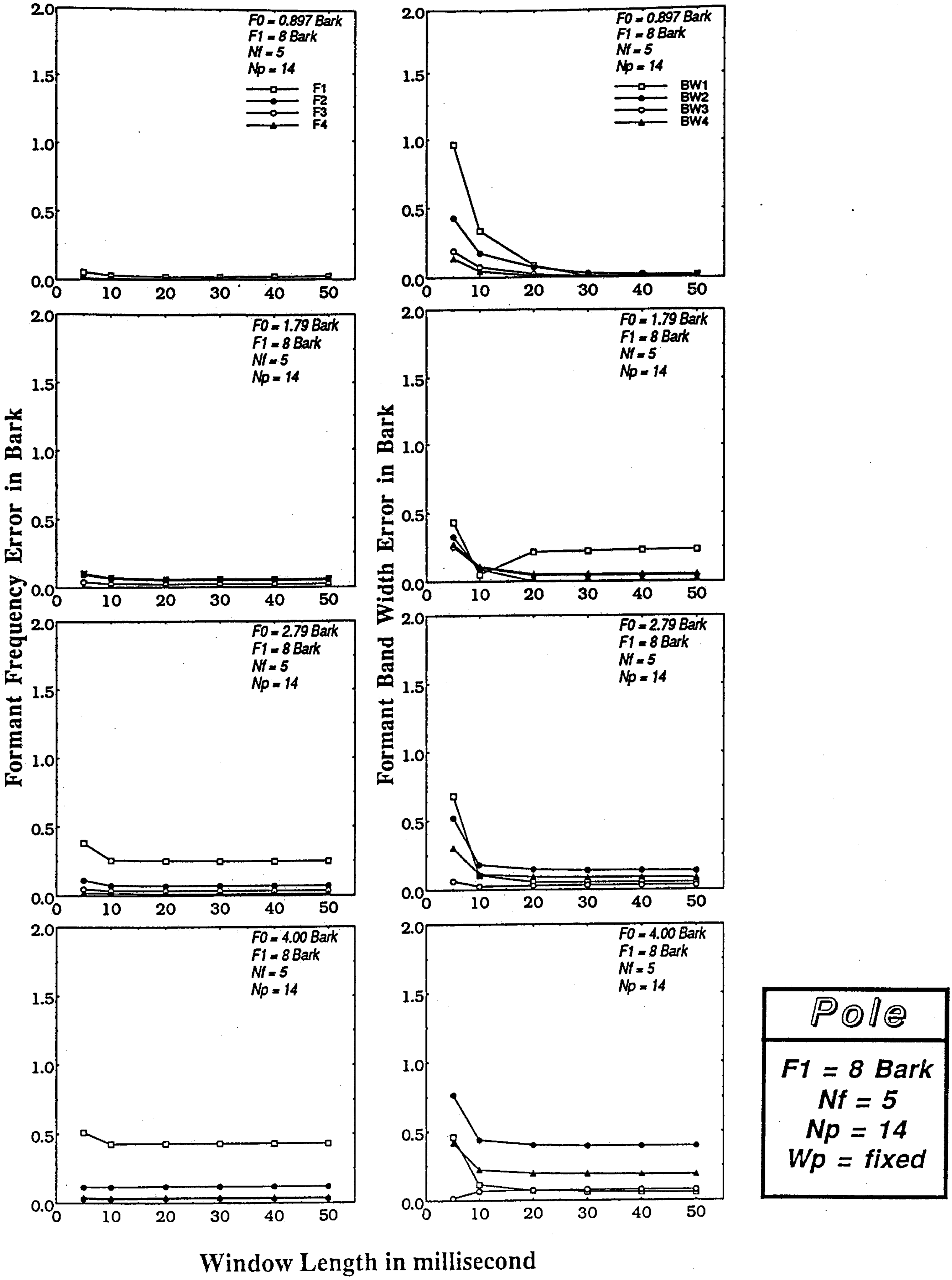


図4.1.4 フォルマント推定誤差の絶対値と分析窓長との関係

(ii) 窓位置

ここでは、安定した推定値が得られる窓長での分析において、窓の位置がフォルマント推定精度にどの様な影響を及ぼすかを検討する。

前項と同じ12種類の合成中性母音に対して、窓長を30msecに固定し、窓位置を0.5msecから0.6msecのステップで変化させながらフォルマント周波数とフォルマント帯域幅を推定し、それらの推定値の誤差が窓長によってどの様に変化するかを測定した。ここで、窓位置を変化させる範囲は、いずれのF0の合成母音に対してもその波形の一周期分より長くなるようにとり、また窓位置の変化のステップは、その整数倍が合成母音の1/F0にそれぞれ一致するように選んだ。窓長と窓位置以外の分析条件は、前項のものと同じである。

図4.1.8～図4.1.10はフォルマント周波数Fiとフォルマント帯域幅Biの推定誤差の絶対値E(Fi)とE(Bi),(i=1,2,3,4)を分析窓位置Wpを横軸にして描いたものである。図4.1.8はF1=3Bark、図4.1.9はF1=5Bark、図4.1.10はF1=8Barkの合成中性母音に対する結果である。各図とも左側の列がフォルマント周波数の推定誤差の絶対値を、右側の列がフォルマント帯域幅の推定誤差の絶対値を描いたものであり、左右の列とも上から順にF0が低いものから高いものへと並んでいる。すなわち、(a)(e)はF0=0.897Bark、(b)(f)はF0=1.79Bark、(c)(g)はF0=2.79Bark、(d)(h)はF0=4.00Barkの場合をそれぞれ示す。

まず、F1=3Barkの合成中性母音に対するフォルマント周波数の推定誤差について述べる。

F0が0.897Barkの場合には、図4.1.8(a)に示されるように、Wpを0msec～15msecの範囲で変化させてもE(Fi)(i=1,2,3,4)は常に0.07Bark以下で、推定誤差の絶対値は窓位置の影響を受けない。F0が1.79Bark～4.00Barkの場合(図4.1.8(b)～(c))、推定誤差の絶対値は、F0=1.79BarkでのE(F1)、F0=2.79BarkでのE(F2)、F0=4.00BarkでのE(F1)のように大きな値をとるものもあるが、変動幅はいずれのE(Fi)でも0.01Bark以下であり、やはり窓位置の影響を受けない。

このように、分析窓の長さが30msecの場合には、各F0に対して窓位置に依存しない安定したフォルマント周波数の推定値が得られることがわかる。

次に、F1=3Barkの合成中性母音に対するフォルマント帯域幅の推定誤差について述べる。フォルマント帯域幅の推定においても、周波数の推定の場合と同様のことが言える。

F0が0.897Barkの場合には、図4.1.8(e)に示されるように、Wpを0msec～15msecの範囲で変化させてもE(Bi)(i=1,2,3,4)は常に0.16Bark以下で、その変動幅も小さく推定誤差の絶対値は窓位置の影響を受けない。F0が1.79Bark～4.00Barkの場合(図4.1.8(f)～(h))でも同様で、E(Bi)は窓位置の変化にかかわらず常に0.5Bark以下で、その変動幅も小さい。

すなわち、フォルマント帯域幅の推定を行なう場合にも、分析窓の長さが30msecの場合には、窓位置にかかわらず各F0に対して安定した推定値が得られる。

このように、F1=3Barkの合成中性母音に対して、分析窓長30msecの極推定法でフォルマント周波数とフォルマント帯域幅を推定する場合、分析窓の位置にかかわらず各F0に対して安定した推定値が得られる。

なお、図4.1.8(e)(f)(g)で認められる窓位置の変化にともなうE(Bi)の増減は、合成母音の波形の周期(F0=0.897Bark,1.79Bark,2.79Bark,4.00Barkに対してそれぞれ

11msec,5.5msec,3.5msec,2.4msec)に同期したものではない。すなわち、分析区間と入力波形との位相関係の影響が現われたものではなく、合成過程で波形そのものに生じた非定常性のためと考えられる。

同様にして、F1=5Bark,F1=8Barkの合成中性母音に対しても、図4.1.9、図4.1.10に示されるように、分析窓長30msの極推定法によれば、各F0に対して窓位置に影響されない安定したフォルマント周波数、フォルマント帯域幅の推定ができることがわかる。

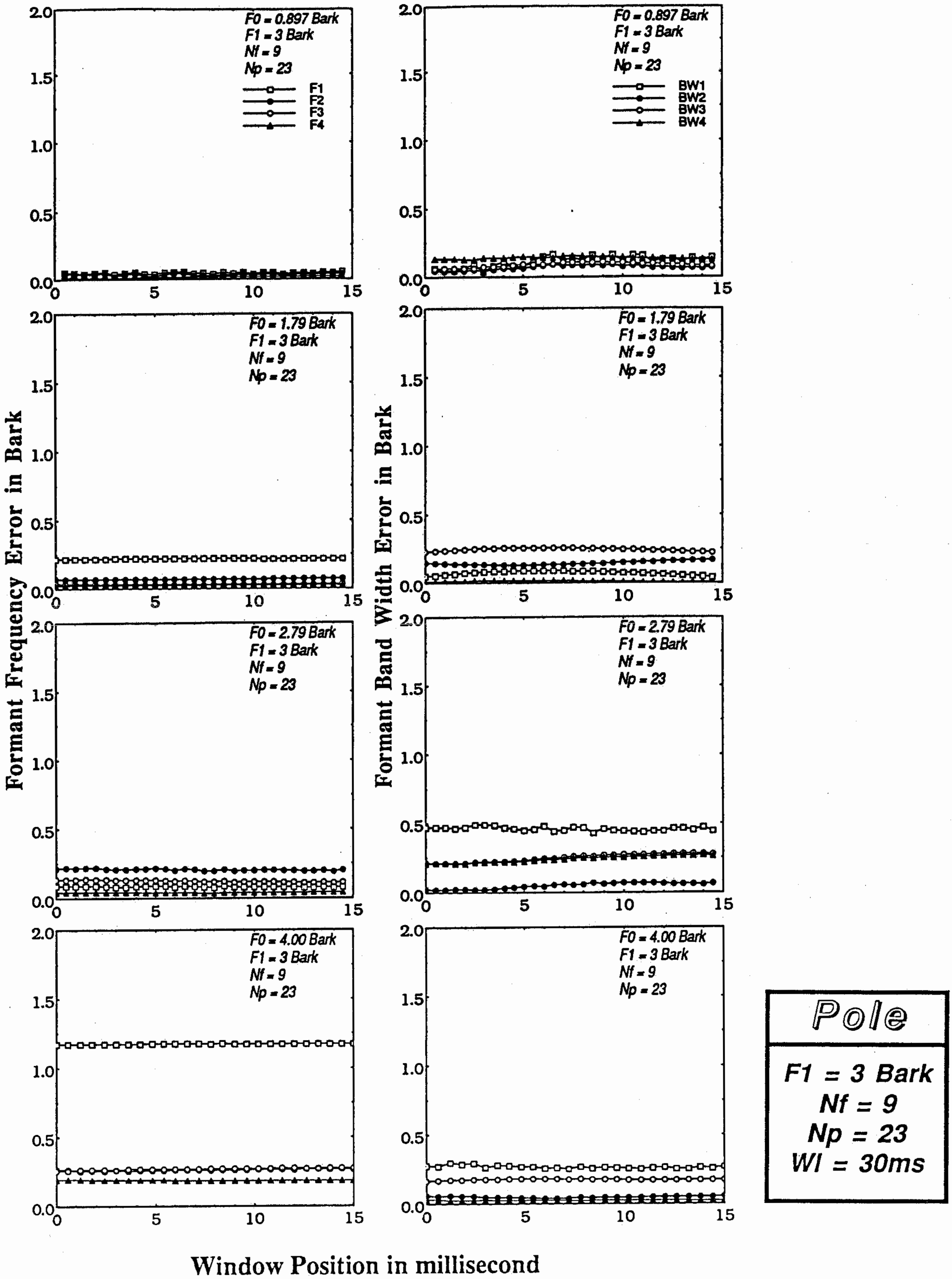


図4.1.5 フォルマント推定誤差の絶対値と分析窓位置との関係

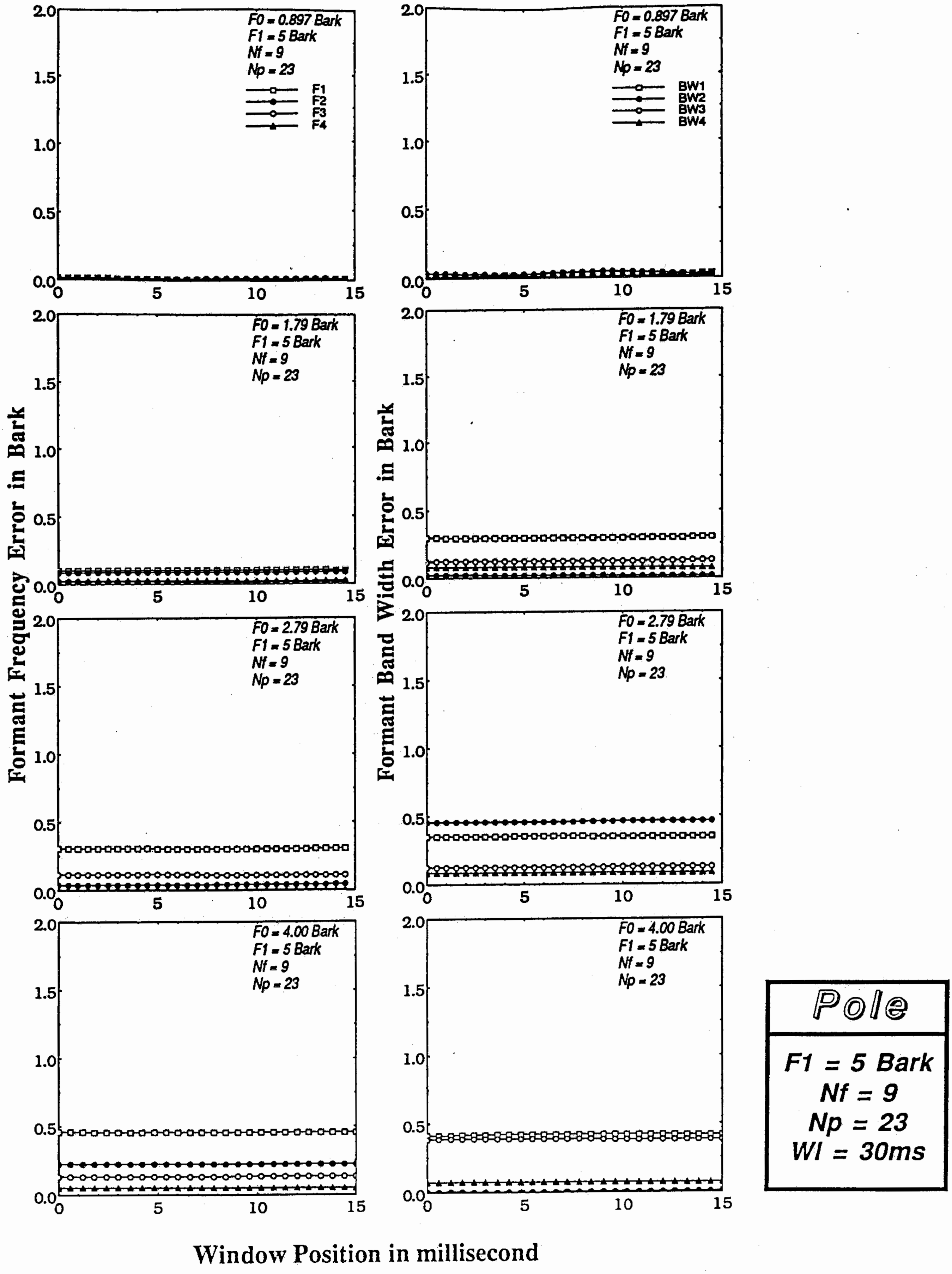


図4.1.6 フォルマント推定誤差の絶対値と分析窓位置との関係

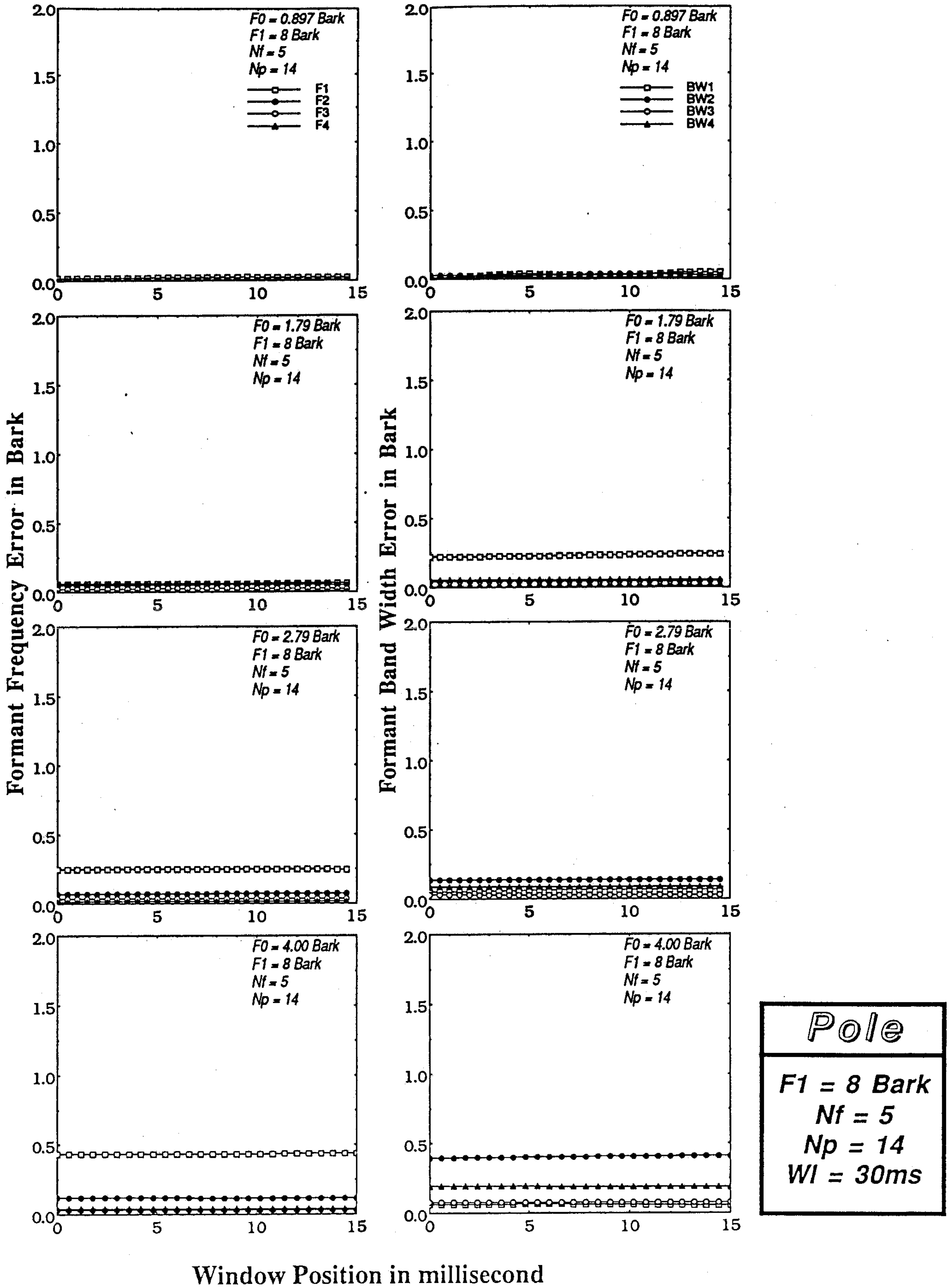


図4.1.7 フォルマント推定誤差の絶対値と分析窓位置との関係

4.1.2 分析次数の影響

本節では、分析の次数がフォルマント周波数、フォルマント帯域幅の推定値に及ぼす影響を調べた結果を示す。

図4.1.8(a)～図4.1.14(a)は、フォルマント周波数の推定誤差の絶対値を分析次数Npを横軸にして描いたものである。各図とも左上から右下へ資料のF0が高いものから低いものへと並んでいる。

図4.1.8(b)～図4.1.14(b)は、フォルマント帯域幅の推定誤差の絶対値について同様に求めたものである。

図4.1.8、図4.1.9、図4.1.10は、F1=3 Barkでフォルマントの個数Nfがそれぞれ5個、7個、9個の資料に対する結果である。

図4.1.11、図4.1.12、図4.1.13は、F1=5 BarkでNf=5、7、9の資料に対する結果である。

図4.1.14は、F1=8 BarkでNf=5の資料に対する結果である。

図4.1.15は、フォルマント周波数の推定誤差の絶対値がある閾値E以下であるような分析次数の上限と下限を、資料のF0を横軸にして描いたものである。この図ではE=0.5 Barkに設定した。第1段、第2段、第3段はそれぞれF1=3 Bark、5 Bark、8 Barkの資料に対する結果を示し、第1列、第2列、第3列はそれぞれNf=5、7、9の資料に対する結果を示す。

図4.1.16(a)(b)～図4.1.22(a)(b)は、資料のDFTパワスペクトルとαパラメータより求められたスペクトル包絡である。図中に、合成時のフォルマント周波数を矢印で、推定されたフォルマント周波数を垂線で示す。各図とも左の列がF1～F4の推定に成功した例（推定誤差の絶対値が0.5 Bark以下の場合）を、右の列がF1～F4のいずれかの推定に失敗した例（推定誤差の絶対値が0.5 Barkを超える場合）を示す。

また各図とも、(a)の右上は分析次数が大きすぎることによる失敗例を、(a)の右下は分析次数が小さすぎることによる失敗例を、そして、(b)の右は資料のF0が大きすぎることによる失敗例を示している。

