CONTENTS

ビジュアルヘルプ – Waves

数値の配列を含む Igor オブジェクトを表すために、「ウェーブ」という用語を使います。
ウェーブは「ウェーブフォーム」の略語です。
Igor の主な目的は、ウェーブの保存、分析、変換、および表示です。
ヘルプ Introduction to Igor Pro（Getting Started.ihf）では、ウェーブに関する基本的な考え方を紹介しています。
ヘルプ Getting Started.ihf ファイルにある「Guided Tour（ガイドツアー）」セクションでは、これらの考え方を理解していただけるように作成されています。
このセクションでは、これらの概念について基本的な理解があることを前提としています。

このセクションでは、１次元数値ウェーブについて説明します。
ウェーブは最大４次元まであり、テキストデータも格納することができます。
多次元ウェーブについては、ヘルプ Multidimensional Waves で説明しています。
テキストウェーブについては、このヘルプファイルで説明します。

ウェーブを扱うための主なツールは、Igor の組み込み演算および関数、およびウェーブフォームの代入機能です。
組み込み演算および関数の詳細については、ヘルプ Igor Pro Reference（Igor Reference.ihf）で説明しています。

このセクションでは次のことを説明しています：
- ウェーブの概要
- ウェーブの作成、Kill（削除）、管理の操作
- ウェーブのプロパティの設定と検証
- ウェーブフォームの代入
- その他のトピック

データのウェーブフォームモデル

ウェーブは、いくつかのコンポーネントとプロパティで構成されています。
最も重要なものは次のとおりです。
- ウェーブ名
- X スケーリングのプロパティ
- X の単位
- データ値の配列
- データの単位

データのウェーブフォームモデルは、ポイント番号インデックスから X 値への直線的なマッピング、つまり、データが X 次元で等間隔に配置されていることを前提としています。
これは、多くの種類の科学・工学機器から取得されたデータおよび数学的に合成されたデータに当てはまります。
データが等間隔で配置されていない場合は、2 つのウェーブを使用して XY ペアを形成することができます。
次の「データの XY モデル」セクションを参照してください。

ウェーブは、他の標準的なプログラミング言語の配列に似ています。

配列（array0）　　ウェーブ（wave0）

インデックス	値
0	3.74
1	4.59
2	4.78
3	5.89
4	5.66

ポイント番号	X 値	データ値
0	0	3.74
1	.001	4.59
2	.002	4.78
3	.003	5.89
4	.004	5.66

標準的な言語における配列には、名前（この例では array0）と複数の値が含まれます。
特定の値を参照するには、インデックスを使います。

ウェーブにも名前（この場合は wave0）とデータ値があります。
配列との違いは、２つのインデックスがあることです。
１つ目はポイント番号と呼ばれ、配列のインデックスまたは行番号と同じです。
２つ目は X 値と呼ばれ、データの自然な X 単位（秒、メートルなど）で表されます。
ポイント番号と同様に、X 値もメモリには保存されず、Igor によって計算されます。

X 値はそのポイント番号と、ウェーブの X スケールに関連付けられています。
X スケールは、ウェーブに設定できるプロパティです。
ウェーブの X スケールは、次の式を使って、指定されたポイント番号の X 値を計算する方法を指定します。

```
x[p] = x0 + p*dx
```

x[p] は、ポイント p の X 値です。
２つの数値 x0 と dx は、ウェーブの X スケーリングプロパティを構成します。
x0 は、開始 X 値です。
dx は、あるポイントから次のポイントまでの X 値の差です。
X 値は、データの X 軸に沿って等間隔で配置されています。

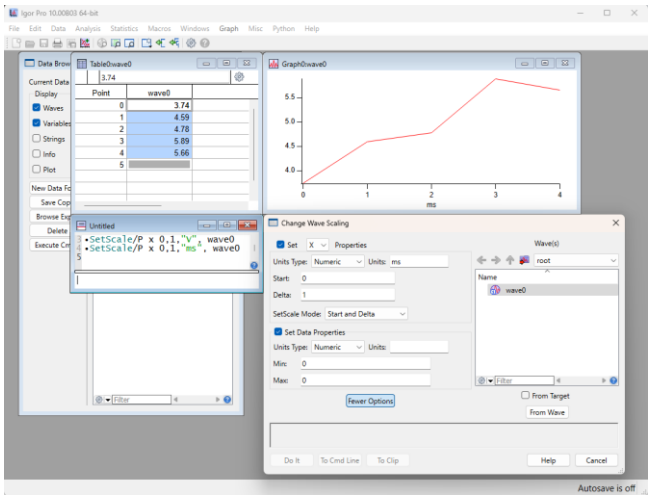
Igor の SetScale コマンドは、ウェーブの X スケーリングを設定します。
Change Wave Scaling ダイアログ（メニュー Data → Change Wave Sca;ing）を使って、SetScale コマンドを生成することができます。

Igor はなぜこのモデルを使用してデータを表現するのでしょうか？
このモデルを採用したのは、ウェーブフォームデータを適切に表示、分析、変換するために必要なすべての情報を提供するためです。

データの X スケーリング、およびデータ値に加えて X 単位とデータ単位を指定することで、１ステップで適切なグラフを作成することができます。
Igor に次のように指示することができます。

```
Display wave0
```

すると、次のようなグラフを表します。



データが X 軸上に均一に配置されている場合、X 軸のスケールを理解し適切に適用することが重要です。

X 方向のスケール情報は、積分、微分、フーリエ変換などの操作や、面積関数などの関数において不可欠です。
また、自然単位を使用して単一の値または値の範囲を参照できるため、ウェーブフォームの割り当ても簡略化されます。

Igor のウェーブは、最大４次元まで持つことができます。
これらの次元を X、Y、Z、T と呼びます。
X スケーリングは、次元の拡大／縮小に相当します。

各次元には、開始インデックス値 （x0、y0、z0、t0） とデルタインデックス値 （dx、dy、dz、dt） があります。
多次元ウェーブの詳細については、ヘルプ Multidimensional Waves（Multidimensional Waves.ihf）を参照してください。

データの XY モデル

データが X 軸に沿って等間隔に配置されていない場合、単一のウェーブでは表現できません。
XY ペアとして 2 つのウェーブを使う必要があります。

XY ペアでは、一方のウェーブのデータ値が X 値となり、もう一方のウェーブのデータ値が Y 値となります。
両方のウェーブの X スケールは関係ないため、x0 と dx 成分が 0 と 1 であるデフォルトの状態のままにしておきます。
これにより次のようになります。

```
x[p] = 0 + 1*p
```

これは、特定の点の X 値はそのポイント番号と同じであることを意味します。
これを「ポイントスケーリング」と呼びます。
以下は、ポイントスケーリングが適用されたサンプルデータです。

