CONTENTS

ビジュアルヘルプ – WaveMetrics 提供のプロシージャ

このファイルでは、業務で役立つ可能性のある WaveMetrics 提供のプロシージャファイルについて説明します。WaveMetrics Procedures フォルダー内のファイルについて、ここで説明します。

インクルードするプロシージャは、メニューやダイアログが追加するもの、コマンドライン等で呼び出すものがあります。

Igor Pro のデフォルト機能だけで判断するのではなく、「こんな機能はついていないのかな？」と思ったら、ぜひ、下記のリストを確認してみてください。
欲しい機能が見つかるかもしれません。

役立つ WM（WaveMetrics）プロシージャの探し方

まず、#include 構文を使ってプロシージャファイルにアクセスする方法について説明します。

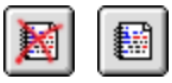
Windows のエクスプローラーを使ってフォルダー構造を見て、ファイルをダブルクリックしてその内容を確認し、その後、そのプロシージャファイルを開いて使うこともできますが、以下の理由から #include メソッドを使うことが推奨されます。

1. 今後、WaveMetrics プロシージャフォルダーの構造が変更されたとしても、コードは引き続き正常に動作します。
2. ファイルは読み取り専用で開かれるため、誤って変更してしまうことを防げます。

ヘルプファイル（WM Procedures Index.ipf）を使うとよい理由

単にファイルを閲覧するのではなく、WM Procedures Index.ipf ファイルを使う利点は以下の通りです。

- Edit メニューから Find を選択してキーワードを検索できます。
- #include <procedureFileName> というテキストをメインの Procedure Window に挿入するには、#include 文の左側にあるプロシージャアイコンのボタンをクリックします。

   #include <procedureFileName>

-  をクリックすると、現在のエクスペリメントからそのファイルを除外できます。
- #include <procedureFileName> というテキストをコピーし、任意の Procedure ウィンドウに貼り付けることで、#include メソッドを使ってファイルを開くことができます。

現在のエクスペリメントで #include ファイルを使う

現在のエクスペリメントに含めたい項目を見つけたら、#include ステートメントの左側にあるプロシージャアイコンのボタンをクリックしてください。

 #include <procedureFileName>

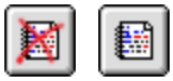
現在のエクスペリメントから含まれているファイルを削除するには、赤い「×」のマークが付いたボタンをクリックしてください。

これらのボタンは、INSERTINCLUDE と DELETEINCLUDE コマンドを実行します。
詳細については、ヘルプ Including a Procedure File、Operation Queue、NotebookAction を参照してください。

別のエクスペリメントで #include ファイルを使う

別のエクスペリメントに組み込みたい内容を見つけたら、#include で始まるテキストをコピーし、もう一方のエクスペリメントを開いて、そのテキストを Procedure ウィンドウに貼り付けてください。
Procedure ウィンドウを閉じたり、コンパイルボタンをクリックしたりすると、Igor がそのファイルを開き、そのルーチンを使用可能にします。

例えば、WM Procedures Index.ipf ファイルに次のような記述があったとしましょう。

 #include <Split Axis>

これは次のことを行うためのプロシージャが含まれています。

現在のエクスペリメントとは異なるエクスペリメントのプロシージャにアクセスするには、以下の手順に従ってください。

1. 「#include <Split Axis>」と書かれたテキストを選択します。
2. これをクリップボードにコピーします。
3. 別のエクスペリメントを開きます。
4. Windows メニューから Procedure Windows→Procedure Window を選択します。
5. コピーした行を Procedure ウィンドウの上部付近に貼り付けてください。
6. Procedure ウィンドウを閉じるか、下部にある Compile ボタンをクリックしてください。

これで、これらのプロシージャが利用可能になります。
多くは Macros メニューからアクセスできます。

注記： プロシージャウィンドウによっては、Macros メニュー以外のメニューにプロシージャ項目が表示される場合があります。

場合によっては、ルーチンを一時的にだけ必要とし、余計な Procedure ウィンドウが画面に残るのを避けたいこともあるでしょう。
ルーチンの使用が終わったら、#include 行の先頭にコメント記号「//」を挿入するか、その行を完全に削除してください。
これにより、次回のコンパイル時にそのインクルードファイルは読み込まれなくなります。

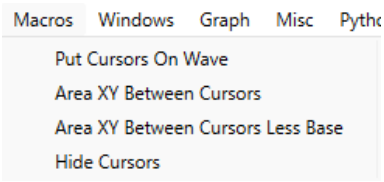
（PDF ファイルにはプロシージャをインクルードする機能はありませんので、ボタンは省略してあります）

WaveMetrics 分析プロシージャ

#include <AreaXYBetweenCursors>

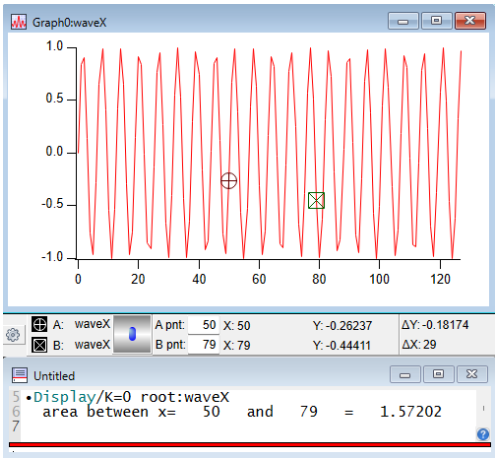
AreaXY 関数が含まれます。
<AreaXY>と同様ですが、カーソルを使って、面積を計算する X 範囲を指定できる点が異なります。

インクルードすると、Macros メニューに項目が追加されます。



Macros→Put Cursors On Wave を選択すると、グラフ上にカーソルが配置できるようになります。

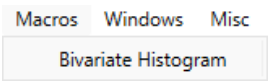
Area XY Between Cursors を選択すると、履歴エリアにカーソル間の面積が表示されます。



#include <Bivariate Histogram>

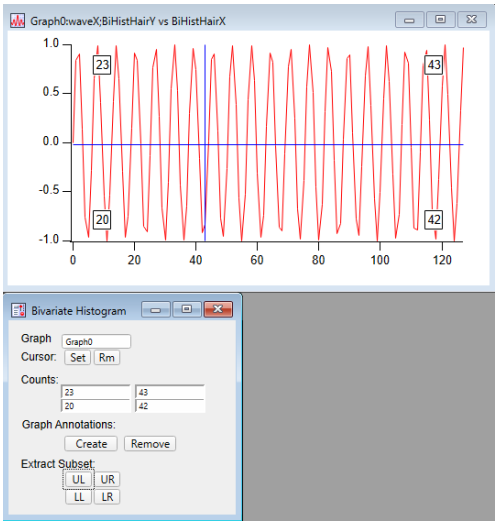
２変量ヒストグラムコントロールパネルを作成するためのプロシージャが含まれています。
グラフ上に十字カーソルを表示し、その十字線で定義される各象限に含まれるデータポイントの数を報告します。
また、特定の象限を新しいデータセットとして抽出することも可能です。
フローサイトメトリーの結果を分析することを目的としていますが、他の分野でも十分に役立つ可能性があります。
通常軸および対数軸のいずれでも動作します。

インクルードすると、Macros メニューに項目が追加されます。



グラフを最前面にした状態で Macros→Bivariate Histogram を選択すると、グラフ上にライン状のカーソルが表示され、移動できるようになります。

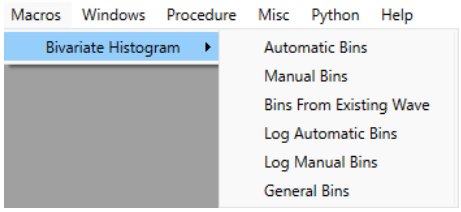
Graph には、グラフウインドウの名前を入力します。
Counts には各エリアのポイント数があり、Graph Annotation を作成すると、各象限に表示されます。
Extract Subset ボタンをクリックすると、そこに含まれるポイントを抽出したウェーブが作成されます。



#include <Bivariate Histogram 2>

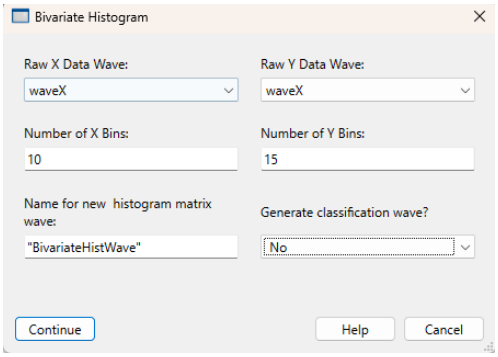
２つの 1D ウェーブが与えられた場合に、行列ウェーブ内で２変量ヒストグラムを作成するためのプロシージャが含まれています。
行列の各セルは、X と Y の各ビン内の値の個数を表します。
正規の等間隔ビン、対数ビン、または補助ウェーブで定義される可変幅のビンを扱います。

使用方法については、プロシージャファイル内（WaveMetrics Procedures→Analysis→Bivariate Histrogram 2.ipf）のコメントを参照してください。



インクルードすると、Macros メニューに項目が追加されます。

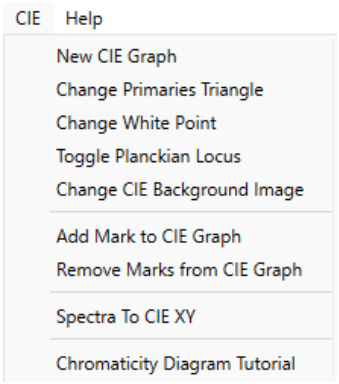
選択したメニュー項目に従って、ビン数などの各種の設定ダイアログが表示されます。



Name for new histogram matrix wave には結果を格納するウェーブ名を指定します。

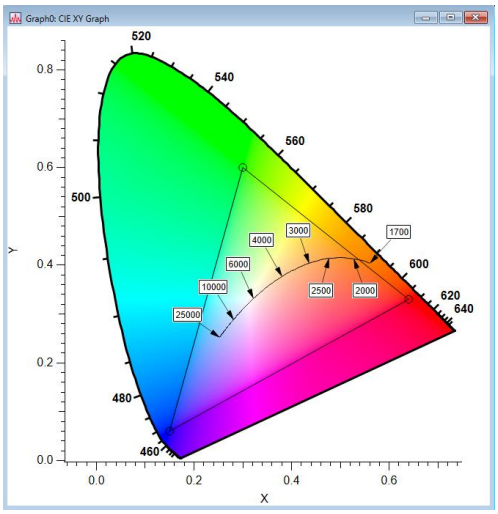
#include <CIE Chromaticity>

メインメニューに CIE メニューを追加します。
このメニューには、CIE XY 色度図を作成する項目や、色度関数に基づいて輝度対波長（nm 単位）から CIE XY 色を算出する項目が含まれます。
また、色度と CIE 色体系に関するチュートリアルへのリンクも用意されています。



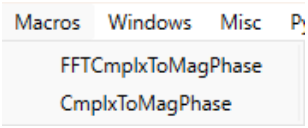
New CIE Graph を選択して、CIE グラフを表示し、その後、各メニュー項目で色の調整を行います。

右の画面は Toggle Planckian Locus メニューを選択したときの画面です。

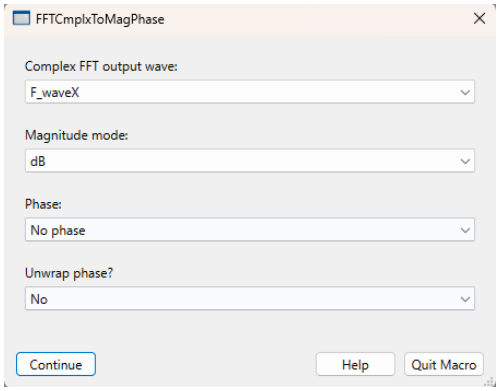


#include <CmplxToMagPhase>

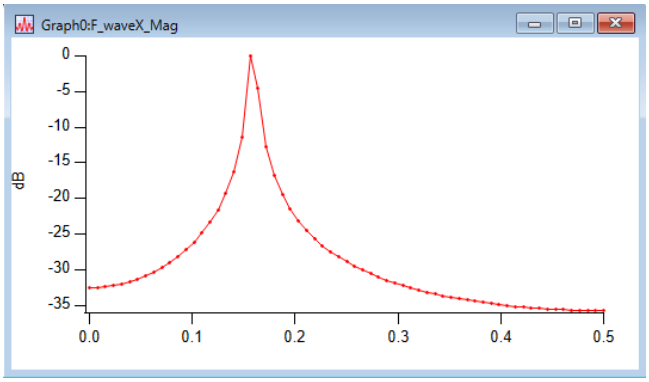
（おそらく FFT の結果である）複素数ウェーブを、振幅ウェーブと位相ウェーブに分解します。



Macros→FFTCmplxToMagPhase を選択すると、振幅と位相に関する設定画面が表示されます。

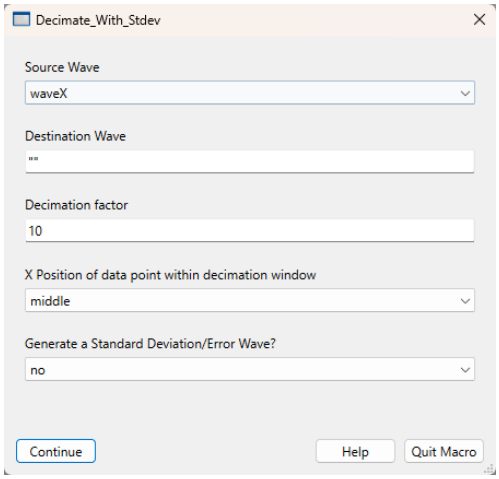


Continue をクリックすると、例えば、次のようなグラフが表示されます。

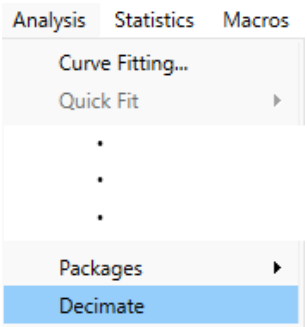


#include <Decimation>

入力ウェーブの n ポイントごとに平均化し、出力ウェーブを 1 ポイントに集約する Decimate プロシージャが含まれています。



Analysis メニューに Decimate が追加されます。
選択すると Decimation_With_Stdev ダイアログが開きます。



#include <DFTMagPhase>

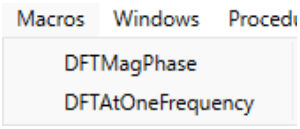
FTMagPhase プロシージャと同様ですが、計算には処理速度の遅い離散フーリエ変換が使われる点が異なります。

ユーザーが選択可能な周波数の開始値と終了値

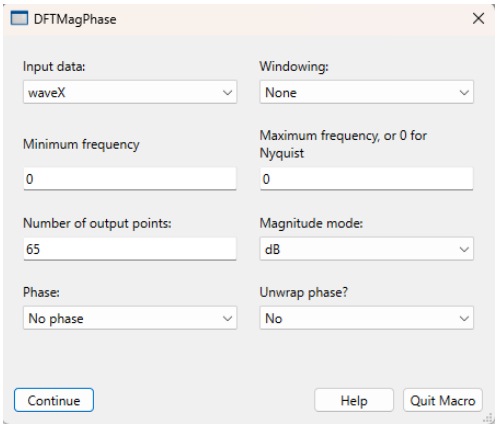
ユーザーが選択可能な周波数帯の数

また、ユーザーが指定した単一の周波数における振幅と位相を計算する DFTAtOneFrequency プロシージャも含まれています。

インクルードすると、Macros メニューに項目が追加されます。



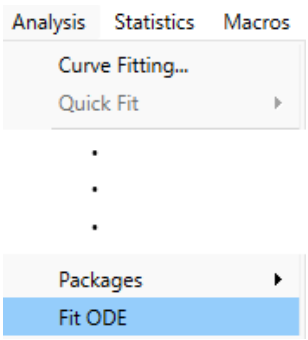
Macros→DFTToMagPhase を選択すると、周波数、振幅と位相に関する設定画面が表示されます。

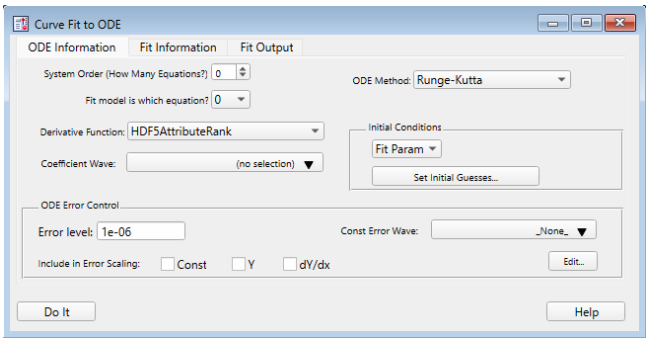


`#include <FitODE>`

微分方程式の解に対するカーブフィッティングを行うためのコントロールパネルを作成します。

Analysis メニューに Fit ODE が追加されます。
選択すると Curve Fit to ODE ダイアログが開きます。

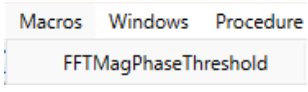




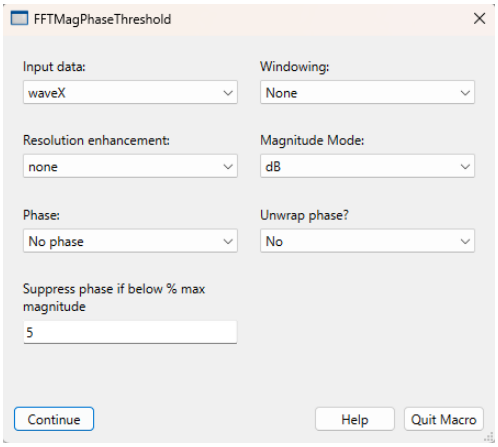
`#include <FTMagPhaseThreshold>`

FTMagPhase プロシージャと同様ですが、以下の機能が追加されています。

・　　振幅の小さい信号の位相値は無視する



Macros→FFTMagPhaseThreshold を選択すると、振幅と位相に関する設定画面が表示されます。

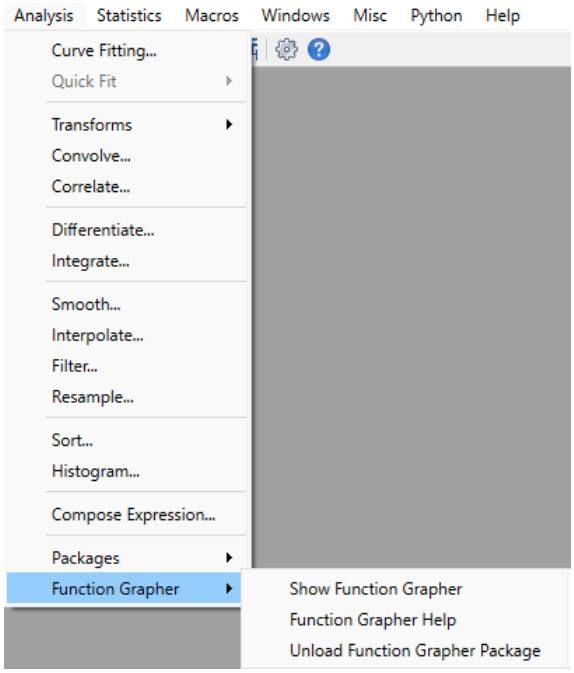


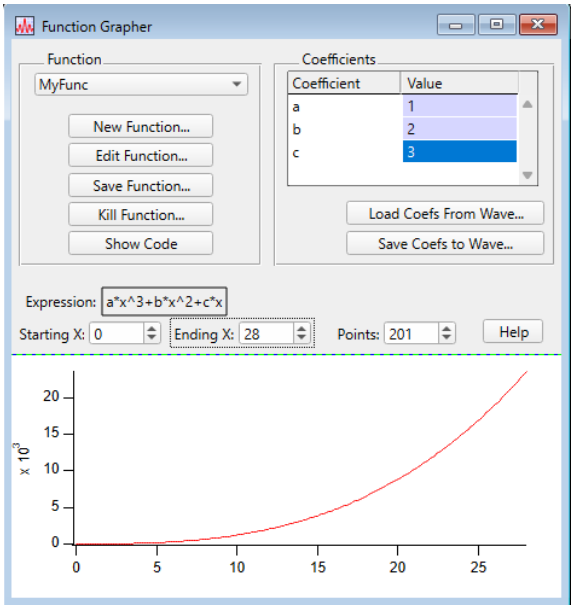
#include <Function Grapher>

Function Grapher は、数式を簡単にグラフ化できるように設計されたプロシージャパッケージです。
このパッケージは、グラフと各種コントロールで構成されており、式内の定数の変更、独立変数の範囲、グラフ表示に使うポイントの密度などが、どのように影響するかを手軽に確認することができます。
Function Grapher は、指定された式に基づいて、その式を実装する Igor ユーザー定義関数を作成します。

Analysis メニューに Function Grapher が追加されます。
Show Function Grapher を選択すると Function Grapher ダイアログが開きます。