図4.1.16、図4.1.17、図4.1.18は、F1=3 BarkでNf=5、7、9の資料に対する結果である。

図4.1.19、図4.1.20、図4.1.21は、F1=5 BarkでNf=5、7、9の資料に対する結果である。

図4.1.22は、F1=8 BarkでNf=5の資料に対する結果である。

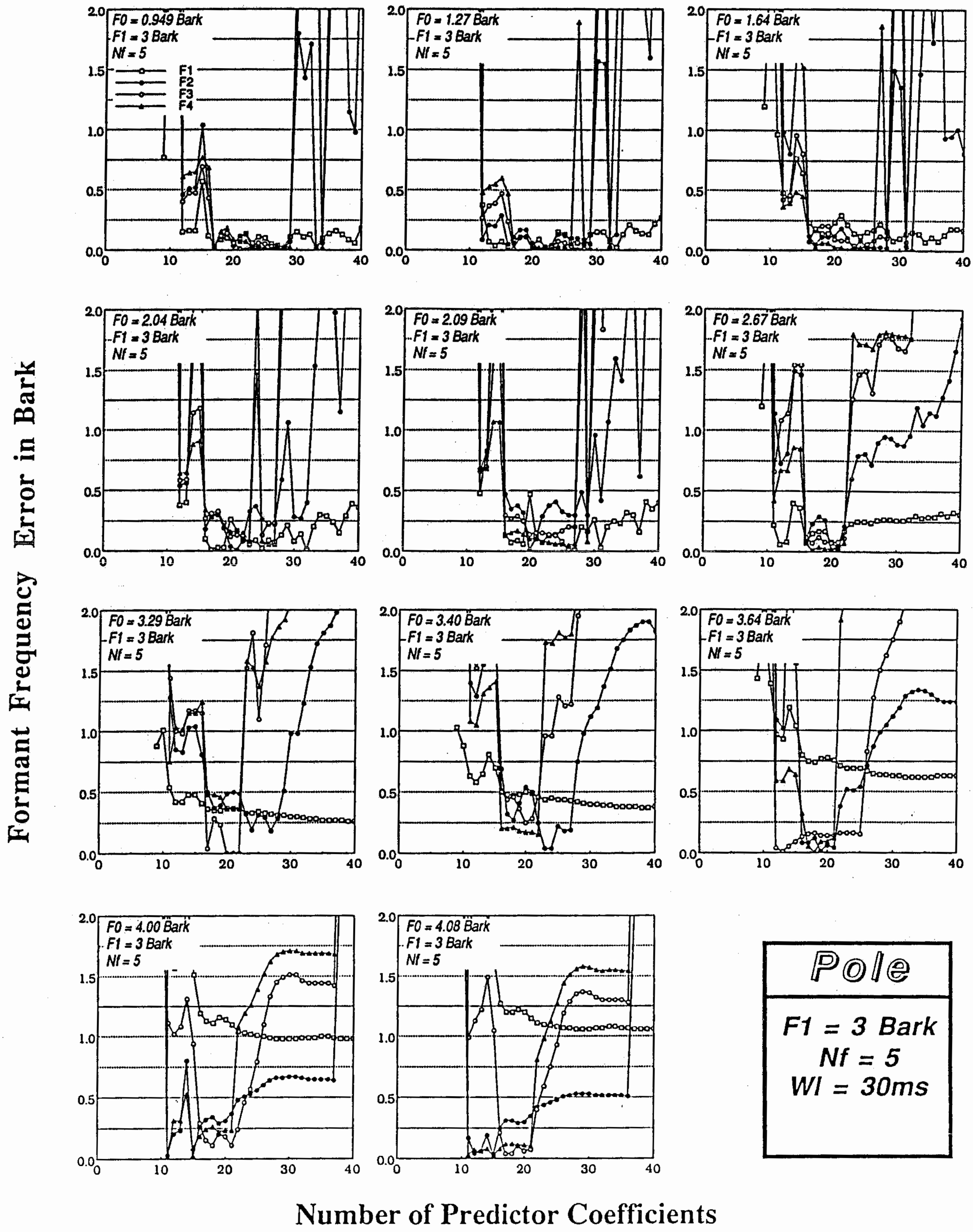


図4.1.8(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

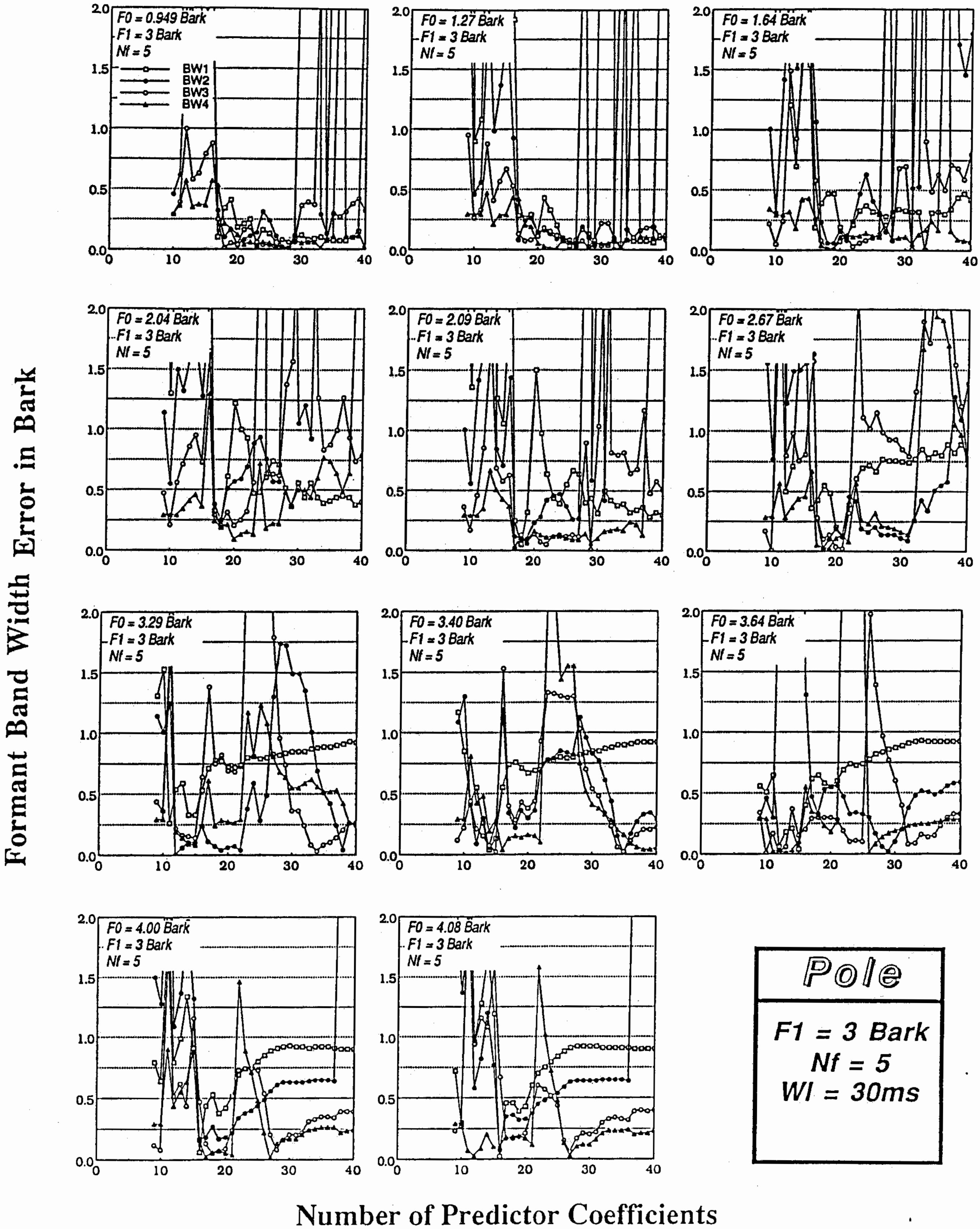


図4.1.8(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

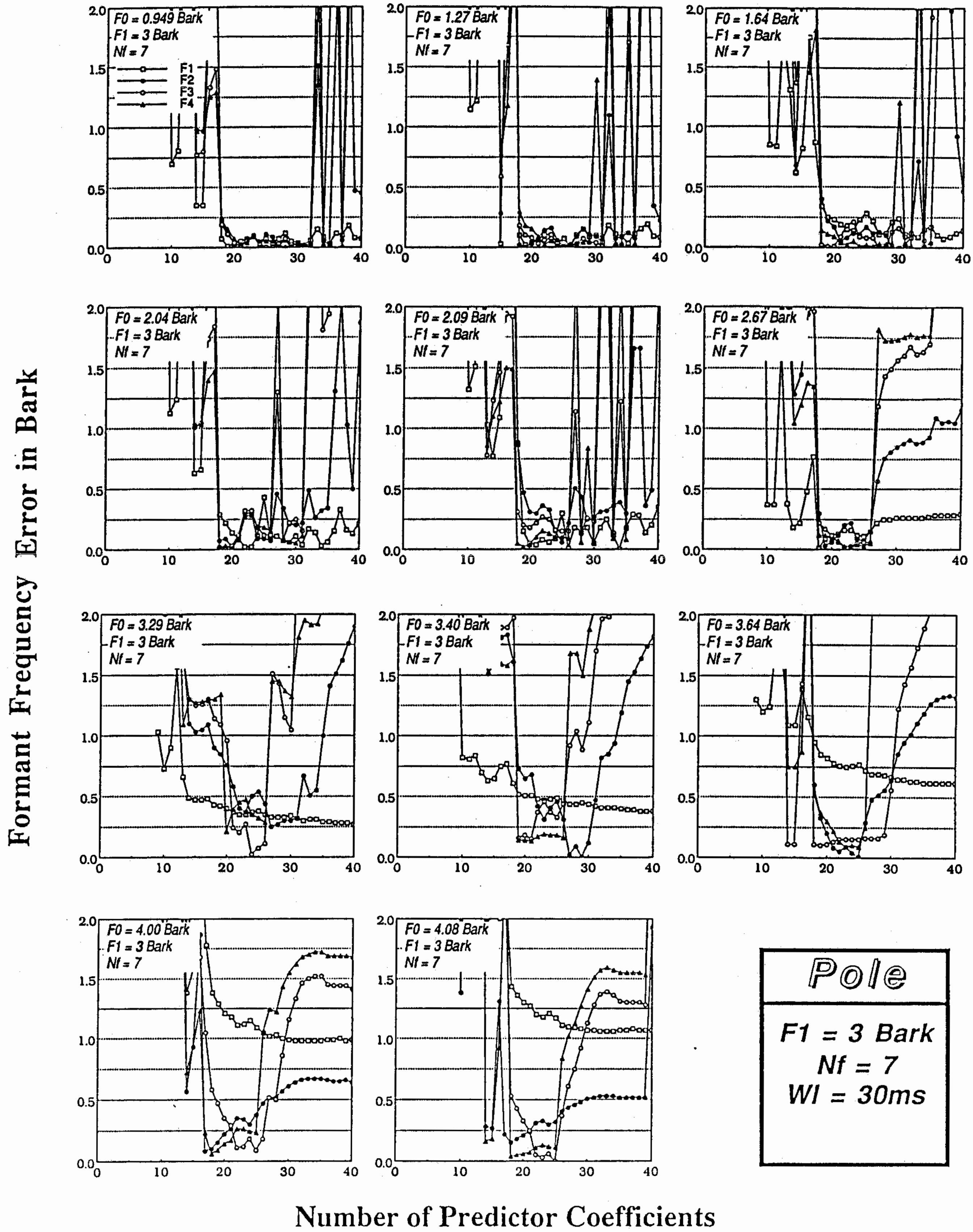


図4.1.9(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

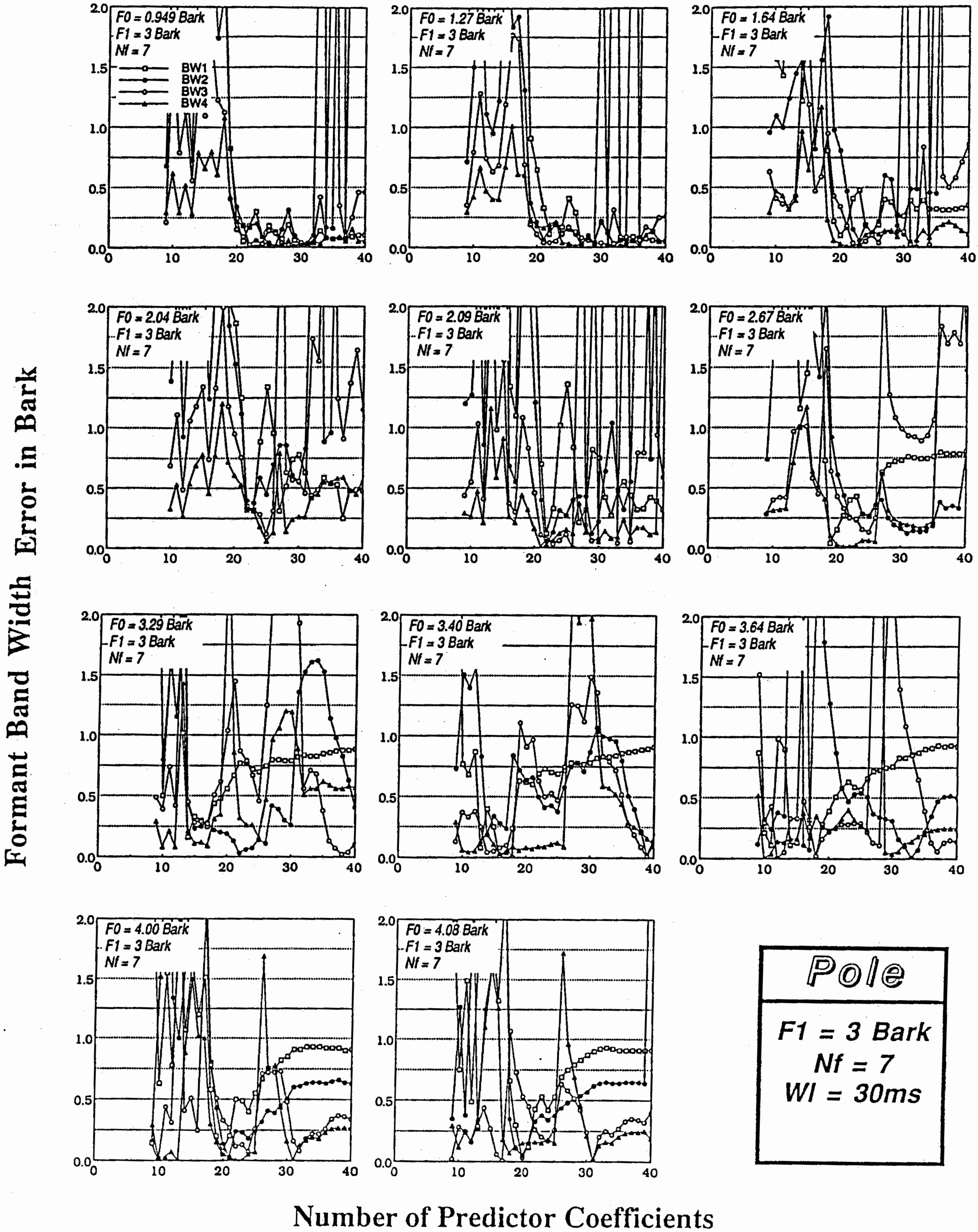


図4.1.9(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

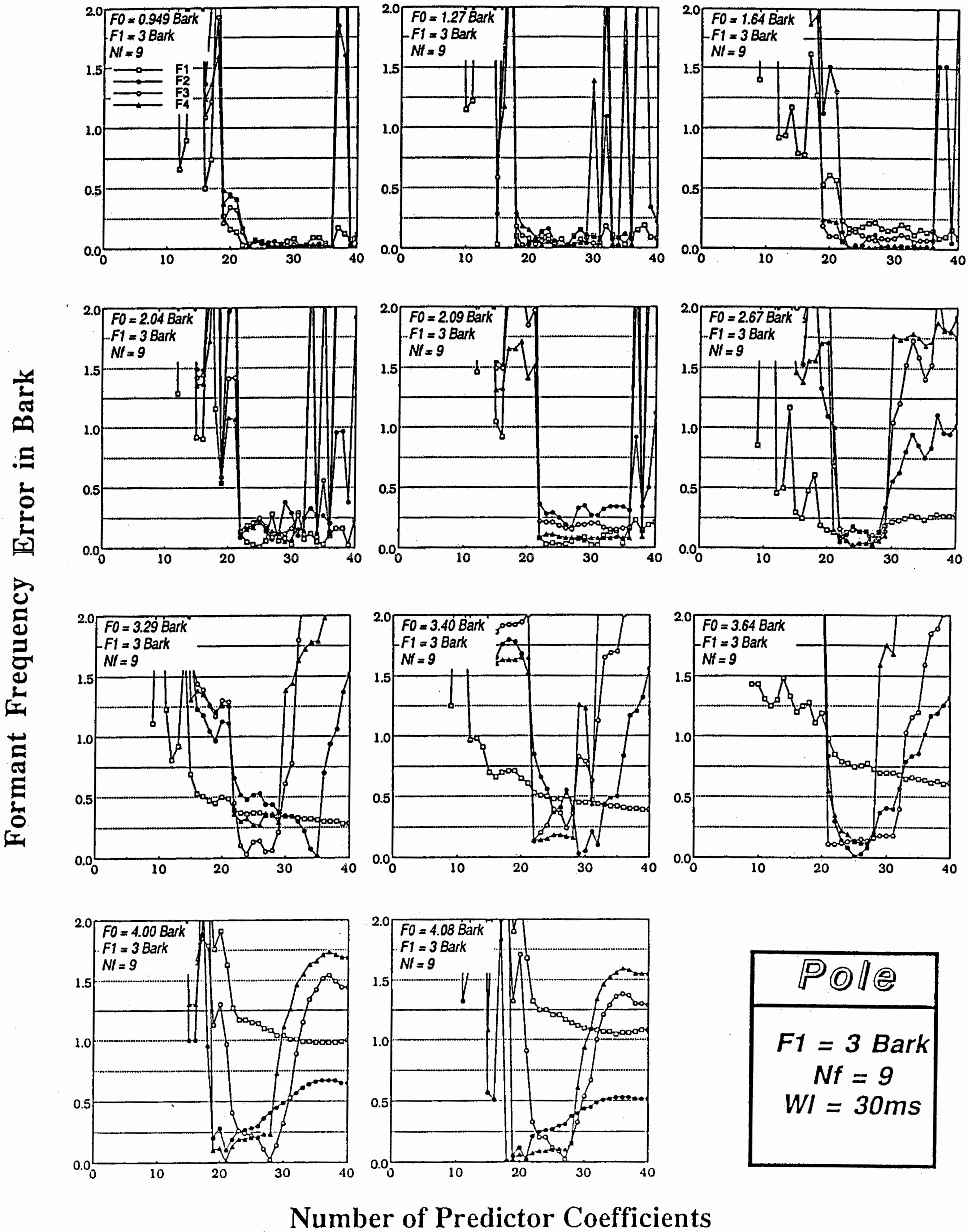


図4.1.10(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

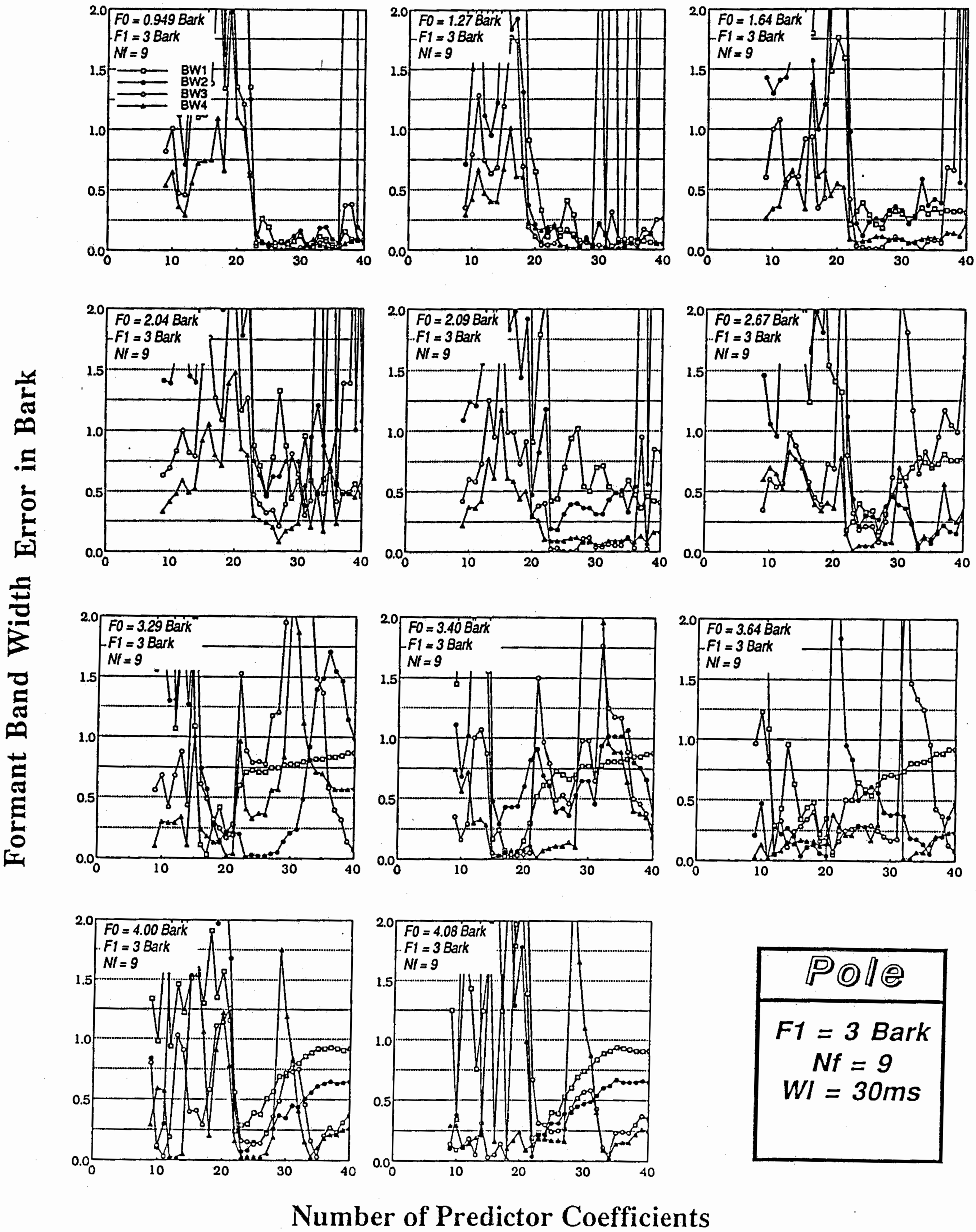


図4.1.10(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

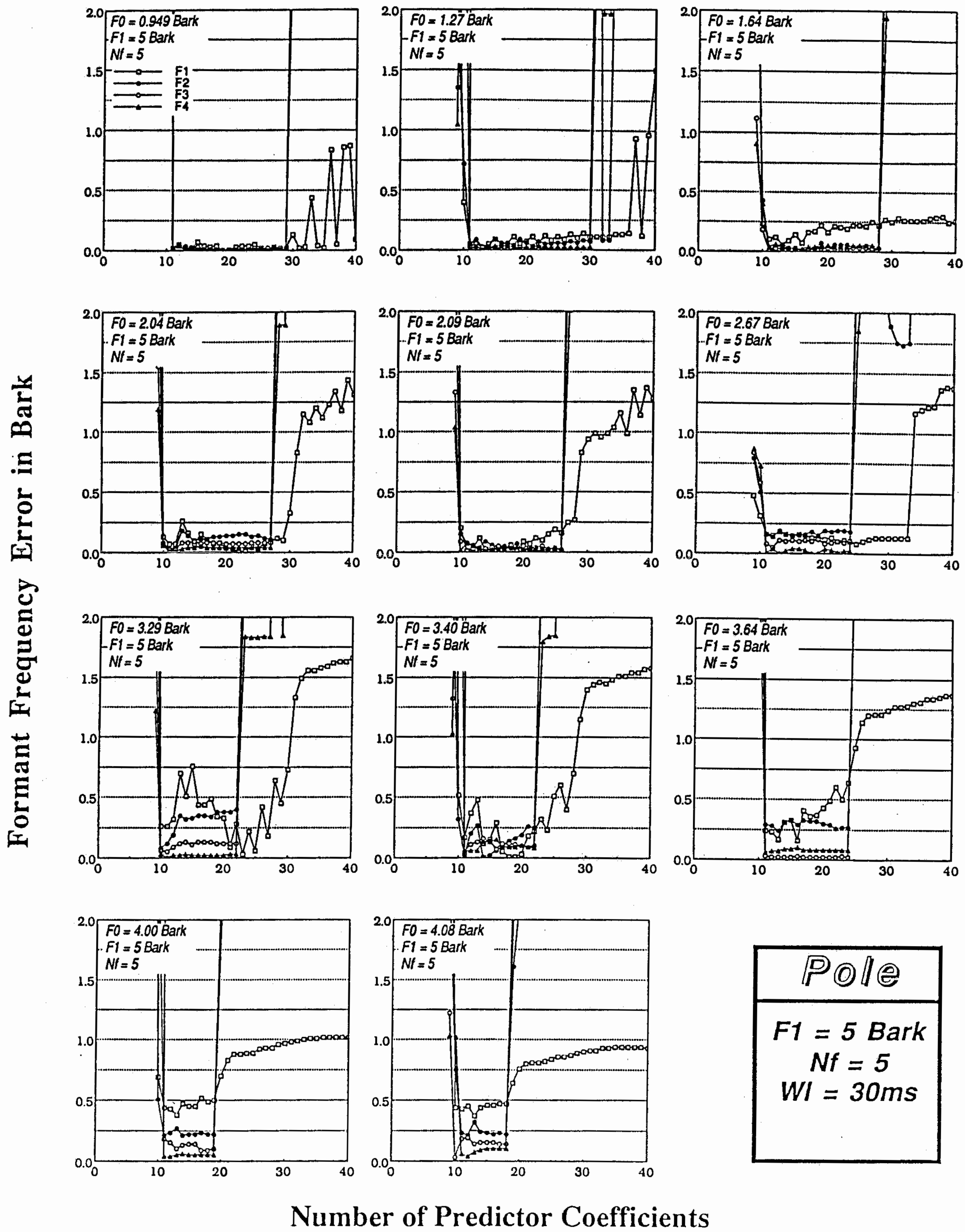


図4.1.11(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

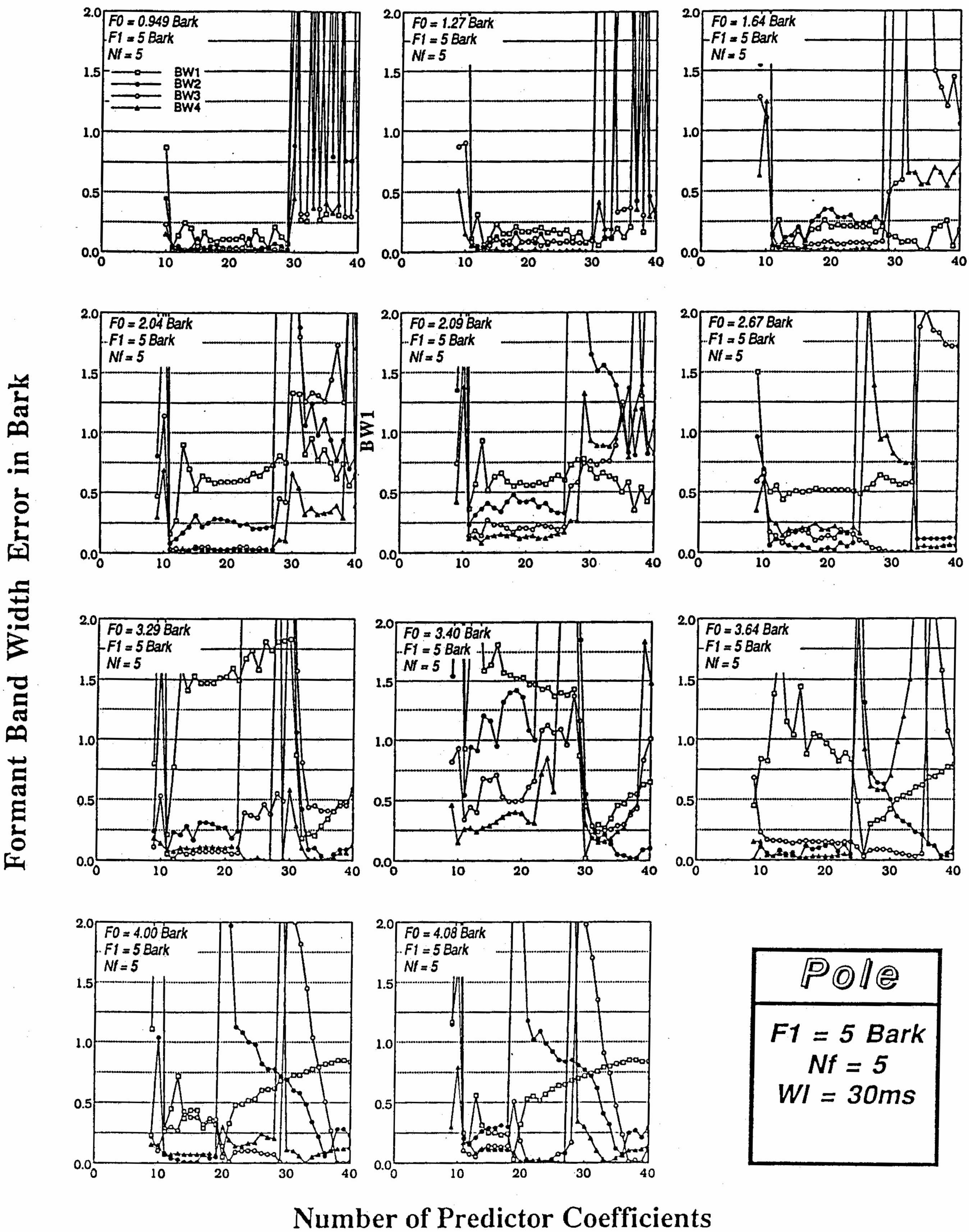


図4.1.11(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

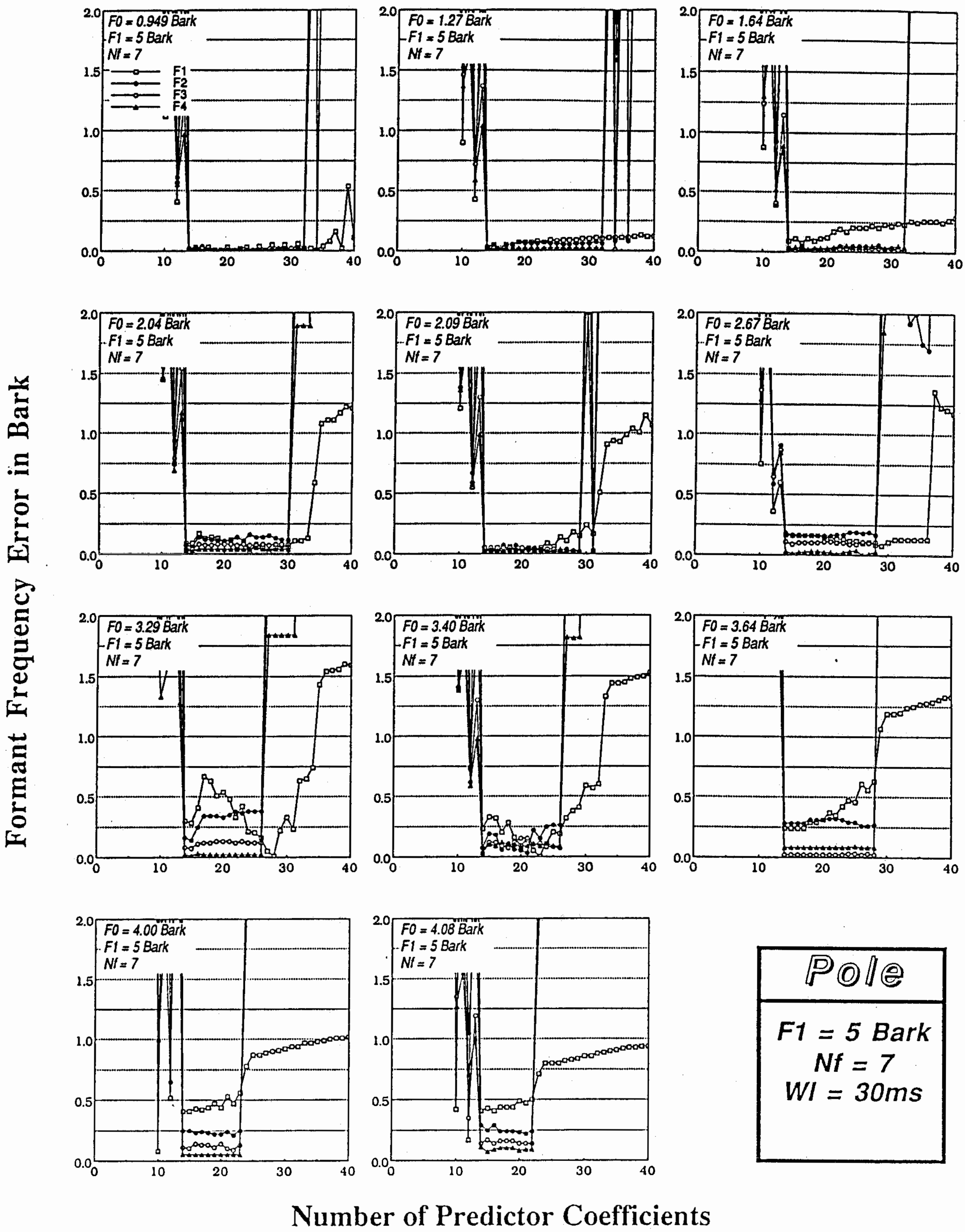


図4.1.12(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

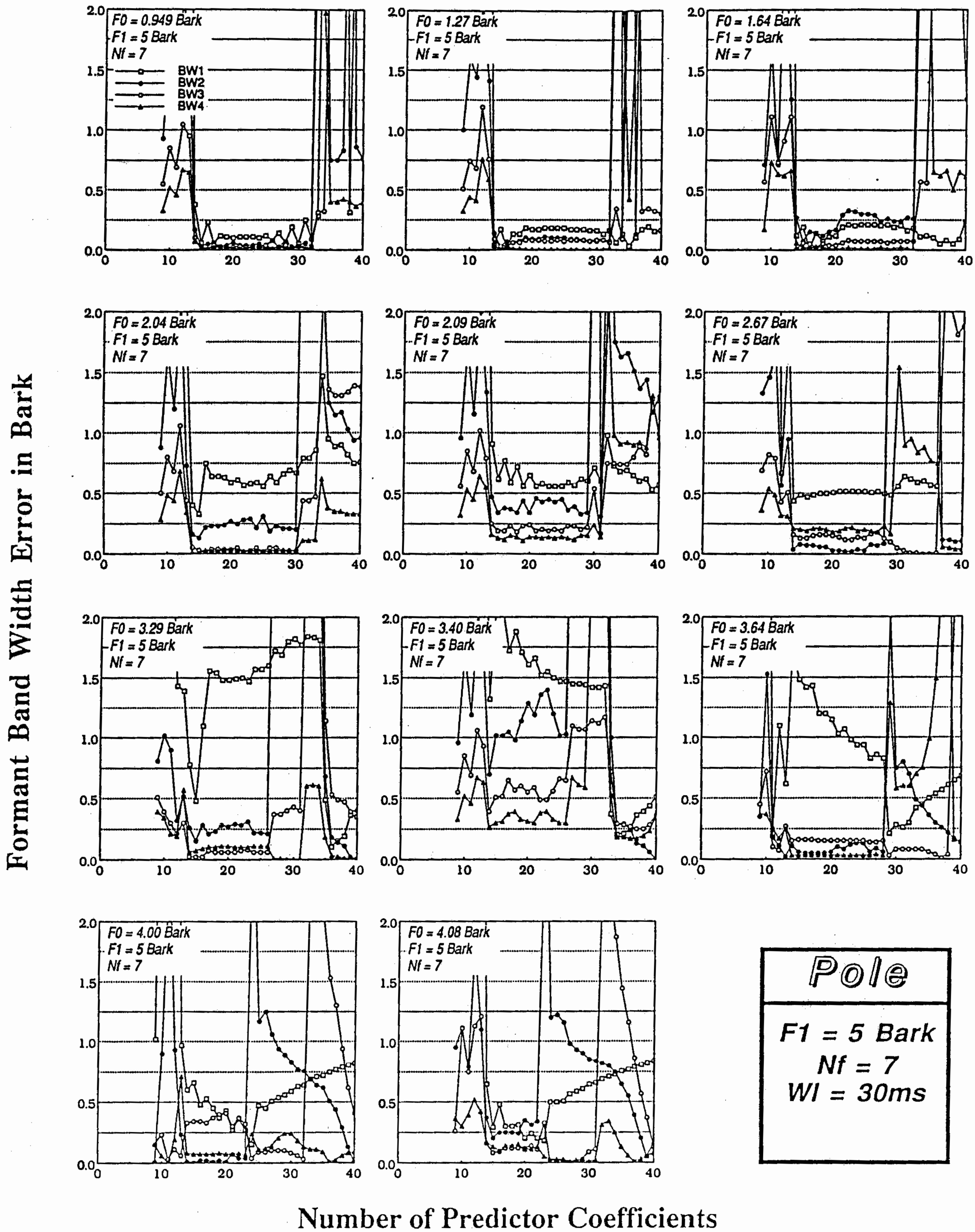


図4.1.12(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

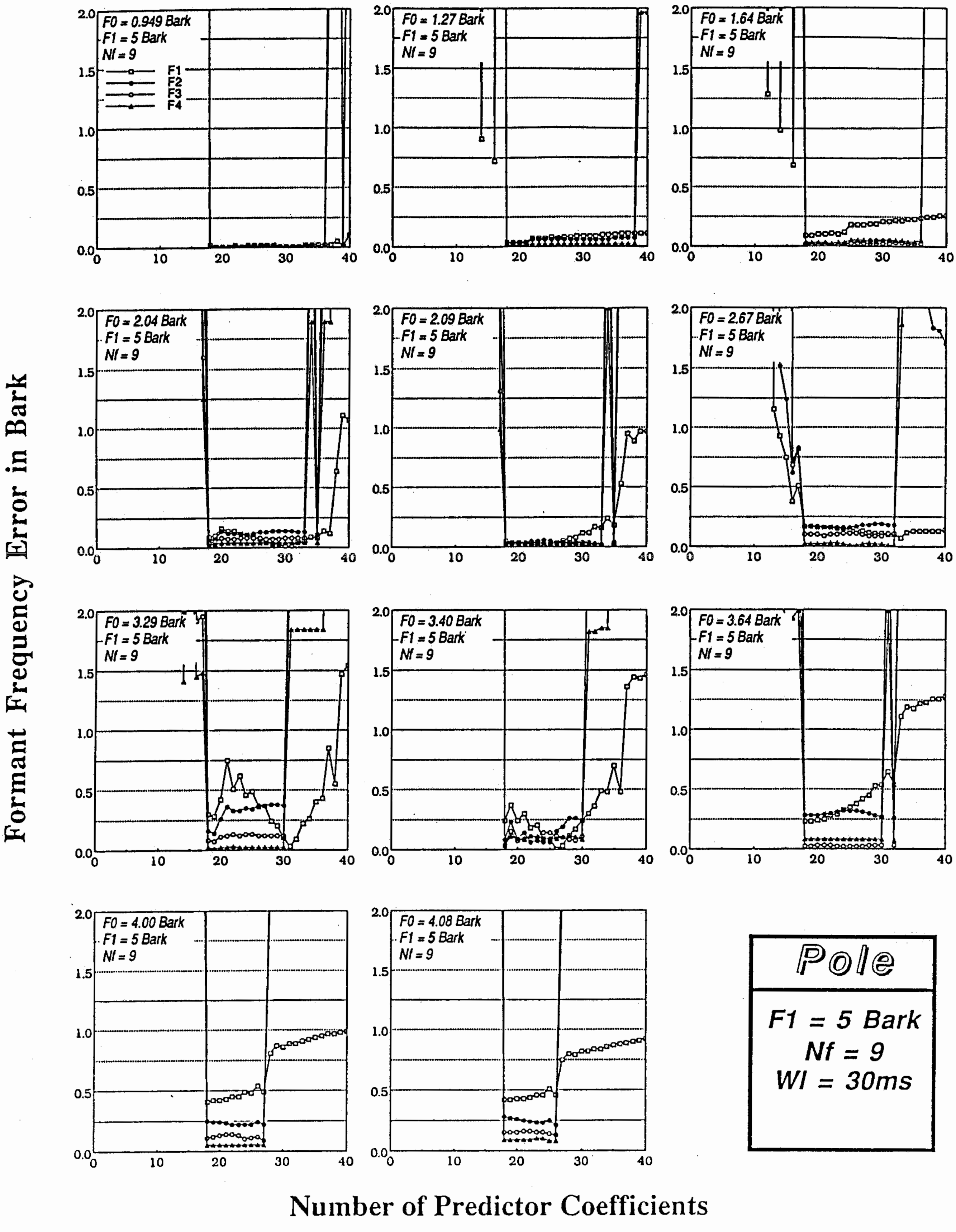


図4.1.13(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

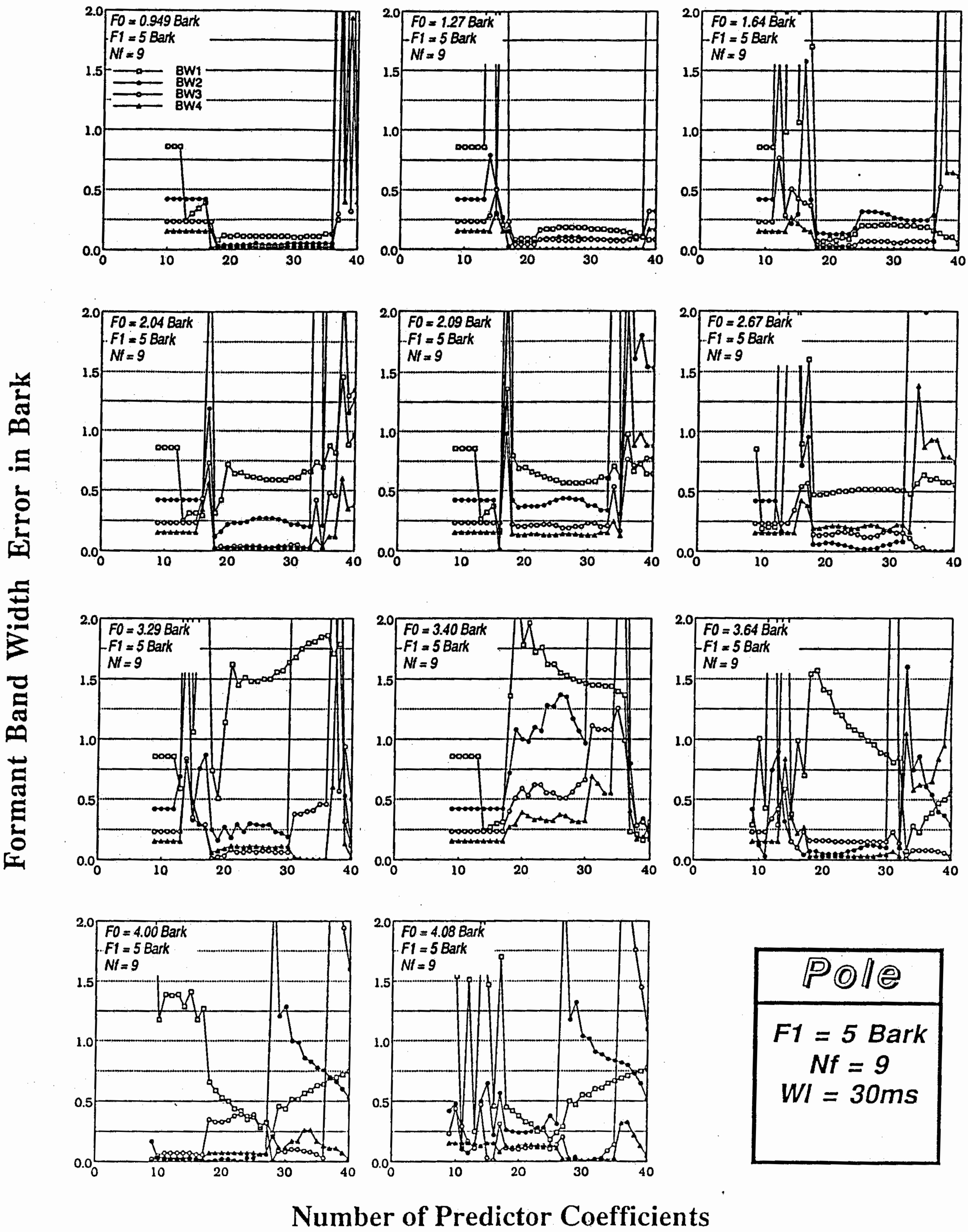


図4.1.13(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

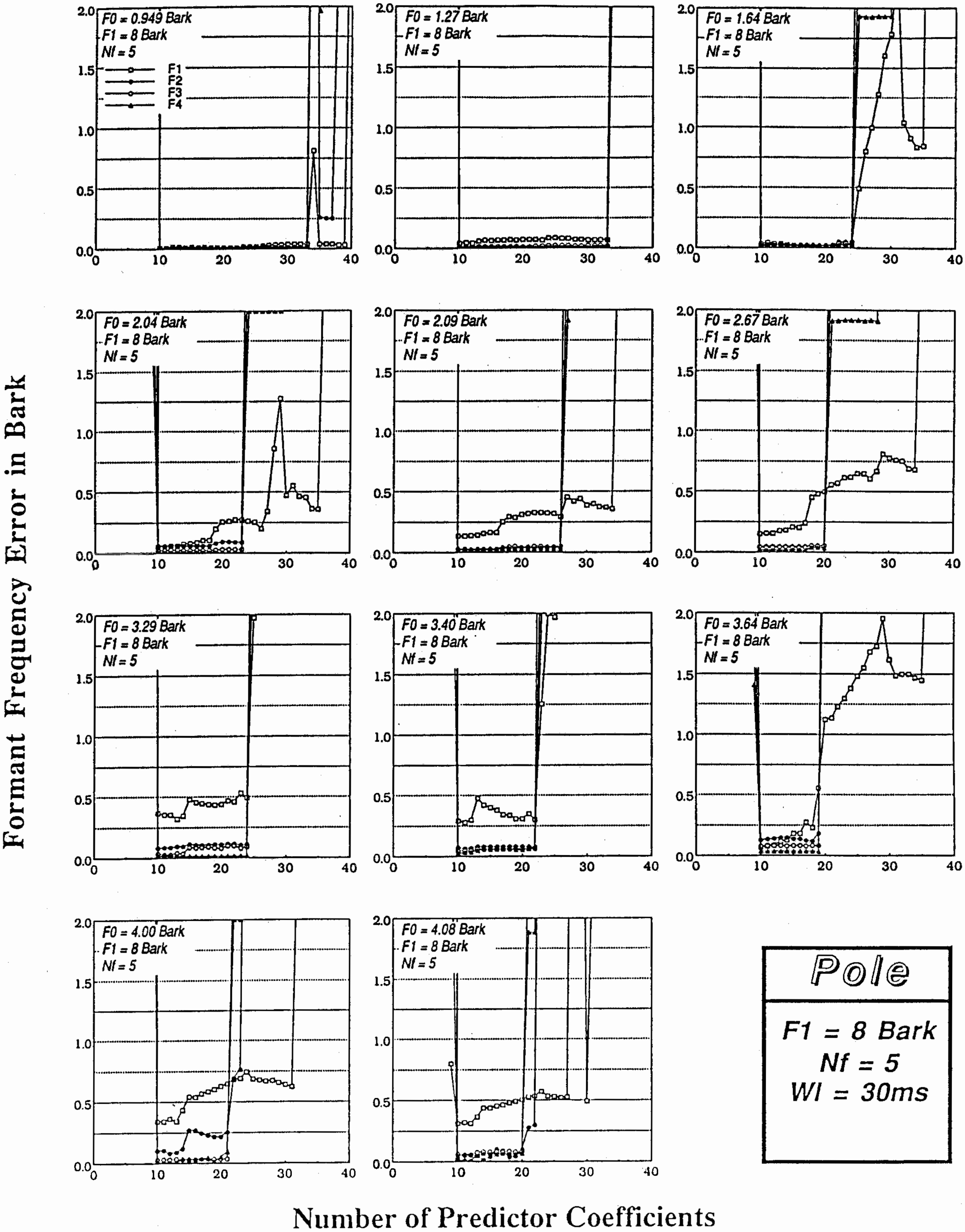


図4.1.14(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

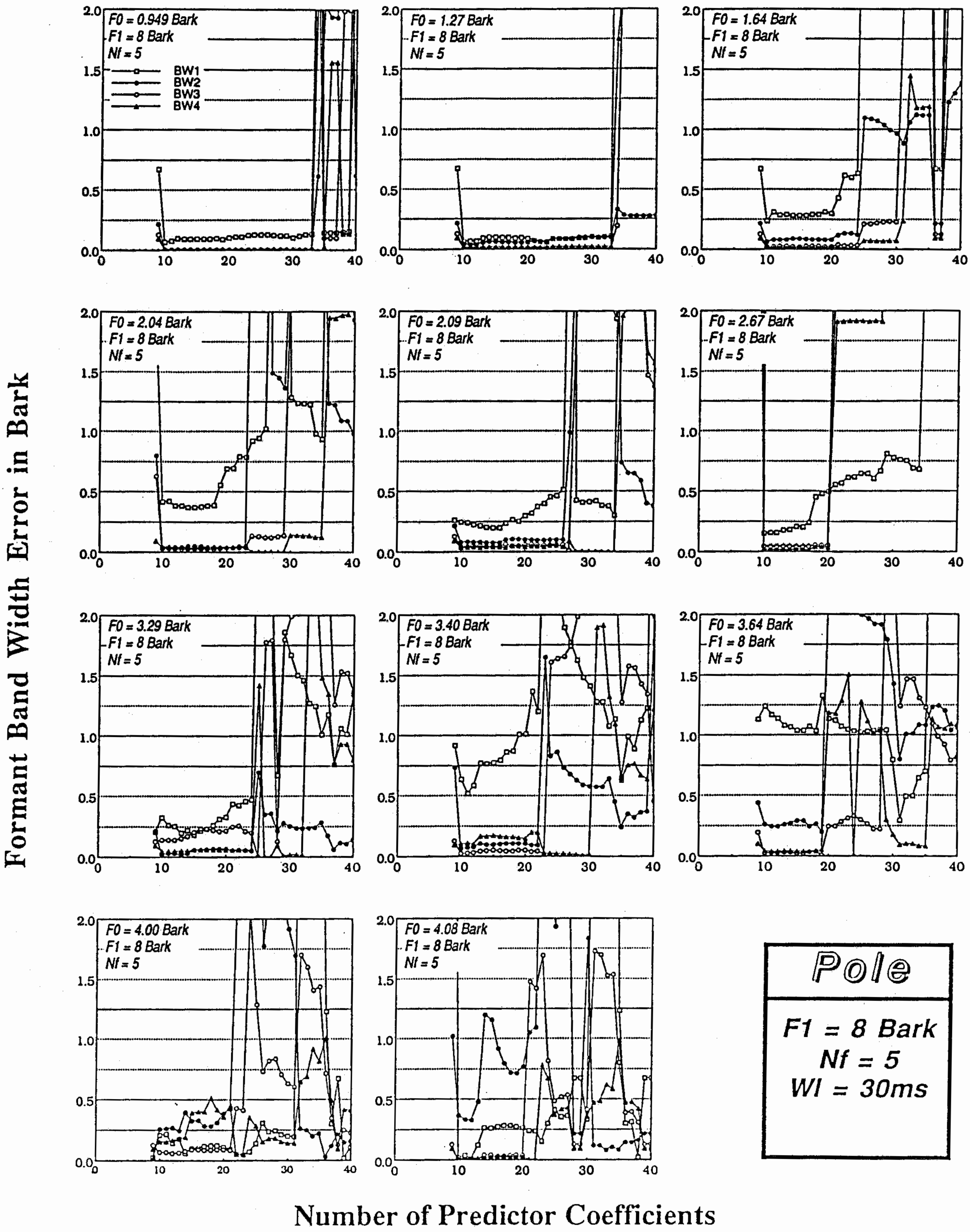


図4.1.14(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

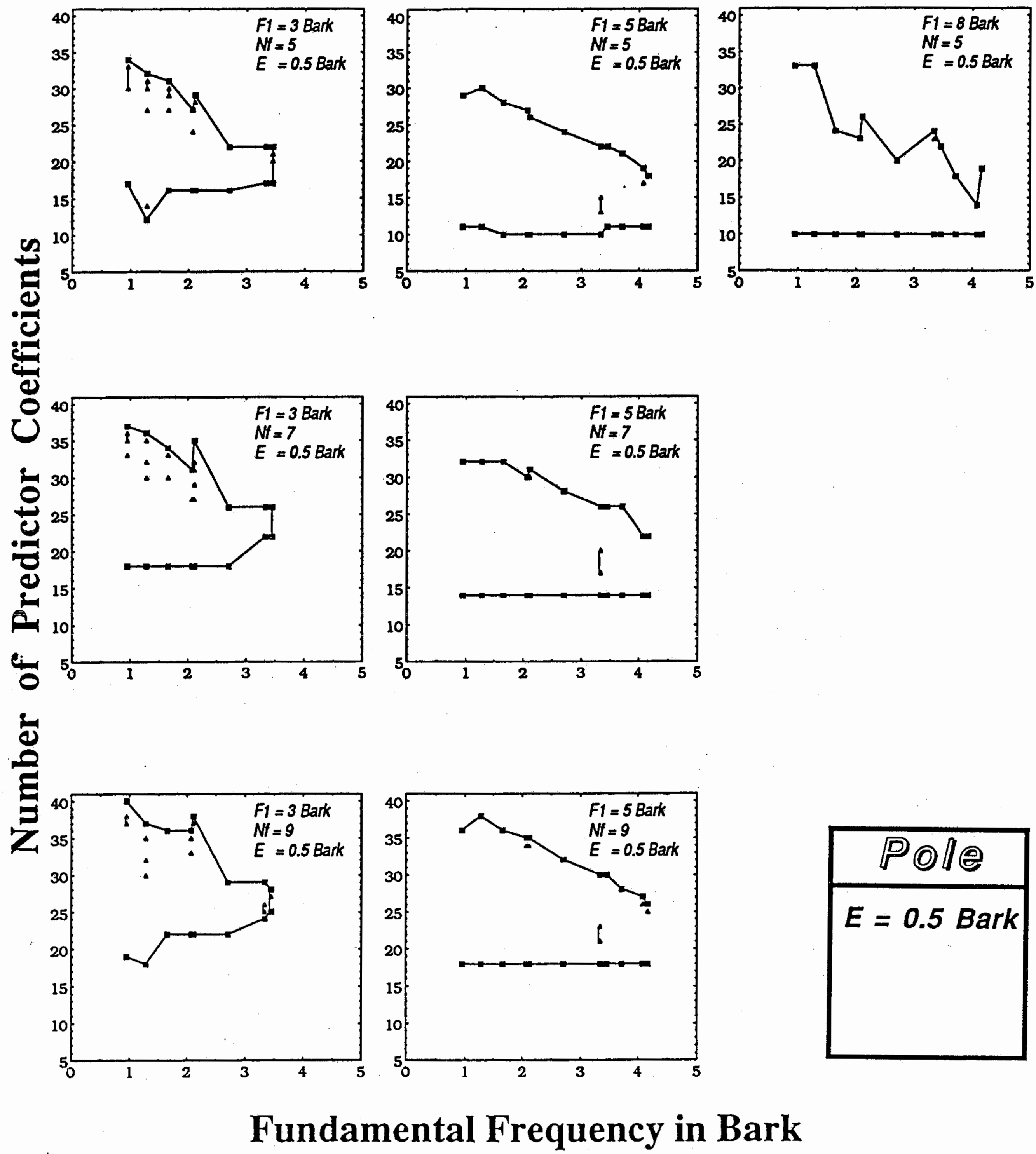


図4.1.15 フォルマント周波数の推定誤差の絶対値が0.5Bark以下である分析次数の上限、下限と資料の基本周波数との関係（中性母音）

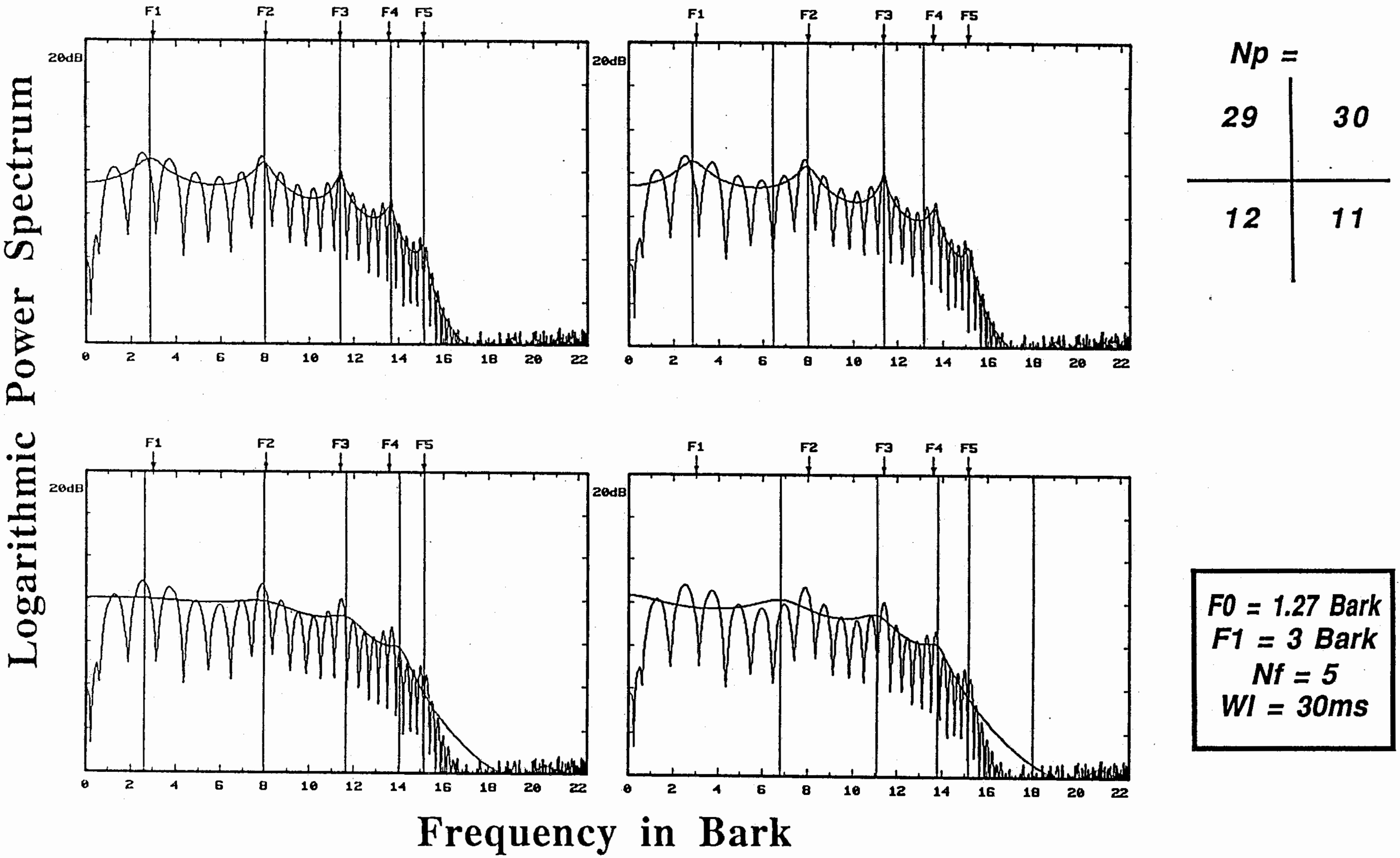


図4.1.16(a) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

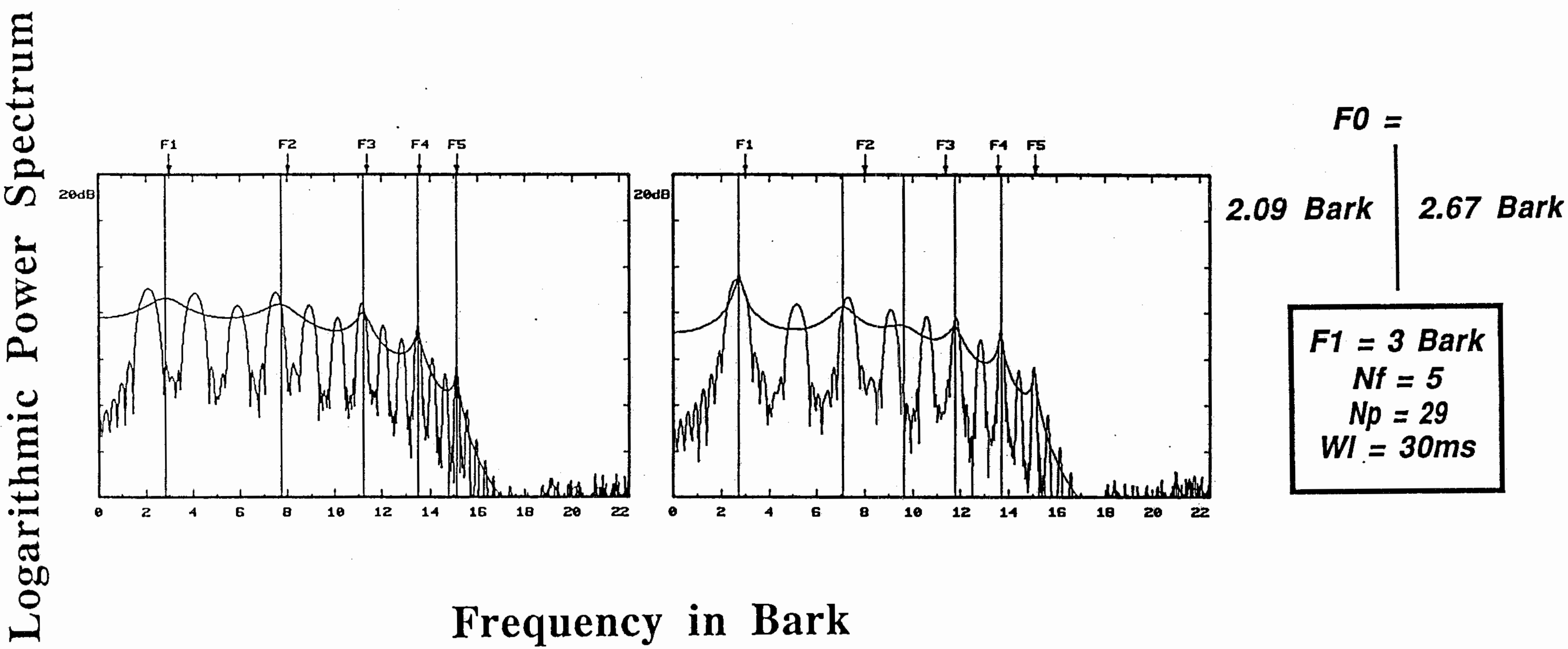


図4.1.16(b) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

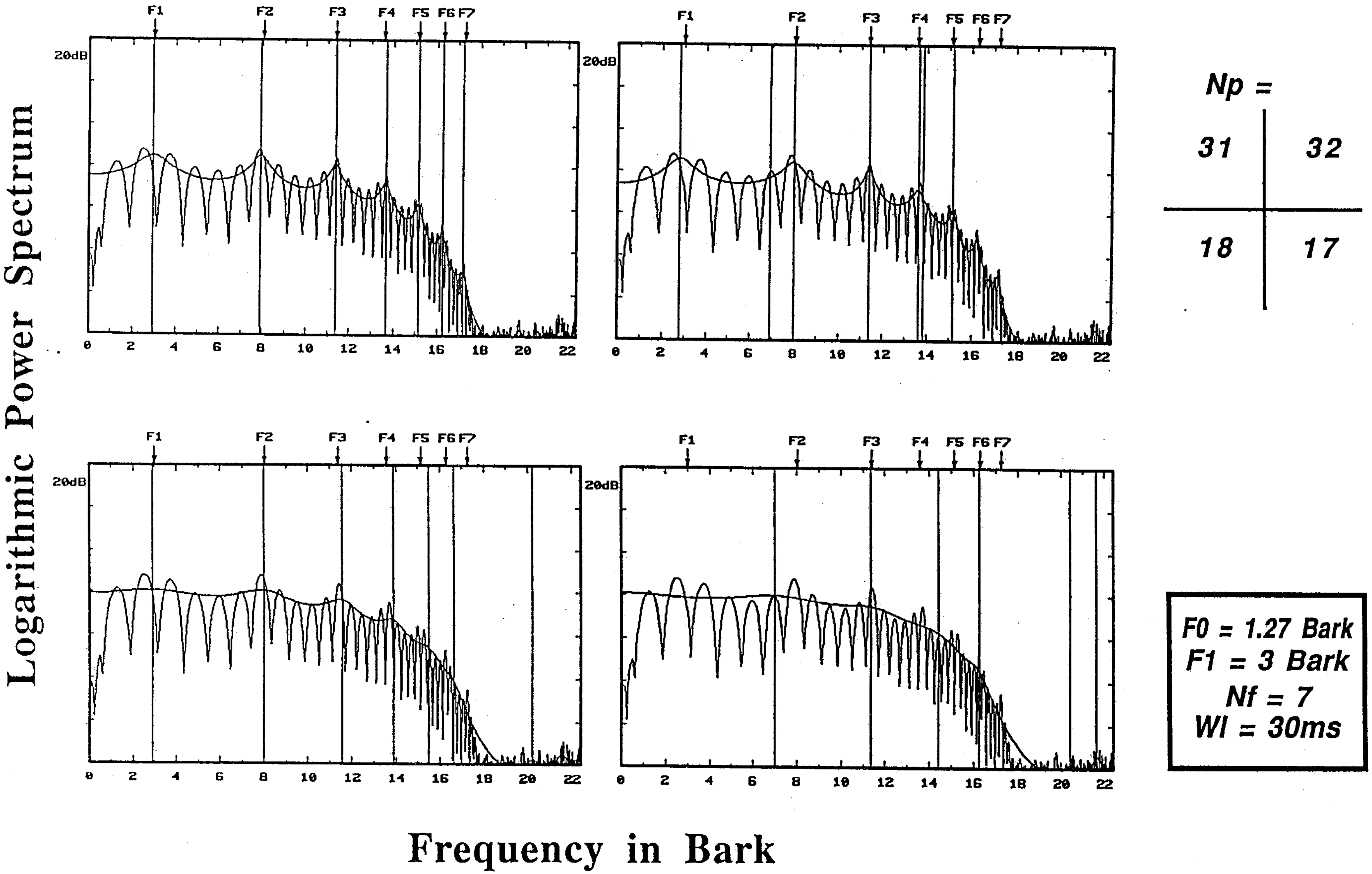


図4.1.17(a) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

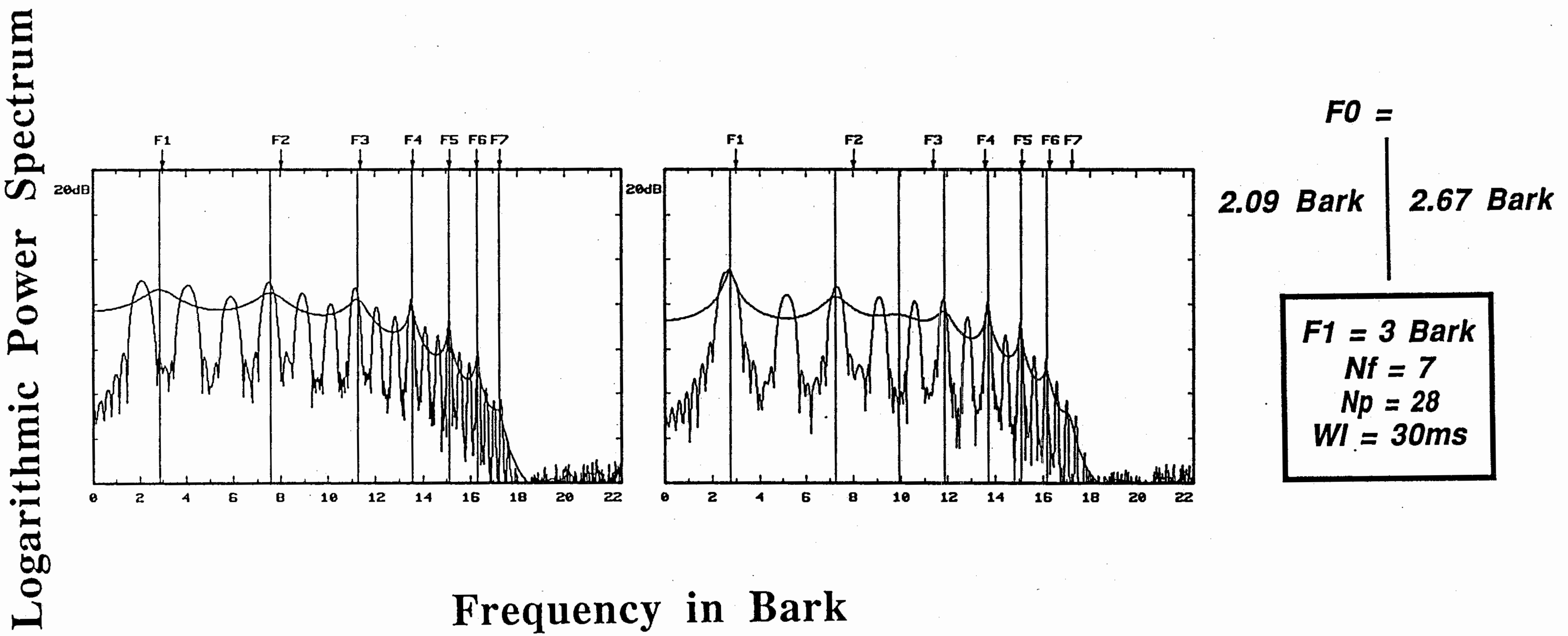


図4.1.17(b) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

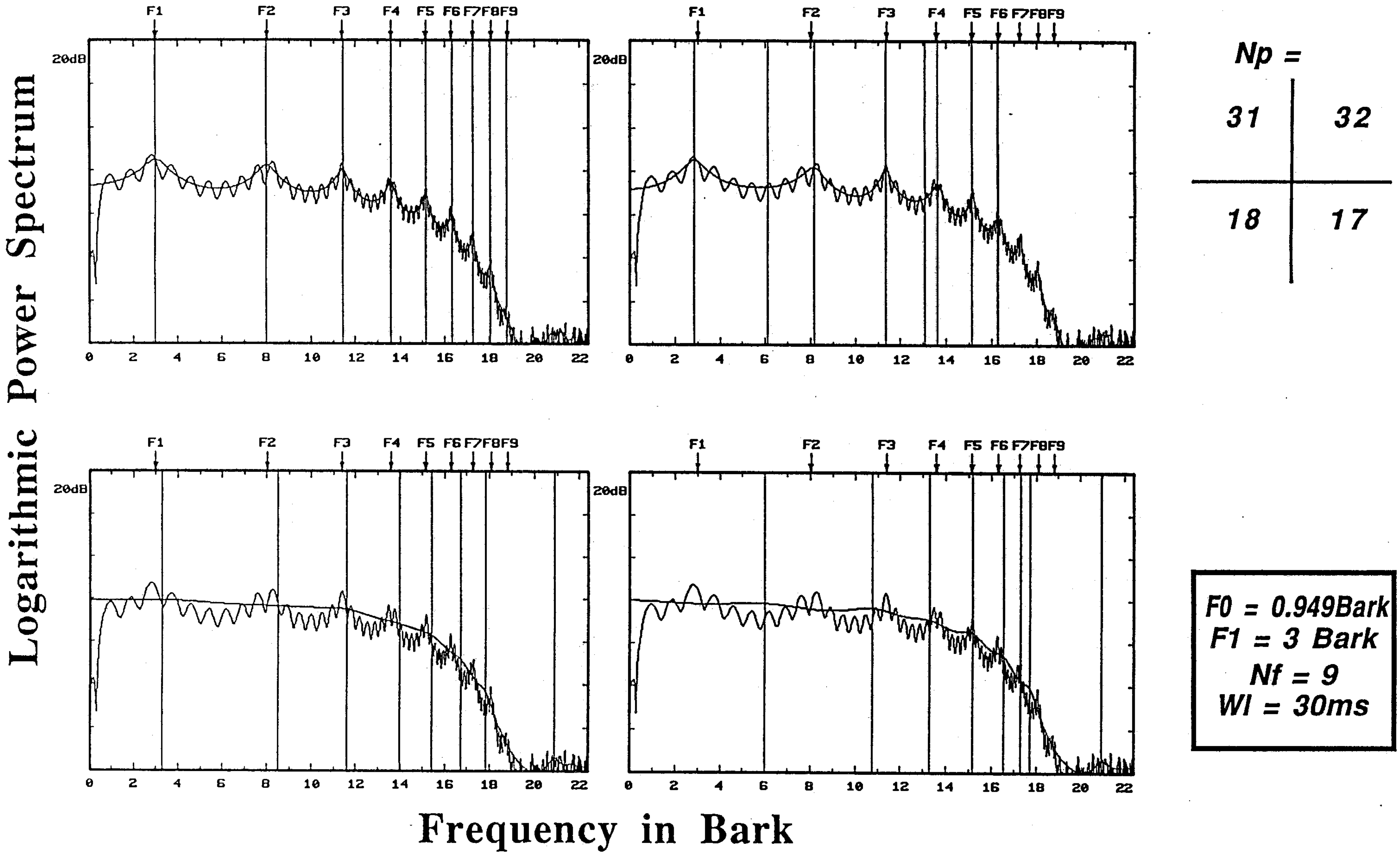


図4.1.18(a) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

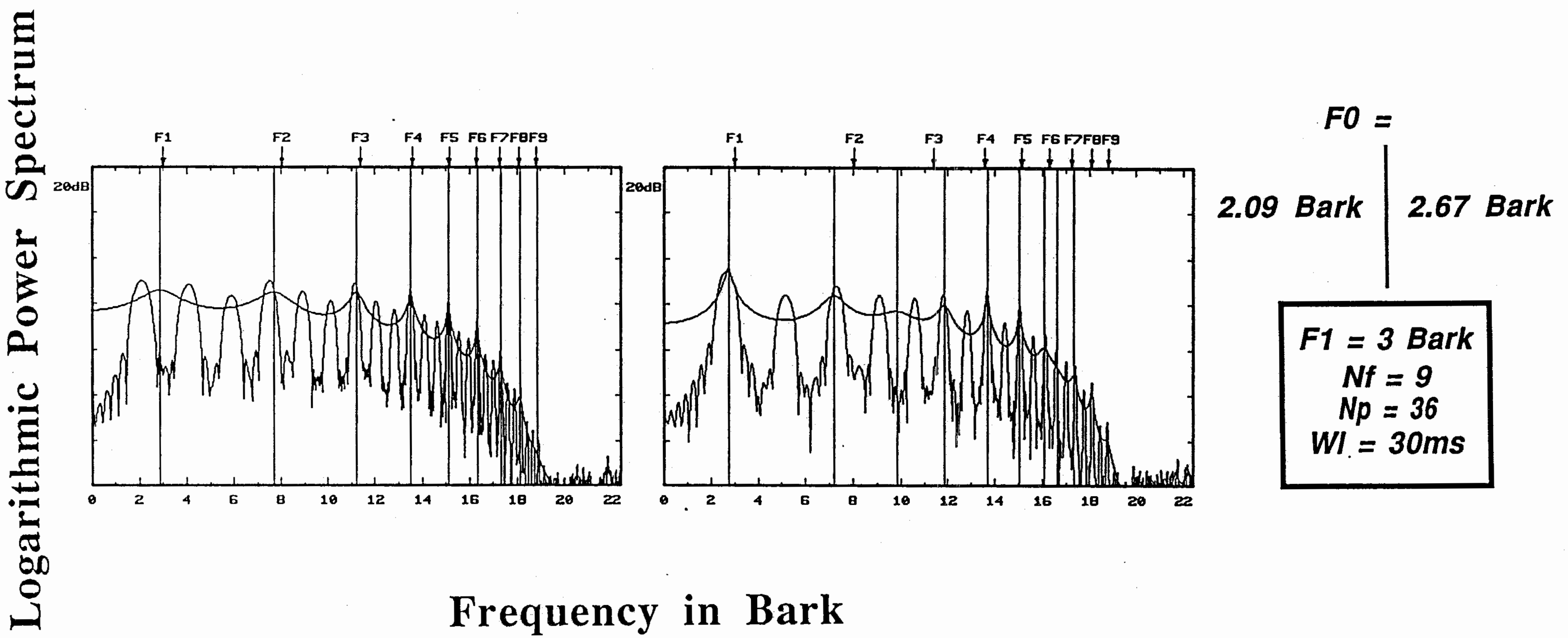


図4.1.18(b) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

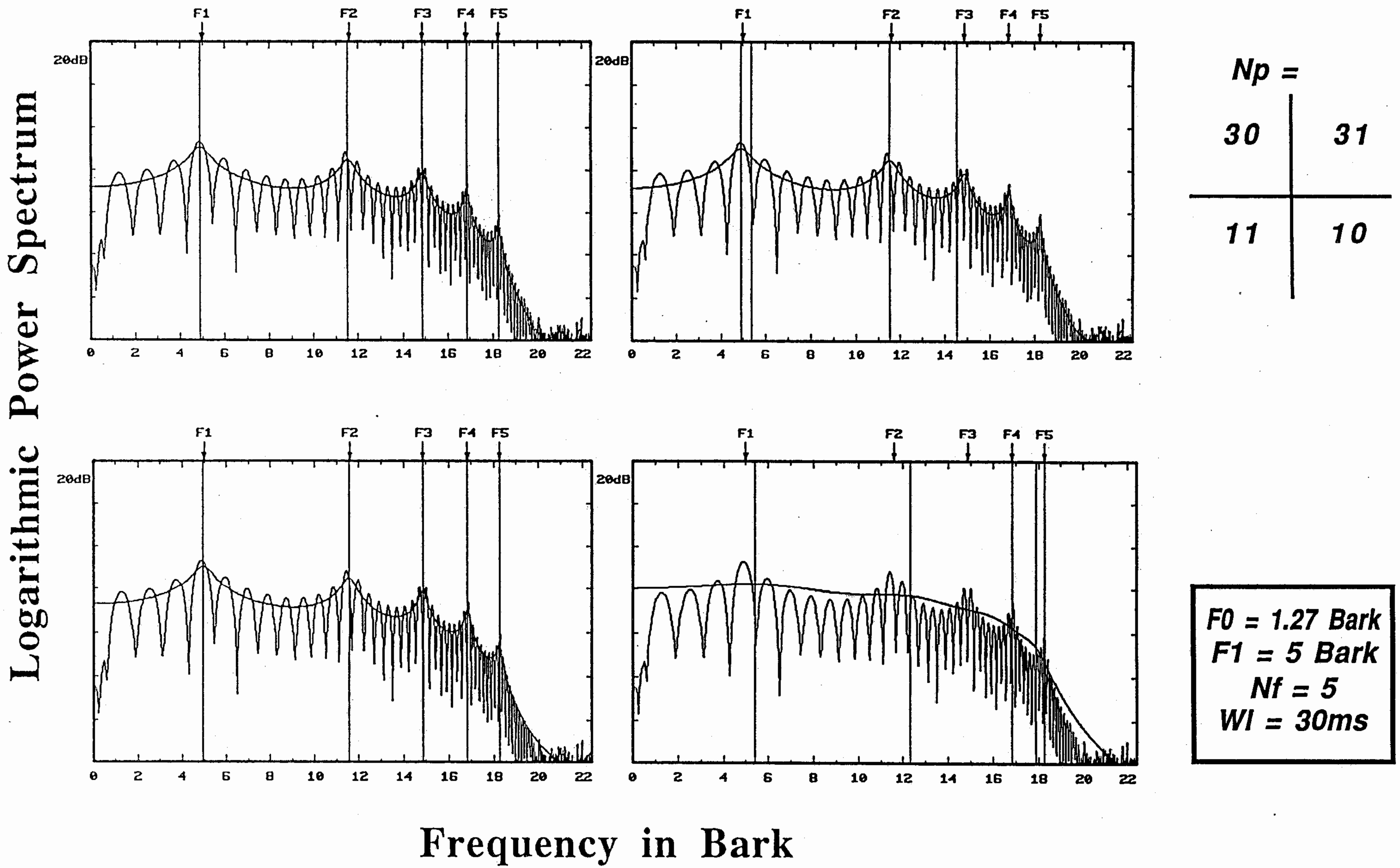


図4.1.19(a) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

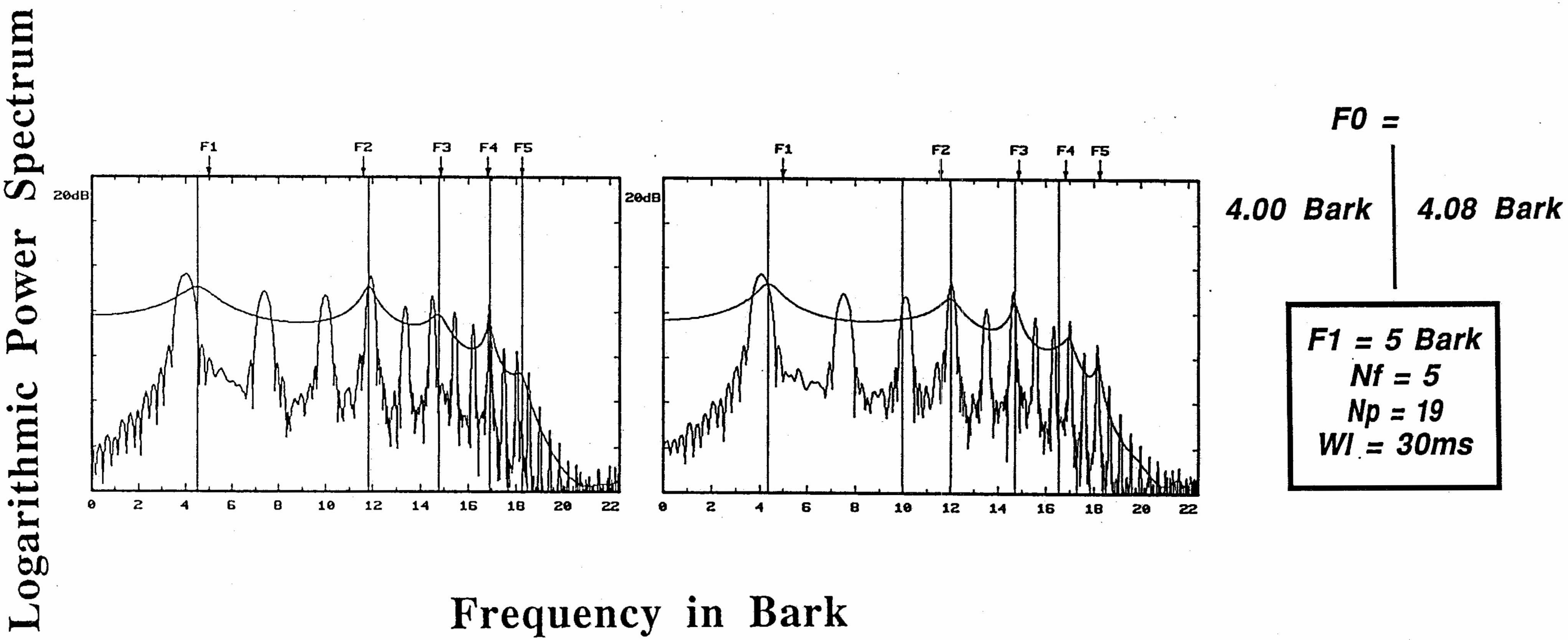


図4.1.19(b) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

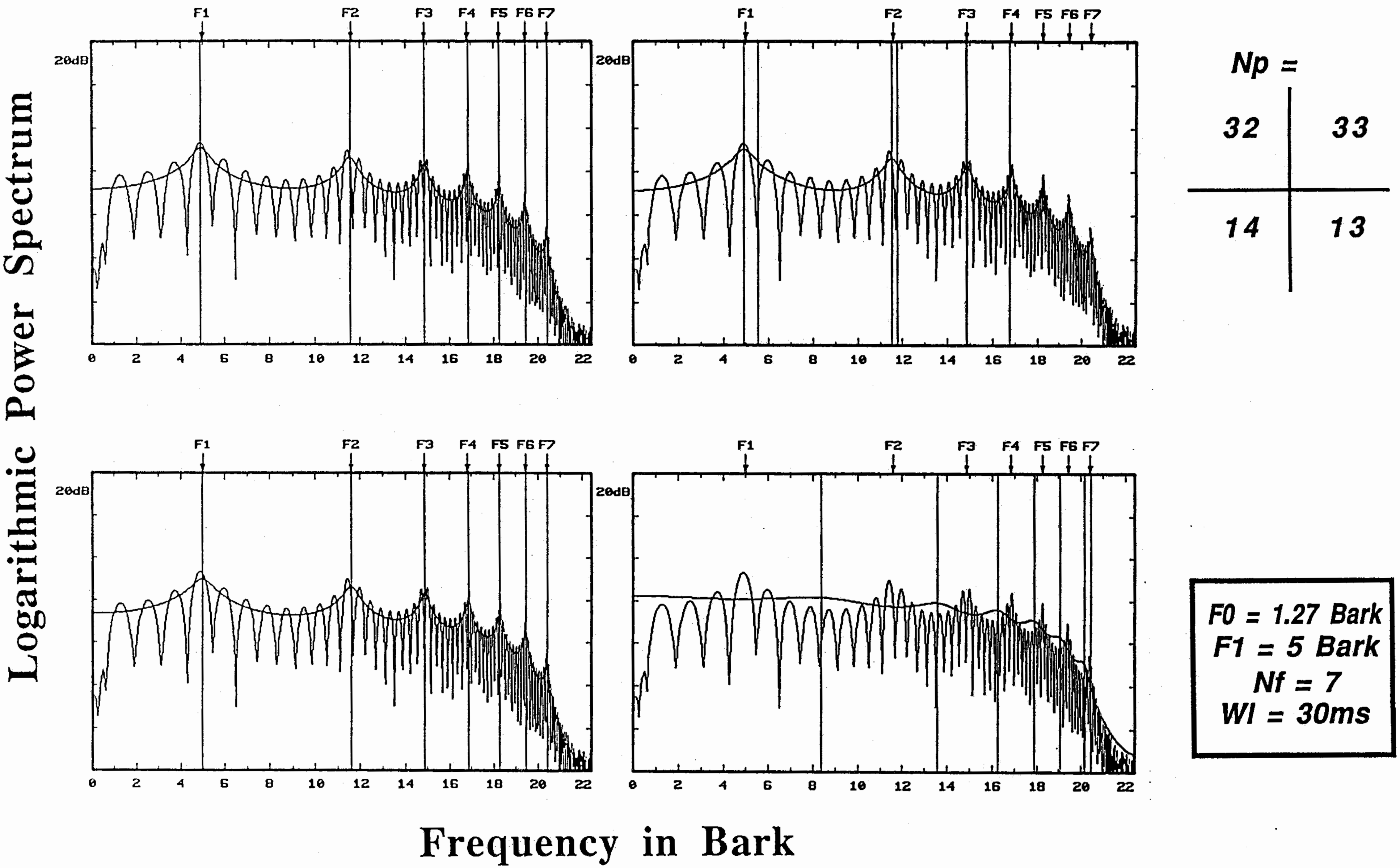


図4.1.20(a) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

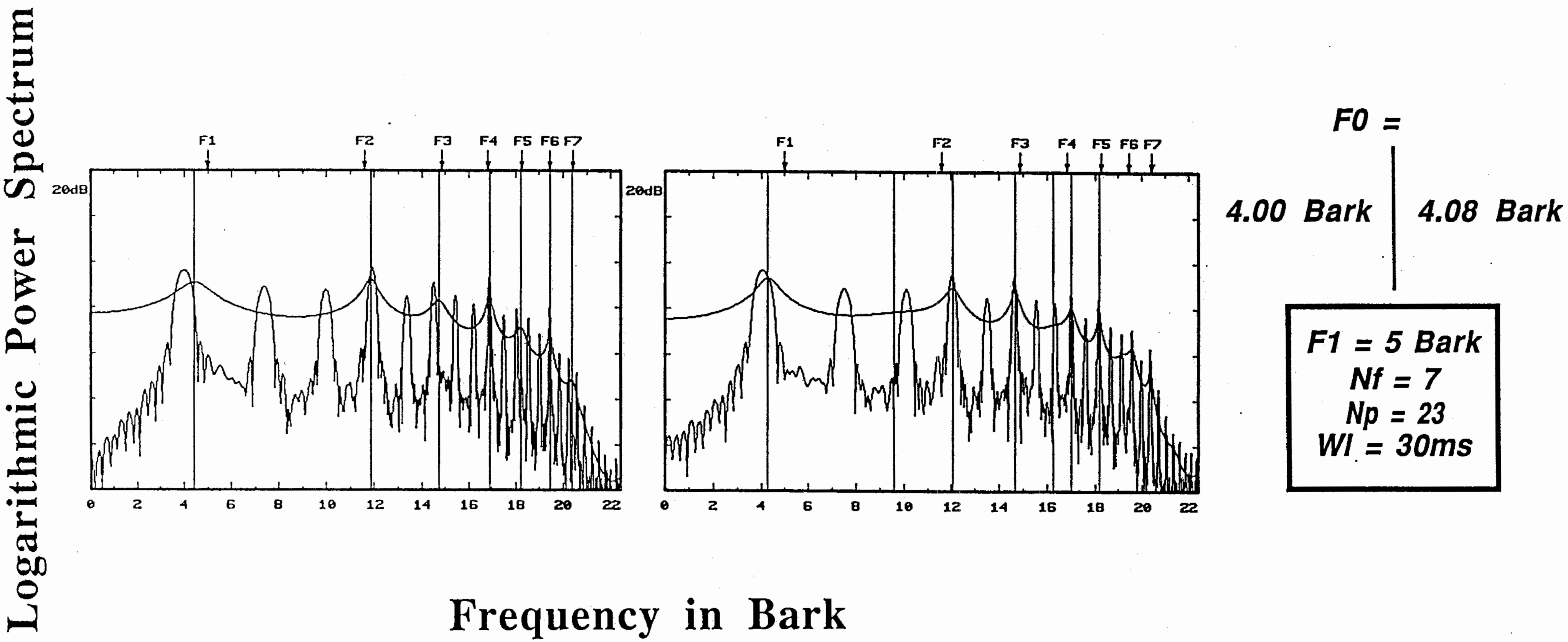


図4.1.20(b) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

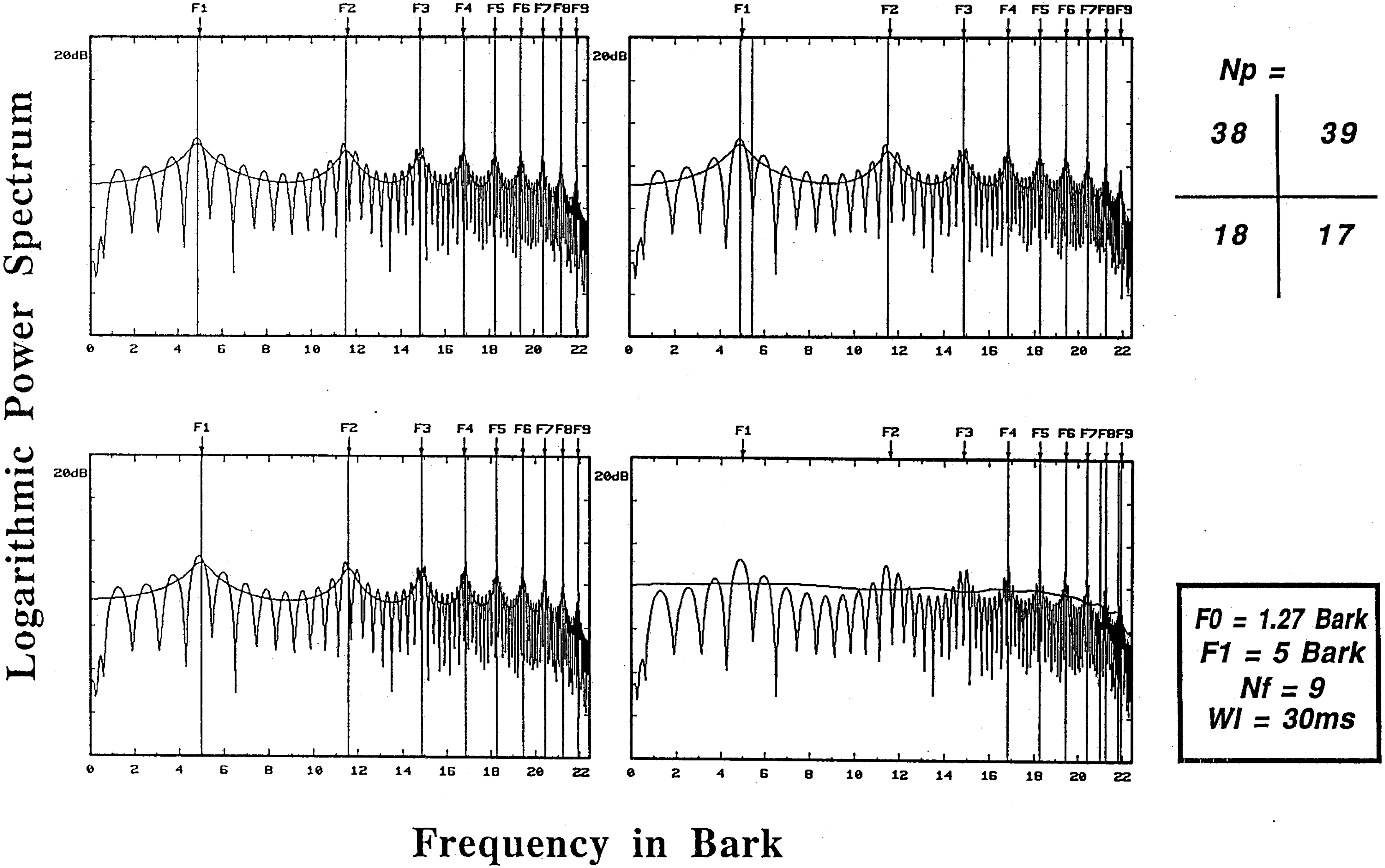


図4.1.21(a) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

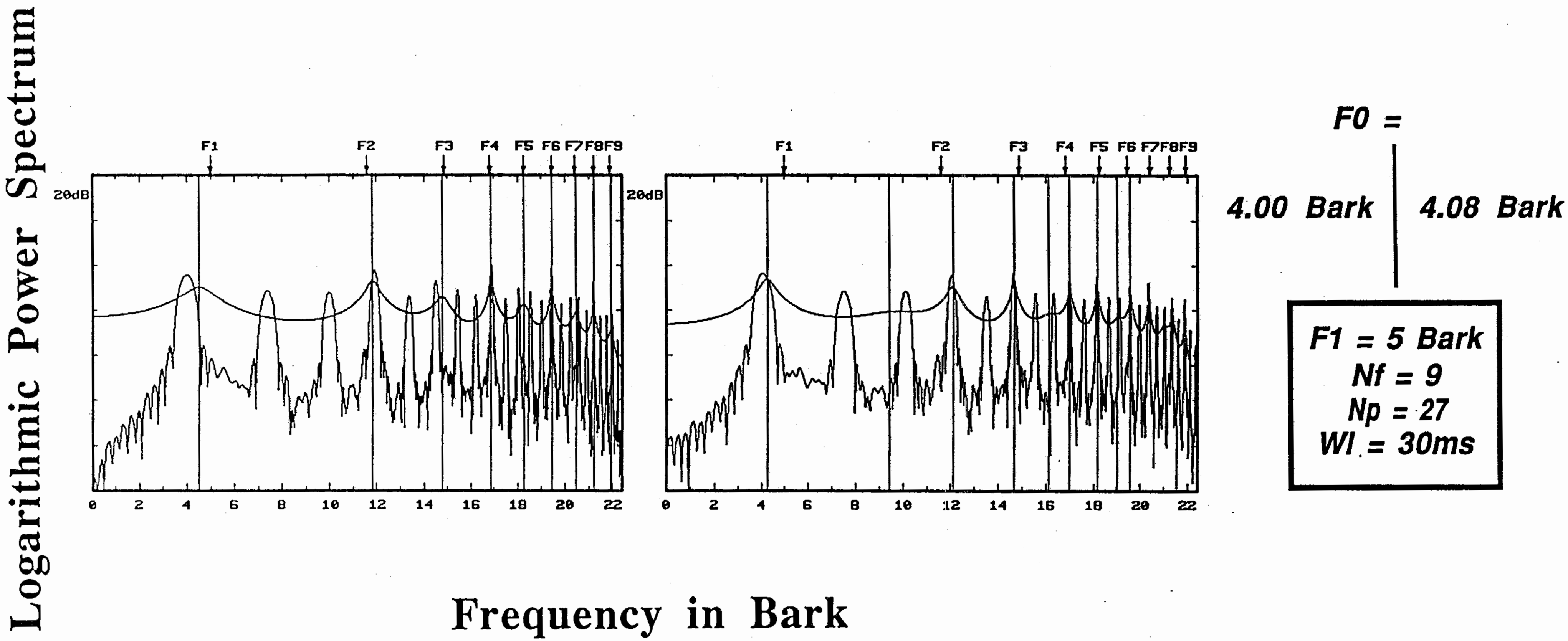


図4.1.21(b) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

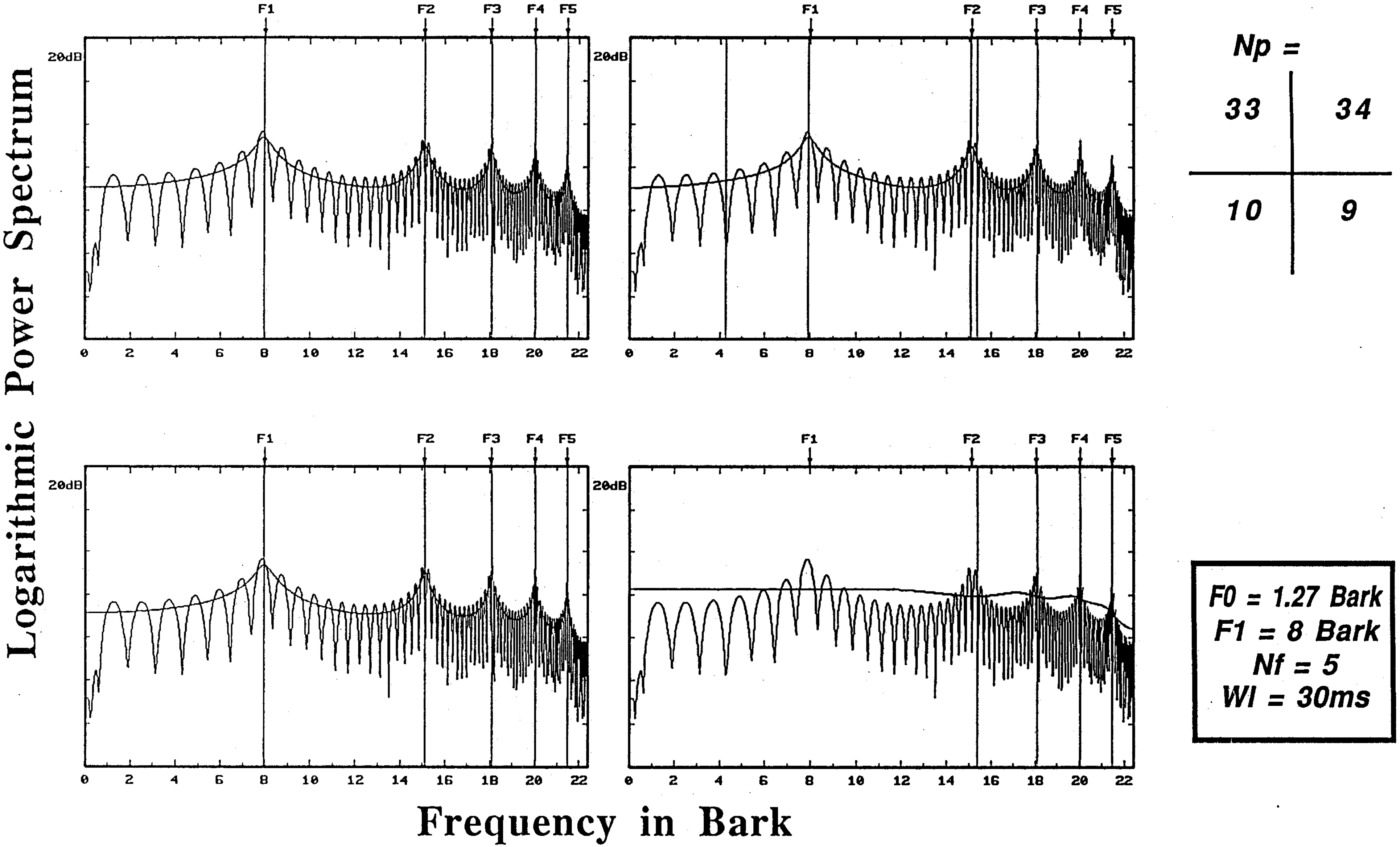


図4.1.22(a) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

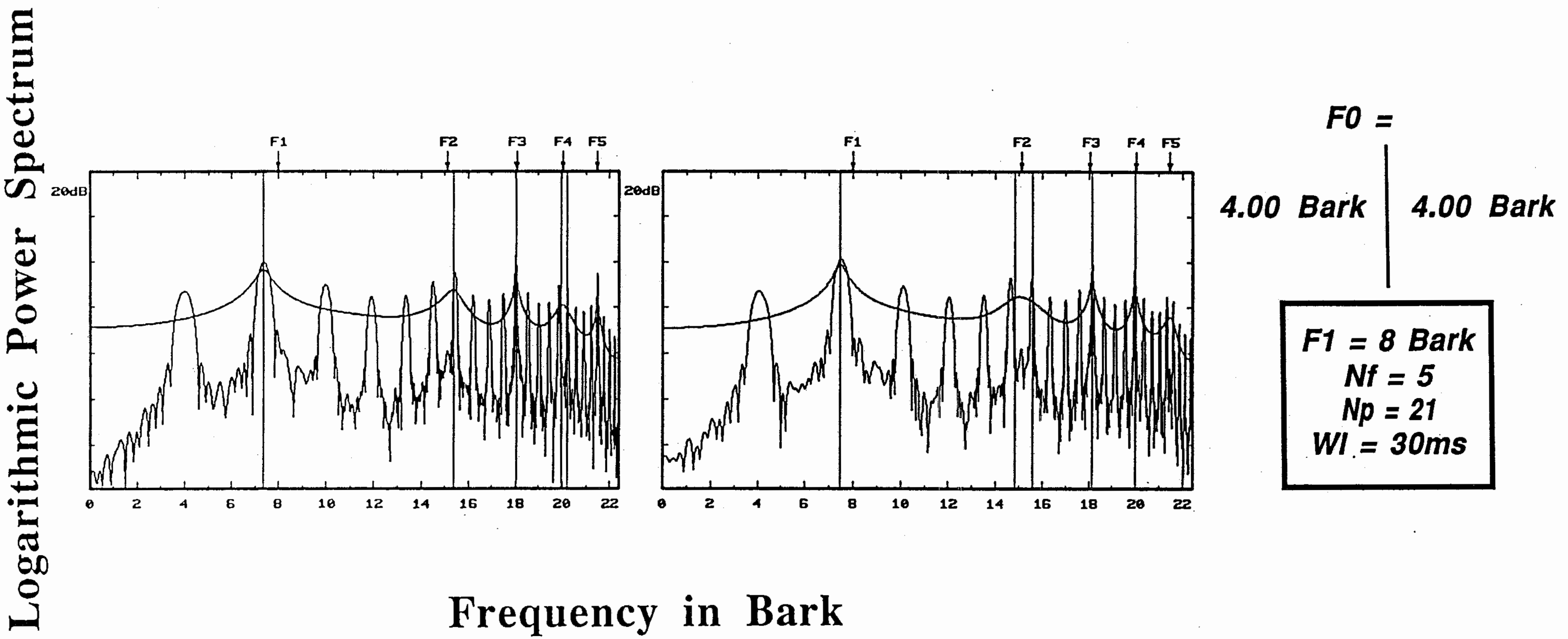


図4.1.22(b) 資料のDFTパワスペクトルと推定されたスペクトル包絡
矢印は合成時のフォルマント周波数、垂線は推定されたフォルマント周波数を示す

4.1.3 基本周波数、帯域幅の影響

本節では、資料の基本周波数、合成時のフォルマント帯域幅がフォルマント周波数、フォルマント帯域幅の推定値に及ぼす影響を調べた結果を示す。

(i) 基本周波数

図4.1.23(a)～図4.1.25(a)は、フォルマント周波数の推定誤差を資料のF0を横軸にして描いたものである。各図とも左上から右下へ第1～第4フォルマントに対する誤差の図を示す。

図4.1.23(b)～図4.1.25(b)は、フォルマント帯域幅について同様に求めたものである。

図4.1.23、図4.1.24、図4.1.25は、それぞれF1=3 Bark、5 Bark、8 Barkの資料に対する結果を示している。

(ii) 帯域幅

F1=5 Barkのフォルマントセットに対して、フォルマント帯域幅を4通りに変化させた資料を用い、前項と同様フォルマント推定誤差と資料のF0との関係を調べた。

図4.1.26(a)～図4.1.29(a)は、フォルマント周波数の推定誤差を資料のF0を横軸にして描いたものである。各図とも左上から右下へ第1～第4フォルマントに対する誤差の図を示す。

図4.1.26(b)～図4.1.29(b)は、フォルマント帯域幅について同様に求めたものである。

図4.1.26、図4.1.27、図4.1.28、図4.1.29は、F1=5 Barkで、合成時のフォルマント帯域幅がそれぞれ50 Hz、100 Hz、200 Hz、300 Hzの資料に対する結果を示している。

