CONTENTS

ビジュアルヘルプ – Igor Pro リファレンス（1）

このファイルには、Igor Pro のコマンド、関数、キーワードに関するリファレンス資料が含まれています。

コマンドと関数のシンタックス

外部コマンド（XOP）と外部関数（XFUNC）については、ここでは扱いません。
これらに関する詳細については、Igor Help Browser の「Command Help」タブと XOP ファイルと同じフォルダーにある XOP ヘルプファイルを参照してください。

以下の関数とコマンドの説明において、イタリック体で表記されているものは、数値または文字列式を指定できるパラメーターを示しています。
イタリック体ではないキーワードは、表示されているとおりにリテラルとして入力する必要があります。
これらの説明に含まれるコンマ、スラッシュ、中括弧、丸括弧は、常にリテラルとして扱われます。
角括弧は、オプションのフラグやパラメーターを囲んでいます。
省略記号（...）は、その直前の要素が何度でも繰り返される可能性があることを示しています。

斜体で表記されたパラメーターは、ユーザーが指定する値を表します。
「Name」で終わる斜体の単語は名前（ウェーブの名前例えば）であり、「Str」で終わるものは文字列です。
「Spec」（「仕様」を意味する）で終わる斜体の単語は、通常、説明文の中でさらに詳しく定義されています。
これらの接尾辞が使われていない場合、イタリック体の単語は数値式（リテラル数値、変数名、関数名、またはそれらの有効な組み合わせなど）を表します。

文字列と変数名は異なりますが、「文字列置換」を使えば、変数名が期待される場所で文字列を使うことができます。
文字列式の前頭に $ 演算子を付けるだけです。
ヘルプ String Substitution Using $ を参照してください。

ここでは構文の記述が数行にわたる場合がありますが、実際に作成するコマンドは 1 行に収める必要があります。

多くのコマンドには、オプションの「フラグ」があります。
値を指定できるフラグ（例えば、Make コマンドの /N=*n* フラグ。ここで *n* は値です）では、括弧を追加する必要がある場合があります。
例えば：

```
Make/N=1 aNewWave
```

ここでは *n* がリテラル「1」であるため、この記述は有効です。
n に数値式（リテラルな数値以外のもの）を使うには、括弧が必要です：

```
Make/N=(numberOfPoints) aNewWave    // 括弧がない場合はエラー
```

関数、コマンド、キーワードの使用方法の詳細については、ヘルプ Working with Commands、Programming Overview、Operations in Functions を参照してください。

コマンド

Abort [*errorMessageStr*]

Abort コマンドは、プロシージャの実行を中止します。

パラメーター

オプションの *errorMessageStr* は文字列式で、指定された場合、エラーアラートに表示されるエラーメッセージを指定します

詳細

Abort は、マクロがエラー状態になった時に、その実行を中止するための手段を提供します。

参照

ヘルプ Aborting Functions、Aborting Macros、Flow Control for Aborts
DoAlert コマンド

AddFIFOData *FIFOName*, FIFO_*channelExpr* [, *FIFO*_channelExpr]...

AddFIFOData コマンドは、FIFO_*channelExpr* 式を倍精度浮動小数点数として評価し、その値を指定された FIFO に格納します。

詳細

FIFO 内の各チャネルに対して、1 つの FIFO_*channelExpr* が存在する必要があります。

FIFO はデータ収集に使われます。

参照

ヘルプ FIFOs and Charts

AddFIFOVectData *FIFOName*, FIFO_*channelKeyExpr* [, FIFO_*channelKeyExpr*]...

AddFIFOVectData コマンドは AddFIFOData と似ていますが、式においてキーワードを使うことで、通常のチャネルには 1 つの数値を、特殊な画像ベクトルチャネルのデータを含むウェーブを指定できる点が異なります。

詳細

FIFO 内の各チャネルに対して、1 つの FIFO_*channelKeyExpr* が存在する必要があります。

FIFO_*channelKeyExpr* は、以下のいずれかになります：

- num=*numericExpression*
- vect=*wave*

最適な結果を得るには、ウェーブのデータポイント数は FIFO チャネルの定義に使ったポイント数と同じにし、データ型も同一にする必要があります。
NewFIFOChan を参照してください。

参照

ヘルプ FIFOs and Charts

AddMovieAudio *soundWave*

AddMovieAudio コマンドは、現在開いているムービーのオーディオトラックにオーディオサンプルを追加します。

パラメーター

soundWave には、振幅が -128 から +127 の範囲にあり、ムービーの冒頭で使われたプロトタイプ *soundWave* と同じ時間スケールを持つオーディオサンプルが含まれています。

フラグ

/Z　エラーの表示を抑制します。/Z オプションを使う場合は、コマンドが成功したかどうかを確認するために、出力変数 V_Flag を確認してください。

詳細

適切な形式のウェーブファイルを用意することで、16 ビットとステレオ音声付きのムービーを作成できます。
16 ビット音声を指定するには、ウェーブファイルの形式を符号付き 16 ビット整数（Make または Redimension での /W フラグ）にする必要があります。
ステレオを指定するには、2 列（または任意のチャネル数）のウェーブファイルを使ってください。

出力変数

V_Flag　コマンドが成功した場合は 0 に、失敗した場合は 0 以外のエラーコードに設定します。

V_Flag は、/Z フラグを使った場合にのみ設定されます。

参照

ヘルプ Movies
NewMovie コマンド、AddMovieFrame コマンド

AddMovieFrame [/PICT=*pictName*]

AddMovieFrame コマンドは、トップグラフ、ページレイアウト、Gizmo ウィンドウ、または指定された画像を、現在開いているムービーに追加します。

Igor Pro 7.0 で、ページレイアウトと Gizmo ウィンドウのサポートが追加されました。

ムービーを生成するプロシージャを作成する時は、ターゲットウィンドウへのすべての変更処理が完了した後、かつ AddMovieFrame を呼び出す前に、DoUpdate コマンドを呼び出す必要があります。
これにより、ウィンドウに加えられた変更をすべて処理できるようになります。

Igor Pro 7 以降では、NewMovie を呼び出した時点でのターゲットウィンドウが記憶され、AddMovieFrame を呼び出した時にそのウィンドウが最前面のウィンドウでなかったとしても、AddMovieFrame によってそのウィンドウが使われます。

/PICT フラグが指定された場合、ターゲットウィンドウの代わりに、画像ギャラリー（ヘルプ Pictures を参照）から指定された画像が使われます。

フラグ

/Z　エラーの表示を抑制します。/Z オプションを使う場合は、コマンドが成功したかどうかを確認するために、出力変数 V_Flag を確認してください。

出力変数

V_Flag　コマンドが成功した場合は 0 に、失敗した場合は 0 以外のエラーコードに設定します。

V_Flag は、/Z フラグを使った場合にのみ設定されます。

参照

ヘルプ Movies
NewMovie コマンド、AddMovieAudio コマンド

AddWavesToBoxPlot [/W=*winName* /T=*traceName* /INST=*traceInstance*] *wave* [, *wave*] ...

AppendBoxPlot で作成された既存のボックスプロットのトレースに、1D ウェーブを追加します。

AddWavesToBoxPlot は、Igor Pro 8.0 で追加されました。

ボックスプロットのトレースでは、トレース内の各データセットを定義するために複数のウェーブが必要となる場合があり、またウェーブ名が非常に長くなることもあるため、AppendBoxPlot で作成されたリストにウェーブを追加するための AddWavesToBoxPlot コマンドが用意されています。

フラグ

/T=*traceName*
/INST=*traceInstance*
これらのフラグは、ウェーブを追加する既存のボックスプロットトレースの名前とインスタンス番号を指定します。
/INST を指定せずに /T を使うことも可能です。
その場合、インスタンス番号 0 のトレースが使われます。
/T を指定せずに /INST を使わないでください。

トレース名とトレースインスタンス番号に関する詳細は、ヘルプ Creating Graphs を参照してください。

/T と /INST の両方が指定されていない場合、デフォルトではグラフ上に表示されている最前面のボックスプロットトレースが使われます。
これは、最も最近追加されたボックスプロットトレースになります。

/W=*winName*
指定されたグラフウィンドウまたはサブウィンドウに追加します。
省略された場合、AddWavesToBoxPlot はアクティブなウィンドウまたはサブウィンドウに対して実行されます。

winName を使ってサブウィンドウを特定する場合は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

元の AppendBoxPlot コマンドに X ウェーブが含まれている場合、ボックスプロットデータセットのウェーブのリストに含まれるウェーブの総数は、X ウェーブのデータポイント数を超えてはなりません。

ボックスプロットのトレースが複数列のウェーブで定義されている場合、このコマンドを使ってウェーブを追加することはできません。

参照

ヘルプ Box Plots
AppendBoxPlot コマンド、ModifyBoxPlot コマンド

AddWavesToViolinPlot [/W=*winName* /T=*traceName* /INST=*traceInstance*] *wave* [, *wave*] ...

AppendViolinPlot で作成された既存のバイオリンプロットのトレースに、1D ウェーブを追加します。

AddWavesToViolinPlot は Igor Pro 8.0 で追加されました。

バイオリンプロットのトレースでは、トレース内の各データセットを定義するために多数のウェーブが必要となる場合があり、またウェーブの名前が非常に長くなることもあるため、AppendViolinPlot で作成されたリストにウェーブを追加するための AddWavesToViolinPlot コマンドが用意されています。

フラグ

/T=*traceName*
/INST=*traceInstance*

これらのフラグは、ウェーブを追加する既存のバイオリンプロットトレースの名前とインスタンス番号を指定します。
/INST を指定せずに /T を使うことも可能です。
その場合、インスタンス番号 0 のトレースが使われます。
/T を指定せずに /INST を使わないでください。

トレース名とトレースインスタンス番号に関する詳細は、ヘルプ Creating Graphs を参照してください。

/T と /INST の両方が指定されていない場合、デフォルトではグラフ上に表示されている最前面のバイオリンプロットトレースが使われます。
これは、最も最近追加されたバイオリンプロットトレースになります。

/W=*winName*

指定されたグラフウィンドウまたはサブウィンドウに追加します。
省略された場合、AddWavesToViolinPlot はアクティブなウィンドウまたはサブウィンドウに対して実行されます。

winName を使ってサブウィンドウを特定する場合は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

元の AppendViolinPlot コマンドに X ウェーブが含まれている場合、バイオリンプロットデータセットのウェーブのリストに含まれるウェーブの総数は、X ウェーブのデータポイント数を超えてはなりません。