X ウェーブ

ポイント番号	X 値	データ値
0	0	0.0
1	1	0.013
2	2	0.021
3	3	0.029
4	4	0.042

Y ウェーブ

ポイント番号	X 値	データ値
0	0	3.74
1	1	4.59
2	2	4.78
3	3	5.89
4	4	5.66

X 値は XY モデルにおいて何の役割も果たしません。
したがって、考え方を変えて、XY ペアを次のように見直します。

X ウェーブ

ポイント番号	データ値
0	0.0
1	0.013
2	0.021
3	0.029
4	0.042

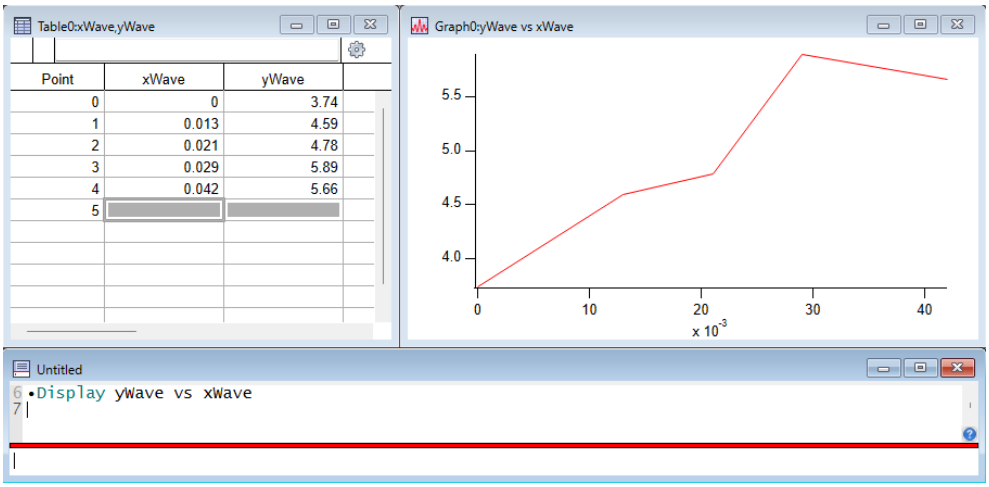
Y ウェーブ

ポイント番号	データ値
0	3.74
1	4.59
2	4.78
3	5.89
4	5.66

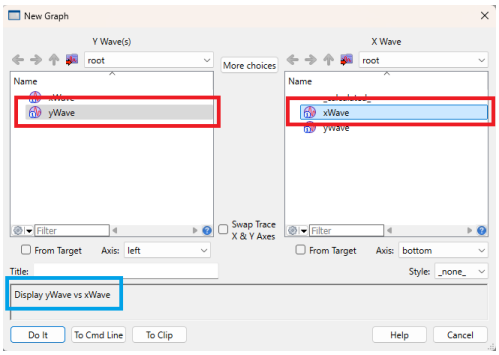
次のコマンドを実行します。

```
Display yWave vs xWave
```

次のようなグラフが得られます。



GUI では次のように操作します。
メニュー Windows → New Graph を選択します。New Graph ダイアログで Y Wave で yWave、X Wave で xWave を選択します。画面下には、上記のコマンドと同じものが生成されます。
Do It をクリックして実行します。



一部の操作（例：高速フーリエ変換や畳み込み）では、等間隔に配置されたデータが必要です。
このような場合、補間によって、データの間隔を均一にしたバージョンを作ることを推奨します。
ヘルプ Converting XY Data to a Waveform（Analysis.ihf）を参照してください。

一部のユーザーは、データが均一に配置されている場合でも、慣れているため XY モデルを使い続けています。これは間違いです。
データが等間隔である場合は、ウェーブフォームモデルを習得して使うことをお勧めします。
グラフ化と分析を大幅に簡略化し、Igor プロシージャの記述も容易になります。

ウェーブの作成

次の方法でウェーブを作成できます：

- ファイルからデータをロードする
- テーブルで入力、またはペーストする
- Make コマンドを使う（ダイアログを使う、またはコマンドラインで直接入力）
- Duplicate コマンドを使う（ダイアログを使う、またはコマンドラインで直接入力）

ほとんどの人は、ファイルからデータをロードすることから始めます。
Igor はテキストファイルからデータをロードすることができます。
この場合、ファイル内のテキストの各列に対してウェーブを作成します。
バイナリファイルや他のプログラムで作成されたアプリケーション固有のファイルからもデータをロードすることができます。
ファイルからデータをインポートする方法については、ヘルプ Importing Data（Importing and Exporting Data.ihf）を参照してください。

テーブルにデータを手動で入力することができます。
この方法は、データ量が少ない場合にのみお勧めします。
ヘルプ Using a Table to Create New Waves（Tables.ihf）を参照してください。

数式でデータを合成するには、まず Make コマンドを使ってウェーブを作成します。
このコマンドは、一時的に使うウェーブを作成するために、プロシージャ内でもよく使われます。

Duplicate コマンドは、重要で便利なツールです。
Igor の組み込みコマンドの多くは、その場にあるデータを変換します。
したがって、元のデータと変換後のコピーの両方を保持したい場合は、Duplicate を使って元のデータのクローンを作成してください。

ウェーブ名

すべてのウェーブには名前が付いているため、コマンドから参照することができます。
また、ダイアログのリストやポップアップメニューからウェーブを選択したり、ウェーブフォームの割り当て文でウェーブを参照したりする場合にも、ウェーブの名前を使います。

ダイアログ、コマンドライン、または Data Browser を使って、Make、Duplicate、Rename コマンドを実行する場合は、ウェーブ名を選択する必要があります。

Igor のすべての名前は、大文字と小文字を区別しません。
wave0 と WAVE0 は、同じウェーブを指します。

ウェーブ名に使用できる文字の種類に関するルールは、標準とリベラルの 2 つのカテゴリに分類されます。
標準名とリベラル名はいずれも、長さは 255 バイトまでに制限されています。

Igor Pro 8.0 以前では、ウェーブ名は 31 バイトに制限されていました。
長いウェーブ名を使う場合は、ウェーブファイルおよび Experiment ファイルに Igor Pro 8.0 以降が必要です。

標準名は、アルファベット文字（A~Z または a~z）で始まり、ASCII アルファベット文字、数値文字、およびアンダーバー文字のみを含めることができます。
スペース、ハイフン、ピリオド、および非 ASCII 文字を含むその他の文字は使用できません。
この制限は、ウェーブフォームの代入文を含むコマンドで名前を明確に識別できるように、標準名に対して設定されています。

一方、リベラル名には、制御文字（タブやキャリッジリターンなど）と次の 4 文字を除く、あらゆる文字を使用できます：

```
"    '    :    ;
```

標準名は、コマンドや式内で引用符なしで直接使うことができますが、リベラル名は必ず引用符で囲む必要があります。
例えば、

```
Make wave0; wave0 = p                    // wave0 は標準名
Make 'wave 0'; 'wave 0' = p              // 'wave 0' はリベラル名
```

引用符で囲まれていない限り、コマンド内のリベラルな名前を明確に識別できません。
例えば、

```
wave0 = miles/hour
```

では、miles/hour は、 1 つのウェーブの場合もあれば、 2 つのウェーブの商の場合もあります。

それらを明確にするために、コマンドやウェーブフォームの数式で使う場合は、リベラルな名前をシングルクオートで囲む必要があります。
例えば、

```
wave0 = 'miles/hour'
Display 'run 98', 'run 99'
```