詳細はメニューから Function Grapher Help を選択してヘルプを参照してください。





#include <fuzzyClasses>

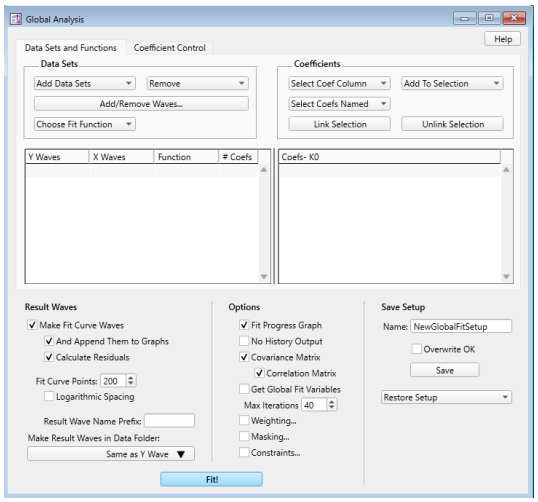
このプロシージャは、ファジー論理を用いて inWave 内の 2 つのクラスを特定します。
その結果である両クラスの平均値と分散は、変数 Var1、Var2、mean1、mean2 に格納され、必要に応じて履歴エリアに出力されます。
また、このプロシージャは 2 つのメンバーシップウェーブを生成し、それらを現在のデータフォルダーに保存します。
各ウェーブの各要素は、inWave 内の対応する要素が、それぞれ第 1 クラスまたは第 2 クラスに属する確率を表しています。

このプロシージャにはメニュー項目はありません。
例えば、コマンドラインで次のように実行します。

```
fuzzyClasses(waveX,0)
```

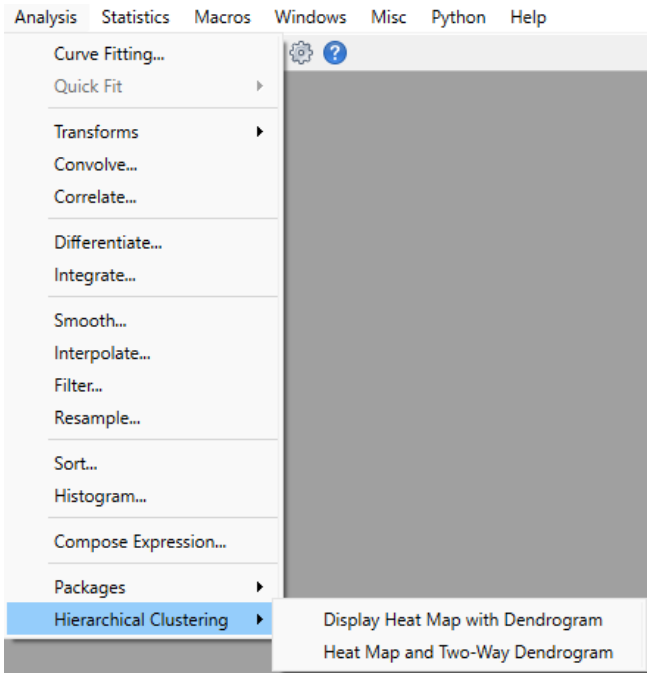
#include <Global Fit 2>

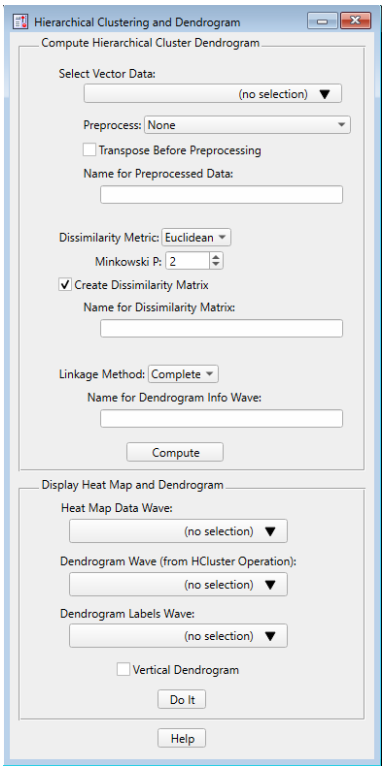
Global Fit と同様ですが、より柔軟性があります。
複数のデータセットをフィッティングする時に複数の関数の使用をサポートし、フィッティング係数の任意の連動設定が可能です。
このパッケージは、Analysis→Packages→Global Fit を選択しても読み込むことができます。



#include <HeatMap_and_Dendrogram>

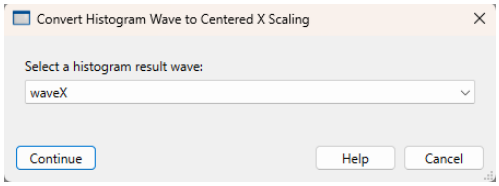
HCluster の GUI です。
HCluster の結果から、階層的クラスタリングのデンドログラムを含むヒートマップを作成します。

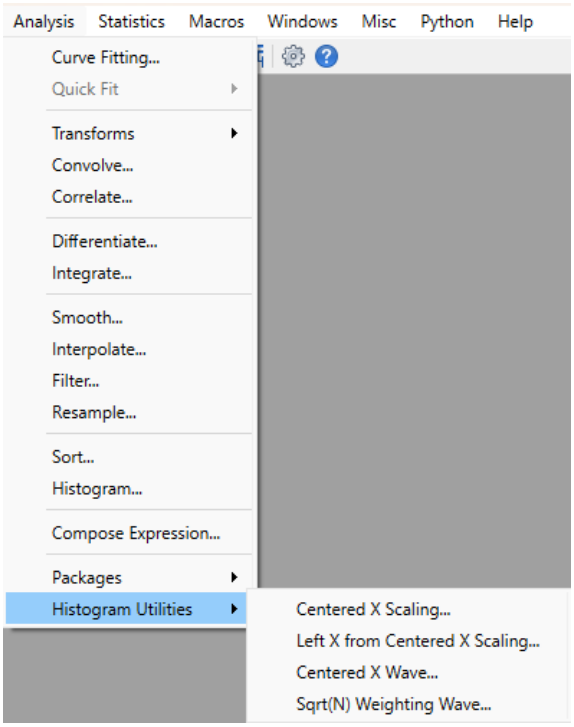




#include <HistogramUtilities>

Histogram コマンドを使う時に発生する、X 値が中央にない問題に対処するためのプロシージャが含まれています。
また、ヒストグラムへのカーブフィッティングを行う時に、適切な重み付けウェーブを作成するためのユーティリティも提供しています。
詳細については、プロシージャファイルを参照してください。

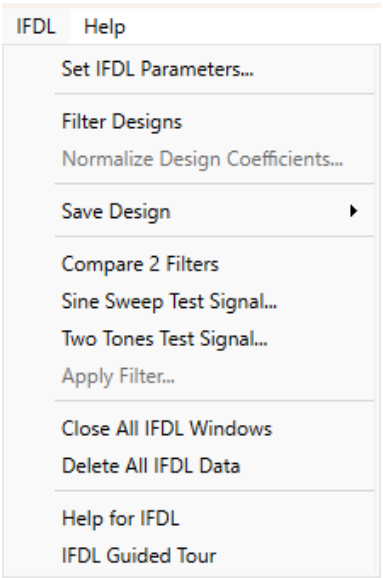




#include ":IFDL v4 Procedures:IFDL"

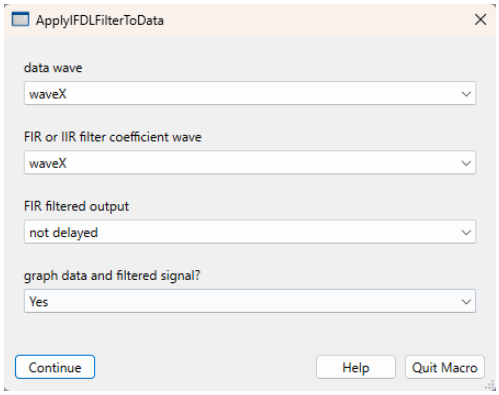
Igor Filter Design Laboratory（IFDL）を使うと、ローパス、ハイパス、バンドパス、ノッチ、微分、ヒルベルト、ウィンドウ、および任意の有限インパルス応答（FIR）フィルター、ならびに「アナログプロトタイプ」の無限インパルス応答（IIR）フィルターを設計することができます。

インクルードすると、メインメニューに IFDL メニューが追加されます。



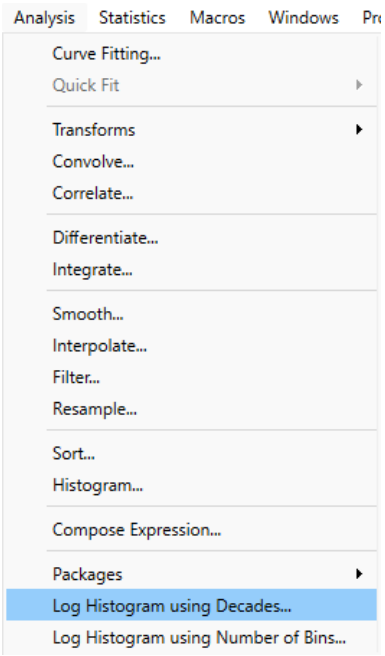
IFDL Guided Tour で基本的な操作を説明しています。
詳細は Help for IFDL を参照してください。

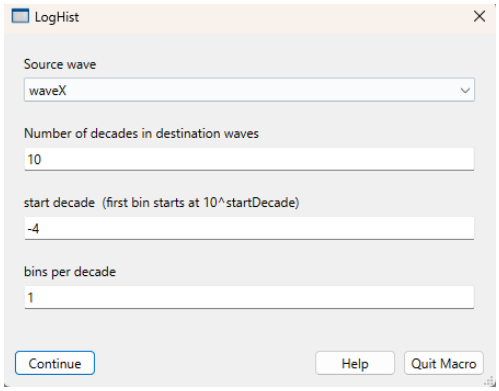
また、Analysis メニューに Apply IFDL Filter To Data メニューアイテムが追加されます。

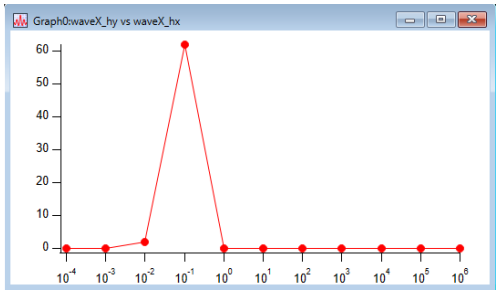


#include <Log Histogram>

ソースウェーブの対数ヒストグラムを表す XY ウェーブのペアを作成します（つまり、ビンが対数配列で配置されます）。







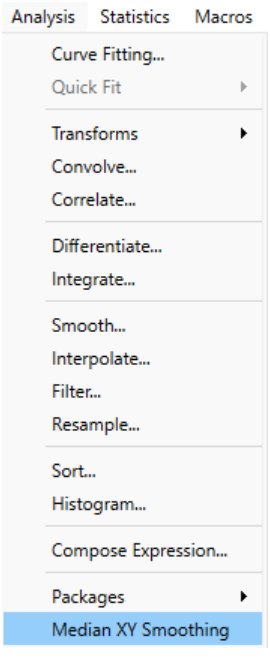
#include <Median>

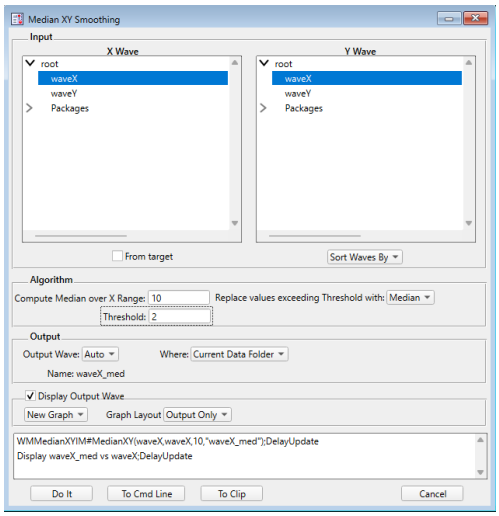
ウェーブの中央値を返す Median、MedianEO、MedianXYSmoothing 関数が含まれています。
このインクルード文の代わりに #include <Median XY Smoothing Dialog> を使ってください。

#include <Median XY Smoothing Dialog>

#include <Median> の、Median、MedianEO、MedianXYSmoothing 関数が含まれています。

Analysis→Packages メニューに Median XY Smoothing を追加します。これにより、XY データに対して中央値平滑化を計算するためのダイアログ形式のパネルが表示されます。
#include <Median> に含まれる旧式の Median XY Smoothing マクロよりも推奨されます。





#include <Multipeak Fitting>

Gaussian、Lorenzian、Voigt、指数修正 Gaussian など、さまざまなピーク形状に対応したパッケージです。

詳細については、「Multipeak Fit Demo.pxp」エクスペリメントを参照してください。

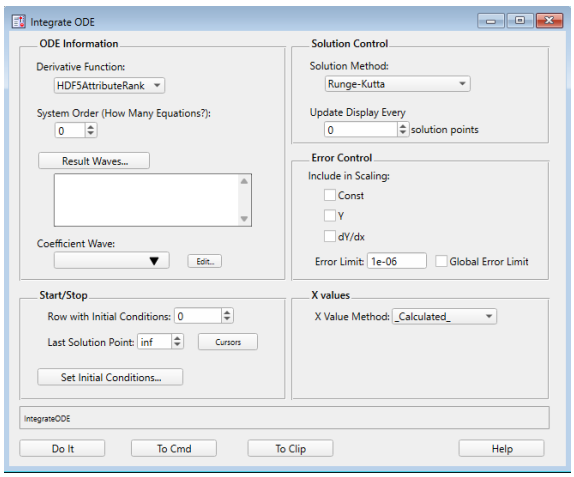
#include <Manual Peak Adjust>

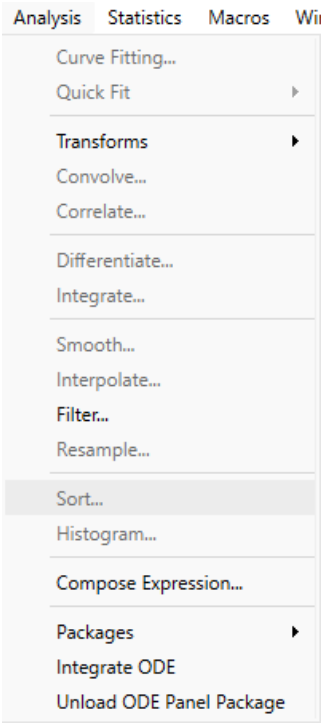
主に「Multipeak Fitting」での使用を想定していますが、独自のエクスペリメントで利用できるルーチンが含まれている場合があります。
ルーチンの使用方法については、「Multipeak Fit Demo」のエクスペリメント例を参照してください。
ユーザーがガウス型のピークをインタラクティブにドラッグして形状を変更できるようにします。
（このプロシージャをインクルードしてもメニューに変化はありません）

#include <ODE Panel>

微分方程式の解を求めるためのコントロールパネルを作成します。
IntegrateODE コマンド用のインターフェイスです。

インクルードすると、Analysis メニューに Integrate ODE メニューが追加されます。





#include <Peak AutoFind>

主に「Multipeak Fitting」での使用を目的としていますが、独自のエクスペリメントで利用できるルーチンが含まれている場合があります。
ルーチンの使用方法については、「Multipeak Fit Demo」のエクスペリメント例を参照してください。
データを自動的に解析してピーク数を推定し、ガウス型パラメーターの適切な推定値を提示するルーチンが含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Peak Functions>

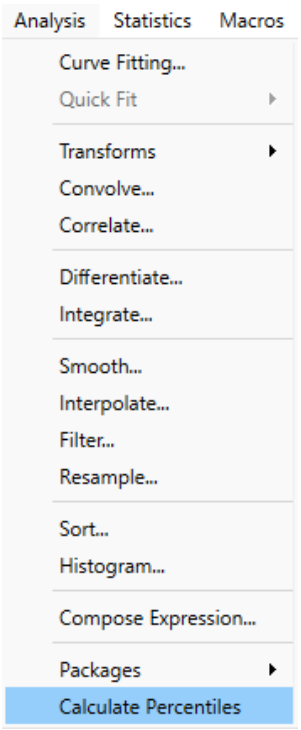
Gaussian、Lorentzian、Voigt の各フィッティング関数が含まれています。
これらは、３つの XOP モジュールによって提供される外部関数（XFUNC）のユーザー定義バージョンで

す。
詳細については、「Multipeak Fit Demo」のエクスペリメントを参照してください。

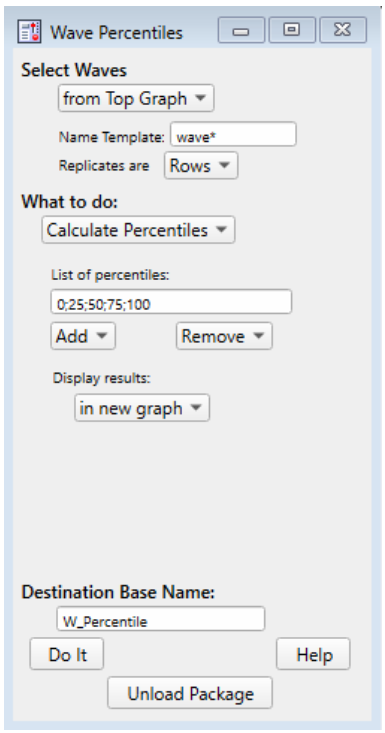
#include <Percentile and Box Plot>

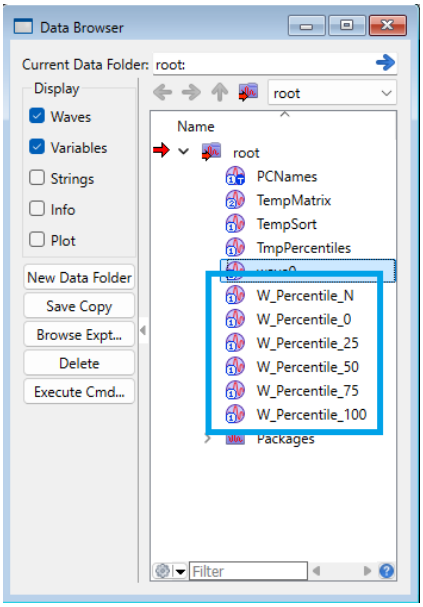
一連のウェーブデータまたは複数列のウェーブデータについて、各ポイントごとのパーセンタイルを計算する機能や、独自のウェーブデータを用いてボックスプロットを作成する機能が含まれています。
パーセンタイルおよびボックスプロットの機能にグラフィカルインターフェースを提供するコントロールパネルが含まれています。

インクルードすると、Analysis メニューに Calculate Percentile メニューが追加されます。



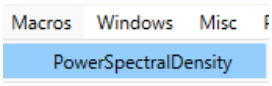
最前面にグラフを表示して、パネルを設定して、Do It をクリックします。





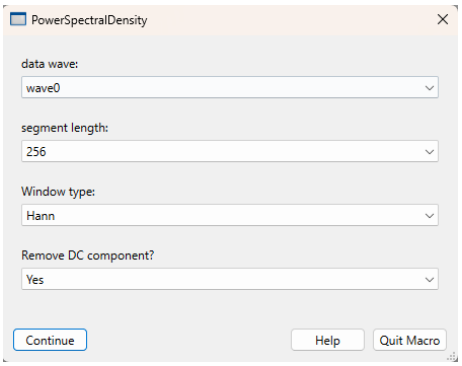
#include <Power Spectral Density>

長い入力データウェーブを受け取り、パワースペクトル密度関数を計算する PowerSpectralDensity マクロが含まれています。



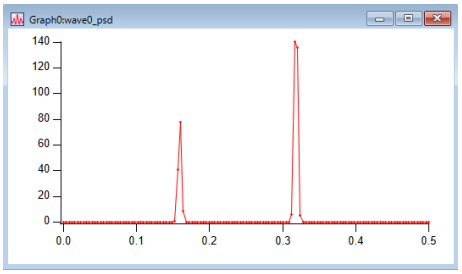
インクルードすると、Macros メニューに PowerSpectralDensity メニューが追加されます。
このプロシージャには、以下の特徴があります。

- 結果の自動表示
- 元のデータは変更されない
- ウィンドウ関数のポップアップリスト
- ユーザーが設定可能なセグメント長



下位互換性を確保するため、古い（非推奨の）PSD プロシージャが含まれています。

パワースペクトル密度の計算方法の詳細については、PSD Demo エクスペリメントを参照してください。




```
#include <Smooth Wave With Blanks>
```

SmoothWaveWithBlanks プロシージャが含まれています。
これにより、欠損値（NaN）を含むデータに対してガウス平滑化を行うことができます。
このプロシージャは、欠損値の間のデータを独立した区間として扱います。
（このプロシージャをインクルードしても GUI 上に変化はありません）

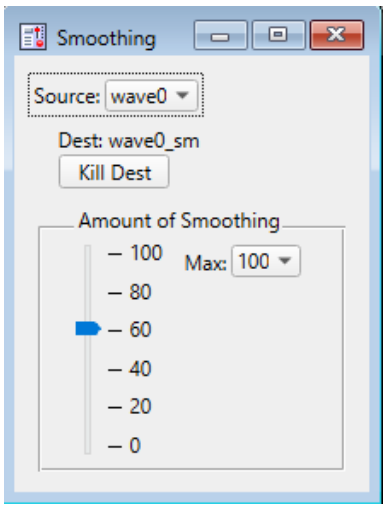
```
#include <Smoothing Control Panel>
```