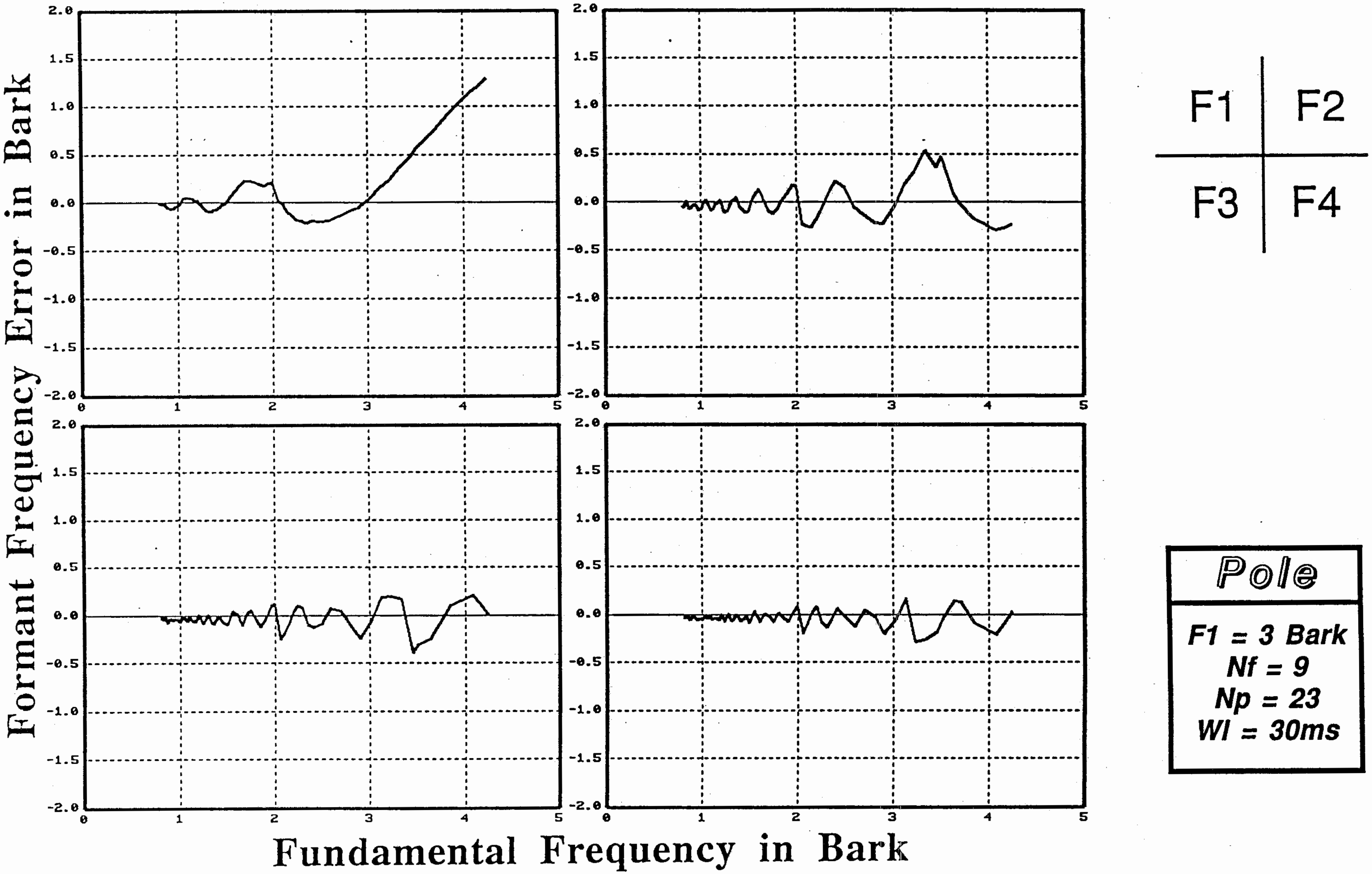


図4.1.23(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

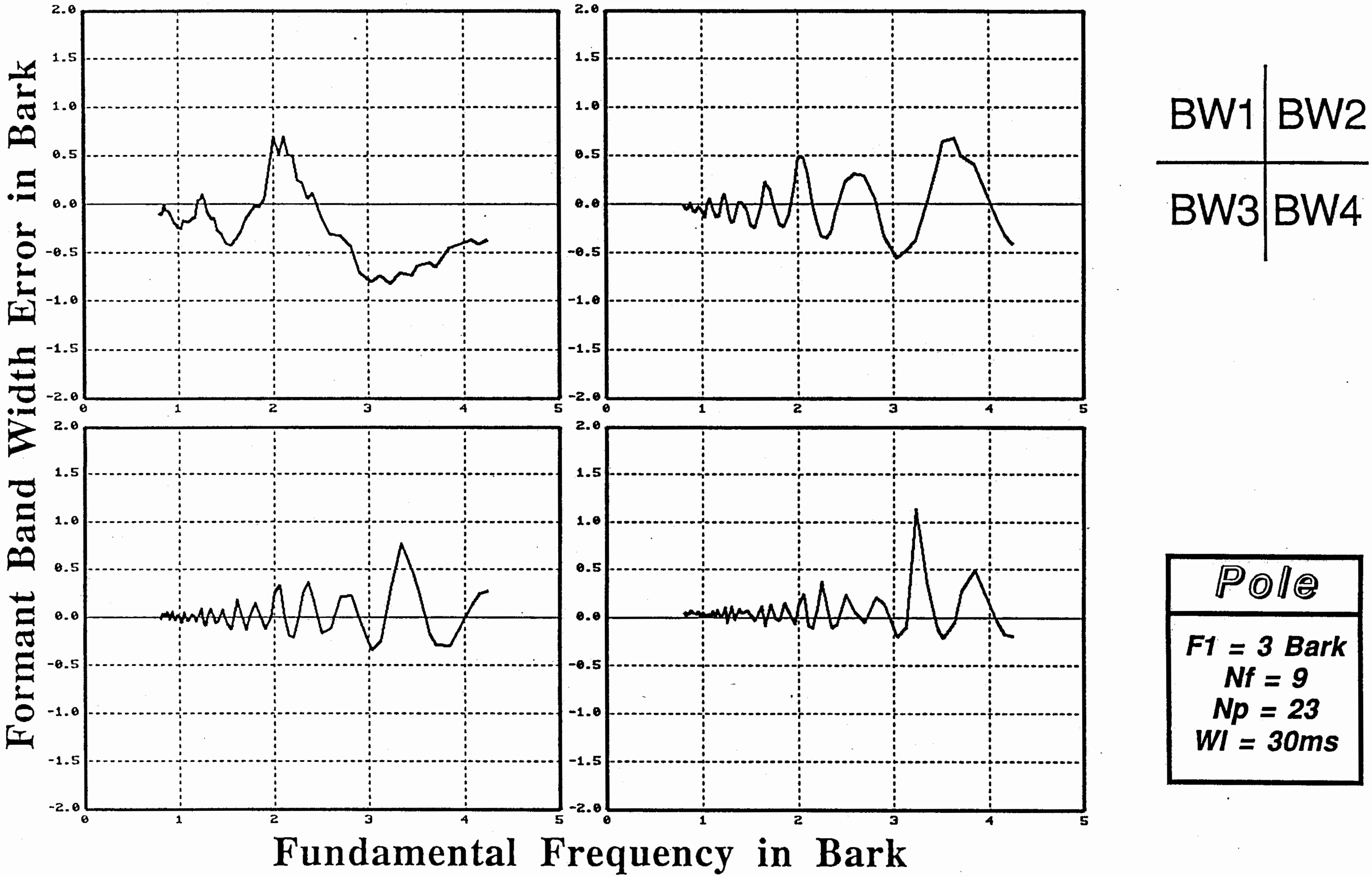


図4.1.23(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

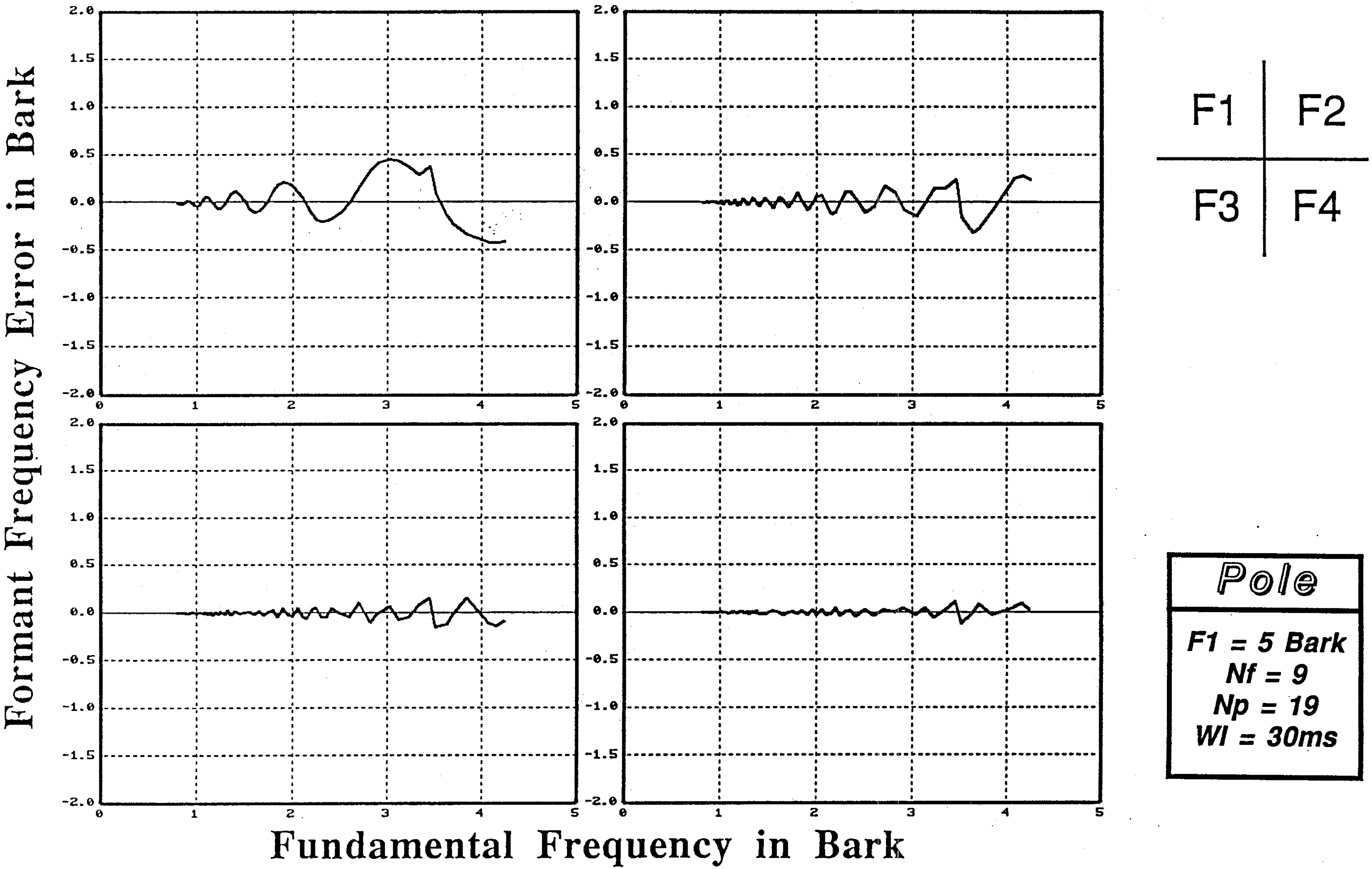


図4.1.24(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

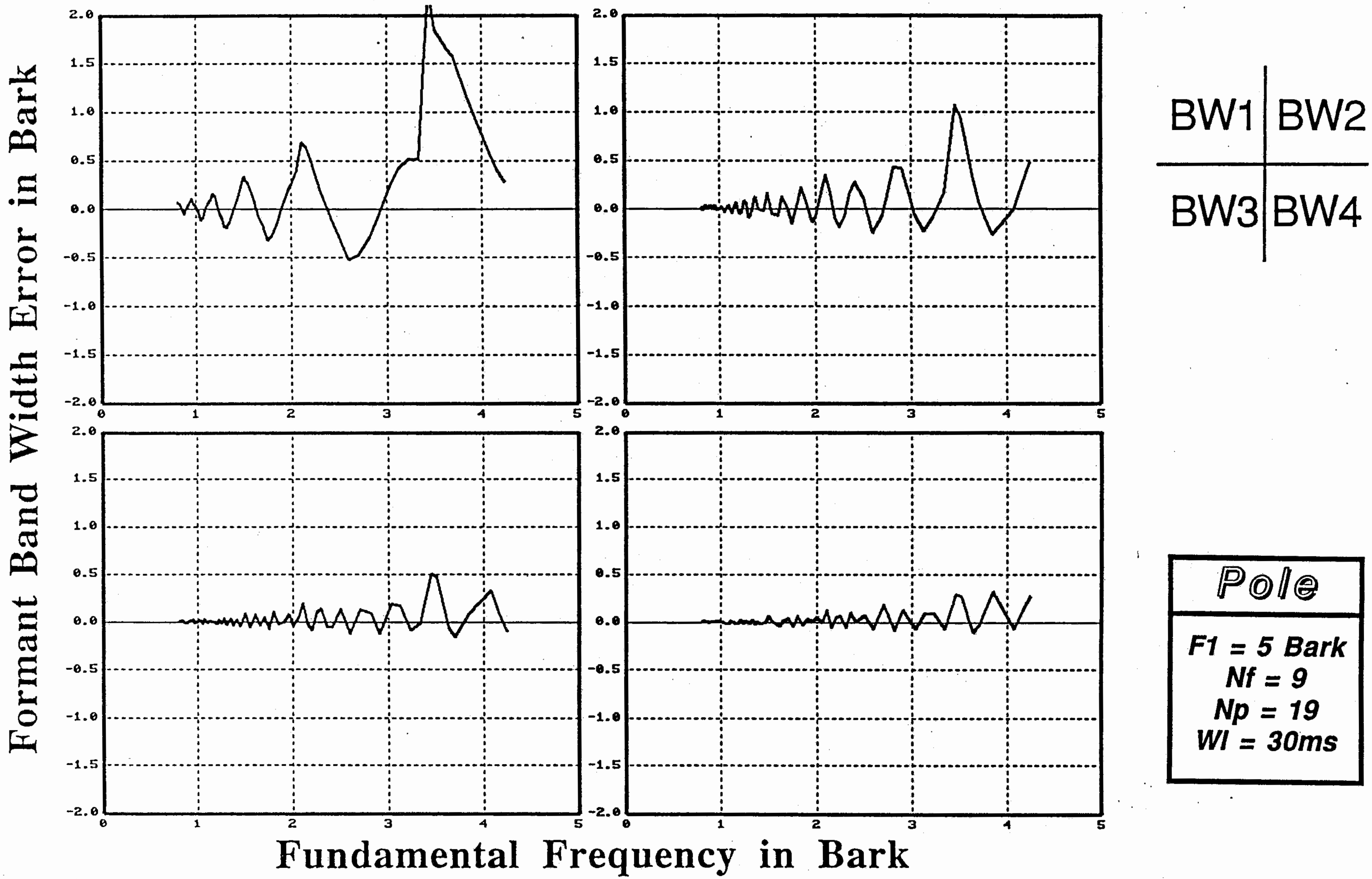


図4.1.24(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

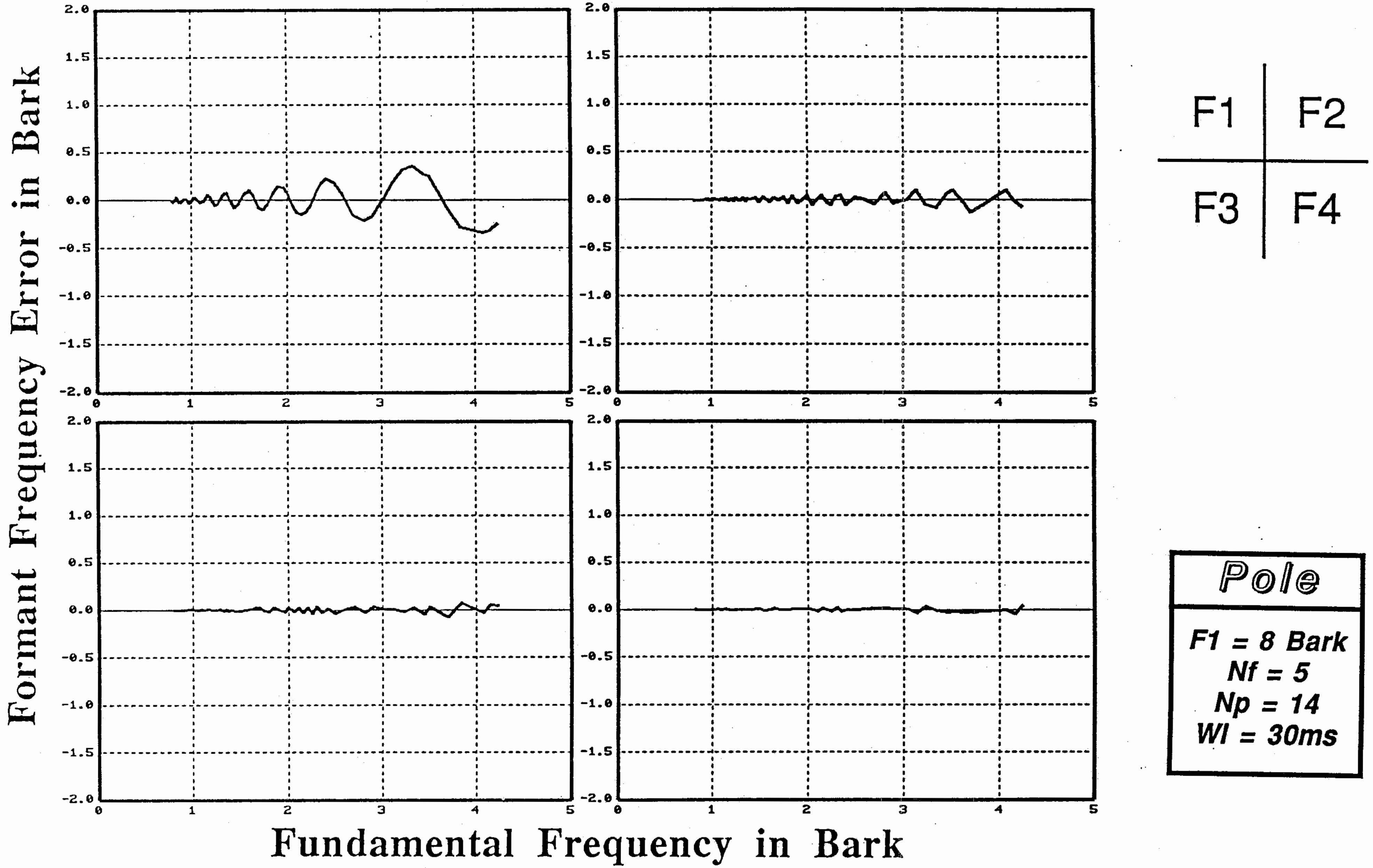


図4.1.25(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

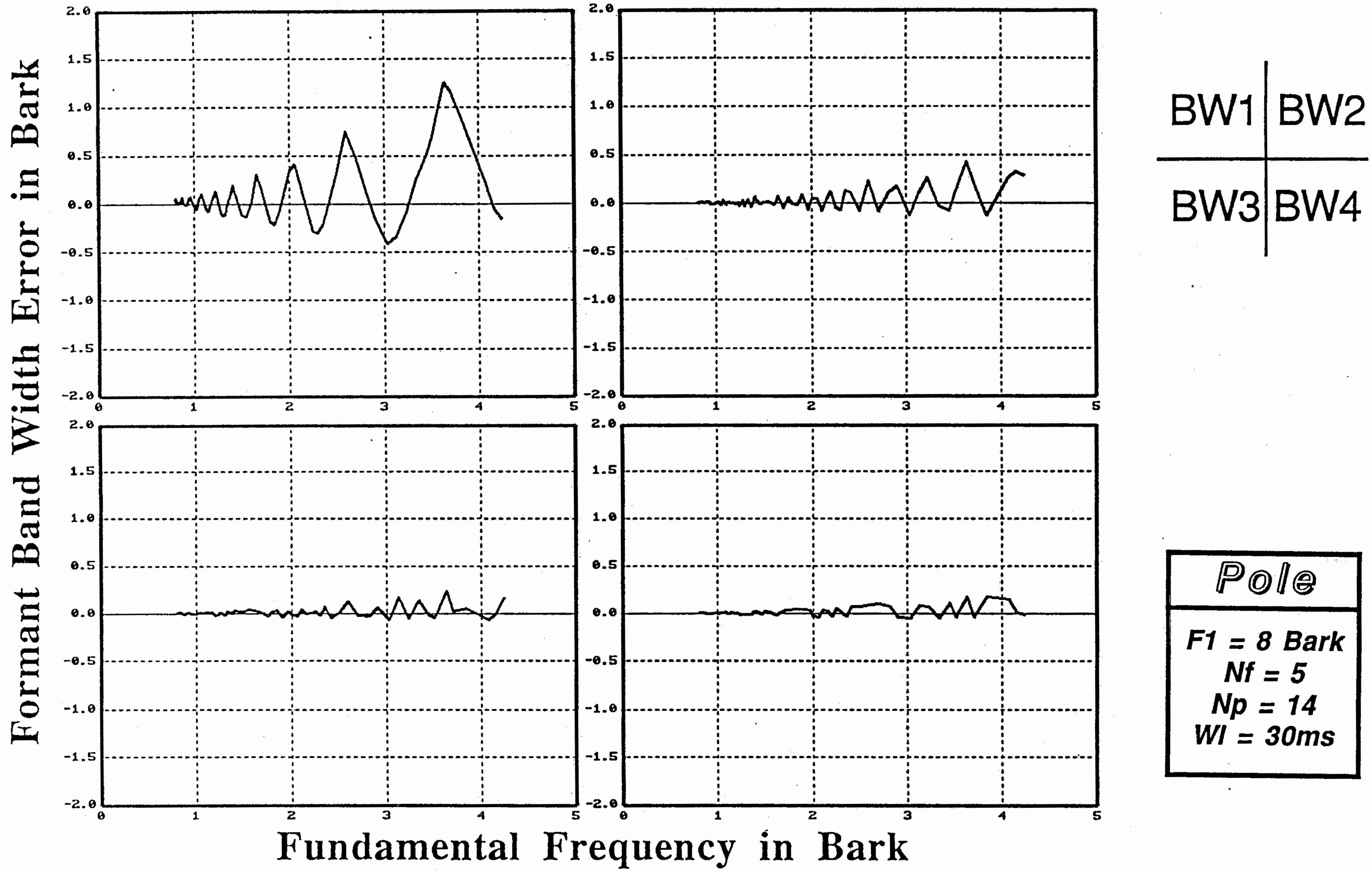


図4.1.25(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

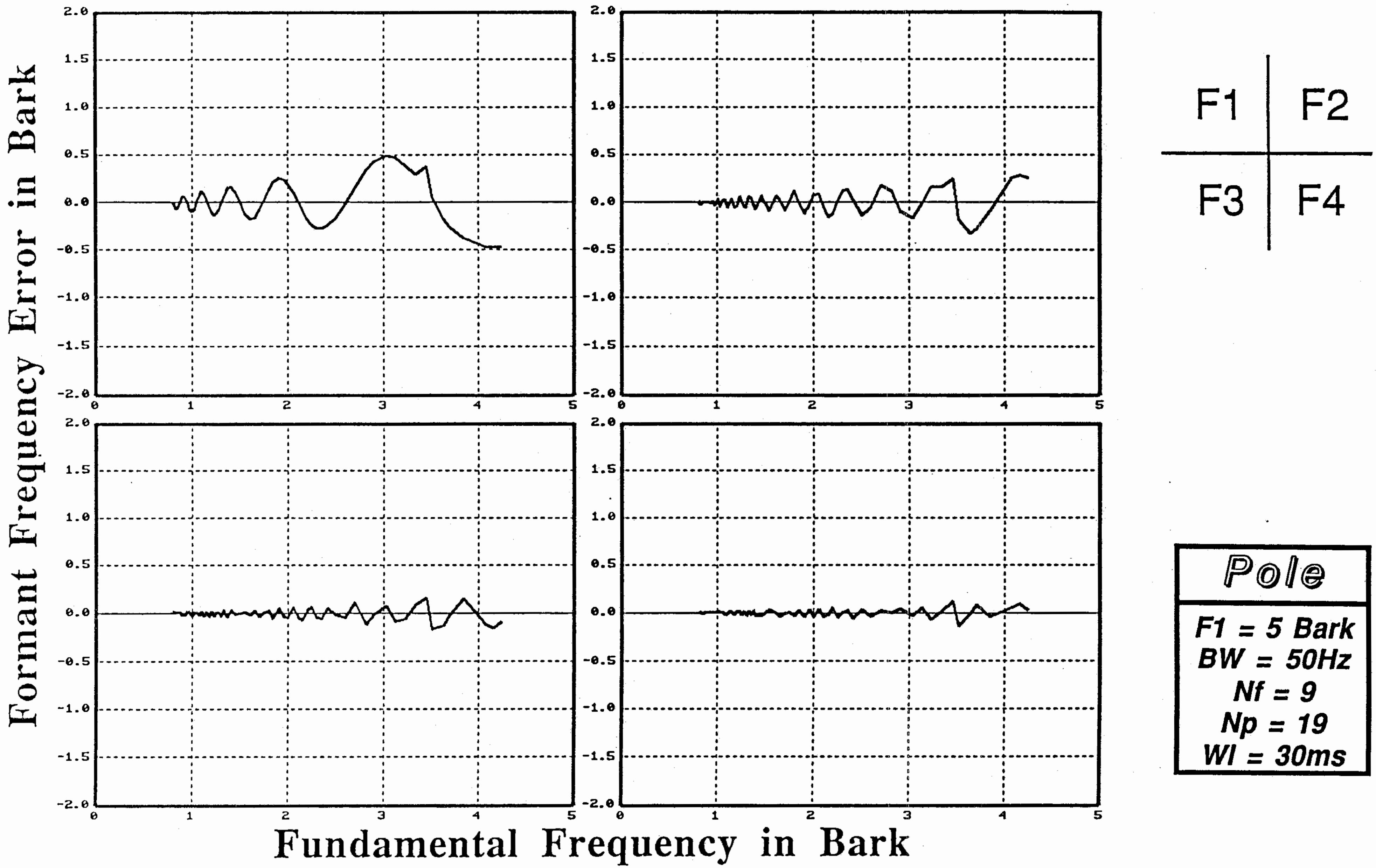


図4.1.26(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

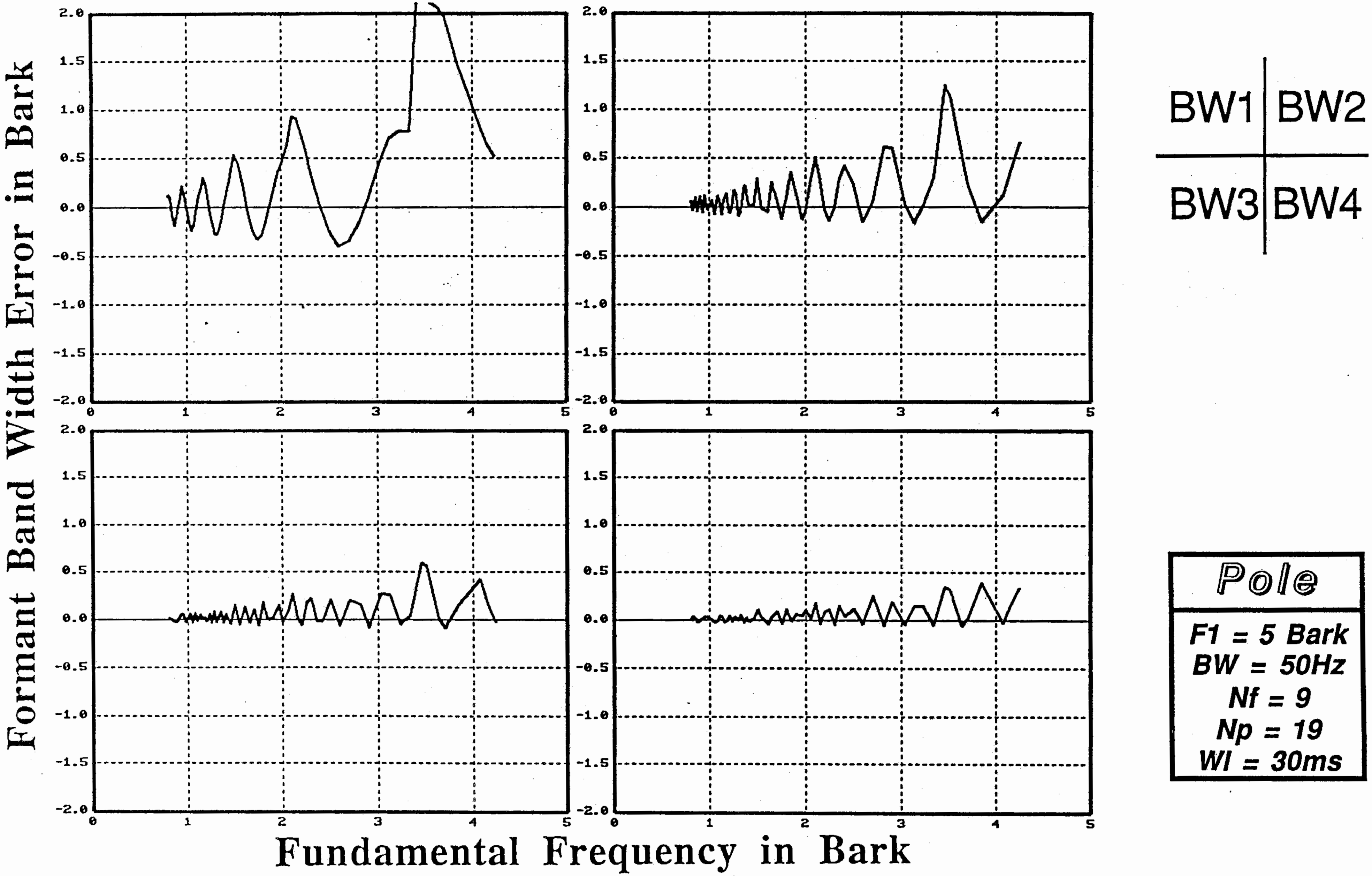


図4.1.26(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

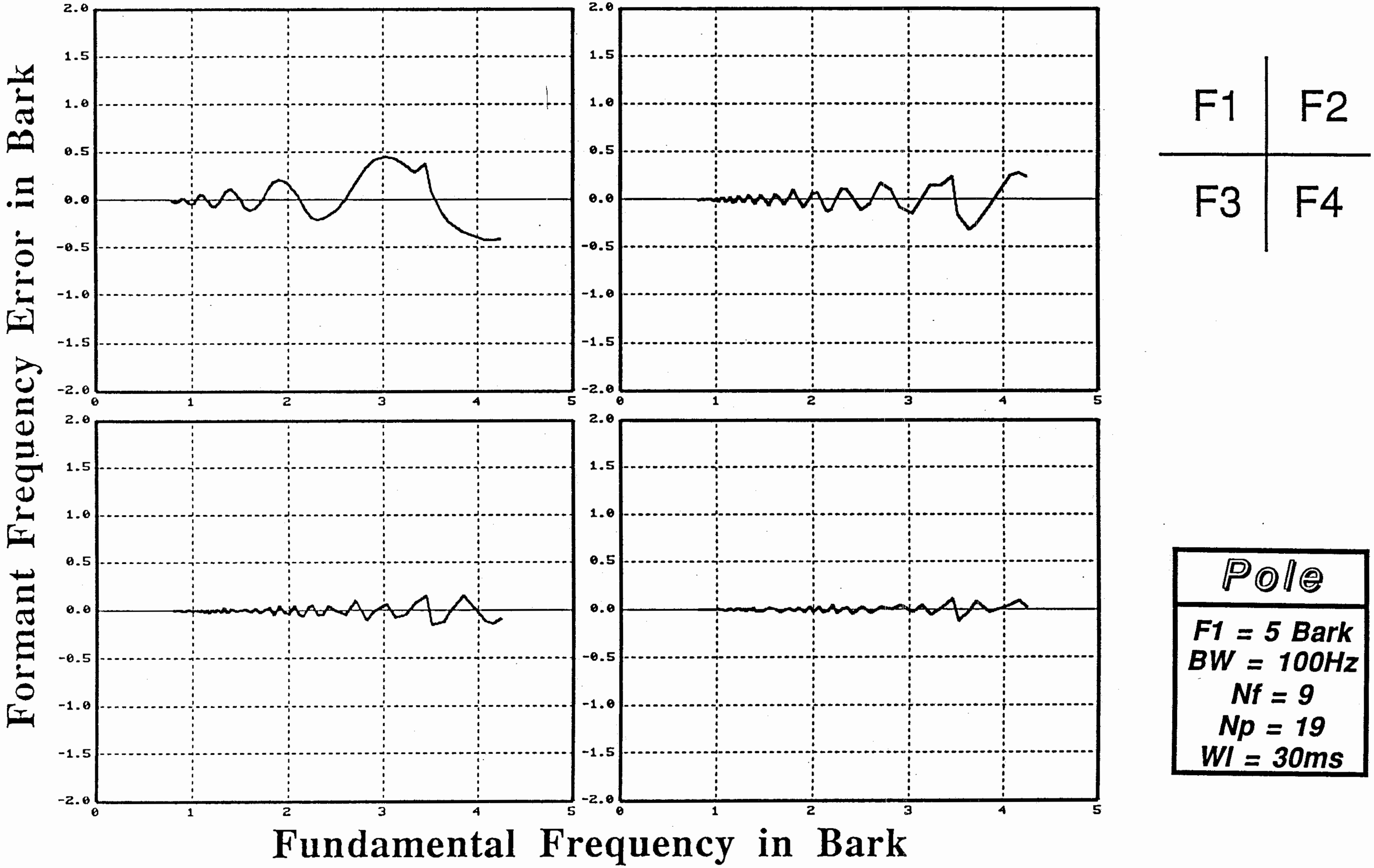


図4.1.27(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

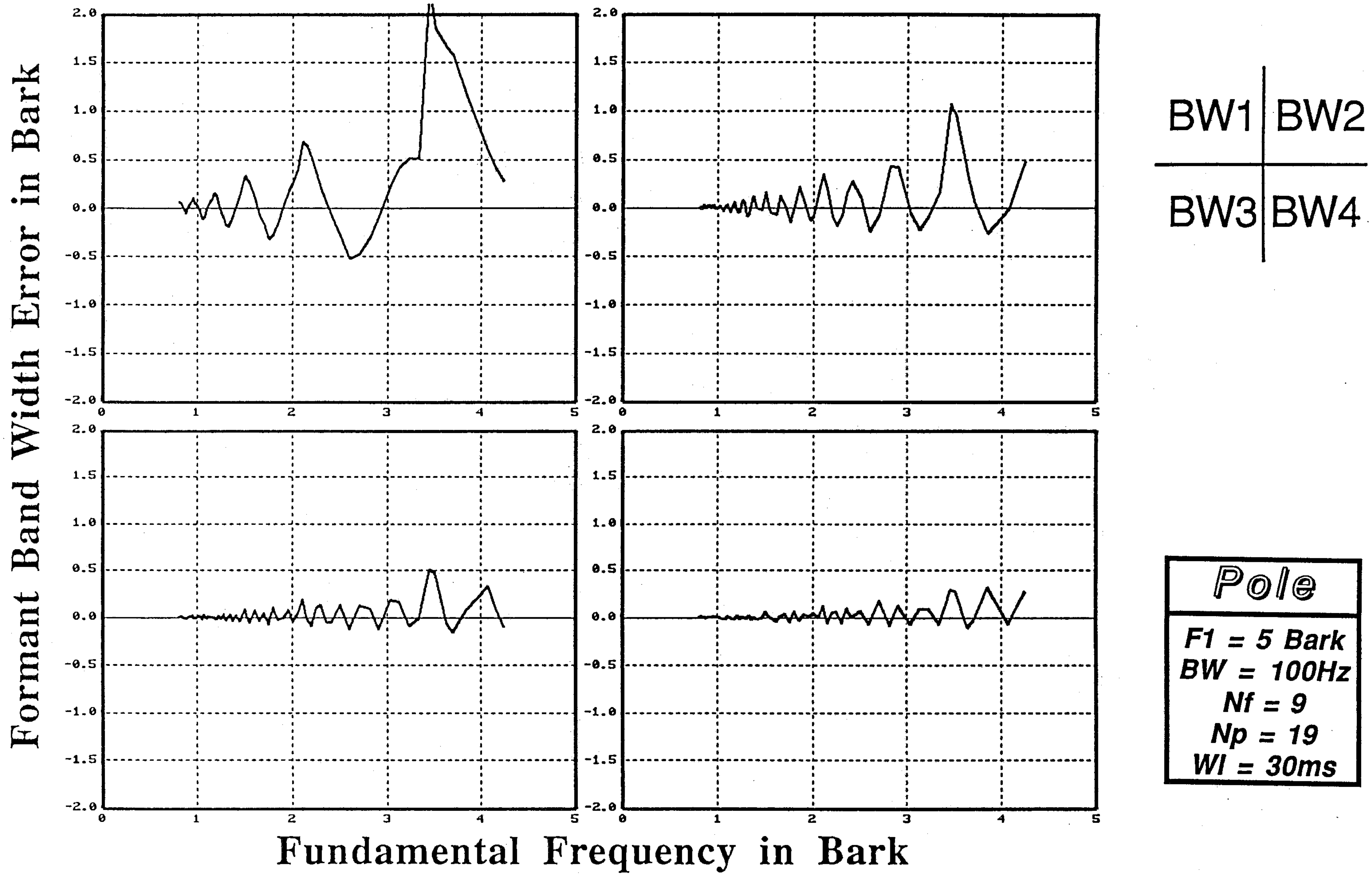


図4.1.27(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

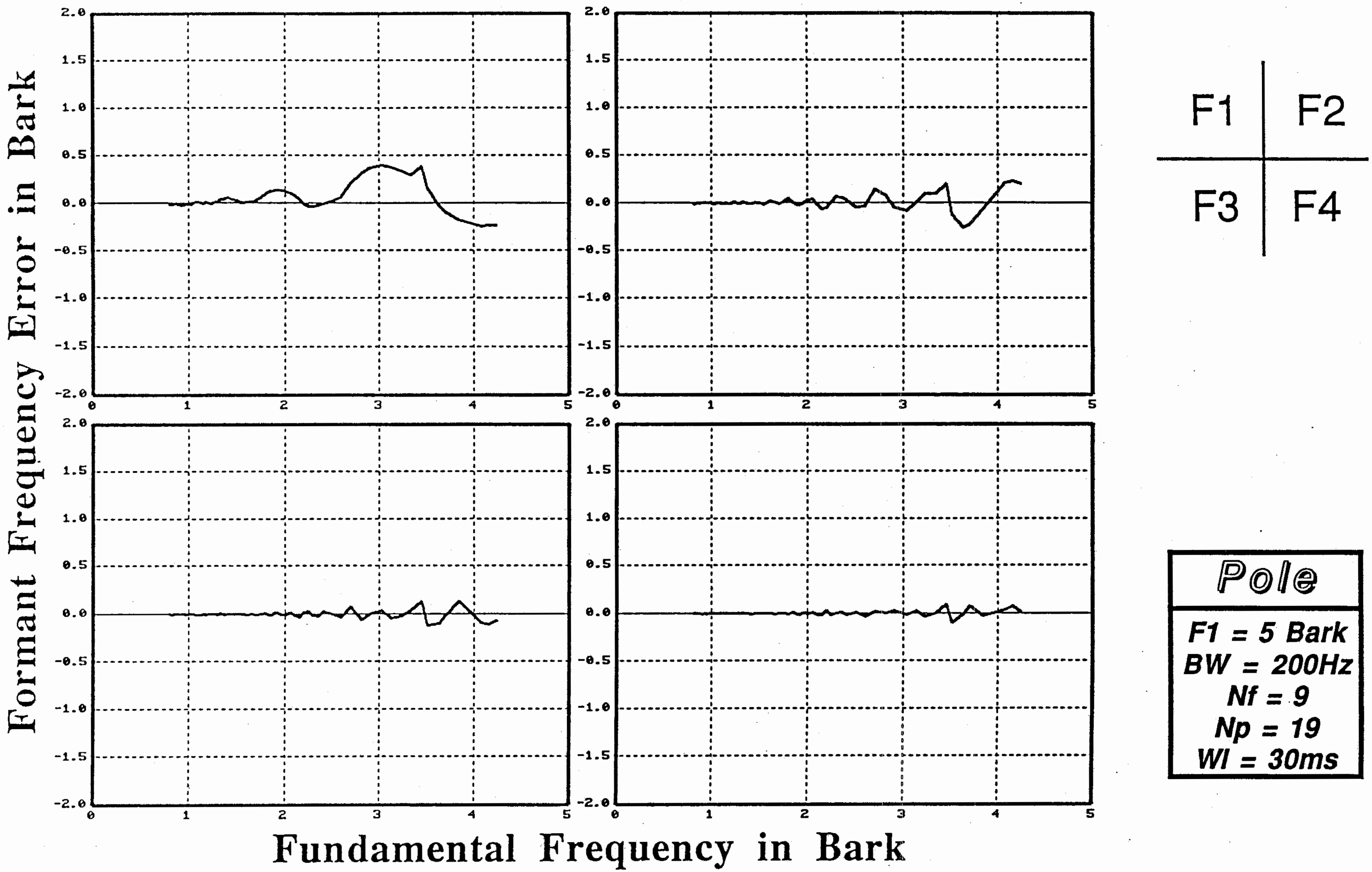


図4.1.28(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

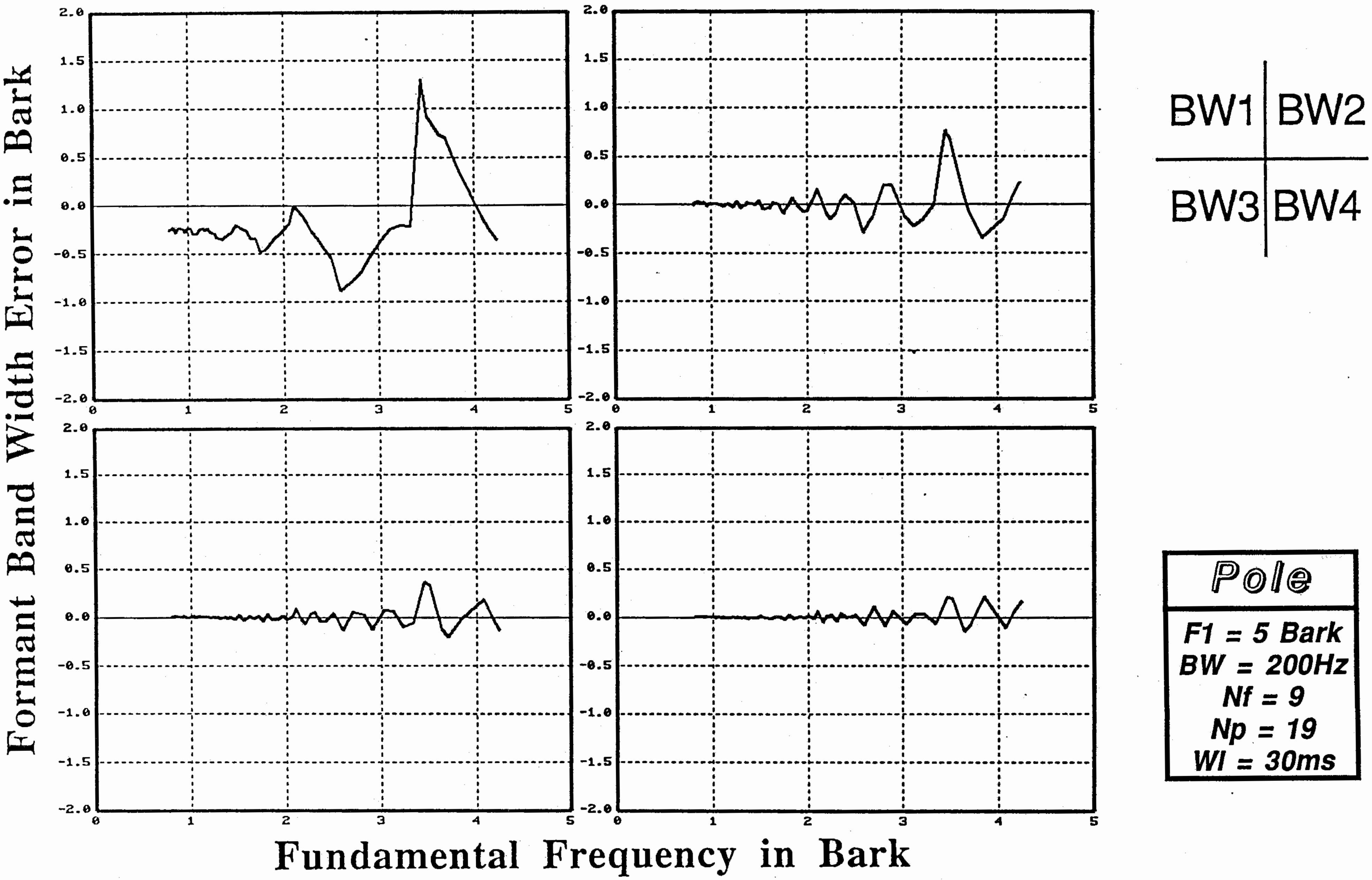


図4.1.28(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

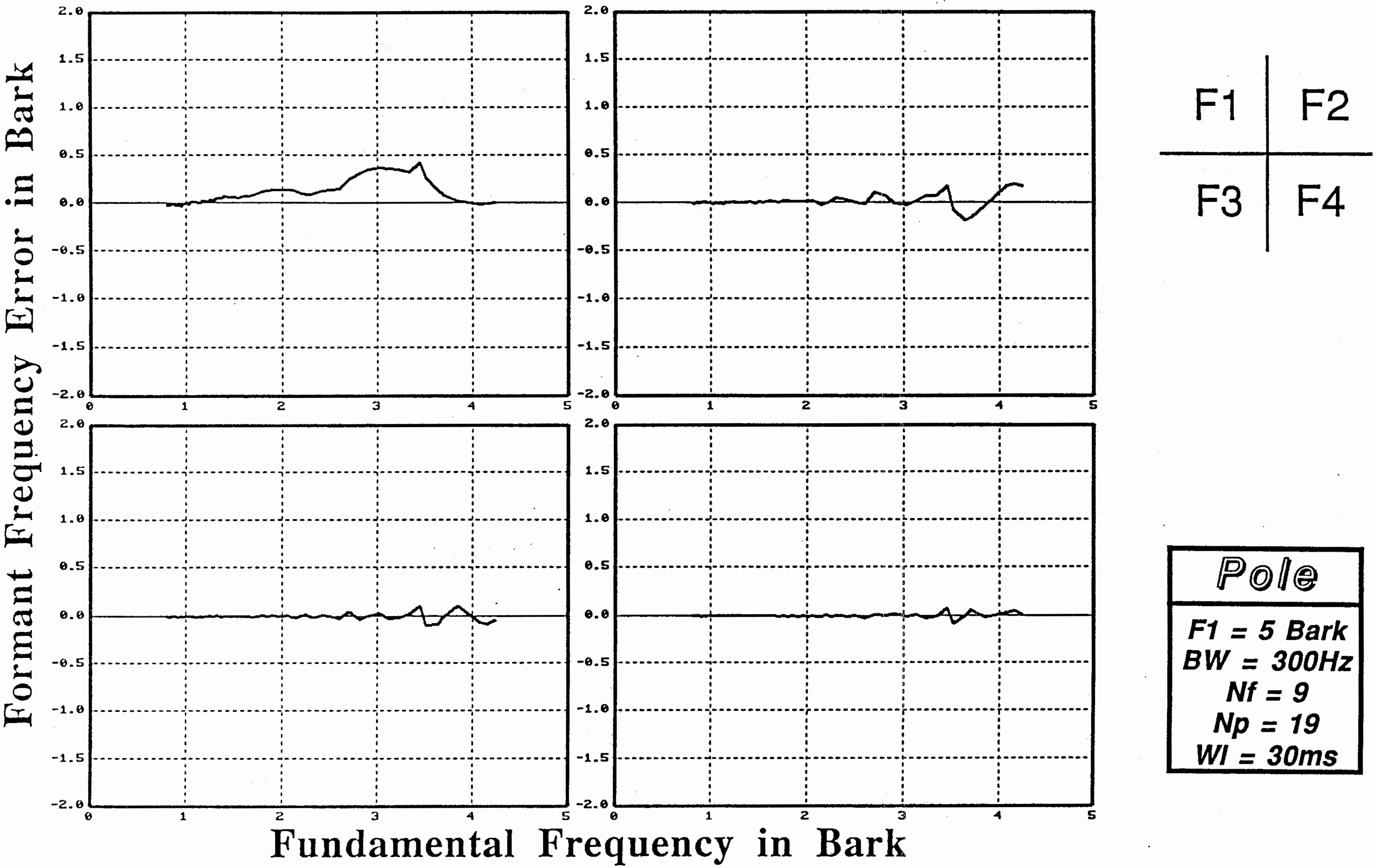


図4.1.29(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

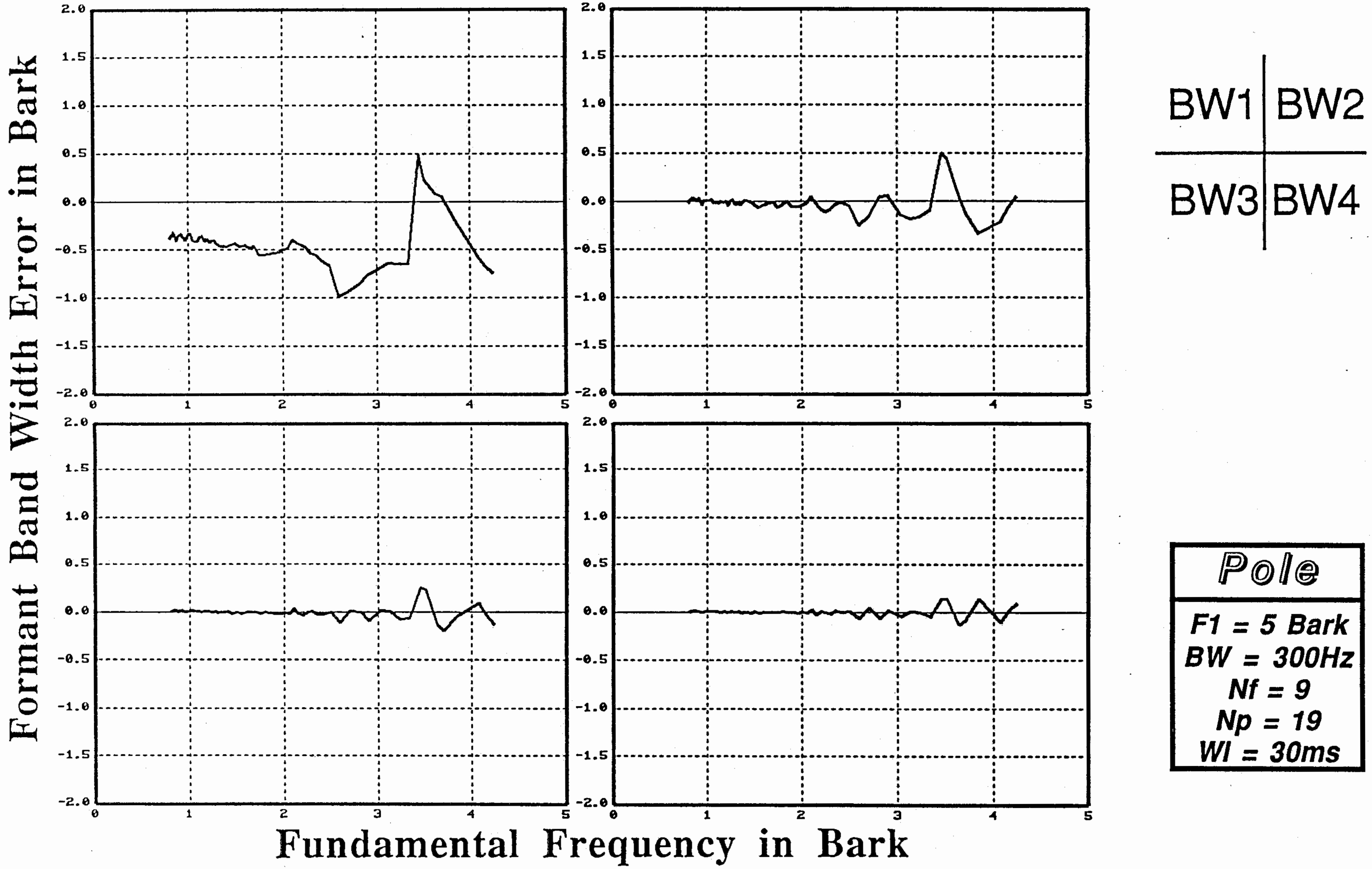


図4.1.29(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

4.2 スペクトル包絡のピーク抽出の結果

本節では、ピーク抽出法を用いてフォルマント周波数、フォルマント帯域幅を推定する際に、分析パラメータと資料の合成パラメータとの関係が、推定値にどの様な影響を及ぼすか調べた結果を示す。分析条件は全て前節と同様である。

4.2.1 分析窓の影響

本項では、用いる分析窓の長さおよび位置が、推定値にどの様な影響を及ぼすか調べた結果を示す。

(i) 窓長

図4.2.1～図4.2.3は、フォルマント周波数とフォルマント帯域幅の推定誤差の絶対値を分析窓長Wlを横軸にして描いたものである。各図とも左側の列がフォルマント周波数の推定誤差の絶対値を、右側の列がフォルマント帯域幅の推定誤差の絶対値を描いたものであり、左右の列とも上から順にF0が低いものから高いものへと並んでいる。

図4.2.1、図4.2.2、図4.2.3は、それぞれF1=3Bark、5Bark、8Barkの合成中性母音に対する結果である。

(ii) 窓位置

図4.2.4～図4.2.6はフォルマント周波数とフォルマント帯域幅の推定誤差の絶対値を分析窓位置Wpを横軸にして描いたものである。各図とも左側の列がフォルマント周波数の推定誤差の絶対値を、右側の列がフォルマント帯域幅の推定誤差の絶対値を描いたものであり、左右の列とも上から順にF0が低いものから高いものへと並んでいる。

図4.2.4、図4.2.5、図4.2.6は、それぞれF1=3Bark、5Bark、8Barkの合成中性母音に対する結果である。

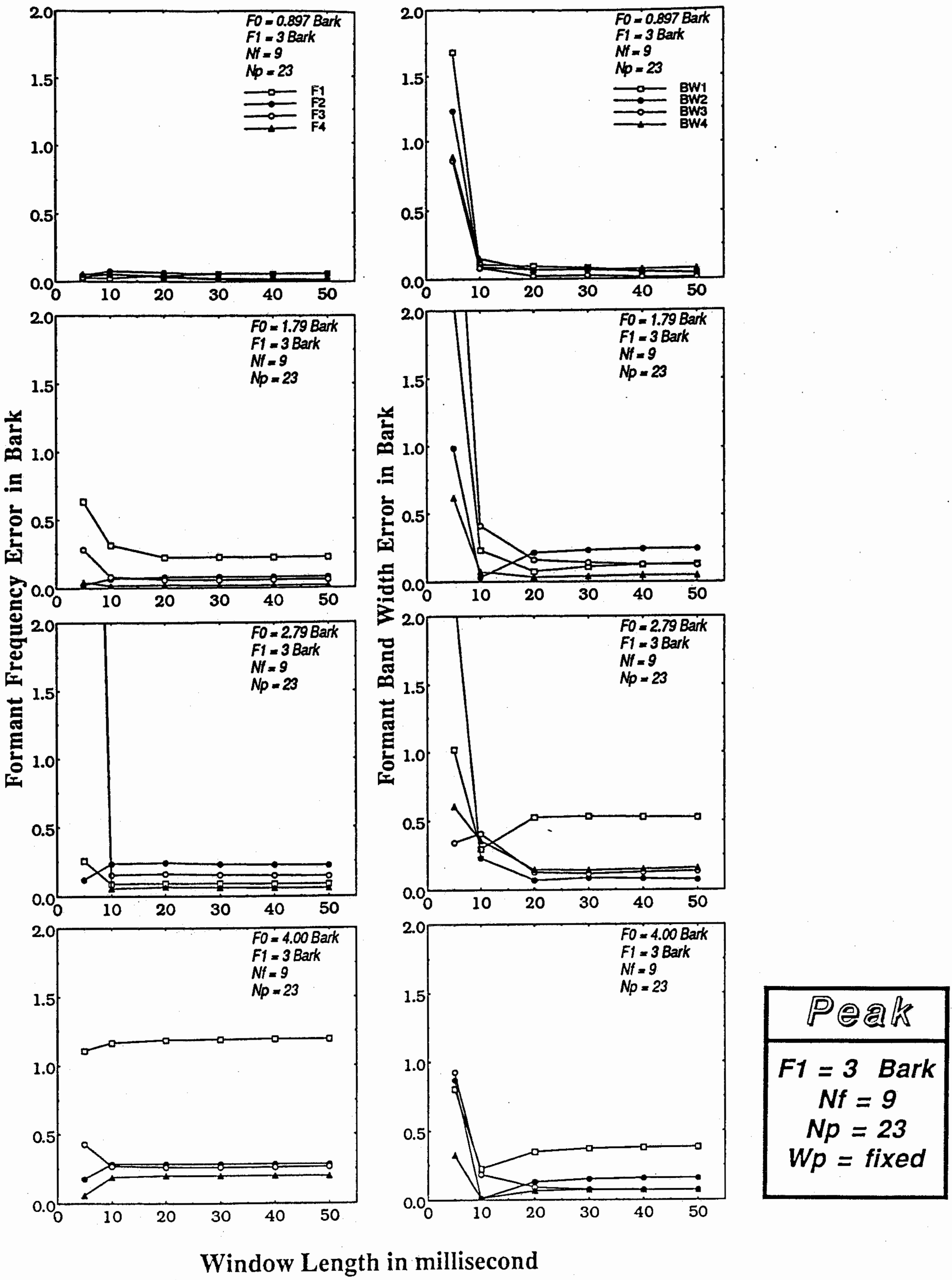


図4.2.1 フォルマント推定誤差の絶対値と分析窓長との関係

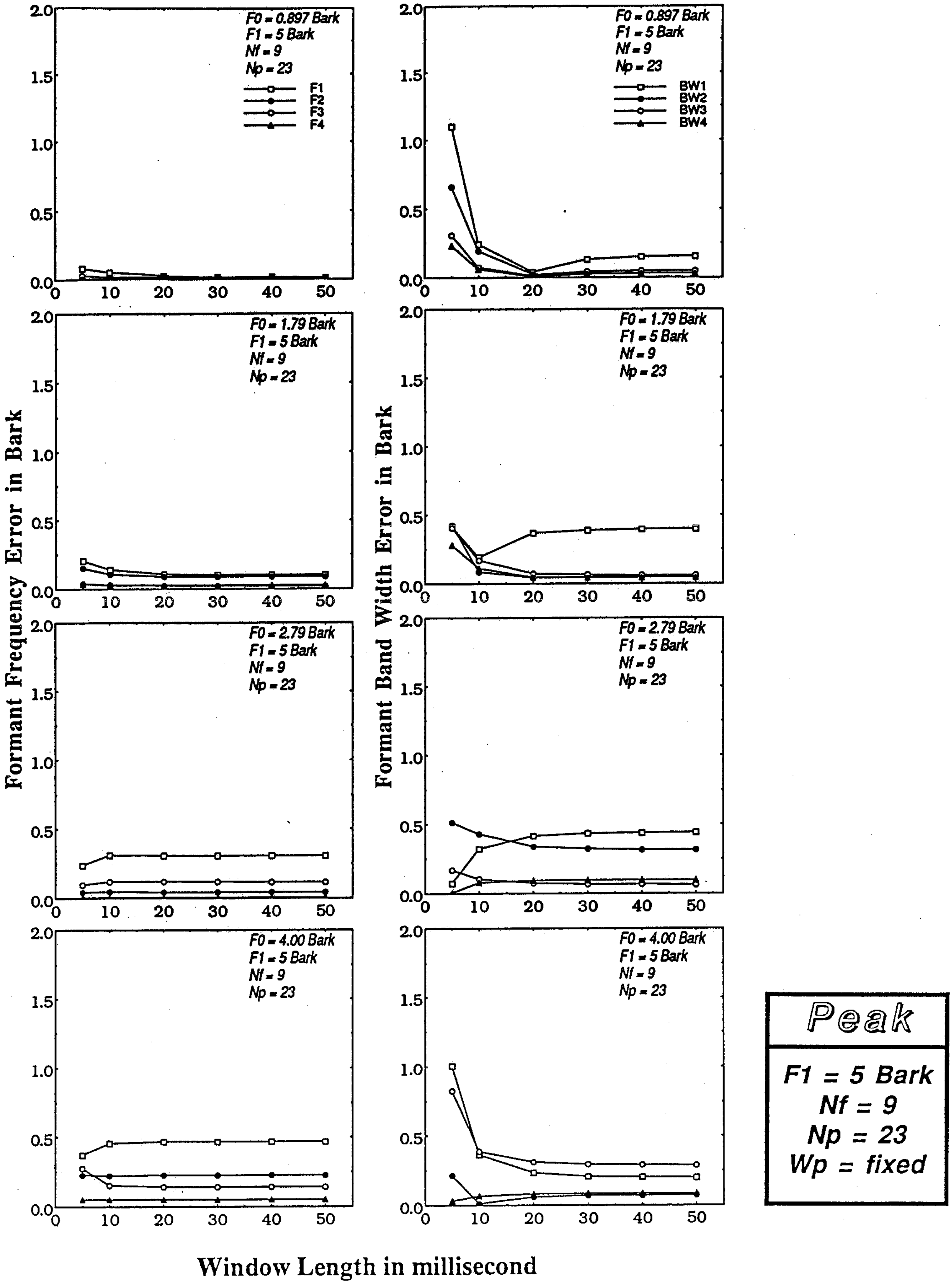


図4.2.2 フォルマント推定誤差の絶対値と分析窓長との関係

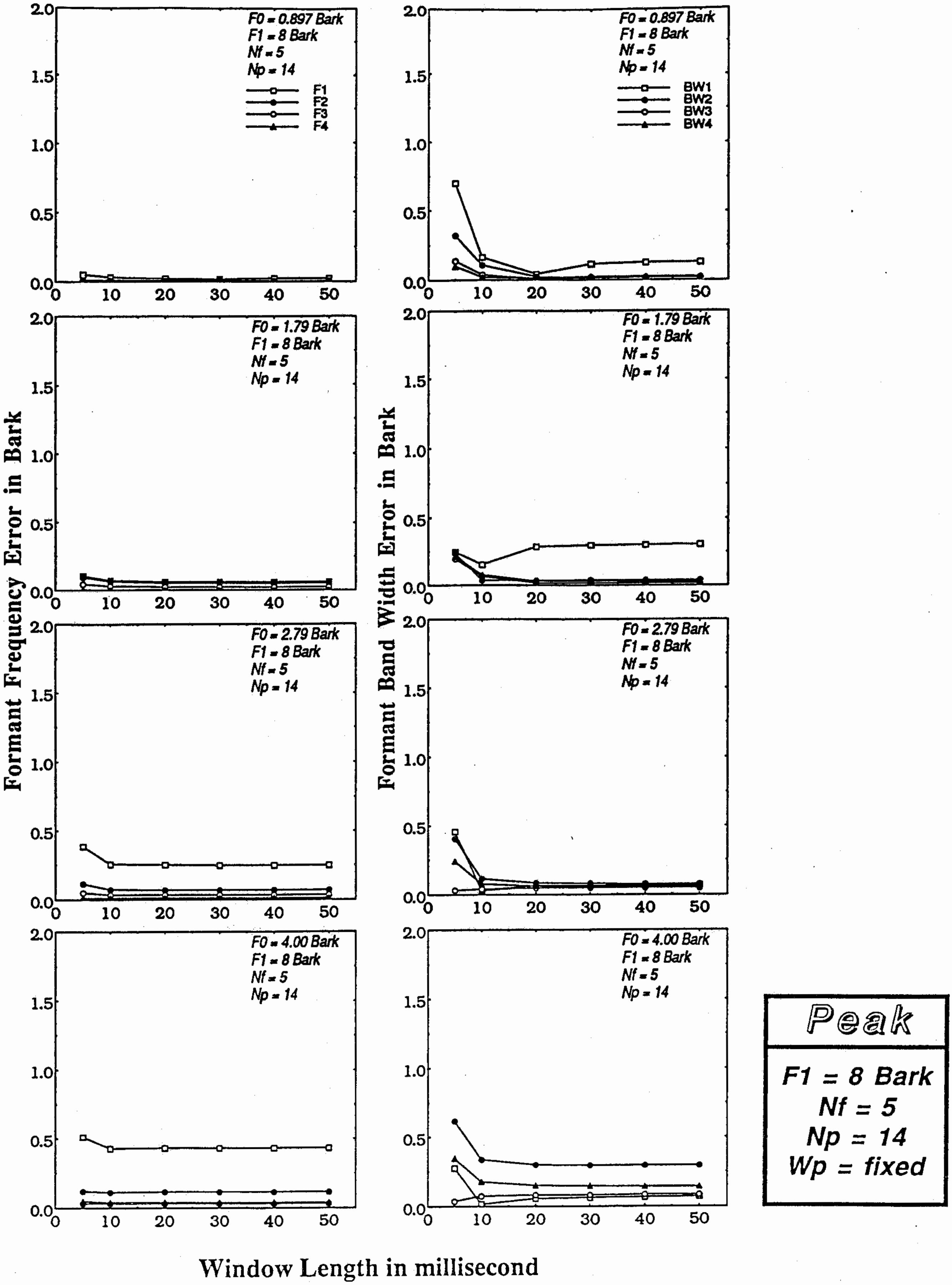


図4.2.3 フォルマント推定誤差の絶対値と分析窓長との関係

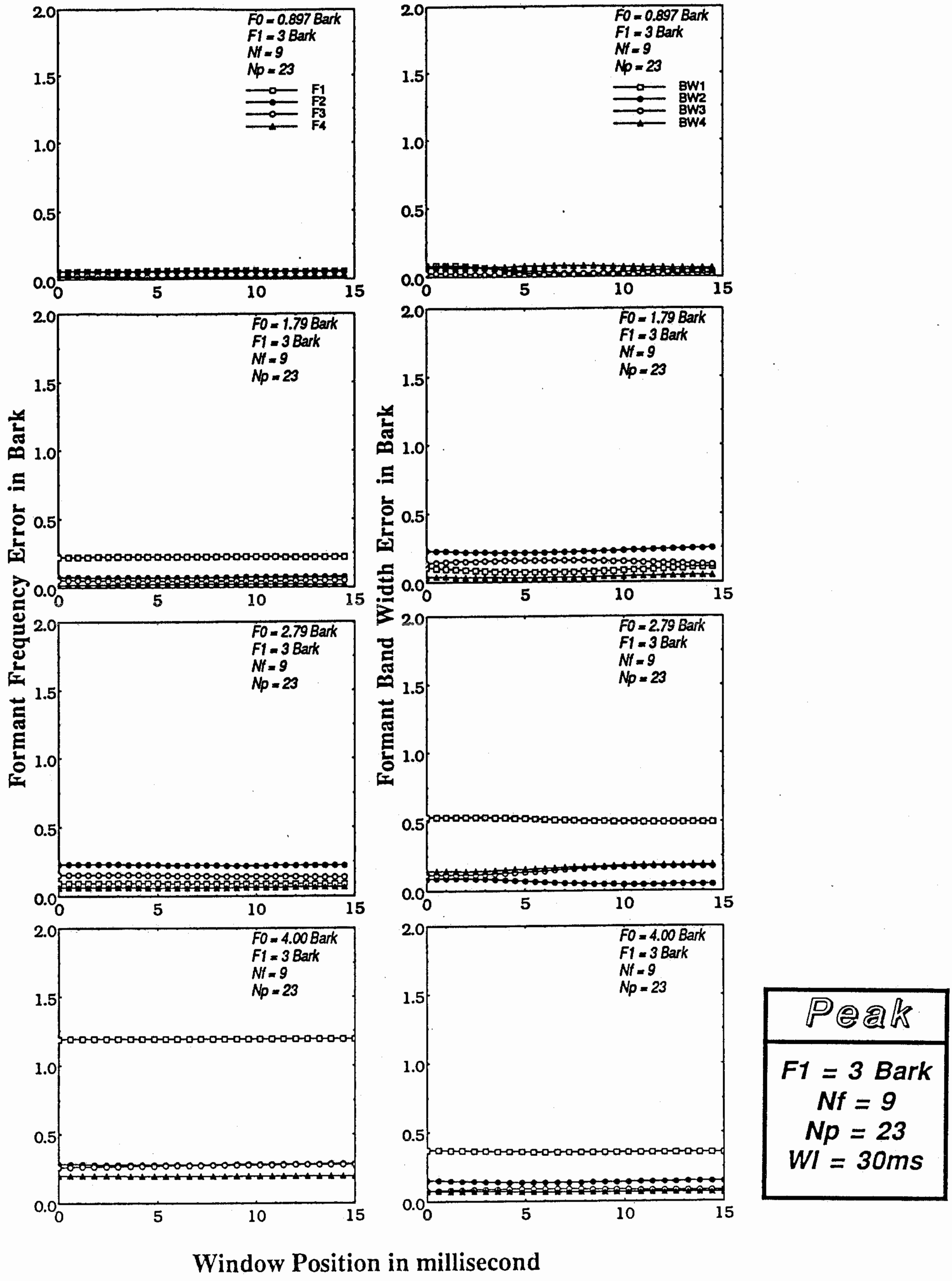


図4.2.4 フォルマント推定誤差の絶対値と分析窓位置との関係

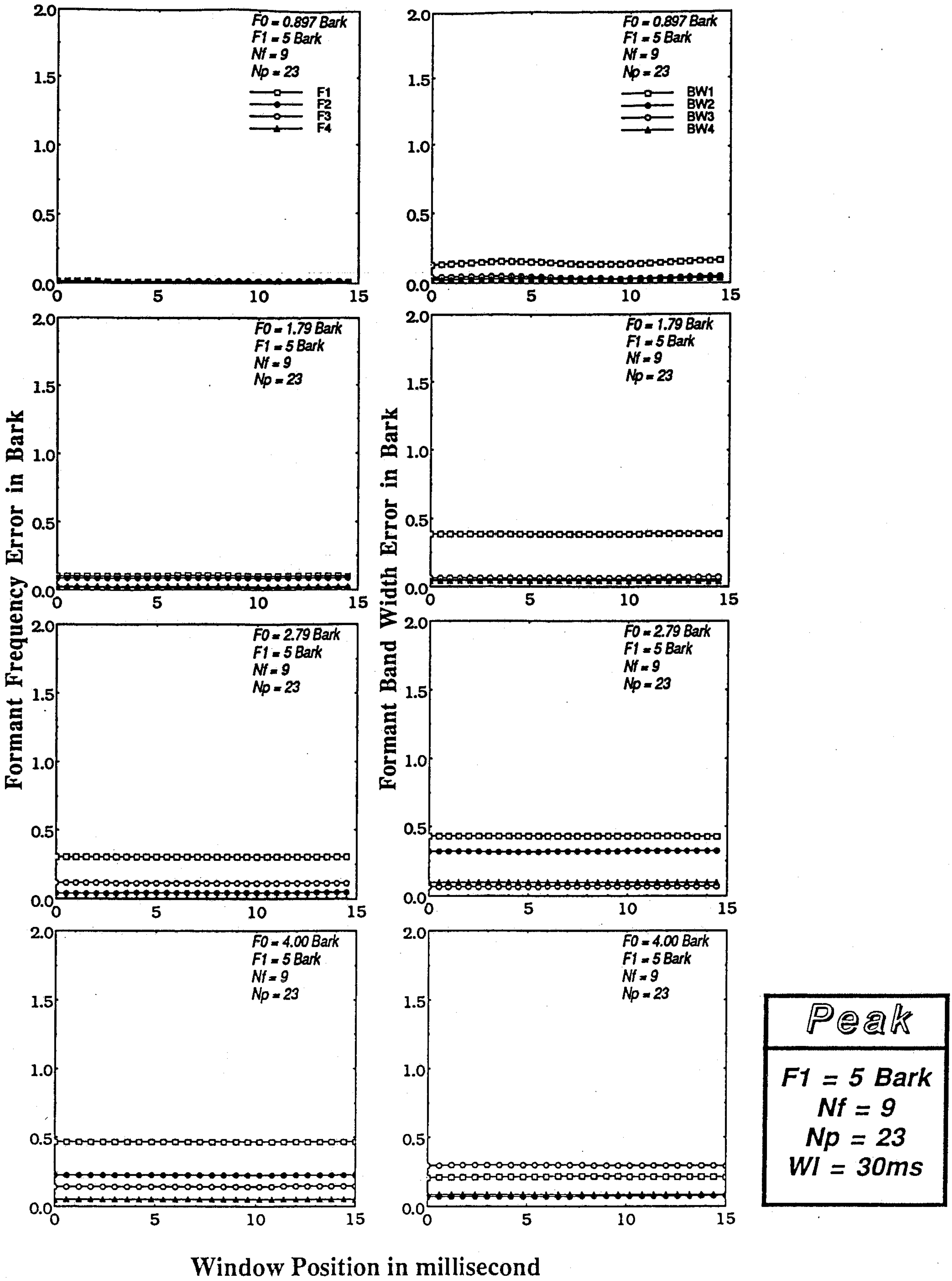


図4.2.5 フォルマント推定誤差の絶対値と分析窓位置との関係

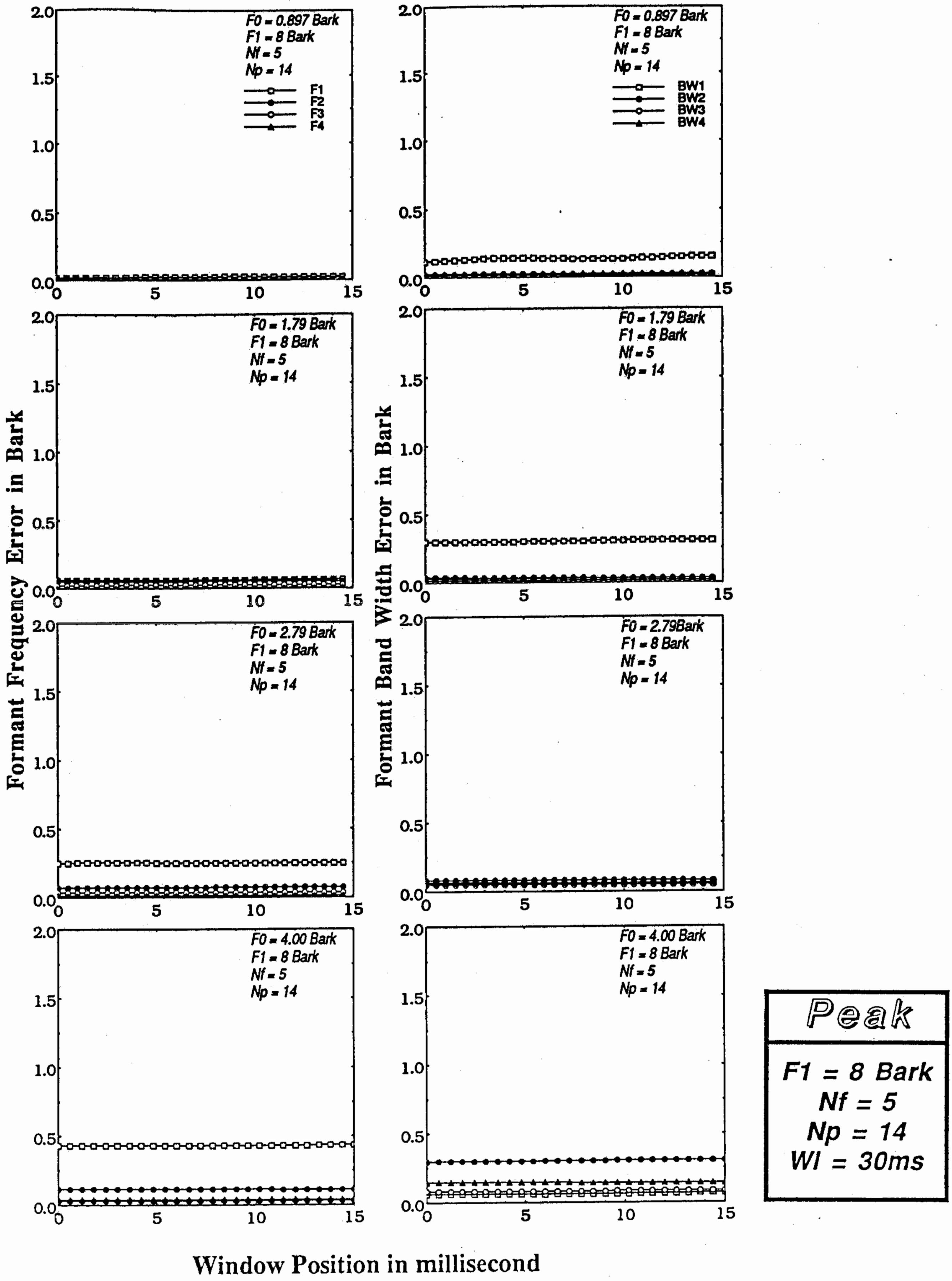


図4.2.6 フォルマント推定誤差の絶対値と分析窓位置との関係

4.2.2 分析次数の影響

本節では、分析の次数がフォルマント周波数、フォルマント帯域幅の推定値に及ぼす影響を調べた結果を示す。

図4.2.7(a)～図4.2.13(a)は、フォルマント周波数の推定誤差の絶対値を分析次数Npを横軸にして描いたものである。各図とも左上から右下へ資料のF0が高いものから低いものへと並んでいる。
図4.2.7(b)～図4.2.13(b)は、フォルマント帯域幅の推定誤差の絶対値について同様に求めたものである。
図4.2.7、図4.2.8、図4.2.9は、F1=3 Barkでフォルマントの個数Nfがそれぞれ5個、7個、9個の資料に対する結果である。
図4.2.10、図4.2.11、図4.2.12は、F1=5 BarkでNf=5、7、9の資料に対する結果である。
図4.2.13は、F1=8 BarkでNf=5の資料に対する結果である。

図4.2.14は、フォルマント周波数の推定誤差の絶対値がある閾値E以下であるような分析次数の上限と下限を、資料のF0を横軸にして描いたものである。この図ではE=0.5 Barkに設定した。第1段、第2段、第3段はそれぞれF1=3 Bark、5 Bark、8 Barkの資料に対する結果を示し、第1列、第2列、第3列はそれぞれNf=5、7、9の資料に対する結果を示す。