注記： リベラル名で動作するプロシージャを記述するには、Igor プログラマは追加の作業とテストが必要になります （ヘルプ Programming with Liberal Names（Programming Techniques.ihf）を参照）。
潜在的な問題とその解決方法について理解するまで、リベラル名を使うことを避けることをおすすめします。

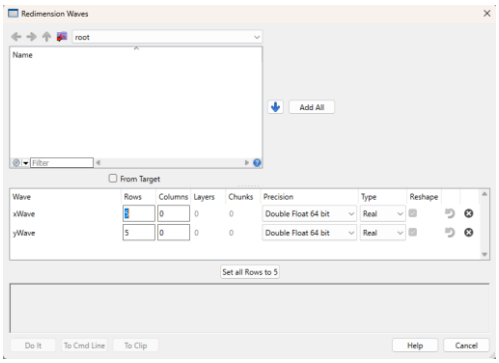
オブジェクト名に関する一般的な説明は、ヘルプ Object Names（Using Igor.ihf）を参照してください。

次元の数

ウェーブは１次元から４次元まで作成できます。
これは、ウェーブを作成するときに決定します。
この設定は、Redimension コマンドで変更できます。
詳細については、ヘルプ Multidimensional Waves（Multidimensional Waves.ihf）を参照してください。

ウェーブのデータ形式

各ウェーブには、そのウェーブが格納するデータの種類のデータ形式が設定されています。
ウェーブのデータ形式は、ウェーブを作成するときに設定します。
Data Browser、Redimension コマンド、または Redimension Waves ダイアログを使って変更できます。



ウェーブのデータ形式には３つのクラスがあります。

- 数値データ形式
- テキスト
- 参照（ウェーブ参照とデータフォルダー参照）

各数値データ形式は、実数型または複素数型のいずれかです。
テキストデータ形式と参照データ形式は、複素数型にはできません。
参照データ形式はプログラミングにおいてのみ使用されます。

WaveType 関数を使って、ウェーブのデータ形式をプログラムで決定することができます。

数値ウェーブのデータ形式

この表は、Igor で使うことができる数値ウェーブのデータ形式を示しています。

データ形式	範囲	ポイント毎のバイト
倍精度浮動小数点	≅15 桁の十進数	8
単精度浮動小数点	≅7 桁の十進数	4
符号付き 64bit 整数	-2^63 ～ 2^63 – 1	8
符号付き 32bit 整数	-2,147,483,647 ～ 2,147,483,648	4
符号付き 16bit 整数	-32,768 ～ 32,767	2
符号付き 8bit 整数	-128 ～ 127	1
符号なし 64bit 整数	0 ～ 2^64 - 1	8

符号付き 32bit 整数	0 ～ 4,294,967,295	4
符号付き 16bit 整数	0 ～ 65,535	2
符号付き 8bit 整数	0 ～ 255	1

64bit 整数型は、Igor Pro 7.0 で追加されました。

ほとんどの作業では、単精度ウェーブが適しています。

単精度ウェーブは、倍精度の半分だけのメモリとディスク容量しか使用しません。
FFT および一部の特殊な演算を除き、Igor はソースウェーブの数字の精度に関係なく、計算には倍精度を使います。
しかし、単精度のダイナミックレンジが狭く精度が低いことは、すべてのデータに適しているわけではありません。範囲や精度の制限による数値誤差についてよく知らない場合は、解析には倍精度を使うことをお勧めします。

整数ウェーブは、データ取得を目的としており、分析には使用できません。
詳細については、「整数ウェーブ」セクションを参照してください。

デフォルトのウェーブのプロパティ

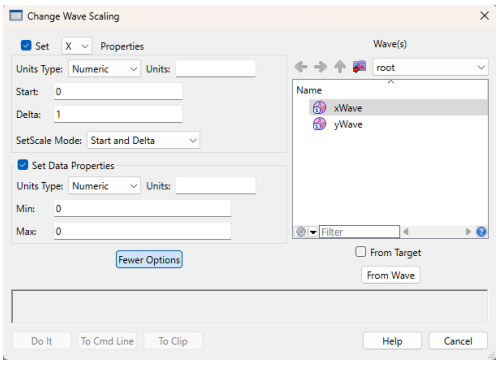
オプションフラグを指定せずに Make コマンドを使ってウェーブを作成すると、そのウェーブには以下のデフォルトのプロパティが設定されます。

プロパティ	デフォルト
ポイント数	128
データ形式	実数、単精度浮動小数点
X スケーリング	x0=0, dx=1（ポイントスケーリング）
X 単位	空
データ単位	空

これらは、ウェーブの主なプロパティです。
プロパティの包括的なリストについては、「ウェーブのプロパティ」セクションを参照してください。

ファイルから読み込んだり、テーブルに入力してウェーブを作成した場合、ポイント数以外はデフォルトのプロパティがすべて同じになります。

ウェーブの作成方法にかかわらず、XY ペアではなくウェーブフォームを表す場合は、Change Wave Scaling ダイアログ（Data メニュー）を使って、X スケーリングと単位を設定する必要があります。



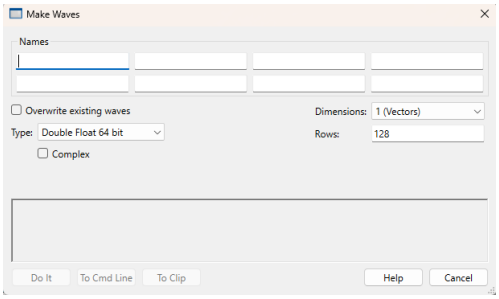
Make コマンド

ほとんどの場合、ファイルからデータをロードして（「データのインポート」を参照）、テーブルに入力して（「テーブルを使って新しいウェーブを作成する」を参照）、または既存のウェーブを複製して（「Duplocate コマンド」を参照）、ウェーブを作成することになります。

Make コマンドは、新しいウェーブを作るために使われます。
詳細については、ヘルプ Make（Igor Reference.ihf）を参照してください。

Make を使う理由の一部を以下に示します。

- 操作の学習のためのウェーブを作る
- 数学関数をプロットする
- 解析操作の結果を保持する
- カーブフィッティングで使われるパラメーターや、プロシージャ内の一時的な結果など、さまざまなデータを保持する



Make Waves ダイアログは、作成操作を行うためのインターフェイスです。
これを使うには、Data メニューから Make Waves を選択します。

ウェーブには明確なポイント数があります。
列の末尾の空白セルを自動的に無視するスプレッドシートプログラムとは異なり、Igor には「未使用ポイント」という概念はありません。
ウェーブのポイント数は、Redimension Waves ダイアログまたは Redimension コマンドで変更できます。

「Overwrite existing waves」オプションは、作成しようとしているウェーブと同じ名前のウェーブがあるかどうか分からない場合や、そのことを気にする必要がない場合に便利です。

Make コマンドの例

カーブフィッティングで使う係数を作る：

```
Make/O coefs = {1.5, 2e-3, .01}
```

数学関数をプロットするためのウェーブを作る：

```
Make/O/N=200 test; SetScale x 0, 2*PI, test; test = sin(x)
```