SmoothingControlPanel プロシージャが含まれています。
これは、Smooth Wave With Blanks プロシージャファイルのインターフェイスとして機能するコントロールパネルを提供します。
（#include <Smooth Wave With Blanks> 文を含みます）

このファイルは、グラフ内の個々のトレースに対して操作を行うコントロールパネルの作成方法を示しています。
これは、独自のプロシージャファイルを作成する時の出発点として活用できます。

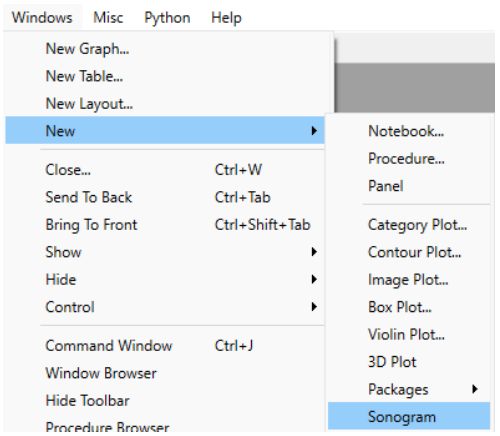
平滑化を行うグラフを最前面に生じして、コマンドウィンドウで次を実行するとパネルが表示されます。

```
SmoothingControlPanel()
```

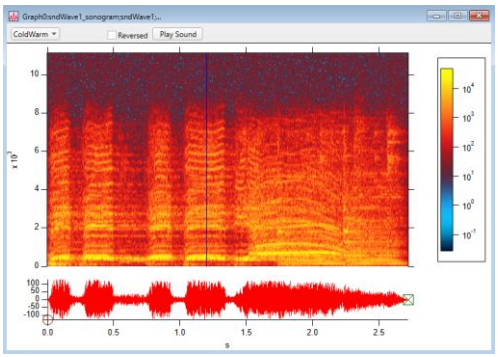



```
#include <Sonogram>
```

Windows→New サブメニューに Sonogram 項目が追加されます。
Sonogram は、Hanning または Gabor の方法を使って、経過時間に対する短時間周波数スペクトルを表示する画像を生成します。



詳細については、Sonogram Demo エクスペリメントを参照してください（File→Example Experiments→Movies & Audio→Sonogram Demo）。



#include <Sum of Wave>

テクニカルノート「TN018 Area & Integration」に含まれる 3 つの関数、PSumWave、XSumWave、SumWave を含んでいます。
これらの関数は、ウェーブ内のデータ値の合計を返すもので、範囲の指定方法だけが異なります。
組み込み関数である sum 関数も参照してください。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Varimax>

指定されたイプシロンを条件として、ウェーブに対してバリマックス回転を行うプロシージャが含まれています。
このアルゴリズムは、Henry F. Kaiser による 1959 年の論文に基づいており、正規化を行った後、 2 つのベクトルを同時に回転させる手法を採用しています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

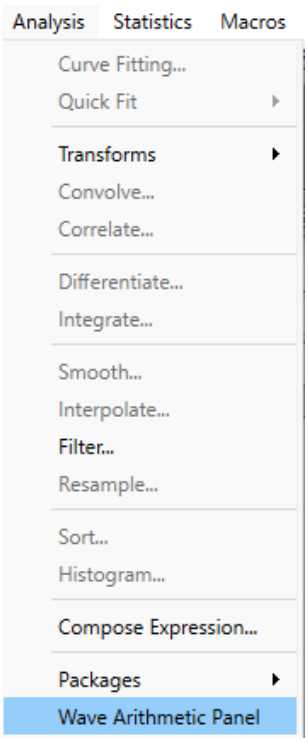
#include <Walsh1D>

入力ウェーブに対して 1D Walsh 変換を行う関数 doWalsh1DTransform(inWave) が含まれています。
ウェーブのデータポイントは 2 のべきでなければなりません。
変換結果は、現在のデータフォルダー内のウェーブ W_WalshTransform に保存されます。
（このプロシージャをインクルードしても GUI 上に変化はありません）

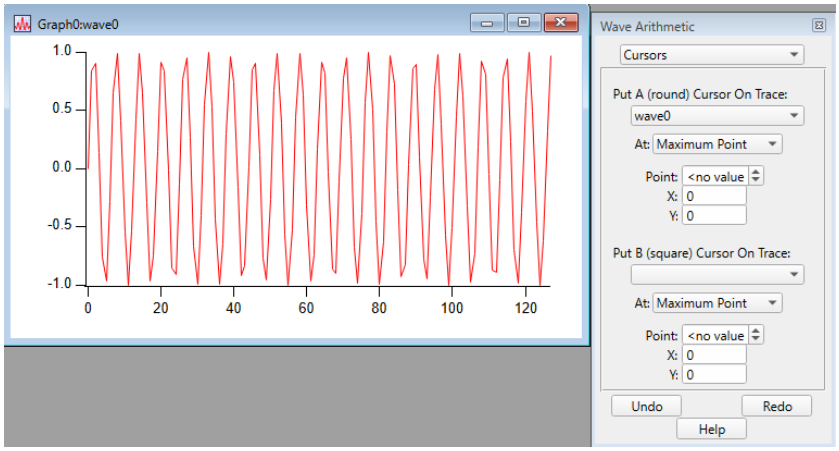
#include <Wave Arithmetic Panel>

さまざまなウェーブ演算を GUI で操作できるようにするコントロールパネルを作成するためのプロシージャが含まれています。
グラフカーソルを使ってウェーブを特定し、ポイント＆クリック操作で、ウェーブの加算、減算、乗算、定数による除算、あるいはあるウェーブと別のウェーブの除算を行うことができます。
また、選択したポイントが値 1.0 になるようにウェーブを正規化したり、値を差し引いて一定のベースラインを除去したりすることも可能です。

すべての演算はウェーブフォームデータと XY データの両方で実行可能です。
また、X 軸の範囲が重複しているが同一ではないデータにも対応しています。

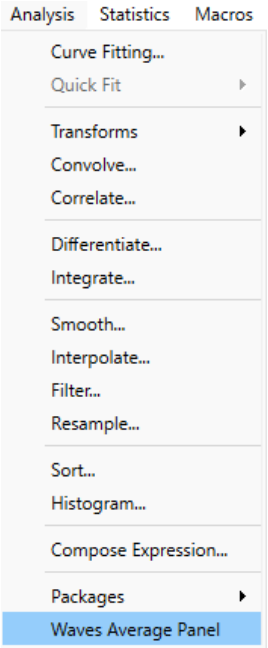


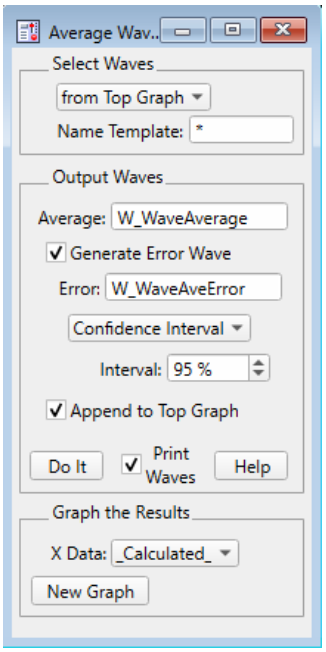
グラフ上のトレースに対して簡単な操作を行ったり、グラフ上にカーソルを配置したりするための（非常に）シンプルなパネルも備えています。
グラフに表示されているデータに対してのみ自動スケーリングが適用されているかのように、グラフの縦軸スケールを設定するボタンも含まれています。




```
#include <Waves Average>
```

Igor に fWaveAverage 関数を追加します。
fWaveAverage は、ウェーブリストをポイントごとに平均化し、平均値を含む新しいウェーブを作成します。
オプションで、データの標準偏差、平均の標準誤差、または平均の信頼区間を含むことができる第二のウェーブ（誤差ウェーブ）を作成することもできます。
使いやすいインターフェイスを提供するコントロールパネルを作成します。
このコントロールパネルを使うと、テーブルの選択、グラフウィンドウ、または単純な名前テンプレートからウェーブのリストを生成できます。



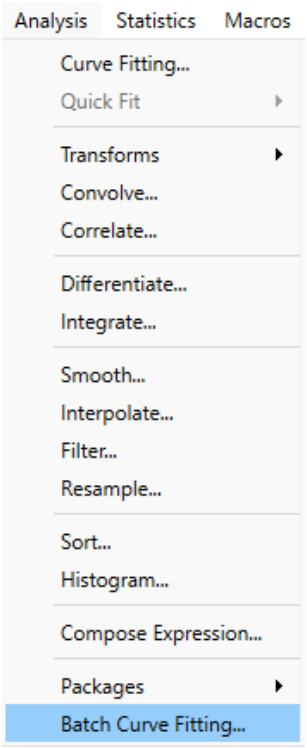


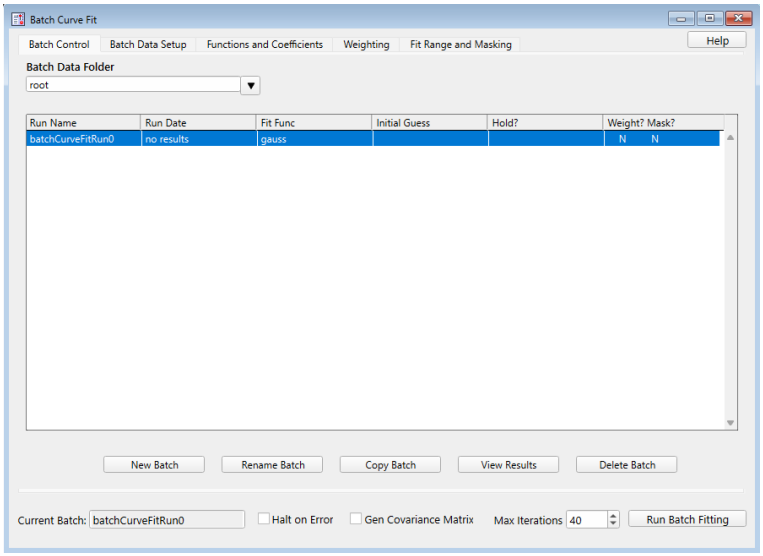

```
#include <WM_MDFitFunctions> // 廃止
```

これらはすべて、現在組み込みのフィッティング関数となっている多次元フィッティング関数のセレクションです。

#include <WMBatchCurveFit>

Batch Curve Fitting プロシージャを使うと、複数のデータセットを、組み込みのフィッティング関数または任意のユーザー定義のフィッティング関数にフィッティングさせることができます。
「バッチ」とは、ウェーブに保存された類似のデータセットの集合であり、共通のフィッティング関数、初期条件、重み付けとマスキングのウェーブが適用されています。
各データセットは、ウェーブフォーム、XY ペア、または 2D ウェーブフォームの列として保存することができます。





#include <WMFitFunctions>

ありふれたものからあまり知られていないものまで、多種多様なフィッティング関数が含まれています。
パラメーターの１つを一定に保つことができる線形カーブフィッティングも含まれています。
現在では組み込みの線形および多項式フィッティング関数でもパラメーターを固定できるようになったため、この機能は不要となっています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <WMFunctions>

このプロシージャも、ありふれたものから意外なものまで、さまざまな機能が含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

WaveMetrics アノテーションプロシージャ

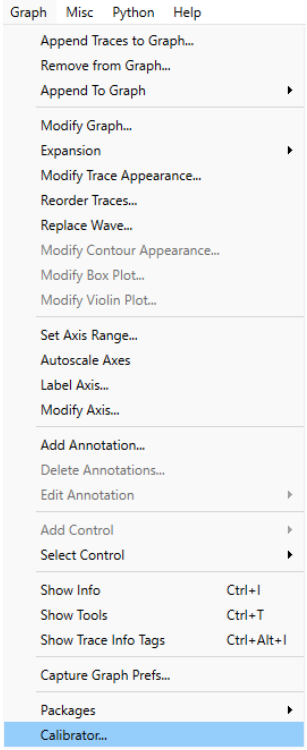
#include <AnnotationInfo Procs>

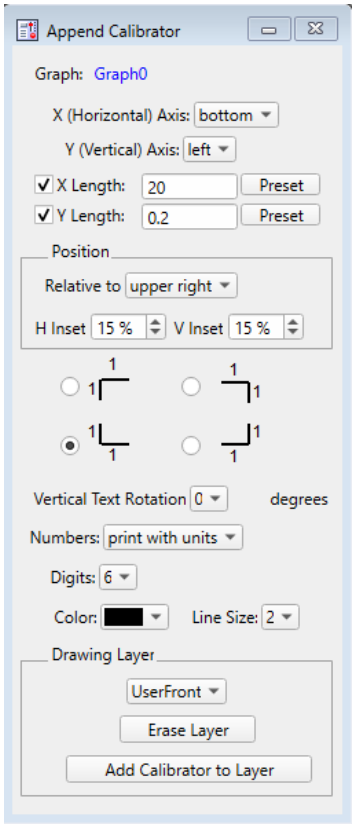
グラフやページレイアウトウィンドウ内のタグ、テキストボックス、または凡例に関する情報を取得するために使用できる一連の関数が含まれています。
例えば、タグが関連付けられているウェーブを特定したり、そのウェーブ上のタグが付けられたポイントを特定したりすることができます。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Append Calibrator>

最前面のグラフに「キャリブレーター」を追加するプロシージャ「Calibrator」を提供します。
キャリブレーターとは、地図の縮尺のように、寸法が示された線または２本の線のことです。

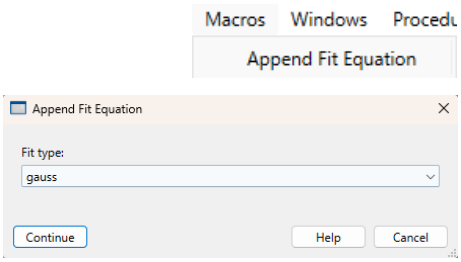
このプロシージャでは、描画ツールを使って、キャリブレーターを１つのグループ化されたオブジェクトとして作成します。
キャリブレーターの線は３ポイントの太さです。
キャリブレーターを変更するには、通常、まず描画ツールの「ブルドーザー」メニュー（  ）を使ってグループを解除します。





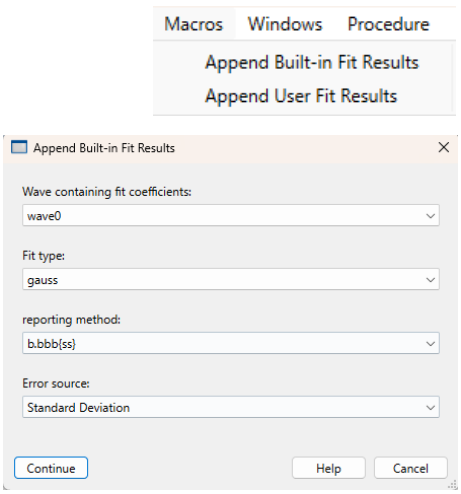
#include <Append Fit Equation>

組み込み関数によるカーブフィッティング実行後に実行すると、フィッティング結果から得られた実際の値を使って、フィッティングされた方程式を含むテキストボックスを追加または変更するプロシージャ「AppendFitEquation」を提供します。
値は、推定誤差に応じて適切に丸められます。
このルーチンは、テクニカルノート TN001 を基に作成されました。



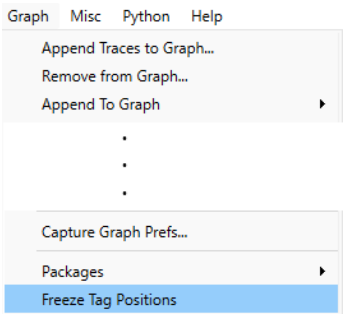
#include <Append Fit Results>

テクニカルノート TN001 に含まれるプロシージャで、最新のカーブフィッティングの結果を、見やすくフォーマットされたテキストボックスに入れて、最前面のグラフに追加するものです。
AppendBIFitResults は Igor の組み込みカーブフィッティング用であり、AppendUserFitResults はユーザー定義のカーブフィッティングで使われます。



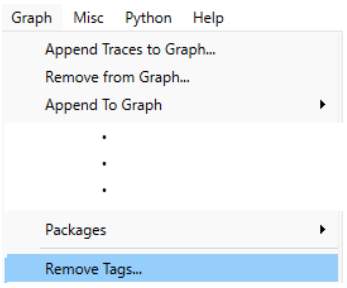
#include <FreezeUnFreezeTags>

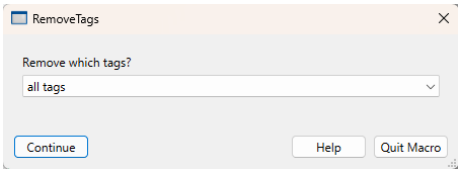
Graph メニューに Freeze Tag Positions または Un-freeze Tag Positions として表示される WM_FreezeOrUnFreezeTags プロシージャが含まれています。
このプロシージャは、等高線プロットのラベルの位置を変更するのに役立ちます。
ModifyContour の labels キーワードのヘルプを参照してください。



#include <Remove Tags>

Remove Tags プロシージャが含まれており、これは Graph メニューに表示されます。
このプロシージャは、最前面のグラフからすべてのタグを削除するか、または「画面外」（軸の範囲外）に配置されているすべてのタグを削除します。
各タグを個別にダブルクリックし、Change Annotation ダイアログで Delete をクリックするよりも簡単です。





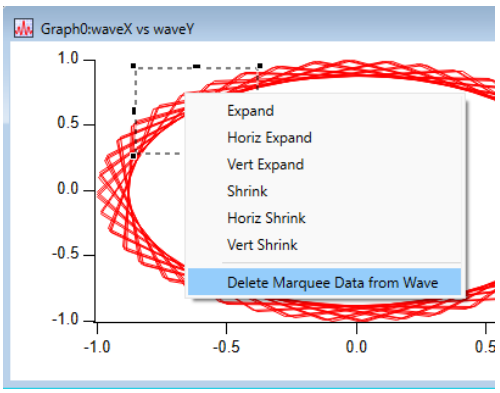
#include <Straighten Tags>

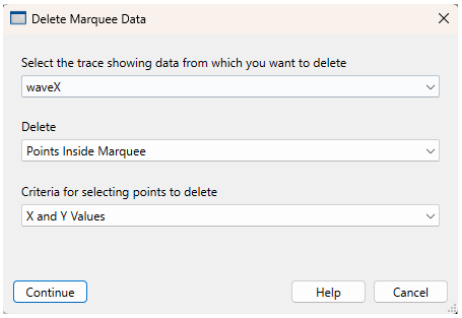
Straighten Tags プロシージャが含まれており、これは Graph メニューに表示されます。
このプロシージャは、最前面のグラフのタグの接続線を、垂直、水平、または 45 度間隔のいずれかに整列させます。
その時、現在の接続線に最も近い方向が選択されます。

WaveMetrics データ操作プロシージャ

#include <Delete Marquee Points>

XY ペアウェーブからデータを削除するために使用できる一連のルーチンを含んでいます。
データをグラフ化し、データの一部を囲むようにマーキーをドラッグして選択し、マーキー内をクリックした時に表示される通常のポップアップメニューから Delete Marquee Data from Wave メニュー項目を選択します。
するとダイアログが表示され、マーキーの内側または外側のデータ値を削除できるようになります。





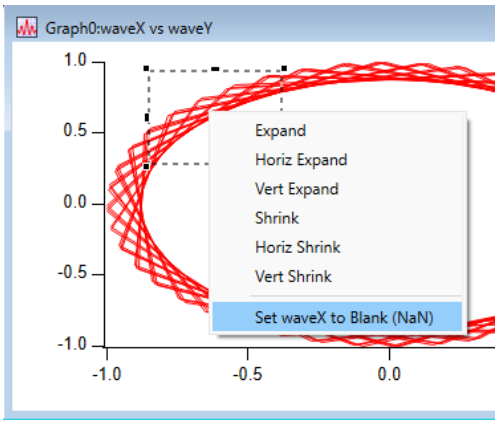
#include <Drag Spline>

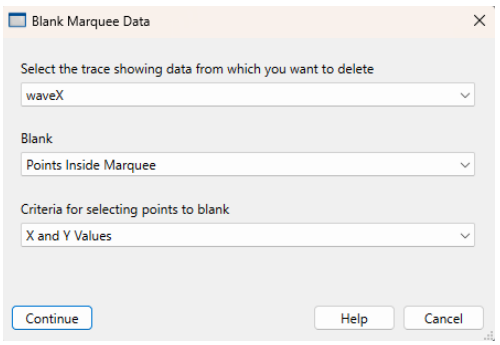
ノイズの多いデータセットにスプライン曲線を当てはめ、ドラッグ操作でスプラインを微調整できる一連のルーチンを含んでいます。
これを使う前に、まず平滑化スプラインを試してみてください（Interpolate コマンドのヘルプを参照してください）。

これらのルーチンは、現在のデータフォルダー内のウェーブに対してのみ動作します。

#include <Marquee Blanks>

選択した x と y の範囲において、グラフ化されたウェーブフォームまたは XY ウェーブフォームペアのデータを NaN に設定するルーチンを含みます。
データをグラフ化し、データの一部を囲むようにマーキーをドラッグして選択し、マーキー内をクリックした時に表示される通常のポップアップメニューから Set <data> to Blank (NaN) を選択します。
その後、ダイアログが表示され、選択範囲の内側または外側のデータ値を空白に設定できます。