バイオリンプロットのトレースが複数列のウェーブで定義されている場合、このコマンドを使ってウェーブを追加することはできません。

参照

ヘルプ Violin Plots
AppendViolinPlot コマンド、ModifyViolinPlot コマンド

AdoptFiles [*flags*]

AdoptFiles コマンドは、外部ファイルやウェーブを現在のエクスペリメントに取り込みます。

次にエクスペリメントを保存すると、ファイルとウェーブデータは、パックされたエクスペリメントの場合はエクスペリメントファイル内に、パックされていないエクスペリメントの場合はエクスペリメントフォルダー内に保存されます。
外部ファイルへの参照は削除されます。

AdoptFiles は、Execute/P 経由以外では関数内から呼び出すことはできません。

フラグ

/A　　エクスペリメント内のすべての外部ノートブック、ユーザープロシージャファイル、すべてのウェーブを取り込みます。WaveMetrics プロシージャファイルは取り込まれません。/A は /NB/UP/DF と同等です。

/DF　　エクスペリメント外部に保存されたすべてのウェーブを取り込みます。

/DF=*dataFolderPathStr*
　　指定されたデータフォルダー内にある、エクスペリメントの外部に保存されたすべてのウェーブを取り込みます。

/I　　Adopt All ダイアログを表示し、ユーザーがそこで選択した項目を採用します。

/NB　　すべての外部ノートブックファイルを取り込みます。

/UP　　すべての外部ユーザープロシージャファイルを取り込みます。

/W=*winTitleOrName*
　　指定されたノートブックまたはプロシージャファイルを採用します。/W オプションは Igor Pro 7.02 で追加されました。

　　winTitleOrName は文字列ではなく名前であるため、/W は次のように指定します：

```
/W=$"New Polar Graph.ipf"
```

　　または

```
/W=Notebook0
```

　　独立モジュールを使う場合、winTitleOrName は、Procedure ウィンドウタイトルにスペースを１つ入れ、その後に括弧で囲んだ独立モジュールの名前を指定します。詳細については、ヘルプ Independent Modules を参照してください。

/WP　　すべての WaveMetrics プロシージャファイルを採用します。

/WV=*wave*　指定されたウェーブのみを採用します。

詳細

現在のエクスペリメントとは別の場所に保存されたファイルおよびウェーブのみが採用されます。
このような独立したファイルに関する説明については、ヘルプ References to Files and Folders を参照してください。

実際に採用されたオブジェクトの数が V_Flag に返されます。

１つのウェーブのみを採用するには、次のようにします：

```
AdoptFiles/WV=wave
```

１つのノートブックまたはプロシージャウィンドウのみを採用するには、AdoptFiles/W=*winTitleOrName* を使います。

コマンドラインとマクロの例

```
// コマンドラインまたはマクロから AdoptFiles を使う
AdoptFiles/I                          // Adopt All ダイアログを表示する
AdoptFiles/A/WP                       // 採用できるものはすべて採用する
AdoptFiles/DF/NB/UP/WP                // 採用できるものはすべて採用する
AdoptFiles/DF=root:subfolder          // root:サブフォルダーに保存されている外部のウェーブをすべて採
```

用する

```
AdoptFiles/W=$"Proc0.ipf"                  // Proc0.ipf が外部に保存されている場合は、それを採用する
AdoptFiles/WV=GetWavesDataFolder(wave0,2)  // wave0 が外部に保存されている場合は、それを採用する
```

関数の例

```
// ユーザー定義関数から AdoptFiles を使う場合 - Execute/P を使う必要がある
Execute/P "AdoptFiles/A"                   // すべてのユーザーファイルとウェーブのスケジュール設定を行う
Execute/P "AdoptFiles/WV="+GetWavesDataFolder(w,2)     // ウェーブ w の採用スケジュールを設定する
```

参照

ヘルプ Adopt All、Adopting Notebook and Procedure Files、Avoiding Shared Igor Binary Wave Files、Avoiding Shared Igor Binary Wave Files

APMath [/EX=*exDigits* /N=*numDigits* /V/Z] *destStr=Expression*

APMath コマンドは、基本的な数式に対して任意精度の計算を行います。
最終結果は、指定された文字列 destStr に変換され、これを表示したり、別の APMath コマンドにおいて（指定された精度での）値として使ったりすることができます。

パラメーター

destStr　代入式の代入先文字列を指定します。*destStr* が既存の変数でない場合、このコマンドによって作成されます。関数内で実行する場合、*destStr* がまだ存在しない場合はローカル変数となります。

Expression　定数、ローカル変数、グローバル変数、参照変数、または文字列、さらにウェーブ要素を含む代数式で、以下の演算子と組み合わせて使われるものです：

オペレーター	**プロシージャ**
^	高
* /	…
+ -	低

オペレーター

サポートしているオペレーター：

+	スカラー値の加算
-	スカラー値の減算
*	スカラー値の乗算
/	スカラー値の除算
^	累乗

スカラーパラメーターを持つ APMath 関数

スカラーパラメーターに対する任意精度演算では、以下の関数がサポートされています。

sqrt(*x*)	*x* の平方根
cbrt(*x*)	*x* の立方根
pi	括弧は不要です
sin(*x*)	*x* の sine（正弦）
cos(*x*)	*x* の cosine（余弦）

tan(*x*)　　*x* の tangent（正接）

asin(*x*)　　*x* の inverse sine（逆正弦）

acos(*x*)　　*x* の inverse cosine（逆余弦）

atan(*x*)　　*x* の inverse tangent（逆正接）

atan2(*y*,*x*)　*y*/*x* の inverse tangent（逆正接）

log(*x*)　　*x* の対数

log10(*x*)　　*x* の対数（底 10）

exp(*x*)　　指数関数 e^*x*

pow(*x*,*n*)　　*x* の *n* 乗（*n* は必ずしも整数である必要はない）

sinh(*x*)　　*x* の hyperbolic sine（双曲正弦）

cosh(*x*)　　*x* の hyperbolic cosine（双曲余弦）

tanh(*x*)　　*x* の hyperbolic tangent（双曲正接）

asinh(*x*)　　*x* の inverse hyperbolic sine（逆双曲正弦）

acosh(*x*)　　*x* の inverse hyperbolic cosine（逆双曲余弦）

atanh(*x*)　　*x* の inverse hyperbolic tangent（逆双曲正接）

ceil(*x*)　　*x* より大きい整数の中で最小のものを返す

comp(*x*,*y*)　　*x*==*y* の場合は 0、*x*>*y* の場合は 1、*y*>*x* の場合は -1

factorial(*n*)　整数 *n* の階乗

floor(*x*)　　*x* より小さい最大の整数

gcd(*x*,*y*)　　*x* と *y* の最大公約数

lcd(*x*,*y*)　　*x* と *y* の最小公倍数（*x***y*/gcd(*x*,*y*) で与えられる）

sgn(*x*)　　*x* の符号を返す。*x*==0 の場合は 0

binomial(*n*,*k*)
　　整数 *n* と *k* に対する二項関数

Bernoulli(*n*)
　　ベルヌーイ数 Bn（Bn(1)=-1/2）

Stirling2(*n*,*k*)
　　第 2 種スターリング数

ウェーブパラメーターを持つ APMath 関数

ウェーブに対する任意精度演算では、以下の関数がサポートされています。

次の制限は、ウェーブに関する以下のすべての APMath 関数に適用されます。

1. パラメーター *w* は、単純なウェーブ参照でなければなりません。
   ウェーブファイルへのデータフォルダーのパスや、ウェーブファイルを指す $ 式とすることはできません。

2. ウェーブは、実数の数値ウェーブでも、文字列形式で任意精度の数値を含むテキストウェーブでも構いません。

3. 複素数ウェーブは使用できません。

4. 64 ビット整数のウェーブは使用できません。

5. ウェーブに NaN または INF が含まれている場合、これらの関数はエラーを返します。

6. 多次元ウェーブは 1D として扱われます。

kurtosis(w)　ウェーブ w 全体の尖度。尖度に関する説明については、WaveStats コマンドを参照してください。kurtosis 関数は Igor Pro 8.0 で追加されました。

mean(w)　ウェーブ w 全体の平均値。Igor Pro 8.0 で追加されました。

skew(w)　ウェーブ w 全体の歪度。歪度に関する説明については、WaveStats コマンドを参照してください。skew 関数は Igor Pro 8.0 で追加されました。

sum(w)　ウェーブ w 全体の合計値。sum 関数は Igor Pro 8.0 で追加されました。

variance(w)　ウェーブ w 全体の分散。分散に関する説明については、WaveStats コマンドを参照してください。variance 関数は Igor Pro 8.0 で追加されました。

フラグ

/EX=*exDigits*　計算の中間段階で、有効桁数（/N）に加えて追加される桁数を指定します。

/N=*numDigits*　最終結果の精度を指定します。中間計算の結果に桁数を追加するには、/EX を使ってください。

/V　詳細モード：代入処理を実行するだけでなく、その結果を履歴に出力します。

/Z　エラーを報告しません。

詳細

デフォルトでは、すべての任意精度の数学演算は *numDigits*=50 と *exDigits*=6 で実行され、その結果、少なくとも小数点以下 56 桁の最終結果が得られます。
組み込みの変数型にはこれほど高い精度の数値を表現できるものがないため、任意精度の数値は文字列として格納する必要があります。
このコマンドでは、文字列と定数の間で自動的に変換が行われます。
上記のすべての数値関数は、指定された精度を使用して評価されます。
このコマンドでサポートされていない関数が必要な場合は、それらを事前に計算し、結果をローカル変数に格納する必要がある場合があります。

このコマンドは結果を *destStr* に格納しますが、*destStr* は実行前に存在している場合も、存在していない場合もあります。
コマンドラインからこのコマンドを実行する場合、*destStr* がまだ存在しないときは、現在のデータフォルダー内のグローバル文字列として作成されます。
すでに存在している場合は、コマンドの結果がその値を上書きします（通常の文字列代入と同様です）。
ユーザー関数内では、*destStr* はローカル文字列、SVAR、または参照渡しされた文字列のいずれかになります。
destStr がこれらに該当しない場合、このコマンドはその名前でローカル文字列を作成します。

任意精度の算術演算は、同等の浮動小数点演算に比べてはるかに処理速度が遅くなります。
実行時間は桁数に比例するため、/N フラグを使って、必要な最小桁数に評価を制限すべきです。