画像やコンタープロットのための 2D ウェーブを作る：

```
Make/O/N=(20,20) w2D; w2D = (p-10)*(q-10)
```

カテゴリプロットのためのテキストウェーブを作る：

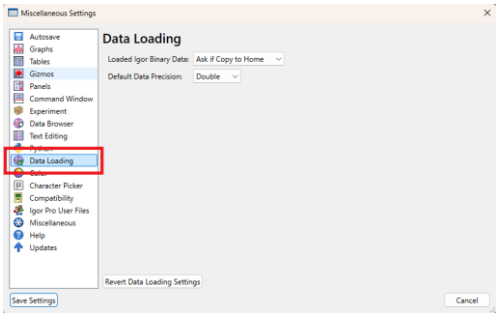
```
Make/O/T quarters = {"Q1", "Q2", "Q3", "Q4"}
```

既存のウェーブのクローンを作ることは、しばしば役に立ちます。
そのためには Make を使わないでください。
代わりに、Duplicate コマンドを使ってください。

Make/O はウェーブの内容を保持しません。
例えば、ポイント数、数値の精度、数値のタイプを変更すると、ウェーブにゴミが残ります。
したがって、Make/O を実行した後は、ウェーブの内容について何かを想定してはいけません。
ウェーブが存在することがわかっている場合は、Make の代わりに Redimension コマンドを使用できます。
Redimension はウェーブの内容を保持します。

ウェーブと Miscellaneous Settings ダイアログ

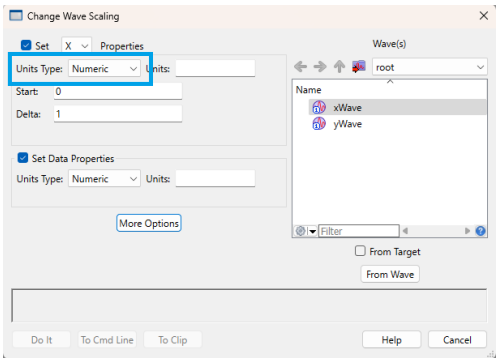
Make Waves ダイアログの Type ポップアップの状態、テーブルに入力して作成するウェーブの精度、および Igor バイナリウェーブの読み込み方法 （コピーするか共有するか） は、Miscellaneous Settings ダイアログの Data Loading カテゴリを使って設定されます。



次元とデータスケーリングの変更

1D ウェーブを作成すると、デフォルトの X スケール、X 単位、およびデータ単位が設定されます。
これらのプロパティを変更するには、SetScale コマンドを使ってください。

Change Wave Scaling ダイアログは、SetScale コマンドのインターフェイスです。
これを使うには、Data メニューから Change Wave Scaling を選択します。



スケーリングされた次元インデックスは、通常の数値、日付、時刻、または日付と時刻の値を表すことができます。
一般的な場合、これらは通常の数値を表し、ダイアログの Set X Properties セクションにある Units Type ポップアップメニューはデフォルト値の Numeric のままにしておいてください。

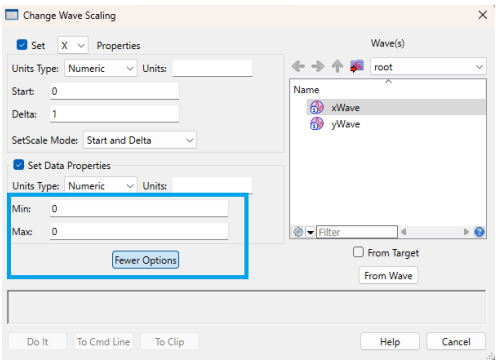
データがウェーブフォームデータの場合は、適切な開始と X デルタの値を入力してください。
データが XY データの場合は、Start に 0、Delta に 1 を入力してください。
これにより、デフォルトの「ポイントスケーリング」が適用され、ポイントの X 値はポイント番号と同一になります。

通常は、Set X Properties と Set Data Properties のチェックボックスを選択したままにしておいてください。
X または Data プロパティのみを設定するコマンドを生成したい場合は、そのうちの 1 つを選択解除してください。
多次元データを扱う場合、Set X Properties の X は、ポップアップメニューから Y、Z、または T に変更できます。
ヘルプ Multidimensional Waves（Multidimensional Waves.ihf）を参照してください。

特定のウェーブのプロパティを確認したい場合は、リストでそれをダブルクリックするか、ウェーブを選択して From Wave ボタンをクリックしてください。
そのウェーブのプロパティに応じて、すべてのダイアログ項目が設定されます。

Igor は、次元とデータ単位を使って、グラフの軸に自動的にラベルを付けます。
49 バイト以下の単位を扱うことができます。
通常では、単位は「m」、「s」、「g」などの短い標準的な略語を使用してください。
データに複雑な単位が含まれている場合は、その複雑な単位を入力するか、単位を空白のままにしておくことをお勧めします。

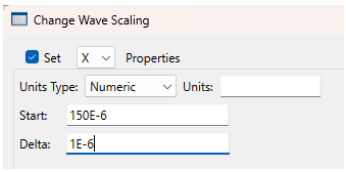
高度な次元とデータスケーリング



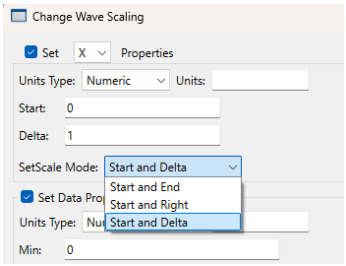
More Options をクリックすると、ダイアログにいくつかの追加項目を表示します。
これらは、X スケーリングを指定する 2 つの追加の方法を提供し、ウェーブの「データフルスケール」値を設定することができます。
これらのオプションは通常は必要ありませんが、完全を期すためにこのセクションで説明します。

X 値を計算する方法は 1 つしかありませんが、x0 と dx の値を指定する方法は 3 つあります。

SetScale Mode ポップアップメニュー（More Options ボタンをクリックすると表示）を使って、上記のスケール設定の意味を変更します。
最も簡単な方法は、x0 と dx を直接指定することです。



これはダイアログ内の Start and Delta モードであり、スケーリングを設定する唯一の方法は、このモードを選択することです。
ただし、More Options ボタンをクリックした場合を除きます。
例として、デジタイザーが t=0 から 150 µs 後に 1 MHz のサンプリングレートで取得したデータがある場合、Start には 150E-6、Delta には 1E-6 を入力します。



X スケーリングを指定する他の 2 つの方法は、開始値と終了値を指定し、ポイント数から dx を計算するようにすることです。
Start and End モードでは、最後のデータポイントの X 値を指定します。
Start and Right モードを使うと、最後の区間の終了位置の X 座標を指定できます。

例えば、デジタイザーが 100 ポイントのウェーブを作成したと仮定すると、どちらのモードでも Start に 150E-6 を入力します。

Start and End モードを選択した場合、End には 249E-6（150E-6 + 99×1E-6）を入力します。
Start and Right モードを選択した場合、Right には 250E-6 を入力します。

min および max エントリを使うと、ウェーブの「データフルスケール」と呼ばれるプロパティを設定できます。
このプロパティは重要な目的には使われません。
Igor は、計算やグラフ作成の目的でこれを使うことはありません。
これは、ウェーブデータを取得した条件を記録するための単なる手段です。
例えば、データがデジタルオシロスコープから取得され、±10V の範囲で測定された場合、最小値に-10、最大値に+10 を入力できます。
ウェーブを作成すると、これらは両方とも最初は 0 に設定されます。
データに意味のあるフルスケールがある場合は、それらを適切に設定してください。
それ以外の場合は、0 のままにしておいてください。
一方、データ単位は、次元単位と同様にグラフ作成のために使われます。