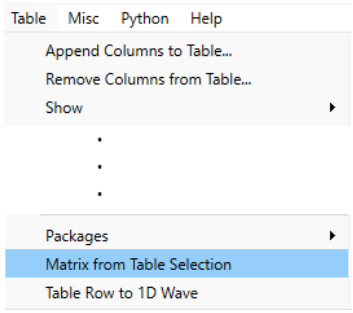


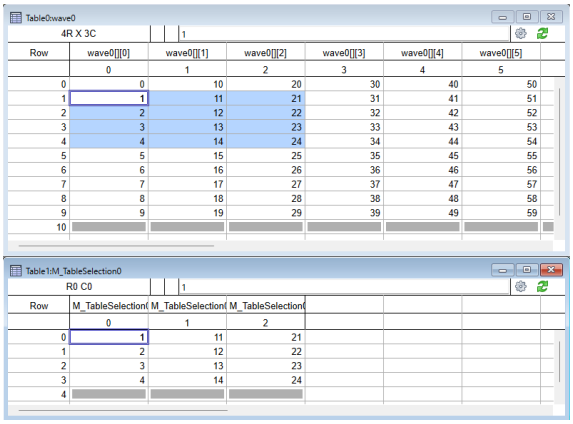
#include <MatrixFromTable>

テーブル内の長方形の選択範囲からマトリクスウェーブを作成します。

テーブルの選択した行から 1D ウェーブを作成します。

Tables メニューに 2 つのメニュー項目を追加します。



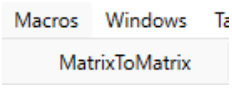


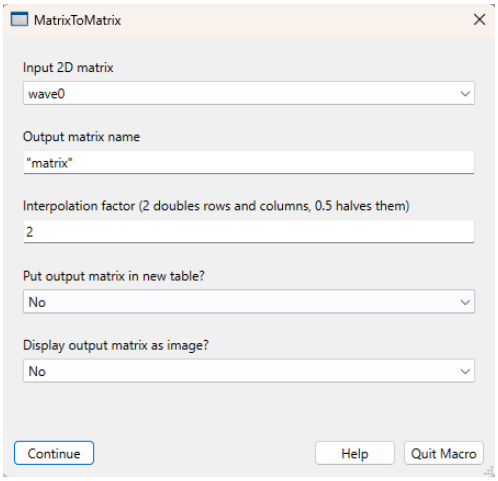
#include <MatrixToMatrix>

2D ウェーブ（行列）を補間（行と列の追加）または間引き（行と列の削除）する MatrixToMatrix プロシージャが含まれています。
結果は、グラフとして画像で表示したり、テーブル形式で表示したりすることができます。

このルーチンは、現在のデータフォルダー内のウェーブデータに対してのみ動作します。

Macros メニューにメニュー項目を追加します。



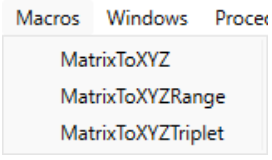


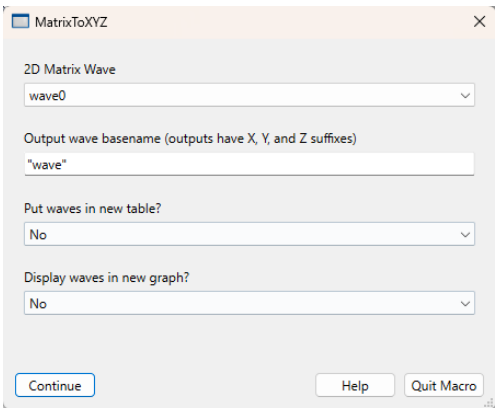
#include <MatrixToXYZ>

Z 値の 2D 行列を、X、Y、Z の 3 つの独立したウェーブに変換するプロシージャが含まれています。
MatrixToXYZ は行列全体を X、Y、Z のウェーブに変換し、MatrixToXYZRange は指定された XY 領域を X、Y、Z のウェーブに変換します。
結果はグラフ（3 つのトレースとして）またはテーブル（3 つの列として）で表示できます。

また、Z 値の 2D 行列を、3D プロッター「Gizmo」（3D Graphics のヘルプを参照）や AppendXYZContour などで使われるような XYZ トリプレット形式（0、1、2 列目にそれぞれ X、Y、Z のデータが含まれる 3 列の行列）に変換する MatrixToXYZTriplet も含まれています。

これらのルーチンは、現在のデータフォルダー内のウェーブに対してのみ動作します。





#include <Multi-dimensional Utilities>

多次元ウェーブスケーリングに関連する関数が含まれています。

x2pntMD は、1D ウェーブ用の x2pnt の多次元版です。
x、y、z、または t のスケーリング値を指定すると、最も近い行、列、レイヤー、またはチャンクのインデックスを返します。

また、pnt2xMD、rightxMD、lastxMD も含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Reduce Matrix Size>

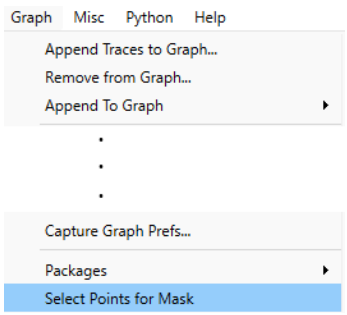
サンプリングまたは平均化によって、行列（2D ウェーブ）の行数や列数を削減する関数 ReduceMatrixSize を追加します。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Remove Points>

ウェーブまたは XY ペアのウェーブからデータポイントを削除するための関数を提供します。
RemoveOutliers は、指定された y 範囲外にあるデータ値を削除します。
RemoveOutliersXY は、XY ペアに対して同様の処理を行います。
RemoveNaNs は、ウェーブフォーム内のすべての欠損値を削除します。
RemoveNaNsXY は、XY ペア内のすべての欠損値を削除します。
DeletePointsXY は、x と y の範囲内または範囲外にあるポイントを削除します。
（このプロシージャをインクルードしても GUI 上に変化はありません）

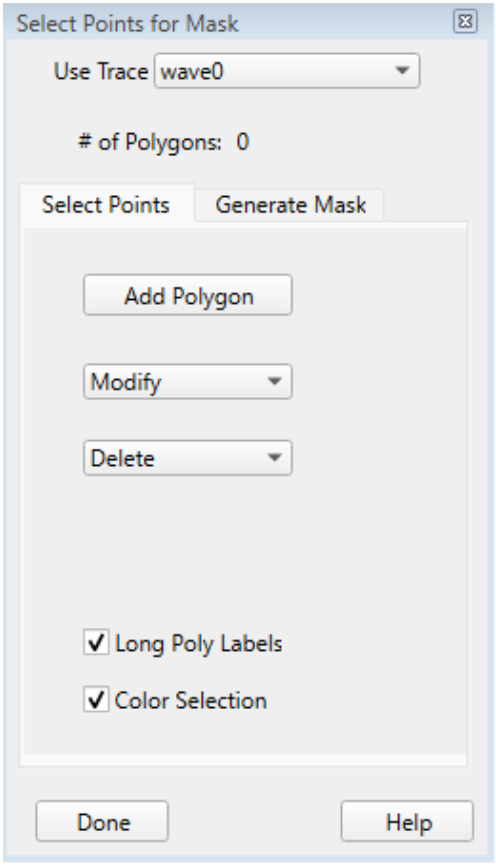
#include <Select Points for Mask>

グラフ上のポイントを中心にポリゴンを描画し、ポリゴンの内側／外側の点を示す選択可能な値を持つウェーブを作成します。
数値ウェーブまたはテキストウェーブを作成できます。
1 と 0、または 1 と NaN で構成される数値ウェーブは、フィッティングマスクウェーブ（ヘルプ Using a Mask Wave を参照）やグラフマスクウェーブ（ヘルプ ModifyGraph for Traces の mask キーワードを参照）として使うのに適しています。
数値ウェーブは、f(z) ウェーブとしても使用できます（ヘルプ Setting Trace Properties from an Auxiliary (Z) Wave を参照）。



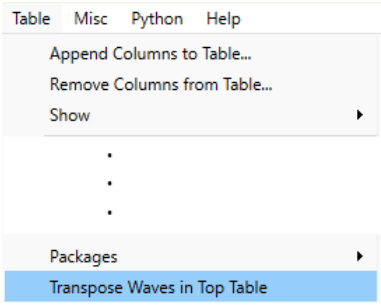
数値ウェーブまたはテキストウェーブのいずれかを、テキストマーカーウェーブとして使用できます（ヘルプ ModifyGraph for Traces の textMarker キーワードを参照）。

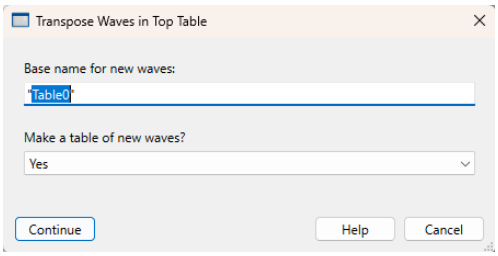
Graph メニューにメニュー項目を追加します。
グラフを最前面に表示し、Graph→Select Points for Mask を選択すると、パネルが表示されます。



#include <Transpose Waves In Table>

WMTransposeTableMenu と WMTransposeWavesInTable プロシージャを含み、Table メニューに Transpose Waves In Top Table 項目を追加します。
これらを組み合わせることで、最前面のテーブルにあるウェーブの転置ウェーブが生成され、オプションで新しいテーブルに表示されます。
例えば、それぞれ 10 行を含む 3 つのウェーブを持つテーブルがある場合、それぞれ 3 行の 10 個の新しいウェーブが生成されます。





#include <XY Pair To Waveform Panel>

（新しいバージョンでは、すでにこの項目は Data→Packages メニューにデフォルトで表示されています）

Data→Packages メニューに XY Pair to Waveform というメニュー項目が追加されます。

XY2WFM#ShowXYPairToWaveformPanel() は、この処理を簡単に実行できるパネルを開きます。

現在では使われなくなった <XY Pair to Waveform> とは異なり、このパネルは、X ウェーブが不要で、Y ウェーブの X スケーリングを設定することで置き換え可能である場合も識別します。
また、このパネルは整数ウェーブやテキスト Y ウェーブにも対応しています。

#include <XYZtoMatrix>

（新しいバージョンでは、すでにこの項目は Data→Packages メニューにデフォルトで表示されています）

X、Y、Z の各ウェーブ、または XYZ のトリプレットウェーブを、Z 値の 2D 行列に補間する 3 つのマクロと、X と Y の値とそれに対応する Z 値のグリッドを、Z 値の 2D 行列に再配置する 1 つのマクロが含まれています。
出力された行列は、グラフとして画像で表示することも、テーブル形式で表示することもできます。

XYZtoMatrix は、X ウェーブと Y ウェーブの XY 領域全体を行列に補間します。

XYZtoMatrixRange は、指定された XY 領域を行列に変換します。

XYZTripletToMatrix は、入力が XYZ トリプレットウェーブである点を除けば、XYZtoMatrixRange と同じです。
出力行列は、グラフ上で画像として、またはテーブル形式で表示できます。

XYGridandZtoMatrix は、X と Y 値の等間隔グリッドを構成する X、Y、Z 値を含む 3 つのウェーブ（これらはすでに列優先または行優先の順序でソート済み）を、X と Y の最小値から最大値までをカバーする Z 値の行列に再配置します。

これらのルーチンは、現在のデータフォルダー内のウェーブに対してのみ動作します。

#include <XYZtoTripletToXYZ>

（新しいバージョンでは、すでにこの項目は Data→Packages メニューにデフォルトで表示されています）

3 つの独立した X、Y、Z のウェーブと、X、Y、Z の値を格納した 2D 3 列の行列（「トリプレット」）との間で変換を行う 2 つのマクロと 2 つの関数が含まれています。
この行列では、X の値が 0 列目、Y の値が 1 列目、Z の値が 2 列目に格納されています。

これらのルーチンは、現在のデータフォルダー内のウェーブに対してのみ動作します。

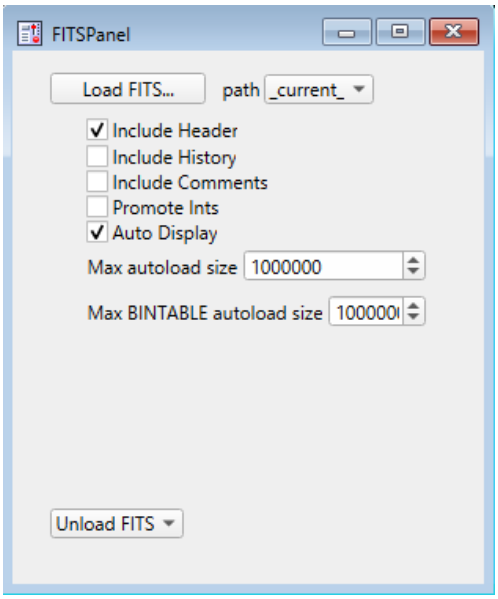
WaveMetrics ファイル入出力プロシージャ

#include <FITS Loader>

NASA の Flexible Image Transport System（FITS）ファイルを読み込む関数が含まれています。
この関数はプライマリ配列を読み込み、2D または 3D データの画像、1D データの通常のグラフを作成します。

詳細については、デモエクスペリメント File→Example Experiments→Imaging→FITS Loader Demo を参照してください。

（新しいバージョンでは、すでにこの項目はメニューにデフォルトで表示されています。Data→Load Waves→Packages）



#include <HDF Loader>

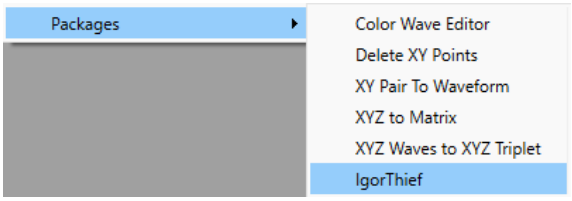
階層データ形式バージョン 4（HDF4）ファイルの読み込みをサポートするルーチンを含みます。
「More Extensions」フォルダー内の HDF XOP が必要です。

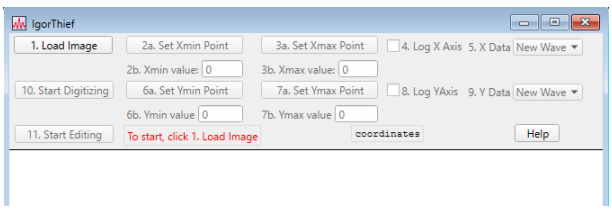
File→Example Experiments→Programming の HDF Demos エクスペリメントで使われています。
HDF Demos エクスペリメントには、<HDF Utilities> ルーチンの使い方を解説したノートブックが含まれています。

#include <IgorThief>

グラフィックファイルを手動でトレースし、グラフに表示されている値を取得するための GUI を追加します。
いわゆる「デジタイザー」の機能です。

この GUI は、インクルード後、Data→Packages→Igor Thief メニューから起動します。





#include <Image Saver>

Save Waves メニューに Save Image... というメニュー項目を追加します。

（新しいバージョンでは、すでにこの項目はメニューにデフォルトで表示されています）

表示される Save Image File ダイアログには、ImageSave コマンドを行うためのインターフェイスが用意されています。

#include <Load Row Data>

テキストファイルの各行からデータを読み込み、１つの 1D ウェーブまたは複数の 1D ウェーブに格納するためのユーティリティが含まれています。
組み込みのテキストファイルローダーは、列指向です。

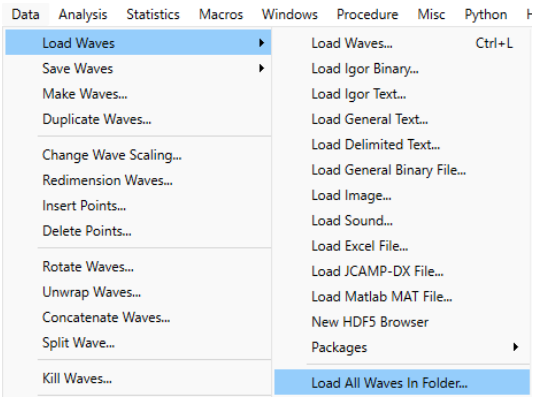
LoadRowDataInto1DWaves プロシージャは、行指向のテキストファイルの内容を複数の 1D ウェーブに読み込みます（複数形です）。

LoadRowDataInto1DWave プロシージャは、行指向のテキストファイルの内容を１つの 1D ウェーブに読み込み、すべての行をつなぎ合わせます。

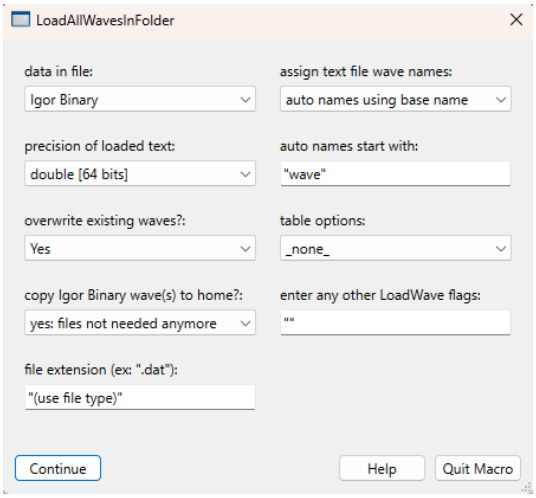
詳細については、メニュー File→Example Experiments→Techniques→Load Row Data のサンプルエクスペリメントを参照してください。

#include <Wave Loading>

フォルダー内のすべての Igor バイナリウェーブファイル、またはテキストファイルをウェーブとして読み込む LoadAllWavesInFolder プロシージャが含まれています。
このプロシージャは、Data→Load Waves サブメニューに表示されます。



また、外部ファイルへの参照が不要になるよう、すべての共有ウェーブを取り込む CopyAllWavesToHome プロシージャも含まれています（ヘルプ Sharing Versus Copying Igor Binary Wave Files を参照）。



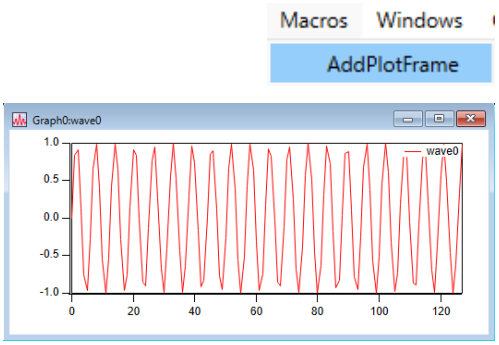
#include <WaitForFileProcs>

WaitForFileToExist と WaitForWriteAccesToFile 関数を追加します。
これらは、別のプログラムがファイルを書き込み、閉じ終わるまで待機し、Igor プロシージャがそのファイルを開いて処理できるようにするのに役立ちます。
（このプロシージャをインクルードしても GUI 上に変化はありません）

WaveMetrics グラフ関連プロシージャ

#include <AddPlotFrame>

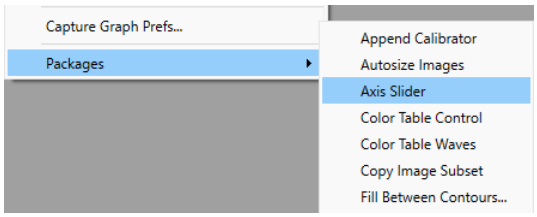
グラフのプロット領域の周囲に長方形を描画する、簡単なプロシージャ AddPlotFrame が含まれています。
これは、積み重ね軸やずらし軸を使っている時に、枠も表示したい場合に利用できます。

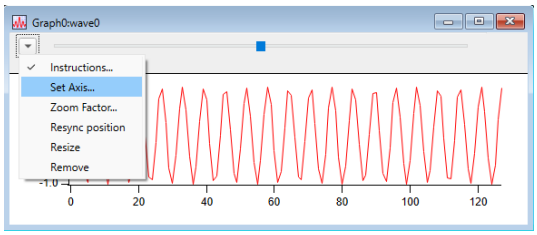


#include <AxisSlider>

最前面のグラフにスライダーを追加し、長いトレースを簡単にスクロールできるようにします。
また、拡大倍率を設定するためのポップアップメニューも追加されます（10 倍の場合、グラフにはトレース全体の 10 分の 1 が表示されます）。
下記にある「GraphMagnifier」も参照してください。

（新しいバージョンでは、すでにこの項目はメニューにデフォルトで表示されています）





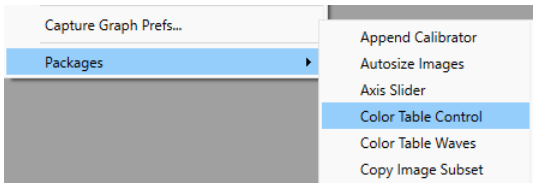
#include <Axis Utilities>

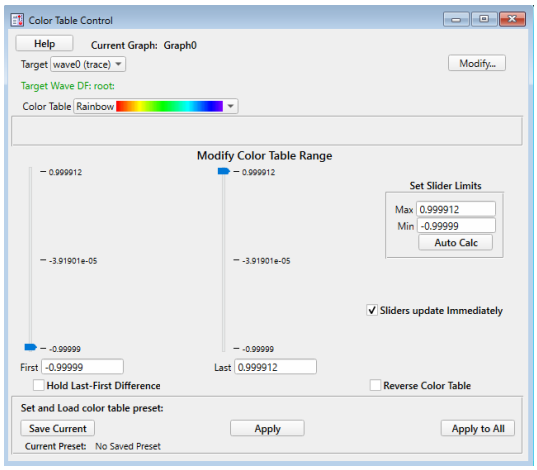
グラフの軸に関する情報を返す文字列関数が含まれています。
HVAxisList は、横軸または縦軸のリストを返します。
AxisOrientation は、指定された軸が指定された向きになっているかどうかを真偽値で返します。
AxisUnits は指定された軸の単位を返し、AxisLabelText は軸のラベルテキストを返します。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Color Table Control Panel>