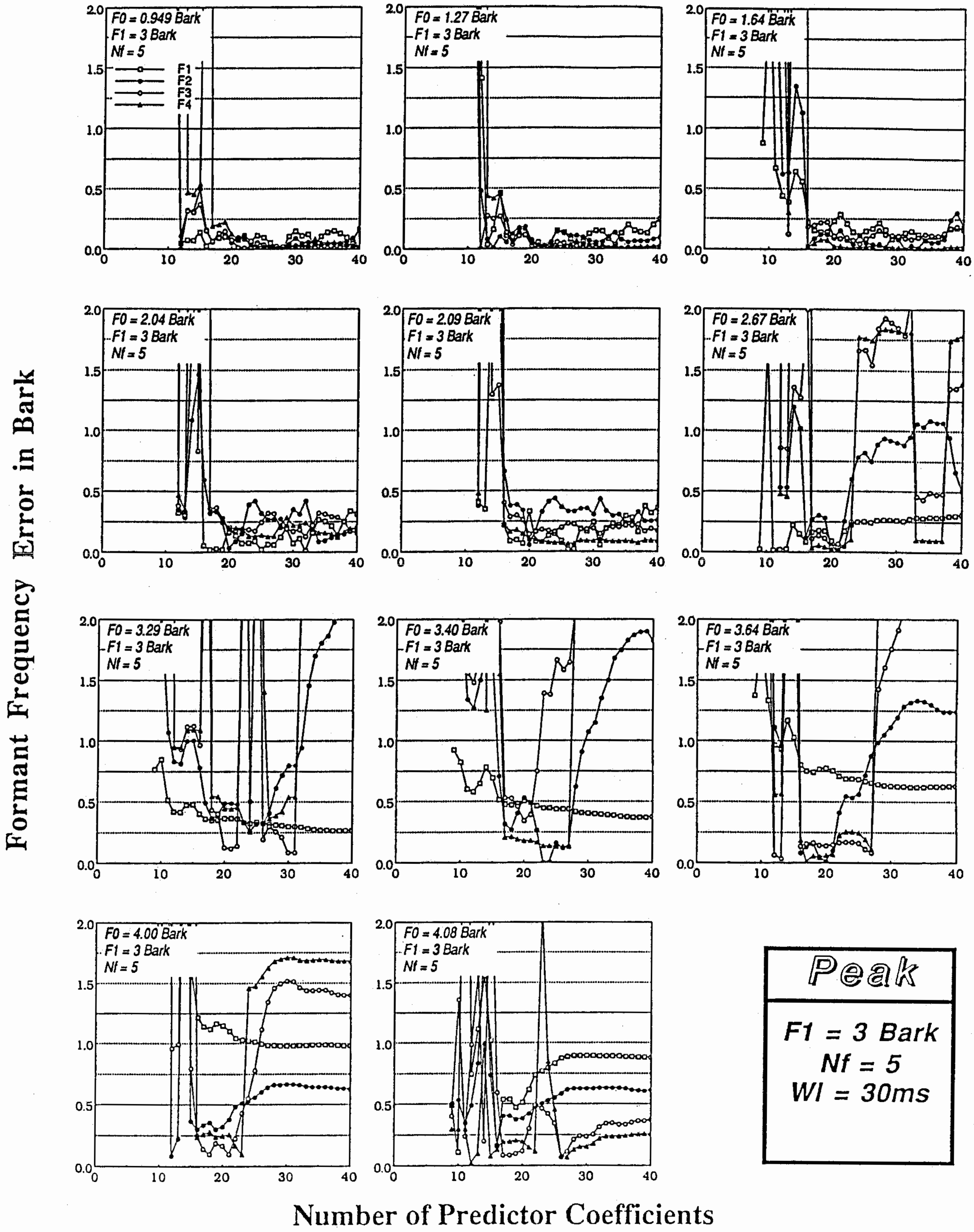


図4.2.7(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

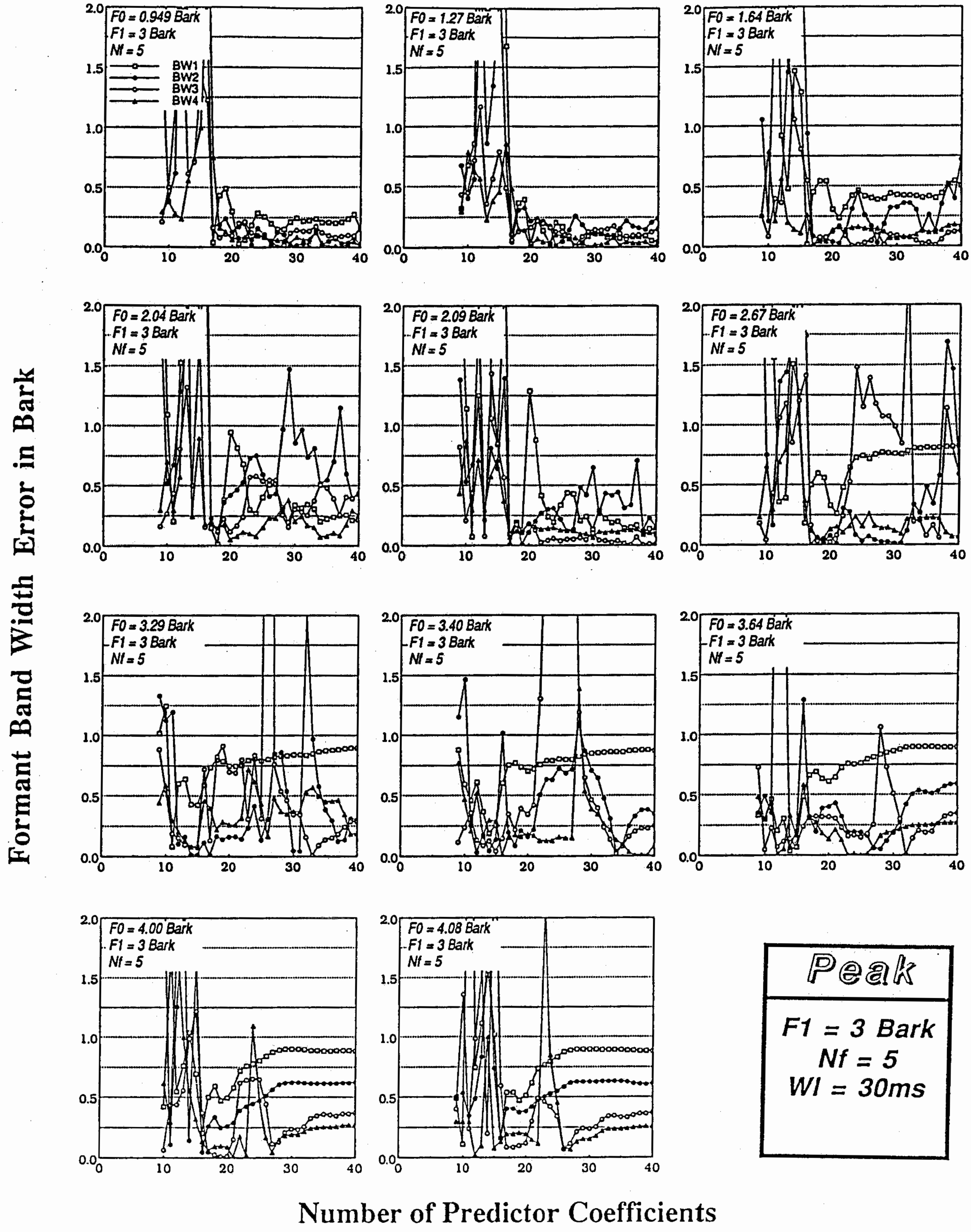


図4.2.7(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

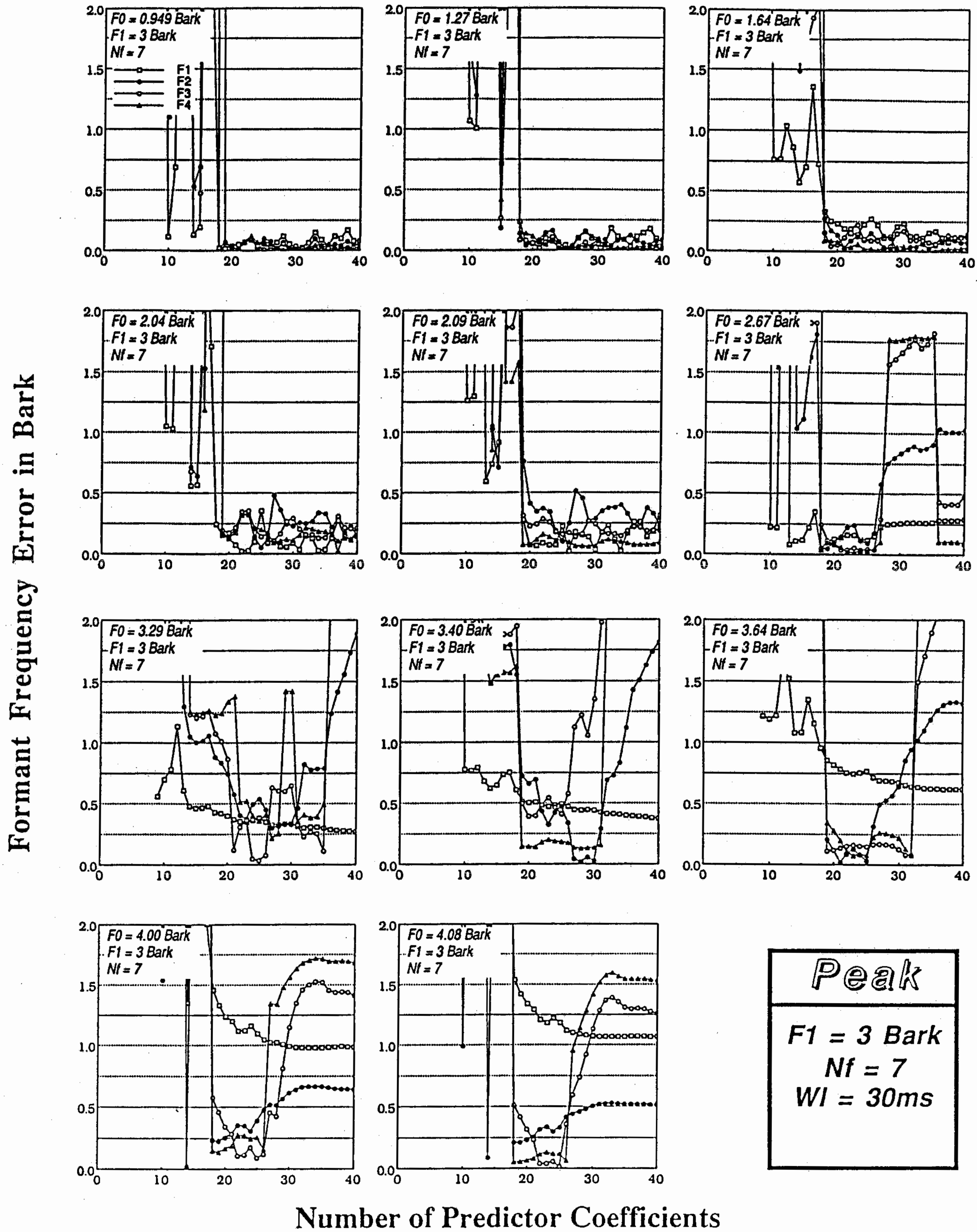


図4.2.8(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

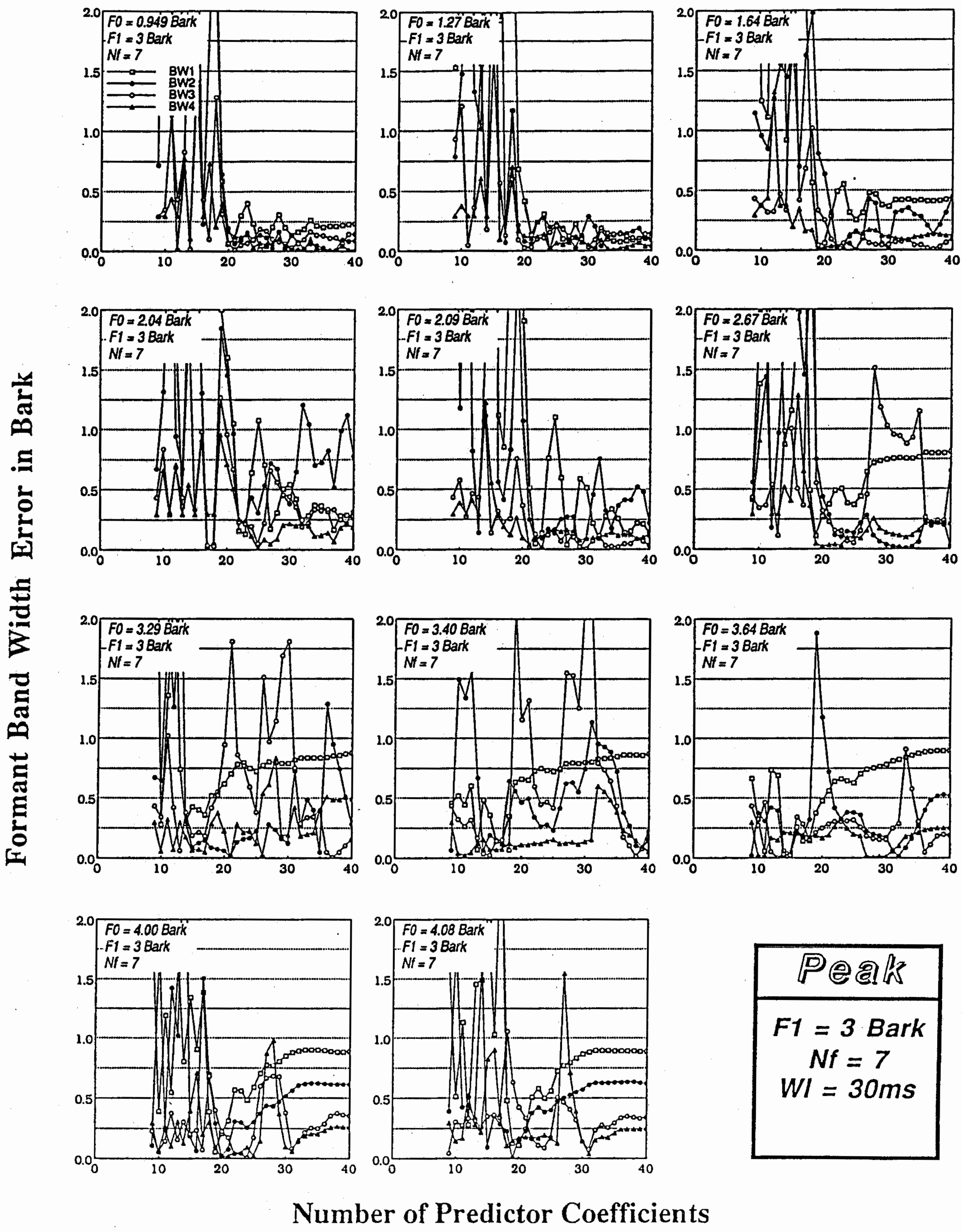


図4.2.8(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

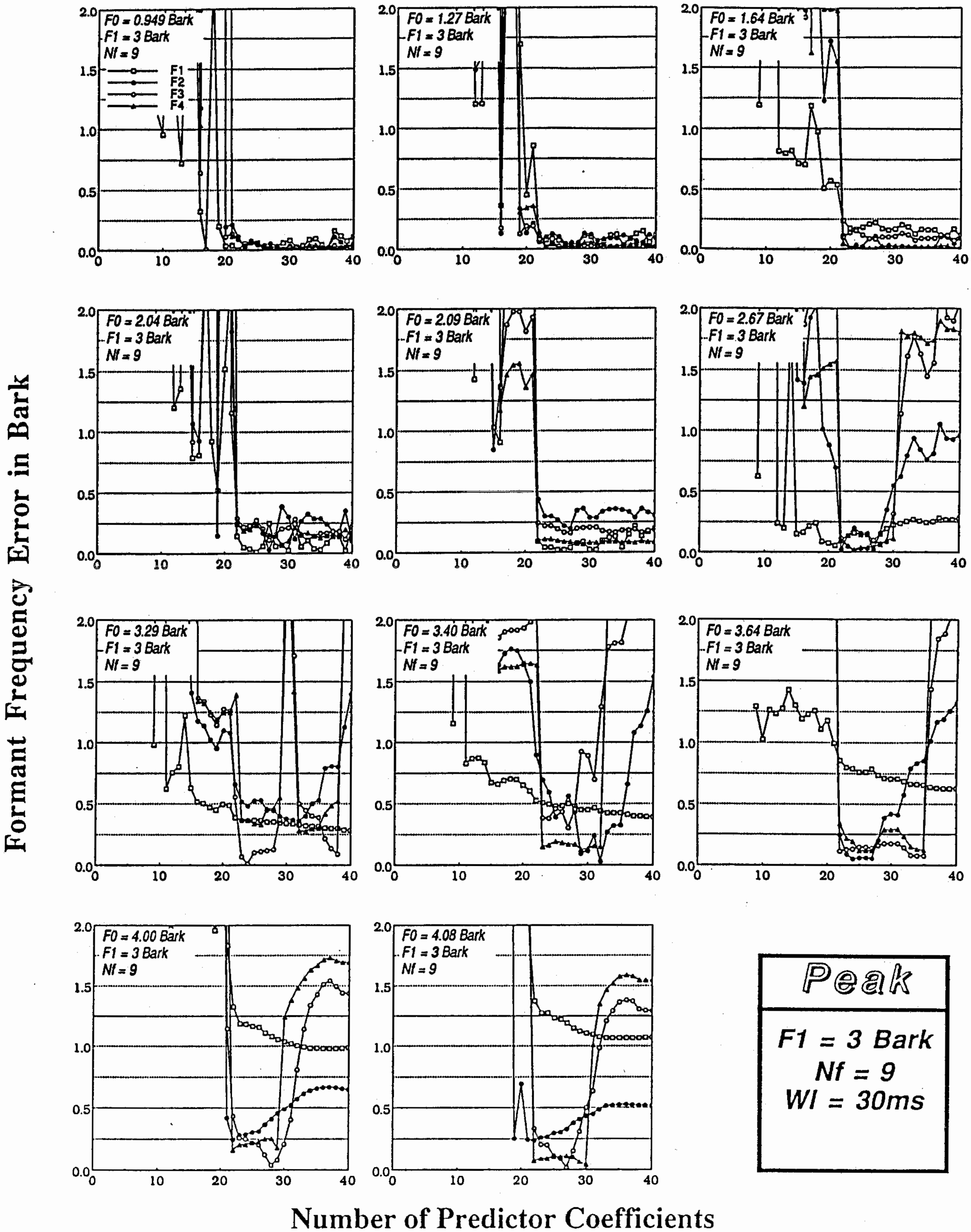


図4.2.9(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

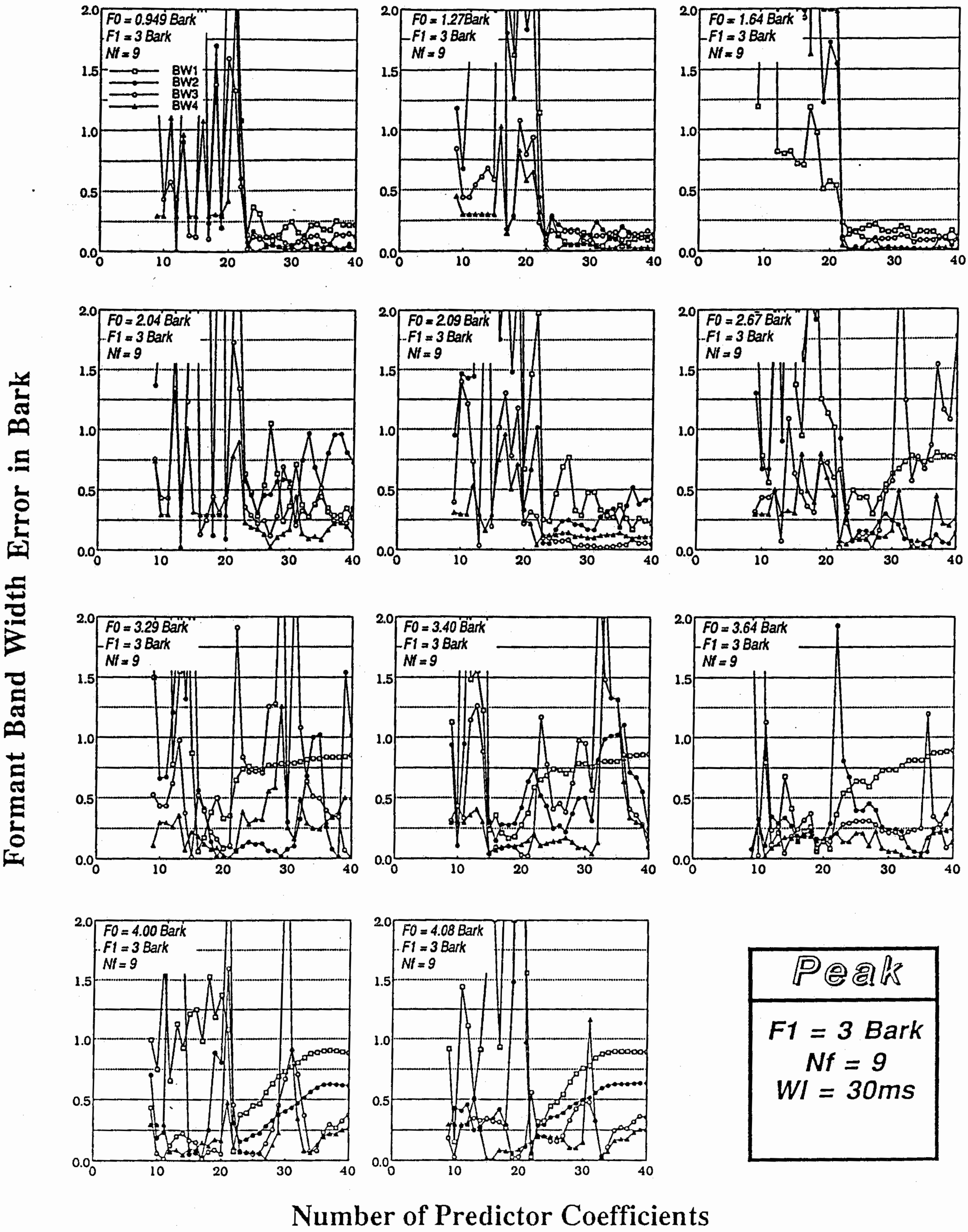


図4.2.9(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

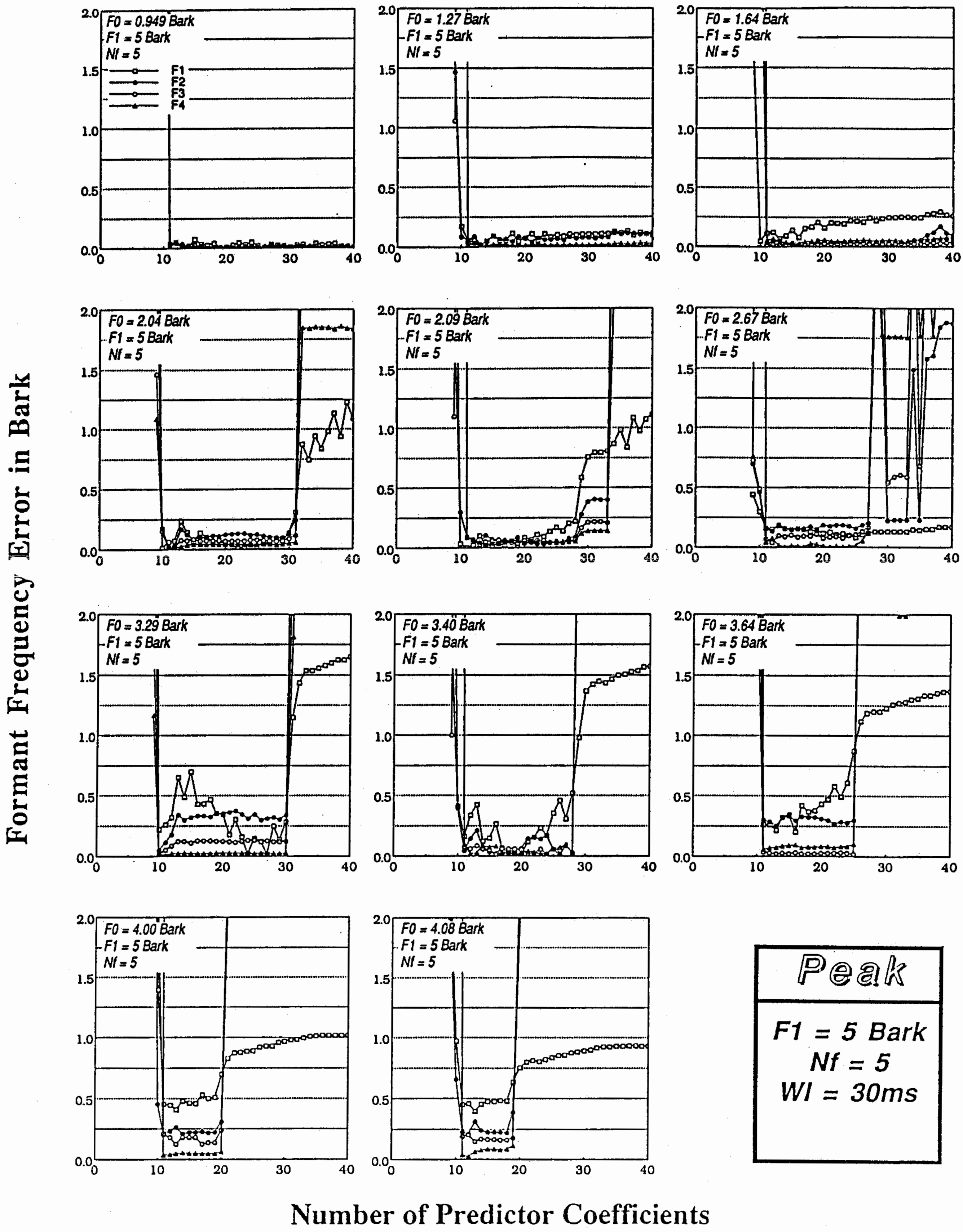


図4.2.10(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

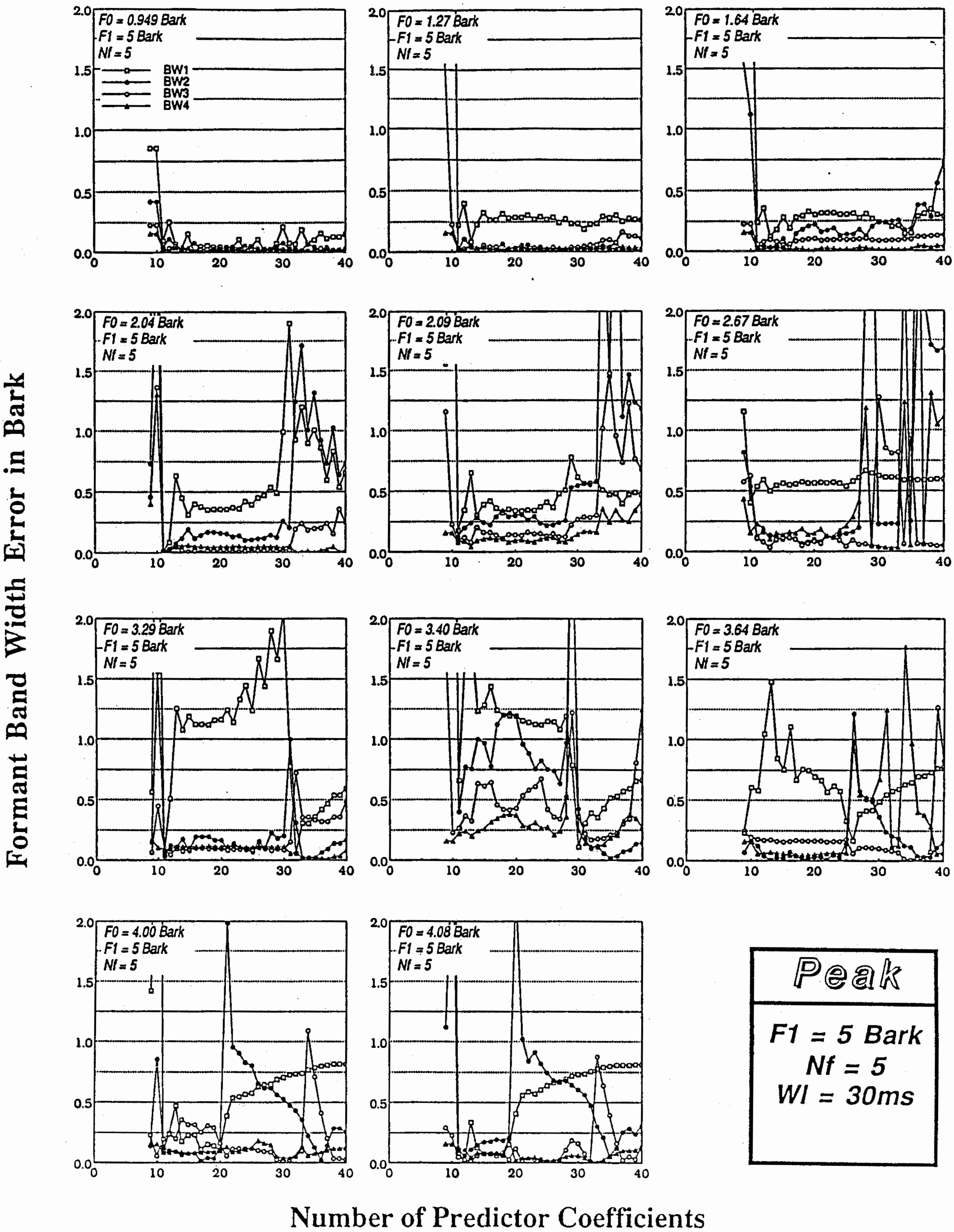


図4.2.10(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

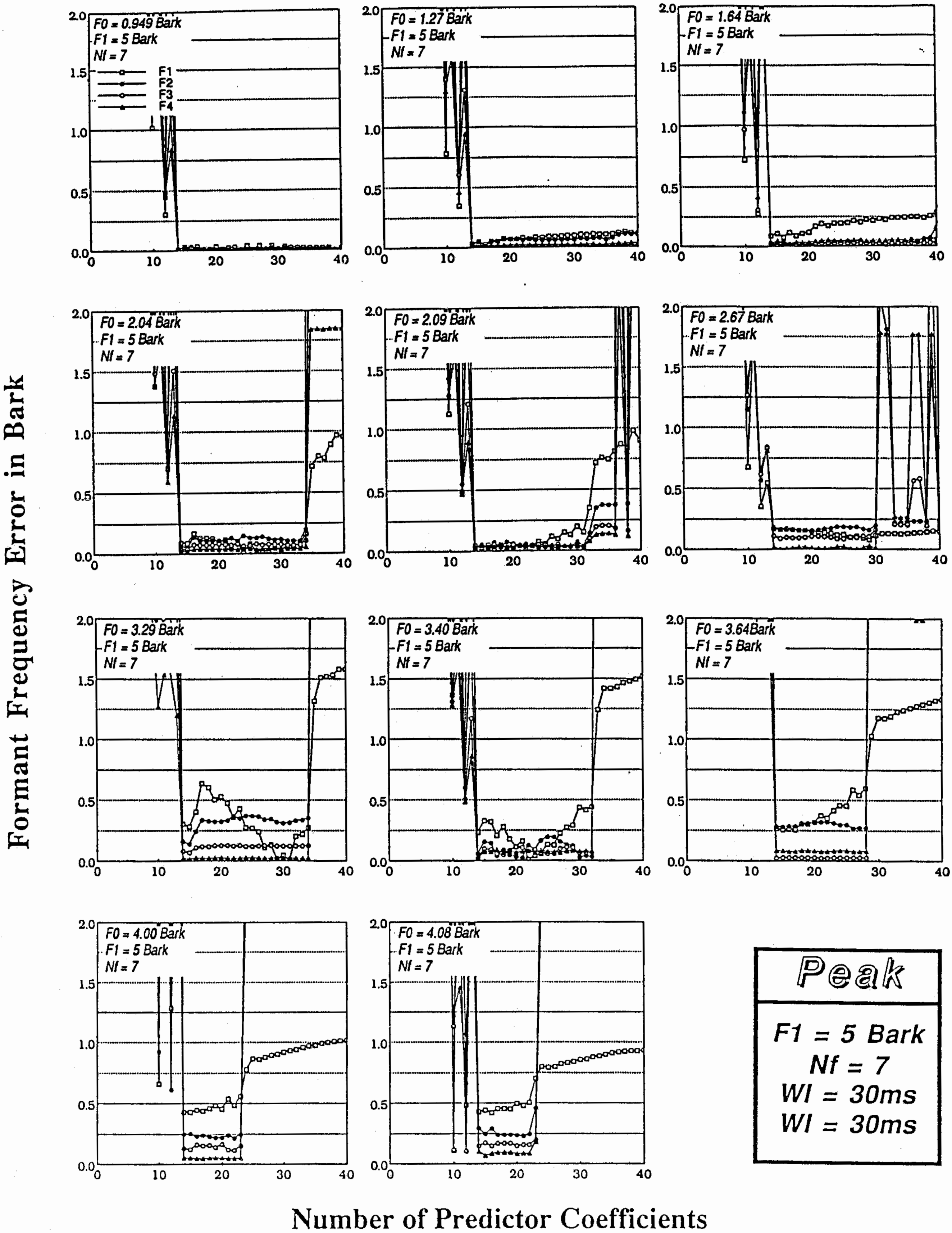


図4.2.11(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

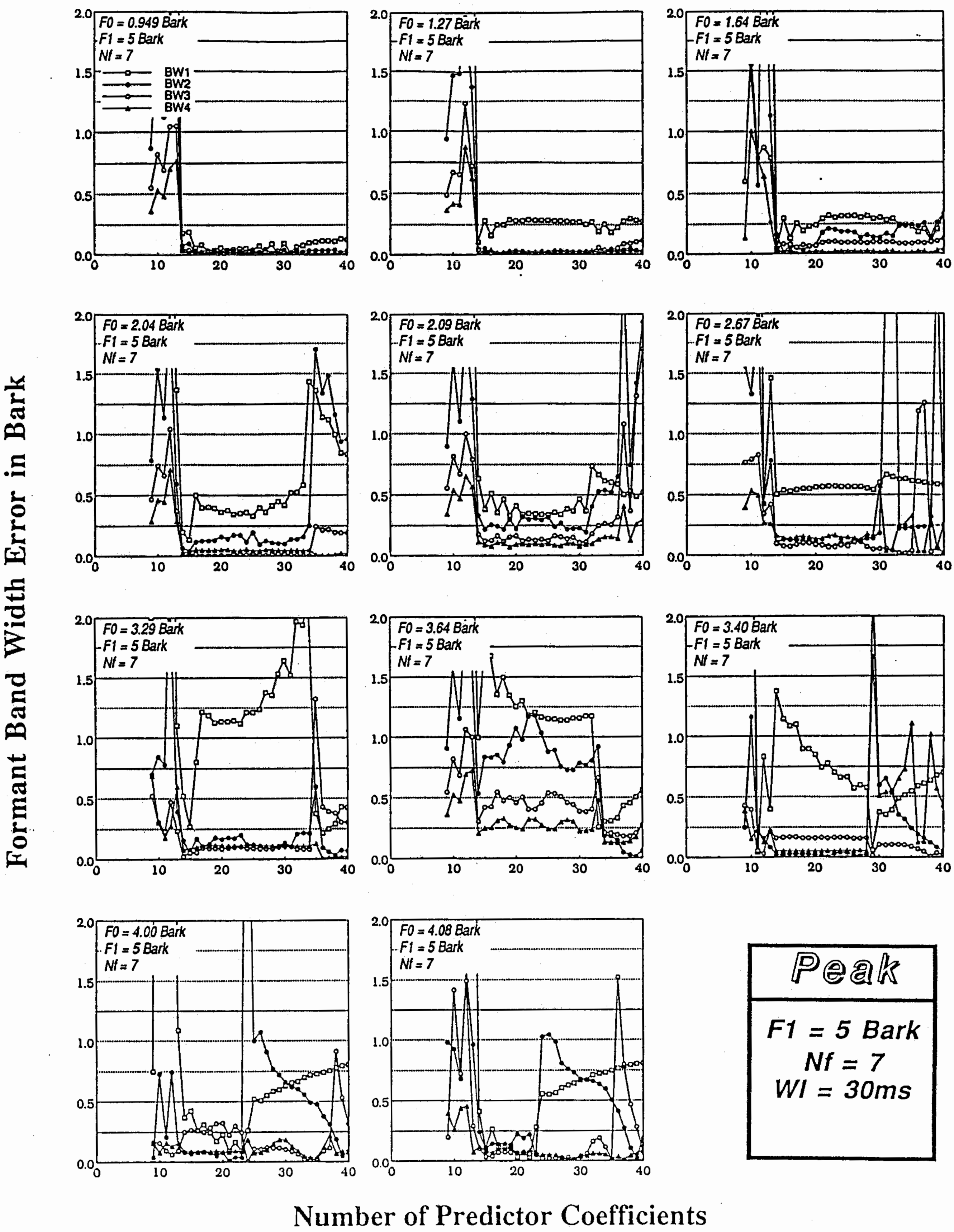


図4.2.11(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

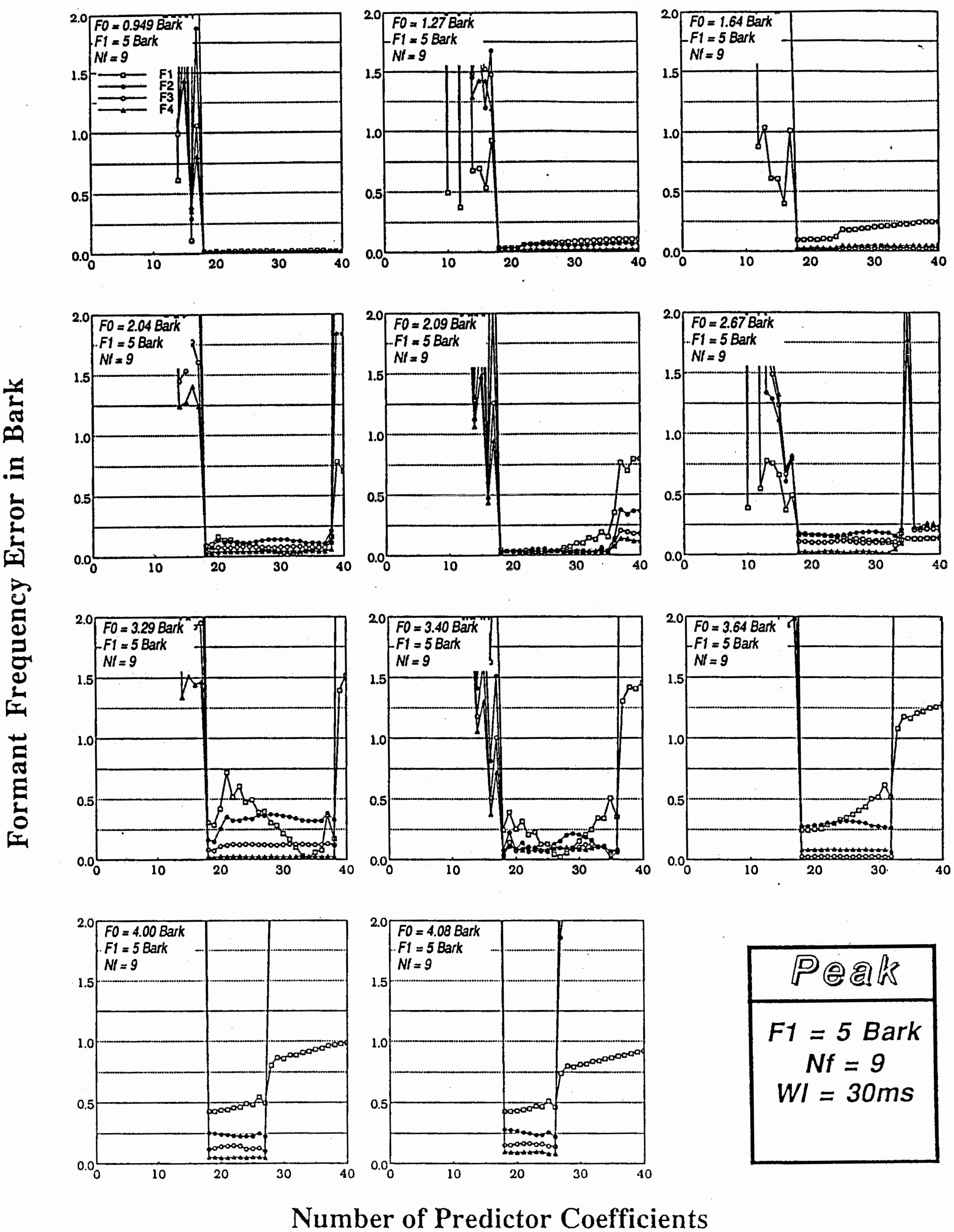


図4.2.12(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

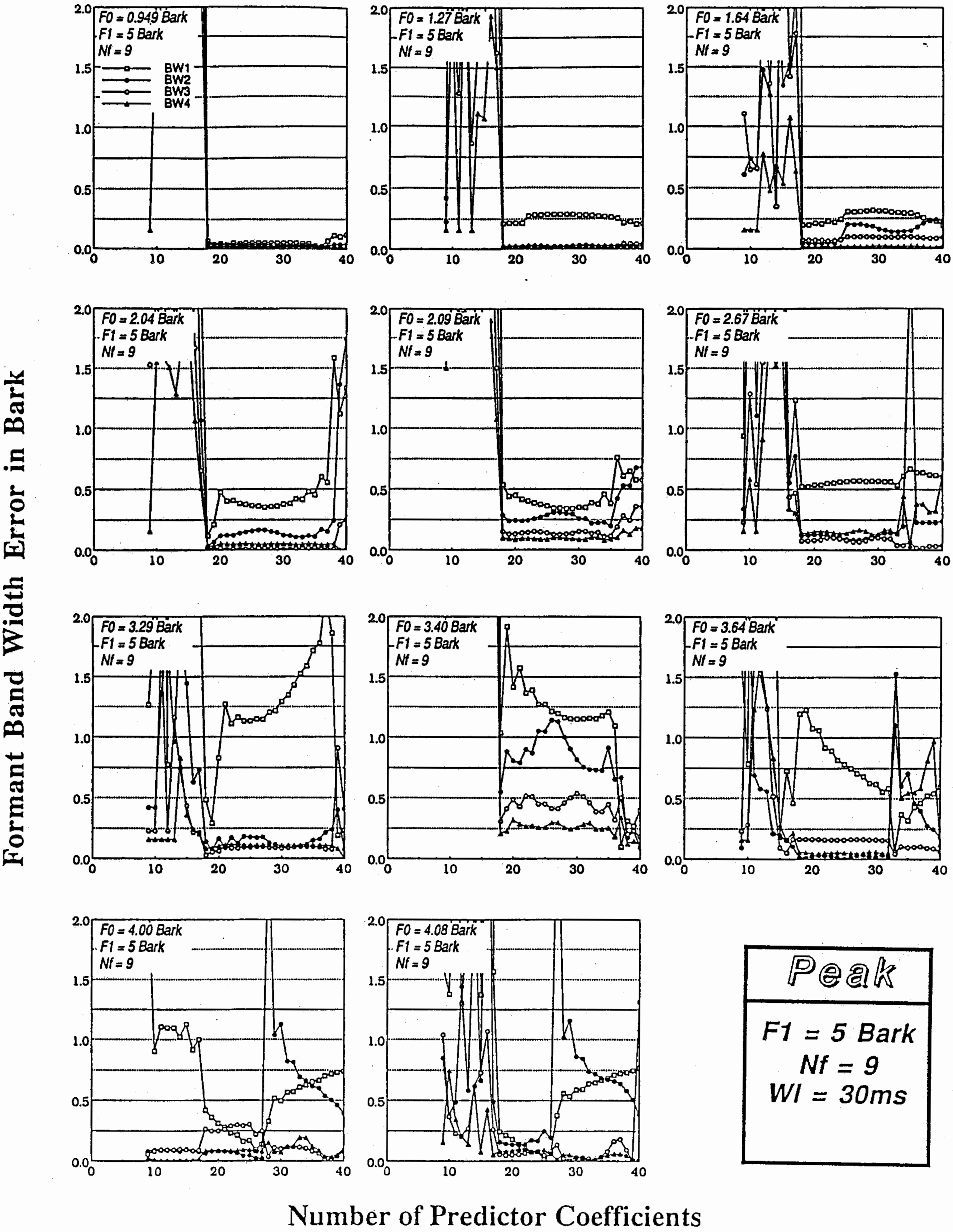


図4.2.12(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

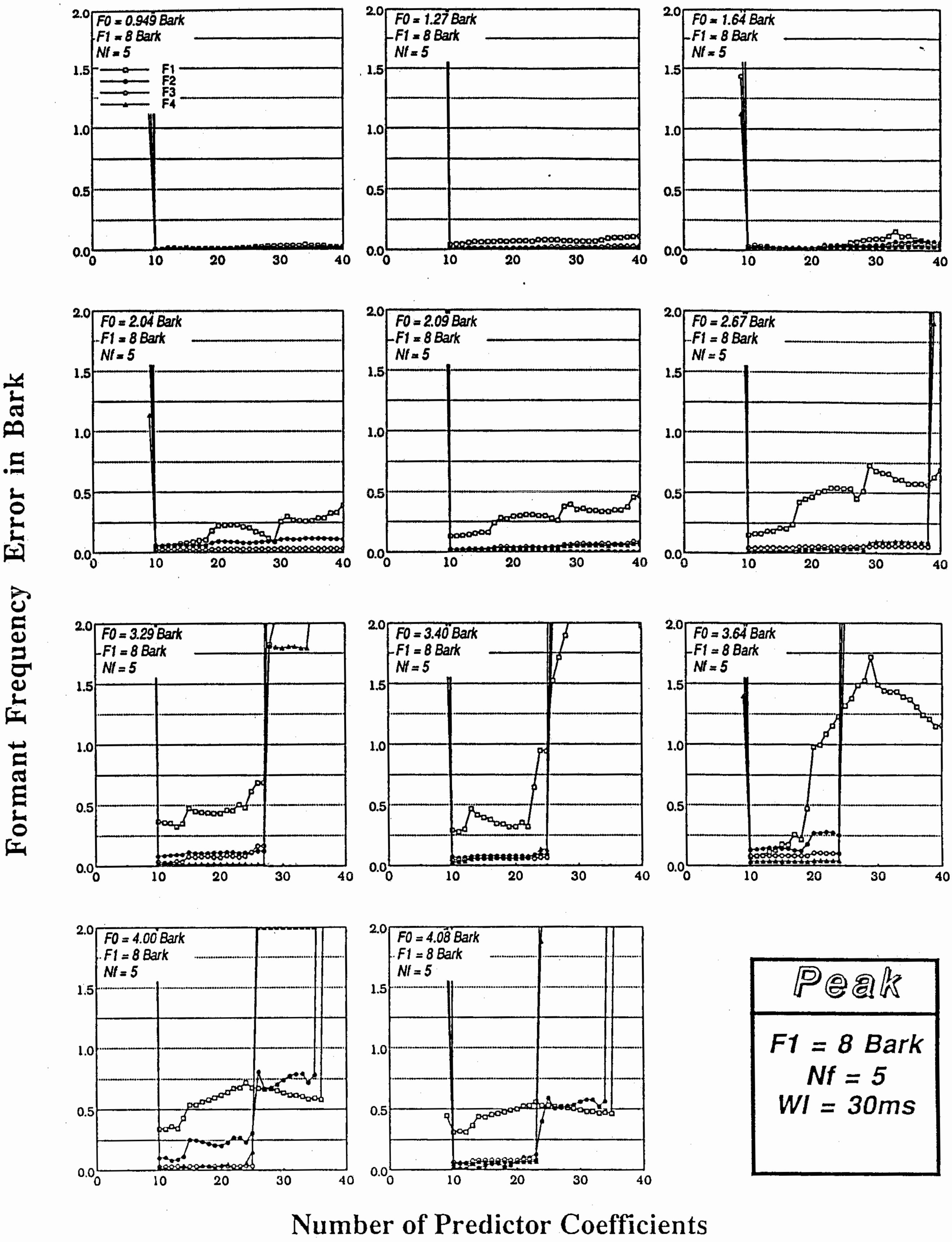


図4.2.13(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（中性母音）

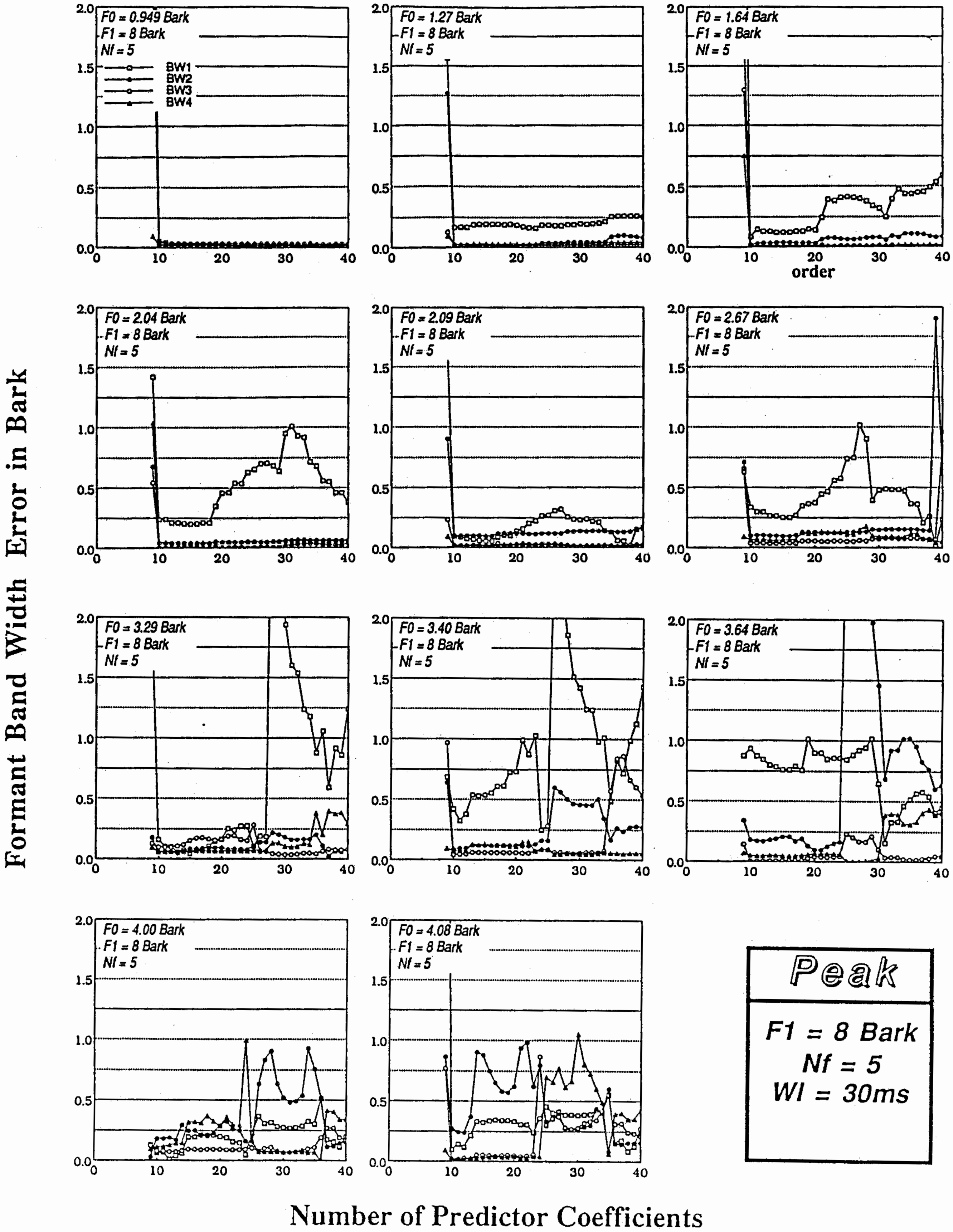


図4.2.13(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（中性母音）

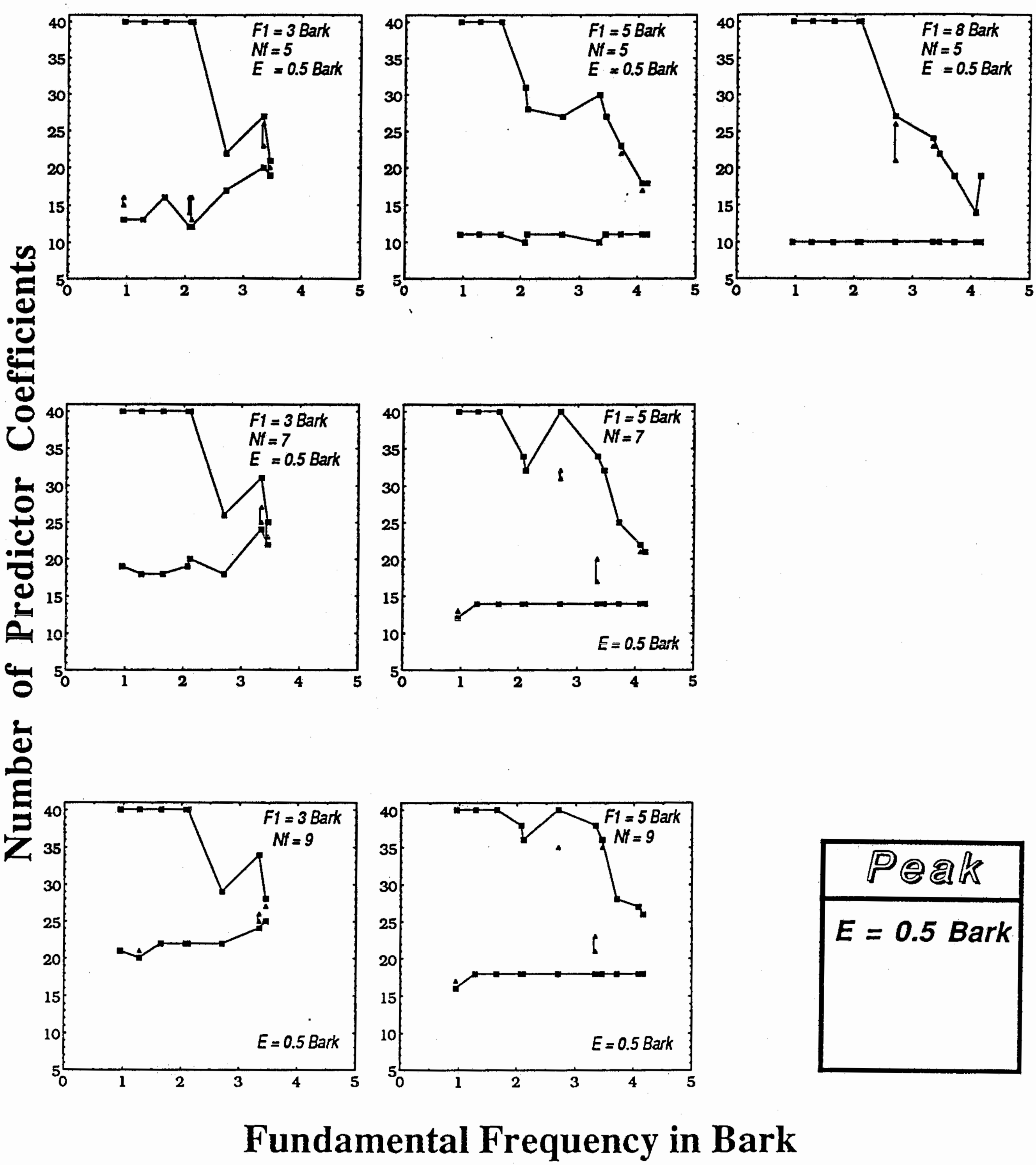


図4.2.14 フォルマント周波数の推定誤差の絶対値が0.5Bark以下である分析次数の上限、下限と資料の基本周波数との関係（中性母音）

4.2.3 基本周波数、帯域幅の影響

本節では、資料の基本周波数、合成時のフォルマント帯域幅がフォルマント周波数、フォルマント帯域幅の推定値に及ぼす影響を調べた結果を示す。

(i) 基本周波数

図4.2.15(a)～図4.2.17(a)は、フォルマント周波数の推定誤差を資料のF0を横軸にして描いたものである。各図とも左上から右下へ第1～第4フォルマントに対する誤差の図を示す。

図4.2.15(b)～図4.2.17(b)は、フォルマント帯域幅について同様に求めたものである。

図4.2.15、図4.2.16、図4.2.17は、それぞれF1=3 Bark、5 Bark、8 Barkの資料に対する結果を示している。

(ii) 帯域幅

図4.2.18(a)～図4.2.21(a)は、フォルマント周波数の推定誤差を資料のF0を横軸にして描いたものである。各図とも左上から右下へ第1～第4フォルマントに対する誤差の図を示す。

図4.2.18(b)～図4.2.21(b)は、フォルマント帯域幅について同様に求めたものである。

図4.2.18、図4.2.19、図4.2.20、図4.2.21は、F1=5 Barkで、合成時のフォルマント帯域幅がそれぞれ50 Hz、100 Hz、200 Hz、300 Hzの資料に対する結果を示している。