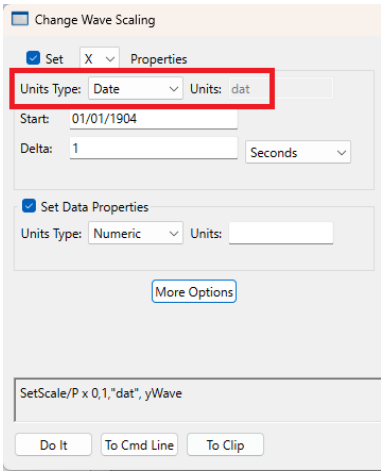
日付、時刻、日付＆時刻の単位

単位「dat」は特殊です。
これは、ウェーブのスケールされた次元インデックスまたはデータ値に、日付、時刻、または日付と時刻の情報が含まれていることを Igor に伝えます。

ウェーブフォームデータがある場合は、ウェーブフォームの X 単位を「dat」に設定してください。

XY データがある場合は、X ウェーブのデータ単位を「dat」に設定してください。
この場合、日付を正確に表現するのに十分な精度を得るためには、X ウェーブは倍精度浮動小数点である必要があります。

例えば、１日に１回測定される量が含まれるウェーブフォームがある場合、ウェーブの X 単位を「dat」に設定し、開始 X 値を最初の測定が行われた日付に設定し、Delta X 値を１日に設定します。
Units Type ポップアップメニューから Date を選択すると、X 単位が「dat」に設定されます。
開始値は、Igor が内部で日付を表現する形式である 1904 年 1 月 1 日からの秒数ではなく、日付として入力することができます。
ウェーブフォームをグラフ化すると、X 単位が「dat」であることを認識し、X 軸に日付が表示されます。



ウェーブフォームではなく XY ペアの場合、ダイアログの Set Data Properties セクションの Units Type ポップアップメニューから Date を選択して、X ウェーブのデータ単位を「dat」に設定します。
XY ペアをグラフ化すると、X ウェーブに日付が含まれていることを認識し、X 軸に日付を表示します。

Units Type ポップアップメニューは、ウェーブのどのプロパティにも直接対応していません。
つまり、ウェーブには単位形式のプロパティはありません。
これらのメニューは、ダイアログが適切な形式で値を表示できるように、処理する値の種類を伝えるだけです。

日付をどのように表現するかについては、「日付/時刻のウェーブ」セクションを参照してください。

テーブル内の日付と時間に関する情報は、ヘルプ Date/Time Formats（Importing and Exporting Data.ihf）を参照してください。

グラフの日付と時間に関する情報は、ヘルプ Date/Time Axes（Graphs.ihf）を参照してください。

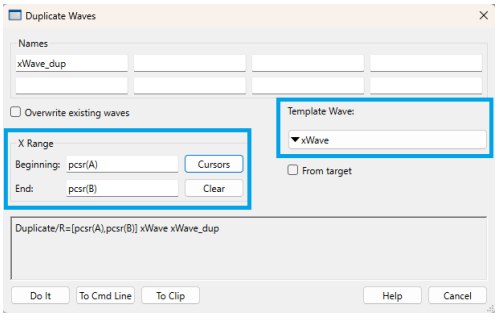
Duplicate（複製）コマンド

Duplicate は、便利で頻繁に使われるコマンドです。
既存のウェーブとまったく同じ新しいウェーブを作ることができます。
また、ウェーブの一部を複製することができるため、大きなウェーブを小さなウェーブに分割する簡単な方法も提供しています。

以下は Duplicate を使う理由です。

- 変換の結果（例：積分、微分、FFT）を保持しつつ、元のデータを保持する。
- カーブフィッティングの「宛先」を保持する。
- プロシージャ内で一時的な結果を保持する。
- ウェーブの一部を抽出する。

Duplicate Waves ダイアログは、Duplicate コマンドを行うためのインターフェイスです。
これを使うには、Data メニューから Duplicate Waves を選択します。
Cursors ボタンは、グラフと組み合わせて使います。
テンプレートのウェーブをグラフ化することができます。
次に、抽出したいテンプレートのセクションにカーソルを置きます。
Data メニューから Duplicate Waves を選択し、Cursors ボタンをクリックします。



Do It をクリックします。
これにより、カーソルで指定したテンプレートウェーブのセクショ
ンが複製されます。

Duplicate を使うべきところを誤って Make コマンドを使ってしまうことがあります。
例えば、カーブフィッティングの宛先ウェーブは、ソースウェーブと同じポイント数、数値タイプ、および数値精度を持つ必要があります。
ソースウェーブを複製することで、これが確実に実現します。

Duplicate コマンドの例

ウェーブを複製し、その複製を変形する：

```
Duplicate/O wave0, wave0_d1; Differentiate wave0_d1
```

Duplicate を使って、テンプレートウェーブのプロパティを継承する：

```
Make/N=200 wave0; SetScale x 0, 2*PI, wave0; wave0 = sin(x)
Duplicate wave0, wave1; wave1 = cos(x)
```

カーブフィッティング用の宛先ウェーブを作成する：

```
Duplicate/O data1, data1_fit
CurveFit gauss data1 /D=data1_fit
```

ウェーブの前半と後半を比較する：

```
Duplicate/O/R=[0,99] data1, data1_1
Duplicate/O/R=[100,199] data1, data1_2
Display data1_1, data1_2
```

新しいウェーブ名がすでに存在するかどうかはわからない、あるいはそれを気にしない場合、Duplicate には /O フラグ （上書き） をよく使います。

ウェーブの Kill（削除）

KillWaves コマンドは、現在の Experiment からウェーブを削除します。
これにより、ウェーブが使用していたメモリが解放されます。
不要になったウェーブは、リストやダイアログのポップアップメニューを乱雑にします。
それらを削除することで、この乱雑さを軽減することができます。

以下は、KillWaves を使用する状況の例です。

- ファイルから読み込んだデータの確認が完了した。
- 実験用に作成したウェーブの使用が終了した。
- プロシージャで一時的に使うために作成したウェーブが、必要なくなった。

Kill Waves ダイアログは、KillWaves コマンドを行うためのインターフェイスです。
これを使うには、Data メニューから Kill Waves を選択してください。

Igor は、グラフ、テーブル、またはユーザー定義関数で使用されているウェーブを削除することはできません。
そのため、これらのウェーブはリストに表示されません。

注記： Igor は、ウェーブがマクロから参照されているかどうかを判別できません。
そのため、マクロから参照されているが、それ以外では使われていないウェーブは削除することができます。

この最も一般的なケースは、グラフを閉じて再作成マクロとして保存する場合です。
グラフで使われていたウェーブは、マクロでのみ使われるようになり、それらを削除することができます。
すると、グラフの再作成マクロを実行しても、グラフを再作成できなくなります。