Graph メニューに Color Table Control 項目を追加し、カラーテーブルまたはカラーインデックスウェーブに基づいて、画像、コンター、f(z) トレースに表示される色をインタラクティブに設定するためのスライダーベースの GUI を実装します。

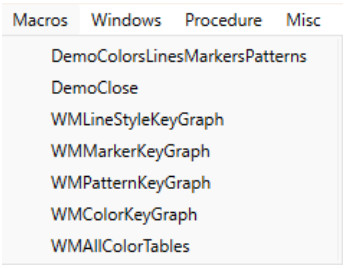
（新しいバージョンでは、すでにこの項目はメニューにデフォルトで表示されています）



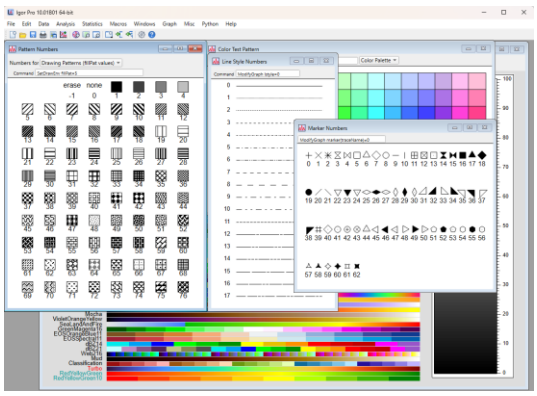


#include <ColorsMarkersLinesPatterns>

グラフの線種、マーカー、パターンのプログラミングコード番号や、標準カラーパレット、カラーテーブルの色に関する RGB 値を表示するために呼び出せる関数が含まれています。
パターンなどをクリックするだけで、デモ用グラフが関連するプログラミングコードを生成します。
お気に入りのパターンの番号を調べるのに適しています。



DemoColorsLinesMarkersPatterns マクロを使うと、すべてのデモグラフを表示でき、DemoClose を使うと、それらをすべて非表示にできます。



#include <CsrTraceName>

グラフカーソルが位置しているトレースの名前を取得する文字列関数が含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Drawing Axes>　　// 廃止

Macros メニューに Draw Derived Axis メニュー項目を追加します。
この項目を選択すると、グラフに「派生」軸を追加できるコントロールパネルが表示されます。
例えば、波長軸がある場合、波数軸を作成することができます。
また、このパッケージを使うと、任意の位置に目盛線を配置した軸を作成することも可能です。

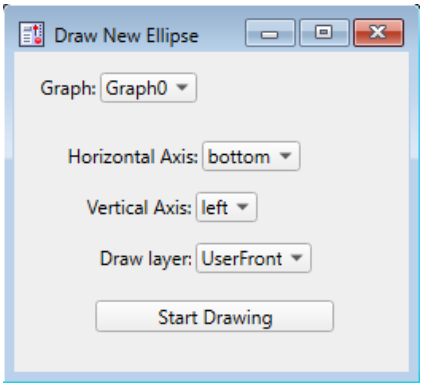
これは Transform Axis パッケージに置き換えられました。
このパッケージは同様の機能を提供しますが、実装方法が異なり、これまでより優れています。

#include <DrawEllipseOnGraph>

グラフ上に楕円を描画、グラフ上の楕円を編集します。
組み込みの描画ツールの楕円ツールとは異なり、水平・垂直以外の軸方向の楕円も描画可能です。
このプロシージャファイルにはヘルプファイルが付属しています。

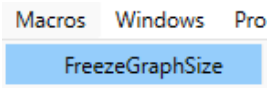
インクルードすると Graph メニューに Oblique Ellipses というメニュー項目が追加されます。

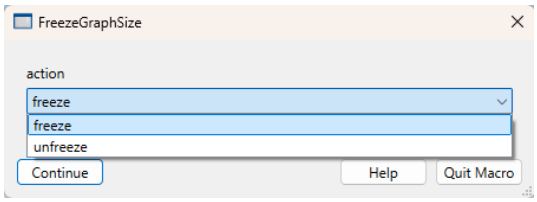
DrawArc コマンドも参照してください。



#include <Freeze Graph Size>

現在のサイズで最前面のグラフを固定できるシンプルなマクロ FreezeGraphSize が含まれています。
固定を解除することも可能です。
このマクロは、描画ツールを使う時に、描画座標系などの詳細に気を配ることなく、WYSIWYG（見たままがそのまま出力される）の結果を得られるようにするためのものです。
ページレイアウトで固定されたグラフを使う場合や印刷する場合でも、常に現在のサイズで表示されます。



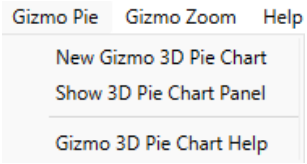


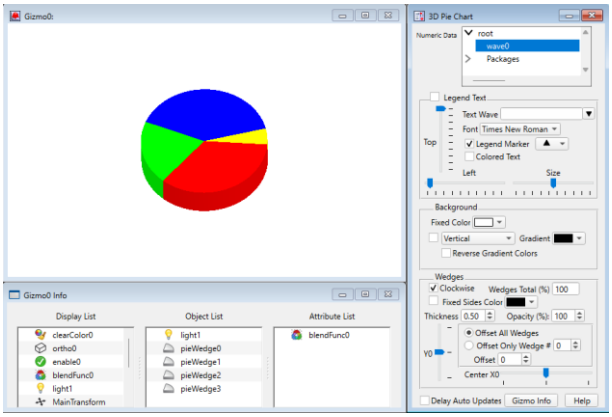
関数版（FreezeUnfreezeGraphSize）も用意されています。

#include <Gizmo3DPieChart>

Gizmo ウィンドウに 3D 円グラフを作成するプロシージャです。
Gizmo XOP の詳細については、ヘルプ Gizmo Overview を参照してください。

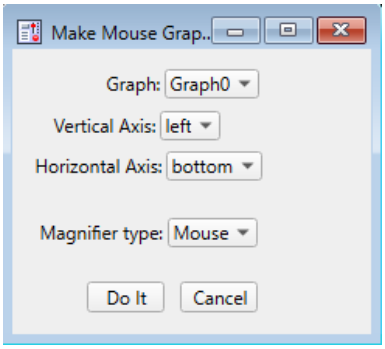
インクルードするとメインメニューに Gizmo Pie というメニュー項目が追加されます。



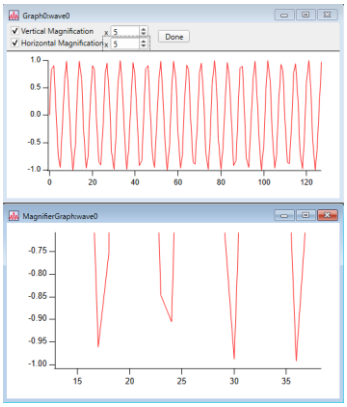


#include <GraphMagnifier>

グラフの拡大表示機能を提供し、グラフの詳細を確認できるようにします。
元のグラフと全く同じグラフをもう 1 つ作成します。
拡大表示する領域は、カーソルの周囲（マウス拡大表示）か、グラフのカーソルで囲まれた領域（カーソル拡大表示）のいずれかを選択できます。
上記の AxisSlider も参照してください。



インクルードすると Graph メニューに Graph Magnifier というメニュー項目が追加されます。



```
#include <Graph Utility Procs>
```

グラフ、軸、およびトレースに関連する関数が含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

CopyAxisSettings： ある軸の設定を読み取り、それを別の軸に適用します。

CopyTraceSettings： あるトレースからトレース設定を読み取り、別のトレースに適用します。

ApplyStyleMacro： スタイル再作成マクロを含む文字列からコマンドを実行します。

WMSetGraphSizePoints
WMSetGraphSizePixels： 余白のサイズを考慮してグラフのサイズを設定し、グラフのサイズを固定または解放します。

WMGetGraphPlotBkgColor： ModifyGraph gbRGB で設定された色を返します。

WMGetGraphWindowBkgColor： ModifyGraph wbRGB で設定された色を返します。

WMGetLayoutWindowBkgColor： ModifyLayout bgRGB で設定された色を返します。

WMGetColorsFromTopGraph： ウィンドウおよびプロット領域の背景、トレース、軸、目盛り、軸ラベル、目盛ラベル、f(z) と画像プロットのカラーテーブル、カラーインデックスウェーブ、直接 RGB 画像、注釈、および描画ツールから色を収集します。
関連する AskUserWhichColors ルーチンは、色を含むか除外するかをユーザーに選択させるために、色のカテゴリを提示します。

WMGetRECREATIONFromInfo
WMGetRECREATIONInfoByKey： AxisInfo、TraceInfo、ImageInfo などから RECREATION セクションを抽出するのに役立ちます。

Axis Utilities も参照してください。

```
#include <HighLowCloseOpen>
```

株価の日次推移を追跡するには、「高値・安値・終値・始値」のグラフを使います。
このグラフでは、１日の４つの価格すべてが表示されます。
このプロシージャでは、既存のグラフにそのようなグラフを追加します。

このプロシージャファイルは、Graph メニューに Append High Low close Open Trace を追加し、さらに線の太さと色をコントロールするための２つのサブメニューを追加します。
線の色と太さは複数の設定項目によってコントロールされるため、これらのサブメニューを設けることで、すべての設定値を統一しやすくなります。

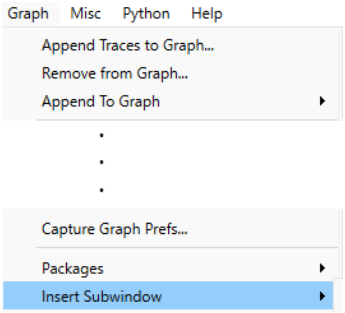
Igor 6.2 以降ののカスタムマーカー機能により、このようなチャート表示が可能になりました。
この機能では、始値と終値のマーカーが、実質的に半分の大きさのマーカーとして描画され、始値は左向き、終値は右向きに表示されます。

このプロシージャはそれほど複雑ではありません。現在のデータフォルダー内に、高値と安値のウェーブを１つのウェーブに統合したウェーブを作成するだけです。

入力データは、長さが等しい５つのウェーブのセットです：{高値、安値、終値、始値} vs 日付。
出力では、個々の高値と安値のウェーブの代わりに、６つ目の高値・安値のウェーブがグラフに追加されます。

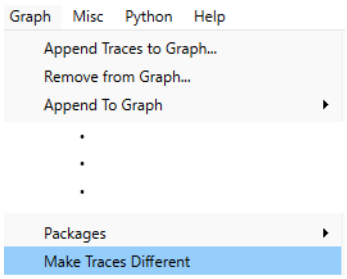
```
#include <InsertSubwindowInGraph>
```

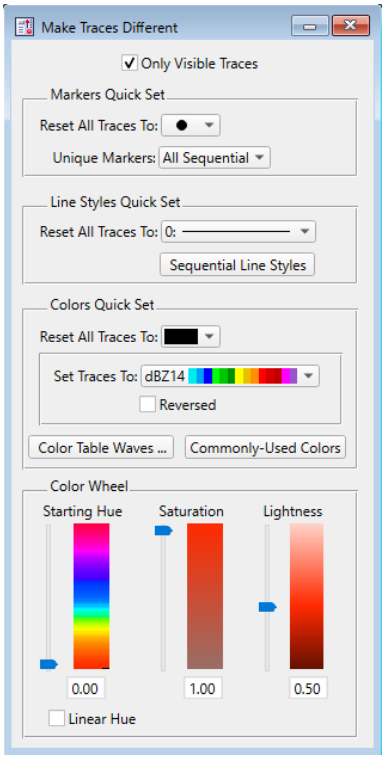
WMSubwindowInGraphPanel() は、最前面のグラフにサブウィンドウとして挿入する名前付きテーブルまたはグラフを選択するためのシンプルなパネルを表示します。
これは、WMInsertSubwindowIntoGraph() を呼び出すことで行われます。




```
#include <KBColorizeTraces>
```

グラフ内のすべてのトレースの色を１つの色に設定したり、色相、彩度、明度のスライダーコントロールで計算された連続した色範囲に設定したりするためのプロシージャが含まれています。
CreateKBColorizePanel プロシージャを使って、Colorize Traces パネルを表示します。






```
#include <PieChart>
```

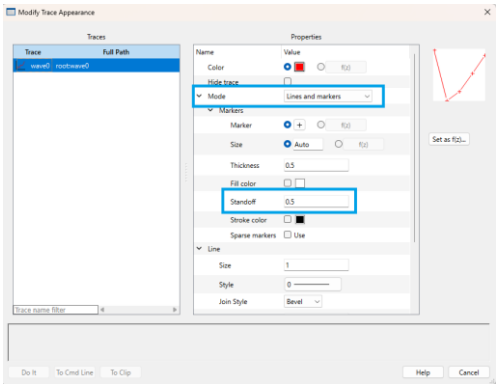
描画ツールを使ってグラフウィンドウに円グラフを作成するプロシージャが含まれています。
Windows→New→Packages→2D Pie Chart を選択すると、パネルベースの GUI が表示されます。
（新しいバージョンでは、すでにこの項目はメニューにデフォルトで表示されています）

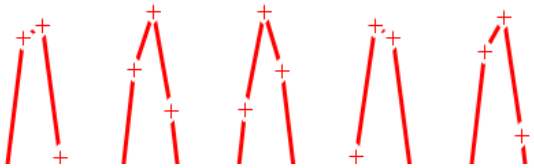
Igor 6.32 で、新しい関数 PieChartForProgrammers()、ModifyPieChart() を使って円グラフを作成、変更する機能が追加されました。
また、それ以前は静的（プライベート）だった一部のルーチン、構造体、および定数定義が、現在の円グラ

フの値を読み取るためにパブリックになりました。
特に重要なのは、構造体 PieChartInfo と関数 GetPieStruct() です。

#include <Marker Standoff Contextual Menus>

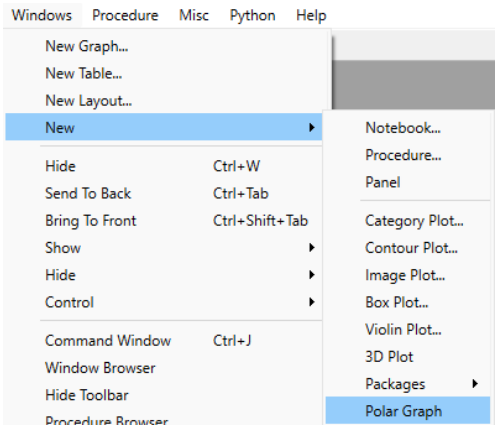
個々のトレースおよびすべてのトレースに対して、マーカーのオフセットを設定するコンテキストメニューが追加されます。
マーカーのオフセットは Lines and Markers mode (4) でのみ機能し、マーカーの周囲に線が描かれないオフセットの隙間を作ります。
新しいバージョンでは、この機能は、Modify Trace Appearance ダイアログに実装されています。

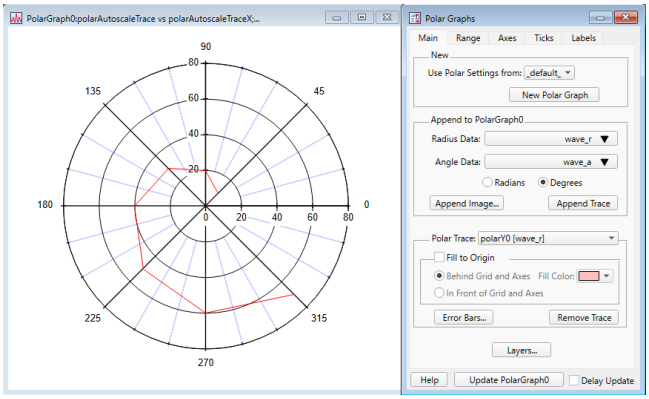




#include <New Polar Graphs>

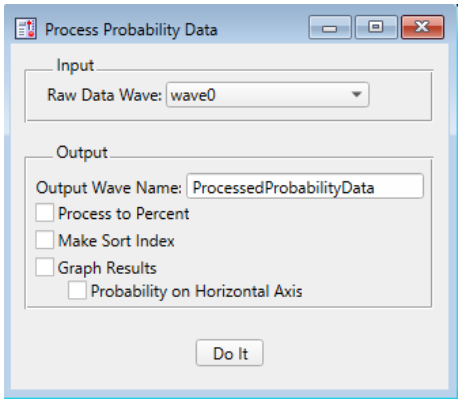
この極座標プロット用パッケージは、グラフ内で極座標プロットを作成、編集するためのパネルを備えています。
パッケージには、より詳細なヘルプファイルが組み込まれています。
パネル内の Help ボタンをクリックするだけで表示されます。
Windows → New → Polar Graph を選択して、Polar Graph パネルを開いてください。

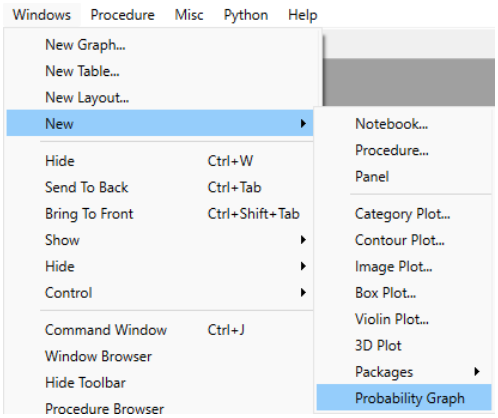




#include <ProcessProbabilityData>

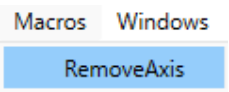
新しい TransformAxis パッケージを使って、ランダムなサンプルを正規確率グラフに適した形式に処理し、グラフを作成します。





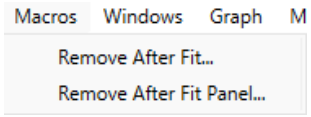
#include <Remove Axis Proc>

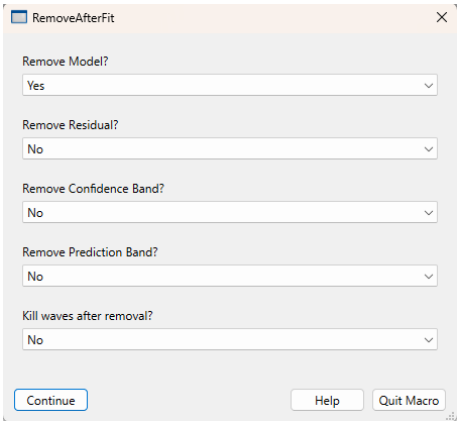
グラフから軸を削除するプロシージャ「RemoveAxis」が含まれています。このプロシージャは、その軸に対してプロットされたすべてのウェーブを削除することで、グラフから軸を取り除くことができます。



#include <RemoveAfterFit>

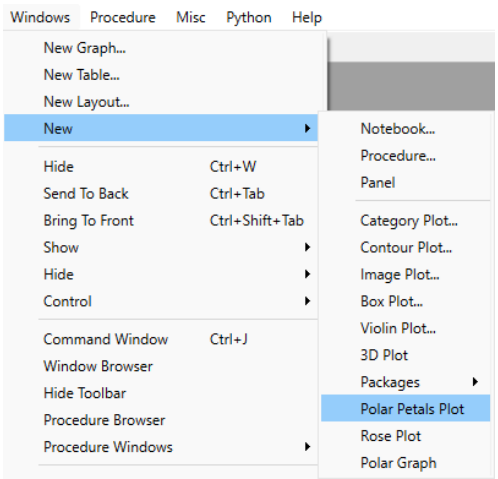
カーブフィッティングを行った後、グラフから自動的に追加されたさまざまなトレースを簡単に削除する方法を提供します。

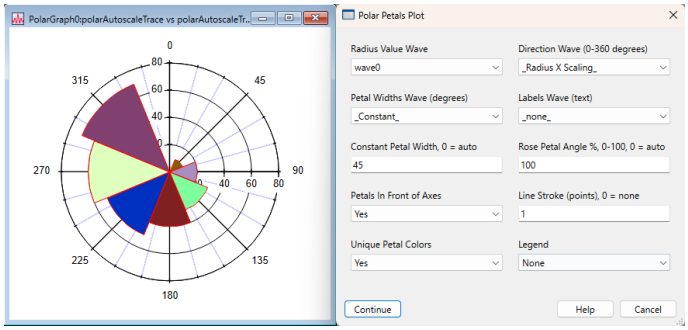




#include <Polar Petals Plot>

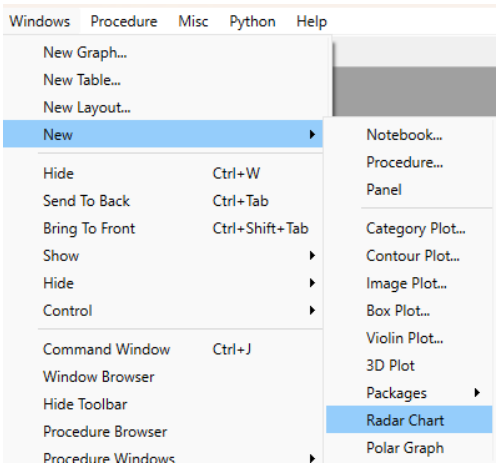
極座標グラフプロシージャを使って、半径ウェーブとオプションで指定した方向ウェーブから「花びら（petal）」プロットを作成します。
Windows→New→Polar Petals Plot を選択して、Polar Petals Plot ダイアログを開きます。
Polar Petals Plot ダイアログの Help をクリックすると、Polar Petals Plot Help.ihf ヘルプファイルを表示できます。