出力変数

Igor Pro 8.0 以降では、APMath コマンドは以下の変数に情報を返します。

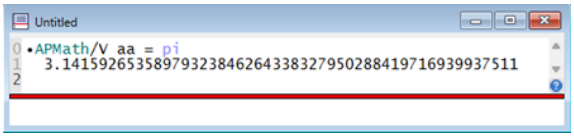
V_Flag　　操作が成功した場合は 0 に、失敗した場合は 0 以外のエラーコードに設定します。

V_value　　出力文字列の倍精度表現に設定します。

例

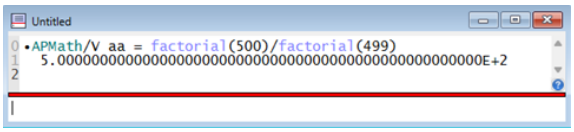
円周率を 50 桁まで計算：

```
APMath/V aa = pi
```



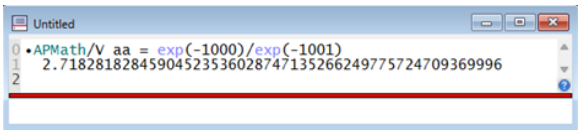
大きな階乗の比を評価：

```
APMath/V aa = factorial(500)/factorial(499)
```



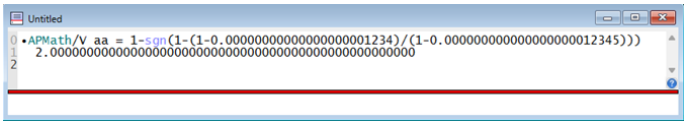
大きな階乗の比を評価：

```
APMath/V aa = exp(-1000)/exp(-1001)
```



異なるサイズの値の除算：

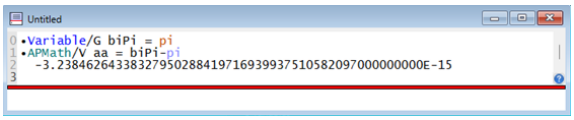
```
APMath/V aa = 1-sgn(1-
    (1-0.00000000000000000001234)
    /(1-0.00000000000000000012345)))
```



これを倍精度で評価しようとすると、異なる結果が得られます。

組み込みの pi と任意精度の pi の違い：

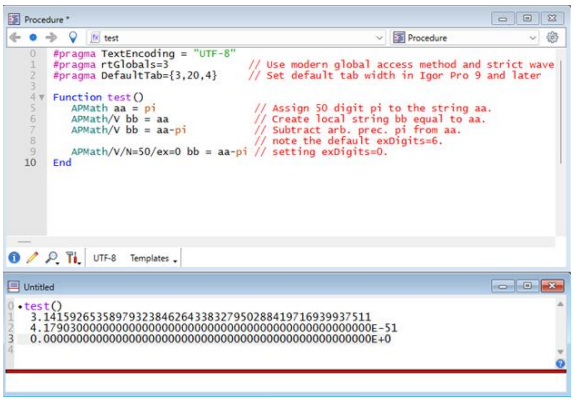
```
Variable/G biPi = pi
APMath/V aa = biPi-pi
```



精度のコントロール：

<プロシージャウィンドウ>

```
Function test()
   // 文字列 aa に 50 桁の円周率を代入
   APMath aa = pi
   // aa と同じ値を持つローカル文字列 bb を作成
   APMath/V bb = aa
   // aa から任意精度の pi を引く
   APMath/V bb = aa-pi
   // デフォルトの exDigits=6 に注意
   // exDigits=0 に設定
   APMath/V/N=50/ex=0 bb = aa-pi
End
```

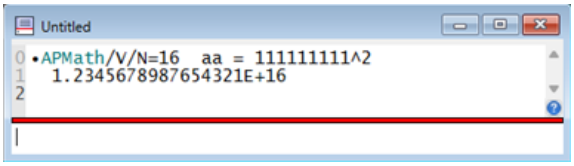


<コマンドライン>

```
test()
```

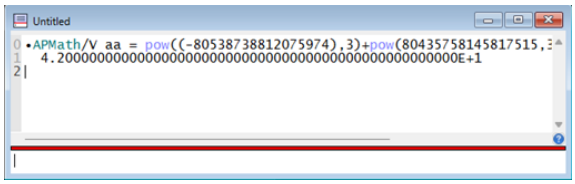
精度コントロール：

```
APMath/V/N=16  aa = 111111111^2
```



42 に関する「3 つの立方数の和」の問題の解法：

```
APMath/V aa =
  pow((-80538738812075974),3)
  +pow(80435758145817515,3)+
  pow(126021232973356631,3)
```



Append

Append コマンドは、対象のウィンドウに応じて、AppendToGraph、AppendToTable、または AppendToLayout として解釈されます。
これは、ユーザー定義関数を実行している場合には機能しません。
そのため、現在は Append ではなく、AppendToGraph、AppendToTable、または AppendToLayoutObject を使うことを推奨します。

AppendBoxPlot [*axis flags*][/W=*winName* /TN=*traceName* /VERT[=*doVert*] /CATL[=*doCatLabels*]] *wave*[, *wave*, ...] [*vs xWave*]

AppendBoxPlot コマンドは、ターゲットまたは名前付きグラフにボックスプロットのトレースを追加します。
ボックスプロットは、データ値の分布の概要を表示する方法の 1 つです。
データ分布の概要を表示する別の方法として、バイオリンプロットがあります。

AppendBoxPlot は、Igor Pro 8.0 で追加されました。

Igor Pro では、ボックスプロットのトレースは 1 つのグラフトレースとして扱われます。
グラフからの削除やトレースの並べ替えなど、多くのコマンドは、他のグラフトレースと同様にボックスプロットのトレースに対しても同様に機能します。

注記
DisplayBoxPlot コマンドはありません。Display の後に AppendBoxPlot を実行してください。

箱ひげ図のトレースに含まれる 1 つのボックスプロットのデータは、1 つの 1D ウェーブ全体、または複数列からなるウェーブの 1 つの列のいずれかから取得されます。
トレースに含まれるボックスプロットの数は、ウェーブのリストに含まれる 1D ウェーブの数、または 1 つの複数列ウェーブの列数によって決まります。
1D ウェーブと複数列ウェーブを混在させることはできません。

最大 100 個の個別の 1D ウェーブをリストアップできます。
ボックスプロットのトレースに 100 個以上のボックスを表示したい場合は、複数列のウェーブを使うか、AddWavesToBoxPlot コマンドを使ってリストに追加する必要があります。

xWave を指定しない場合、各ボックスプロットは数値軸上の X=0、1 などの位置に配置されます。
データが複数列のウェーブ形式である場合、配置はデフォルトで行列ウェーブの X 軸スケーリングに基づいて行われます。
数値の *xWave* を指定することで、各ボックスプロットを X 軸上の任意の位置に配置することができます。
テキストの *xWave* を指定すると、ボックスプロットはカテゴリ軸を使って表示されます。
複数列のウェーブで /CATL フラグを使うと、ディメンションラベルをカテゴリラベルとして使ってカテゴリ X 軸が生成されます。
ヘルプ Category Plots と Dimension Labels を参照してください。

xWave に含まれるポイント数は、1D Y ウェーブの数、あるいは複数列のウェーブにおける列の数以上でなければなりません。
ウェーブのリストが長くなりすぎてコマンドラインに入力しきれなくなる場合があるため、リストよりも長い

xWave を使うことができます。
リストを完成させるには、AddWavesToBoxPlot を使ってください。

フラグ

/L/R/B/T　　これらの軸フラグは、AppendToGraph で使われるものと同じです。

/CATL[=*doCatLabels*]
列次元のラベルを使って、カテゴリ軸上にボックスプロットを作成します。この時、列次元のラベルがカテゴリラベルとして使われます。*doCatLabels* は 0 または 1 で、/CATL は /CATL=1 と同等です。

/TN=*traceName*
トレースに独自の名前を付けることができます。これは、同じ名前を持ちますが異なるデータフォルダーからのウェーブを表示する時に便利です。詳細はヘルプ User-defined Trace Names を参照してください。ただし、AppendBoxPlot の /TN フラグは、コマンド名の直後という通常の位置に指定します。

/VERT[=*doVert*]
個々のボックスプロットを Y 軸に沿って縦方向に配置します。*doVert* は 0 または 1 で、/VERT は /VERT=1 と同等です。

/VERT は ModifyGraph SwapXY と似ていますが、トレースごとに処理を行います。横長のボックスプロットを作成するには、ModifyGraph SwapXY または AppendBoxPlot/VERT のいずれかを使ってください。

/W=*winName*
指定されたグラフウィンドウまたはサブウィンドウに追加します。省略された場合、AppendBoxPlot はアクティブなウィンドウまたはサブウィンドウを対象とします。
プロシージャ、マクロ、またはコマンドラインで使う場合、これは最初に指定されるフラグでなければなりません。

winName を使ってサブウィンドウを特定する場合は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

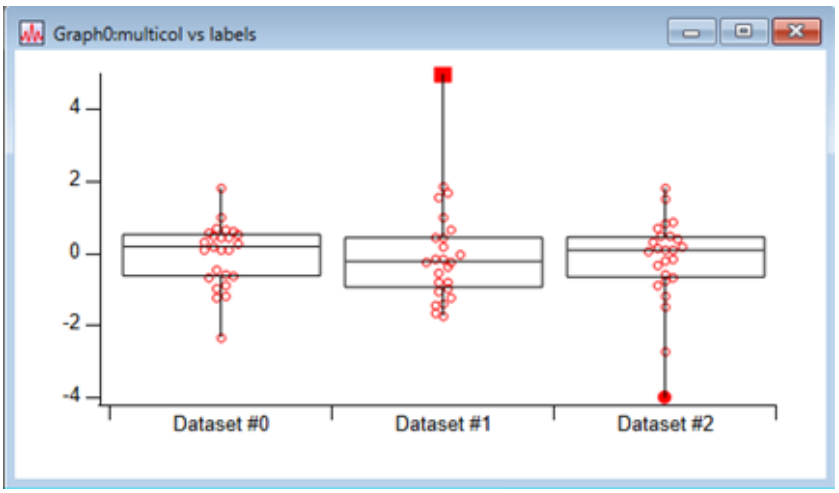
詳細

ボックスプロットトレースの全体的な外観の一部は、ModifyGraph を使って設定できます。
デフォルトでは、線の色は黒で、外れ値ではないデータポイントのマーカーは、通常のトレースマーカーの 0.7 倍の大きさの空の円になります。
Modify Box Plot ダイアログまたは ModifyBoxPlot コマンドを使うと、通常のトレースと同様に、ModifyGraph で設定されたトレースの色、線の太さ、線の破線スタイル、マーカー、マーカーのサイズを使うように選択できます。
ボックスプロットトレースのさまざまな部分について、これらの特性を詳細にコントロールするには、ModifyBoxPlot を使います。
外れ値、極端な外れ値、外れ値ではないデータを表すマーカーは、ModifyGraph で設定されたマーカーよりも優先されます。