KillWaves は、ウェーブがロードされた「ソースファイル」と呼ばれる Igor バイナリウェーブファイルを削除することができます。
通常、これは必要ありません。
なぜなら、削除するウェーブはディスクに保存されたことがないか、パックされた Experiment ファイルの一部として保存されているため、スタンドアロンファイルから読み込まれていないからです。

「Kill all waves not in use」オプションは、一連のウェーブをロード、グラフ化、処理するプロシージャを含む Experiment を作成した場合に使います。
１つのウェーブのバッチを処理したら、すべてのグラフとテーブルを削除し、Experiment 内のすべてのウェーブを削除して、次のバッチのロードに備えます。
これは、現在のデータフォルダー内のウェーブにのみ影響します。
他のデータフォルダー内のウェーブは削除されません。

KillWave コマンドの例

以下は KillWaves を使った簡単な例です。

```
// すべてのターゲットウィンドウとすべてのウェーブを Kill します。
// ターゲット以外のウィンドウ （プロシージャおよびヘルプウィンドウ） は Kill しません。
Function KillEverything()
	String windowName

	do
		windowName = WinName(0, 1+2+4+16+64)	// 次のターゲットウィンドウを取得
		if (CmpStr(windowName, "") == 0)		// 名前が "" の場合、
			break							// 完了してループを抜ける
		endif
		KillWindow $windowName				// このターゲットウィンドウを Kill
	while (1)

	KillWaves/A								// すべてのウェーブを Kill
End

// プロシージャで一時的に使ったウェーブを削除する例です。
Function Median(w)							// wave w の平均値を返す
	Wave w

	Variable result

	Duplicate/O w, temp						// ウェーブのクローンを作成
	Sort temp, temp							// クローンをソート
	result = temp[numpnts(temp)/2]

	KillWaves temp							// クローンを Kill

	return result
End
```

その他の例については、「WaveMetrics Procedures」フォルダー内の「Kill Waves」プロシージャファイルを参照してください。

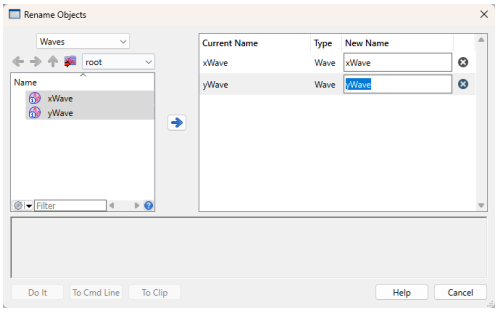
ウェーブのブラウズ

Data Browser（Data メニュー）では、任意の時点で存在するウェーブ（および文字列や変数）を確認できます。
また、存在するデータフォルダーを確認したり、現在のデータフォルダーを設定したりすることもできます。
Data Browser の詳細については、ヘルプ Data Folders（Data Folders.ihf）を参照してください。

Igor Pro 6 には、メニュー Data → Browse Waves からアクセスできる Browse Waves ダイアログがありました。
このダイアログは Data Browser と同じ機能を提供していたため、Igor Pro 7.0 ではダイアログとメニュー項目が削除されました。

ウェーブ名の変更

ウェーブは、次のようにして名前を変更できます。



- Data Browser
- Rename ダイアログ（Data メニュー）
- コマンドラインでの Rename コマンド

Rename コマンドは、ウェーブだけでなく他のオブジェクトの名前も変更できます。

ウェーブの名前の変更理由としては、次のようなものがあります。

- ファイルから複数のウェーブをロードすると、Igor は自動的にウェーブに名前を付けてしまう。
- ウェーブの命名規則を決定し、既存のウェーブをその規則に従わせたい。
- 既存のウェーブと同じ名前のウェーブセットをロードしようとしていて、既存のウェーブを削除したいが、メモリには残しておきたい（新しいデータフォルダーに移動しても同じことができます）。

Rename コマンドを使うには、Data メニューから Rename を選択します。
これにより、Rename Objects ダイアログが表示されます。

ウェーブの次元の変更

Redimension コマンドでは、次のウェーブのプロパティを変更できます。

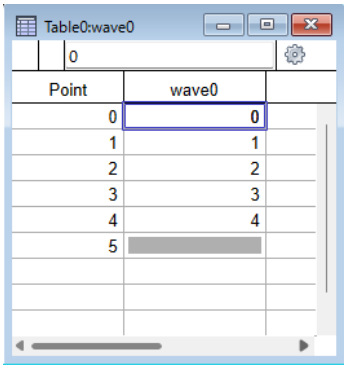
- ウェーブ内の次元数
- 各次元の要素の数
- 数値の精度（例：単精度、倍精度）
- 数値の形式（例：実数、複素数）

Redimension Waves ダイアログは、Redimension コマンドを行うためのインターフェイスです。
これを使うには、Data メニューから Redimension Waves を選択します。

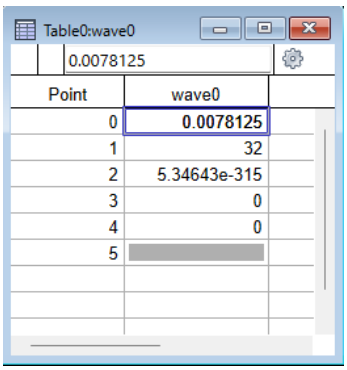
Redimension がウェーブに新しい要素を追加すると、数値ウェーブの場合は 0 に、テキストウェーブの場合は空白に設定されます。

次のコマンドは、ウェーブの数値の精度を変更する 2 つの方法を示しています。
Redimension はウェーブの内容を維持しますが、Make は内容を維持しません。

```
Make/N=5 wave0=x
Edit wave0
Redimension/D wave0      // これは wave0 の内容を保持する
```




```
Make/N=5 wave0=x
Edit wave0
Make/O/D/N=5 wave0       // これは wave0 の内容を保持しない
```



データを保持したまま 1 次元ウェーブを 2 次元ウェーブに変換する方法（つまり、形の変更）については、ヘルプ Vector (Waveform) to Matrix Conversion（Multidimensional Waves.ihf）を参照してください。

Igor では、ウェーブを数値からテキスト、またはその逆に変更することはできません。
次の例は、数値ウェーブをテキストコピー、テキストウェーブを数値コピーする方法を示しています。

```
Make/N=10 numWave = p
Make/T/N=(numpnts(numWave)) textWave = num2str(numWave)          // 数値 -> 文字列
Make/N=(numpnts(textWave)) numWave2 = str2num(textWave)          // 文字列 -> 数値
```

ただし、num2str は精度が 6 桁で出力されるため、精度が失われる可能性があります。

ポイントの挿入

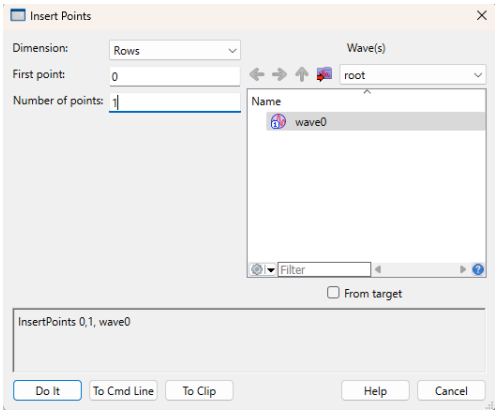
ウェーブに新しいポイントを挿入するには 2 つの方法があります。
次の方法で行います。