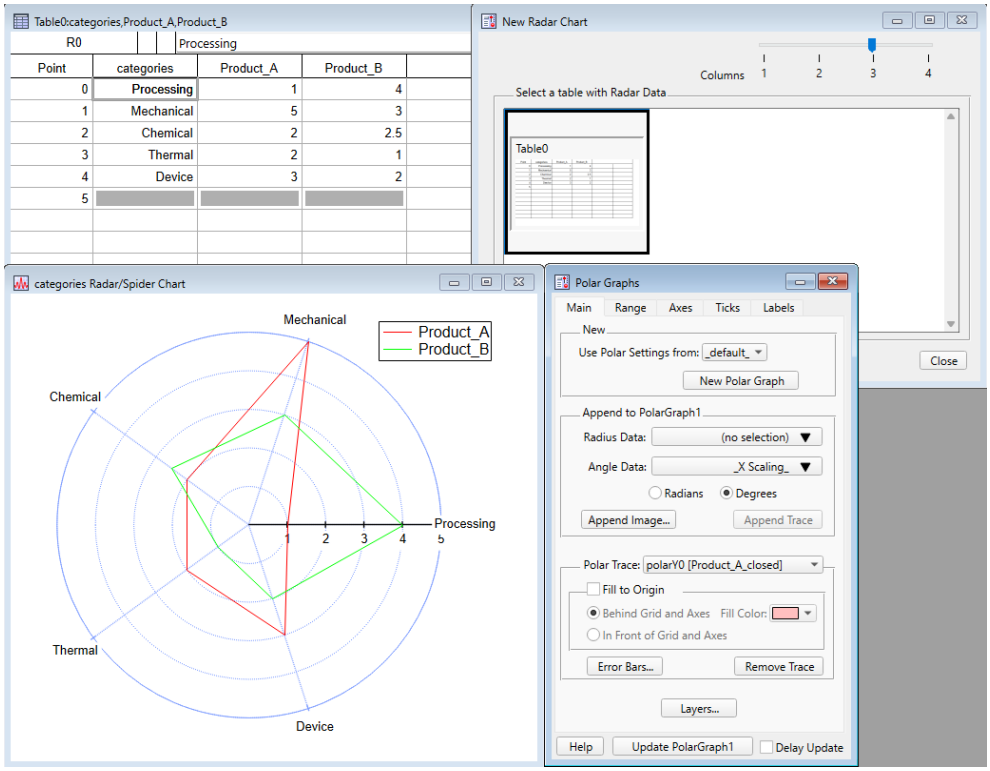





```
#include <RadarChart>
```

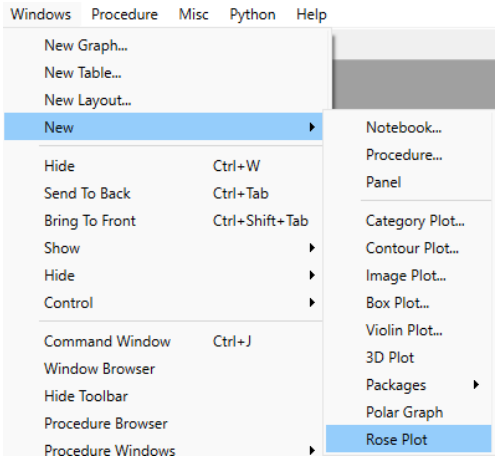
「レーダー」、「スパイダー」、または「スター」チャートを作成するプロシージャです。
これらは放射状のカテゴリプロットの一種で、極座標グラフプロシージャを使ってグラフウィンドウ内で、１つ以上の対象に関する数値とカテゴリの関係を示すために使われます。
詳細については、ヘルプ Radar Chart Procedure を参照してください。
Windows→New→Radar/Spider Chart を選択して、Radar Chart ダイアログを開きます。
また、Radar Chart ダイアログで Help をクリックすると、RadarChart.ihf ヘルプファイルにアクセスできます。
スパイダープロットは、レーダーチャートの派生形です。

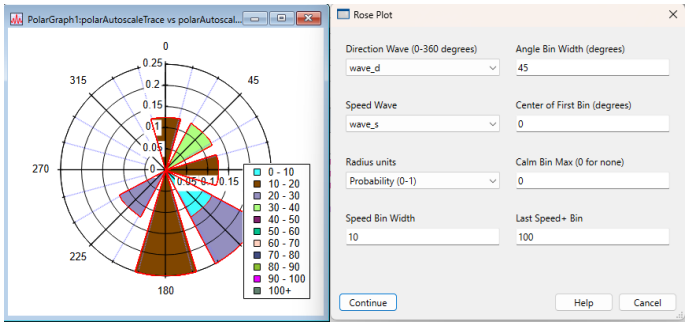




#include <RosePlot>

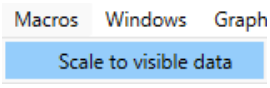
Polar Graphs プロシージャを使って、グラフウィンドウにローズプロット（風速や風向の確率を示すためによく使われる）を作成するプロシージャです。
詳細については、ヘルプ Rose Plot Procedure を参照してください。
Windows→New→Rose Plot を選択して、Rose Plot ダイアログを開きます。
また、Rose Plot ダイアログの Help をクリックすると、RosePlot.ihf のヘルプファイルを表示することもできます。





#include <SetAxisRangeToVisibleData>

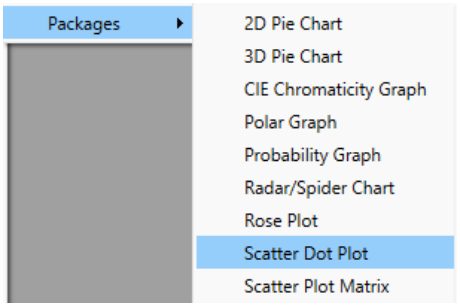
関連する横軸の X 範囲内に収まるデータのみに基づいて、上部のグラフの縦軸をスケーリングします。



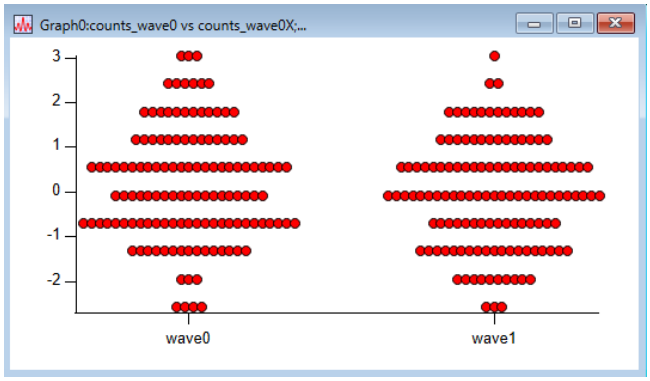
これは、「自動スケーリング」を選択した場合の動作とは異なります。
自動スケーリングを行う場合、表示されているかどうかにかかわらず、グラフ化されたウェーブのすべてのデータを含むように縦軸の範囲を設定します。
詳しい手順は、プロシージャファイルの先頭にあるコメントに記載されています。

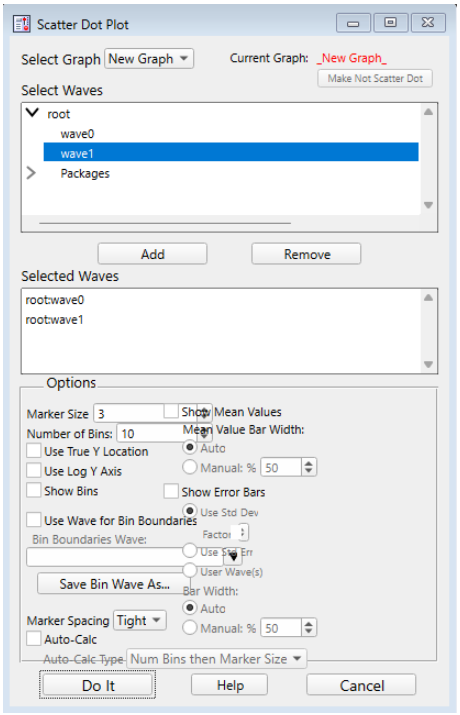
#include <Scatter Dot Plot>

散布図を作成するパネルを作成するためのプロシージャが含まれています。
散布図は、カテゴリプロット、散布図、ヒストグラムの要素を併せ持っています。
カテゴリプロットと同様に、複数のデータセットの合計件数を表示し、各データセットは X 軸上にラベル付けされます。
散布図と同様に、データの分布の傾向を示します。
ヒストグラムと同様に、データを範囲ごとに分類し、すべての値がその範囲内に収まるようにします。



（新しいバージョンでは、すでにこの項目はメニューにデフォルトで表示されています）
Windows→New→Packages にあります。





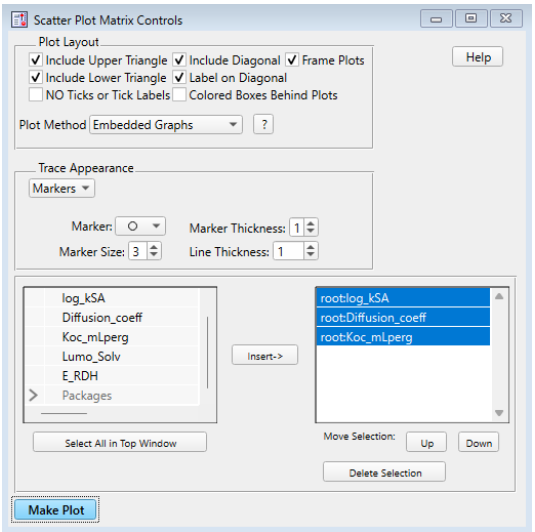
このパネルの使用方法については、File→Example Experiments→Graphing Techniques にあるデモエクスペリメント「Scatter Dot Plot Demo」を参照してください。

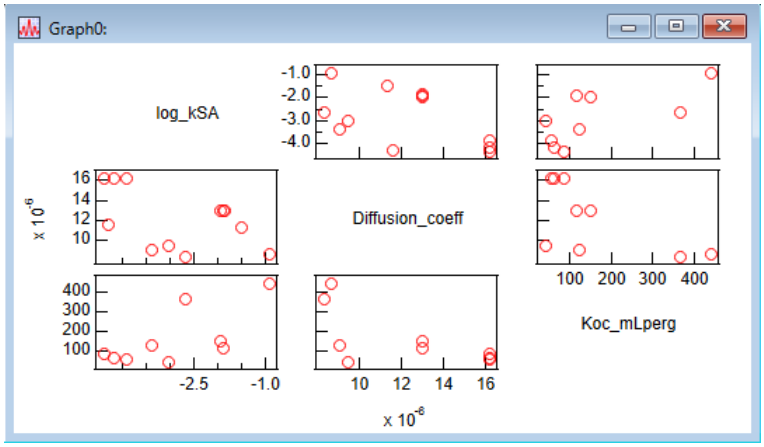
`#include <Scatter Plot Matrix>`

ウェーブのリストから散布図の行列を作成するプロシージャが含まれています。
生成されるグラフは、リスト内のウェーブのすべての組み合わせを X 座標と Y 座標として用いた散布図の行列を示します。

（新しいバージョンでは、すでにこの項目はメニューにデフォルトで表示されています）
Windows→New→Packages にあります。

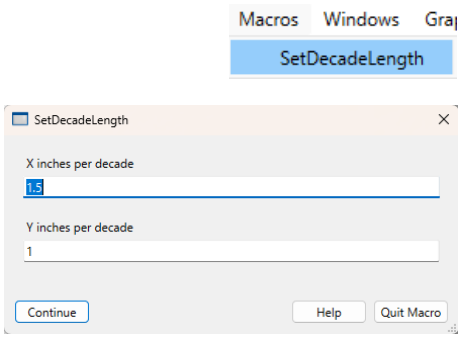
このプロシージャの使用方法については、File→Example Experiments→Graphing Techniques にあるデモエクスペリメント「Scatter Plot Matrix Demo」を参照してください。





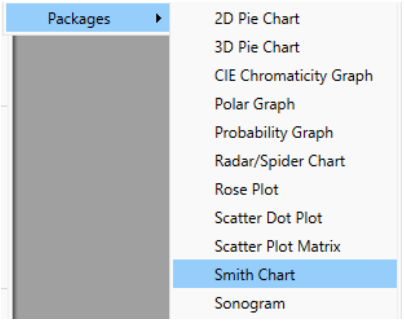
#include <SetDecadeLength>

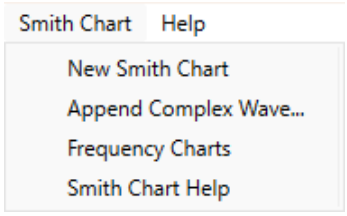
このモジュールには、SetDecadeLength というプロシージャが含まれています。
これは、最前面のウィンドウが対数対数グラフであることを前提として、グラフのプロット領域のサイズを設定し、横軸および縦軸の 1 デケードの長さがそれぞれ xInches および yInches になるようにします。
その次元を変更したくない場合は、xInches または yInches に 0 を入力してください。

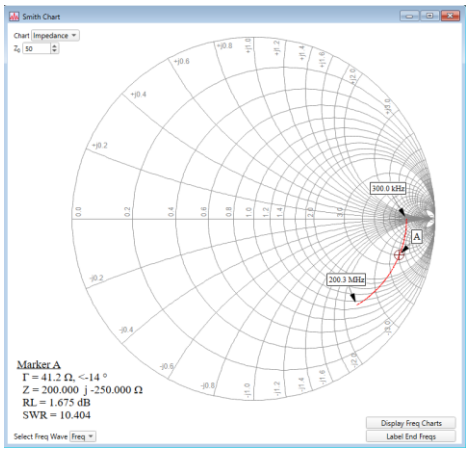


#include <SmithChart>

複素反射係数ガンマを表示するために使われるスミスチャートを生成するプロシージャです。
詳細は、ヘルプ Smith Chart Procedure を参照してください。
Smith Chart→New Smith Chart を使って、スミスチャートのグラフを作成します。
スミスチャートメニューには他にも項目がありますが、それらはヘルプファイルで説明されています。





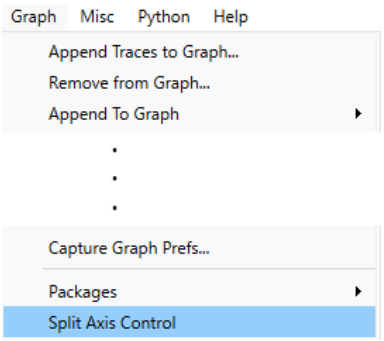


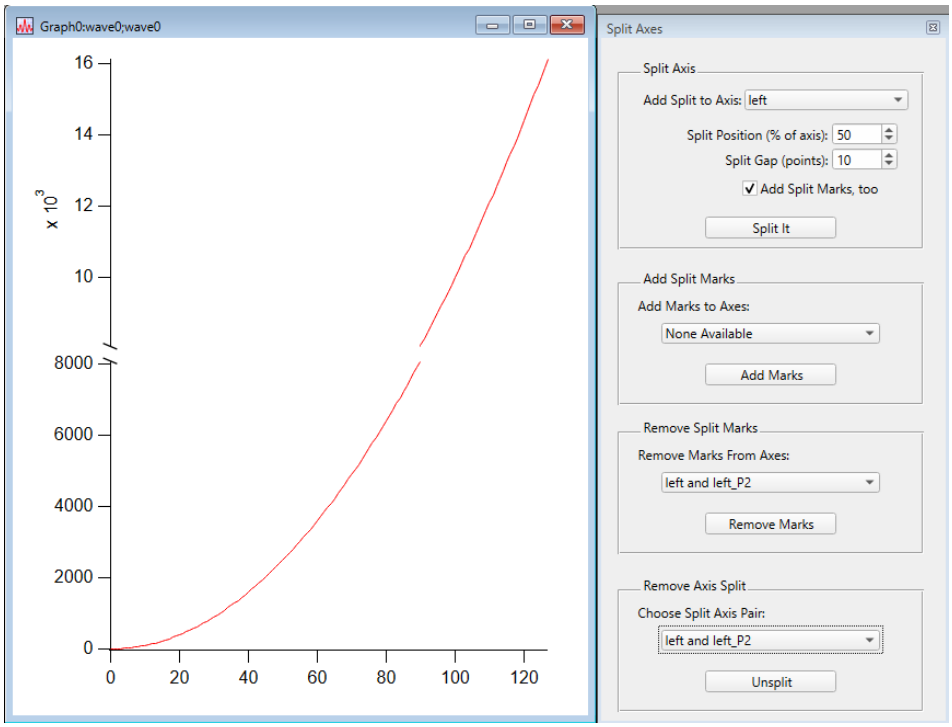
（新しいバージョンでは、Smith Chart の項目はメニューにデフォルトで表示されています）
Windows→New→Packages にあります。

#include <Split Axis>

２つ以上のセグメントに分割された軸を作成するためのプロシージャが含まれています。
SplitAxis というプロシージャは、指定された軸を２つに分割し、AddSplitAxisMarks というプロシージャは、描画オブジェクトとして目盛を追加します。
詳細については、「Split Axes」という名前のサンプルエクスペリメントを参照してください。

インクルードすると Graph メニューに Split Axis Control という項目が追加されます。

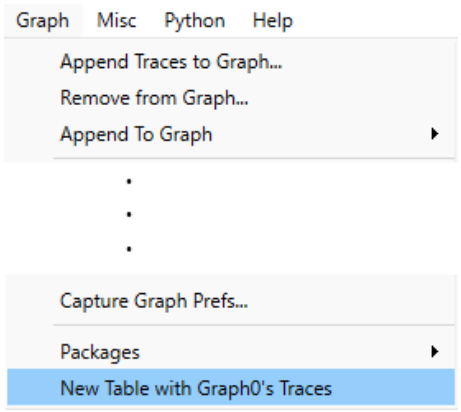


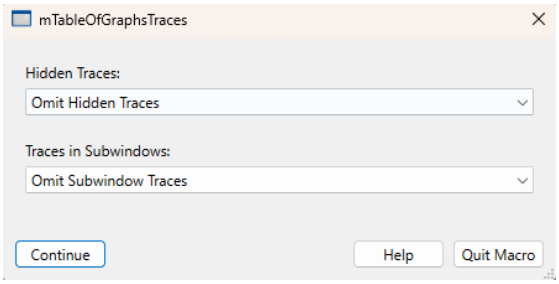


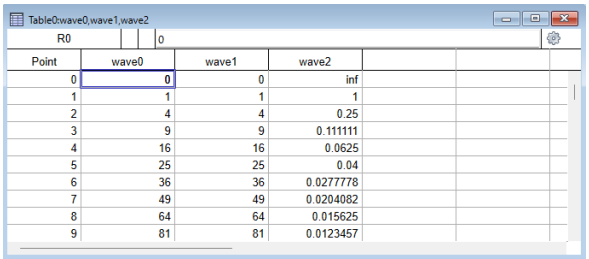

```
#include <TableOfGraphsTraces>
```

グラフに表示されているウェーブを含むテーブルを作成するための Graph メニューの定義が含まれています。

非表示のトレースやサブウィンドウ内のトレースを選択するためのオプションがあります。







Point	wave0	wave1	wave2
0	0	0	inf
1	1	1	1
2	4	4	0.25
3	9	9	0.111111
4	16	16	0.0625
5	25	25	0.04
6	36	36	0.0277778
7	49	49	0.0204082
8	64	64	0.015625
9	81	81	0.0123457

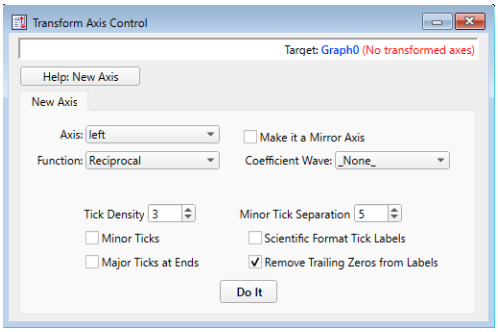
```
#include <TransformAxis1.2>
```

変換軸グラフを作成します。
つまり、変換された空間にデータをプロットし、適切な目盛線を持つ軸を作成します。
また、変換された空間の値を表示する鏡像軸も作成します。

相互変換軸や確率軸が組み込まれており、いくつかの温度換算機能も備えています。

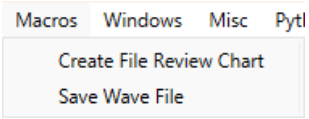
変換軸の柔軟性を高め、グラフの変化に合わせて変換軸を調整する機能を強化しています。
以前の TransformAxis プロシージャで作成された変換軸は、TransformAxis1.2 とは互換性がありません。