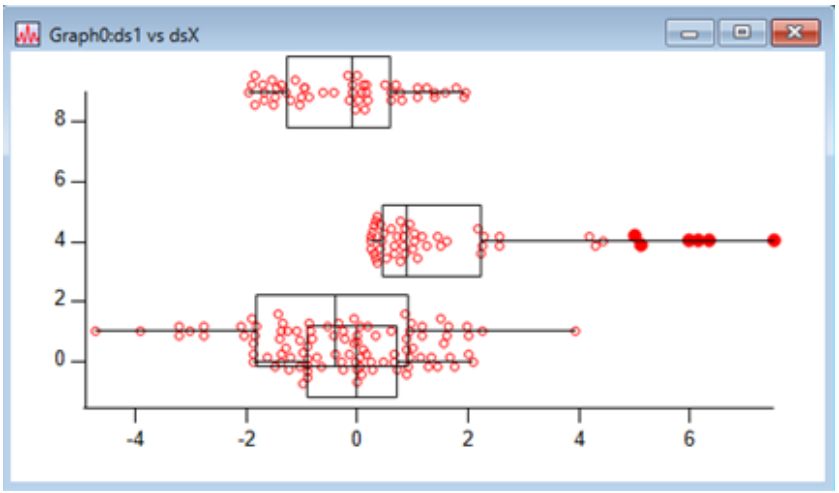
例

```
// 垂直ボックスプロットのデモ
Make/O/N=(25,3) multicol            // 25 行 3 列のウェーブ
SetRandomSeed(.4)
multicol = gnoise(1)                // 正規分布に従う 3 つのデータセット
multicol[20][1] = 5                 // 極端な外れ値
multicol[13][2] = -4                // 外れ値
Make/O/N=3/T labels                 // カテゴリプロットを作成するためのテキストウェーブ
```

```
labels = "Dataset #"+num2str(p)     // カテゴリプロットの x 軸のラベル
Display; AppendBoxPlot multicol vs labels
```




```
// 水平ボックスプロットのデモ
Make/O/N=50 ds1, ds2, ds3, ds4
Make/O/N=4 dsX
ds1 = gnoise(1)
ds2 = gnoise(2)
ds3 = logNormalNoise(0, 1)
ds4 = enoise(2)
dsX = p^2
Display; AppendBoxPlot ds1,ds2,ds3,ds4 vs dsX
ModifyGraph swapXY=1              // ボックスを水平にする
ModifyGraph margin(top)=20        // 上部の余白が狭すぎる可能性があります
```



参照

Display、AppendToGraph、ModifyGraph、ModifyBoxPlot コマンド
ヘルプ Box Plots、Violin Plots

AppendImage [/G=*g* /LAYR=*layer* /W=*winName*][*axisFlags*] *matrix* [*vs* {*xWaveName*, *yWaveName*}]

AppendImage コマンドは、行列を画像としてターゲットグラフまたは名前付きグラフに追加します。
デフォルトでは、画像は左軸と下軸に対してプロットされます。

パラメーター

matrix は、偽色またはインデックスカラー画像用の N×M 行列であるか、あるいは赤、緑、青の各データ平面を含む 3D N×M×3 ウェーブである場合があります。
また、４番目の平面にアルファ値を含む 3D N×M×4 ウェーブである場合もあります。

行列に３つまたは４つ以外の複数の平面が含まれている場合、あるいは３つまたは４つの平面と複数のチャンクが含まれている場合、ModifyImage plane キーワードを使って、表示したいサブセットを指定することができます。

xWaveName と *yWaveName* を指定する場合、*xWaveName* は X 座標の値を、*yWaveName* は Y 座標の値を指定します。
これにより、ピクセルサイズが不均一な画像が生成されます。
どちらの場合も、* を使うことで、行列の次元スケーリングに基づいて計算された値を指定できます。
xWaveName または *yWaveName* を使う場合は、以下の「詳細」のセクションを参照してください。

フラグ

axisFlags　フラグ /L/R/B/T は、AppendToGraph で使われるものと同じです。

/G=*g*　*g*=1：　３色または４色の画像を直接（RGB）カラーとして自動検出しないようにします。
g=1：　/G フラグを指定しない場合と同じ（デフォルト）。

/LAYR=*layer*　画像を挿入する画像レイヤーを選択します。
layer=0：　画像をレイヤー0 の最前面（つまり、レイヤー0 のリストの最後）に、トレースや軸の下に配置します。
layer=1：　画像をレイヤー1 の最前面（つまり、レイヤー1 のリストの最後）に、トレースや軸よりも上に配置します。

/LAYR を省略すると、/LAYR=0 と同じになります。

/PACK=*pkMode*

このフラグを使って、画像のバイナリ・パッキング方式を指定します。デフォルトでは、pkMode=*packSTD* となります。ここで、2D 数値ウェーブはグレースケール画像を表し、3D ウェーブはカラー画像を表します。詳細は、ヘルプ Direct Color Packing Modes を参照してください。

/W=*winName*

指定されたグラフウィンドウまたはサブウィンドウに追加します。省略された場合、処理はアクティブなウィンドウまたはサブウィンドウに適用されます。プロシージャ、マクロ、またはコマンドラインで使う場合、これは最初に指定されるフラグでなければなりません。

winName を使ってサブウィンドウを特定する場合は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

グラフに画像を追加する場合、各画像データポイントは長方形として表示されます。
オプションの X と Y ウェーブを指定して、長方形の辺の座標を定義することができます。
これらのウェーブには、行列の X 軸（行）または Y 軸（列）の次元よりも１つ多いデータポイントが含まれている必要があります。
また、これらのウェーブは、厳密に増加するか、厳密に減少するものでなければなりません。
詳細については、ヘルプ Image X and Y Coordinates を参照してください。

偽色の場合、行列内の値はカラーテーブルに線形マッピングされます。
ModifyImage コマンドで使われる ctab キーワードを参照してください。
インデックスカラーの場合、行列内の値は Z 値として解釈され、ユーザーが指定した３列のカラー行列から参照されます。

ModifyImage コマンドで使われる cindex キーワードを参照してください。
ダイレクトカラーの NxMx3 ウェーブには、各ピクセルの実際の赤、緑、青の値が含まれます。
NxMx4 ウェーブには、アルファチャンネルが追加されます。
数値型が unsigned bytes の場合、輝度の範囲は 0 から 255 です。
その他のすべての数値型の場合、輝度の範囲は 0 から 65535 です。

デフォルトでは、非直接色行列は、Grays カラーテーブルと自動スケールモードを使って、最初は偽色として表示されます。
行列が複素数の場合、画像は Z 値の絶対値、すなわち sqrt(real2 + imag2) を使って表示されます。

参照

ヘルプ Image X and Y Coordinates、Color Blending
NewImage、ModifyImage、RemoveImage コマンド

AppendLayoutObject [*flags*] *objectType objectName*

AppendLayoutObject コマンドは、1 つのオブジェクトをトップレイアウト、または /W フラグで指定されたレイアウトに追加します。
このコマンドの対象は、アクティブなページ、または /PAGE フラグで指定されたページとなります。

AppendToLayout コマンドとは異なり、AppendLayoutObject はユーザー定義関数内で使用できます。
したがって、新しいプログラムでは AppendToLayout の代わりに AppendLayoutObject を使うべきです。

パラメーター

objectType は、追加するオブジェクトのタイプを指定します。
これは、`graph`、`table`、`picture`、`Gizmo` のいずれかのキーワードです。

objectName は、追加するグラフ、テーブル、画像、または Gizmo ウィンドウの名前です。

objectType と *objectName* の間にはスペースを入れてください。
コンマは使用できません。

フラグ

/D=*fidelity*　*fidelity* を 0 に設定するとオブジェクトは低精度になり、1（デフォルト）に設定すると高精度になります。これは画面への描画にのみ影響し、エクスポートや印刷には影響しません。低い精度はより高速ですが、精度は低下します。描画に非常に時間がかかるグラフにのみ使うべきです。

/F=*frame*　*frame*=0：　フレームなし
frame=1：　シングルフレーム（デフォルト）
frame=2：　ダブルフレーム
frame=3：　影付きフレーム

/T=*trans*　*trans*=0：　不透明（デフォルト）
trans=1：　透明

これを有効にするには、オブジェクト自体も透明でなければなりません。注釈には、独自の透明／不透明設定があります。グラフは、背景が白の場合にのみ透明になります。PICT ファイルは、作成時に透明または不透明に設定されている場合がありますが、Igor Pro では不透明な PICT を透明にすることはできません。

/PAGE=*page*　指定されたページにオブジェクトを追加します。

ページ番号は 1 から始まります。現在のページを表示するには、/PAGE を省略するか、*page*=0 を指定してください。

/PAGE フラグは Igor Pro 7.0 で追加されました。

/R=(*l*, *t*, *r*, *b*)　オブジェクトのサイズと位置を設定します。省略した場合、オブジェクトはデフォルトのサイズと位置で配置されます。*l*、*t*、*r*、*b* は、それぞれオブジェクトの左、上、右、下の座標を表します。座標はポイント単位で表され、用紙の左上隅を基準としています。

/W=*winName*　*winName* は、オブジェクトを追加するページレイアウトウィンドウの名前です。/W を省略した場合、または *winName* が $"" の場合、最前面のページレイアウトが使われます。

参照

NewLayout、ModifyLayout、RemoveLayoutObjects、Textbox、Legend コマンド

AppendMatrixContour [*axisFlags*][/W=*winName* /F=*formatStr*] *zWave* [*vs* {*xWave*, *yWave*}]

AppendMatrixContour コマンドは、Rainbow カラーテーブルを使って、自動スケーリングされたコンターレベルを持つ z 値の行列のコンターを、ターゲットグラフまたは名前付きグラフに追加します。