- InsertPoints コマンドを使う
- テーブルで入力またはペーストする

このセクションでは、InsertPoints コマンドについて説明します。
テーブルへの入力やペーストに関する情報は、ヘルプ Tables（Tables.ihf）を参照してください。

InsertPoints コマンドを使うと、1D ウェーブの開始位置、中央、または終了位置に新しいデータポイントを挿入することができます。
多次元ウェーブに新しい要素を挿入することもできます。
例えば、2D 行列ウェーブに新しい列を挿入することができます。
挿入された値は、数値ウェーブの場合は 0、テキストウェーブの場合は “” になります。

Insert Points ダイアログは、InsertPoints コマンドのインターフェイスです。
これを使うには、Data メニューから Insert Points を選択します。



最初のポイントに入力した値が、選択したウェーブの選択した次元の要素数よりも大きい場合、新しいポイントは次元の最後に追加されます。
InsertPoints は、ウェーブの次元を変更することができます。たとえば、1D ウェーブに列を挿入すると、2D ウェーブになります。

Insert Points を選択した時点で最前面のウィンドウがテーブルの場合、テーブルの選択内容に基づいてダイアログの項目を事前設定します。

ポイントの削除

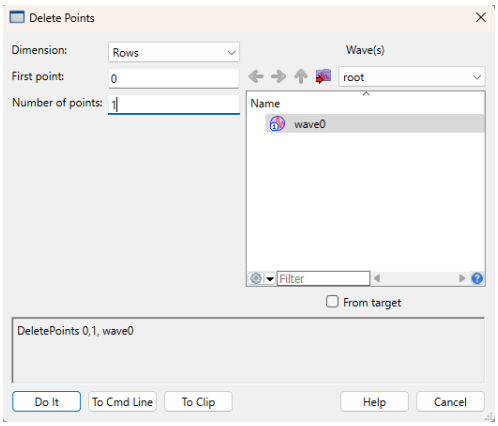
ウェーブからポイントを削除するには 2 つの方法があります。
次の方法で行います。

- DeletePoints コマンドを使う
- テーブルでカットする

このセクションでは、DeletePoints コマンドについて説明します。
テーブルでの切り取りに関する情報は、ヘルプ Tables（Tables.ihf）を参照してください。

DeletePoints コマンドを使うと、1D ウェーブの先頭、中央、または末尾からデータポイントを削除できます。
多次元ウェーブから要素を削除することもできます。
例えば、2D 行列ウェーブから列を削除できます。

Delete Points ダイアログは、DeletePoints コマンドのインターフェイスです。
これを使うには、Data メニューから Delete Points を選択します。



最初のポイントに入力した値が、選択したウェーブの選択した次元の要素数よりも大きい場合、DeletePoints はそのウェーブに対して何もしません。
要素の数が多い場合、DeletePoints は指定された最初の要素から次元末尾までを削除します。

すべての要素を削除してウェーブを 1D にする場合を除き、DeletePoints はウェーブの次元を変更しません。
変更する場合は、Redimension を使ってください。

Delete Points を選択した時点で最前面のウィンドウがテーブルの場合、テーブルの選択内容に基づいてダイアログの項目を事前設定します。

ウェーブフォームの演算と代入

ウェーブフォーム演算は、Igor の分析機能の中で非常に柔軟で強力な部分です。
標準的なプログラミング言語で単一の変数に代入を行うのと同じように、ウェーブ全体またはウェーブの一部に対して代入文を記述することができます。

ウェーブ代入文では、左側にウェーブ、右側に数式が現れます。
以下に例を挙げます。

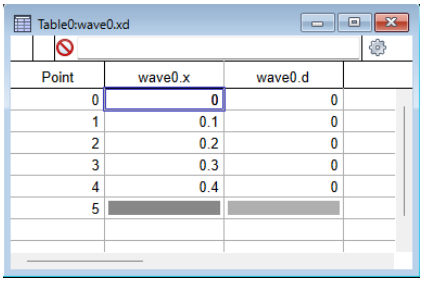
```
wave0 = sin(x)
wave0 = log(wave1/wave2)
wave0[0,99] = wave1[100 + p]
```

左側のウェーブは「宛先ウェーブ」と呼ばれます。
右側のウェーブは「ソースウェーブ」と呼ばれます。

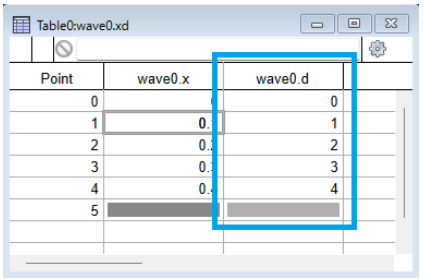
ウェーブ代入文を実行すると、右辺の式が、宛先ウェーブの各ポイントについて 1 回ずつ評価されます。
各評価の結果は、宛先ウェーブの対応するポイントに格納されます。

実行中、シンボル p は、設定中の宛先ウェーブのポイントの番号と同じ値になり、シンボル x はそのポイントの X 値と同じ値になります。
特定のポイントの X 値は、そのポイントの番号とウェーブの X スケールによって決まります。
これを確認するには、以下のコマンドを 1 つずつ実行してください。

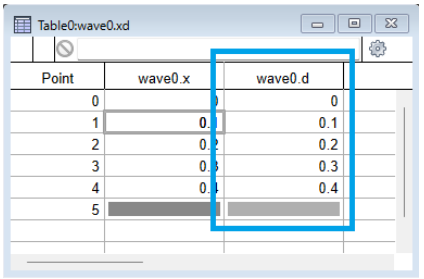
```
Make/N=5 wave0; SetScale/P x 0, .1, wave0; Edit wave0.xy
```




```
wave0 = p
```




```
wave0 = x
```



最初の代入文は、wave0 の各ポイントの値をそのポイント番号に設定します。
2 番目の代入文は、wave0 の各ポイントの値をそのポイントの X 座標に設定します。

ソースウェーブは、評価されるポイントでデータ値を返します。
例えば、

```
wave0 = log(wave1/wave2)
```

Igor は、wave0 の各ポイントについて、右側の式を 1 回ずつ評価します。
式の評価ごとに、wave1 および wave2 は、評価対象のポイントのデータ値を返します。
したがって、次の 2 つのコマンドは同等です。

```
wave0 = log(wave1/wave2)
wave0 = log(wave1[p]/wave2[p])
```

ウェーブの代入の理解

ウェーブの代入を理解するには、p と x の意味を理解することが重要です。
このために、Igor の内部で何が起こっているかを考えてみるとよいでしょう。
ウェーブの代入文は、Igor の内部でループを実行します。

p は内部ループのループインデックスで、設定先のウェーブで設定されるポイントの範囲を動きます。
前の例では、p は 0 から始まり、内部ループを回すたびにインクリメントされ、wave0 のポイント数 N-1 まで増加します。
ここで、N は wave0 のポイント数です。
p の各値に対して、右側の式が評価され、その結果は wave0 の対応するポイントに格納されます。