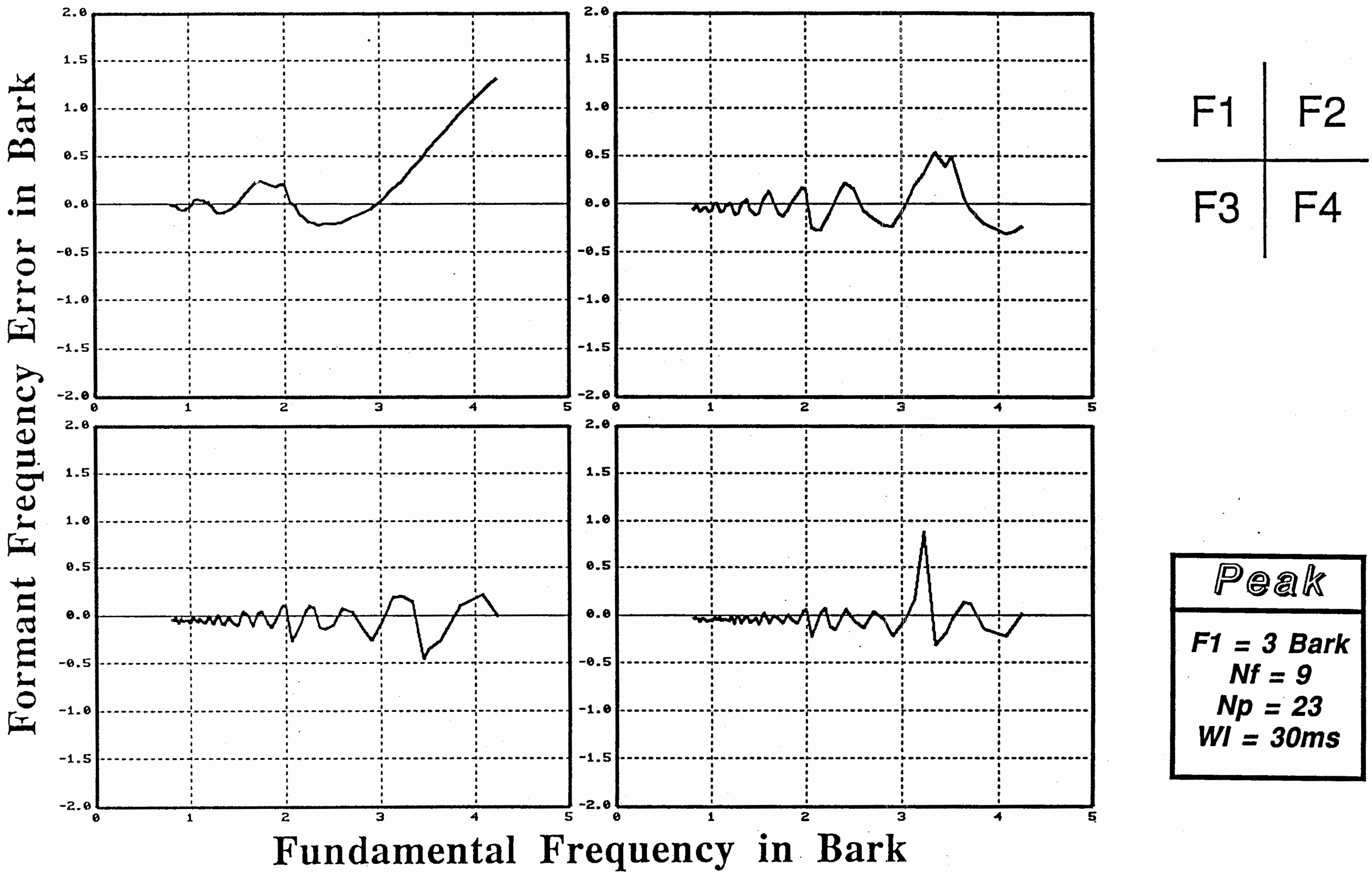


図4.2.15(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

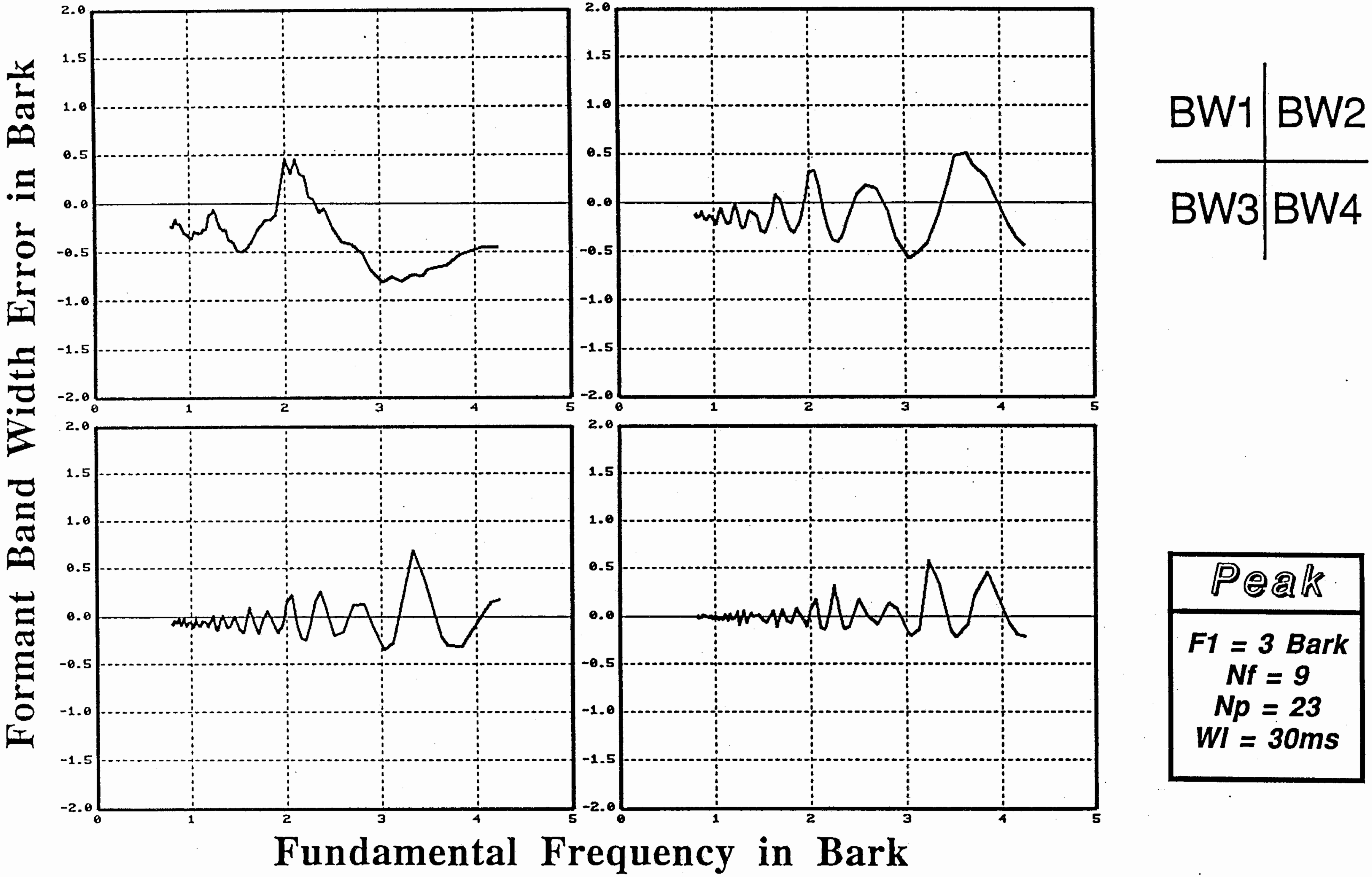


図4.2.15(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

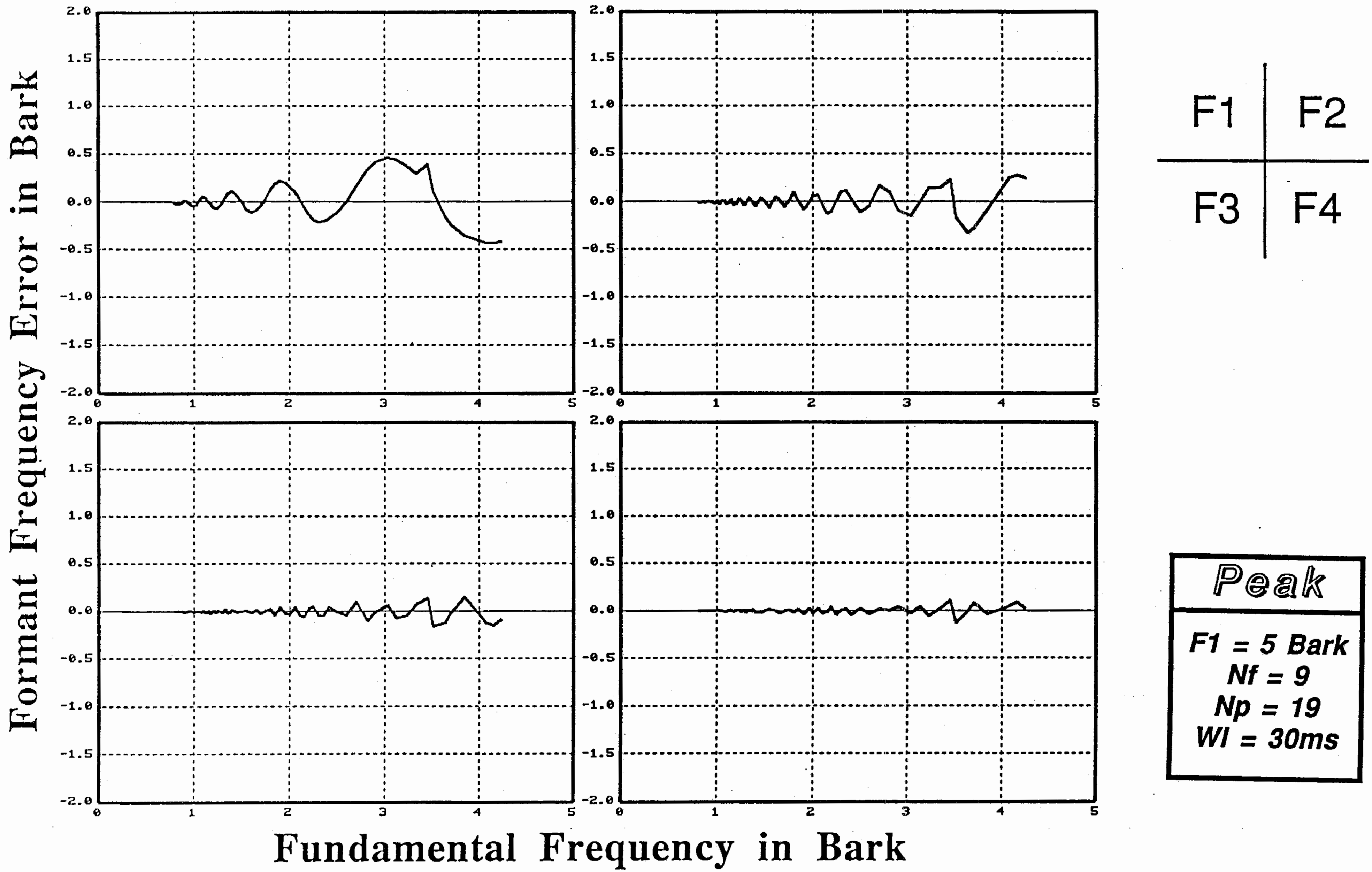


図4.2.16(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

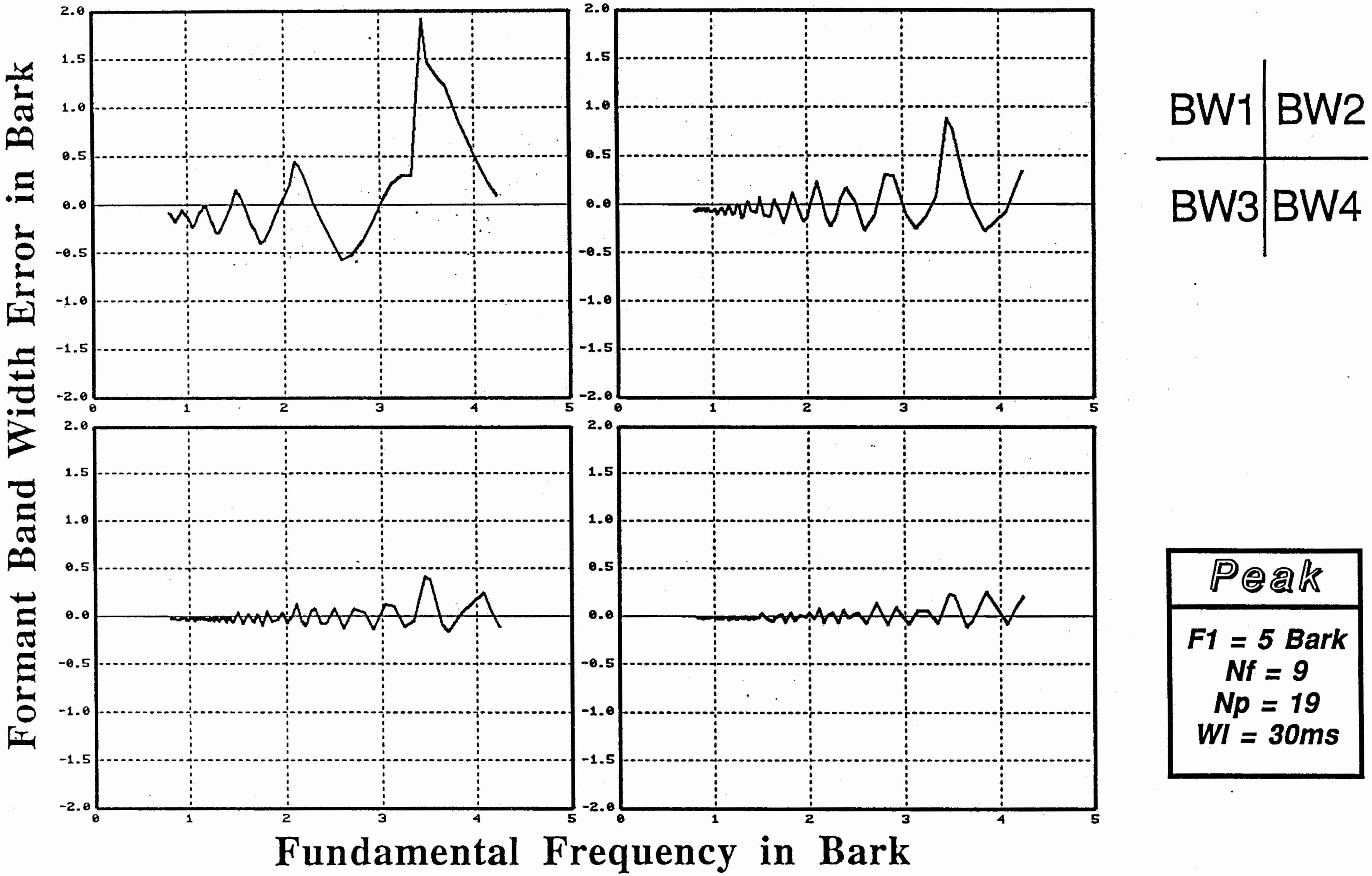


図4.2.16(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

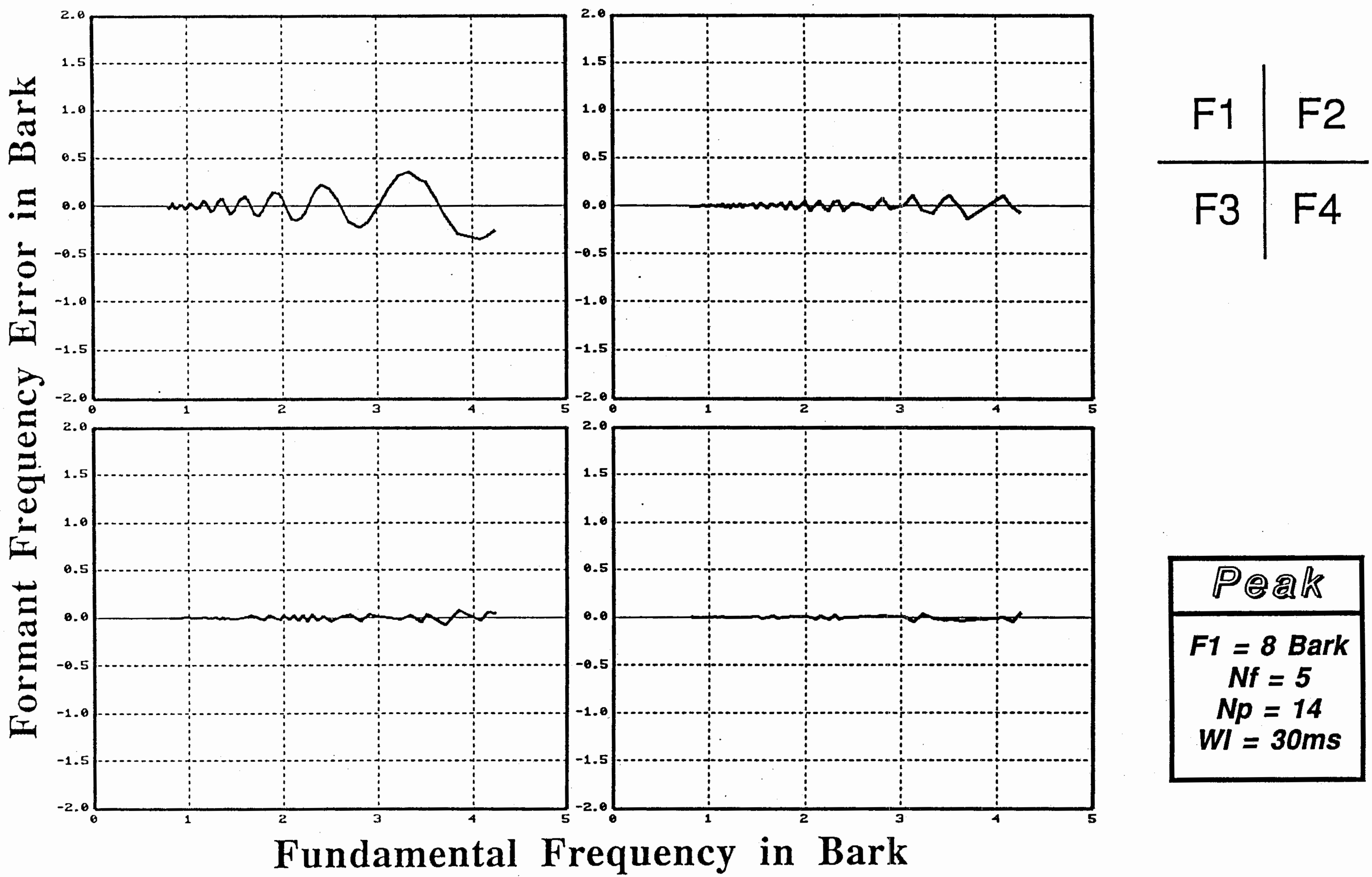


図4.2.17(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

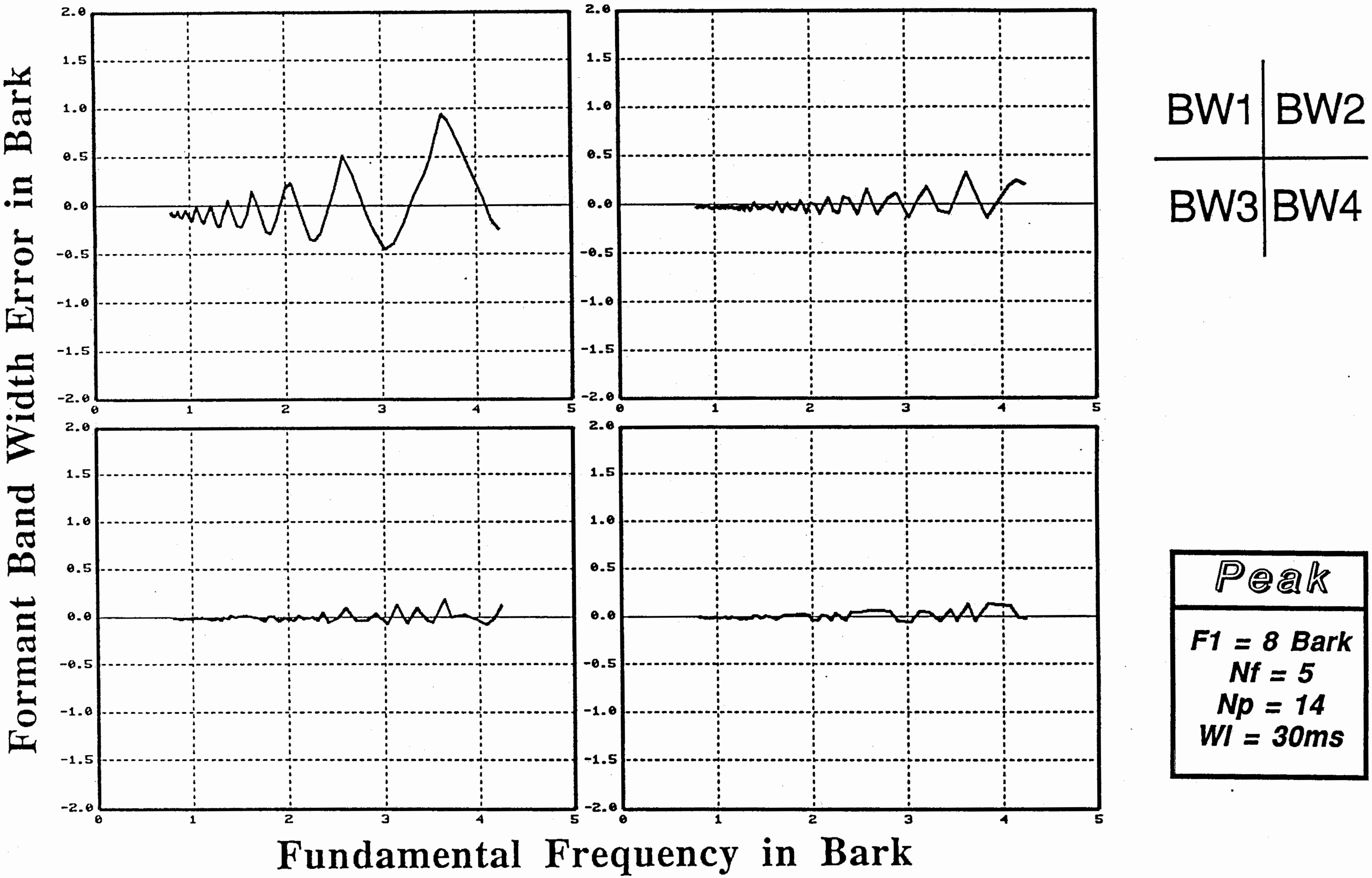


図4.2.17(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

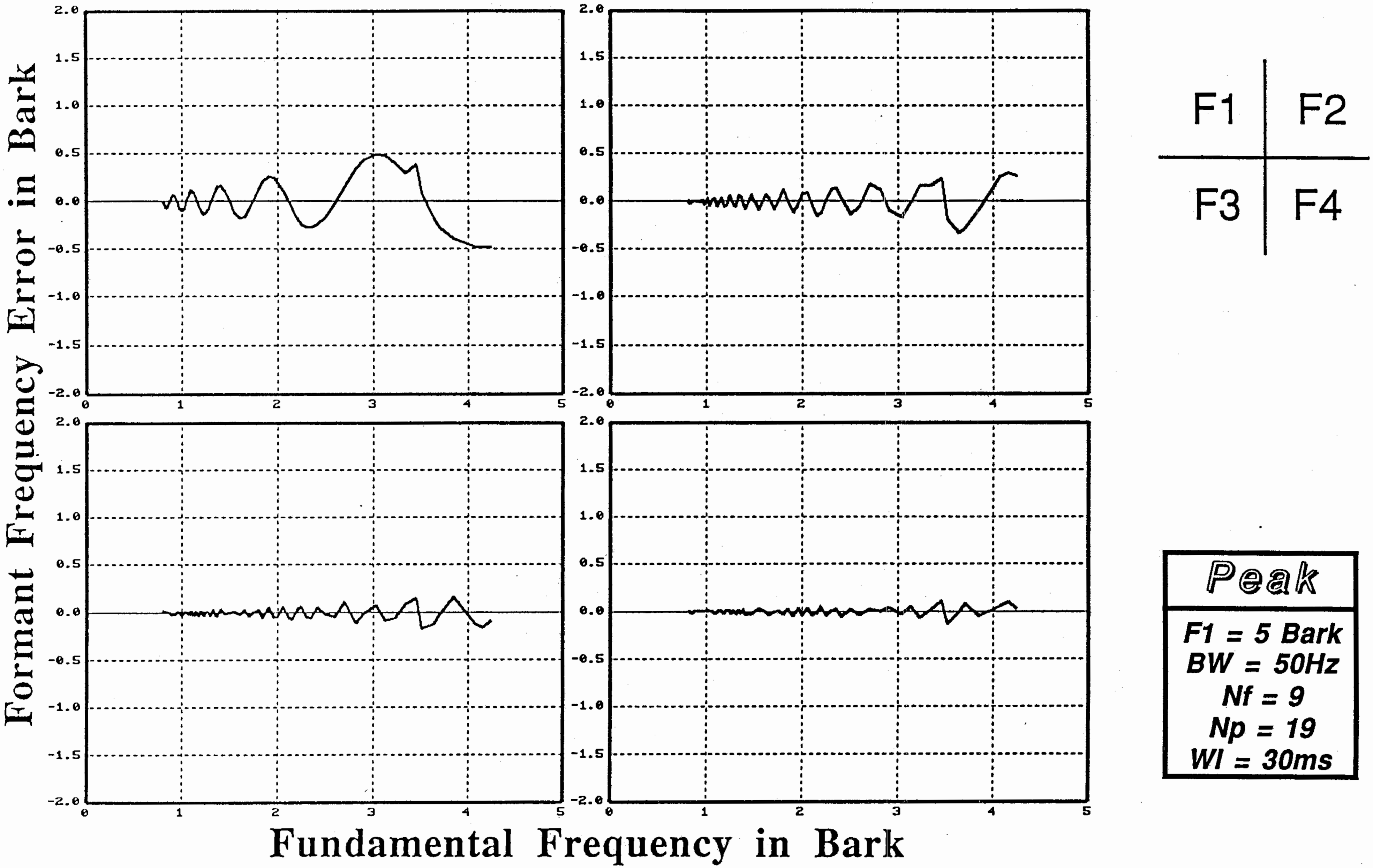


図4.2.18(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

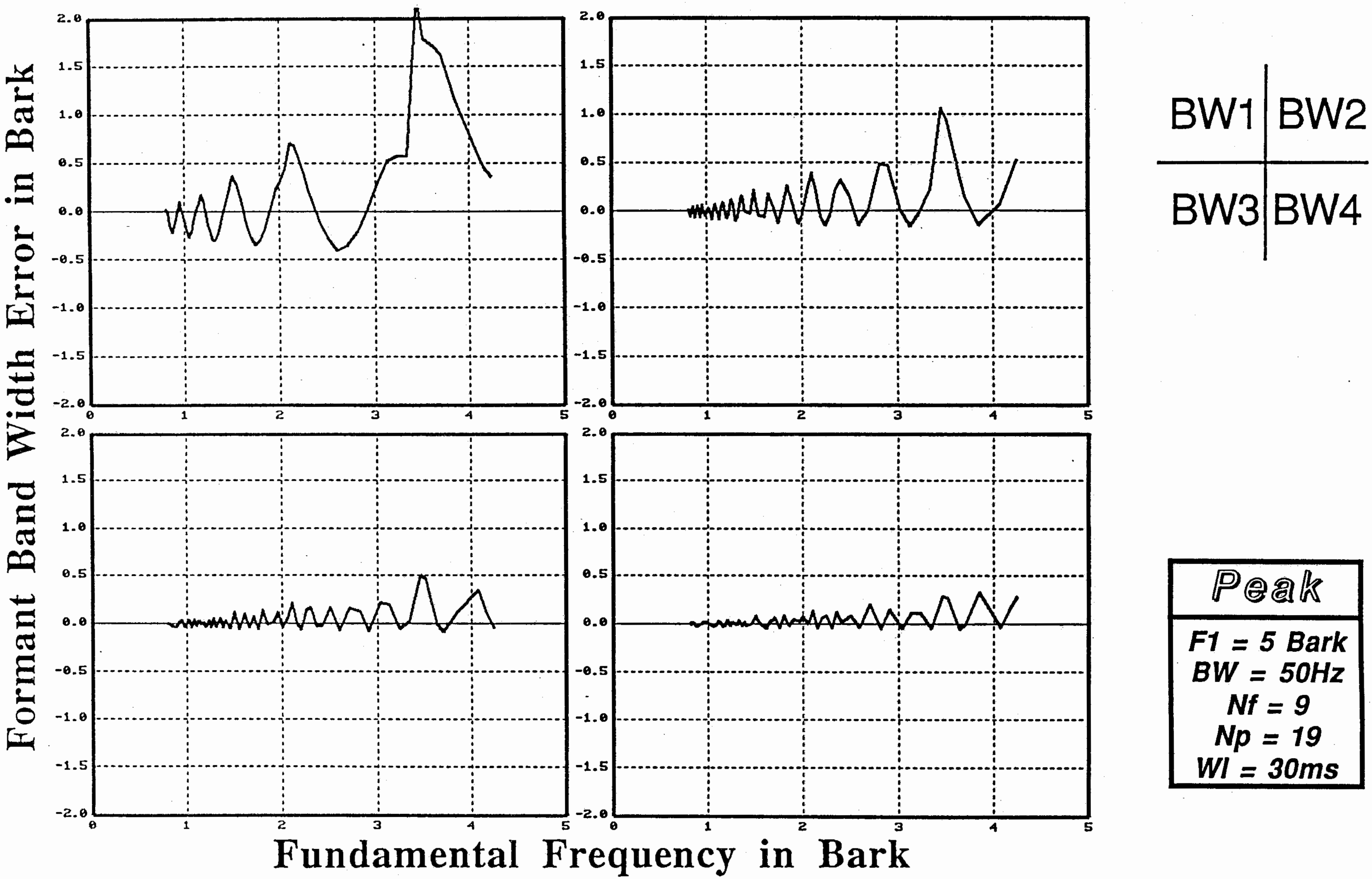


図4.2.18(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

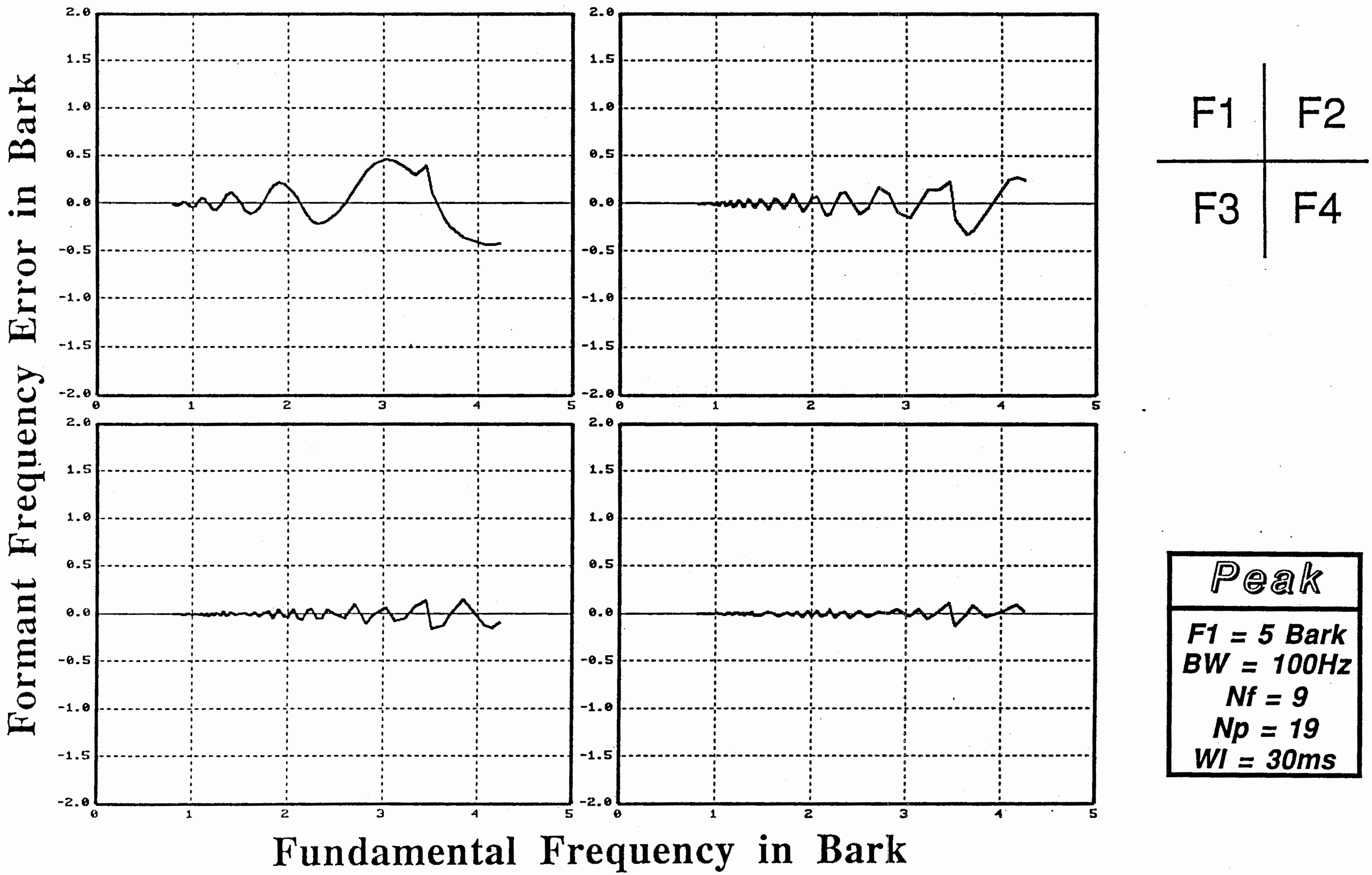


図4.2.19(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

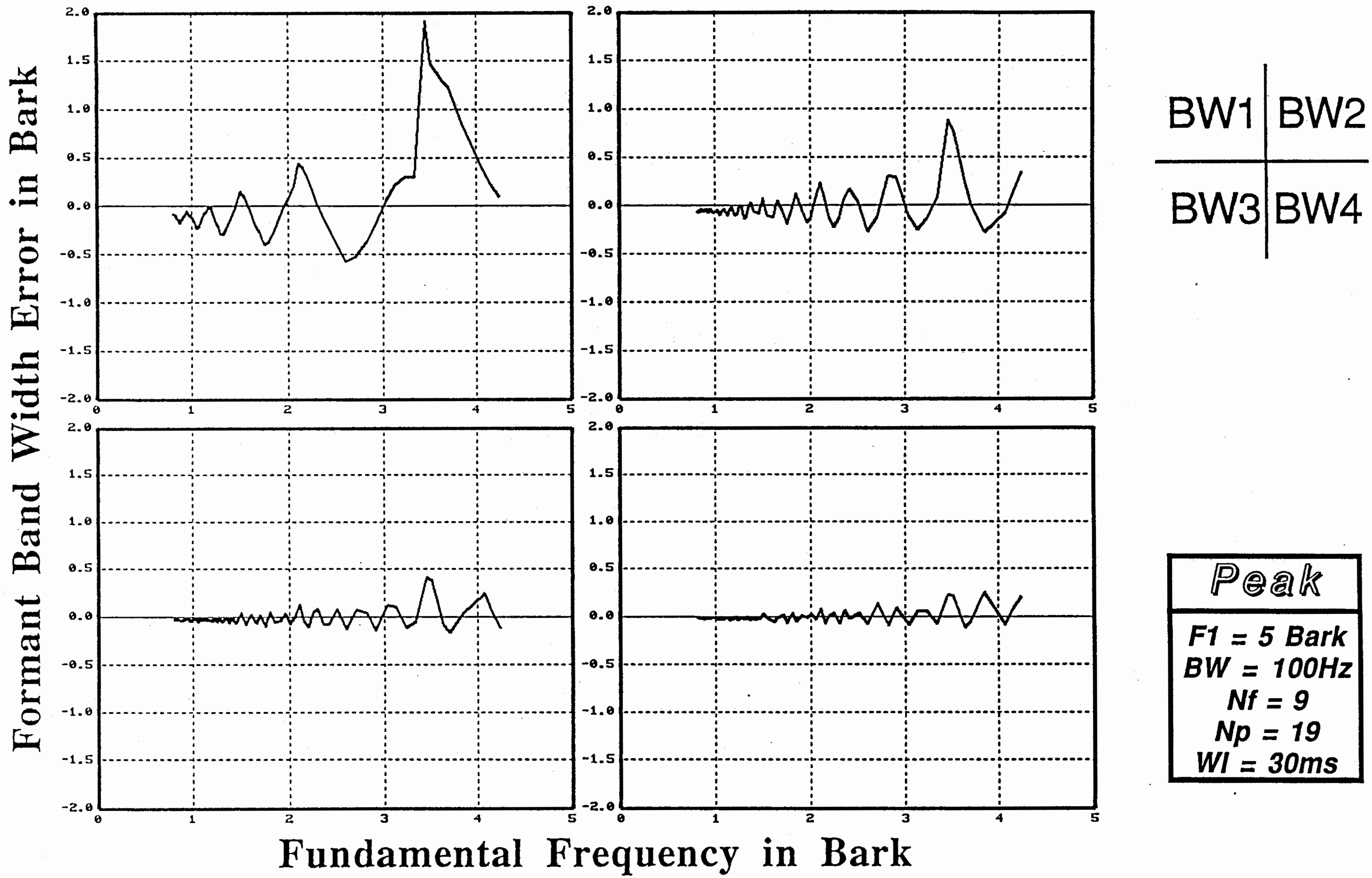


図4.2.19(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

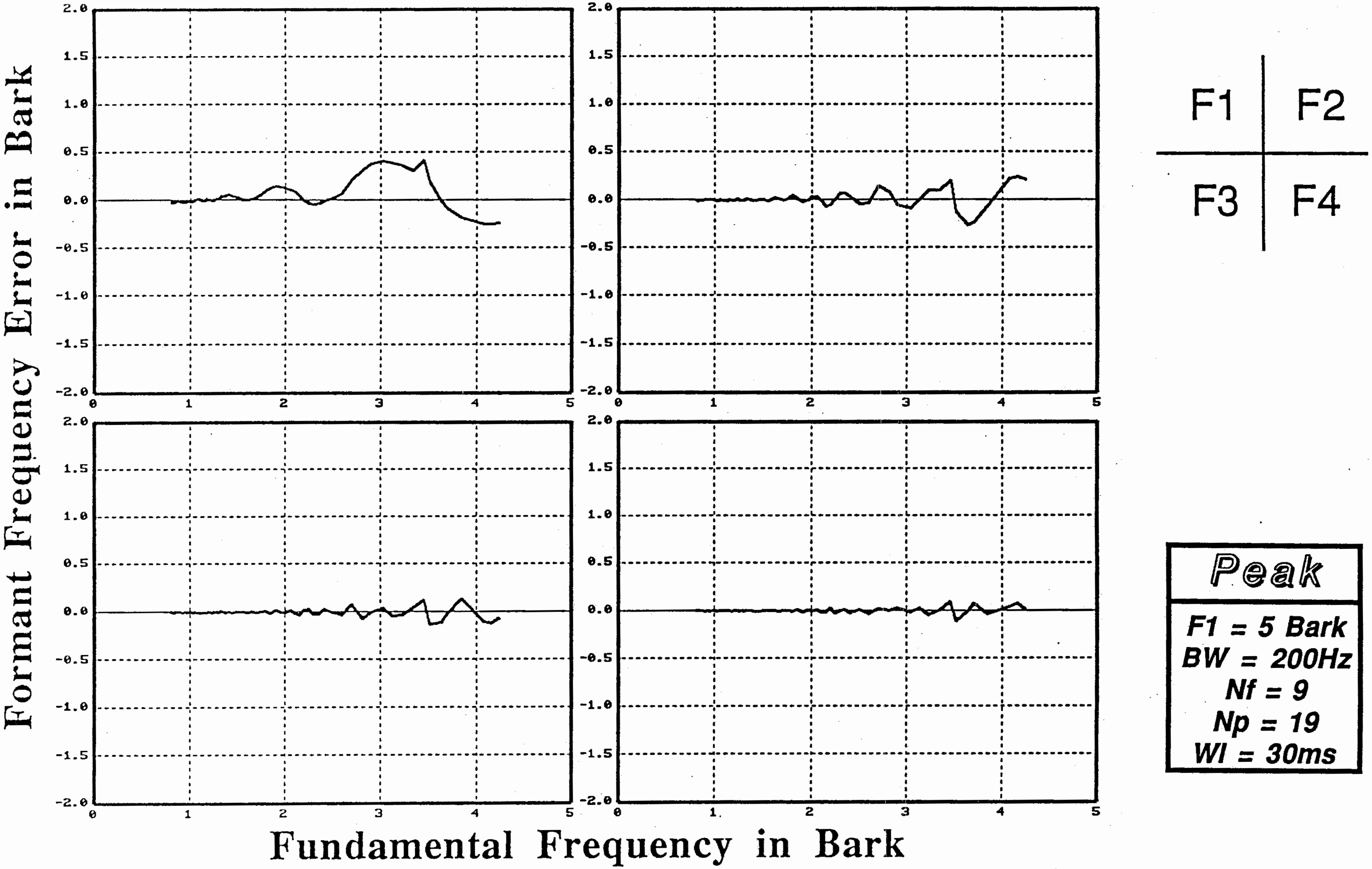


図4.2.20(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

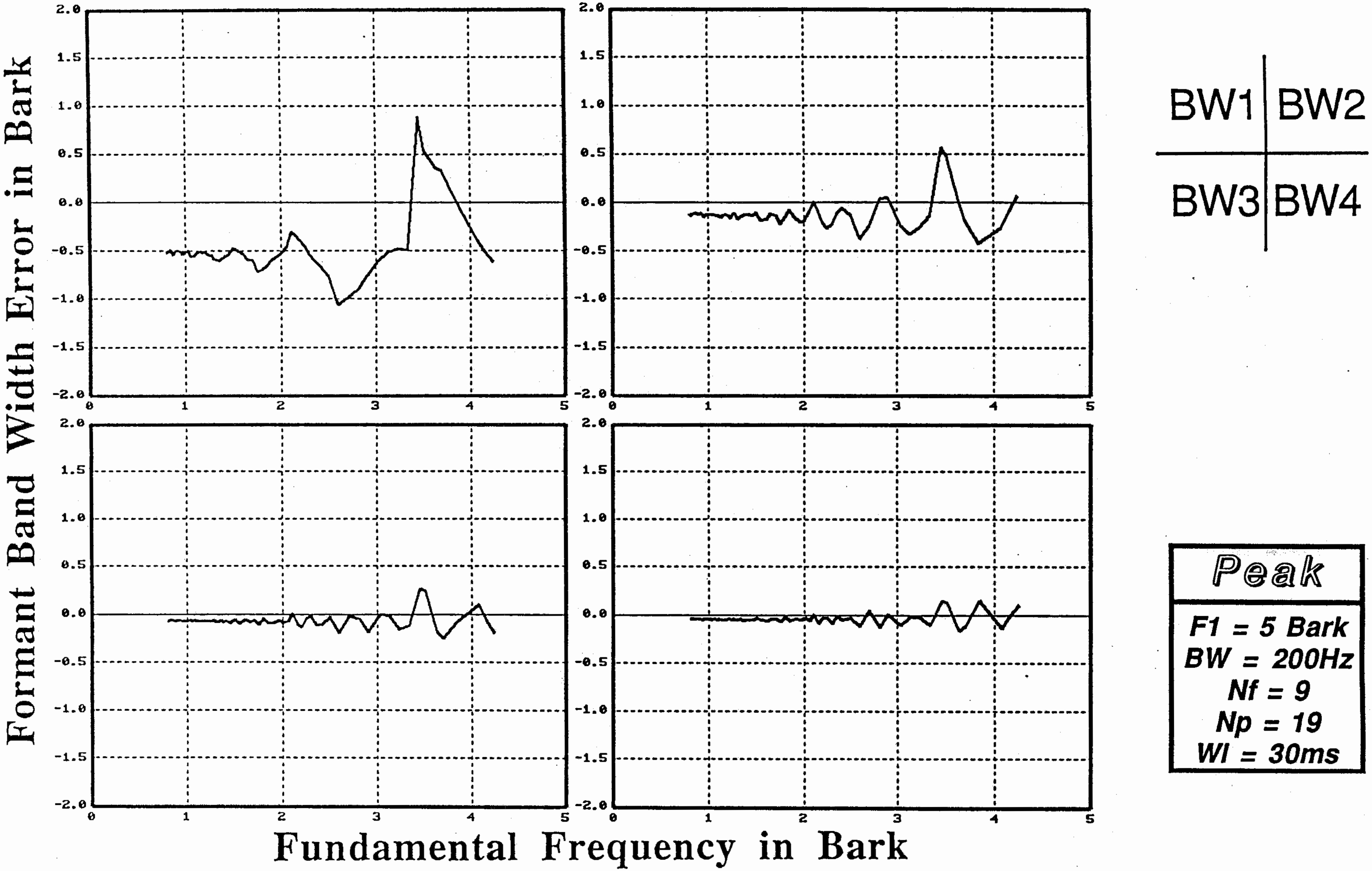


図4.2.20(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

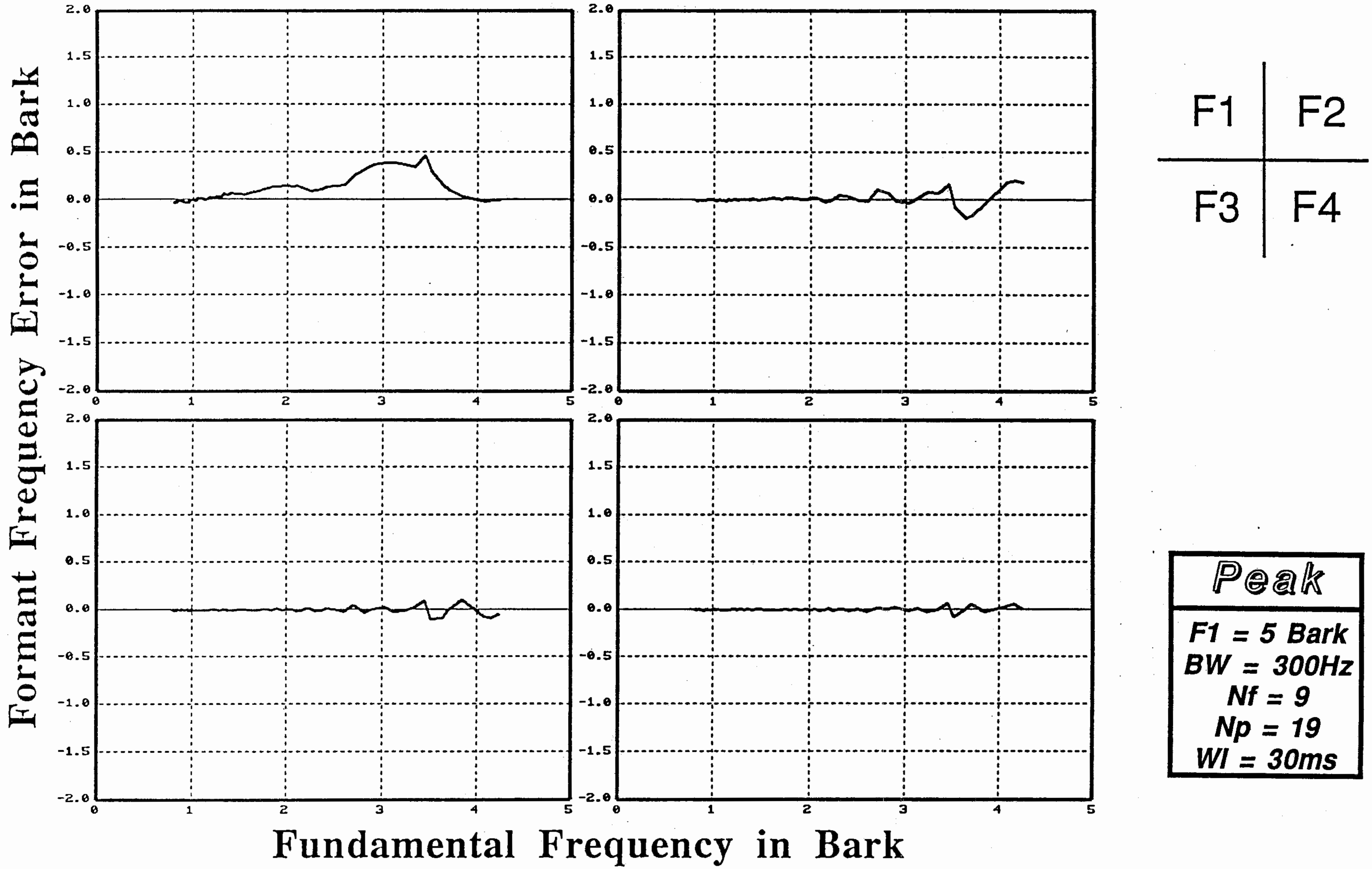


図4.2.21(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（中性母音）

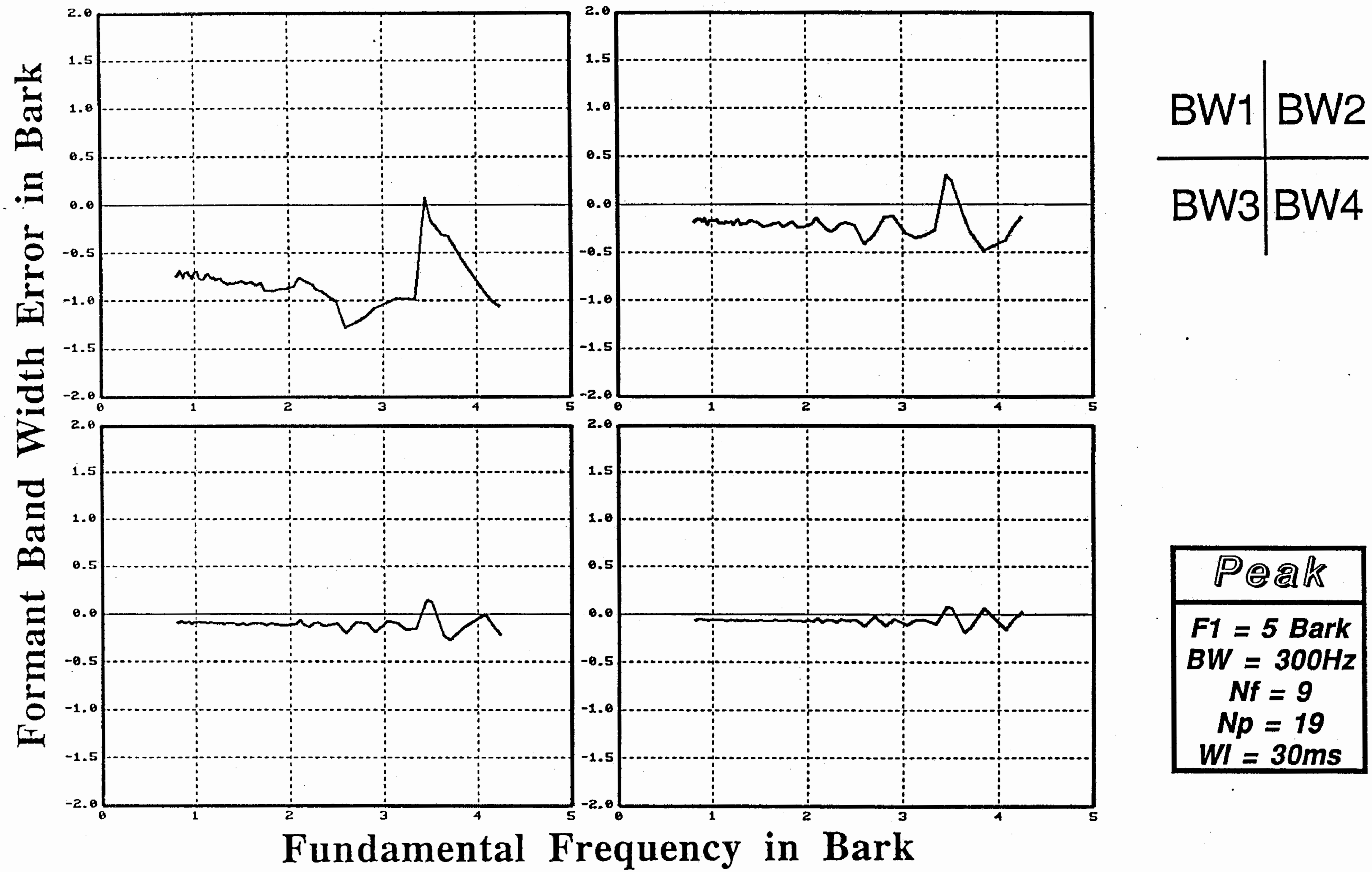


図4.2.21(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（中性母音）

5. 5母音の分析結果

本章では、日本語5母音の合成母音を資料として、分析を行なった結果を示す。

5.1 極推定の結果

本節では、極推定法を用いて合成5母音のフォルマント周波数、フォルマント帯域幅を推定する際に、分析パラメータと資料の合成パラメータとの関係が、推定値にどの様な影響を及ぼすか調べた結果を示す。

5.1.1 分析次数の影響

本節では、分析の次数がフォルマント周波数、フォルマント帯域幅の推定値に及ぼす影響を調べた結果を示す。

図5.1.1(a)～図5.1.5(a)は、フォルマント周波数の推定誤差の絶対値を分析次数Npを横軸にして描いたものである。各図とも左上から右下へ資料のF0が高いものから低いものへと並んでいる。

図5.1.1(b)～図5.1.5(b)は、フォルマント帯域幅の推定誤差の絶対値について同様に求めたものである。

図5.1.1、図5.1.2、図5.1.3、図5.1.4、図5.1.5は、それぞれ合成母音/i/、/e/、/a/、/o/、/u/の資料に対する結果である。

図5.1.6は、フォルマント周波数の推定誤差の絶対値がある閾値E以下であるような分析次数の上限と下限を、資料のF0を横軸にして描いたものである。この図ではE=0.5 Barkに設定した。図の配置は図3.1に示されるような、F1-F2平面上での各母音の位置関係に対応している。参考として、図4.1.15に示された中性母音に対するものも、図中の対応する位置に配置して再度示した。

5.1.2 基本周波数の影響

本節では、資料の基本周波数がフォルマント周波数、フォルマント帯域幅の推定値に及ぼす影響を調べた結果を示す。

図5.1.7(a)～図5.1.11(a)は、フォルマント周波数の推定誤差を資料のF0を横軸にして描いたものである。各図とも左上から右下へ第1～第4フォルマントに対する誤差の図を示す。

図5.1.7(b)～図5.1.11(b)は、フォルマント帯域幅について同様に求めたものである。

図5.1.7、図5.1.8、図5.1.9、図5.1.10、図5.1.11は、それぞれ合成母音/i/、/e/、/a/、/o/、/u/の資料に対する結果を示している。