注記

DisplayContour コマンドはありません。`Display; AppendMatrixContour` を使ってください。

zWave は行列（2D ウェーブ）でなければなりません。

一連の XYZ トリプレットにコンターを描画するには、AppendXYZContour を使います。

xWave と *yWave* の仕様を指定すると、*xWave* は行の X 値を、*yWave* は列の Y 値を提供します。
これにより、Z 値の「不均等なグリッド」が実装されます。

xWave と *yWave* の指定を省略した場合、*zWave* の X と Y のスケーリング済みインデックスを X と Y の値として使います。
また、*xWave* または *yWave* の代わりに *（アスタリスク記号）を使った場合も、*zWave* のスケーリング済みインデックスを使います。

マクロ内で、デフォルト値に基づいてコンターが計算、表示される前にコンターレベルの外観を変更するには、「;DelayUpdate」を付加し、AppendMatrixContour コマンドの直後に適切な ModifyContour コマンドを記述します。
最後の ModifyContour コマンドを除くすべてのコマンドにも、同様に「;DelayUpdate」を付加する必要があります。
関数内では DelayUpdate は必要ありませんが、DoUpdate を使うと、すべての関数が完了するまで待機するというデフォルトの動作ではなく、コンタートレースを即座に生成させることができます。

コマンドラインでは、Display コマンド、それに続く AppendMatrixContour コマンド、ModifyContour コマンドを、セミコロンで区切ってすべて 1 行に入力することができます。

```
Display; AppendMatrixContour MyMatrix; ModifyContour ...
```

フラグ

axisFlags　フラグ /L/R/B/T は、AppendToGraph で使われるものと同じです。

/F=*formatStr*　コンタートレースに割り当てる名前を指定します。詳細は以下を参照してください。

/W=*winName*　指定されたグラフウィンドウまたはサブウィンドウに追加します。省略された場合、処理はアクティブなウィンドウまたはサブウィンドウに適用されます。プロシージャ、マクロ、またはコマンドラインで使う場合、これは最初に指定されるフラグでなければなりません。

winName を使ってサブウィンドウを特定する場合は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

AppendMatrixContour は、コントートレースを作成して表示します。
これらのトレースは、Modify Contour Appearance ダイアログを使ってまとめて変更することも、Modify Trace Appearance ダイアログを使って個別に変更することもできます。
ほとんどの場合、トレースを個別に変更する必要はありません。

デフォルトでは、コントートレースに、*zWave* とコンターレベルを示す名前（例えば：「zWave=1.5」）を自動的に付けます。
これらのトレース名は、Modify Trace Appearance ダイアログとおよび凡例で確認できます。
ほとんどの場合、デフォルトのトレース名で問題ありません。

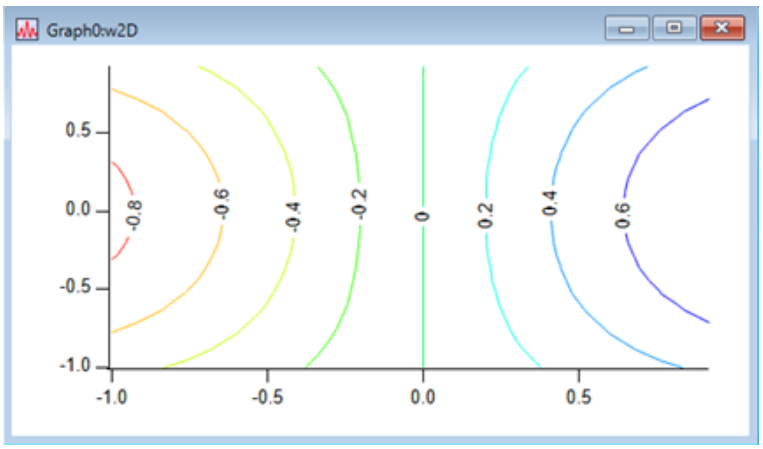
コントートレースの名前をコントロールしたい場合（凡例内の名前設定などで必要になることがあります）、/F=*formatStr* フラグを使ってください。
このフラグは、printf コマンドで説明されているようなフォーマット文字列を使います。
デフォルトのフォーマット文字列は 「%.17s=%g」 であり、これにより 「zwave=1.5」 のようなトレース名が生成されます。
formatStr には、少なくとも %f または %g（等高線を挿入するために使う）あるいは %d（等高線の 0 から始まるインデックスを挿入するために使う）を含める必要があります。
zWave 名を挿入するには、%s を含めてください。

以下に、フォーマット文字列の例をいくつか挙げます。

formatStr	結果の名前の例	フォーマット
"%g"	"100", "1e6", "-2.05e-2"	(<level>)
"z=%g"	"z=100","z=1e6", "z=-2.05e-2"	(z=<level>)
"%s %f"	"zWave 100.000000"	(<wave>, space, <level>)
"[%d]=%g"	"[0]=100", "[1]=1e6"	([<index>]=<level>)

例

```
Make/O/N=(25,25) w2D                          // 行列を作成
SetScale x -1, 1, w2D                         // 行スケーリングを設定
SetScale y -1, 1, w2D                         // 列スケーリングを設定
w2D = sin(x) * cos(y)                         // 行列に値を保存
Display; AppendMatrixContour w2D; DelayUpdate
ModifyContour w2D autoLevels={*,*,9}          // 約９段階の自動レベル
```



参照

Display、AppendToGraph、AppendXYZContour、ModifyGraph、ModifyContour、FindContour コマンド
ヘルプ Contour Plots

デモ

メニュー File→Example Experiments→Sample Graphs→Contour Demo

AppendText [/W=*winName* /N=*name* /NOCR[=*n*]] *textStr*

AppendText コマンドは、直近で作成された注釈、またはターゲット、名前付きグラフ、レイアウトウィンドウ内の名前付き注釈に、改行文字と *textStr* を追加します。
注釈には、タグ、テキストボックス、カラースケール、凡例が含まれます。

パラメーター

textStr には、フォントやフォントサイズ、その他の書式設定をコントロールするためのエスケープコードを含めることができます。
詳細については、ヘルプ Annotation Escape Codes を参照してください。

フラグ

/N=*name* 指定されたタグまたはテキストボックスに *textStr* を追加します。

/NOCR[=*n*] 先頭の改行文字の追加を省略します（これにより、複数の AppendText コマンドを使って長い行を作成できます）。/NOCR=0 は /NOCR を指定しない場合と同じ動作をし、/NOCR=1 は単に /NOCR を指定した場合と同じ動作をします。

W=*winName* 指定されたグラフ、レイアウトウィンドウ、またはサブウィンドウ内の注釈にテキストを追加します。/W オプションを指定しない場合、AppendText は最前面のグラフ、レイアウトウィンドウ、またはサブウィンドウ内の注釈にテキストを追加します。プロシージャ、マクロ、またはコマンドラインで AppendText を使用する場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する場合は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

テキストボックス、タグ、または凡例は、最大 100 行まで指定できます。
カラースケールは 1 行指定でき、この行はカラースケールの主軸ラベルとなります。

参照

Tag、TextBox、ColorScale、Legend、ReplaceText コマンド
ヘルプ Annotation Escape Codes

AppendToGizmo [*flags*] *keyword* [=*value*]

AppendToGizmo コマンドは、Gizmo オブジェクトまたは属性のコマンドを、最前面の Gizmo ウィンドウ、あるいは /N フラグで指定された Gizmo ウィンドウに追加します。

フラグ

/D 項目を表示リストの一番下に追加します。

/N=*gizmoName*

対象となる Gizmo ウィンドウの名前を指定します。/N を省略した場合、AppendToGizmo は最前面の Gizmo ウィンドウに対して処理を行います。

/Z　エラーは出力されません。

キーワード

attribute　Gizmo に対し、コマンドの残りの部分が属性に関するものであることを伝えます。
例えば：

```
AppendToGizmo attribute pointSize=1.0, name=ps0
```

alphaTestFunc={*alphaTFuncSpec*, *reference*}

アルファ値が、指定された基準値に対する選択されたテストに合格しなかったピクセルを破棄します。

alphaTFuncSpec は、以下のいずれかの値です：

0x0200：　GL_NEVER
0x0202：　GL_EQUAL
0x0203：　GL_LEQUAL
0x0204：　GL_GREATER
0x0205：　GL_NOTEQUAL
0x0206：　GL_GEQUAL
0x0207：　GL_ALWAYS

reference は [0,1] の範囲内の値です。

ambient={*R*, *G*, *B*, *A*, *face*}

マテリアルの特性を設定します。

R、*G*、*B*、*A* の値は、[0,1] の範囲にあります。

face は次のいずれかです：

0x0404：　GL_FRONT
0x0405：　GL_BACK
0x0408:　GL_FRONT_AND_BACK

axes=*type*　デフォルトの軸オブジェクトを追加します。

type は次のいずれかです：

boxAxes：　12 軸、表示領域の両側に 1 軸ずつ
tripletAxes：　原点に描かれた XYZ のトリプレット
customAxis：　ユーザーが指定した 1 つの直線軸
defaultAxes：　display list にどのタイプの軸オブジェクトも存在しない場合にのみ、boxAxes オブジェクトが追加されます。このオブジェクトは、まずオブジェクトリストに追加され、その後表示リストに追加されます。

bar3d=*bar3dWave*

3D 棒グラフのデータウェーブを指定します。

詳細については、ヘルプ 3D Bar Plots を参照してください。

blendFunction={*source*, *dest*}

ブレンドが有効になっている場合に、ブレンドコマンドを指定する属性を追加します。

source は、以下のいずれかの値です：

0x0000 : GL_ZERO
0x0001 : GL_ONE
0x0300 : GL_SRC_COLOR
0x0301 : GL_ONE_MINUS_SRC_COLOR
0x0306 : GL_DST_COLOR
0x0307 : GL_ONE_MINUS_DST_COLOR
0x0302 : GL_SRC_ALPHA
0x0303 : GL_ONE_MINUS_SRC_ALPHA
0x0304 : GL_DST_ALPHA
0x0305 : GL_ONE_MINUS_DST_ALPHA
0x8001 : GL_CONSTANT_COLOR
0x8002 : GL_ONE_MINUS_CONSTANT_COLOR
0x8003 : GL_CONSTANT_ALPHA
0x8004 : GL_ONE_MINUS_CONSTANT_ALPHA
0x0308 : GL_SRC_ALPHA_SATURATE

dest は、以下のいずれかの値です：

0x0000 : GL_ZERO
0x0001 : GL_ONE
0x0300 : GL_SRC_COLOR
0x0301 : GL_ONE_MINUS_SRC_COLOR
0x0306 : GL_DST_COLOR
0x0307 : GL_ONE_MINUS_DST_COLOR
0x0302 : GL_SRC_ALPHA
0x0303 : GL_ONE_MINUS_SRC_ALPHA
0x0304 : GL_DST_ALPHA
0x0305 : GL_ONE_MINUS_DST_ALPHA
0x8001 : GL_CONSTANT_COLOR
0x8002 : GL_ONE_MINUS_CONSTANT_COLOR
0x8003 : GL_CONSTANT_ALPHA
0x8004 : GL_ONE_MINUS_CONSTANT_ALPHA