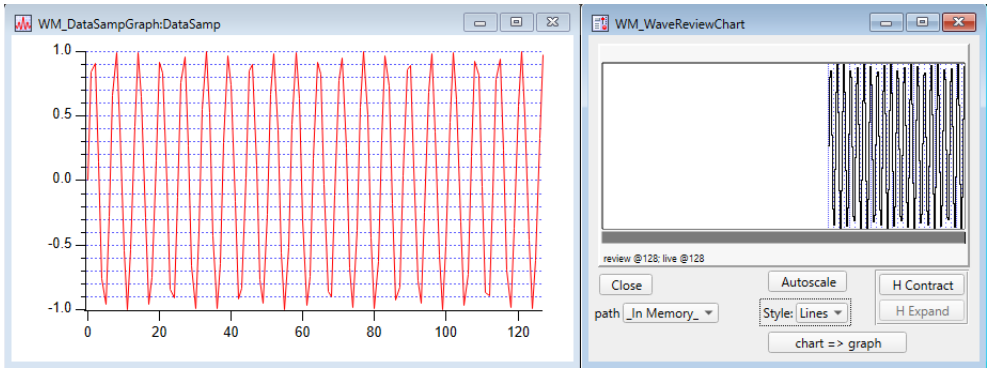
インクルードすると Graph メニューに Transform Axis という項目が追加されます。




```
#include <Wave Review Chart>
```

バイナリウェーブファイル（拡張子 .bwav）またはメモリ内のウェーブをスクロールして確認できるチャートを備えたコントロールパネルを作成します。
Igor FIFO ファイルの確認も可能です。

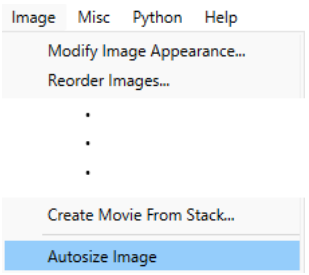


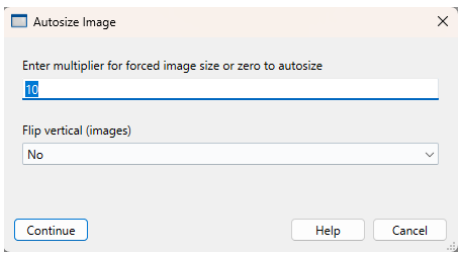


WaveMetrics 画像プロットプロシージャ

```
#include <Autosize Images>
```

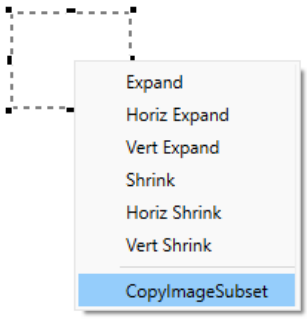
このパッケージを使うと、画像、特に写真のサイズを適切に調整するのが簡単になります。
Image メニューに Autosize Image が追加されます。
また、独自のコードで役立つかもしれない DoAutoSizeImage という関数も含まれています。





```
#include <CopyImageSubset>
```

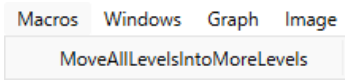
CopyImageSubset という選択範囲プロシージャが含まれています。
これは Macros メニューには表示されませんが、マーキーメニューには表示されます。
これを使うには、画像プロット上で選択範囲をドラッグして作成し、その範囲内をクリックして、ポップアップメニューから CopyImageSubset を選択します。
選択範囲内の画像の一部がコピーされ、新しい画像プロットに表示されます。

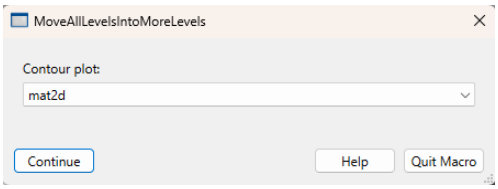


WaveMetrics コンタープロットプロシージャ

```
#include <Contour Levels>
```

MoveAllLevelsIntoMoreLevels プロシージャを実行すると、現在の自動、手動、またはウェーブからのすべてのレベルを More Levels ダイアログに移動させることができます。
そこで、レベルを再入力することで等高線レベルを調整することが可能です。

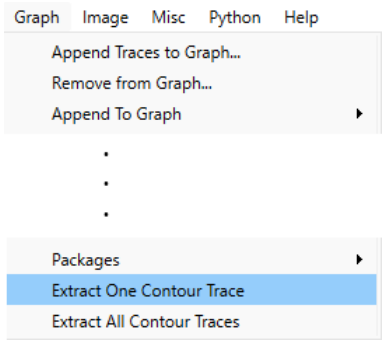


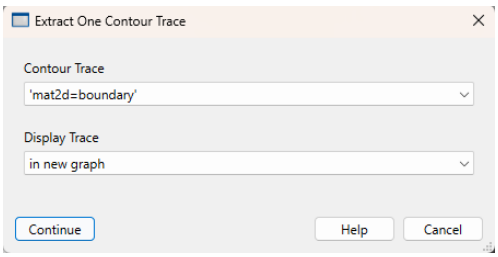



```
#include <Extract Contours As Waves>
```

トレースデータは、データフォルダーを持たない X ウェーブや Y ウェーブで構成されているため、分析や確認のためにアクセスすることはできません。
このプロシージャファイルは、Graph メニューにメニュー項目を追加し、トレースの X ウェーブと Y ウェーブのコピーを、現在のデータフォルダーに個別に、あるいはサブデータフォルダーに一括で抽出できるようにします。
抽出されたトレースは、新規または既存のテーブルやグラフに表示することができます。

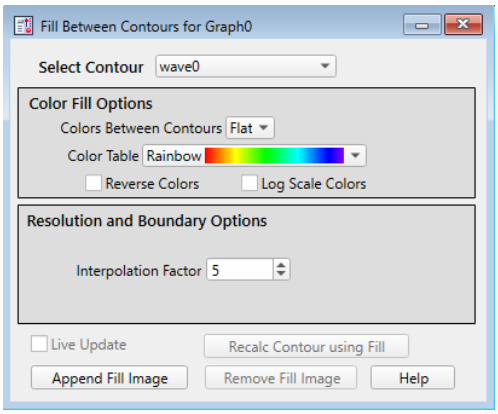
各 X-Y ウェーブペアには、１つのレベルに対応するコントートレース全体が含まれています。
トレースの連続していない部分は、NaN によって区切られています。
これらの NaN の位置を特定するには、付属の WMFindNaNValue() ルーチンを使ってください。

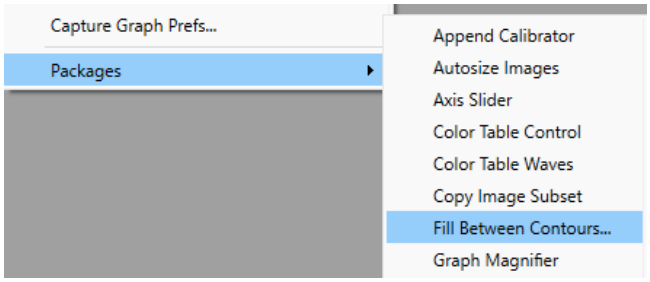




#include <FillBetweenContours>

等高線間の領域を塗りつぶす画像を作成するための、パネルベースの GUI を実装しています。
この GUI を呼び出す最も簡単な方法は、Graph→Packages→Fill Between Contours メニューを使うことです（別途、インクルードしなくても既にメニューに含まれています）。





（この方法を使って、不均一な x、y、z 値の行列から画像プロットを作成することもできます。画像の作成が完了したら、等高線プロットをグラフから削除することができます。）

これらのルーチンは、等高線レベル間で同じ値を持つ「スライス」画像を作成することができます。

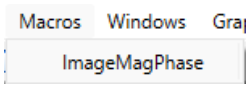
この補間ルーチンにより、等高線付近の外観が改善されますが、等高線自体については、次のコマンドを使ってさらに調整が必要になる場合があります。

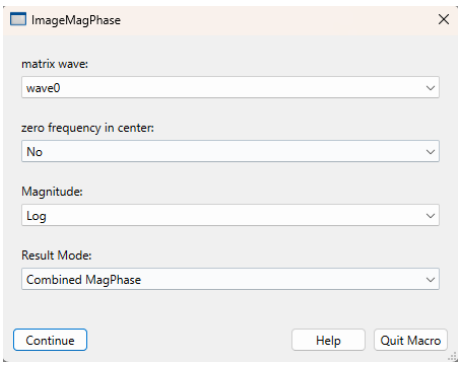
```
ModifyContour contourInstanceName  interpolate=2
```

詳細については、ヘルプ Fill Between Contours を参照してください。

#include <Image MagPhase>

行列データに対して 2D FFT を実行し、その結果を振幅画像と位相画像として個別に、あるいは振幅、位相の合成画像として表示する ImageMagPhase プロシージャが含まれています。





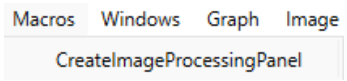
#include <WMImageInfo>

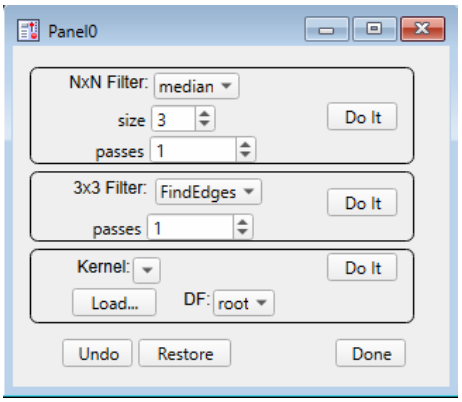
このプロシージャファイルには、画像プロットに関する情報を取得するための関数が含まれています。
含まれる関数には、
WM_GetColorTableMinMax　　WM_ColorTableForImage
WM_ColorTableReversed　　WM_ImageColorTabInfo
WM_ImageColorLookupWave　　WM_ImageColorIndexWave
があります。
（このプロシージャをインクルードしても GUI 上に変化はありません）

WaveMetrics 画像処理プロシージャ

```
#include <Image Processing Panel>
```

このパッケージは、ガウスぼかしやエッジ検出といった一般的な画像処理手法を簡単に実行できるコントロールパネルを作成します。






```
#include <ColorSpaceConversions>
```

このプロシージャファイルには、ある色空間から別の色空間へ変換するための関数が約 16 個含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

WaveMetrics 数学プロシージャ

```
#include <Math Utility Functions>
```

2 つの関数を含んでいます。

- log2(x) は 2 を底とする対数を返します
- CeilPwr2(x) は、x 以上で最小の 2 の累乗を返します

CeilPwr2 は、FFT 処理において、データを収容できる最小の 2 の累乗を計算するために使われます。
（このプロシージャをインクルードしても GUI 上に変化はありません）

WaveMetrics 統計プロシージャ

```
#include <AllStatsProcedures>
```

統計解析ツールを提供する一連のプロシージャファイルが含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

これらは、1 つの 1D ウェーブを自動解析する Ancilla1D パッケージ、ANOVA Power Panel、2 乗設計を解析するための twoSquare プロシージャ、そして以下のプロット機能で構成されています。

statsAutoCorrPlot()　　statsBoxPlot()
statsPlotHistogram()　　statsPlotLag()
statsProbPlot()

さらに、以下の便利な関数が含まれます。

WM_2MeanConfidenceIntervals()　　WM_2MeanConfidenceIntervals2()
WM_BernoulliCdf()　　WM_BinomialPdf()
WM_CIforPooledMean()　　WM_CompareCorrelations()

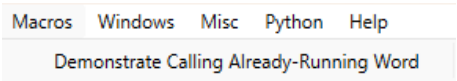
WM_EstimateMinDetectableDiff()
WM_EstimateReqSampleSize2()
WM_GetANOVA1Power()
WM_GetHarmonicMean()
WM_GetPooledVariance()
WM_MeanConfidenceInterval()
WM_OneTailStudentT()
WM_RankForTies()
WM_RegressionInversePrediction()
WM_SSEstimatorFunc()
WM_SSEstimatorFunc3()
WM_WilcoxonPairedRanks()
WM_EstimateReqSampleSize()
WM_EstimateSampleSizeForDif()
WM_GetGeometricAverage()
WM_GetPooledMean()
WM_MCPointOnRegressionLines()
WM_OneTailStudentA()
WM_PlotBiHistogram()
WM_RankLetterGradesWithTies()
WM_SetupANOVAPowerPanel()
WM_SSEstimatorFunc2()
WM_VarianceConfidenceInterval()
WM_WM_CronbachAlpha()

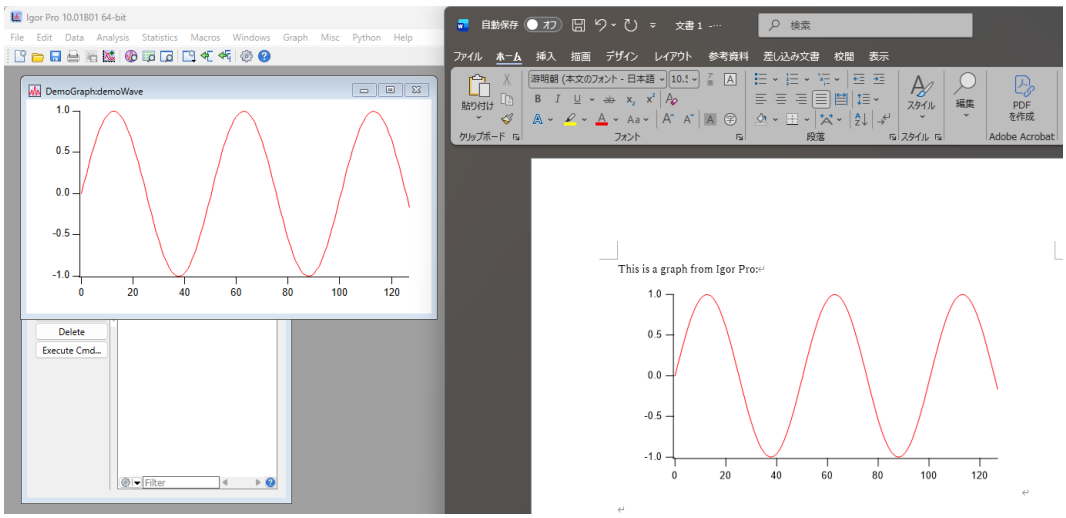
WaveMetrics ユーティリティプロシージャ

`#include <CallMicrosoftWord>`

Igor から Microsoft Word へコマンドやデータを送信するスクリプトファイルを生成、実行するためのユーティリティルーチンを提供します。
これは Windows Script Host に依存しており、Microsoft Windows でのみ利用可能です。

このプロシージャをインクルードすると、Macros メニューでデモを見ることができます。





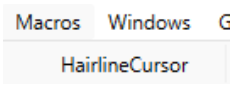
`#include <ControlBarManagerProcs>`

グラフのコントロールバーにコントロールを追加したり、削除したりする時に、適切な処理を行うための関数を定義します。
（このプロシージャをインクルードしても GUI 上に変化はありません）

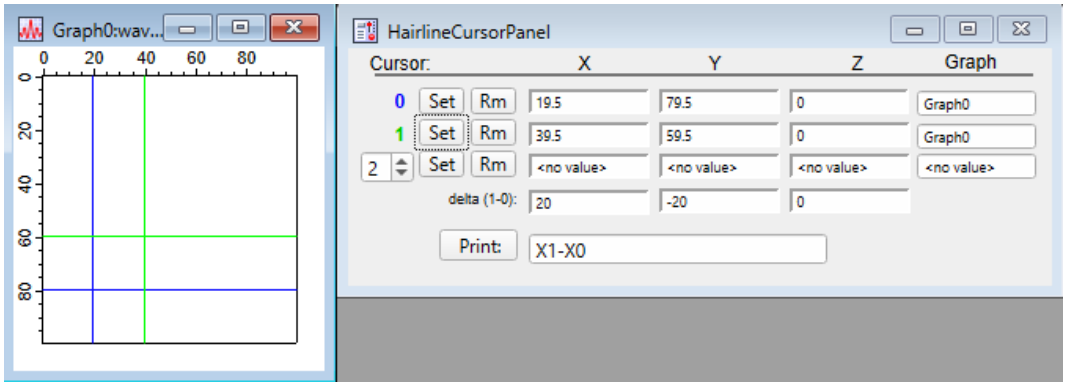
定義されている関数：
ExtendControlBar()　　ListControlsInControlBar()
MoveControls()　　GetControlBarHeight()

MoveControls() は、他の目的にも役立つ場合があります。

#include <Cross Hair Cursors>



グラフに十字カーソルを追加できるコントロールパネルを作成するパッケージです。
特に画像や等高線プロットで役立ちます。
特定の xy 座標における z 値を読み取ったり、差を確認したりすることもできます。



#include <CustomControl Definitions>

WMCustomControlAction 構造体内の CustomControl の eventCode メンバーに対する定数定義が含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Execute Cmd On List>

リスト内の各項目に対して指定されたコマンドを実行する関数 ExecuteCmdOnList(cmdTemplate, list) を提供します。
list は、通常はウェーブ名である項目をセミコロンで区切ったリストです。
cmdTemplate は、項目の位置に 「%s」 が挿入された Igor コマンドです。
注：cmdTemplate 内で %s は 1 回のみ使用できます。

ExecuteCmdOnQuotedList は、リストにウェーブファイル名やデータフォルダー名が含まれている場合に使う ExecuteCmdOnList のバリエーションです。
この関数は、リスト内のリベラル名（スペースを含む名前など）に一重引用符を追加します。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <FunctionProfiling>

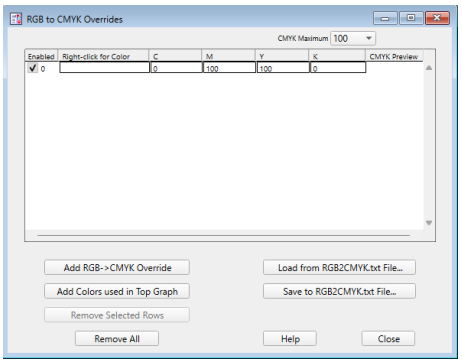
ユーザー関数のコード内のボトルネックを簡単に特定する方法を提供します。
使用方法については、ファイル先頭のコメントを参照してください。
（このプロシージャをインクルードしても GUI 上に変化はありません）

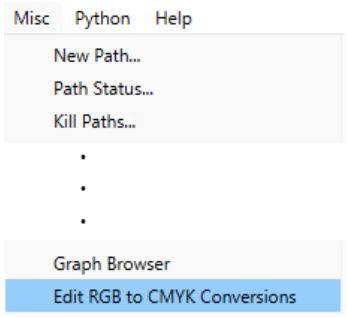
#include <HierarchicalListWidget>

Igor のリストボックスコントロールを汎用的な階層リストに変換するための基盤を提供するプロシージャです。
コンテンツのコントロールはクライアントコードが行います。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <IgorRGBtoCMYKPanel>

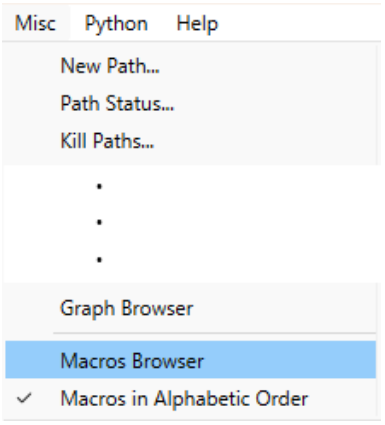
IgorRGBtoCMYKPanel プロシージャは、RGB グラフィックスを CMYK 形式でエクスポートする時に Igor が行う変換を上書きするための、テーブル形式のエディタを実装しています。
詳細については、ヘルプ Exporting Colors (Windows) を参照してください。

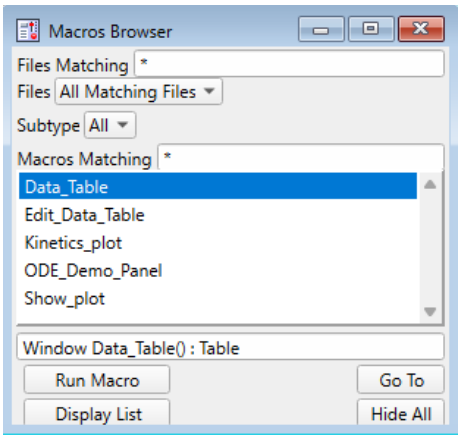




#include <MacrosBrowser>

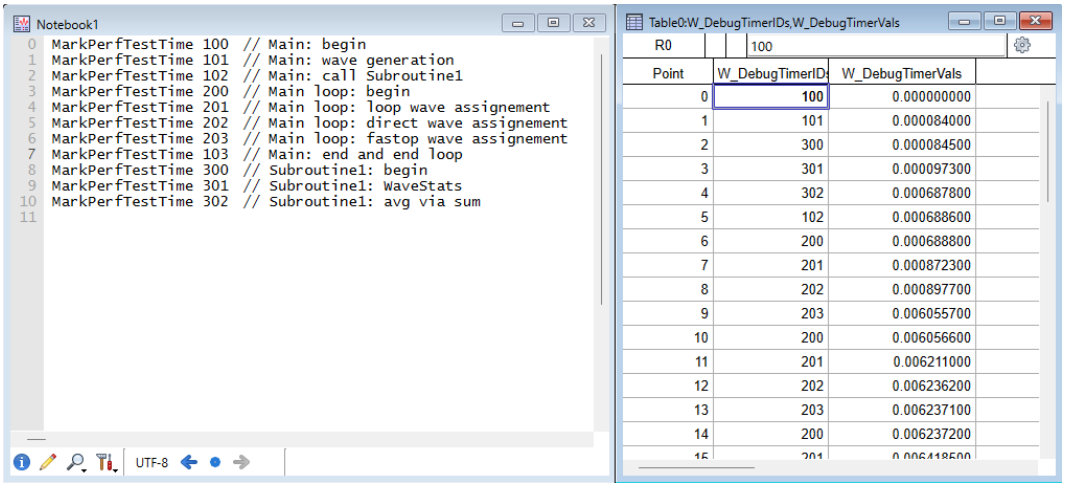
Misc メニューに Macros Browser 項目を追加し、現在のエクスペリメントに含まれるウィンドウマクロを一覧表示するパネルを表示します。
この機能を使うにはエクスペリメントをコンパイルする必要があるため、コードを編集中には動作しませんが、多数のウィンドウマクロを含む大規模なエクスペリメントにおいて、特定のウィンドウマクロを探す時に非常に役立ちます。
Subtype ポップアップと Macros Matching ワイルドカード検索を使ってマクロを検索し、Go To をクリックすると、そのマクロを含む Procedure ウィンドウが開きます（マクロ名をダブルクリックしても、Go To ボタンをクリックしたのと同じ動作になります）。
Display List をクリックすると、一致するプロシージャファイル内のすべての一致するマクロの一覧が表示されます。
Hide All をクリックすると、表示されているすべての Procedure ウィンドウを一括で非表示にできます。