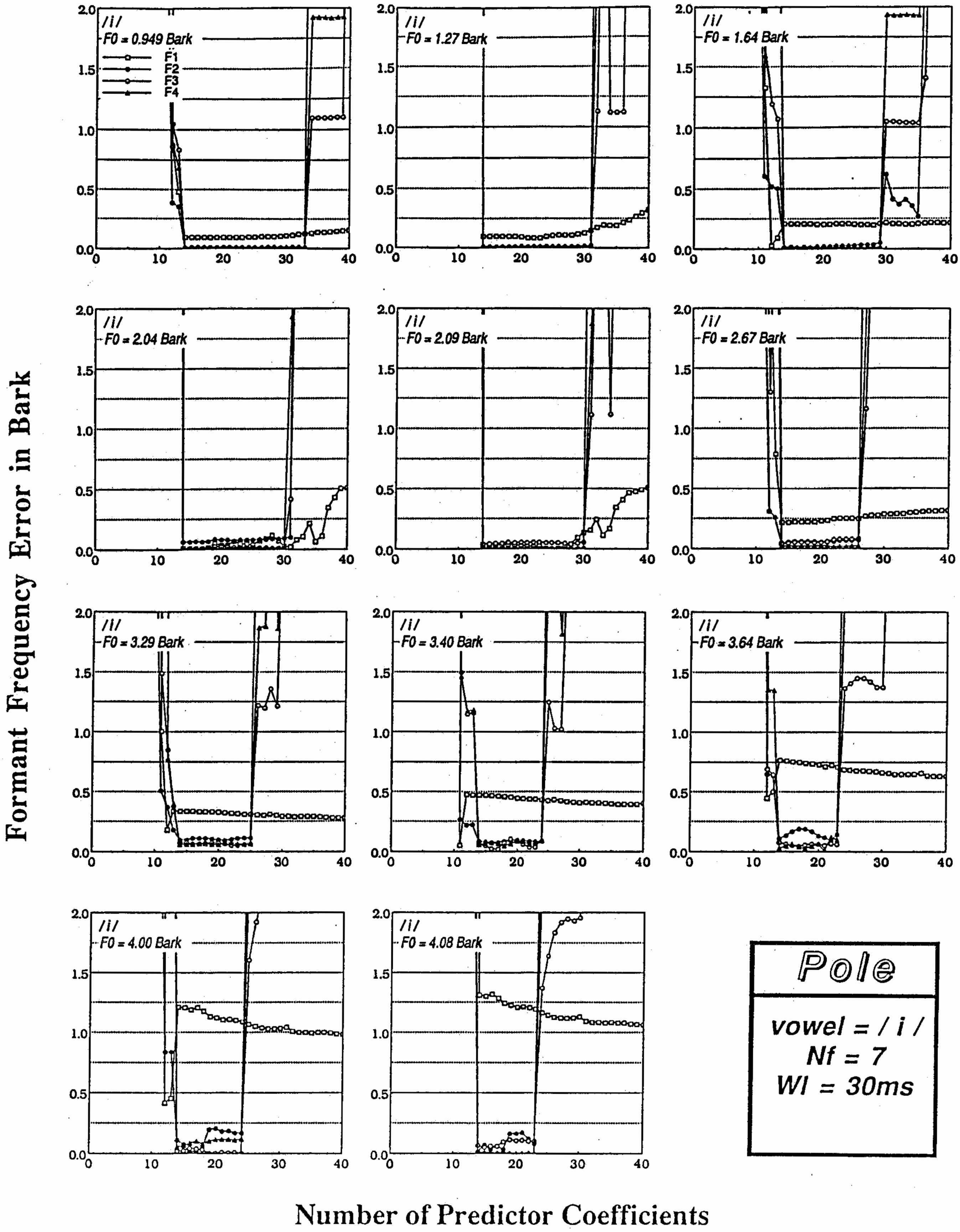


図5.1.1(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（5母音）

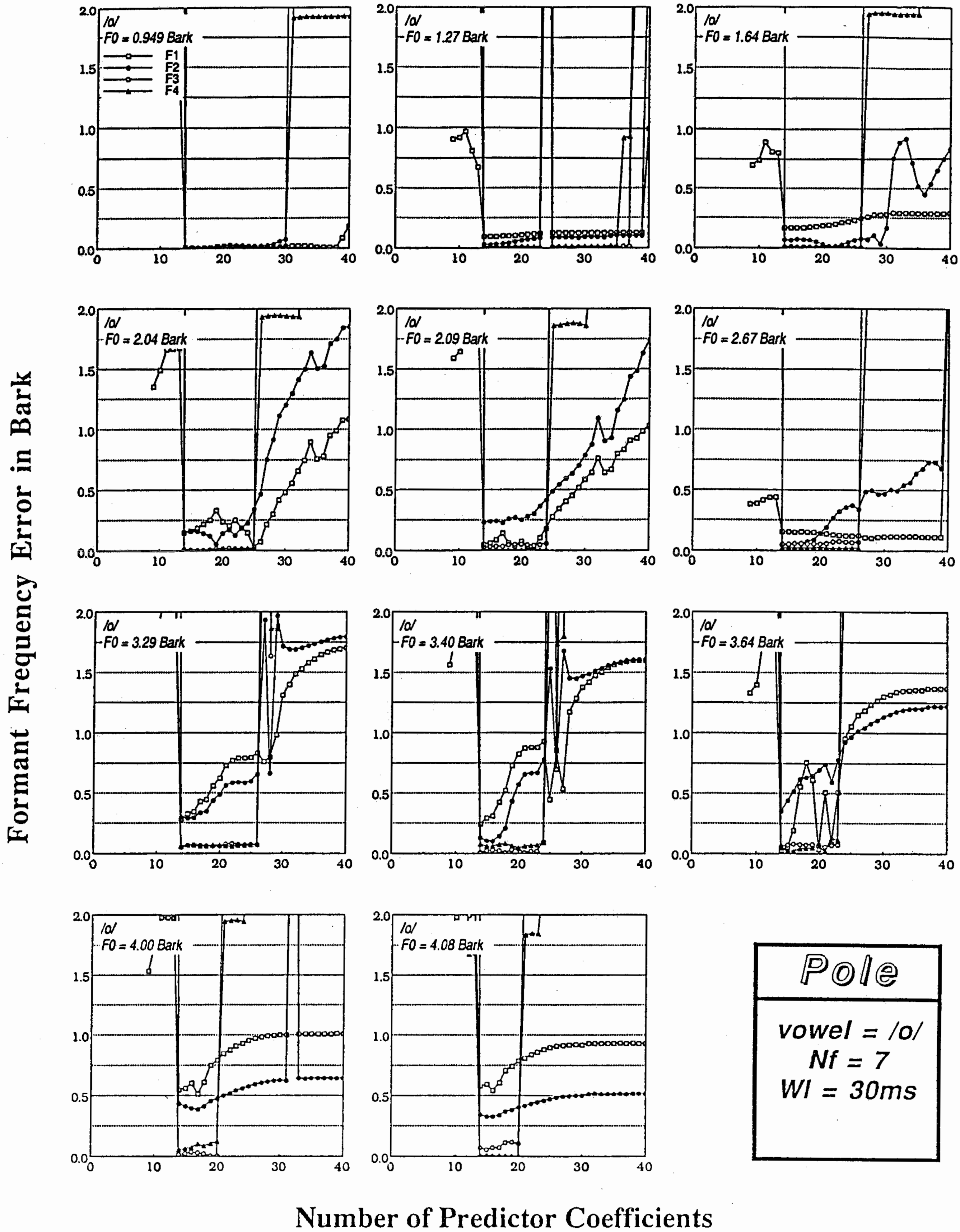


図5.1.1(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（５母音）

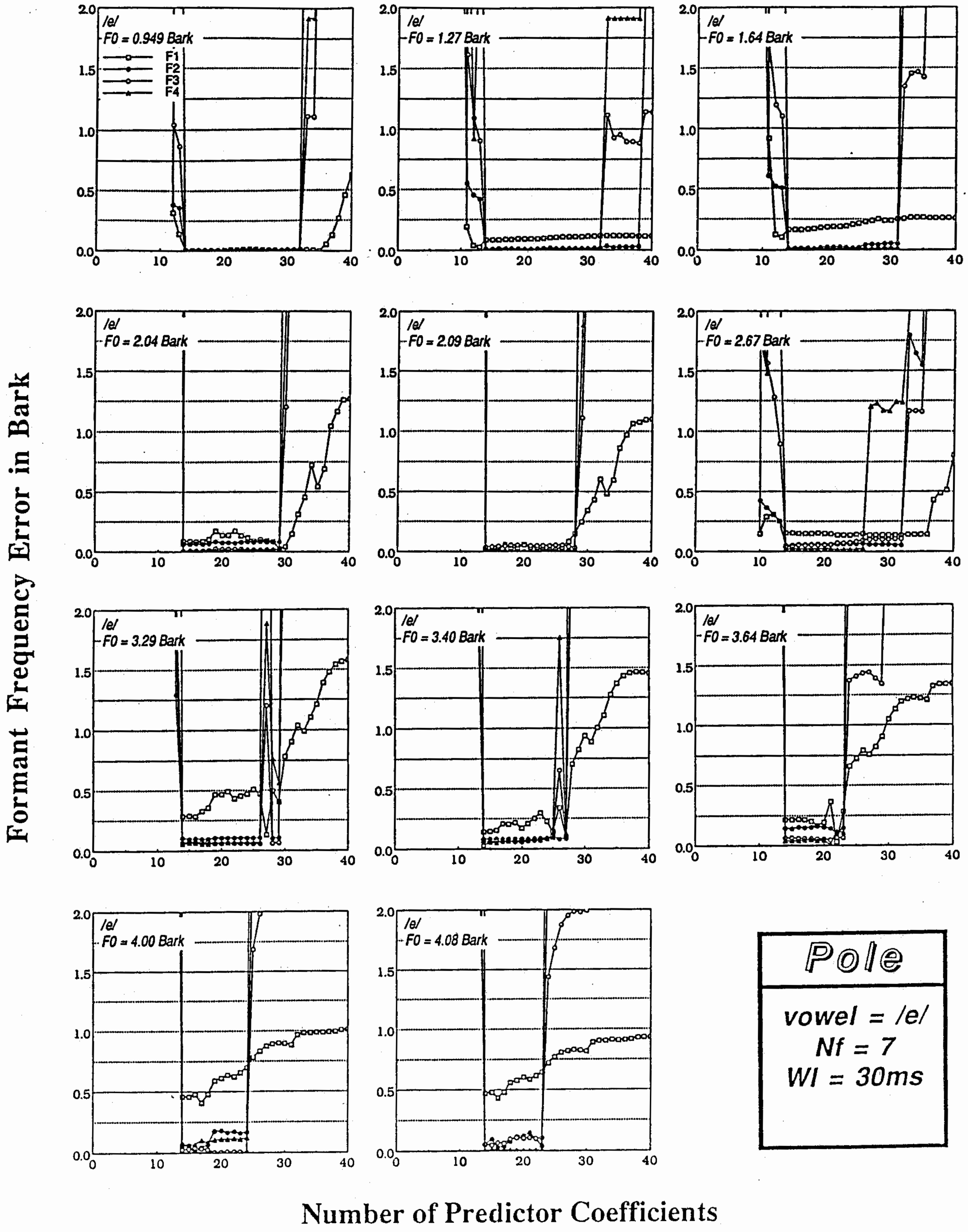


図5.1.2(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（５母音）

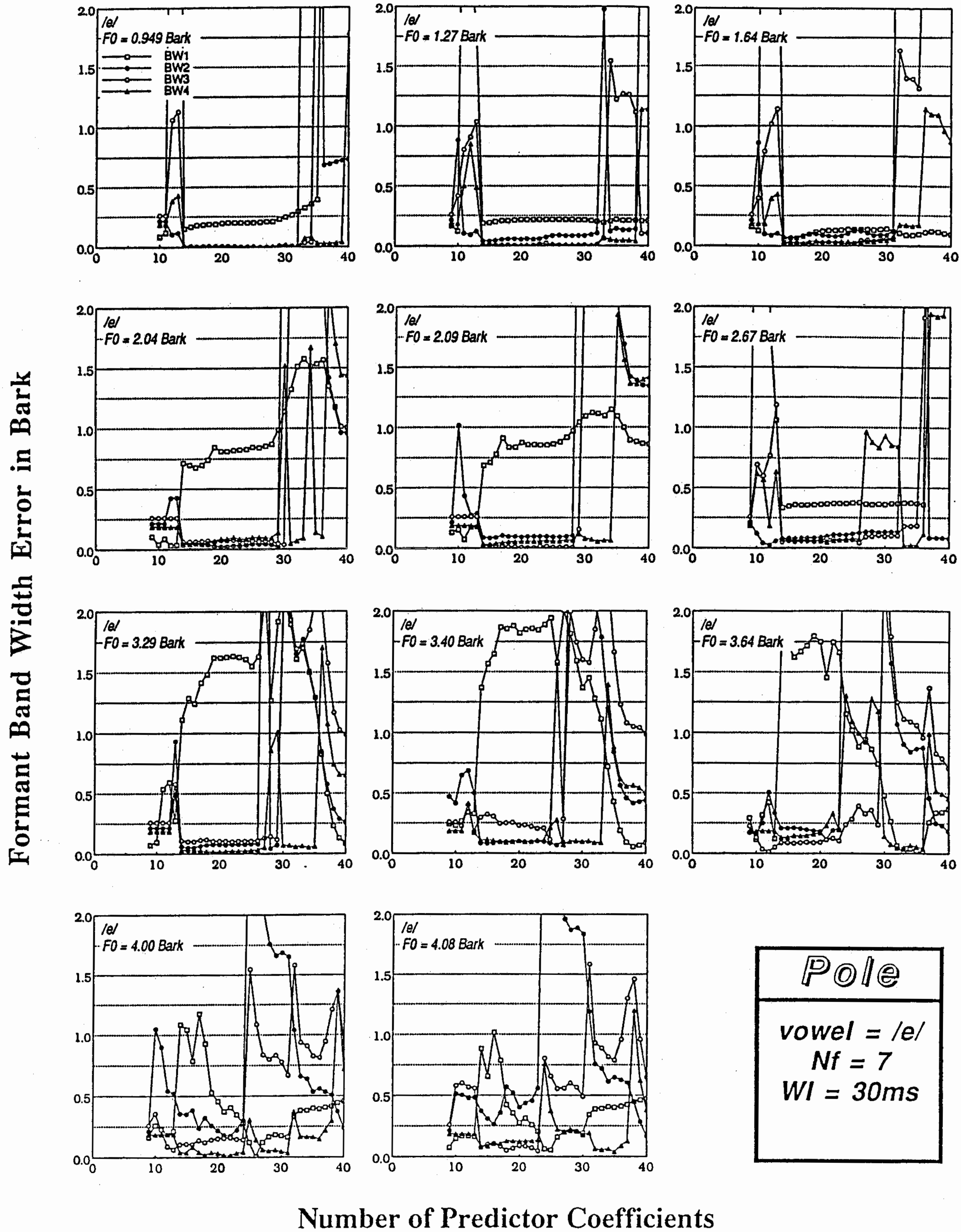


図5.1.2(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（５母音）

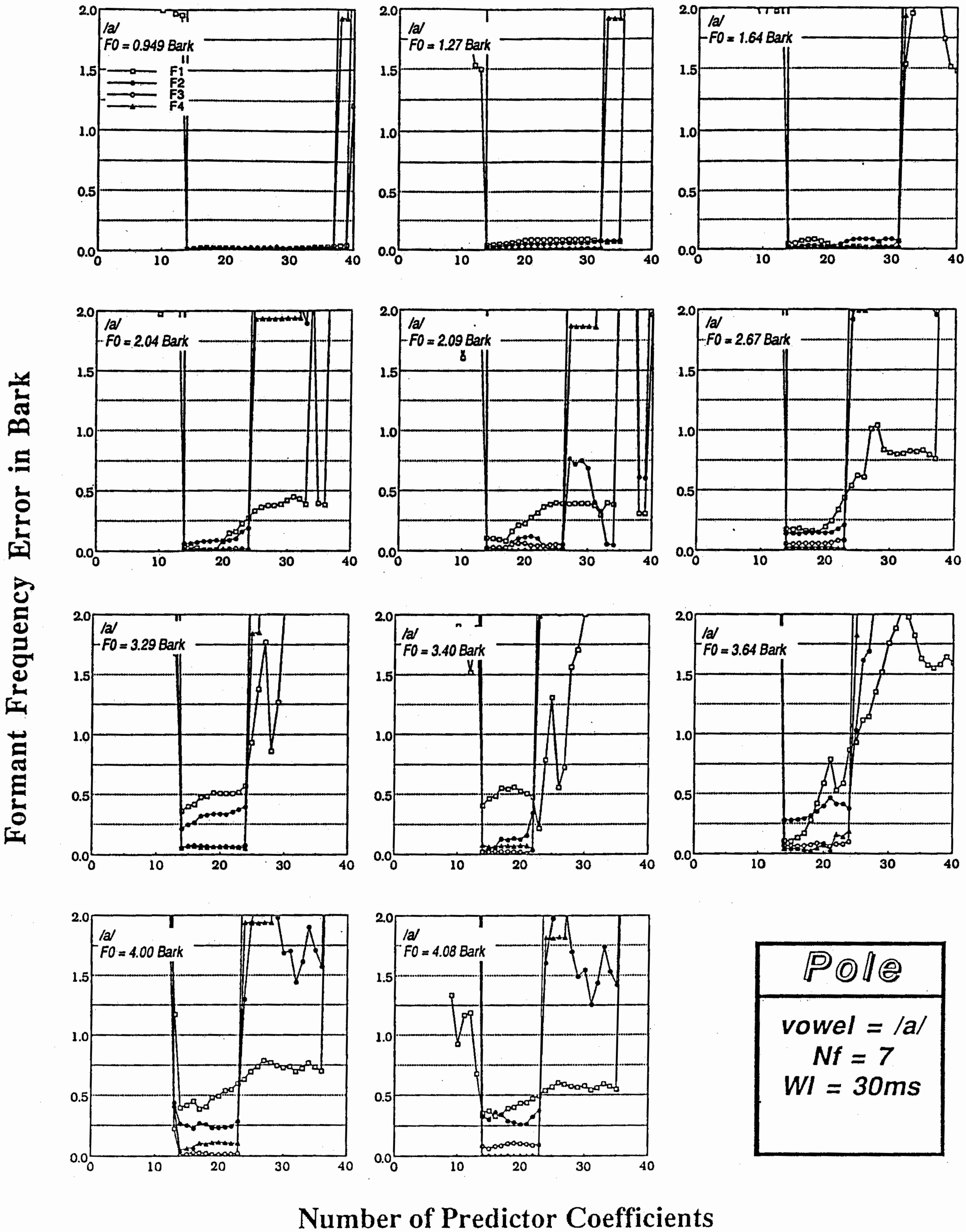


図5.1.3(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（５母音）

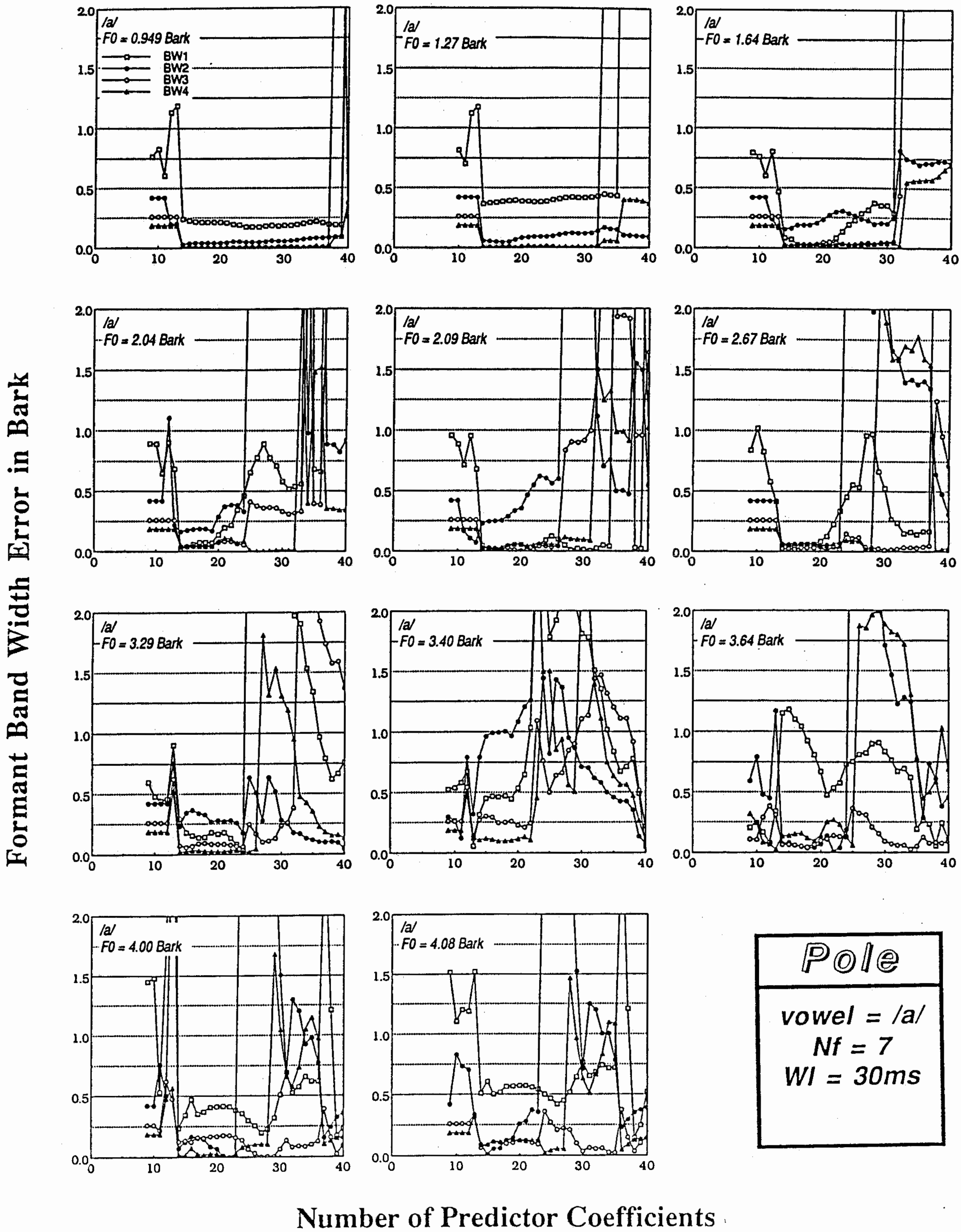


図5.1.3(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（５母音）

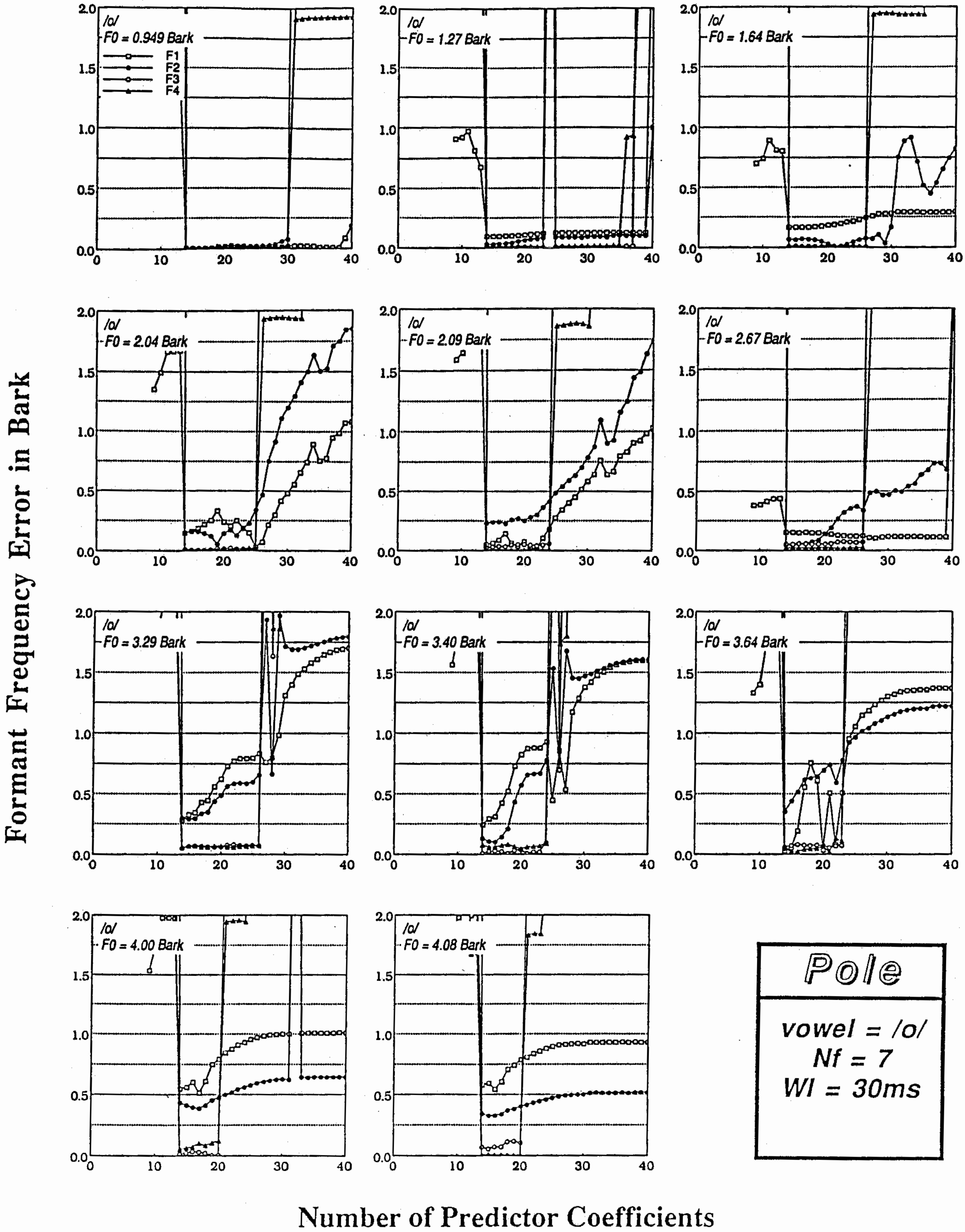


図5.1.4(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（５母音）

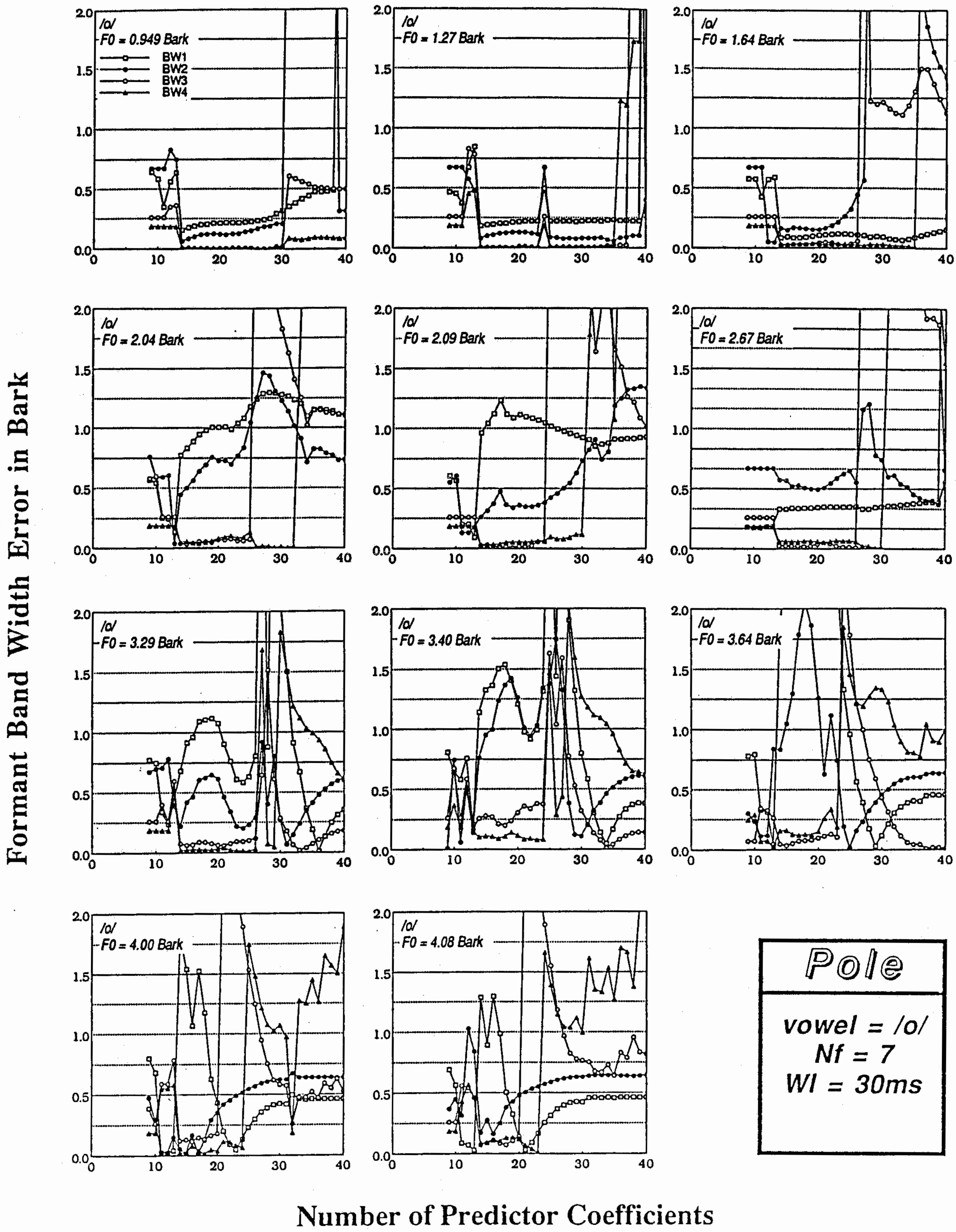


図5.1.4(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（５母音）

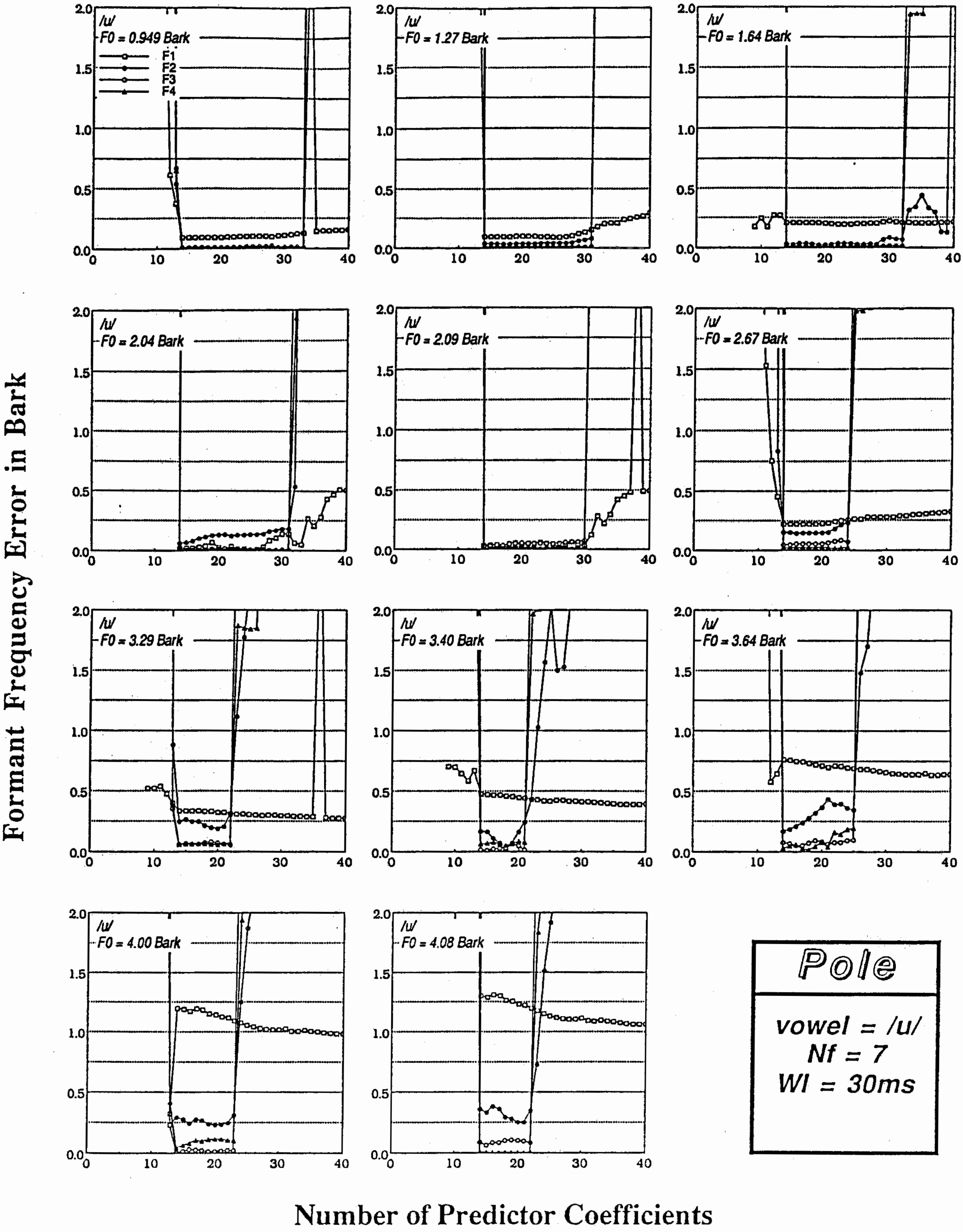


図5.1.5(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（５母音）

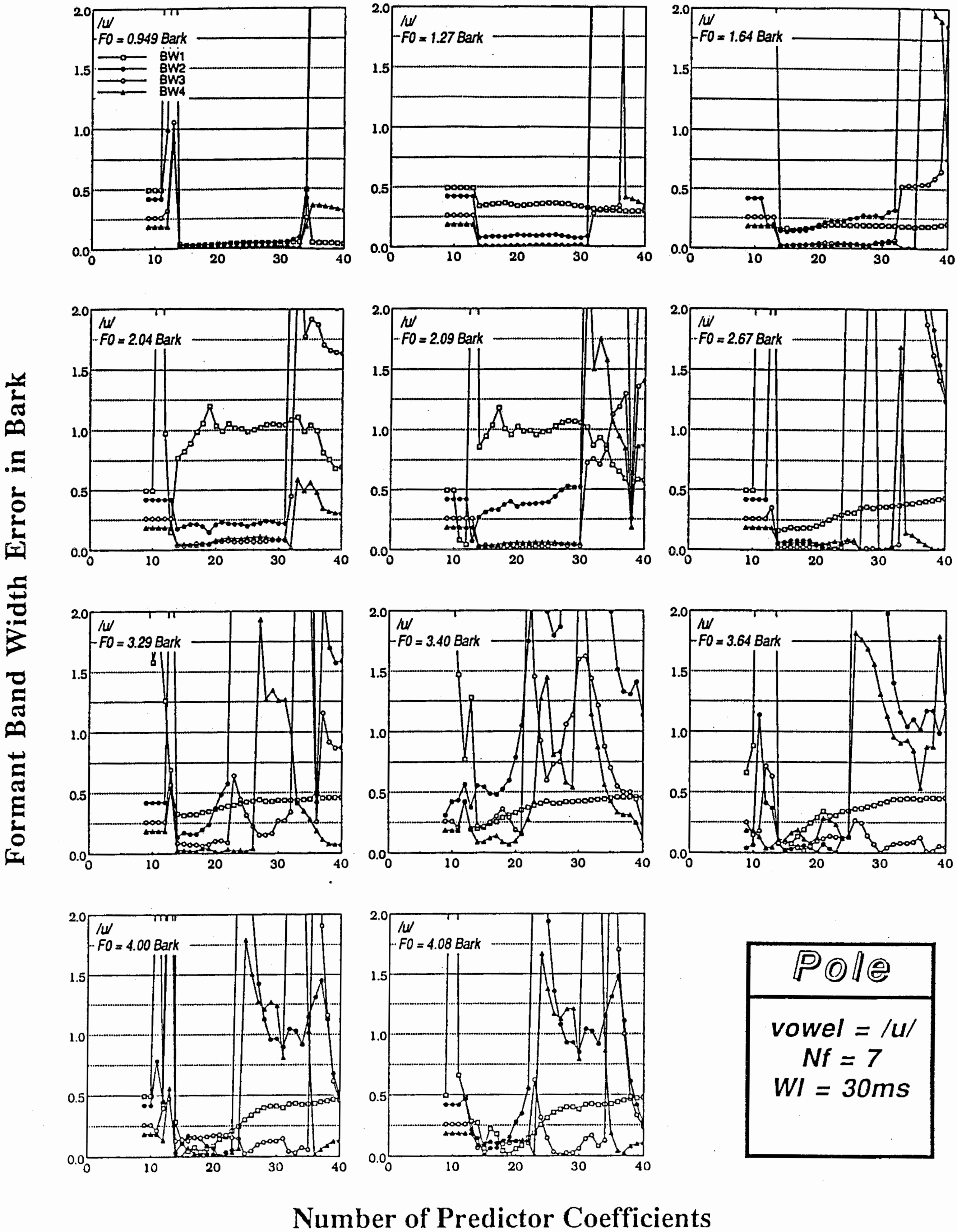


図5.1.5(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（5母音）

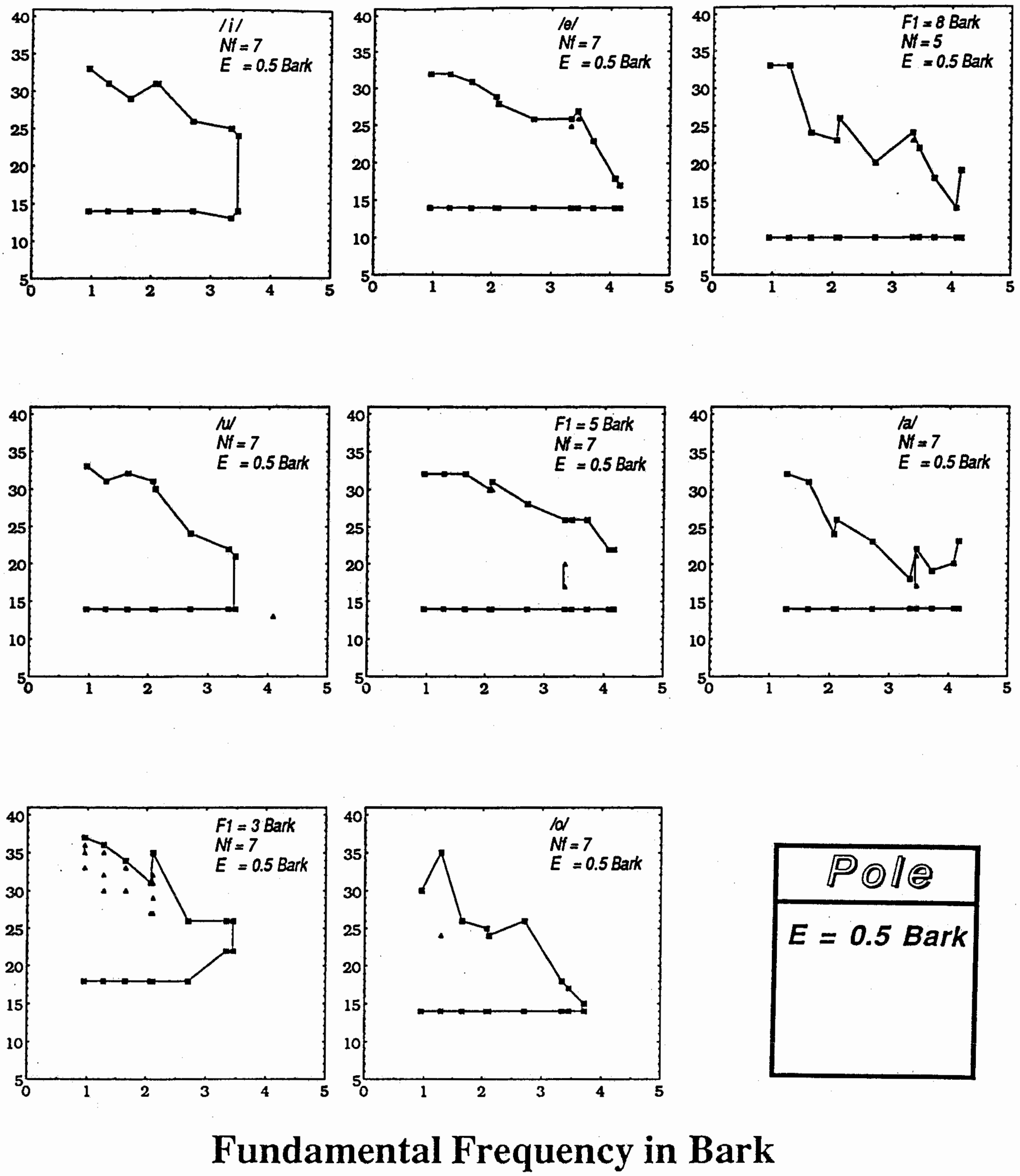


図5.1.6 フォルマント周波数の推定誤差の絶対値が0.5Bark以下である分析次数の上限、下限と資料の基本周波数との関係（５母音）

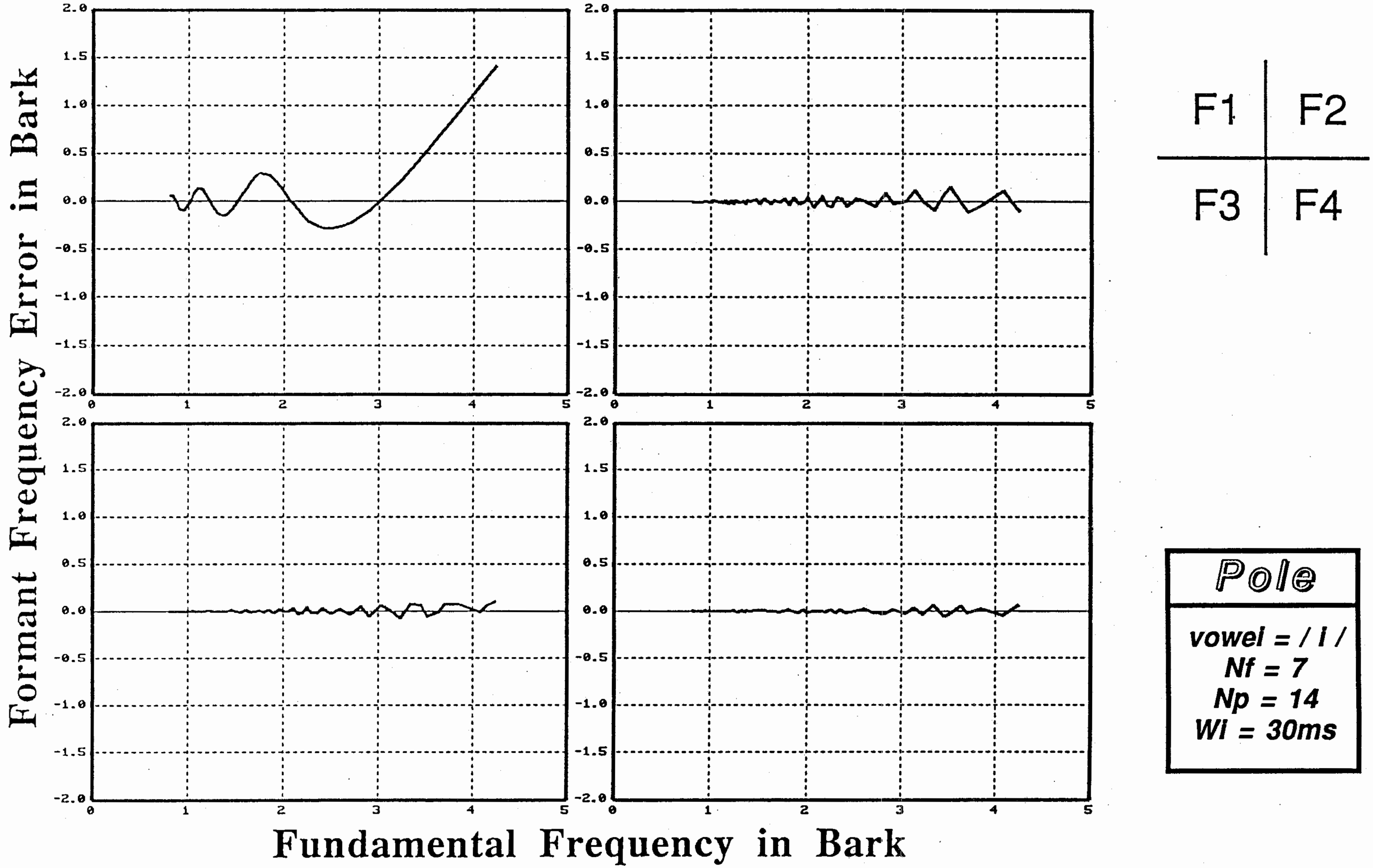


図5.1.7(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（５母音）

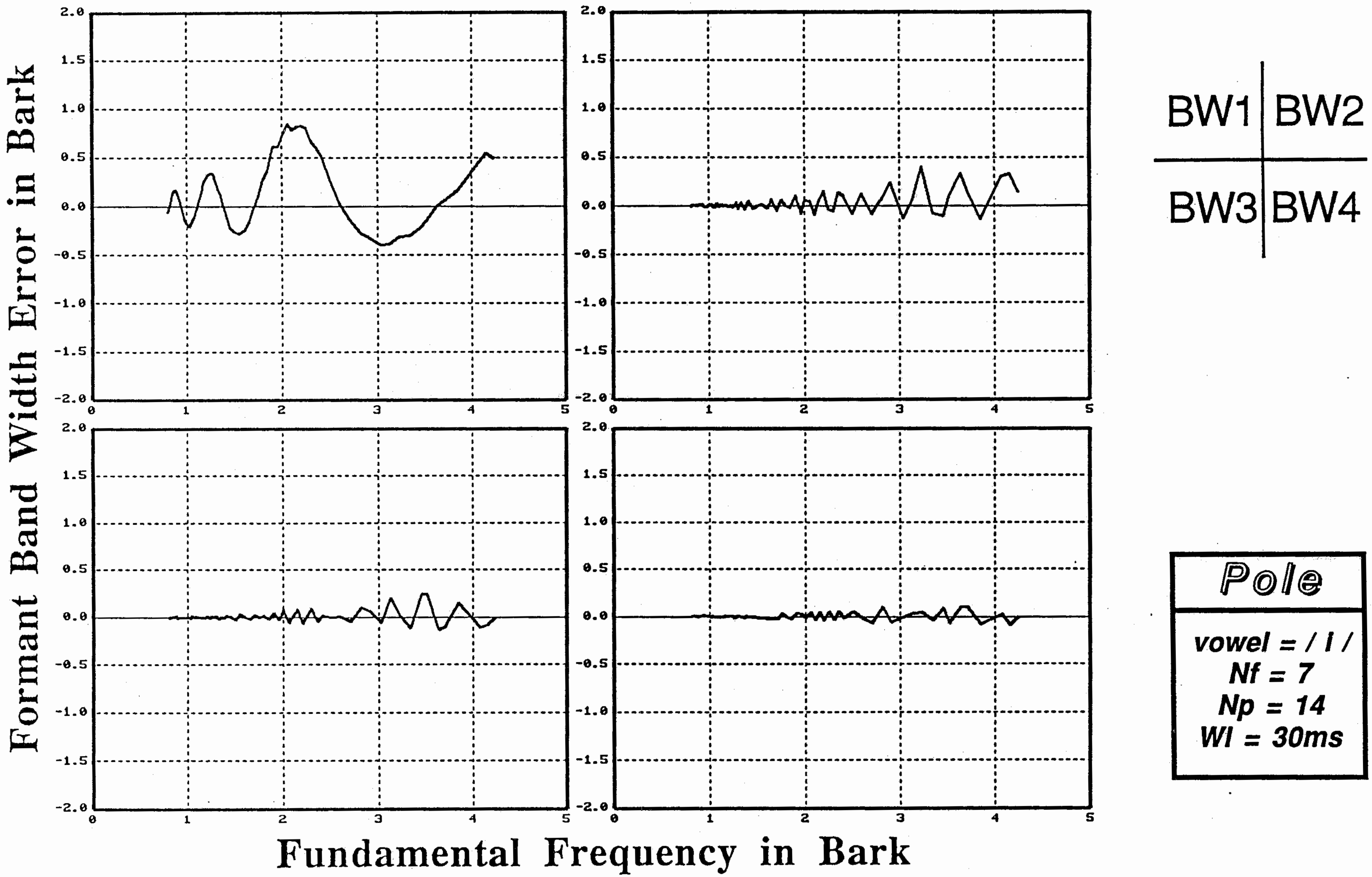


図5.1.7(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（５母音）

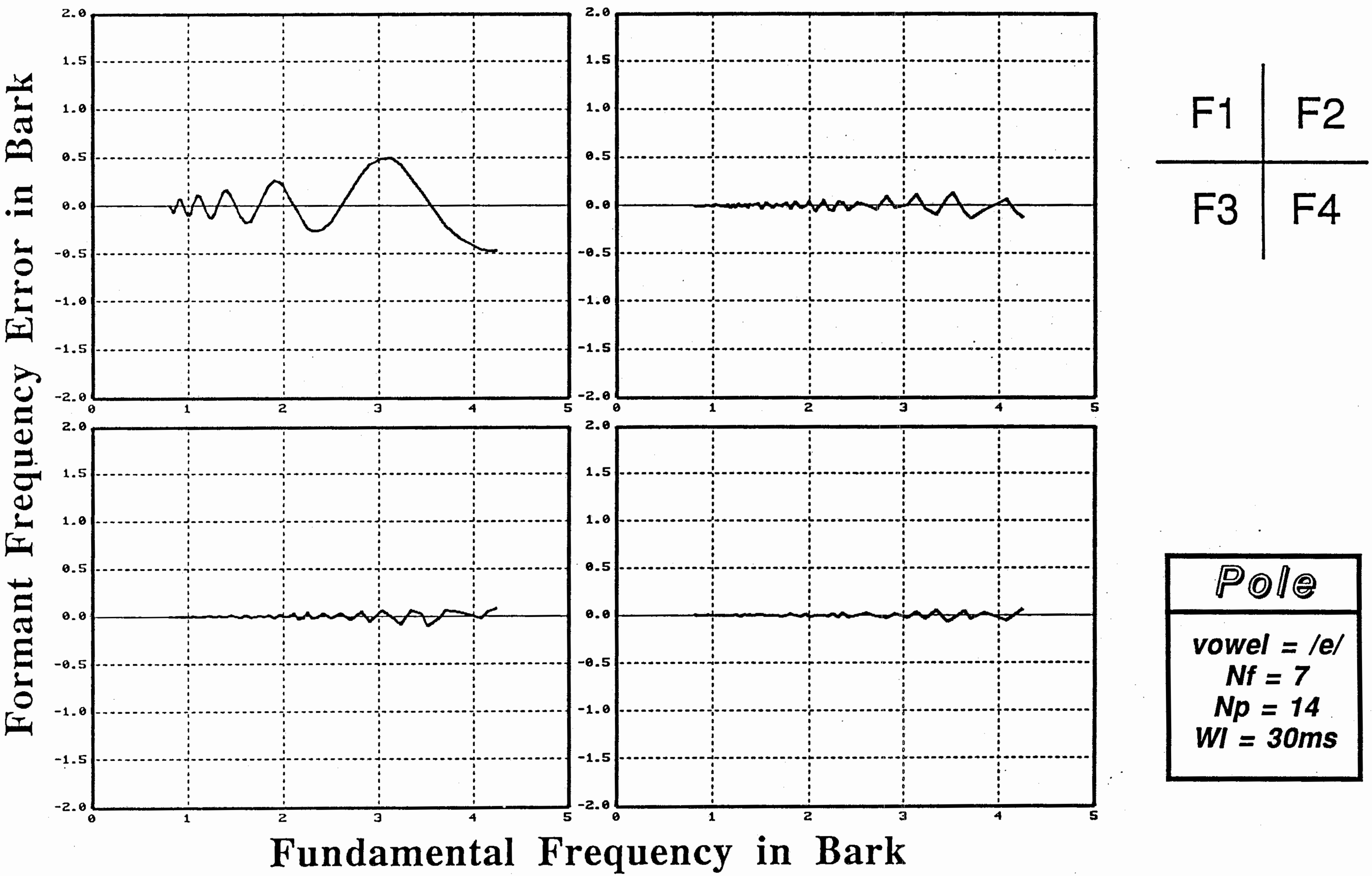


図5.1.8(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（５母音）

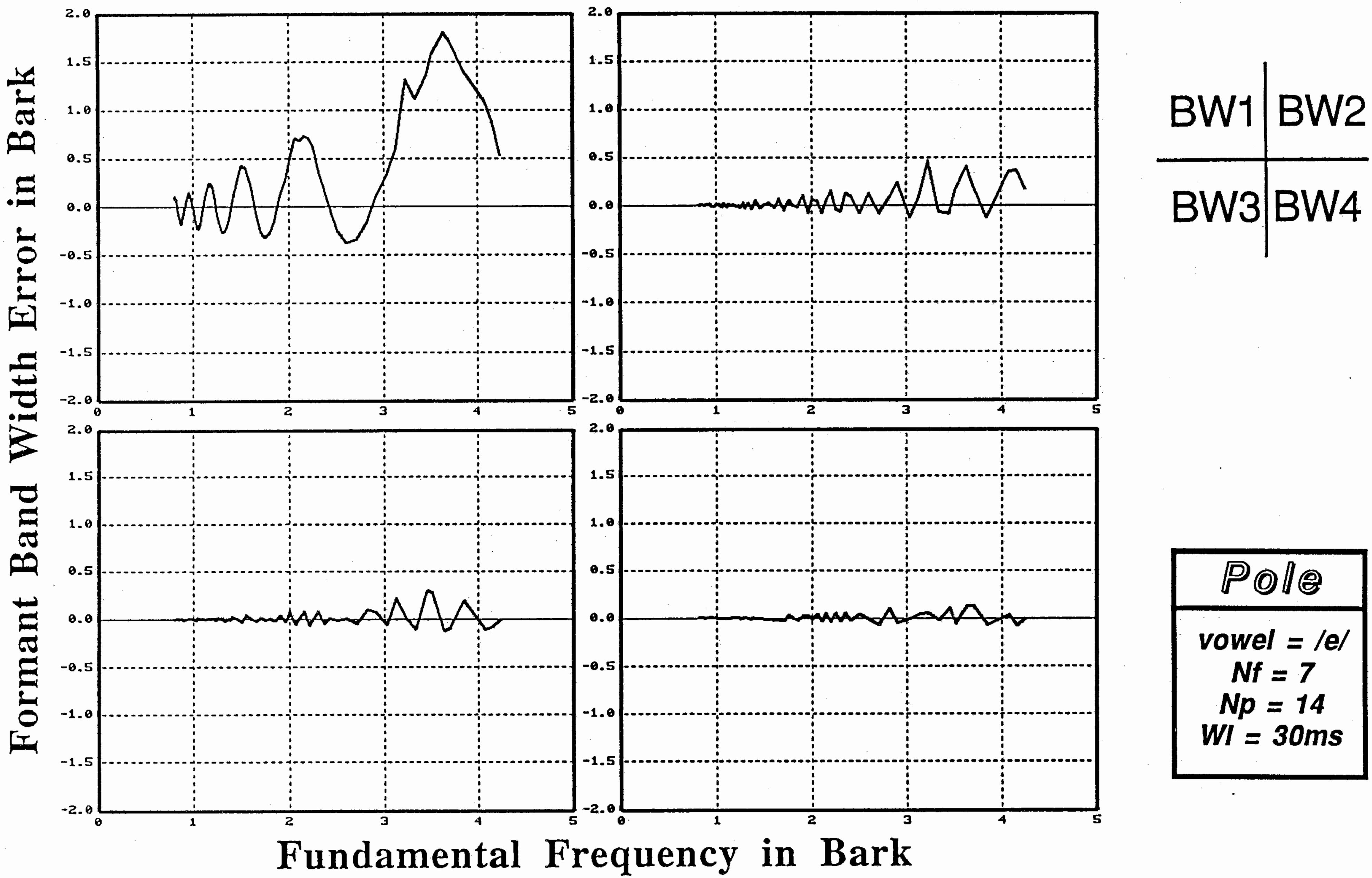


図5.1.8(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（5母音）

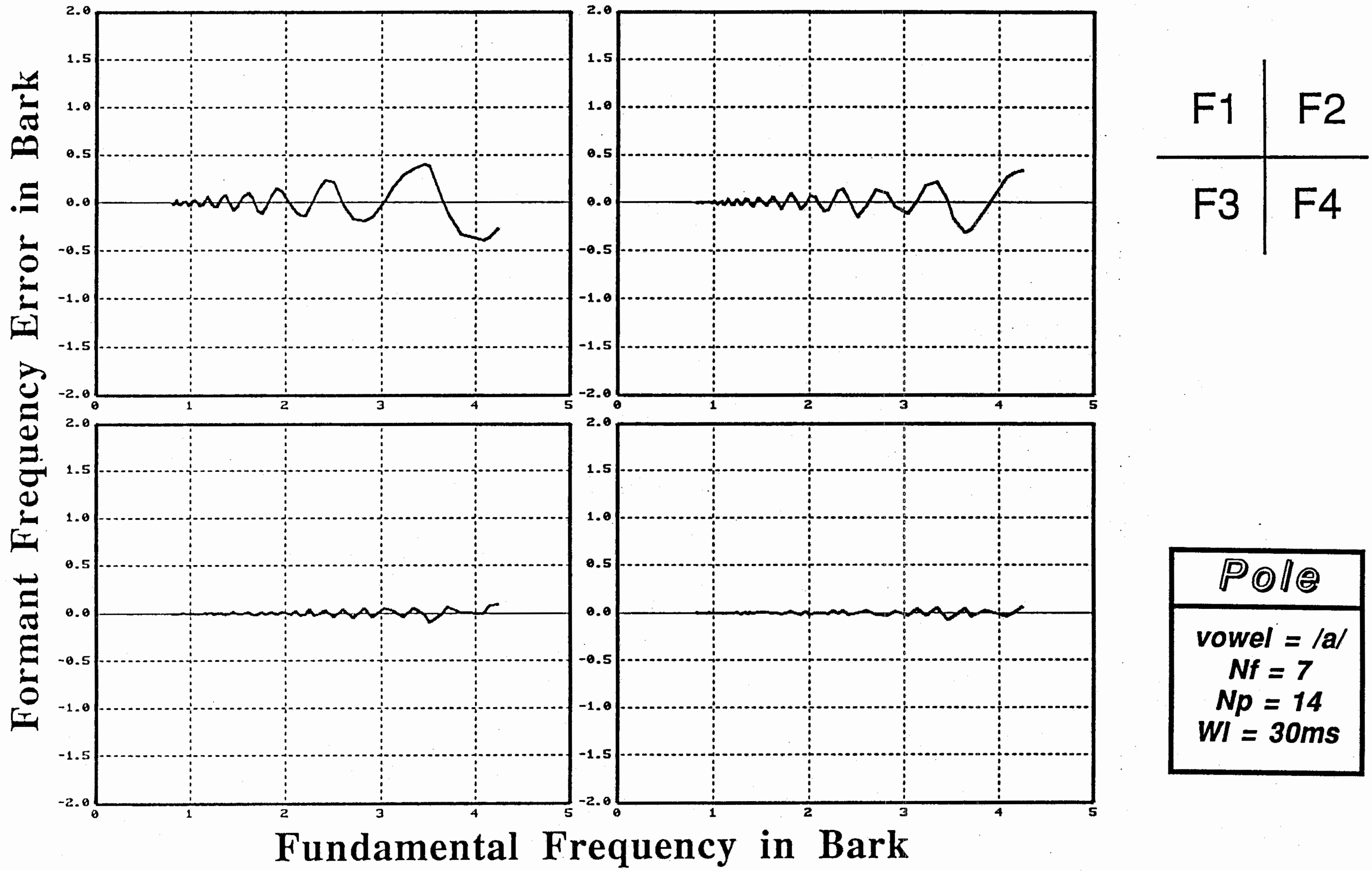


図5.1.9(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（5母音）

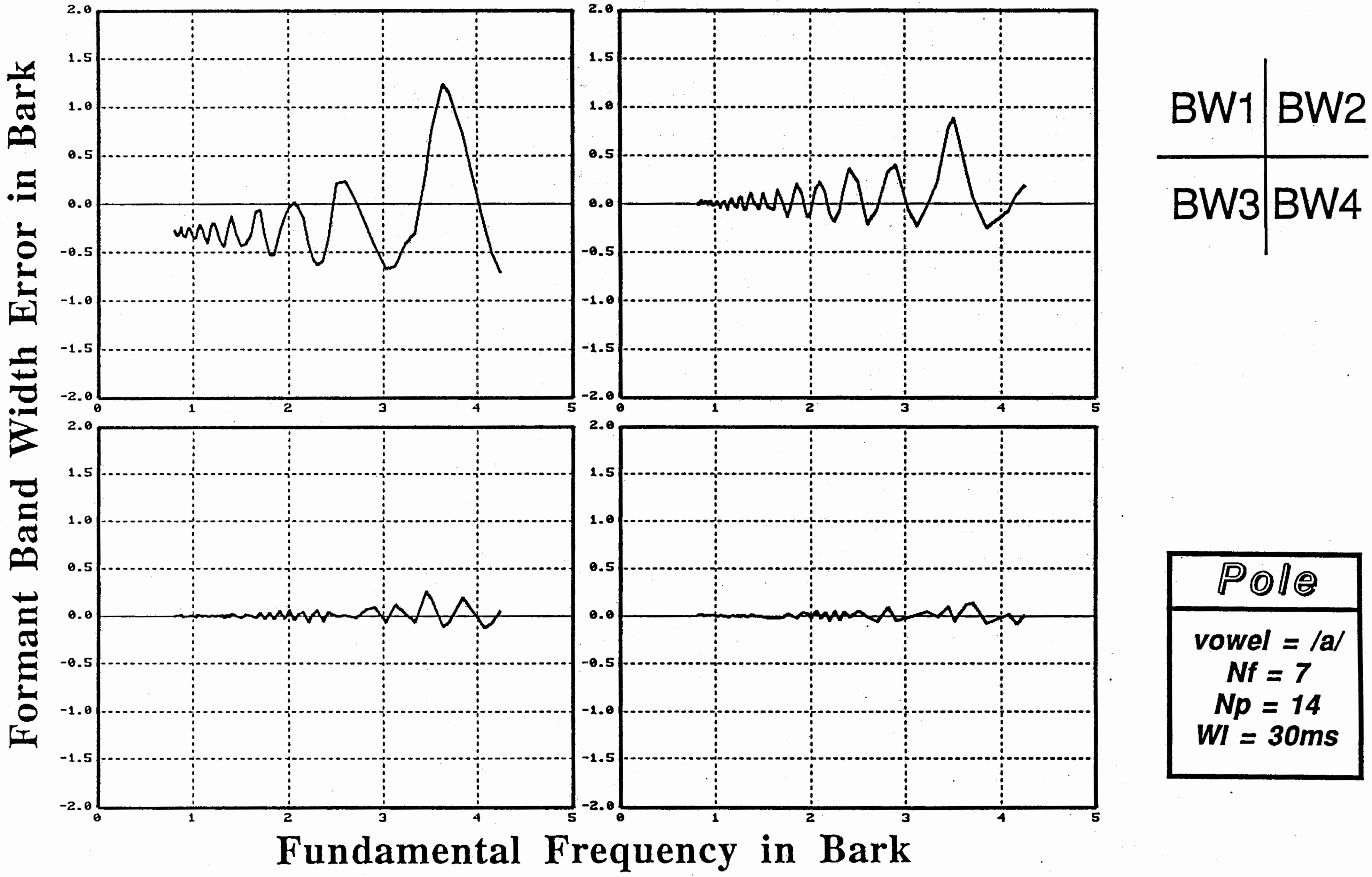


図5.1.9(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（5母音）

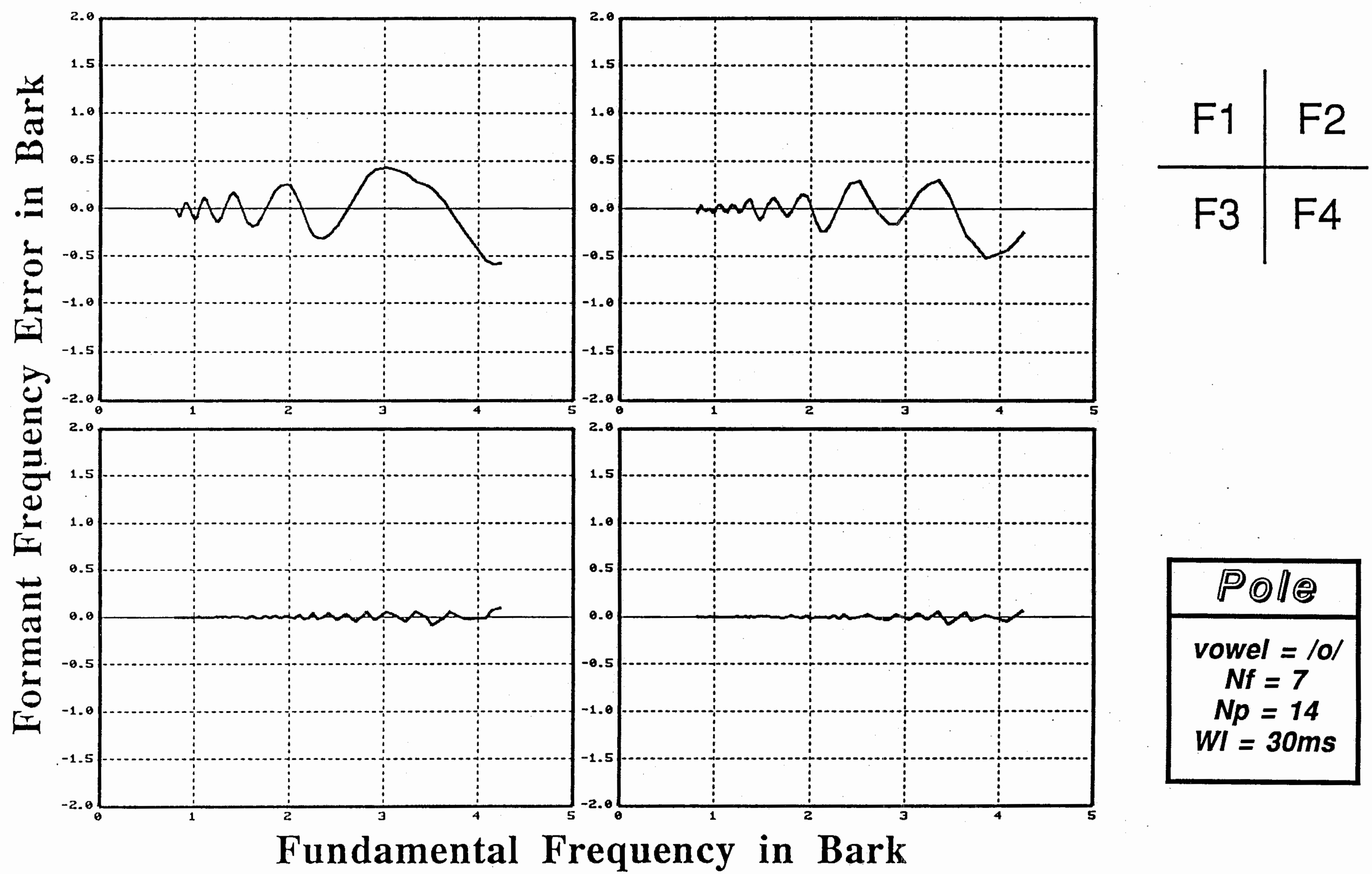


図5.1.10(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（5母音）

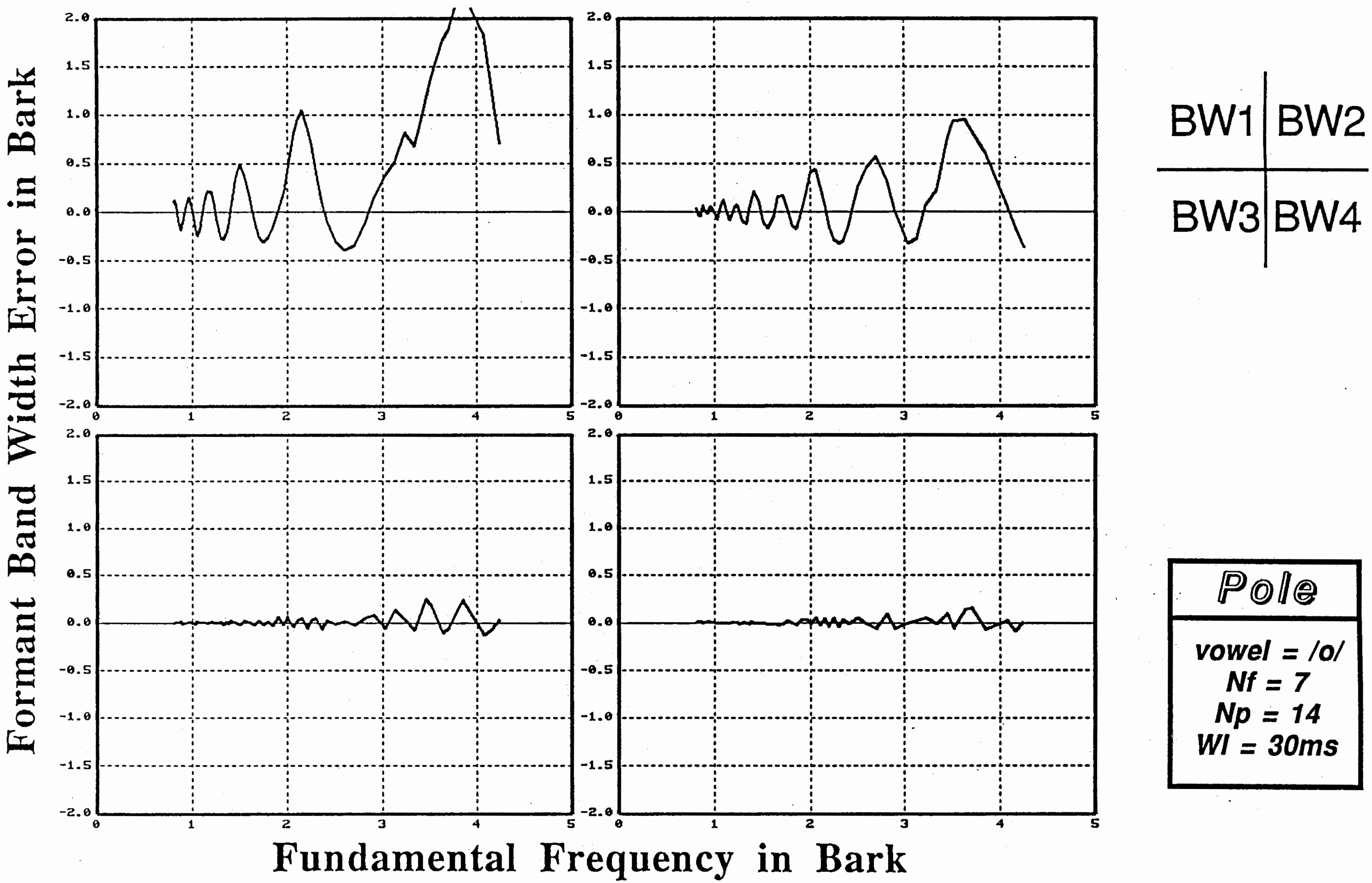


図5.1.10(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（5母音）

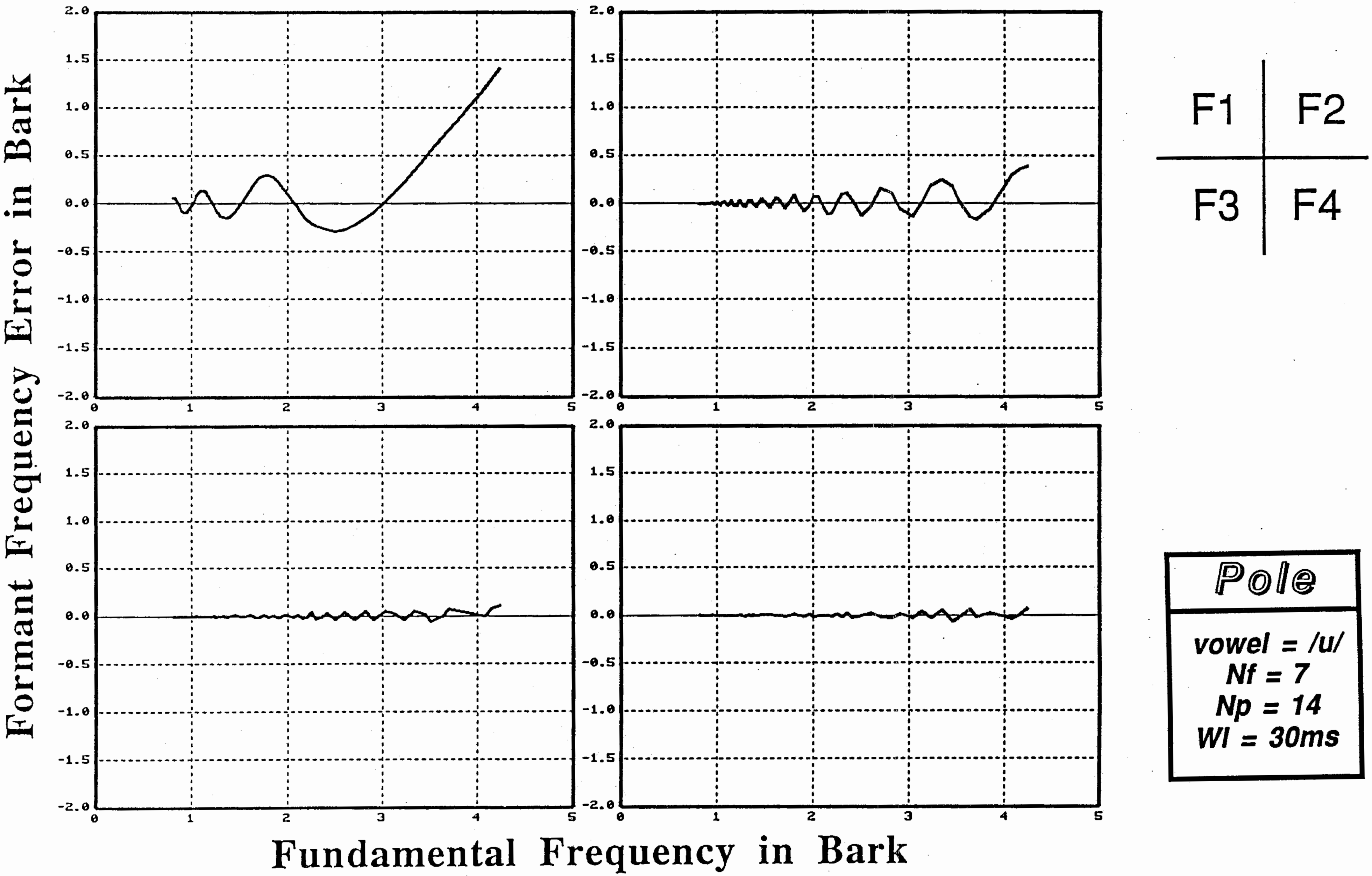


図5.1.11(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（5母音）

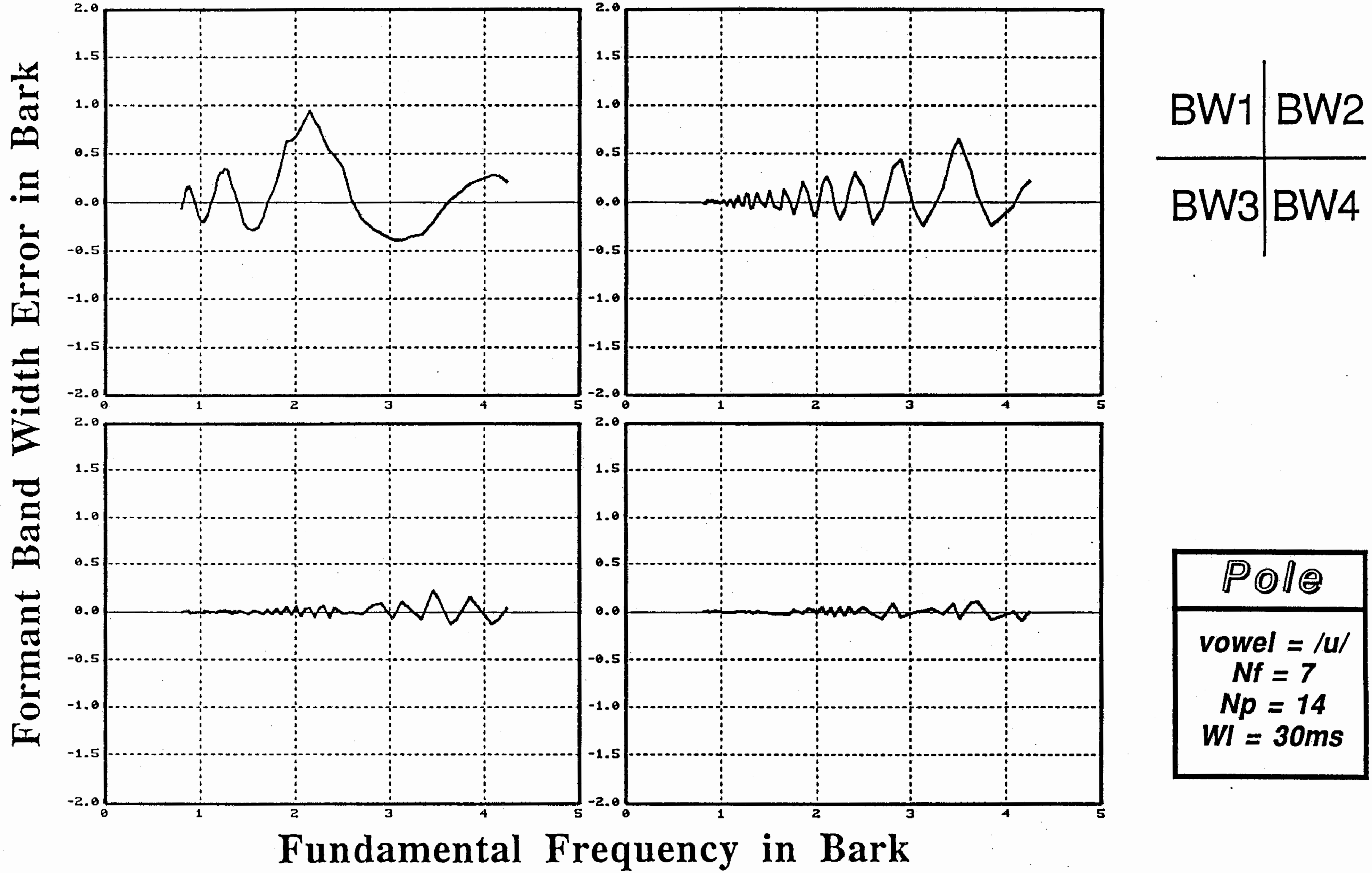


図5.1.11(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（５母音）

5.2 スペクトル包絡のピーク抽出の結果

本節では、ピーク抽出法を用いて合成5母音のフォルマント周波数、フォルマント帯域幅を推定する際に、分析パラメータと資料の合成パラメータとの関係が、推定値にどの様な影響を及ぼすか調べた結果を示す。分析条件は全て前節と同様である。

5.2.1 分析次数の影響

本節では、分析の次数がフォルマント周波数、フォルマント帯域幅の推定値に及ぼす影響を調べた結果を示す。

図5.2.1(a)～図5.2.5(a)は、フォルマント周波数の推定誤差の絶対値を分析次数Npを横軸にして描いたものである。各図とも左上から右下へ資料のF0が高いものから低いものへと並んでいる。

図5.2.1(b)～図5.2.5(b)は、フォルマント帯域幅の推定誤差の絶対値について同様に求めたものである。

図5.2.1、図5.2.2、図5.2.3、図5.2.4、図5.2.5は、それぞれ合成母音/i/、/e/、/a/、/o/、/u/の資料に対する結果である。

図5.2.6は、フォルマント周波数の推定誤差の絶対値がある閾値E以下であるような分析次数の上限と下限を、資料のF0を横軸にして描いたものである。この図ではE=0.5 Barkに設定した。図の配置は前節と同様、図3.1に示されるような、F1-F2平面上での各母音の位置関係に対応している。参考として、図4.2.14に示された中性母音に対するものも、図中の対応する位置に配置して再度示した。

5.2.2 基本周波数の影響

本節では、資料の基本周波数がフォルマント周波数、フォルマント帯域幅の推定値に及ぼす影響を調べた結果を示す。

図5.2.7(a)～図5.2.11(a)は、フォルマント周波数の推定誤差を資料のF0を横軸にして描いたものである。各図とも左上から右下へ第1～第4フォルマントに対する誤差の図を示す。

図5.2.7(b)～図5.2.11(b)は、フォルマント帯域幅について同様に求めたものである。

図5.2.7、図5.2.8、図5.2.9、図5.2.10、図5.2.11は、それぞれ合成母音/i/、/e/、/a/、/o/、/u/の資料に対する結果を示している。