例

```
AppendToGizmo attribute blendFunction={0x302,0x303}, name=blendFunc0
```

box={*length*, *width*, *height*}
表示ボリューム座標の ±1 の範囲内で指定された長さ、幅、高さのデフォルトのボックスオブジェクトを追加します。
name キーワードを使って、新しいオブジェクトに名前を付けます。
例えば：

```
AppendToGizmo box={1,1,1}, name=box0
```

詳細は、ヘルプ Box Objects を参照してください。

color={*R*, *G*, *B*, *A*}
[0,1] の範囲の RGBA 成分を持つ色属性を追加します。
例えば：

```
AppendToGizmo attribute color={1.0,0.0,0.0,1.0}, name=color0
```

colorScale=*name*（非推奨）
name という名前のデフォルトのカラースケールオブジェクトを追加します。

Gizmo のカラースケールは非推奨となりました。代わりに標準の注釈を使ってください。
詳細は、ヘルプ Changes to Gizmo Text を参照してください。

cylinder={*baseRadius*, *topRadius*, *height*, *slices*, *rings*}
シリンダーオブジェクトを追加します。
例えば：

```
AppendToGizmo cylinder={1,1,1,25,5}, name=cylinder0
```

詳細は、ヘルプ Cylinder Objects を参照してください。

disk={*innerRadius*, *outerRadius*, *slices*, *rings*, *startAngle*, *sweepAngle*}
ディスクオブジェクトを追加します。
例えば：

```
AppendToGizmo disk={0,1,20,20,0,360}, name=disk0
```

詳細は、ヘルプ Disk Objects を参照してください。

default3DBarChart=*srcWave*
2D ウェーブ *srcWave* に対して、デフォルトの 3D 棒グラフを追加して表示します。
例えば：

```
AppendToGizmo default3DBarChart=my2DWave
```

defaultIsoSurface=*src3DWave*
デフォルトの等値面（iso-surface）を追加して表示します。
例えば：

```
AppendToGizmo defaultIsoSurface=src3DWave
```

Gizmo は、*src3DWave* の平均値を等値面の値として設定します。
この等値面では、前面を赤色、背面を青色で表示します。

defaultIsoSurface キーワードは、Igor Pro 9.0 で追加されました。

defaultPath=*srcTripletWave*
3 列の 2D ウェーブ *srcTripletWave* に対して、デフォルトのパスプロットを追加して表示します。
例えば、次のように、幅 1 単位の黒い線とデフォルトの座標軸を使ってパスオブジェクトを描画します：

```
AppendToGizmo defaultPath=srcTripletWave
```

defaultPath キーワードは、Igor Pro 9.0 で追加されました。

defaultScatter=*srcWave*
トリプレット型の *srcWave* に対して、デフォルトの散布図を作成して表示します。
例えば：

```
AppendToGizmo defaultScatter=myTripletWave
```

defaultSurface=*srcWave*
2D ウェーブ *srcWave* のデフォルトのサーフェスプロットを追加して表示します。
例えば：

```
AppendToGizmo defaultSurface=my2DWave
```

defaultSurface キーワードは、Igor Pro 9.0 で追加されました。

defaultVolumeSlices=*src3DWave*

ボリュームデータの中心を通る X、Y、Z 方向の断面を表す 3 つのサーフェスオブジェクトを追加し、デフォルトの軸とともに表示します。
例えば：

```
AppendToGizmo defaultVolumeSlices=src3DWave
```

3 つの直交断面は、rainbow カラーテーブルを使って描画されています。

defaultVolumeSlices キーワードは、Igor Pro 9.0 で追加されました。

defaultVoxelgram=*src3DWave*

デフォルトのボクセルグラムを追加して表示します。
例えば：

```
AppendToGizmo defaultVoxelgram=src3DWave
```

Gizmo は、ボクセルグラムの値を *src3DWave* の平均値に設定します。
また、ボクセルグラムの許容誤差を *src3DWave* の分散の 5% に設定します。
src3DWave のポイント数が 50,000 を超える場合、ボクセルグラムは点マーカーを使います。
それ以外の場合は、キューブマーカーを使います。

defaultVoxelgram キーワードは、Igor Pro 9.0 で追加されました。

diffuse={*R*, *G*, *B*, *A*, *face*}

マテリアルの特性を設定します。

RGBA の値は [0,1] の範囲にあります。

face は、以下のいずれかの値です：

0x0404： GL_FRONT
0x0405： GL_BACK
0x0408： GL_FRONT_AND_BACK

emission={*R*, *G*, *B*, *A*, *face*}

マテリアルの特性を設定します。

RGBA の値は [0,1] の範囲にあります。

face は、以下のいずれかの値です：

0x0404： GL_FRONT
0x0405： GL_BACK
0x0408： GL_FRONT_AND_BACK

freeAxesCue={*xOff*, *yOff*, *zOff*, *scale*}

オフセット *xOff*、*yOff*、*zOff*（±1 の表示ボリューム座標で表される）の位置に、等方性スケーリング係数を用いて軸キューオブジェクトを追加します。
例えば：

```
AppendToGizmo freeAxesCue={0.5,0,0,1},name=fAxes0
```

group

デフォルトの空のグループオブジェクトを追加します。
例えば：

AppendToGizmo group, name=group0

詳細は、ヘルプ Group Objects を参照してください。

image=*imageWave*

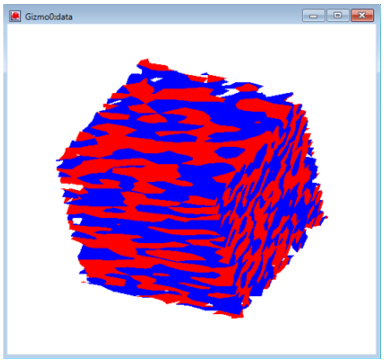
Gizmo 画像オブジェクトを追加します。

imageWave は、3D の符号なしバイトウェーブです。

詳細は、ヘルプ Gizmo Image Plots を参照してください。

isosurface=srcWave

srcWave をデータウェーブとして使って、等値面オブジェクトを追加します。
例えば：




```
Make/O/N=(10,20,30) data=enoise(10)
AppendToGizmo isosurface=data,name=iso0
```

詳細は、ヘルプ Isosurface Plots を参照してください。

matrix4x4=*matWave*

2D（4×4）の実数ウェーブ *matWave* に基づいて、matrix4x4 オブジェクトを追加します。

light=*lightType*

デフォルトのライトを追加します。

lightType はキーワードであり、方向指定型または位置指定型のいずれかです。

詳細は、ヘルプ Gizmo Lights を参照してください。

line={*x1*, *y1*, *z1*, *x2*, *y2*, *z2*}

+/1- 表示ボリューム座標で指定された 2 つの頂点の間に行オブジェクトを追加します。
例えば：

```
AppendToGizmo line={-1,-1,-1,1,1,1}, name=line0
```

lineWidth=*width*（非推奨）

線の幅をピクセル単位で指定する線幅属性を追加します。

name=*objName*

パラメーターを使って作成されるオブジェクトの名前を指定します。
例えば：

```
AppendToGizmo disk={0,1,20,20,0,360},name=disk0
```

path=*srcWave*

トリプレットウェーブ *srcWave* のデータに対応するパスオブジェクトを追加します。

詳細は、ヘルプ Path Plots を参照してください。

pieWedge=*name*

指定された名前の piewedge オブジェクトを追加します。
例えば：

```
AppendToGizmo pieWedge=pieWedge0
```

詳細は、ヘルプ Pie Wedge Objects を参照してください。

pointSize=*size*（非推奨）

ポイントのサイズをピクセル単位で設定する属性を追加します。

quad={*x1*, *y1*, *z1*, *x2*, *y2*, *z2*, *x3*, *y3*, *z3*, *x4*, *y4*, *z4*}

指定された ±1 の表示座標範囲にまたがるクワッドオブジェクトを追加します。
例えば、キューブの下部 Z 軸を埋めるクワッドを追加するには：

```
AppendToGizmo quad={-1,-1,-1,-1,1,-1,1,1,-1,1,-1,-1}, name=quad0
```

詳細は、ヘルプ Quad Objects を参照してください。

quadricDrawStyle=*drawStyle*（非推奨）

二次曲面オブジェクト（球体、円柱、円盤）の描画スタイルを決定する属性を追加します。
サポートされている描画スタイルは以下の通りです：

100010： Point - サーフェスの頂点を点で示します。
100011： Line - サーフェスのポリゴンを線として描画します。
100012： Fill - 面は塗りつぶされたポリゴンとして描画します。

quadricOrientation=*orientation*（非推奨）

100021： Inside - 法線を内側に向けて描画します。
100122： Outside - 法線を外側に向けて描画します。

quadricNormals=*normals*（非推奨）

100002： None - 法線は計算されません。
100001： Flat - ポリゴンごとに 1 つの法線が生成されます。
100000： Smooth - 頂点ごとに 1 つの法線が生成されます。

ribbon=*ribbonWave*

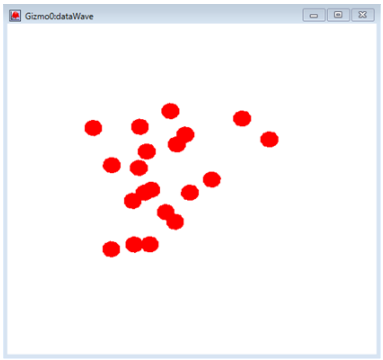
トリプレットウェーブ *ribbonWave* にデフォルトのリボンオブジェクトを追加します。

リボンオブジェクトの詳細は、ヘルプ Ribbon Plots を参照してください。

scatter=*srcWave*

3 列のトリプレットウェーブ *srcWave* をデータウェーブとして使い、散布オブジェクトを追加します。
例えば：

```
Make/N=(20,3) dataWave=gnoise(10)
AppendToGizmo scatter=dataWave,
    name=scatter0
```