#include <PerformanceTestReport>

新しいパフォーマンステストコマンド MarkPerfTestTime のサポートを提供します。
詳細については、プロシージャファイル内のコメントおよびサンプルエクスペリメント PerformanceTesting.pxp を参照してください。

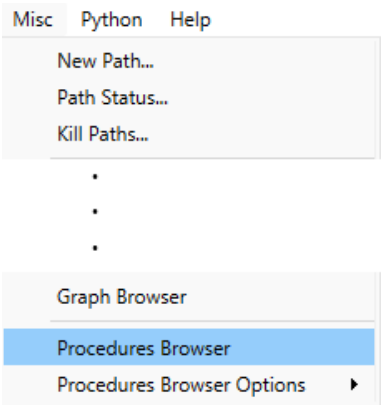


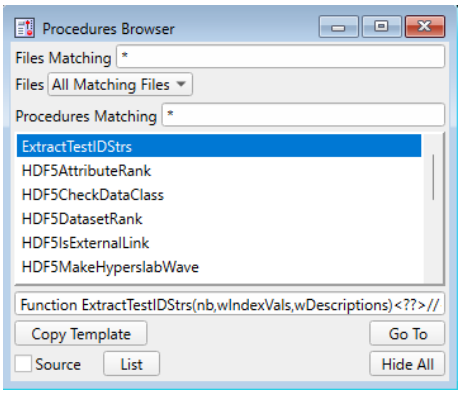
`#include <PopupWaveSelector>`

ボタンまたは SetVariable コントロールを、WaveSelectorWidget を含むコントロールパネルをポップアップ表示する「ポップアップメニュー」に変換するプロシージャです。
WaveSelectorWidget の説明については、以下を参照してください。
（このプロシージャをインクルードしても GUI 上に変化はありません）

`#include <ProcedureBrowser>`

Misc メニューに Procedure Browser 項目を追加し、現在のエクスペリメントに含まれる関数やマクロを一覧表示するパネルを表示します。
この機能を使うにはエクスペリメントをコンパイルしておく必要があるため、コードを編集中には動作しませんが、プロシージャファイルが多数ある大規模なエクスペリメントにおいて、特定のルーチンを探す時に非常に役立ちます。
Procedure Matching のワイルドカード検索機能を使ってルーチンを探し、その後 Go To をクリックすると、選択したルーチンを含む Procedure ウィンドウが開きます（ルーチンをダブルクリックしても、Go To ボタンをクリックするのと同じ動作になります）。
Display List をクリックすると、一致するプロシージャファイル内のすべての一致するルーチンのリストが表示されます。
Hide All をクリックすると、表示されているすべての Procedure ウィンドウを一括で非表示にできます。



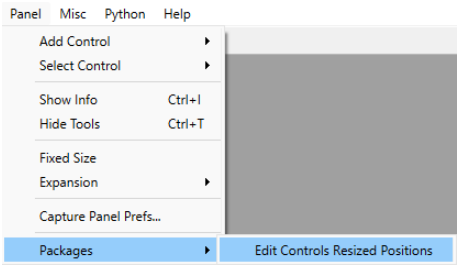


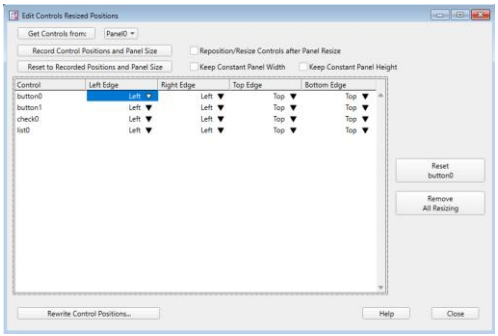
#include <Readback ModifyStr>

TraceInfo や AxisInfo から返される文字列の解析に役立つ文字列ユーティリティ関数 GetNumFromModifyStr(modstr, key, listChar, itemNo) が含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Resize Controls Panel>

Resize Controls Panel.ipf プロシージャファイルは、パネル内のコントロールのサイズ変更動作を設定するためのテーブル形式のエディターを実装しています。
パネルのサイズが変更されると、選択されたモードに従って、各コントロールの各辺が（オプションで）移動またはサイズ変更されます。





これは、リスト、スライダー、またはサブウィンドウを含むパネルで特に役立ちます。
コントロールは、パネルやサブウィンドウの端や中心、あるいはユーザーガイドを基準にして移動させることができます。

Resize Controls.ipf には、編集が完了した時点で必要最小限のコードが含まれています。

#include <Rewrite Controls Position>

Rewrite Controls Position.ipf プロシージャファイルは、Igor のプロシージャコード（関数またはマクロ）において、コントロールの位置、サイズ、タイトルを更新し、必要に応じて状態やユーザーデータの値を無効にする必要があるプログラマー向けに、コード書き換え機能を実装しています。

コードはノートブックウィンドウに書き換えられ、変更箇所は赤文字で強調表示されます。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <SetIgorMenuModeProc>

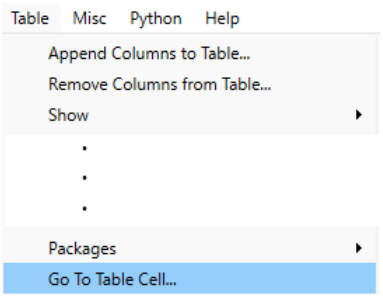
そのまま使うことを意図したものではありません。
すべてのメニューおよびメニュー項目に対して考えられるあらゆる SetIgorMenuMode コマンドが含まれています。
プログラムでのご自身のニーズに合わせて、これらから必要なものを選んで使ってください。
（このプロシージャをインクルードしても GUI 上に変化はありません）

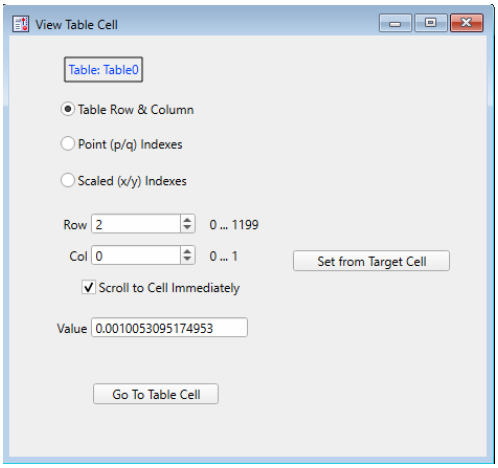
#include <TranslateChars>

文字列関数 TranslateChars(oldStr, oldChars, newChars) が含まれており、この関数は oldChars に含まれる文字を newChars 内の対応する文字に置き換えた後、oldStr を返します。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <ViewTableCell>

アクティブなテーブルをスクロールして特定の行や列を表示したり、列ウェーブの行/列/レイヤー/チャンクインデックス、または x/y/z/t のスケーリング値を表示したりできるコントロールパネルを作成するパッケージです。
セルの値を編集することも可能です。





#include <Value Report>

sigma に基づいて bval を丸めた値を表示する文字列ユーティリティ関数 MakeValueReportString(bval, sigma, method, unitsStr, flags) が含まれています。
これは、カーブフィッティングの結果を出力する時に使われます。
４種類の出力メソッドを利用できます。
例えば、メソッド#1 では、bval と sigma の両方の値を次のように丸めて出力します：b.bbb±s.sss。
テクニカルノート「TN001 Value report」を基にしています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <WalkFolders>

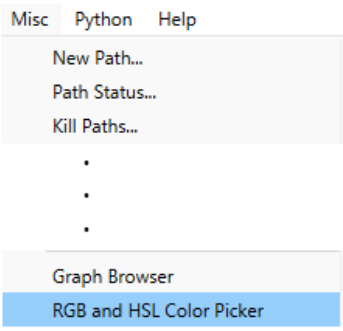
開始ディレクトリ内のフォルダー（およびオプションでファイル）に対して、指定した「コールバック」ルーチン（WM_WalkFoldersCallback の FUNCREF）を呼び出す WM_WalkFolders ルーチンを含みます。
recurse が true の場合、コールバック関数は内部のサブフォルダーに対しても呼び出されます。
（このプロシージャをインクルードしても GUI 上に変化はありません）

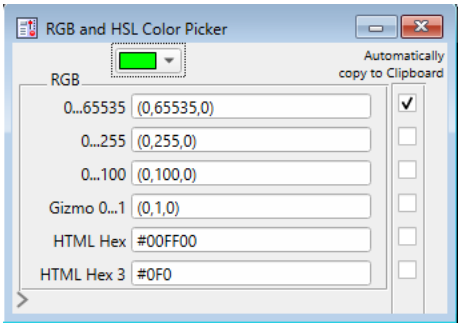
#include <WaveSelectorWidget>

リストボックスコントロールを、ウェーブ、文字列、変数、またはデータフォルダーの階層リストに変換するプロシージャです。
コントロールパネル内で、データフォルダーを意識したウェーブ、文字列、変数、またはデータフォルダーの選択を可能にします。
詳細については、プロシージャファイルのコメントを参照してください。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <WMColorPicker>

Misc メニューに RGB and HSL Color Picker というメニュー項目を追加します。
これにより、色を選択するためのカラーポップアップを備えたパネルが表示されます。
対応する RGB（赤、緑、青）、HSL（色相、彩度、明度）の値が表示されるため、コピーしてプロシージャ内で簡単に使用できます。
0 から 65535 までの RGB 値を表示していた Mac OS 9 のカラーピッカーを懐かしむ Mac OS X ユーザーは、このパネルを使って、Igor のコマンドで使うそれらの値を簡単に確認できます。



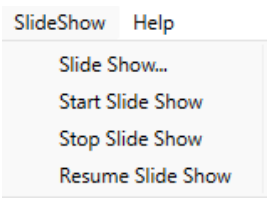


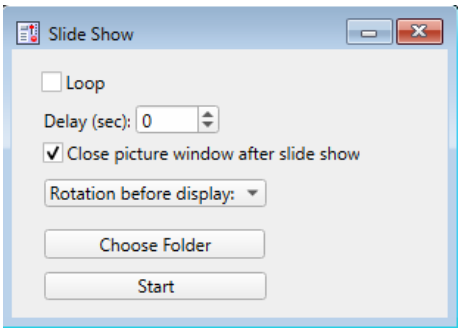
#include <WMDataBase Procs>

複雑なプロシージャパッケージのステータスや設定情報を管理するために使用できる、一連の文字列ユーティリティ関数が含まれています。
これは（旧式の）Polar Plot パッケージで使われており、その使用例として参照することができます。
現在では、データフォルダーやデータグループを格納するための関数内の構造体が利用可能になったため、この機能の有用性は低下しています。

#include <SlideShow>

このパッケージは、ハードディスク上の画像ファイルが含まれているフォルダーを選択し、その画像をスライドショーとして表示できるコントロールパネルを作成します。

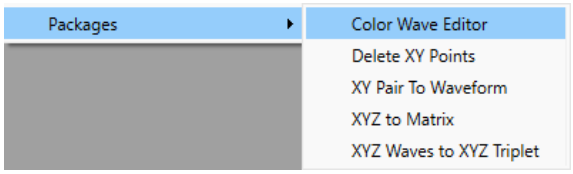


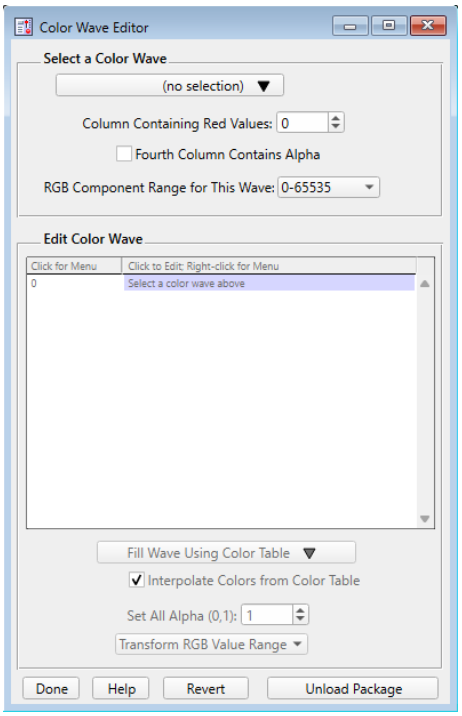


WaveMetrics ウェーブユーティリティプロシージャ

#include <ColorWaveEditor>

このプロシージャファイルは、Data メニュー内のパッケージです。
メニューを使うのが、このプロシージャファイルを読み込む最も簡単な方法です。
画像のインデックスカラーモードや、グラフのトレースにおける「Color as f(z)」によるダイレクトカラーなどで使われるような、３列（またはそれ以上）のカラーウェーブの色の値を簡単に編集する方法を提供します。





#include <Compare Waves>

２つのウェーブが同一であるか、あるいは２つのウェーブの特定の特性が同一であるかを確認できる関数が含まれています。
これは内部のテストに使っています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Kill Waves>

以下の関数が含まれています：

KillAllWaves	RemoveWavesFromWindows
RemoveWavesFromWindow	RemoveWavesFromTable
RemoveWavesFromGraph	KillMatchingWaves

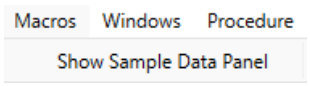
警告：このファイルに含まれるルーチンは、ウェーブファイルを破損させる可能性があります。
使用前に警告をよく読み、ルーチンの動作を十分に理解してください。
（このプロシージャをインクルードしても GUI 上に変化はありません）

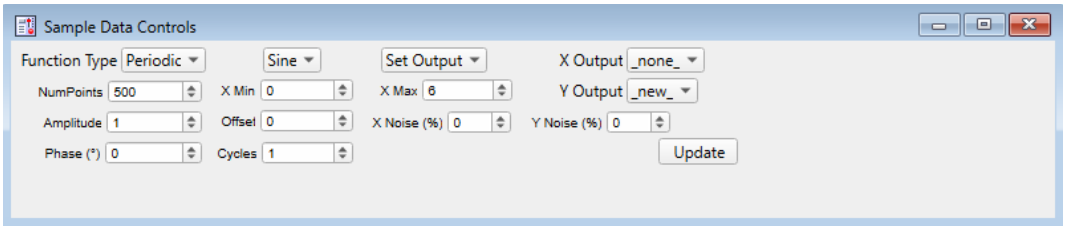
#include <Make Sample Data>

さまざまな種類のデータセットを使って試行できるルーチンを含んでいます。
多様なデータでルーチンをテストする必要がある場合にこれを使ってください。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <Make Sample Data Controls>

さまざまな種類のデータセットで試行ができるコントロールパネルを追加します。
多様なデータに対してルーチンをテストする必要がある場合にこれを使ってください。
このファイルは、「Make Sample Data」プロシージャ用の洗練されたインターフェイスを提供します。





#include <Wave Lists>

これは主に、ポップアップメニューのテキストを設定する時に役立ちます。
また、特定の種類のウェーブを選択するための、より強力な関数も用意されています。
具体的には：WaveListMatchWave、WaveListQualified、WaveNameIndexedExcluding です。

また、現在は非推奨となっている GraphWaveList 関数も含まれています。
この関数は、現在のデータフォルダー内にある限り、グラフ内の x ウェーブまたは y ウェーブのリストを返します。
より良い結果を得るには、他のルーチン、または組み込み関数である TraceNameList、TraceNameToWaveRef、XWaveRefFromTrace を使ってください。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <WaveType>

WaveType 組み込み関数が返す数値コードに従って、指定されたウェーブのリサイズを行う SetWaveType 関数が含まれています（WaveType は、かつてこのファイル内のユーザー定義関数でした）。
（このプロシージャをインクルードしても GUI 上に変化はありません）

WaveMetrics ウィンドウユーティリティプロシージャ

#include <BringDestToFront>

一部の DSP サポートプロシージャで主に使われるユーティリティプロシージャ、BringDestFront(w) が含まれています。
ウェーブ名を指定すると、そのウェーブを含む最前面のグラフを最前面に表示するか、そのウェーブを含むグラフを作成します。

ウェーブを含む最前面のグラフの名前を返す文字列関数 FindGraphWithWave(wave) が含まれています。
（このプロシージャをインクルードしても GUI 上に変化はありません）

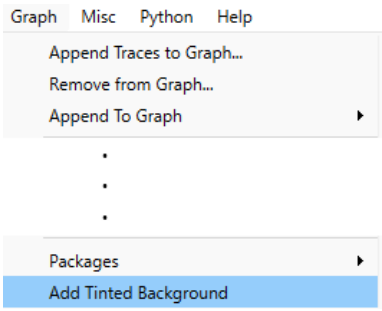
#include <SaveRestoreWindowCoords>

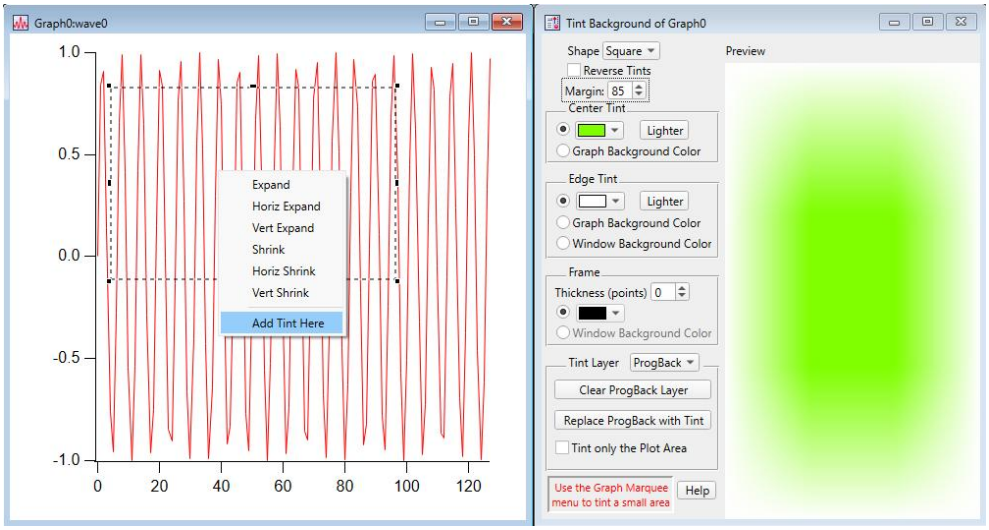
ウィンドウが閉じられた時の位置を記憶し、ウィンドウが再度作成された時にプログラムコードから元の位置に戻すことができる機能を追加します。
これらはすべて、ウィンドウマクロを使わずに実現できます。
これらの機能は、コントロールパネルなど、作成、移動、終了、再作成を繰り返すウィンドウに役立ちます。

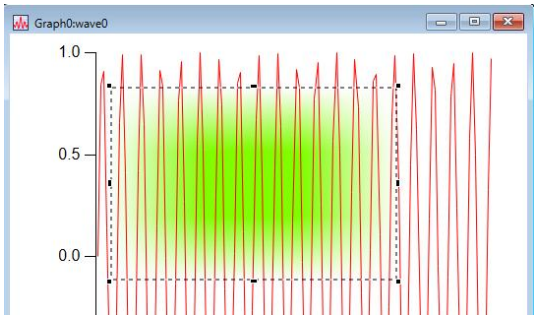
主な関数としては、WC_WindowCoordinatesHook、WC_WindowCoordinatesRestore、WC_WindowCoordinatesSprintf が挙げられます。
使用例については、SaveRestoreWindowCoords.ipf のコードを参照してください。
（このプロシージャをインクルードしても GUI 上に変化はありません）

#include <TintedWindowBackground>

Graph と Layout メニューに Add Tinted Background というメニュー項目を追加します。
これを選択するとパネルが表示され、ウィンドウの背景にシンプルなグラデーションや色付けを追加できるようになります。







#include <WindowBrowser>

ウィンドウ、サブウィンドウ、トレース、軸、ウィンドウマクロなどを階層リスト形式で表示するコントロールパネルを作成するパッケージです。
すでに標準機能としてメニュー Windows→Window Browser として組み込まれています。