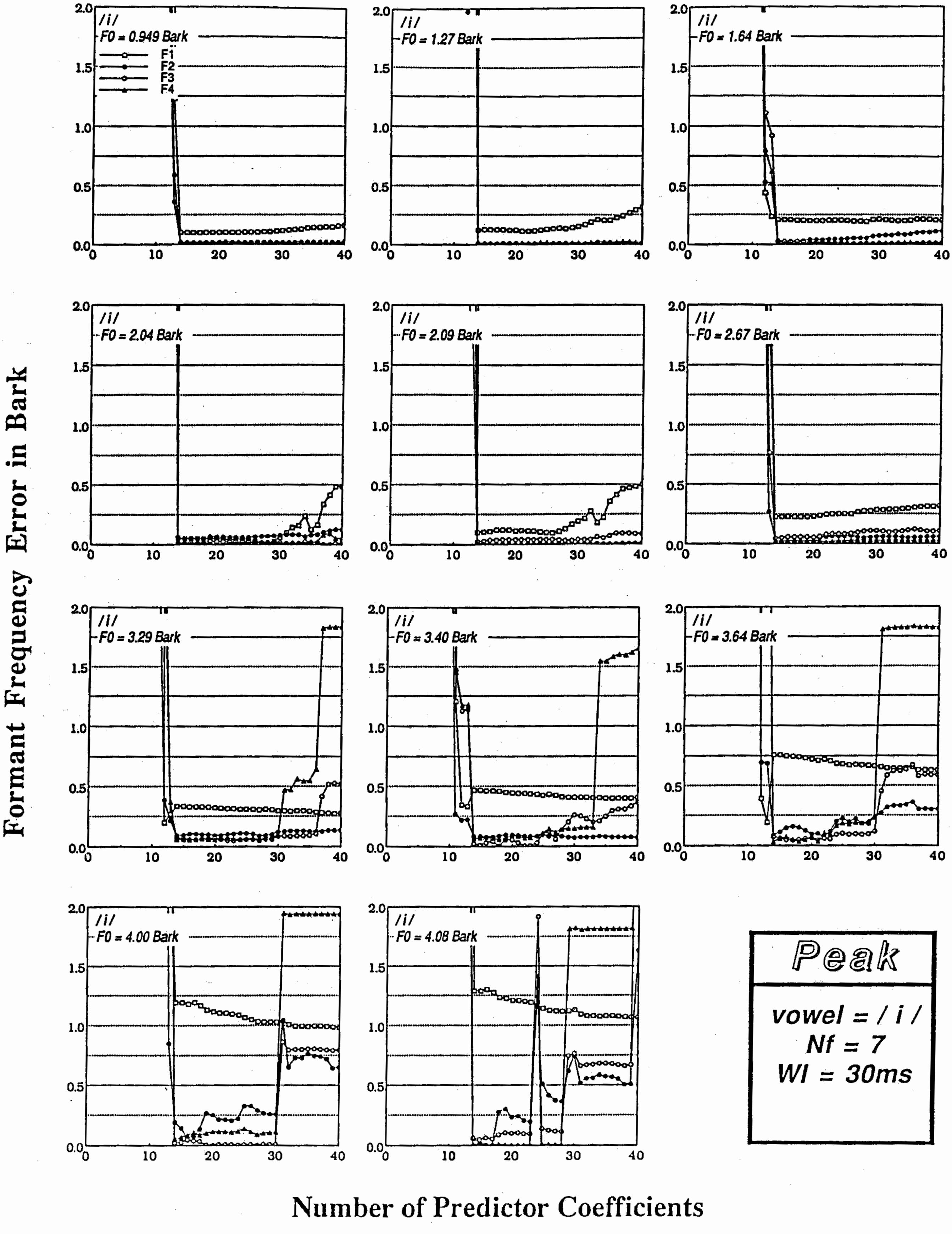


図5.2.1(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（５母音）

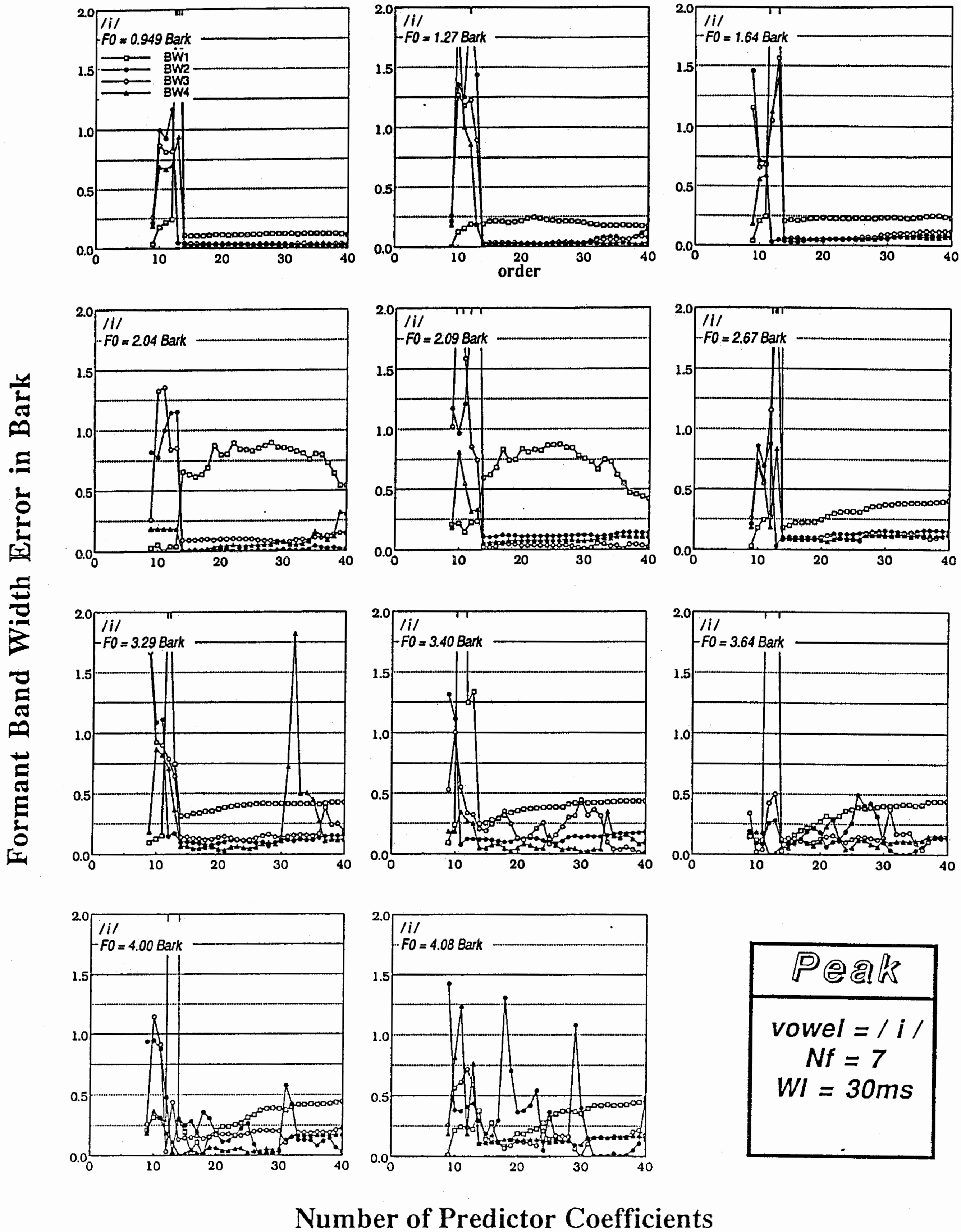


図5.2.1(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（５母音）

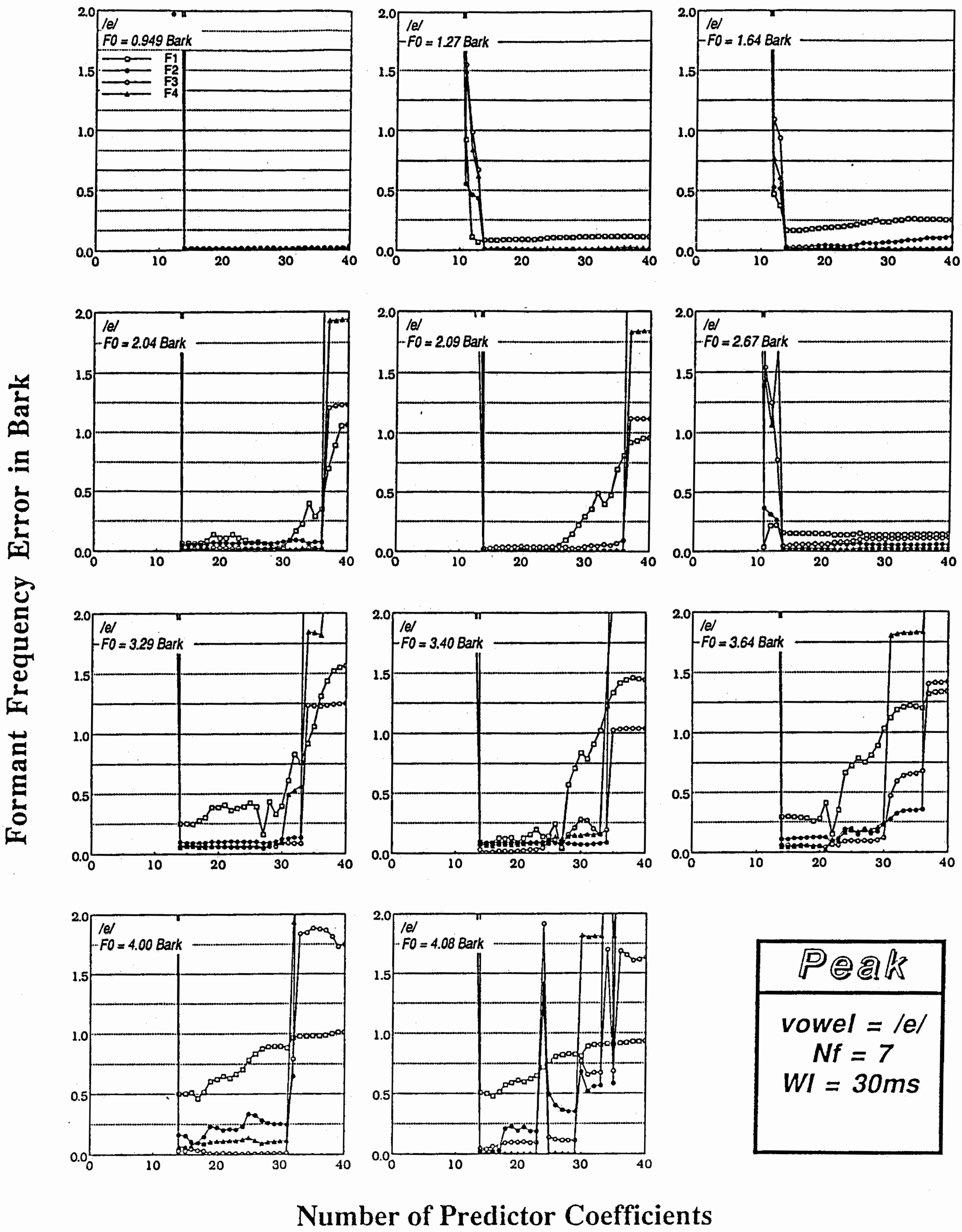


図5.2.2(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（５母音）

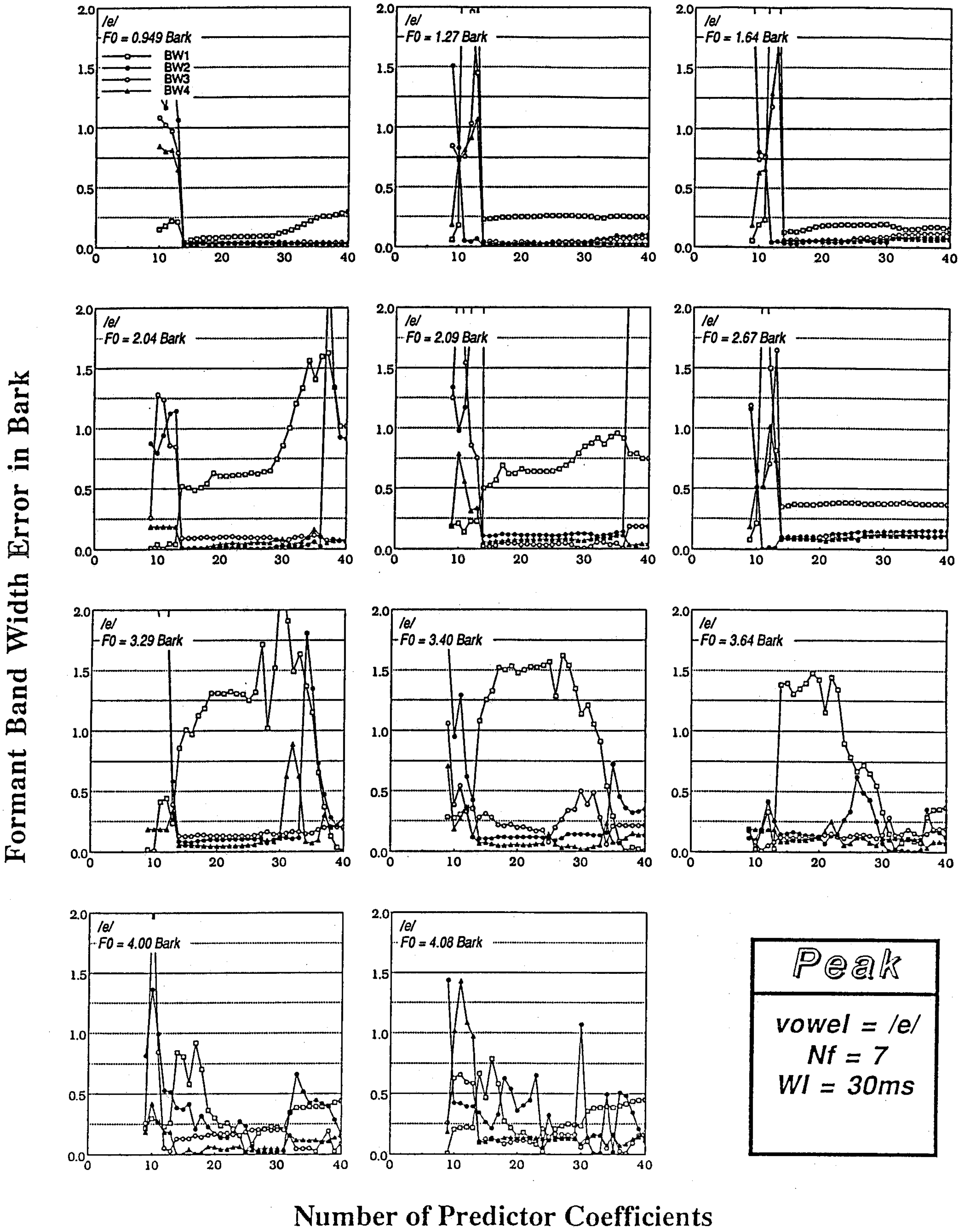


図5.2.2(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（５母音）

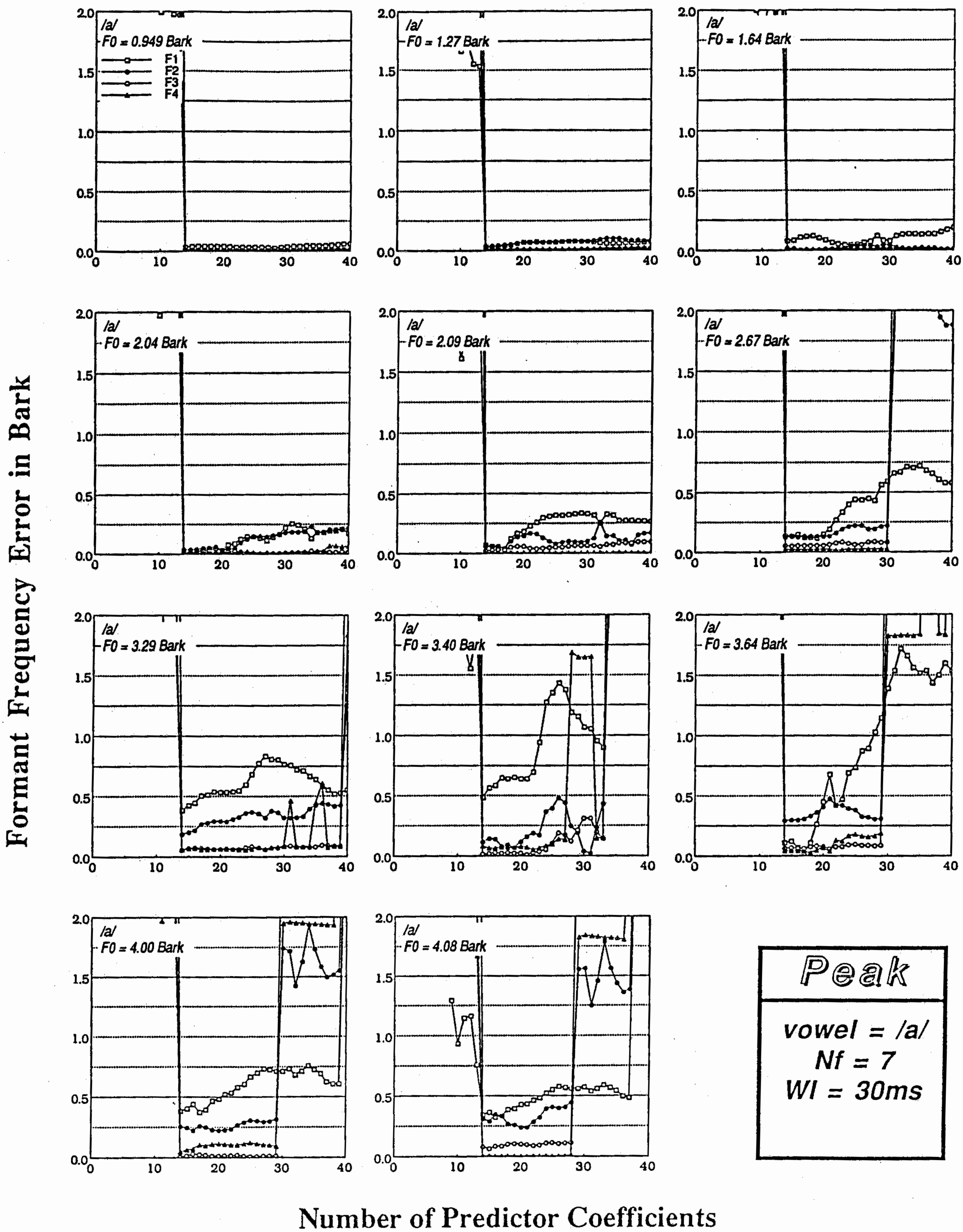


図5.2.3(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（5母音）

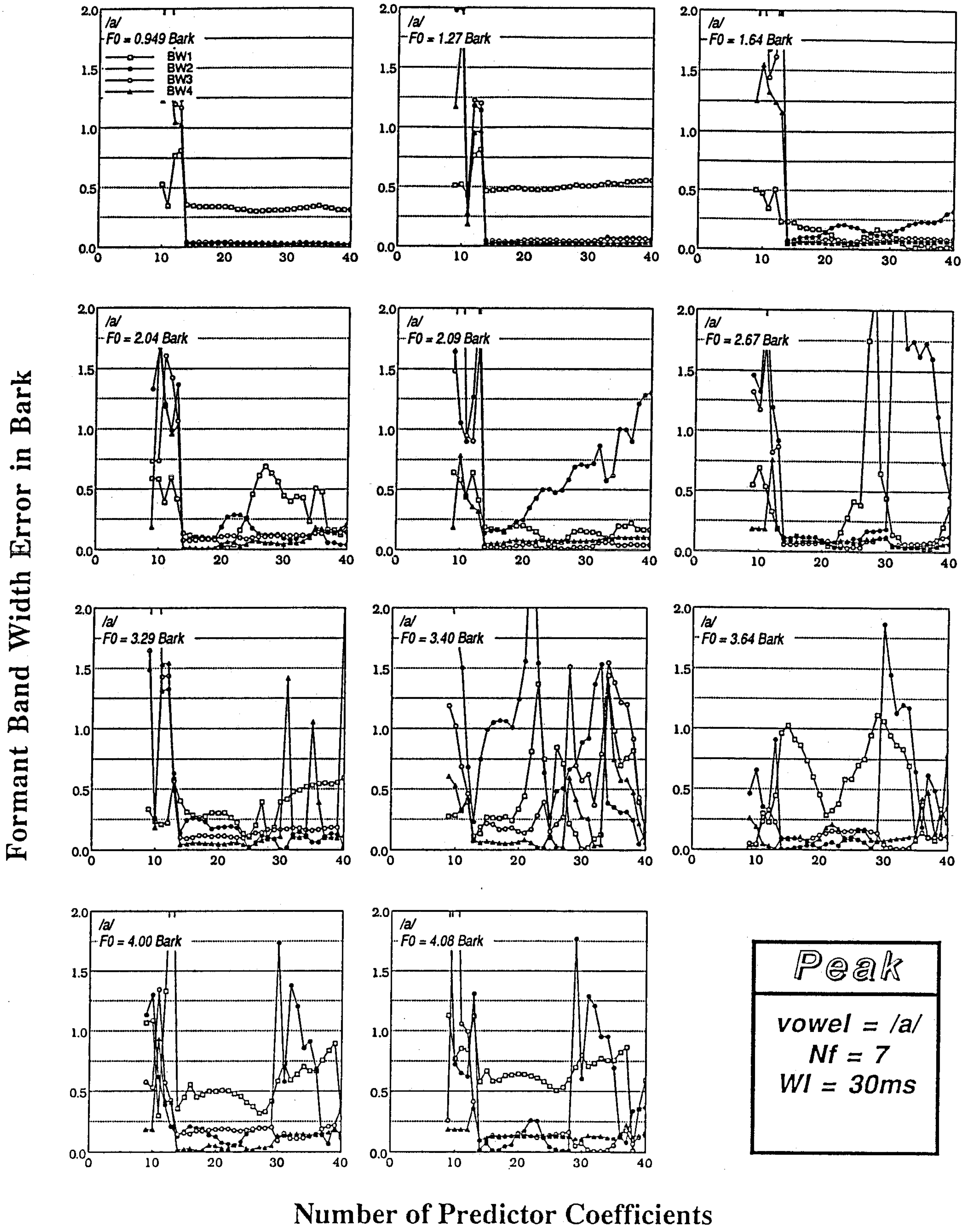


図5.2.3(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（５母音）

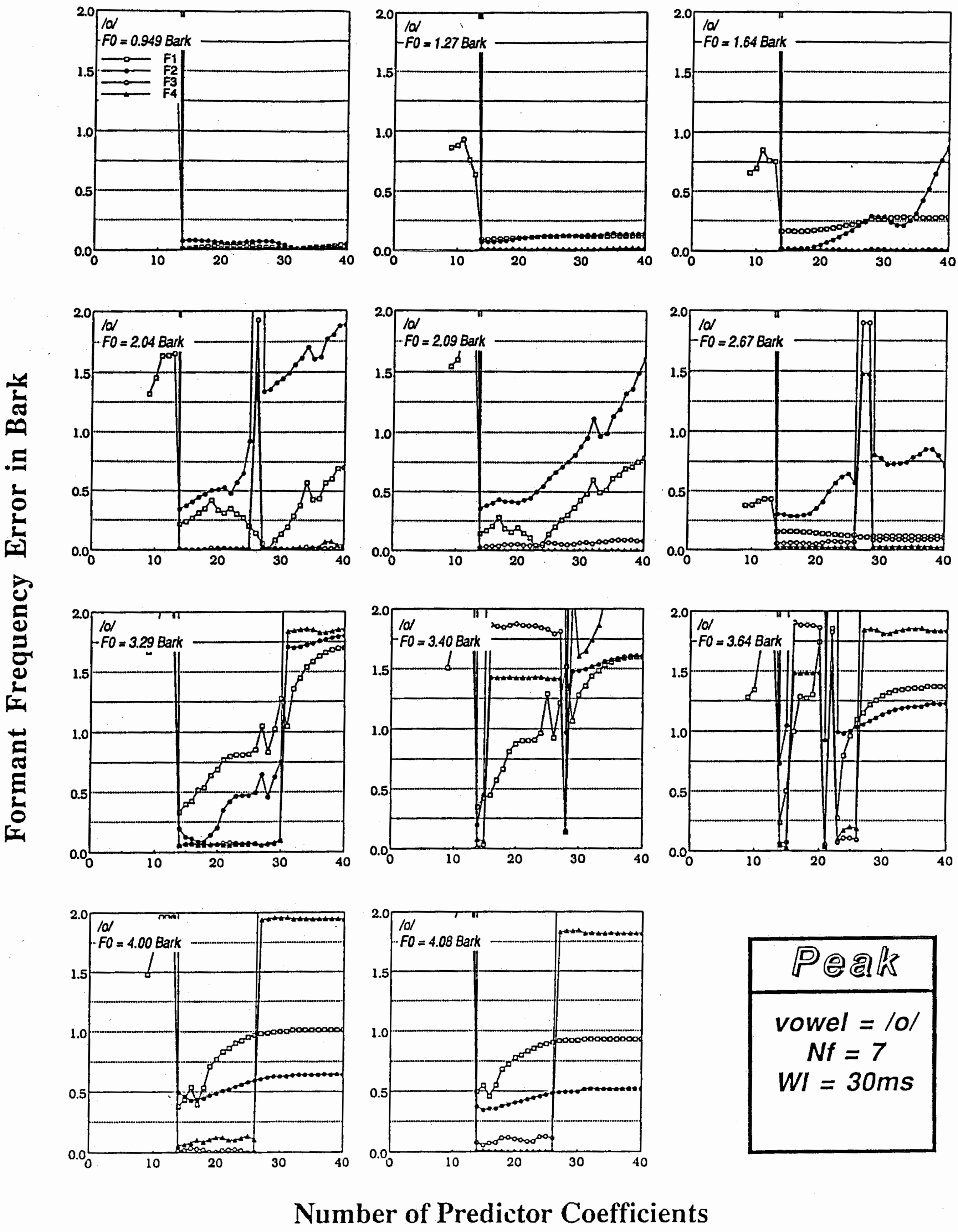


図5.2.4(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（５母音）

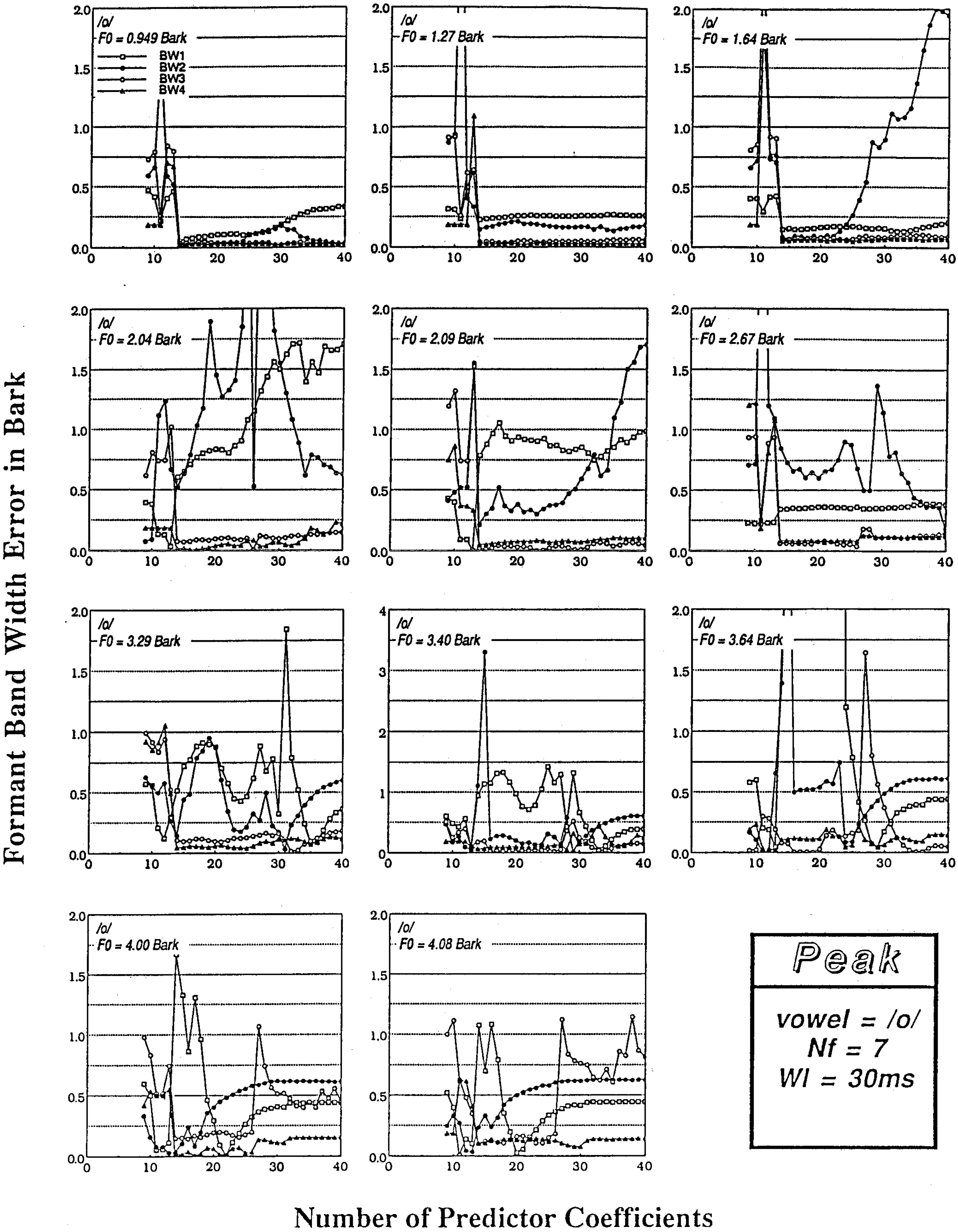


図5.2.4(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（５母音）

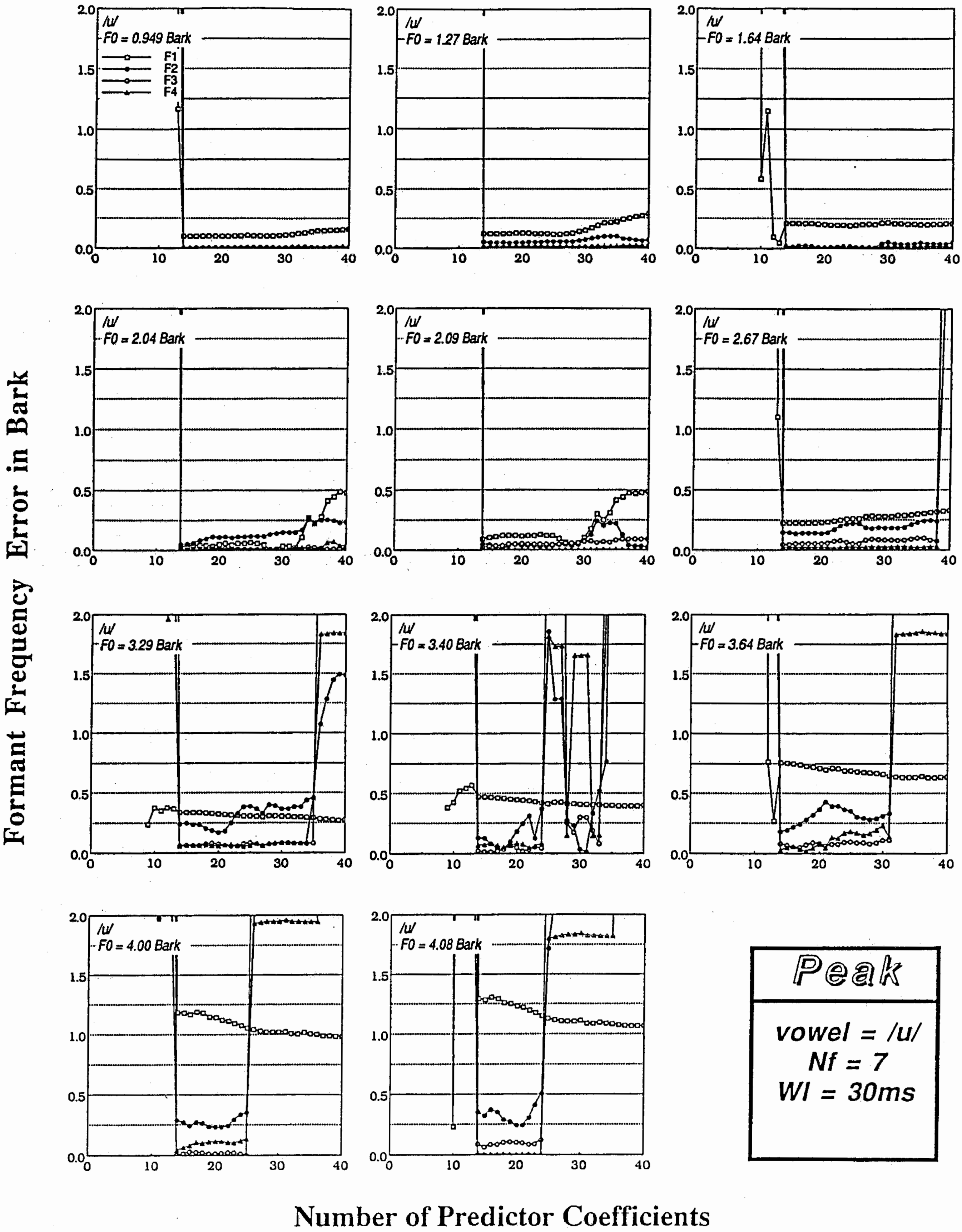


図5.2.5(a) フォルマント周波数の推定誤差の絶対値と分析次数との関係（５母音）

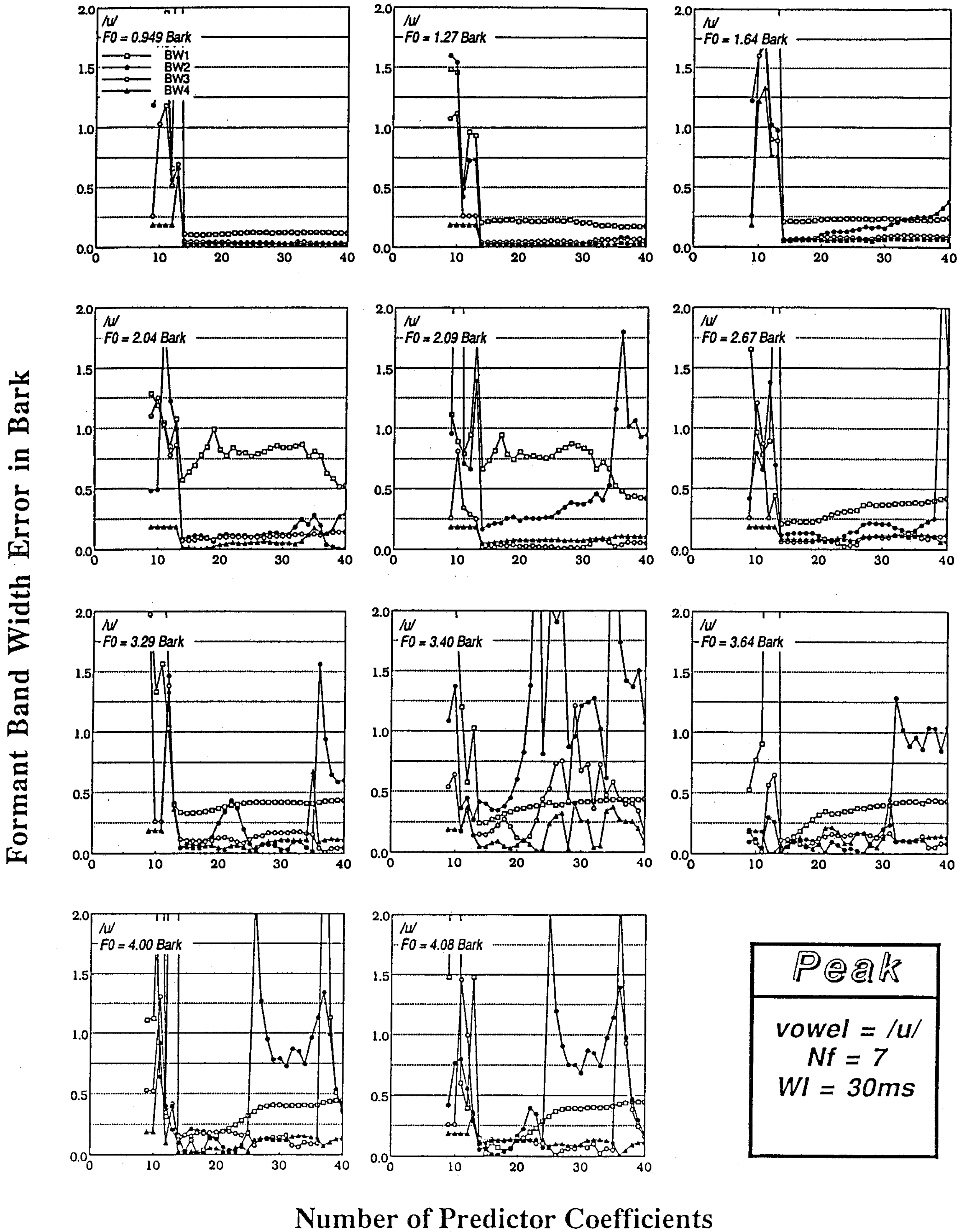


図5.2.5(b) フォルマント帯域幅の推定誤差の絶対値と分析次数との関係（5母音）

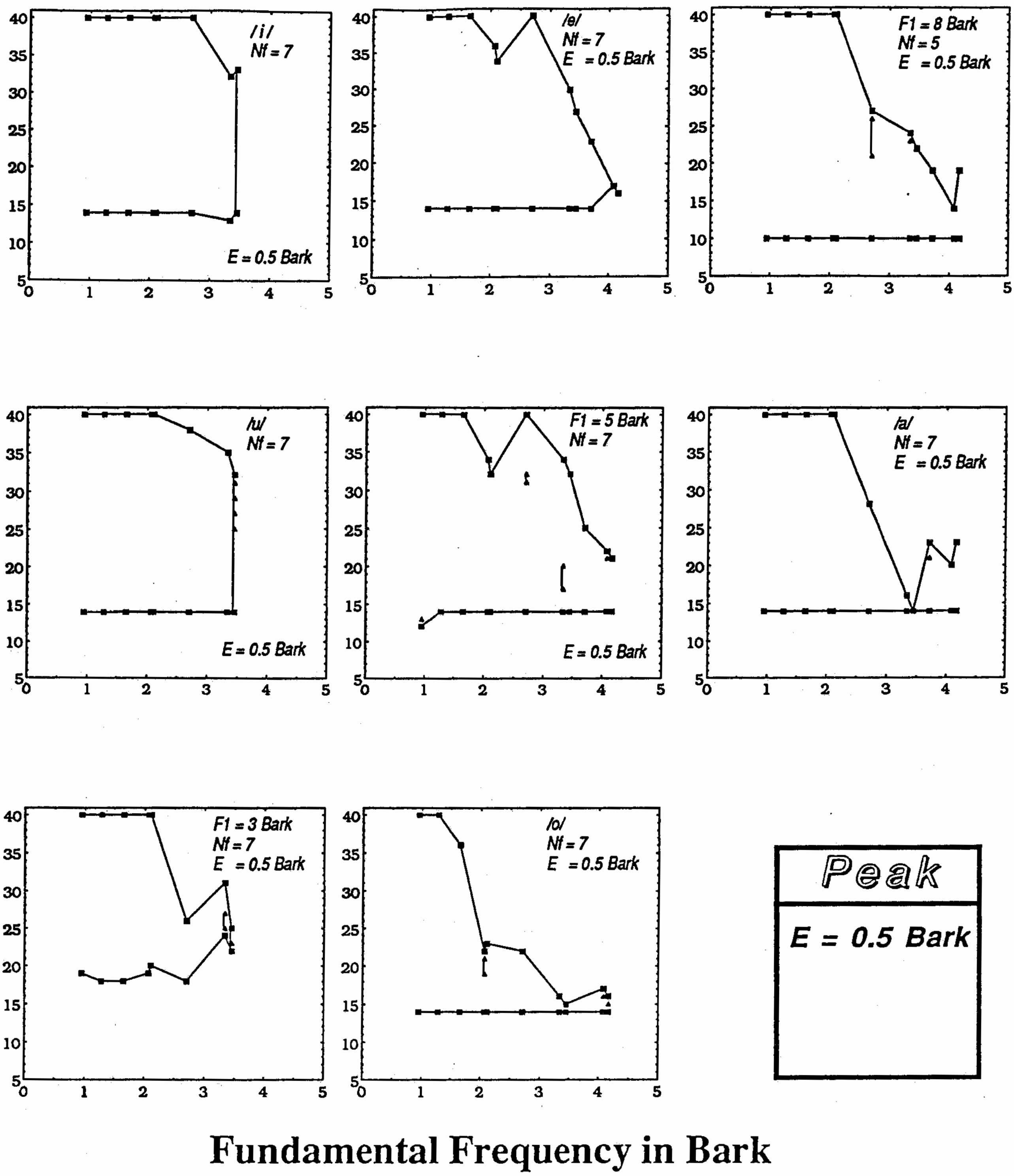


図5.2.6 フォルマント周波数の推定誤差の絶対値が0.5Bark以下である分析次数の上限、下限と資料の基本周波数との関係（５母音）

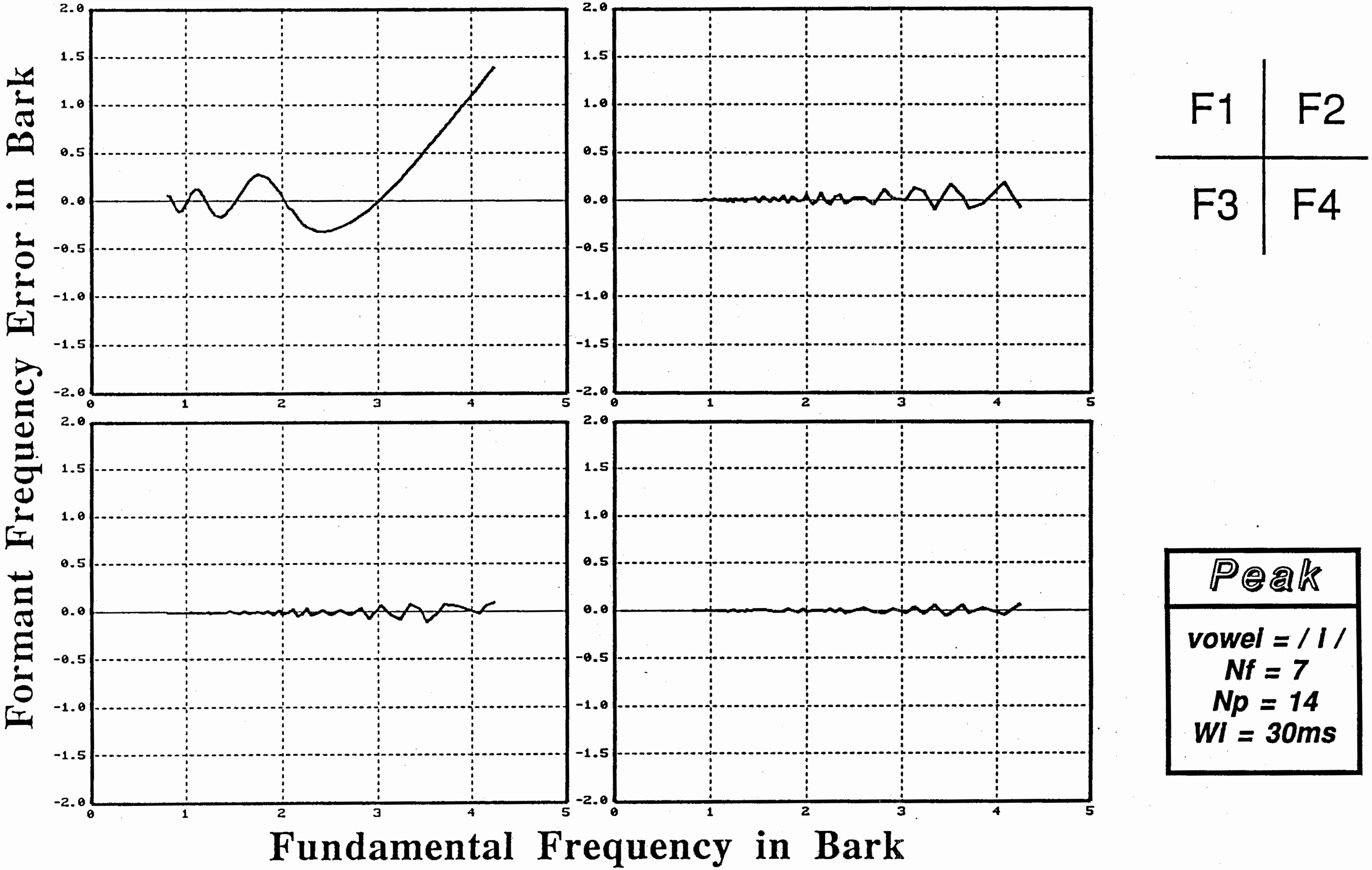


図5.2.7(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（５母音）

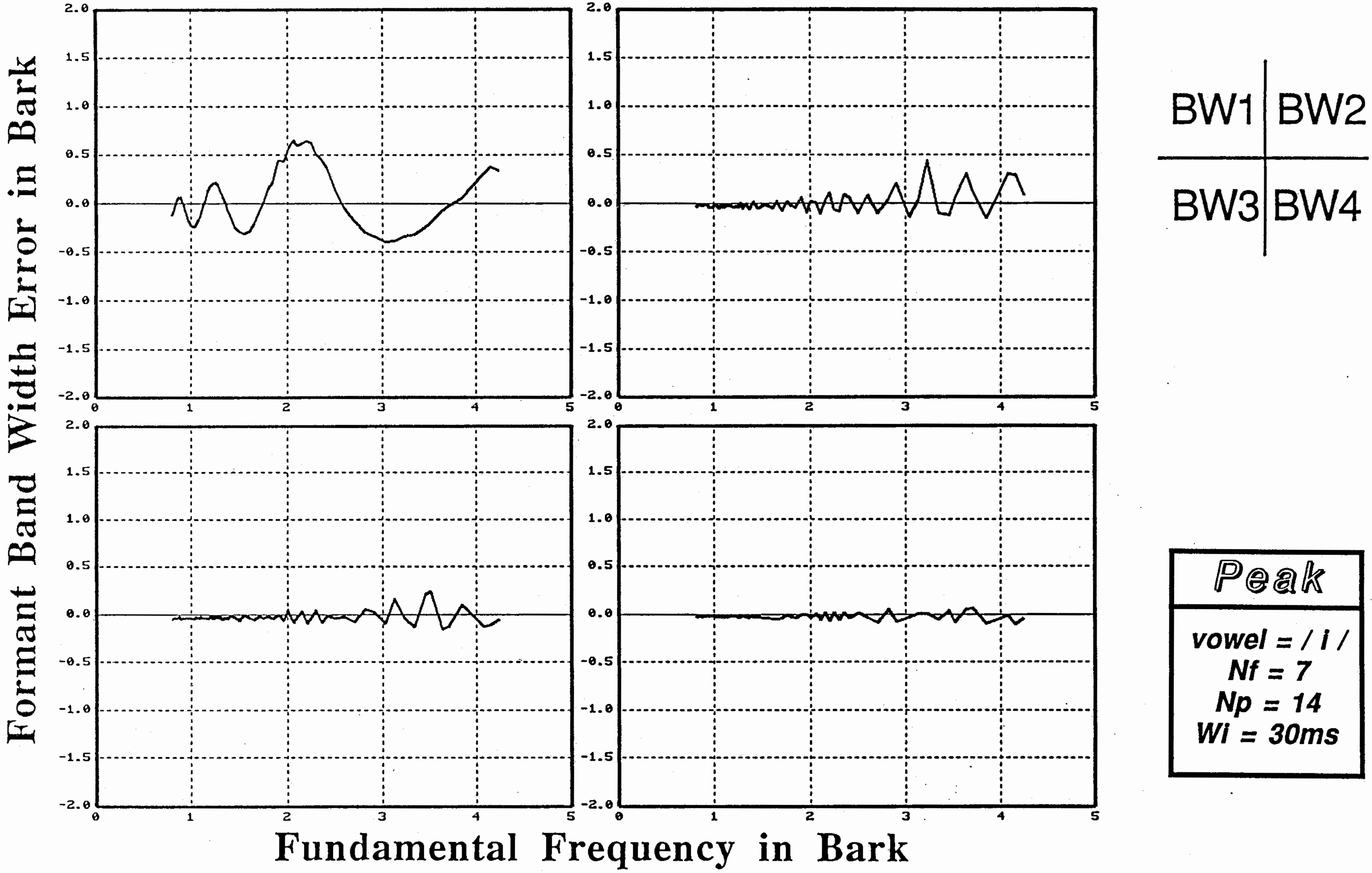


図5.2.7(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（５母音）

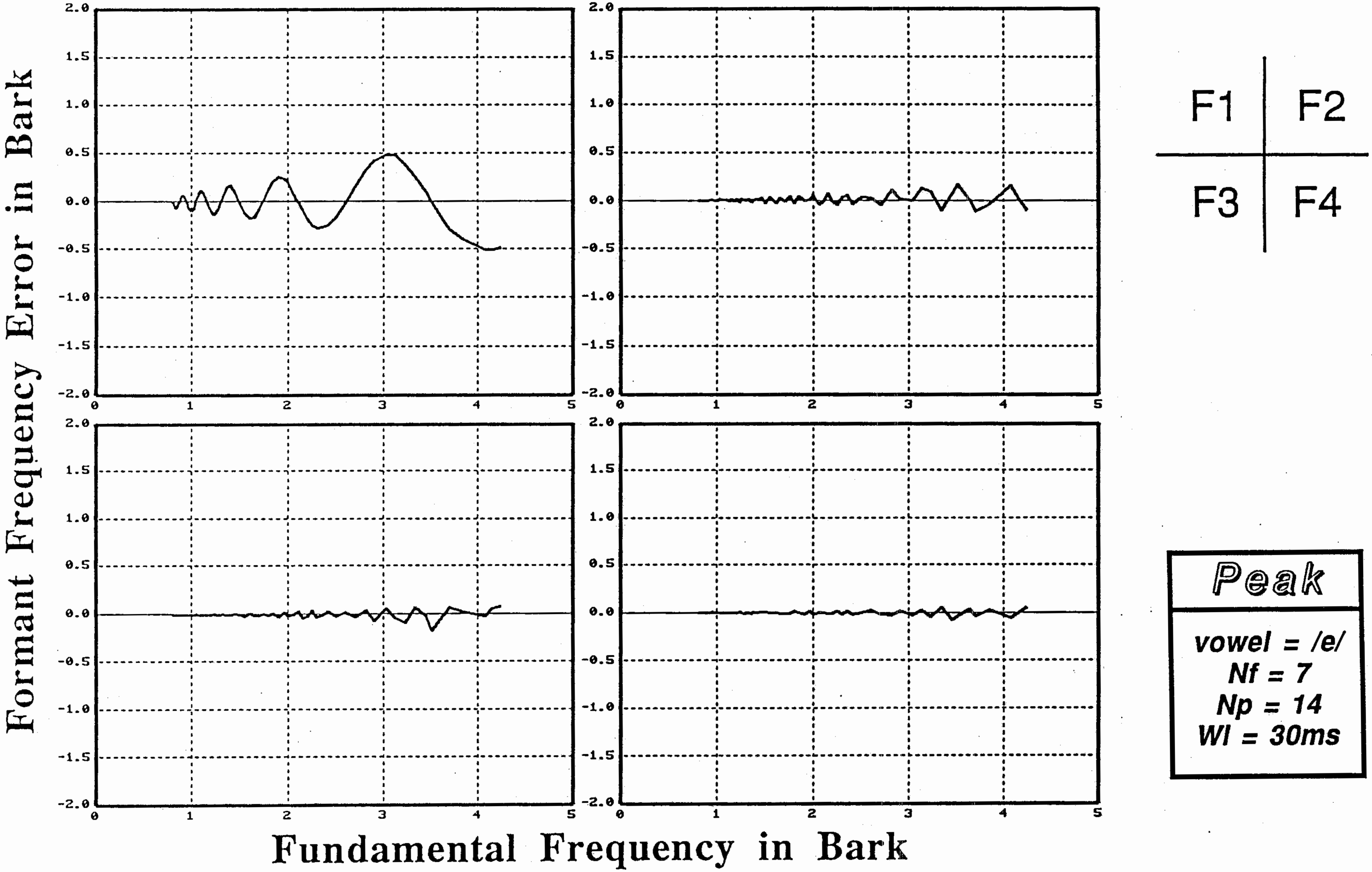


図5.2.8(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（５母音）

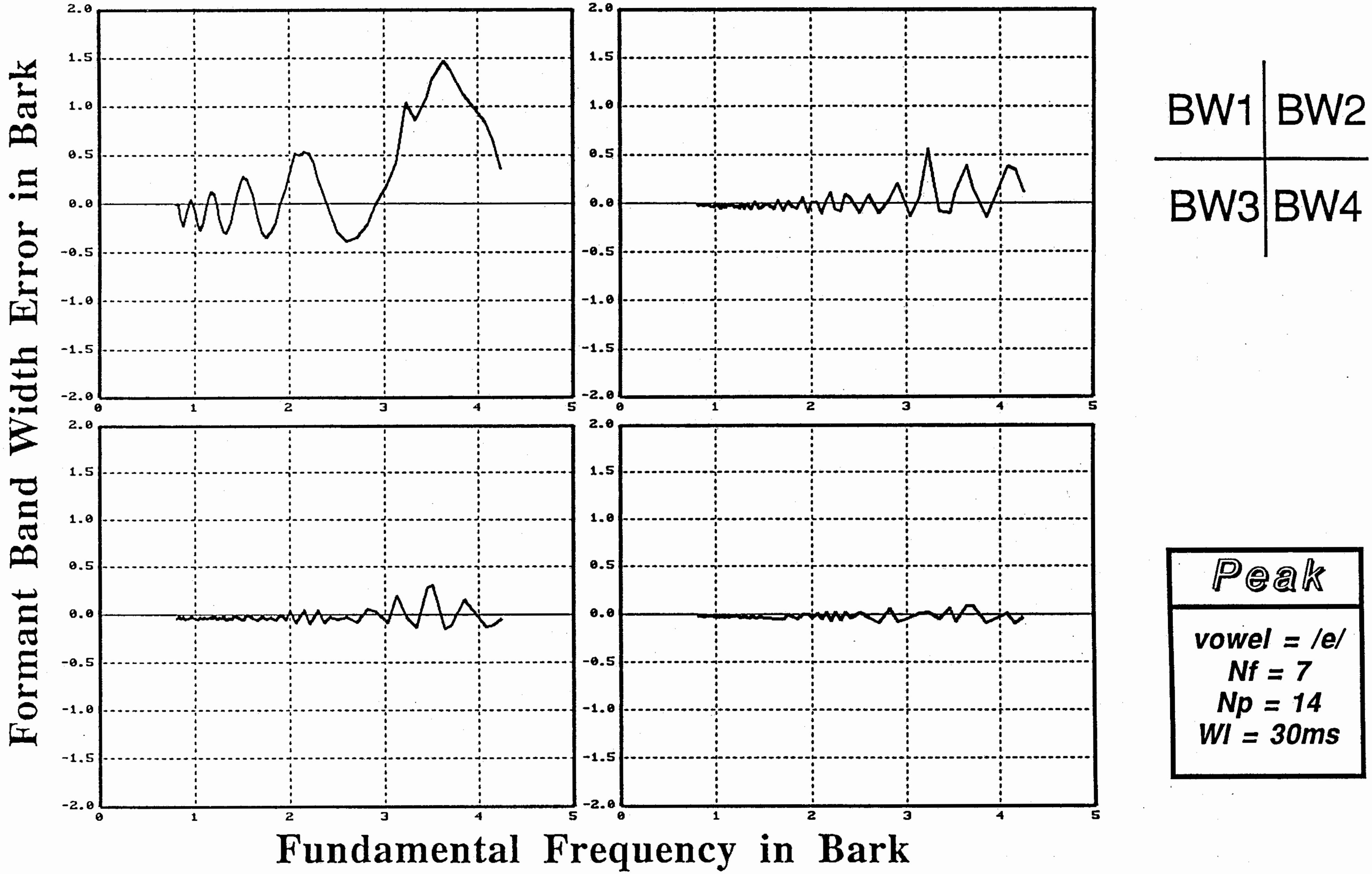


図5.2.8(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（５母音）

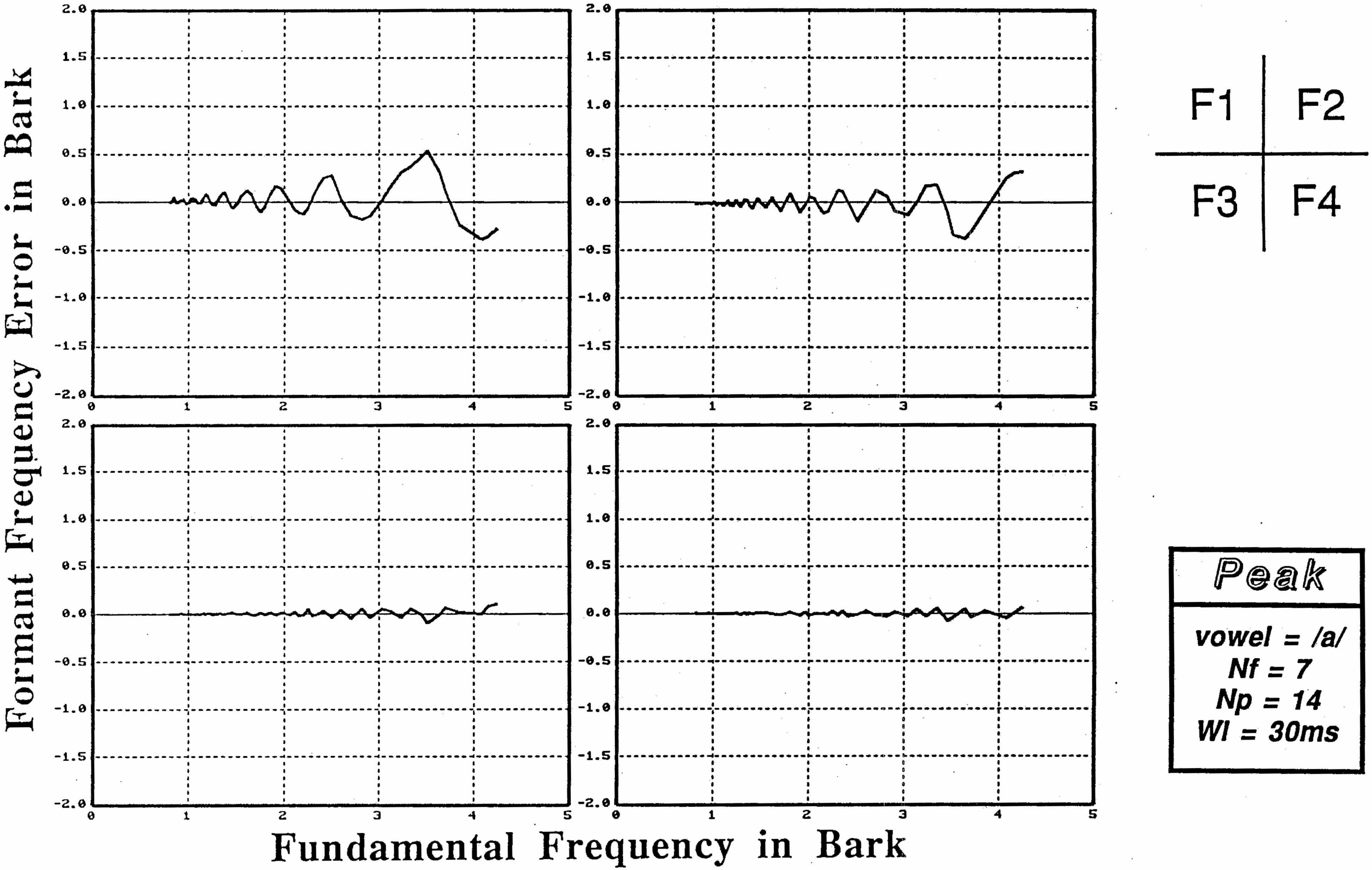


図5.2.9(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（5母音）

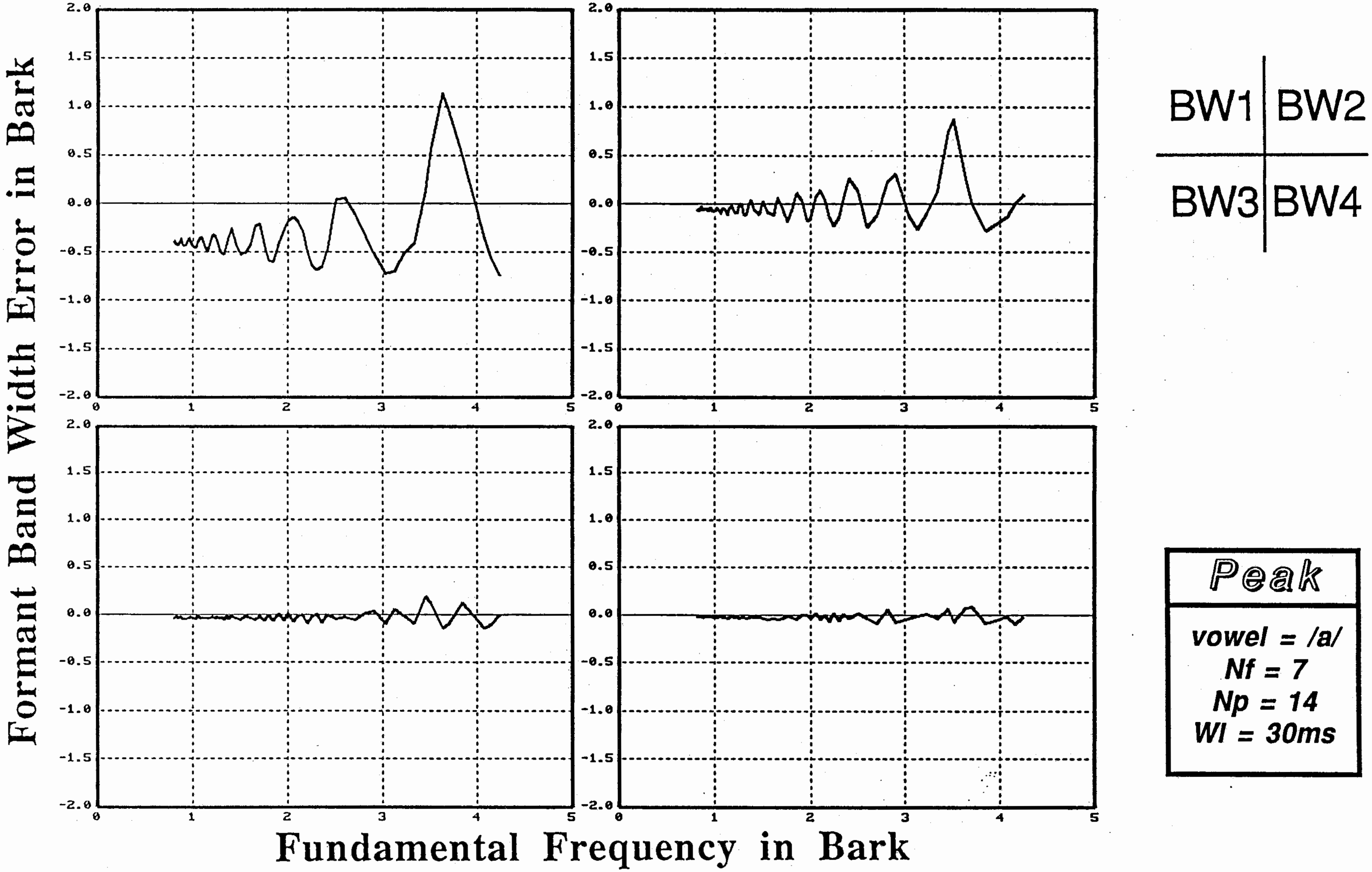


図5.2.9(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（５母音）

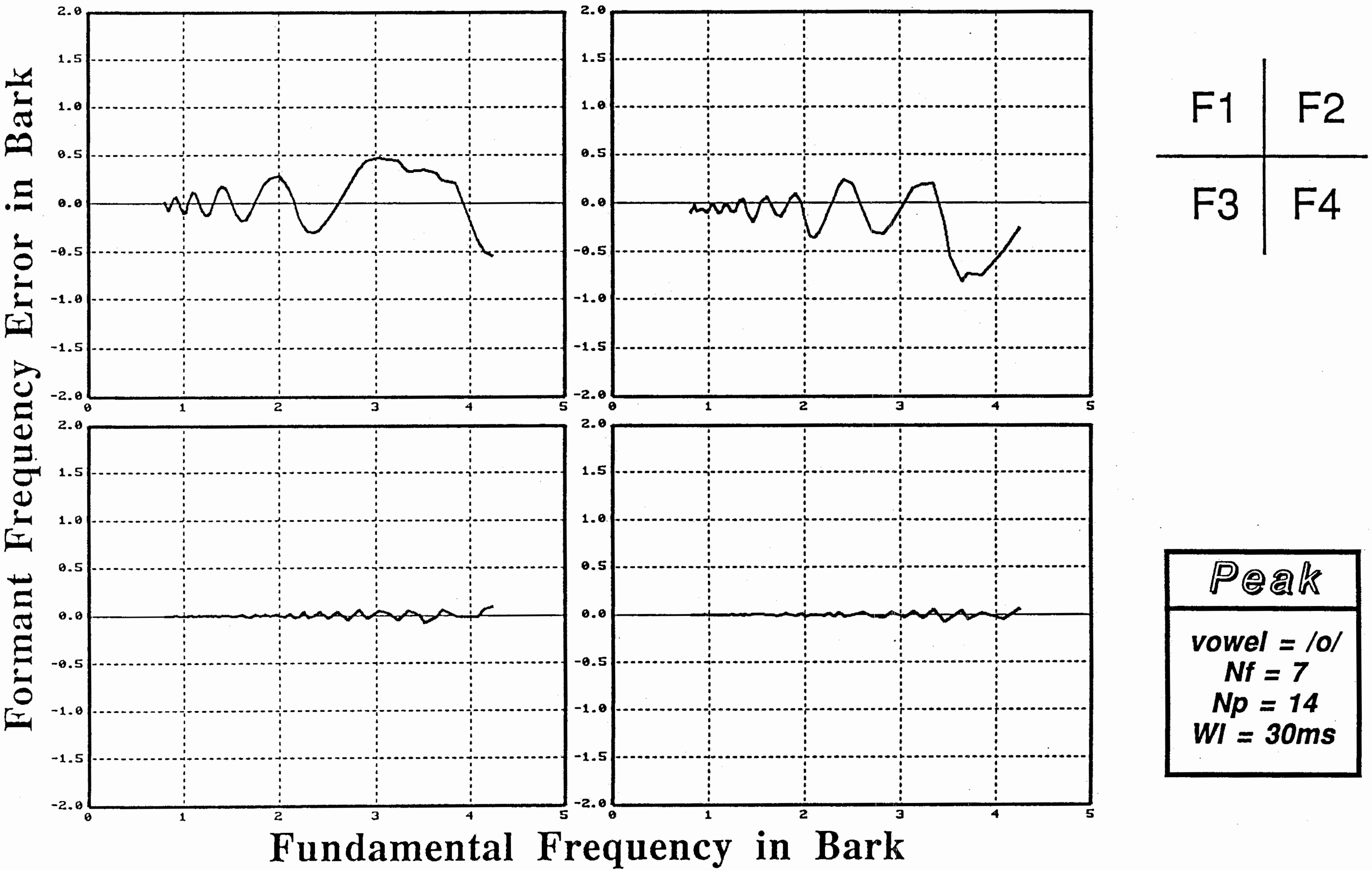


図5.2.10(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（５母音）

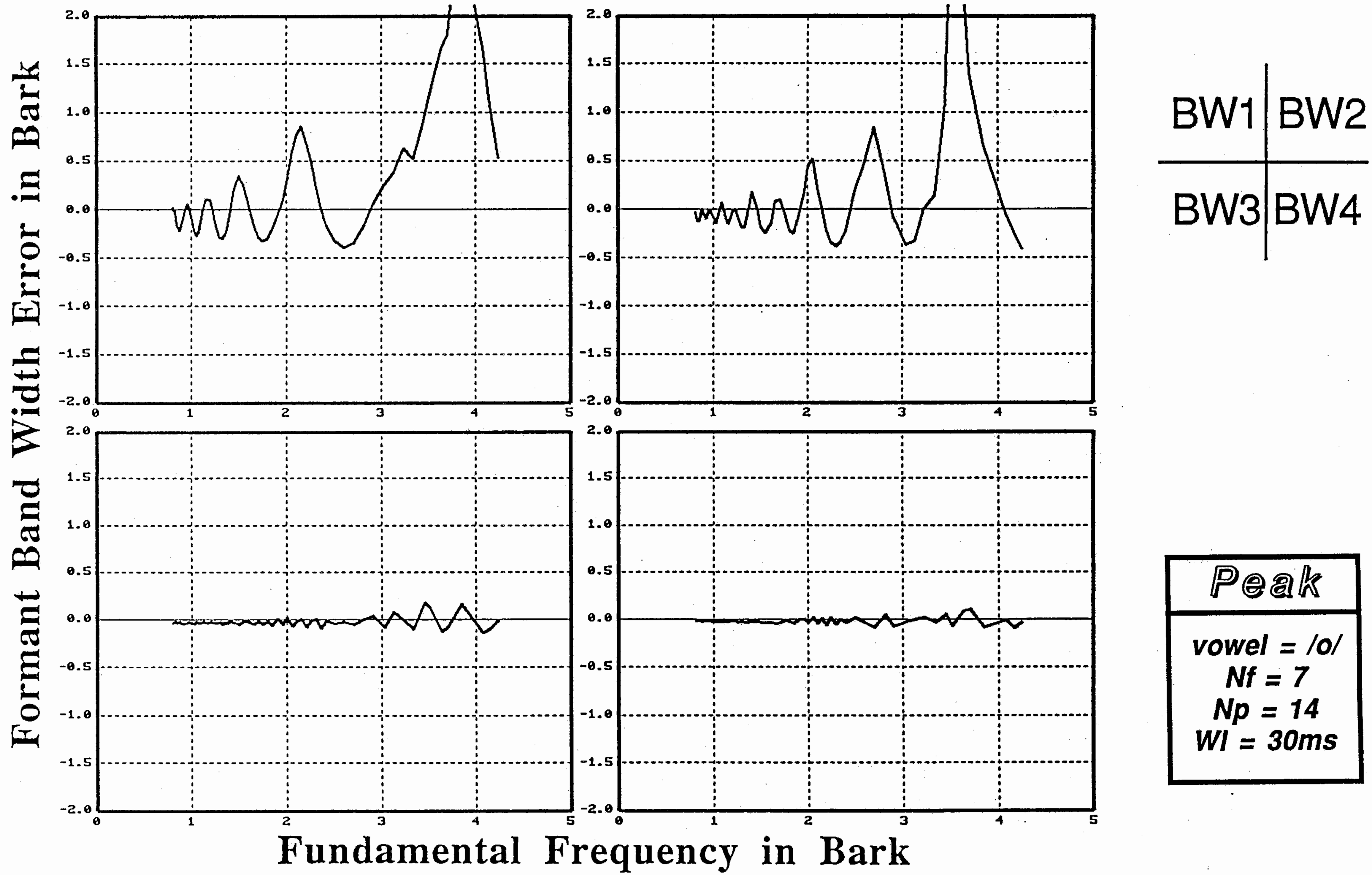


図5.2.10(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（5母音）

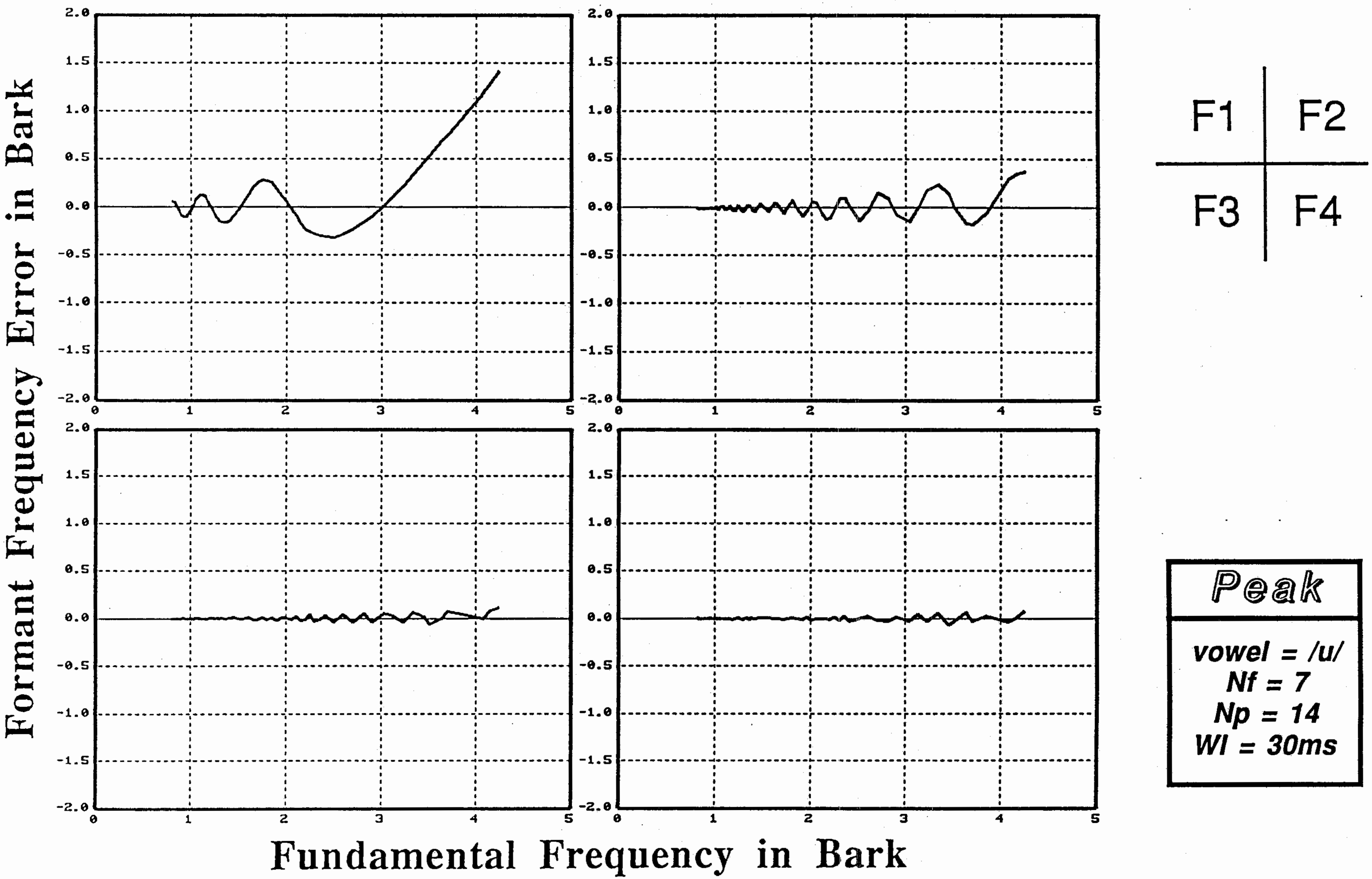


図5.2.11(a) フォルマント周波数の推定誤差と資料の基本周波数との関係（５母音）

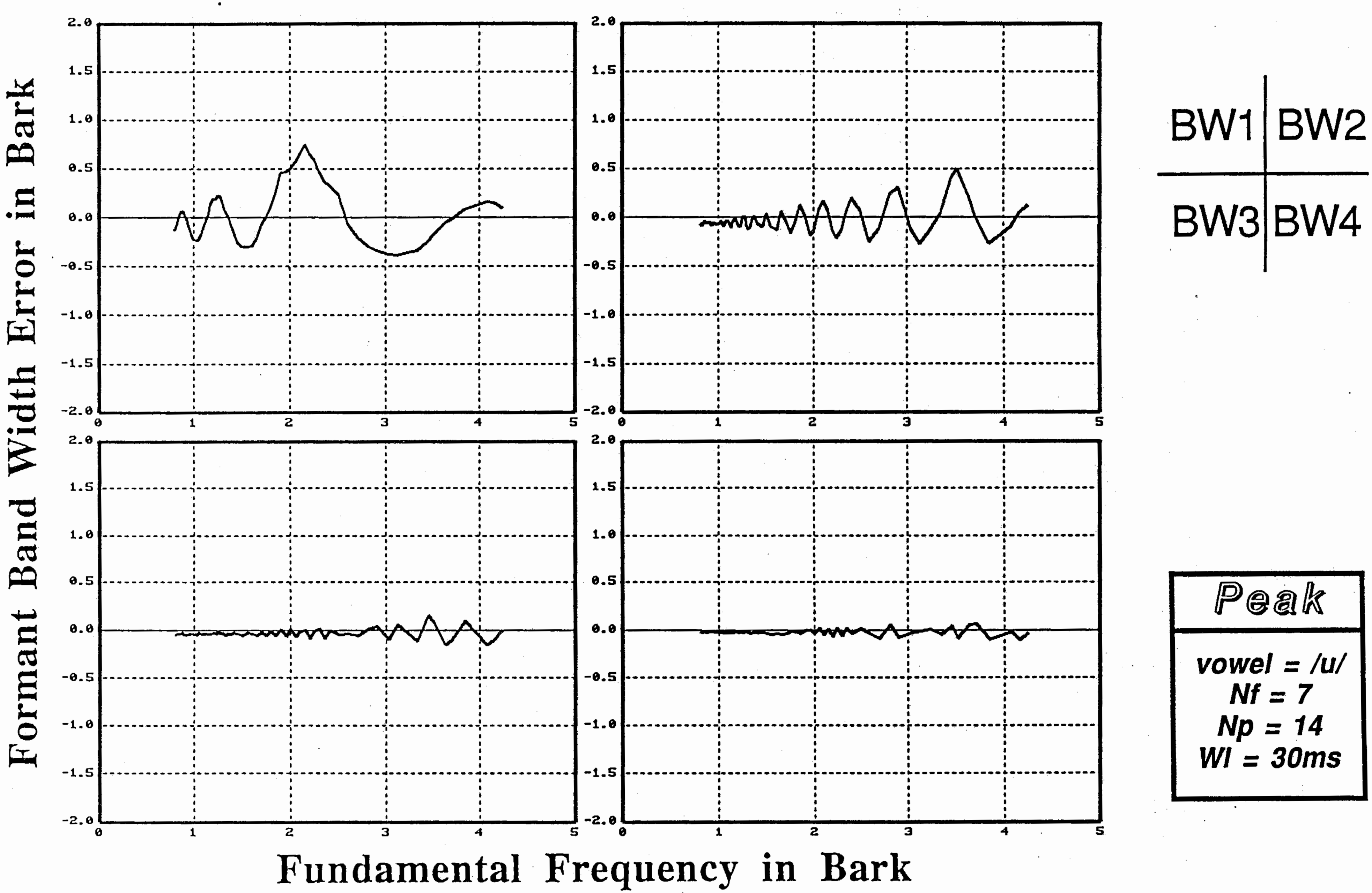


図5.2.11(b) フォルマント帯域幅の推定誤差と資料の基本周波数との関係（５母音）

付録A

1．対数尤度の導出および対数尤度によるスペクトルマッチング
2．Töplitz行列を係数行列に持つ連立一次方程式の求解アルゴリズム
3．Bairstow法による高次代数方程式の求解アルゴリズム
4．全極形パワスペクトルのローカルピーク探索アルゴリズム

付録Ａ－１　対数尤度の導出および対数尤度によるスペクトルマッチング

音声波形の標本値を定常ガウス過程であるとみなし、この仮定のもとで対数尤度の近似表現を導出する。続いて、求めた対数尤度を尺度としたスペクトルマッチングの方法を示し、その物理的意味について述べる。

(1) 対数尤度の導出

音声波形が定常ガウス過程であると仮定すれば、ある時点kでの標本値s(k)の尤度（仮定したモデルから得られる事後確率）は、ガウス分布の定義より次のようになる。

$$p(s(k)\,|\,\Theta) = \frac{1}{\sqrt{2\pi}\sqrt{\sigma^2}} \exp\left\{-\frac{1}{2} s(k)\,\sigma^{-2}\right\} \qquad (A1.1)$$

ここでs(n)の平均値は0と仮定しており、σ^2はs(n)の分散である。

同様に、N個の標本値を $s=(s(0),s(1),...,s(N-1))$ のようにベクトルで表わすと、sの尤度は次のように書ける。

$$p(s\,|\,\Theta) = \frac{1}{(\sqrt{2\pi})^N\sqrt{|C|}} \exp\left\{-\frac{1}{2}\, s\, C^{-1}\, s^t\right\} \qquad (A1.2)$$

ここで、Cはsの共分散行列、s^tはsの転置を示す。

本文の仮定により、母集団のパワスペクトルは全極型なので、次のように書くことができる。

$$\begin{aligned} T(\omega) &= \sigma^2/2\pi \cdot 1/\left|\prod_{i=1}^{p}(1 - z_i z^{-i})\right|^2 \\ &= \sigma^2/2\pi \cdot 1/\left|1+\alpha_1 z^{-1}+...+\alpha_p z^{-p}\right|^2 \\ &= \sigma^2/2\pi \cdot 1/\left(A_0+2A_1\cos\omega+2A_2\cos(2\omega)+...+2A_p\cos(p\omega)\right) \end{aligned} \qquad (A1.3)$$

ここで、$z=-\exp(j\omega)$, $-\pi \leqq \omega \leqq \pi$、$A_j$は$\{\alpha_i\}$の自己相関関数である。

したがって、G,F(s)を(A1.5),(A1.6)式のようにおくことで、$p(s\,|\,\Theta)$は次式のように表わされる。

$$p(s\,|\,\Theta) = \frac{G}{(2\pi\sigma^2)^{N/2}} \exp\left\{-\frac{F(s)}{2\sigma^2}\right\} \qquad (A1.4)$$

$$G = |\sigma^2 C^{-1}|^{1/2} \qquad (A1.5)$$

$$\begin{aligned} F(s) = {} & A_0\sum_{i=0}^{N-1}s(i)^2 + 2A_1\sum_{i=0}^{N-2}s(i)s(i+1) + \ ... \ + 2A_p\sum_{i=0}^{N-p-1}s(i)s(i+p) \\ & - \{(\alpha_1{}^2+...+\alpha_p{}^2)(s(1)^2+s(N)^2)+(\alpha_2{}^2+...+\alpha_p{}^2)(s(2)^2+s(N-1)^2) \\ & + 2(\alpha_1\alpha_2+...+\alpha_{p-1}\alpha_p)(s(1)s(2)+s(N)s(N-1)) \\ & + (\alpha_3{}^2+...+\alpha_p{}^2)(s(3)^2+s(N-2)^2) \\ & + 2(\alpha_2\alpha_3+...+\alpha_{p-1}\alpha_p)(s(2)s(3)+s(N-1)s(N-2)) \\ & + 2(\alpha_1\alpha_3+...+\alpha_{p-2}\alpha_p)(s(1)s(3)+s(N)s(N-2)) \\ & + \cdots \\ & \qquad\qquad \cdots\cdots \\ & + \alpha_p{}^2(s(p)^2+s(N-p+1)^2)+... \\ & + 2\alpha_2\alpha_p(s(2)s(p)+s(N)s(N-p+1))+2\alpha_1\alpha_p(s(1)s(p)+s(N)s(N-P+1))\} \end{aligned} \qquad (A1.6)$$

(A1.6)

対数尤度$L(s|\Theta)$は、(A1.4)式の対数をとることにより、

$$L(s|\Theta) = \log(p(s|\Theta)) = \log G - N/2\cdot\log(2\pi\sigma^2) - F(s)/(2\sigma^2) \tag{A1.7}$$

$N \gg p$のとき、(A1.6)式の{}の中の値は、初めのp+1項にくらべて充分小さいので無視すると、

$$F(s) \fallingdotseq A_0\sum_{i=0}^{N-1}s(i)^2 + 2A_1\sum_{i=0}^{N-2}s(i)s(i+1) + \ldots + 2A_p\sum_{i=0}^{N-p-1}s(i)s(i+p) = N\sum_{i=-p}^{p}A_iR(i) \tag{A1.8}$$

ここで、$R(i)$は

$$R(i) = 1/N\cdot\sum_{j=0}^{N-i-1}s(j)s(j+i)\ ,\ i = 0,1,\ldots,N-1 \tag{A1.9}$$

で計算される$s(n)$の自己相関関数である。
(A1.7)式において、第1項は定数であり、第2項、第3項はともにNに比例して増大する。そこで、$N \gg p$の場合は第1項が無視でき、

$$L(s|\Theta) = -N/2\cdot\{\log 2\pi\sigma^2 + 1/\sigma^2\cdot\sum_{i=-p}^{p}A_iR(i)\} \tag{A1.10}$$

となる。
一方、(A1.3)式から

$$A_i = (\sigma/2\pi)^2\cdot\int\{\cos(i\omega)/T(\omega)\}d\omega\ ,\ i = 0,1,\ldots,p \tag{A1.11}$$

また、(A1.3)式の極z_iは、帯域幅B_i、共振周波数F_iを用いて

$$z_i = \exp(-\pi\tau B_i)\cdot\exp(j2\pi\tau F_i) \tag{A1.12}$$

と表わされる。ここで、τは信号の標本化周期、jは虚数単位である。系が安定であるためには、$B_i > 0$であり、すなわち$|z_i| < 1$となる。このとき

$$\int_{-\pi}^{\pi}|(1-z_iz^{-1})|^2d\omega = \int_{-\pi}^{\pi}(1-2z_i\cos\omega+z_i^2)d\omega = 0 \tag{A1.13}$$

であるので、(A1.3)式の対数をとり、$-\pi\sim\pi$まで積分した式は

$$\int_{-\pi}^{\pi}\log T(\omega)d\omega = 2\pi\log(\sigma^2/2\pi) \tag{A1.14}$$

となる。両辺に指数をとることで

$$\sigma^2 = 2\pi\exp\{1/2\pi\int_{-\pi}^{\pi}\log T(\omega)d\omega\} \tag{A1.15}$$

の関係を得る。
(A1.11),(A1.15)式を(A1.10)に代入すると

$$L(s|\Theta) = -N/2\cdot\{2\log 2\pi + 1/2\pi\cdot\int_{-\pi}^{\pi}\log T(\omega)d\omega$$

$$+\ 1/(2\pi)^2 \cdot \int_{-\pi}^{\pi} \left(\sum_{i=-(N-1)}^{N-1} R(i)\cos(i\omega)\right) d\omega / T(\omega)\} \quad \text{(A1.16)}$$

となる。ここで、

$$\begin{aligned} 1/2\pi \cdot \sum_{i=-(N-1)}^{N-1} R(i)\cos(i\omega) &= 1/(2\pi N) \cdot \left|\sum_{k=0}^{N-1} s(k)\exp(-jk\omega)\right|^2 \\ &= P(\omega) \end{aligned} \quad \text{(A1.17)}$$

なので、これを(A1.16)に代入して、結局、対数尤度は

$$L(s|\Theta) = -N/2 \cdot \{2\log 2\pi + 1/2\pi \cdot \int_{-\pi}^{\pi} (\log T(\omega) + P(\omega)/T(\omega))\, d\omega\} \quad \text{(A1.18)}$$

のように表わすことができる。

(2) 対数尤度によるスペクトルマッチング

上記のようにして求められた対数尤度の物理的意味は、次のように説明できる。
いま、$T(\omega)$が任意ならば、$L(s|\Theta)$の最大値L_{max}は、(A1.18)式で$T(\omega)=P(\omega)$とおくことにより、

$$\begin{aligned} L_{max} &= L(s|\Theta)|_{T()=P()} \\ &= -N/2 \cdot \{2\log 2\pi + 1/2\pi \cdot \int_{-\pi}^{\pi} (\log P(\omega) + 1)\, d\omega\} \end{aligned} \quad \text{(A1.19)}$$

となる。したがって、$L(s|\Theta)$とL_{max}との誤差をEとおくと、(A1.18),(A1.19)式より、

$$\begin{aligned} E &= L_{max} - L(s|\Theta) \\ &= N/4 \cdot \int_{-\pi}^{\pi} (\log(T(\omega)/P(\omega)) + P(\omega)/T(\omega) - 1)\, d\omega \end{aligned} \quad \text{(A1.20)}$$