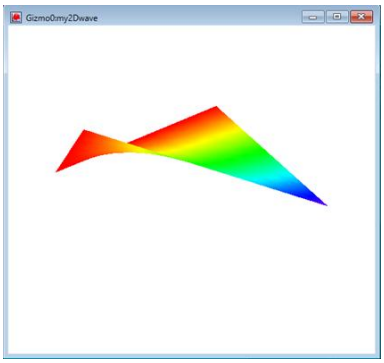


散布オブジェクトの詳細は、ヘルプ 3D Scatter Plots を参照してください。

surface=*srcWave*

2D *srcWave* を使ってサーフェスオブジェクトを追加します。
例えば：

```
Make/N=(20,30) my2Dwave=x*y
AppendToGizmo surface=my2Dwave,
    name=surface0
```



サーフェスオブジェクトの詳細は、ヘルプ Surface Plots を参照してください。

sphere={*radius*, *slices*, *stacks*}
球体オブジェクトを追加します。

radius は、±1 の表示ボリューム座標で表されます。

slices と *stacks* は、サーフェスを描画するために使われるファセットの数を決定します。
例えば：

```
AppendToGizmo sphere={1,10,15}, name=sphere0
```

詳細は、ヘルプ Sphere Objects を参照してください。

string=*strData*（非推奨）
strData は表示するテキストです。

Gizmo 文字列は非推奨となりました。代わりに標準の注釈を使ってください。詳細については、ヘルプ Changes to Gizmo Text を参照してください。

specular={*R*, *G*, *B*, *A*, *face*}
マテリアルの特性を設定します。

RGBA の値は [0,1] の範囲にあります。

face は、以下のいずれかの値です：

0x0404： GL_FRONT
0x0405： GL_BACK
0x0408： GL_FRONT_AND_BACK

strFont=*fontName*（非推奨）
文字列オブジェクトに使われるフォントを設定します。

代わりに標準の注釈を使ってください。詳細は、ヘルプ Changes to Gizmo Text を参照してください。

shininess={*value*, *face*}
マテリアルの特性を表す属性を追加します。

値は [0,128] の範囲です。
値が大きいほど、反射は鏡面反射（光沢）が強くなります。

face は、以下のいずれかの値です：

0x0404： GL_FRONT
0x0405： GL_BACK
0x0408： GL_FRONT_AND_BACK

tetrahedron=*name*
指定された名前の四面体を追加します。

詳細は、ヘルプ Tetrahedron Objects を参照してください。

triangle={*x1*, *y1*, *z1*, *x2*, *y2*, *z2*, *x3*, *y3*, *z3*}
表示ボリューム座標（±1）で指定された頂点を持つ三角形を追加します。
例えば：

```
AppendToGizmo triangle={0,0,0,1,1,1,1,-1,-1}
```

詳細は、ヘルプ Triangle Objects を参照してください。

texture=*name* 指定された名前のテクスチャオブジェクトを追加します。

詳細は、ヘルプ Texture Objects を参照してください。

voxelgram=*srcWave*

3D 実数値ウェーブ *srcWave* をデータウェーブとして使い、ボクセルグラムオブジェクトを追加します。

ボクセルグラムの詳細は、ヘルプ Voxelgram Plots を参照してください。

参照

ModifyGizmo、NewGizmo、ExportGizmo、GetGizmo、GizmoInfo、RemoveFromGizmo コマンド
ヘルプ 3D Graphics

AppendToGraph [/W=*winName* /B[=*axisName*] /C=(*r*,*g*,*b*[,*a*]) /L[=*axisName*] /NCAT/Q/R [=*axisName*] /T[=*axisName*]/VERT] *waveName* [, *waveName*] ... [*vs xwaveName*]

AppendToGraph コマンドは、指定されたウェーブをトレースとしてターゲットグラフまたは指定されたグラフに追加します。
デフォルトでは、トレースは左軸および下軸に対してプロットされます。

パラメーター

waveNames は、既存のウェーブの名前です。

vs xwaveName は、*waveNames* のデータ値を *xwaveName* のデータ値に対してプロットするよう指示します。
それ以外の場合は、*waveName* のデータ値が、*waveName* の X スケールから算出された X 値に対してプロットされます。

既存のカテゴリプロットに新しいトレースを追加する場合、*xwaveName* は、そのプロットの X 軸をコントロールしている既存の値と同じである必要があります。
既存の X 軸が「_labels_」キーワードを使って Y ウェーブの次元ラベルを利用している場合、*xwaveName* は「_labels_」に設定する必要があります。

別の X 軸を使って新しいカテゴリプロットを追加する場合、*xwaveName* には任意の適切なテキストウェーブを指定できます。また、Y 波の寸法ラベルを使用するには「_labels_」を指定することもできます。

行列の個々の行や列を含むデータのサブセットは、サブレンジ（一部）表示構文を使って指定することができます。

waveName の指定に /TN=*traceName* を追加することで、トレースに独自の名前を付けることができます。これは、同じ名前を持ちますが異なるデータフォルダーからのウェーブを表示する場合に便利です。
詳細は、ヘルプ Trace Name Parameters を参照してください。

フラグ

/B[=*axisName*] X 座標を、標準の下軸または指定された下軸に対してプロットします。

/C=(*r*,*g*,*b*[,*a*]) 追加されるトレースの色を設定します。*r*、*g*、*b*、*a* は、色と（オプションで）不透明度を RGBA 値として指定します。

/L[=*axisName*] Y 座標を、標準軸または名前付き左軸に対してプロットします。

/NCAT カテゴリプロットとして表示される図に、原因のトレースを通常通りプロットします。X 値は単なるカテゴリ番号ですが、小数を含むことができます。カテゴリ番号は 0 から始まります。これを使って、ボックスプロットの元のデータポイントの上に重ねて表示することができます。

詳細は、ヘルプ Combining Numeric and Category Traces を参照してください。

/Q　既存の軸のペアにウェーブを追加する場合、特別な高速更新モードが使われます。このモードの副作用として、追加されたウェーブは「変更なし」としてマークされます。これにより、これらのウェーブを含む他のグラフがある場合、それらが正しく更新されなくなる可能性があります。

/R[=*axisName*]　Y 座標を、標準軸または指定された右軸に対してプロットします。

/T[=*axisName*]　X 座標を、標準軸または指定された上軸に対してプロットします。

/TN=*traceName*　トレースに独自の名前を付けることができます。これは、同じ名前を持ちますが異なるデータフォルダーからのウェーブを表示する時に便利です。詳細は、ヘルプ User-defined Trace Names を参照してください。

/VERT　データを縦軸でプロットします。SwapXY（ModifyGraph）と同様ですが、トレースごとに処理されます。

/W=*winName*　指定されたグラフウィンドウまたはサブウィンドウに追加します。省略された場合、処理はアクティブなウィンドウまたはサブウィンドウに適用されます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する場合は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

参照

Display、ModifyGraph コマンド

AppendToLayout [/G=*g* /I/M/R/T/S] *objectSpec* [, *objectSpec*]...

AppendToLayout コマンドは、指定されたオブジェクトをトップレイアウトに追加します。

AppendToLayout コマンドは、ユーザー定義関数内では使用できません。
代わりに AppendLayoutObject コマンドを使ってください。

パラメーター

オプションの *objectSpec* パラメーターは、レイアウトに追加するグラフ、テーブル、テキストボックス、または PICT を指定します。
オブジェクト仕様では、オブジェクトの位置やサイズ、枠の有無、透明か不透明か、高精細表示にするかどうかも指定できます。
詳細については、Layout コマンドを参照してください。

フラグ

/G=*g*　タイル状オブジェクト間の間隔（グラウト）を指定します。/I、/M、または /R が指定されていない場合、単位はポイントになります。

/I　*objectSpec* の座標をインチ単位にします。

/M　*objectSpec* の座標をセンチメートル単位にします。

/R　*objectSpec* の座標をページの印刷領域に対するパーセンテージで指定します。

/S　Stacks オブジェクト

/T　Tiles オブジェクト

参照

Layout コマンドとユーザー定義関数で使うための AppendLayoutObject コマンド。

AppendToTable [/W=*winName*] *columnSpec* [, *columnSpec*]...

AppendToTable コマンドは、指定された列を先頭のテーブルに追加します。
columnSpec は Edit コマンドの場合と同じです。
通常、これらは単にウェーブの名前です。

参照

columnSpec については Edit コマンド、そして RemoveFromTable コマンド

AppendViolinPlot [*axis flags*][/W=*winName* /TN=*traceName* /VERT[=*doVert*] /CATL[=*doCatLabels*]] *wave*[, *wave*, ...] [*vs xWave*]

AppendViolinPlot コマンドは、ターゲットグラフまたは名前付きグラフにバイオリンプロットのトレースを追加します。
バイオリンプロット（「ビーンプロット」とも呼ばれます）は、カーネル密度推定曲線（StatsKDE コマンドを参照）を使ってデータ値の分布の概要を表示する方法です。
データ分布の概要を表示する別の方法として、ボックスプロットがあります。

AppendViolinPlot は、Igor Pro 8.0 で追加されました。

Igor Pro では、バイオリンプロットのトレースは 1 つのグラフトレースとして扱われます。
グラフからの削除やトレースの順序変更など、多くのコマンドは、バイオリンプロットのトレースにおいても他のグラフトレースと同様に機能します。

注記
DisplayViolinPlot というコマンドはありません。Display の後に AppendViolinPlot を実行してください。

パラメーター

バイオリンプロットトレース内の 1 つのバイオリンプロットのデータは、1 つの 1D ウェーブ全体、または複数列ウェーブの 1 つの列から取得されます。
トレース内のバイオリンプロットの数は、ウェーブリスト内の 1D ウェーブの数、または 1 つの複数列ウェーブの列数によって決定されます。
1D ウェーブと複数列ウェーブを混在させることはできません。