のように書ける。
(A1.20)式右辺の被積分関数$(\log(T(\omega)/P(\omega)) + P(\omega)/T(\omega) - 1)$の値は、$T(\omega)$と$P(\omega)$の比が1のとき0となり、比が1から離れるにしたがって単調に増加する。したがって、Eは、$T(\omega)\equiv P(\omega)$の場合のみ0となり、それ以外は$P(\omega)$と$T(\omega)$の違いの程度に依存した正の値をとる。すなわち、EはDFTパワスペクトル$P(\omega)$を全極型パワスペクトル$T(\omega)$で表現する場合の誤差の評価尺度となっている。
そしてこの尺度Eは、$\log(x)<x$の性質より、$T(\omega)/P(\omega)$よりも、$P(\omega)/T(\omega)$をより大きく評価する。言い換えれば、$T(\omega)>P(\omega)$の部分よりも、$P(\omega)>T(\omega)$の部分による誤差を重視する尺度である。このことは、Eを最小にすることで求められる全極型パワスペクトル$T(\omega)$は、DFTパワスペクトル$P(\omega)$の谷の部分よりも山の部分に、より一致するような関数になることを示している。

付録Ａ－２　Töplitz行列を係数行列に持つ連立１次方程式の求解アルゴリズム

線形予測法において、$\{\alpha_i\}$を求めるために連立１次方程式を解く方法について述べる。

一般に、多元連立方程式を計算機上で解く場合には、ガウスの掃き出し法などを用いるが、本文(2.5),(2.15)式の方程式では、係数行列が対角軸について対称でかつ対角方向の成分が一定という特別な形をしている。

このような方程式では、以下に示すように、次数p=1から始めて逐次pを増しながら解いていく解法が確立されている。

(2.19)式で定義される自己相関係数r(i)を用いて(2.5)式または(2.15)式を書き改めると、r(0)=1より、

$$\begin{pmatrix} 1 & r(1) & \cdots & r(p-1) \\ r(1) & 1 & & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & & & \cdot \\ r(p-1) & & & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \cdot \\ \alpha_p \end{pmatrix} = - \begin{pmatrix} r(1) \\ r(2) \\ \cdot \\ \cdot \\ r(p) \end{pmatrix} \qquad \text{(A2.1)}$$

となる。

これを逐次的に解くために、j回目のループで得られたα_iを$\alpha_i^{(j)}$と書く。また、(A2.1)式を構成するパラメータ$r(i)$, α_iの他に$\sigma^2{}_{(j)}$, k_iを導入する。$\sigma^2{}_{(j)}$はj回目のループにおいて

$$\sigma^2{}_{(j)} = 1 + \sum_{i=1}^{j} \alpha_i^{(j)} r(i) \qquad \text{(A2.2)}$$

で定義される値で、j=pのときには

$$\sigma^2{}_{(p)} = \sum_{i=0}^{p} \alpha_i^{(p)} r(i) \ , \quad \alpha_0 = r(0) = 1 \qquad \text{(A2.3)}$$

となり、(2.11)式に(2.13)式を代入して得られる、最小化されたσ^2に等しい。

k_iは、j=iのときのα_iの値を保持するために用いられ、

$$k_i = \alpha_i^{(i)} \qquad \text{(A2.4)}$$

で定義される。

以下に求解のアルゴリズムを漸化的に得る手順を示す。

(1) p = 1 の場合

(A2.1)式は次の等式となる。

$$\alpha_1^{(1)} = -r(1) \qquad \text{(A2.5)}$$

したがって、(A2.2),(A2.4)式より

$$\begin{aligned} k_1 &= \alpha_1^{(1)} \\ &= -r(1) \end{aligned} \qquad \text{(A2.6)}$$

さらに(A2.2),(A2.6)式より

$$\sigma^2_{(1)} = 1 + \alpha_1^{(1)} r(1)$$
$$= 1 - k_1^2 \qquad (A2.7)$$

が得られる。
これらの値がアルゴリズムの初期値となる。

(2) p = 2 の場合
(A2.1)式は次の２元１次方程式となる。

$$\alpha_1^{(2)} + r(1)\alpha_2^{(2)} = -r(1)$$
$$r(1)\alpha_1^{(2)} + \alpha_2^{(2)} = -r(2) \qquad (A2.8)$$

これに、(A2.6)式を代入すると

$$\alpha_1^{(2)} - k_1\alpha_2^{(2)} = -r(1)$$
$$-k_1\alpha_1^{(2)} + \alpha_2^{(2)} = -r(2) \qquad (A2.9)$$

これらを$\alpha_2^{(2)}$,$\alpha_1^{(2)}$について解くと

$$\alpha_2^{(2)} = -1/(1-k_1^2)\cdot\{r(2) + k_1 r(1)\}$$
$$\alpha_1^{(2)} = -r(1) - r(1)\alpha_2^{(2)} \qquad (A2.10)$$

となり、さらに(A2.4),(A2.5),(A2.6),(A2.7)式より

$$\alpha_2^{(2)} = -1/\sigma^2_{(1)}\cdot\{r(2) + k_1 r(1)\}$$
$$= k_2$$
$$\alpha_1^{(2)} = \alpha_1^{(1)} + k_1\alpha_2^{(2)} \qquad (A2.11)$$

を得る。また(A2.2)式で求められる$\sigma^2_{(2)}$を、(A2.6),(A2.9),(A2.11)式によりk_iのみで表わすことで

$$\sigma^2_{(2)} = (1-k_2^2)(1-k_1^2)$$
$$= (1-k_2^2)\sigma^2_{(1)} \qquad (A2.12)$$

となる。

(3) p = 3 場合
(A2.1)式を解いて求められる$\alpha_3^{(3)}$は、これまでに得られた諸式より次のように表わせる。

$$\alpha_3^{(3)} = 1/\sigma^2_{(2)}$$
$$= k_3 \qquad (A2.13)$$

また、(A2.1)式から$\alpha_1^{(3)}$を消去することにより

$$\alpha_2^{(3)} = \alpha_2^{(2)} + \alpha_1^{(2)}\alpha_3^{(3)}$$
$$= \alpha_2^{(2)} + k_3\alpha_1^{(2)} \qquad (A2.14)$$

同様に、(A2.1)式から$\alpha_2^{(3)}$を消去することにより

$$\alpha_1^{(3)} = \alpha_1^{(2)} + \alpha_2^{(2)}\alpha_3^{(3)} = \alpha_1^{(2)} + k_3\alpha_2^{(2)} \tag{A2.15}$$

さらに、$\sigma^2{}_{(3)}$は

$$\sigma^2{}_{(3)} = (1-k_1{}^2)(1-k_2{}^2)(1-k_3{}^2) = (1-k_3{}^2)\sigma^2{}_{(2)} \tag{A2.16}$$

のようになる。

以上の結果をまとめると、

p=1の場合

$$\begin{aligned} k_1 &= -r(1) \\ \alpha_1^{(1)} &= k_1 \\ \sigma^2{}_{(1)} &= 1 - k_1{}^2 \end{aligned} \tag{A2.17}$$

p=2の場合

$$\begin{aligned} k_2 &= -\{r(2) + \alpha_1^{(1)}r(1)\}/\sigma^2{}_{(1)} \\ \alpha_2^{(2)} &= k_2 \\ \alpha_1^{(2)} &= \alpha_1^{(1)} + k_2\alpha_1^{(1)} \\ \sigma^2{}_{(2)} &= (1-k_2{}^2)\sigma^2{}_{(1)} \end{aligned} \tag{A2.18}$$

p=3の場合

$$\begin{aligned} k_3 &= -\{r(3) + \alpha_1^{(2)}r(2) + \alpha_2^{(2)}r(1)\}/\sigma^2{}_{(2)} \\ \alpha_3^{(3)} &= k_3 \\ \alpha_2^{(3)} &= \alpha_2^{(2)} + k_3\alpha_2^{(2)} \\ \alpha_1^{(2)} &= \alpha_1^{(2)} + k_3\alpha_2^{(2)} \\ \sigma^2{}_{(3)} &= (1-k_3{}^2)\sigma^2{}_{(2)} \end{aligned} \tag{A2.19}$$

そこで、(A2.17),(A2.18),(A2.19)式より、p=jの場合を推定すると次式のようになる。

p=jの場合

$$kj = -1/\sigma^2{}_{(j-1)} \cdot \{r(j) + \sum_{i=1}^{j-1}\alpha_i^{(j-1)}r(j-i)\} \tag{A2.20a}$$

$$\alpha_j^{(j)} = k_j \tag{A2.20b}$$

$$\alpha_i^{(j)} = \alpha_i^{(j-1)} + k_j\alpha_{j-i}^{(j-1)}, \quad 1 \leqq i \leqq j-1 \tag{A2.20c}$$

$$\sigma^2{}_{(j)} = (1-k_j{}^2)\sigma^2{}_{(j-1)} \tag{A2.20d}$$

初期値をp=1の場合((A2.17)式)とすると、(A2.20)式によって$\{\alpha_i\}$を計算する漸化式が与えられる。

次に、これらの漸化式を証明する。

(A2.20)式を証明するには、p=jのときの(A2.1)式すなわち

$$\begin{bmatrix} 1 & r(1) & \ldots & r(j-1) \\ r(1) & 1 & & . \\ . & & . & . \\ . & & . & . \\ r(j-1) & & & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_j \end{bmatrix} = - \begin{bmatrix} r(1) \\ r(2) \\ . \\ . \\ r(j) \end{bmatrix} \tag{A2.21}$$

が成り立つという仮定のもとで、(A2.20)式で計算される各パラメータを用いて、p=j+1のときにも(A2.1)式が成り立つことを示せばよい。

まず、(A2.20a)式右辺の分子を

$$r(j)+\sum_{i=1}^{j-1}\alpha_i^{(j-1)}r(j-i)=\sum_{i=0}^{j-1}\alpha_i^{(j-1)}r(j-i)=\gamma_{(j-1)} \tag{A2.22}$$

とおく。

p=j+1のとき、(A2.21),(A2.2)より

$$\gamma_{(j)}=r(j+1)+\sum_{i=1}^{j}\alpha_i^{(j)}r(j+1-i) \tag{A2.23}$$

$$\sigma^2_{(j)}=1+\sum_{i=1}^{j}\alpha_i^{(j)}r(i) \tag{A2.24}$$

と書ける。(A2.21)式の右辺を左辺に移項し、(A2.23)式、(A2.21)式、(A2.24)式の順に並べ、行列形式に書けば次の連立方程式が得られる。

$$\begin{bmatrix} 1 & r(1) & \ldots & r(j) & r(j+1) \\ r(1) & 1 & \ldots & r(j-1) & r(j) \\ \cdot & & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & \cdot \\ r(j) & r(j-1) & & 1 & r(1) \\ r(j+1) & r(j) & & r(1) & 1 \end{bmatrix}\begin{bmatrix} 1 \\ \alpha_1^{(j)} \\ \cdot \\ \cdot \\ \alpha_j^{(j)} \\ 0 \end{bmatrix}=-\begin{bmatrix} \sigma^2_{(j)} \\ 0 \\ \cdot \\ \cdot \\ 0 \\ \gamma_{(j)} \end{bmatrix} \tag{A2.25}$$

(A2.25)式は、係数行列の対称性を使って次のように書き改めることができる。

$$\begin{bmatrix} 1 & r(1) & \ldots & r(j) & r(j+1) \\ r(1) & 1 & \ldots & r(j-1) & r(j) \\ \cdot & & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & \cdot \\ r(j) & r(j-1) & & 1 & r(1) \\ r(j+1) & r(j) & & r(1) & 1 \end{bmatrix}\begin{bmatrix} 0 \\ \alpha_j^{(j)} \\ \cdot \\ \cdot \\ \alpha_1^{(j)} \\ 1 \end{bmatrix}=-\begin{bmatrix} \gamma_{(j)} \\ 0 \\ \cdot \\ \cdot \\ 0 \\ \sigma^2_{(j)} \end{bmatrix} \tag{A2.26}$$

(A2.26)式の両辺にkj+1を乗じて、(A2.25)式に加えれば

$$\begin{bmatrix} 1 & r(1) & \ldots & r(j) & r(j+1) \\ r(1) & 1 & \ldots & r(j-1) & r(j) \\ \cdot & & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & \cdot \\ r(j) & r(j-1) & & 1 & r(1) \\ r(j+1) & r(j) & & r(1) & 1 \end{bmatrix}\begin{bmatrix} 1-0 \\ \alpha_1^{(j)}-k_{j+1}\alpha_j^{(j)} \\ \cdot \\ \cdot \\ \alpha_j^{(j)}-k_{j+1}\alpha_1^{(j)} \\ 0-k_{j+1} \end{bmatrix}$$

$$=-\begin{bmatrix} \sigma^2_{(j)}-k_{j+1}\gamma_{(j)} \\ 0 \\ \cdot \\ \cdot \\ 0 \\ \gamma_{(j)}-k_{j+1}\sigma^2_{(j)} \end{bmatrix} \tag{A2.27}$$

を得る。

(A2.20a)式に(A2.23)式を代入すると

$$k_{j+1} = -\gamma_{(j)}/\sigma^2_{(j)}$$

すなわち

$$\gamma_{(j)} = -\sigma^2_{(j)}k_{j+1} \tag{A2.28}$$

を得る。この式と(A2.20d)式より、(A2.27)式の右辺は

$$\begin{bmatrix} \sigma^2_{(j)}-k_{j+1}\gamma_{(j)} \\ 0 \\ . \\ . \\ 0 \\ \gamma_{(j)}-k_{j+1}\sigma^2_{(j)} \end{bmatrix} = \begin{bmatrix} \sigma^2_{(j)}(1-k_{j+1}{}^2) \\ 0 \\ . \\ . \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \sigma^2_{(j+1)} \\ 0 \\ . \\ . \\ 0 \\ 0 \end{bmatrix} \tag{A2.29}$$

のようになる。さらに、(A2.27)式の左辺の列ベクトルに(A2.20c),(A2.20b)式を代入して(A2.27)式を書き改めると、

$$\begin{bmatrix} 1 & r(1) & \ldots & r(j) & r(j+1) \\ r(1) & 1 & \ldots & r(j-1) & r(j) \\ . & & . & . & . \\ . & & . & . & . \\ r(j) & r(j-1) & & 1 & r(1) \\ r(j+1) & r(j) & & r(1) & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \alpha_1{}^{(j+1)} \\ . \\ . \\ \alpha_j{}^{(j+1)} \\ \alpha_{j+1}{}^{(j+1)} \end{bmatrix} = - \begin{bmatrix} \sigma^2_{(j+1)} \\ 0 \\ . \\ . \\ 0 \\ 0 \end{bmatrix} \tag{A2.30}$$

が得られる。

(A2.30)式の方程式は、展開、整理することで次の2つの部分に分けられる。

$$1 + \Sigma\,\alpha_i{}^{(j+1)}r(i) = \sigma^2_{(j+1)} \tag{A2.31a}$$

$$\begin{bmatrix} 1 & r(1) & \ldots & r(j) \\ r(1) & 1 & & . \\ . & & . & . \\ . & & . & . \\ r(j) & & & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_{j+1} \end{bmatrix} = - \begin{bmatrix} r(1) \\ r(2) \\ . \\ . \\ r(j+1) \end{bmatrix} \tag{A2.31b}$$

(A2.31b)式は、p=j+1のときの(A2.1)式に他ならない。以上により、(A2.20)式の漸化式が証明された。

付録Ａ－３　Bairstow法による高次代数方程式の求解アルゴリズム

線形予測分析で声道伝達関数の極を求めるために、高次代数方程式を解く方法について述べる。

高次代数方程式の数値解法では、二分法、Newton-Raphson法がよく知られているが、本稿では、共役複素解を同時に求めることができるBairstow法を用いる。

Bairstow法は、方程式を構成する多項式の２次因子を求めて、次数を逐次下げていく方法であるから、方程式は偶数次であることが望ましい。実係数代数方程式では、方程式が奇数次であれば少なくとも１個の実解が存在するので、本稿では、求める方程式が奇数次の場合には、二分法により１個の実解を求め、次数を偶数次に下げた後Bairstow法を適用する。次数が２次まで下がれば、２次方程式の解の公式により、解析的に解を求めることができる。

以下に、二分法、Bairstow法の原理を示す。

(1) 二分法の原理

二分法は、少なくとも１つの実解が存在する方程式に対して適用できる。

まず、解が確実に存在する区間を求め、次にその区間を２等分した２つの区間について解の存在を調べ、解の含まれる方をさらに２等分して調べる。この作業を繰り返し行い、解の存在範囲を逐次狭めていくことによって、解が得られる。

実際の手順を以下に示す。

解くべき方程式を、

$$f(x) = 0 \tag{A3.1}$$

とおく。

いま、xのある区間$[x_l, x_r]$, $x_l \leqq x_r$を考えたとき、

$$p = f(x_l) \cdot f(x_r) \tag{A3.2}$$

で表わされるpの値が0以下であれば、$[x_l, x_r]$に解が存在する。そこで、適当な$[x_l, x_r]$から始めて、$p \leqq 0$となるまで$[x_l, x_r]$の区間を広げることで、解が存在する区間を求めることができる。

次に、上記のようにして求めたx_l, x_rを２等分する点x_c

$$x_c = (x_l + x_r) / 2 \tag{A3.3}$$

を計算し、解が$[x_l, x_c]$と$[x_c, x_r]$のどちらの区間に含まれるかを調べる。すなわち、

$$q = f(x_l) \cdot f(x_c) \tag{A3.4}$$

で表わされるqの値を調べ、$q \leqq 0$ならば$[x_l, x_c]$を、$q > 0$ならば$[x_c, x_r]$を解が存在する区間とみなす。このようにして求めた、解の存在する区間を改めて$[x_l, x_r]$とおき、(A3.3),(A3.4)式を用いて、さらに２等分された区間で、解の有無を調べる。

以上の操作を繰り返し、$x_r - x_l$の値がある評価値以下になれば、x_lまたはx_rを収束値として採用し、計算を打ち切る。

(2) Bairstow法の原理

Bairstow法は、実係数の２次以上の代数方程式に適用でき、実係数の２次因子を反復計算によって逐次求めていく方法である。
以下に手順を説明する。
解くべき代数方程式を

$$f(x) = x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_{n-1} x + a_n = 0 \qquad (A3.5)$$

とおく。
いま、p,qを任意の数として、f(x)を(x^2+px+q)で割ったときの商をR(x)、余りをSx+Tとすると、

$$f(x) = (x^2 + px + q)R(x) + Sx + T \qquad (A3.6)$$

と書ける。S,Tはどちらもp,qの関数であるから、これらをS(p,q),T(p,q)と表わすことにする。
S(p,q),T(p,q)を同時に0とするようなp,qが求まれば、それらはf(x)の２次因子を与える。そこで、そのようなp,qを$p+p_d$,$q+q_d$とすると、$S(p+p_d, q+q_d)$,$T(p+p_d, q+q_d)$はテイラー展開することにより、

$$\begin{aligned} S(p+p_d, q+q_d) &= S(p,q) + p_d \cdot \delta S(p,q)/\delta p + q_d \cdot \delta S(p,q)/\delta q \\ &\quad + p_d{}^2/2 \cdot \delta^2 S(p,q)/\delta^2 p + q_d{}^2/2 \cdot \delta^2 S(p,q)/\delta^2 q + \ldots \\ &= 0 \\ T(p+p_d, q+q_d) &= T(p,q) + p_d \cdot \delta T(p,q)/\delta p + q_d \cdot \delta T(p,q)/\delta q \\ &\quad + p_d{}^2/2 \cdot \delta^2 T(p,q)/\delta^2 p + q_d{}^2/2 \cdot \delta^2 T(p,q)/\delta^2 q + \ldots \\ &= 0 \end{aligned} \qquad (A3.7)$$

と書ける。(A3.7)式を一次近似の項までで打ち切ると、

$$\begin{pmatrix} S \\ T \end{pmatrix} + \begin{pmatrix} \delta S/\delta p & \delta S/\delta q \\ \delta T/\delta p & \delta T/\delta q \end{pmatrix} \begin{pmatrix} p_d \\ q_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad (A3.8)$$

のようにp_d,q_dに関する連立方程式が得られる。ここで、S,TはS(p,q),T(p,q)を表わし、以後も、特に区別する必要がなければS(p,q),T(p,q)を単にS,Tと書く。
(A3.8)式の解は、

$$\begin{pmatrix} p_d \\ q_d \end{pmatrix} = -1/D \cdot \begin{pmatrix} \delta T/\delta q & -\delta S/\delta q \\ -\delta T/\delta p & \delta S/\delta p \end{pmatrix} \begin{pmatrix} S \\ T \end{pmatrix} \qquad (A3.9)$$

$$D = \frac{\delta S}{\delta p} \cdot \frac{\delta T}{\delta q} - \frac{\delta S}{\delta q} \cdot \frac{\delta T}{\delta p}$$

で与えられる。すなわち、S,Tおよびその偏微分係数が求まれば、f(x)の２次因子を近似する係数が計算できる。
S,Tおよびその偏微分係数の計算は、次のようにして行なう。
R(x)は(n-2)次の多項式なので、これを

$$R(x) = x^{n-2} + b_1 x^{n-3} + \ldots + b_{n-3} x + b_{n-2} \qquad (A3.10)$$

と書けば、(A3.10)式を(A3.6)式に代入して展開し、(A3.5)式と係数を比較することにより、次の漸化式が得られる。

$$
\begin{aligned}
b_{-1} &= 0 \\
b_0 &= 1 \\
b_k &= a_k - pb_{k-1} - qb_{k-2}\ ,\ k = 1,2,\ldots,n-2
\end{aligned}
\tag{A3.11}
$$

同様に、S,Tは、

$$
\begin{aligned}
S &= b_{n-1} \\
T &= a_n - qb_{n-2} \\
&= b_n + pb_{n-1}
\end{aligned}
\tag{A3.12}
$$

で与えられる。ここで、b_n, b_{n-1}は(A3.11)式のb_kを$k=b_n$, b_{n-1}について求めたものである。

次に、S,Tの偏微分係数を求めるために次式で表わされるc_kを、導入する。

$$c_k \equiv -\delta b_{k+1}/\delta p \tag{A3.13}$$

c_kを求める漸化式は、(A3.11)式をpで偏微分することにより次のように求められる。

$$
\begin{aligned}
c_{-1} &= 0 \\
c_0 &= 1 \\
c_k &= b_k - pc_{k-1} - qc_{k-2}\ ,\ k = 1,2,\ldots,n-3
\end{aligned}
\tag{A3.14}
$$

また、(A3.11)式のb_1,b_2,b_kをqで偏微分すると、

$$
\begin{aligned}
\delta b_1/\delta q &= 0 \\
\delta b_2/\delta q &= -1 \\
\delta b_k/\delta q &= -b_{k-2} - p\delta b_{k-1}/\delta q - q\delta b_{k-2}/\delta q
\end{aligned}
\tag{A3.15}
$$

となる。(A3.14)と(A3.15)を比較することにより、両式を同時に満足するc_kが、

$$c_k = -\delta b_{k+2}/\delta q \tag{A3.16}$$

のように得られる。

結局、S,Tの偏微分係数は、(A3.12)の両辺をp,qで偏微分し、(A3.13),(A3.16)式を代入することで、次のようにc_kを用いて表わすことができる。

$$
\begin{aligned}
\delta S/\delta p &= \delta b_{n-1}/\delta p \\
&= -c_{n-2} \\
\delta T/\delta p &= -q\delta b_{n-2}/\delta p \\
&= qc_{n-3} \\
\delta S/\delta q &= \delta b_{n-1}/\delta q \\
&= -c_{n-3} \\
\delta T/\delta q &= \delta b_n/\delta q + p\delta b_{n-1}/\delta q \\
&= -c_{n-2} - pc_{n-3}
\end{aligned}
\tag{A3.17}
$$

以上の結果(A3.11),(A3.12),(A3.14),(A3.17)式を(A3.9)式に代入すると、p,qの1次近似補正値p_d,q_dが次式のように求められる。

$$p_d = (b_{n-1}c_{n-2} - b_n c_{n-3}) / D$$
$$q_d = (b_n c_{n-2} + b_{n-1}(b_{n-1} - c_{n-1})) / D \qquad (A3.18)$$
$$D = c^2_{n-2} + c_{n-3}(b_{n-1} - c_{n-1})$$

そこで、改めて$p+p_d$,$q+q_d$をp,qとして解が収束するまで同じ計算を繰り返す。このような反復計算によって多項式の２次因子を求めるのが、Bairstow法である。

収束の判定には、適当な収束判定基準値εと、S,T,p_d,q_dの値を用い、３つの条件

$$|S| < \varepsilon |a_n|$$
$$|T| < \varepsilon |a_n| \qquad (A3.19)$$
$$\frac{|p_d| + |q_d|}{|p| + |q|} < \varepsilon$$

が同時に満たされれば、収束したものとみなす。

以上の手順により１組の解が得られるので、方程式の次数をn次から(n-2)次に下げて、次の２次因子を求める。すなわち、(A3.11)式によって求められたb_kを改めてa_kとおいた(n-2)次方程式で、(A3.11),(A3.12),(A3.14),(A3.18)式の計算と(A3.19)式の判定を繰り返す。

付録A－4　全極型パワスペクトルのローカルピーク探索アルゴリズム

全極形パワスペクトル$T(\omega)$が与えられたときに、フォルマント周波数およびフォルマント帯域幅の抽出値$\tilde{F}_i$,$\tilde{B}_i$を求める方法について述べる。

まず前提として、このアルゴリズムでは、入力としてパワスペクトルをデシベル単位で表わし、周波数軸上で標本化したものを仮定している(図A4.1)。これを$t(n)$とすると、

$$t(n) = 10\log T(\omega) \tag{A4.1}$$

ここで角周波数ωが0からπまでN等分に標本化されているとすると、nとωには、

$$n \cdot \pi / N = \omega \tag{A4.2}$$

の関係がある。

ピーク抽出の方法は、２つの段階に分けられる。まず、$t(n)$を$0 \leqq n < N$の範囲で探索し、周波数軸での離散値としてピークを求める段階と、次に、求めた離散値に放物線内挿を施し、より正確なピークを抽出する段階である。以下に、それぞれの段階について述べる。

(1) 離散値でのピーク探索

ここで求めるピークは次のように定義される。

ある$t(n)$, $2 \leqq n < N-2$が、$t(n)$に隣接する４点$t(n-1)$, $t(n-2)$, $t(n+1)$, $t(n+2)$のいずれに比べても大きい場合、$t(n)$は離散値での１つのピークに相当する。

この定義のもとに求めた$t(n)$を$t(n_m)$, $m=1,2,...$とおく。

(2) 放物線内挿

内挿には、前項で求めた$t(n_m)$を中心とする３点$t(n_m-1)$, $t(n_m)$, $t(n_m+1)$を用いる。

周波数軸をn、パワー軸をpとしたn-p平面で、$t(n_m-1)$, $t(n_m)$, $t(n_m+1)$を表わした場合の座標をそれぞれ(n_{-1}, p_{-1}), (n_0, p_0), (n_1, p_1)とすると、

$$\begin{aligned} n_{-1} &= n_m - 1 \\ n_0 &= n_m \\ n_1 &= n_m + 1 \end{aligned} \tag{A4.3a}$$

$$\begin{aligned} p_{-1} &= t(n_m-1) \\ p_0 &= t(n_m) \\ p_1 &= t(n_m+1) \end{aligned} \tag{A4.3b}$$

の関係がある。

ここでは、n-p平面での内挿されたピークの座標(n_{peak}, p_{peak})と-3dBの帯域幅bwとを求める(図A4.2(a))。

求める内挿放物線は、

$$p = an^2 + b'n + c' \tag{A4.4}$$

のように表わされる。放物線のパラメータa, b', c'はn_i, p_i $(i=-1,0,1)$によって直接計算することもできるが、ここではプログラムのソースコードを簡略化するため、でき

るだけ簡単な式でパラメータを表現することを考える。
まず、$t(n_m)$の座標(n_0, p_0)が原点に来るように、次の変数変換によって座標軸を平行移動する(図A4.2(b))。

$$\begin{aligned} x &= n - n_0 \\ y &= p - p_0 \end{aligned} \tag{A4.5}$$

この時の内挿放物線の式を

$$y = ax^2 + bx + c \tag{A4.6}$$

とおく。次に、内挿に用いる3点のx-y平面での座標を$(x_{-1}, y_{-1}), (x_0, y_0), (x_1, y_1)$とおき、(A4.6)式に代入することで、a,b,cに関する連立1次方程式が得られる。

$$\begin{aligned} y_{-1} &= ax_{-1}^2 + bx_{-1} + c \\ y_0 &= ax_0^2 + bx_0 + c \\ y_1 &= ax_1^2 + bx_1 + c \end{aligned} \tag{A4.7}$$

ここで、(A4.3a),(A4.5)式より、

$$\begin{aligned} x_{-1} &= n_{-1} - n_0 = n_m - 1 - n_m \\ &= -1 \\ x_0 &= n_0 - n_0 \\ &= 0 \\ x_1 &= n_1 - n_0 = n_m + 1 - n_m \\ &= 1 \end{aligned} \tag{A4.8a}$$

$$\begin{aligned} y_{-1} &= p_{-1} - p_0 \\ y_0 &= p_0 - p_0 \\ &= 0 \\ y_1 &= p_1 - p_0 \end{aligned} \tag{A4.8b}$$

また、$p_{-1} < p_0, p_1 < p_0$より、

$$\begin{aligned} y_{-1} &< 0 \\ y_1 &< 0 \end{aligned} \tag{A4.8c}$$

(A4.8a),(A4.8b)式を(A4.7)式に代入し、a,b,cについて解くことで、

$$\begin{aligned} a &= (y_1 + y_{-1}) / 2 \\ b &= (y_1 - y_{-1}) / 2 \\ c &= 0 \end{aligned} \tag{A4.9}$$

が得られる。
これらのパラメータa,b,cを用いて、内挿されたピークの座標、帯域幅は次のように、求めることができる。
まず、ピークのx座標x_{peak}は、(A4.6)式の右辺をxで微分して0とおくことにより、

$$x_{peak} = -b / 2a \tag{A4.10}$$

のように得られる。また、y座標y_{peak}は、(A4.10)式を(A4.6)式に代入して、

$$y_{peak} = -b^2/4a \tag{A4.11}$$

となる。

帯域幅bwは、y座標がピークより3dB低い2点のx座標を、$x_{b+}, x_{b-}, (x_{b+} > x_{b-})$とすれば、

$$bw = x_{b+} - x_{b-} \tag{A4.12}$$

によって計算できる。x_{b+}, x_{b-}は(A4.6)式の左辺に$(y_{peak}-3)$を代入し、xについて解くことで次のように求められる。

$$\begin{aligned} x_{b+} &= -b/2a + \sqrt{(-3/a)} \\ x_{b-} &= -b/2a - \sqrt{(-3/a)} \end{aligned} \tag{A4.13}$$

ここで、-3/aの正値性は、(A4.8c),(A4.9)式より導かれるa<0の関係により、保証される。

結局、帯域幅bwは(A4.12),(A4.13)式より、

$$bw = 2\sqrt{(-3/a)} \tag{A4.14}$$

のようにaのみに依存した形で得られる。

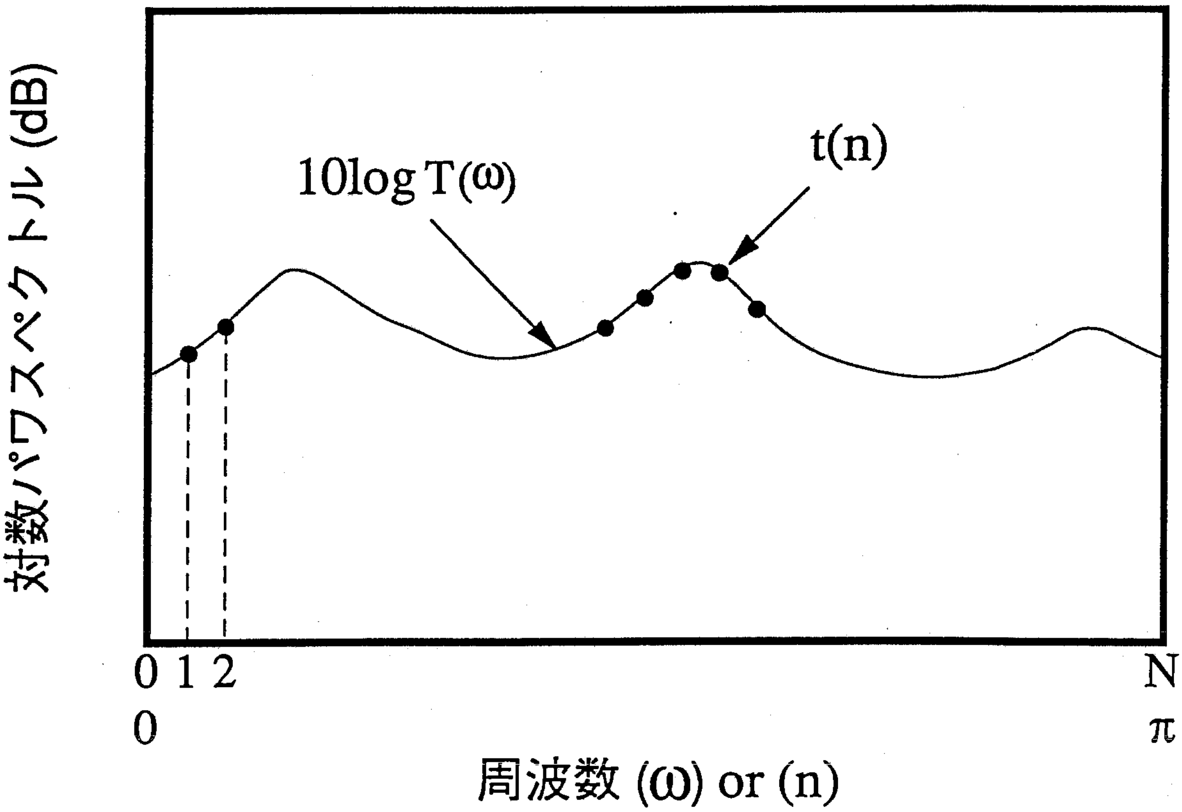
最後に(A4.5)式の関係により、(x,y)から(n,p)に変数変換することで、内挿ピークのn-p平面での座標(n_{peak}, p_{peak})が得られる。

$$\begin{aligned} n_{peak} &= x_{peak} + n_0 \\ p_{peak} &= y_{peak} + p_0 \end{aligned} \tag{A4.15}$$

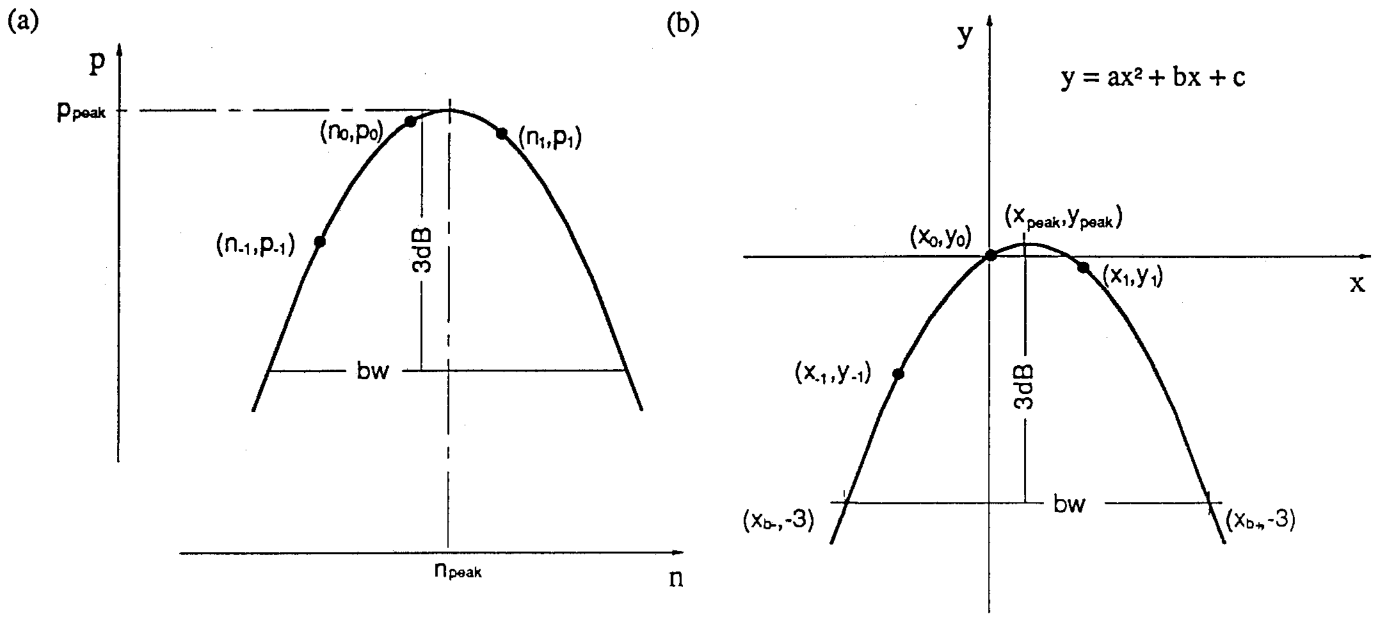
フォルマント周波数$\tilde{F}$、フォルマント帯域幅$\tilde{B}$をHzの単位で得るには、(A4.14)式のbw、(A4.15)式のn_{peak}の値に、Hz単位で周波数軸の標本間隔を乗じる。すなわち、音声波形のサンプリング周波数をf_sとすれば、

$$\begin{aligned} \tilde{F} &= n_{peak} \cdot f_s/2N \\ \tilde{B} &= bw \cdot f_s/2N \end{aligned} \tag{A4.16}$$

によって計算できる。



図A4.1 対数パワスペクトルとその標本値



図A4.2　放物線内挿での変数変換

付録B ソースリスト

1. pre_emphasis.c　　プリエンファシス
2. hanwdw.c　　窓関数（ハニング窓）
3. sigcor.c　　自己相関係数
4. coralf.c　　線形予測係数（α-パラメータ）
5. alfpol.c　　極推定
6. alfpsd.c　　全極型パワスペクトル
7. local_max.c　　ピーク抽出

```
/************************************************************************\
*       "pre_emphasis" emphasizes higher region of signal spectrum      *
*       by 1st-order differencing (1 - mu * 1/z)                        *
*-----------------------------------------------------------------------*
*SYNTAX                                                                 *
*   int pre_emphasis( is, os, n, mu )                                   *
*       short   *is;    :input/signal sample sequence                   *
*       double  *os;    :output/emphasized signal sequence              *
*       int     n;      :input/length of sample sequence                *
*       double  mu;     :input/coefficient of 1/z                       *
*-----------------------------------------------------------------------*
*                               coded by H.Kato         15 Dec. 1988    *
*                               revised                 09 Nov. 1989    *
\************************************************************************/

pre_emphasis( is, os, n, mu )
   short        is[];   /* input signal sample sequence */
   double       os[];   /* emphasized signal sequence */
   int          n;      /* length of sample sequence */
   double       mu;     /* coefficient of 1/z */
   {
        int     i;

        /* 'mu' should be more than 0 and less than 2 */
        if (mu < 0.0 || mu > 2.0) {
                return( 0 );
        }
        else {
                os[0] = (double)is[0];
                for (i = 1 ; i < n ; i++) {
                        os[i] = (double)is[i] - mu * (double)is[i-1];
                }
                return( 1 );
        }
   }
```

```
/*********************************************************************\
*       "hanwdw" generates weighting values of Hanning window         *
*---------------------------------------------------------------------*
*SYNTAX                                                               *
*   int hanwdw( window, n )                                           *
*       double  *window;        :output/weighting values              *
*       int     n;              :input/window length                  *
*---------------------------------------------------------------------*
*                                                                     *
*                    original coded by M.Akagi                        *
*                       revised by H.Kato          16 Nov. 1989       *
\*********************************************************************/

# include       <math.h>

hanwdw( window, n )
   double       window[];       /* weighting values of window function */
   int          n;              /* window length */
   {
        double  d;              /* unit argument step */
        int     i;
        static double
                PI = 3.14159265358979323846264338327950288419716939;

        d = 2.0 * PI / (double)n;
        for (i = 0 ; i < n ; i++) {
                window[i] = 0.5 - 0.5 * cos( d * (double)i );
        }
        return( 1 );
   }
```

```
/**********************************************************************\
*       "sigcor" calculates auto-correlation coefficients from signal    *
*       samples                                                          *
*-----------------------------------------------------------------------*
*SYNTAX                                                                  *
*   int sigcor( n, x, pow, r, p )                                        *
*       int     n;      :input/length of sample sequence                 *
*       double  *x;     :input/signal sample sequence                    *
*       double  *pow;   :output/mean power of input signal               *
*       double  *r;     :output/normalized auto-correlation coefficients*
*       int     p;      :input/number of required auto-correlation       *
*                        coefficients                                    *
*FUNCTIONS                                                               *
*       double  dot();                                                   *
*-----------------------------------------------------------------------*
*                                                                        *
*                       original coded by M.Akagi                        *
*                        revised by H.Kato            16 Nov. 1989       *
\**********************************************************************/

sigcor( n, x, pow, r, p )
   int          n;      /* length of signal samples */
   double       x[];    /* signal sample sequence */
   double       *pow;   /* mean power of input signal */
   double       r[];    /* normalized auto-correlation coefficients */
   int          p;      /* number of required auto-correlation coefficients */
   {
        double  sq_sum; /* square sum of input signal */
        double  dot();  /* convolution */
        int     i, j;

        /* calculate mean square */
        if (n <= 0) {
                *pow = 0.0;
        }
        else {
                sq_sum = dot( x, x, n );
                *pow = sq_sum / (double)n;
                if (p <= 0) {
                        return( 0 );
                }
        }

        /* calculate auto-correlation coefficients */
        if (*pow <= 0.0) {
                for ( i = 0 ; i < p ; i++ ) {
                        r[i] = 0.0;
                }
                return( 0 );
        }
        else if (p >= n) {
                if (n-1 > 0) {
                        for (i = 0 ; i <= n-1 ; i++) {
                                j = n - i;
                                r[i] = dot( x, &x[i], j ) / sq_sum;
                        }
                }
                for (i = n ; i <= p ; i++) {
                        r[i] = 0.0;
                }
                return( 0 );
        }
        else {
                for (i = 0 ; i <= p ; i++) {
                        j = n - i;
                        r[i] = dot( x, x+i, j ) / sq_sum;
                }
                return( 1 );
        }
   }
```

```
/***********************************************************************\
*       "dot" returns convolution of input data sequences              *
*       dot( x, y, n ) = x(0)*y(0) +.....+ x(n-1)*y(n-1)               *
*----------------------------------------------------------------------*
*SYNTAX                                                                *
*   double dot( x, y, n )                                              *
*       double  *x;     :input/data sequence                           *
*       double  *y;     :input/data sequence                           *
*       int     n;      :input/number of data in one sequence          *
*----------------------------------------------------------------------*
*                                                                      *
*                   original coded by M.Akagi                          *
*                       revised by H.Kato           16 Nov. 1989       *
\***********************************************************************/

double dot( x, y, n )
   double       x[];    /* one of two input data sequences */
   double       y[];    /* another input data sequence */
   int          n;      /* number of data in a sequence */
   {
        double d;       /* return value */
        int i;

        d = 0.0;
        for (i = 0 ; i < n ; i++) {
                d = d + *(x+i) * *(y+i);
        }
        return( d );
   }
```

```
/***************************************************************************\
*       "coralf" calculates alpha parameters from auto-correlation           *
*       coefficients                                                         *
*---------------------------------------------------------------------------*
*SYNTAX                                                                      *
*   int coralf( p, r, alf, sigma2 )                                          *
*       int     p;      :input/order of analysis                             *
*       double  *r;     :input/normalized auto-correlation coefficients      *
*       double  *alf;   :output/alpha parameters                             *
*       double  *sigma2;:output/normalized residual power                    *
*---------------------------------------------------------------------------*
*                                                                            *
*                       original coded by M.Akagi .                          *
*                          revised by H.Kato            16 Nov. 1989         *
\***************************************************************************/

coralf( p, r, alf, sigma2 )
   int          p;              /* order of analysis */
   double       r[];            /* normalized auto-correlation coefficients */
   double       alf[];          /* alpha parameters */
   double       *sigma2;        /* normalized residual power */
   {
        double  k,              /* PARCOR coefficient */
                last_alf;       /* alf[] in the latest roop */
        int     i, j;

        if (p <= 1) {
                *sigma2 = 1.0;
                return( 0 );
        }

        /* initialize */
        alf[0]  = 1.0;          /* alf[0] is implicitly 1 */
        k       = -r[1];
        alf[1]  =  k;
        *sigma2 = 1.0 - k * k;

        /* roop for alf[2] --- alf[p] */
        for (j = 2 ; j <= p ; j++) {
                k = r[j];
                for (i = 1 ; i < j ; i++) {
                        k = k + alf[i] * r[j-i];
                }
                k = -k / *sigma2;
                alf[j] = k;

                for (i = 1 ; i*2 <= j ; i++) {
                        last_alf = alf[i];
                        alf[i] = last_alf + k * alf[j-i];
                        if (i*2 >= j) {
                                break;
                        }
                        alf[j-i] = alf[j-i] + k * last_alf;
                }
                *sigma2 = (1.0 - k * k) * *sigma2;
        }
        return( 1 );
   }
```

```c
/***************************************************************************\
*       "alfpol" solves algebraic equation and obtain pole frequencies     *
*       and bandwidths                                                     *
*--------------------------------------------------------------------------*
*SYNTAX                                                                    *
*   int alfpol( p, alf, freq, bw, fs, eps )                                *
*       int     p:      :input/number of poles ; upper limit is 40         *
*       double  *alf:   :input/alpha parameter                             *
*                        alf[] are not changed                             *
*       double  *freq:  :output/pole frequency                             *
*                        0 <= freq[] < Nyquist frequency                   *
*       double  *bw;    :output/pole bandwidth                             *
*                        freq[i], bw[i] ( i = 0,....,p-1 )                 *
*       double  fs;     :input/sampling frequency                          *
*       double  eps;    :input/convergence parameter for Bairstow method   *
*FUNCTIONS                                                                 *
*       int     bairst();                                                  *
*       double  dichotomy();                                               *
*       double  f();                                                       *
*       int     bairstpq();                                                *
*       int     quicksort();                                               *
*       int     findpivot();                                               *
*       int     partition();                                               *
*--------------------------------------------------------------------------*
*                       coded by K.Shikano,             sept. 30,1984      *
*                       revised by Segot,               july   8,1987      *
*                       revised by H.Kato               January 6,1989     *
*                                                       3 Dec. 1989        *
\***************************************************************************/

# include       <stdio.h>
# include       <math.h>
# define        PI      3.141592653590
# define        MAXPOLE 40
# define        MAXITT  32767

alfpol( p, alf, freq, bw, fs, eps )
   int          p;              /* number of poles */
   double       alf[];          /* alpha parameter */
   double       freq[];         /* pole frequency (Hz) */
   double       bw[];           /* pole bandwidth (Hz) */
   double       fs;             /* sampling frequency (Hz) */
   double       eps;            /* convergence estimation value for Bairstow */
   {
        double  x, y, ff;
        int     i;

        if (p <= 0) {
                fprintf( stderr,
                "\nfrom alfpol: !! POLE NO KAZU GA MAINASU YANKe !!!\n");
                return( 0 );
        }
        if (p > 40) {
                fprintf( stderr,
                "\nfrom alfpol: !! POLE NO KAZU GA OOSUGI-MASSe !!!\n");
                return( 0 );
        }

        if (bairst( p, alf, bw, freq, eps ) == 0) {
                fprintf( stderr, "\nfrom alfpol: !! Fail Bairstow !!!\n");
                return( 0 );
        }

        /* the real parts of the roots are stored in bw[] and the imaginary ones
         * in freq[]
         */
        else {
                for (i = 0 ; i < p ; ++i) {
                        x  = *(bw+i);
                        y  = *(freq+i);
                        ff = x*x + y*y;

                        /* the pole bandwidth is
                         * - sampling_frequency / PI * ln( module( root ) )
                         */
                        *(bw+i) = (ff<=0)? 0:(-fs * log(ff) / (PI * 2.0));

                        /* the pole frequency is
                         * sampling_frequency / ( 2 * PI ) * argument( root )
                         */
                        *(freq+i) = fs / ( 2 * PI ) * fabs( atan2( y, x ) );
                        }

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

```
                        if (*(bw+i) <= 0.0) {
                                fprintf( stderr, "\nfrom alfpol:Unstability of\
                                the LPC !!! ( root out of a unit circle )\n");
                                return( 0 );
                        }
                /* return 0 in case of unstability of the LPC */
                }

                /* reordering according to the increasing frequencies */
                quicksort( 0, p-1, freq, bw );

                return( 1 ); /* success */
        }
    }

/********************************\
*                                *
*       root extraction          *
*          routines              *
*                                *
\********************************/

bairst( n, a, rp, ip, eps )
/*   solve algebraic equation by Bairstow method
 *
 * inputs
 *   n   --- order of the algebraic equation
 *   a[] --- a[0]*x**n + a[1]*x**(n-1) + --- + a[n]=0
 *           a[0] = 1.0 implicitly !
 *           a[] are not broken during the process !
 *  eps  --- accuracy of the root estimation
 * outputs
 *  rp[] --- real parts of the estimated roots
 *  ip[] --- imaginary parts of the estimated roots
 *
 */
   int          n;
   double       *a, *rp, *ip, eps;
   {
        int     i;
        double  p[1], q[1];             /* initial guesses for the Bairstow */
        double  a_bis[MAXPOLE + 1];     /* copy of a[]. a_bis[] may be broken */

        /* we have to try several values for p and q untill we find a couple for which
         * the Bairstow iteration algorithm converges
         */
        /* the 1st try */
        p[0] = q[0] = 0.00001;
        for (i = 0 ; i <= n ; i++)
                a_bis[i] = *(a+i);
        if (bairstpq( n, a_bis, rp, ip, eps, p, q ) != 0) return( 1 );
        else {
                /* the 2nd try */
                p[0] = q[0] = 1.0;
                for (i = 0 ; i < n ; i++)
                        a_bis[i] = *(a+i);
                if (bairstpq( n, a_bis, rp, ip, eps, p, q ) != 0) return( 1 );
                else {
                        /* the 3rd try */
                        p[0] = 0.00001;
                        q[0] = 1.0;
                        for (i = 0 ; i < n ; i++)
                                a_bis[i] = *(a+i);
                        if (bairstpq( n, a_bis, rp, ip, eps, p, q ) != 0) return( 1 );
                        else {
                                /* the 4th try */
                                q[0] = 0.00001;
                                p[0] = 1.0;
                                for (i = 0 ; i < n  ; i++)
                                        a_bis[i] = *(a+i);
                                return( bairstpq( n, a_bis, rp, ip, eps, p, q ) != 0 );
                        }
                }
        }
   }

double dichotomy( a, n, eps )
/* this function returns a real root of the polynomial
 * a[0]*x**n + a[1]*x**(n-1) + --- + a[n] = 0
 * by dichotomy method
 */
   double       *a;     /* coefficients of the polynomial */
   int          n;      /* order of the polynomial */
```

```
	double		eps;	/* convergence examiner */
	{
		double	f();
		double	xl, xc, xr;
				/* candidates of a root */
				/*  xl <= xc <= xr    */

		xl = xr = 0;
		while (f( a, n, xl ) * f( a, n, xr ) > 0) {
			xl -= 1;
			xr += 1;
		}
		while ((xr - xl) > eps) {
			xc = (xl + xr) / 2.0;
			if (f( a, n, xl ) * f( a, n, xc ) > 0) {
				xl = xc;
			}
			else {
				xr = xc;
			}
		}
	return( xl );
	}

double f( a, n, x )
/* this function returns the value of the polynomial in the point x
 * it is used when searching a real root of the polynomial by dichotomy
 * a[0] the coefficient of x**n is implicitly 1
 */
	double		*a;	/* coefficients of the polynomial */
	int		n;	/* order of the polynomial */
	double		x;	/* a value to be substituted */
	{
		double	y;	/* substituted polynomial value */
		int	i;

		y = 1.0;
		for (i = 1 ; i <= n ; i++)
			y = y*x + *(a+i);
		return( y );
	}

bairstpq( n, a, rp, ip, eps, p, q )
/*   solve algebraic equation by Bairstow method
 *
 *  if the algorithm does not converge, the routine returns 0
 *  else, it returns 1
 *
 * inputs
 *   n   --- order of the algebraic equation
 *   a[] --- a[0]*x**n + a[1]*x**(n-1) + --- + a[n]=0
 *           a[0] = 1.0 implicitly !
 *           a[] are broken during the process !
 *  eps  --- estimation accuracy
 *  p,q  --- initial guesses of the coefficients of the
 *           normalized factoring 2nd order polynomial
 * outputs
 *  rp[] --- real part of the roots(solution)
 *  ip[] --- imaginary part of the roots (solution)
 *
 */
	int		n;
	double		*a, *rp, *ip, eps, *p, *q;
	{
		int	np, i, j, k;
		double	bk, bk_1, bk_2;		/* temporary polynomial coeffients */
		double	ck, ck_1, ck_2, ck_3;	/* temporary partial differential coeffcients */

		double	pd, qd;			/* increments of coefficients of a estimated
						 * factoring 2nd order polynomial
						 */
		double	S, T;			/* coefficients of residual in estimation */
		double	x;			/* a real root found by dichotomy */
		double	D, d, w;

		*a = 1.0;

		if (n%2 == 1) {
		/* if the polynomial is odd, then there must be a real root */

			x = dichotomy( a, n, eps );
			/* the dichotomy has the advantage over the usual Newton-Raphson,
			 * to work everytime without overflows on the VAX or other problems,
			 * even though it should be somewhat less efficient
```

```
                 */

                /* now we factor the polynomial, since we found one of its roots */
                *(a+1) += x * (*a);
                if (n > 0) {
                        for (i = 2 ; i < n ; i++)
                                *(a+i) += x * *(a+i-1);
                }
                *(rp+n-1) = x;
                *(ip+n-1) = 0.0;
                np = n-1;
        }
        else {
                np = n;
        }
        /* now we get a real polynomial of even order, hence, its roots can be found by
         * complex conjugate pairs, and we try to find a suitable factorization by
         * normalized 2nd order real polynomials whom coefficients are p and q; for
         * that purpose, an iterative algoritm is undertaken
         */
        if (np >= 4) {
        /* if there are at least two factoring 2nd order polynomials */
                for (i = np ; i >= 4 ; i-=2) {
                        for(j = 0 ; j < MAXITT ; j++) {
                                ck_2 = bk_2 = 0.0;
                                ck_1 = bk_1 = *a;

                                for (k = 0 ; k < i ; k++) {
                                        bk = *(a+k+1) - *p * bk_1 - *q * bk_2;

                                        /* check a possible overflow (on a VAX) and
                                         * prevent it
                                         */
                                        if (fabs(bk) > 1e33) {
                                                return( 0 );
                                        }
                                        ck = bk - *p * ck_1 - *q * ck_2;
                                        if (fabs(ck) > 1e33) {
                                                return( 0 );
                                        }
                                        if (k == (i-4)) ck_3=ck;
                                        if (k != (i-1)) {
                                                bk_2 = bk_1;
                                                bk_1 = bk;
                                                ck_2 = ck_1;
                                                ck_1 = ck;
                                        }
                                }

                                D  = ck_2 * ck_2 + ck_3 * (bk_1 - ck_1);
                                if (fabs( D ) < 0.1*eps) {
                                    pd = 1;
                                    qd = 0;
                                }
                                else {
                                    pd = ( ((bk_1 * ck_2) - (bk * ck_3)) / D );
                                    qd = ( ((bk * ck_2) + bk_1 * (bk_1 - ck_2)) / D );
                                }
                                *p += pd;
                                *q += qd;

                                S = bk_1;
                                T = bk + *p * bk_1;

                                /* examine if the Bairstow has converge or not */
                                if (fabs( S ) < fabs( eps * a[i]) &&
                                    fabs( T ) < fabs( eps * a[i]) &&
                                    ( (fabs( pd ) + fabs( qd ))
                                     / (fabs( *p ) + fabs( *q )) ) < fabs( eps )) {
                                        break;
                                }
                        }

                        /* when one 2nd factor has been found, renew polynomial
                         * coefficients
                         */
                        bk_1 = *a;
                        bk_2 = 0.0;
                        for (j = 2 ; j < i ; j++) {
                                bk = *(a+j-1) - *p * bk_1 - *q * bk_2;
                                *(a+j-1) = bk;
                                bk_2 = bk_1;
                                bk_1 = bk;
                        }
```

```
				/* save p and q */
				*(rp+i-2) = *p;
				*(rp+i-1) = *q;
			}
		}
		/* save last pair of polynomial coefficients */
		*rp     = *(a+1) / *a;
		*(rp+1) = *(a+2) / *a;

		/* now calculate roots from cofficients and store them */
		++rp;
		++ip;
		for (i = 1 ; i <= np ; i+=2) {
			w = -0.5 * *(rp-1);
			d = w*w - *rp;
			if (d >= 0.0) {
			/* the case real roots */

				d = sqrt( d ) + fabs( w );
				w = *rp;
				*(ip-1) = 0.0;
				*ip = 0.0;
				if (*(rp-1) <= 0.0) {
					*(rp-1) = d;
					*rp = w / d;
				}
				else {
					*(rp-1) = -w / d;
					*rp = -d;
				}
			}
			else {
			/* the case conjugate complex roots */

				*(rp-1) = w;
				*rp = w;
				*(ip-1) = sqrt( -d );
				*ip = - *(ip-1);
			}
			rp += 2;
			ip += 2;
		}
		rp -= i;
		ip -= i;
		return( 1 );
	}


/********************************\
*                                *
*       sorting routines         *
*                                *
\********************************/

quicksort( i, j, a, b )
/* sorts the list a[i]--------a[j]
 * using the classical and efficient quicksort procedure
 */
	int		i, j;
	double		*a, *b;
	{
		double	pivot;
		int	k, pivotindex;

		pivotindex = findpivot( i, j, a );
		if (pivotindex != -1) {
			/* i.e. if all values are equal, do nothing */
			pivot = *(a+pivotindex);
			k = partition( i, j, pivot, a, b );
			quicksort( i, k-1, a, b );
			quicksort( k,  j, a, b );
		}
	}

findpivot( i , j , a )
/* returns 0 if a[i] and a[j] have identical values, otherwise returns the index of the
 * leftmost two different keys
 */
	int		i, j;
	double		*a;
	{
		double	firstvalue;
		int	k;
```

```
        firstvalue = *(a+i);

        /* scan for different values */
        for (k = i + 1 ; k <= j ; k++) {
                if (*(a+k) > firstvalue) return( k ); /* select larger value */
                else if (*(a+k) < firstvalue) return( i );
        }
        return( -1 );   /* different values were never found */
    }

partition( i, j, pivot, a, b )
    int             i, j;
    double          *a, *b, pivot;
    {
        int     l, r;
        double  q;

        l = i;
        r = j;
        do {
                q = *(a+l); *(a+l) = *(a+r); *(a+r) = q;
                q = *(b+l); *(b+l) = *(b+r); *(b+r) = q;
                while  (*(a+l) <  pivot) l++;
                while  (*(a+r) >= pivot) r--;
        }
        while (l <= r);
        return( l );
    }

/*
 * the end of the file alfpol.c
 */
```

```
/***********************************************************************\
*       "alfpsd" calculates power spectrum density function of         *
*       AR-process from AR-coefficients                                *
*----------------------------------------------------------------------*
*SYNTAX                                                                *
*   int alfpsd( p, alf, psd, n )                                       *
*       int     p;      :input/number of AR-coefficients               *
*       double  *alf;   :input/AR-coefficients                         *
*                        alf[i], i = 0,1, ... ,p                       *
*                        alf[0] = 1.0                                  *
*       double  *psd;   :output/power spectrum dinsity of AR-process   *
*                        at equally spaced n+1 frequencies             *
*                        (psd[0] is density at angular frequency = 0)  *
*                        (psd[n] is density at angular frequency = Pai) *
*       int     n;      :input/required points of psd                  *
*FUNCTIONS                                                             *
*       int exponent();                                                *
*       int set_zero();                                                *
*       int fft();                                                     *
*----------------------------------------------------------------------*
*                                       by H.Kato       17 Dec. 1988   *
*                                       revised         09 Nov. 1989   *
\***********************************************************************/

# include       <math.h>
# define        MAXFFT  4096

alfpsd( p, alf, psd, n )
   int          p;              /* number of AR-coefficients */
   double       *alf;           /* AR-coefficients */
   double       *psd;           /* power spectrum dinsity of AR-process */
   int          n;              /* required points of psd */
   {
        int     exponent(),     /* exponent number where mantissa is 2 */
                set_zero(),     /* fill array with zero         */
                fft();          /* Fast Fourier Transformation */
        int     n2,             /* n2 = n * 2 */
                exp_n,          /* exponent( n ) */
                i;
        double  xr[MAXFFT],     /* real part of FFT      */
                xi[MAXFFT],     /* imaginary part of FFT */
                tmp_p;

        if (n > MAXFFT) {
                return( 0 );
        }

        /* set array and do FFT */
        n2 = n * 2;
        set_zero( xr, n2);
        set_zero( xi, n2);
        for (i = 0 ; i <= p ; i++) {
                xr[i] = alf[i];
        }
        exp_n = exponent( n2 );
        fft( xr, xi, exp_n );

        /* calculate power spectrum */
        for (i= 0 ; i <=  n ; i++) (
                tmp_p  = xr[i] * xr[i] + xi[i] * xi[i];
                psd[i] = 1.0 / (tmp_p / (double)n2);
        }
        return( 1 );
   }

exponent( n )
/* this function returns the exponent number of the input integer where mantisa is 2
 * 2 ** (exponent( n )) = n
 */
   int          n;
   {
        int     i;

        if (n < 0) {
                return( -32768 );
        }
        else {
                for (i = 0 ; 1 ; i++) {
                        if ((int)ldexp( 1.0, i ) >= n) {
                                break;
                        }
                }
                return( i );
        }
```

```
    }

set_zero( x, n )
/* this function fills all the elements of the input array with zero
 */
    double      *x;     /* input and output array */
    int         n;      /* number of elements */
    {
        int     i;

        for (i = 0 ; i < n ; i++){
                x[i] = 0.0;
        }
    }

/*
 * the end of the file alfpsd.c
 */
```

```
/**********************************************************************\
*       "fft" executes Fast Fourier Transformation                    *
*---------------------------------------------------------------------*
*SYNTAX                                                               *
*   int fft( xr, xi, m )                                              *
*       double  *xr;    :input,output/real part of input or output data *
*       double  *xr;    :input,output/imaginary part of input or output *
*                        data                                         *
*       int     m;      :input/ 2 ** m = the number of FFT points     *
*---------------------------------------------------------------------*
*                                                                     *
*                       original coded by M.Akagi                     *
*                         revised by H.Kato           16 Nov. 1989    *
\**********************************************************************/

# include       <math.h>

fft( xr, xi, m )
   double       xr[];   /* real part of data */
   double       xi[];   /* imaginary part of data */
   int          m;      /* 2**m is the number of FFT points */
   {
        double  wr[2], wi[2], tmp_w;    /* rotators */
        int     n, n2, n1;              /* numbers of FFT points */
        int     i, j, k, l;             /* counters */
        double  tr, ti;                 /* temporary data */
        static double
                PI = 3.14159265358979323846264338327950288419716939;

        if (m <= 0) {
                return( 0 );
        }
        n = pow( 2.0, (double)m );              /* n = 2 ** m */
        if (m > 1) {
                if (m > 2) {

                        /* radix-2 FFT */
                        n2 = n;
                        for (l = 0 ; l < m-2 ; l++) {
                                n1 = n2;
                                n2 /= 2;
                                wr[0] = 1.0;
                                wi[0] = 0.0;
                                wr[1] = cos( PI/(double)n2 );
                                wi[1] = -sin( PI/(double)n2 );
                                for (j = 1 ; j <= n2 ; j++) {
                                        for (i = j-1, k = (n-j+n1)/n1 ; k-- ; i+=n1) {
                                                tr = xr[i] - xr[i+n2];
                                                ti = xi[i] - xi[i+n2];
                                                xr[i] = xr[i] + xr[i+n2];
                                                xi[i] = xi[i] + xi[i+n2];
                                                xr[i+n2] = tr * wr[0] - ti * wi[0];
                                                xi[i+n2] = tr * wi[0] + ti * wr[0];
                                        }
                                        tmp_w  = wr[0] * wr[1] - wi[0] * wi[1];
                                        wi[0]  = wr[0] * wi[1] + wi[0] * wr[1];
                                        wr[0]  = tmp_w;
                                }
                        }
                }

                /*  4-point DFT  */
                for (j = 3, k = n/4 ; k-- ; j+=4) {
                        tr = xr[j-3];
                        ti = xi[j-3];
                        xr[j-3] = tr + xr[j-1];
                        xi[j-3] = ti + xi[j-1];
                        xr[j-1] = tr - xr[j-1];
                        xi[j-1] = ti - xi[j-1];
                        tr = xr[j-2];
                        ti = xi[j-2];
                        xr[j-2] = tr + xr[j];
                        xi[j-2] = ti + xi[j];
                        tr = tr - xr[j];
                        ti = ti - xi[j];
                        xr[j] = ti;
                        xi[j] = -tr;
                }
        }

        /*  2-point DFT  */
        for (j = 1, k = n/2 ; k-- ; j+=2) {
                tr = xr[j-1];
                ti = xi[j-1];
```

```
                xr[j-1] = tr + xr[j];
                xi[j-1] = ti + xi[j];
                xr[j] = tr - xr[j];
                xi[j] = ti - xi[j];
        }

        /* bit reversal */
        for(i = 0, j = 0 ; i < n-1 ; i++) {
                if (i < j) {
                        tr = xr[j];
                        ti = xi[j];
                        xr[j] = xr[i];
                        xi[j] = xi[i];
                        xr[i] = tr;
                        xi[i] = ti;
                }
                k = n / 2;
                while (k <= j) {
                        j = j - k;
                        k = k / 2;
                }
                j = j + k;
        }
        return( 1 );
}
```

```
/**********************************************************************\
*       "local_max" finds local peaks of input data sequence          *
*---------------------------------------------------------------------*
*SYNTAX                                                               *
*   int local_max( t, n, f, p, b, m )                                 *
*               double  *t:     :input/data sequence                  *
*                                t[] is expected to be dB data        *
*               int     n;      :input/dimension of t                 *
*               double  *f;     :output/abcissa values of peaks       *
*                                ( frequency of peaks )               *
*               double  *p;     :output/ordinate values of peaks      *
*                                ( power of peaks )                   *
*               double  *b;     :output/-3dB bandwidths of peaks      *
*               int     *m;     :output/number of peaks               *
*FUNCTIONS                                                            *
*       int summit();                                                 *
*---------------------------------------------------------------------*
*                                       coded by M.Akagi              *
*                                       revised by H.Kato  16 Nov. 1989 *
\**********************************************************************/

local_max( t, n, f, p, b, m )
   double       t[];            /* power spectrum envelope */
   int          n;              /* dimension of t */
   double       f[];            /* abcissa values of peaks */
   double       p[];            /* ordinate values of peaks */
   double       b[];            /* -3dB bandwidths of peaks */
   int          *m;             /* number of peaks */
   {
        int     i;

        /* search peaks */
        for (*m = 0, i = 2 ; i < n-2 ; i++) {
                if ((t[i] > t[i-2]) &&
                    (t[i] > t[i-1]) &&
                    (t[i] > t[i+1]) &&
                    (t[i] > t[i+2]))
                {
                    /* parabola interpolation */
                    if (
                    summit( (double)i, t[i-1], t[i], t[i+1], &f[*m], &p[*m], &b[*m] )
                    ) {
                        *m += 1;
                    }
                }
        }
        return( 1 );
   }
```

```
/***********************************************************************\
*       "summit" finds parabola interpolated peak location using       *
*       given 3 points                                                 *
*----------------------------------------------------------------------*
*SYNTAX                                                                *
*   int summit( n0, p_1, p0, p1, n_peak, p_peak, bw )                  *
*               double  n0;      :input/abscissa value of a center point *
*               double  p_1:     :input/ordinate value of a pre-center  *
*               double  p0:      :input/ordinate value of a center point *
*               double  p1:      :input/ordinate value of a post-center *
*               double  *n_peak;:output/abscissa value of a found peak  *
*               double  *p_peak;:output/ordinate value of a found peak  *
*               double  *bw;     :output/-3dB bandwith of a found peak  *
*----------------------------------------------------------------------*
*                                        by H.Kato       16 Dec. 1988  *
*                                        revised          1 May. 1989  *
\***********************************************************************/

# include       <stdio.h>

summit( n0, p_1, p0, p1, n_peak, p_peak, bw )
   double       n0,      /* abscissa value of a center point */
                p_1,     /* ordinate value of a pre-center point */
                p0,      /* ordinate value of a center point */
                p1,      /* ordinate value of a post-center point */
                *n_peak,/* abscissa value of a found peak */
                *p_peak,/* ordinate value of a found peak */
                *bw;     /* -3dB bandwith of a found peak */
   {
        double  sqrt();
        double  a, b:    /* coefficients of parabola function            */
                         /*     Y = a * X**2 + b * X + c                 */
                         /* 'c' is implicitly zero by axis transformation */

        double  y_1, y1;/* transformed ordinate values */

        /* axis transformation */
        y_1 = p_1 - p0;
        y1  = p1  - p0;

        /* prevent the case 'a >= 0' */
        if ((0. < y_1) || (0. < y1)) {
                fprintf( stderr, "summit: No peak exists!!\n" );
                return( 0 );
        }
        else if ((y_1 + y1) == 0) {
                fprintf( stderr, "summit: Flat peak!!\n" );
                return( 0 );
        }

        /* calculate 'a','b', peak location and bandwidth */
        else {
                a = (y1 + y_1) / 2.;
                b = (y1 - y_1) / 2.;

                *bw     = 2. * sqrt( -3. / a );
                *n_peak = -b / (a * 2.) + n0;
                *p_peak = -(b * b) / (a * 4.) + p0;

                return( 1 );
        }
   }
```