最大 100 個の個別の 1D ウェーブをリストに追加できます。
バイオリンプロット 1 つに 100 個以上のバイオリンプロットを表示したい場合は、複数列ウェーブを使うか、AddWavesToViolinPlot コマンドを使ってリストに追加する必要があります。

xWave を指定しない場合、各バイオリンプロットは数値軸上の X=0、1 などの位置に配置されます。
データが複数列のウェーブ形式の場合、デフォルトでは行列ウェーブの X 軸のスケーリングに基づいて配置されます。
数値の *xWave* を指定することで、各バイオリンプロットを X 軸上の任意の位置に配置することができます。
テキストの *xWave* を指定すると、バイオリンプロットはカテゴリ軸を使って表示されます。
複数列のウェーブに対して /CATL フラグを使うと、次元ラベルをカテゴリラベルとして使うカテゴリ X 軸が生成されます。
ヘルプ Category Plots とヘルプ Dimension Labels を参照してください。

xWave に含まれるポイント数は、1D Y ウェーブの数、あるいは複数列のウェーブにおける列の数以上でなければなりません。
ウェーブのリストが長くなりすぎてコマンドラインに入力しきれなくなる場合があるため、リストよりも長い

xWave を使うことができます。
リストを完成させるには、AddWavesToViolinPlot コマンドを使ってください。

フラグ

/L/R/B/T　　これらの軸フラグは、AppendToGraph で使われるものと同じです。

/CATL[=*doCatLabels*]
　　列次元のラベルを使って、カテゴリラベルとしてそれらを指定し、カテゴリ軸上にバイオリンプロットを作成します。*doCatLabels* は 0 または 1 であり、/CATL は /CATL=1 と同等です。

/TN=*traceName*　　トレースに独自の名前を付けることができます。これは、同じ名前を持ちますが異なるデータフォルダーからのウェーブを表示する場合に便利です。詳細はヘルプ User-defined Trace Names を参照してください。ただし、AppendViolinPlot の /TN フラグは、コマンド名の直後という通常の位置に指定します。

/VERT[=*doVert*]　　個々のバイオリンプロットを Y 軸に沿って縦方向に配置します。*doVert* は 0 または 1 であり、/VERT は /VERT=1 と同等です。

　　/VERT は ModifyGraph SwapXY と似ていますが、トレースごとに処理を行います。バイオリンプロットを横向きにするには、ModifyGraph SwapXY または AppendViolinPlot/VERT のいずれかを使ってください。

/W=*winName*　　指定されたグラフウィンドウまたはサブウィンドウに追加します。省略された場合、AppendViolinPlot はアクティブなウィンドウまたはサブウィンドウを対象とします。プロシージャ、マクロ、またはコマンドラインで使う場合、これは最初に指定されるフラグでなければなりません。

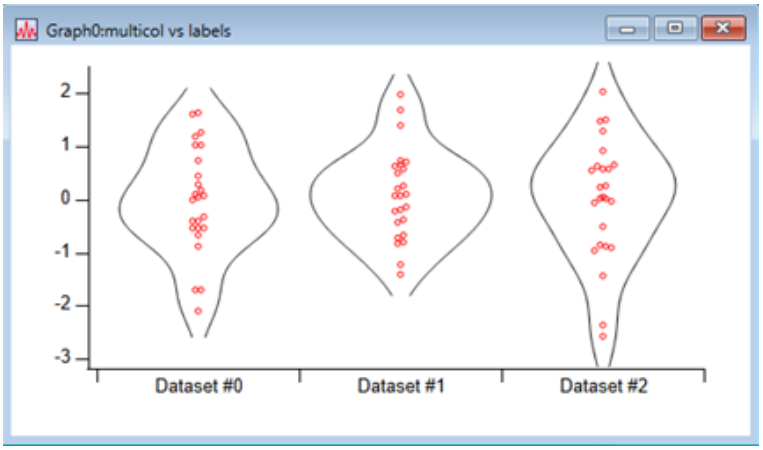
　　winName を使ってサブウィンドウを特定する場合は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

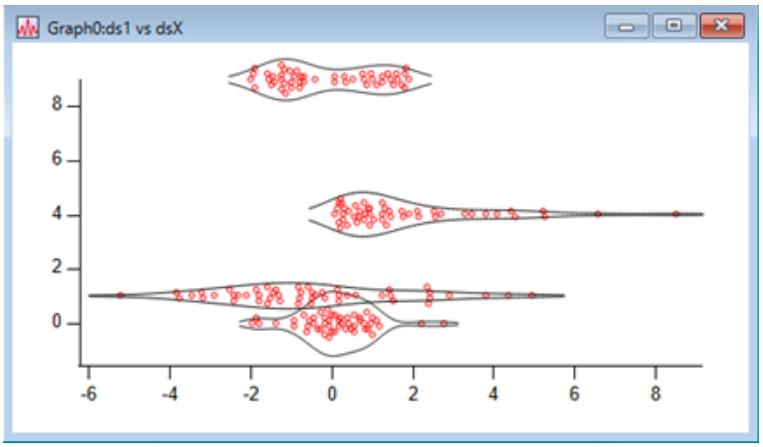
バイオリンプロットのトレースの外観の一部は、ModifyGraph を使って設定できます。
デフォルトでは、線の色は黒で、外れ値ではないデータポイントのマーカーは、通常のトレースマーカーの半分の大きさの空洞の円になります。
Modify Violin Plot ダイアログまたは ModifyViolinPlot コマンドを使うと、通常のトレースと同様に、ModifyGraph で設定されたトレースの色、線の太さ、線の破線スタイル、マーカー、マーカーのサイズを選択できます。
バイオリンプロットのトレースの各部分におけるこれらの特性の詳細なコントロールは、ModifyViolinPlot によって提供されます。

例

```
Make/O/N=(25,3) multicol            // 25 行 3 列のウェーブ
SetRandomSeed(.5)
multicol = gnoise(1)                // 正規分布に従う 3 つのデータセット
Make/O/N=3/T labels                 // カテゴリプロットを作成するためのテキストウェーブ
labels = "Dataset #"+num2str(p)     // カテゴリプロットの x 軸のラベル
Display;AppendViolinPlot multicol vs labels
```




```
Make/O/N=50 ds1, ds2, ds3, ds4
Make/O/N=4 dsX
ds1 = gnoise(1)
ds2 = gnoise(2)
ds3 = logNormalNoise(0, 1)
ds4 = enoise(2)
dsX = p^2
Display;AppendViolinPlot ds1,ds2,ds3,ds4 vs dsX
ModifyGraph swapXY=1                    // バイオリンを水平にする
ModifyGraph margin(top)=20              // 上部の余白が狭すぎる可能性があります
```



参照

Display、AppendToGraph、ModifyGraph、ModifyViolinPlot コマンド
ヘルプ Box Plots、Violin Plots

AppendXYZContour [axisFlags][/W=winName /F=formatStr] zWave [vs {xWave, yWave}]

AppendXYZContour コマンドは、自動スケーリングされたコンターレベルを持ち、Rainbow カラーテーブルを使う XYZ トリプルで構成される 2D ウェーブのコンターを、ターゲットまたは指定されたグラフに追加します。

Z 値の行列にコンターを描画するには、AppendMatrixContour を使います。

注記
DisplayContour というコマンドはありません。Display の後に AppendXYZContour を実行してください。

xWave と *yWave* の仕様を指定する場合、*xWave* は行の X 値、*yWave* は列の Y 値、*zWave* は Z 値を提供し、これら 3 つのウェーブはすべて 1D でなければなりません。

xWave と *yWave* の指定を省略する場合、*zWave* は 4 行以上かつ 3 列以上の 2D ウェーブでなければなりません。
1 列目は X、2 列目は Y、3 列目は Z となります。
それ以降の列は無視されます。

行内の X、Y、Z のいずれかが空白（NaN）の場合、その行は無視されます。

マクロ内で、デフォルト値に基づいてコンターが計算、表示される前にコンターのレベルの外観を変更するには、「;DelayUpdate」を追加し、AppendXYZContour コマンドの直後に適切な ModifyContour コマンドを記述します。
最後の ModifyContour コマンドを除くすべてのコマンドにも、「;DelayUpdate」を追加する必要があります。
関数内では DelayUpdate は必要ありませんが、DoUpdate を使うと、すべての関数が完了するまで待機するというデフォルトの動作ではなく、コンタートレースを直ちに生成させることができます。

コマンドラインでは、Display コマンド、それに続く AppendXYZContour コマンド、ModifyContour コマンドを、セミコロンで区切ってすべて 1 行に入力することができます。
例えば：

```
Display; AppendXYZContour zWave; ModifyContour ...
```

フラグ

axisFlags　フラグ /L/R/B/T は、AppendToGraph で使われるものと同じです。

/F=*formatStr*　コンタートレースに割り当てられる名前を指定します。これは AppendMatrixContour の場合と同じです。

/W=winName　指定されたグラフウィンドウまたはサブウィンドウに追加します。省略された場合、処理はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、これは最初に指定されるフラグでなければなりません。

winName を使ってサブウィンドウを特定する場合は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

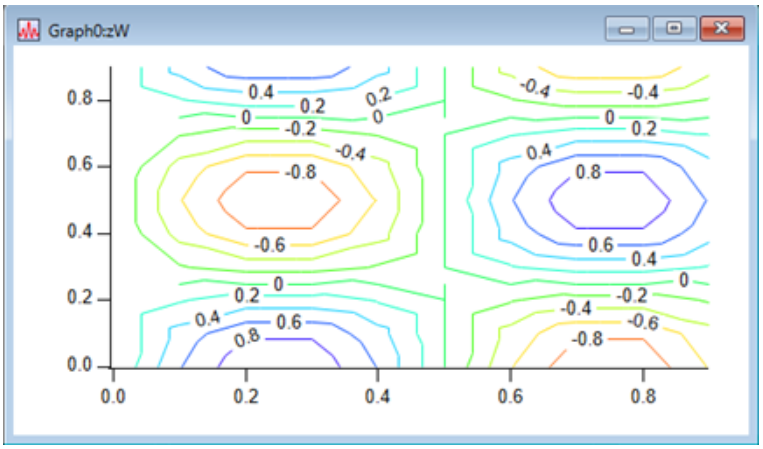
詳細

AppendXYZContour は、コンタートレースを作成して表示します。
これらは、Modify Contour Appearance ダイアログを使ってグループとして変更することも、Modify Trace Appearance ダイアログを使って個別に変更することもできます。
ほとんどの場合、トレースを個別に変更する必要はありません。

コンターレベルの名前の付け方については、AppendMatrixContour コマンドを参照してください。

例

```
Make/O/N=(100) xW, yW, zW                    // X, Y, Z ウェーブを作成
xW = sawtooth(2*PI*p/10)                     // X 値を生成
yW = trunc(p/10)/10                          // Y 値を生成
zW = sin(2*PI*xW)*cos(2*PI*yW)               // Z 値を生成
Display; AppendXYZContour zW vs {xW, yW}
ModifyContour zW autoLevels={*,*,9}          // およそ 9 つの自動レベル
```



参照

Display、AppendToGraph、ModifyGraph、ModifyContour、AppendMatrixContour コマンド
ヘルプ Contour Plots