ウェーブフォームの代入文の右側に wave1 または wave1 [p] がある場合、これは、現在の内部ループの反復中に、宛先ウェーブ (wave0) 内のポイント p における wave1 の値を返します。

x は p と似ていますが、Igor 内部ループのループインデックスではなく、宛先ウェーブの X スケーリングを使ってループインデックスから計算される点が異なります。

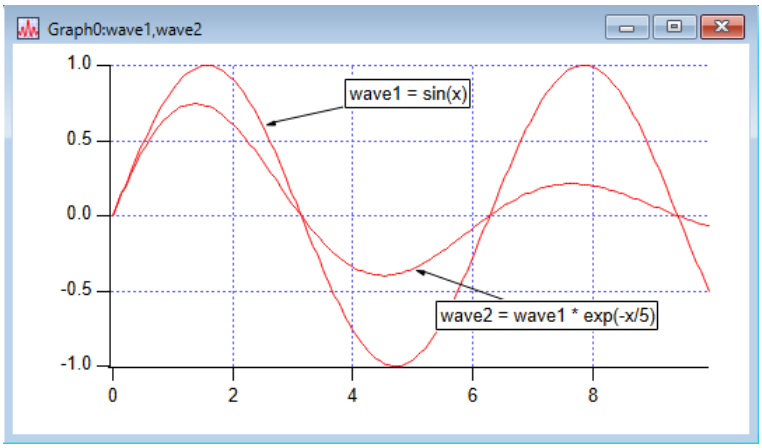
x = x0 + p*dx

ウェーブの代入は Igor の重要な機能であり、理解しておく価値があります。

ウェーブ代入の例

次のコマンドシーケンスは、上記で説明したいくつかの概念を具体的に示しています。

```
Make/N=200 wave1, wave2                 // 2 つのウェーブ、各 200 ポイント
SetScale/P x, 0, .05, wave1, wave2      // X 値を 0 から 10 に設定
Display wave1, wave2                    // ウェーブのグラフを作成
wave1 = sin(x)                          // wave1 の値を代入
wave2 = wave1 * exp(-x/5)               // wave2 の値を代入
```



wave1 は 200 ポイントがあるため、ウェーブの代入文 wave1=sin(x) は、wave1 の各ポイントについて 1 回ずつ、sin(x) を 200 回評価します。
wave1 の最初のポイントはポイント番号 0 であり、wave1 の最後のポイントはポイント番号 199 です。
この例では使われていないシンボル p は、0 から 199 までを意味します。
シンボル x は、SetScale コマンドで指定されたとおり、0 から始まり 0.05 ずつ増加する 200 個の X 値を順に参照します。
各評価の結果は、wave1 の対応するポイントに格納され、wave1 は約 1.5 周期の正弦波になります。

wave2 も 200 ポイントがあるため、ウェーブの代入文 wave2=wave1*exp (-x/5) は、wave2 の各ポイントについて 1 回ずつ、wave1*exp (-x/5) を 200 回評価します。
この代入では、右側の式にはウェーブ wave1 が含まれています。

代入を実行すると、p は 0 から 199 まで変化します。
右側が 200 回評価されるたびに、wave1 は対応するポイントのデータ値を返します。
各評価の結果は、ウェーブ 2 の対応するポイントに保存され、ウェーブ 2 は減衰した正弦波の約 1.5 周期になります。

ウェーブ代入文の効果は、宛先のウェーブのデータ値を設定することです。
代入によって暗示される機能的な関係については記憶しません。
上記の例では、wave1 を変更しても、wave2 は自動的に変更されません。
wave2 が wave1 に対して変更する前の同じ機能的な関係を持つようにするには、wave2=wave1*exp（-x/5）の代入を再実行する必要があります。

機能的な関係を確立する特別な種類のウェーブ代入文があります。
ただ、これは控えめに使ってください。
詳細については、「ウェーブ依存式」セクションを参照してください。

その他のウェーブ代入機能

シンボル p が行次元における現在の要素番号を返すのと同様に、シンボル q、r、s は、多次元ウェーブの列、レイヤー、およびチャンク次元における現在の要素番号を返します。
行の次元にあるシンボル x には、列、レイヤー、およびチャンクの次元にある y、z、t という類似のシンボルがあります。
詳細はヘルプ Multidimensional Waves（Multidimensional Waves.ihf）を参照してください。

複数のプロセッサを使って、実行に時間がかかるウェーブフォームの代入文を実行することができます。
詳細については、ヘルプ Automatic Parallel Processing with Multithread（Advanced Topics.ihf）を参照してください。

右側の式は、宛先ウェーブを含むデータフォルダーのコンテキストで評価されます。
詳細については、ヘルプ Data Folders and Assignment Statements（Data Folders.ihf）を参照してください。

通常、ソースウェーブは、宛先ウェーブと同じポイント数と X スケーリングを持っています。
場合によっては、これが当てはまらないウェーブ代入文を記述すると便利なことがあります。
これについては、「ウェーブのミスマッチ」セクションで説明しています。

インデックスとサブレンジ

Igor では、1 次元ウェーブ内の特定のポイントまたはポイントの範囲を参照するために、X 値インデックスとポイント番号インデックスの 2 つの方法があります。
次の例を見てみます。

```
wave0[54] = 92                  // wave0 のポイント 54 を 92 に設定
wave0(54) = 92                  // wave0 の X=54 を 92 に設定
wave0[1,10] = 92                // wave0 のポイント 1 からポイント 10 を 92 に設定
wave0(1,10) = 92                // wave0 の X=1 から X=10 を 92 に設定
```

角括弧は、ポイント番号を使ってウェーブをインデックス化していることを意味します。
括弧内の数字は、インデックス化されたウェーブのポイント番号として解釈されます。

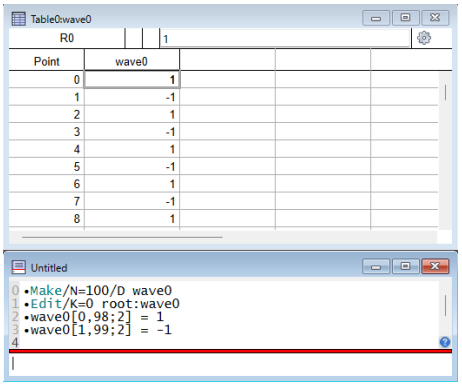
丸括弧は、X 値を使ってウェーブをインデックス化していることを示します。
括弧内の数値は、インデックス化されたウェーブの X 値として解釈されます。
X 値をインデックスとして使う場合、Igor はまず、インデックス付きウェーブの X スケーリングに基づいてその X 値に対応するポイント番号を見つけ、そのポイント番号をポイントインデックスとして使います。

ウェーブにポイントスケーリングが設定されている場合、この 2 つの方法はまったく同じ効果になります。
ただし、ウェーブの X スケーリングをポイントスケーリング以外に設定している場合、これらのコマンドは異なる

動作をします。
どちらの場合も、範囲は両端を含みます。

範囲だけでなく、ポイント番号の増分も指定できます。
例えば、

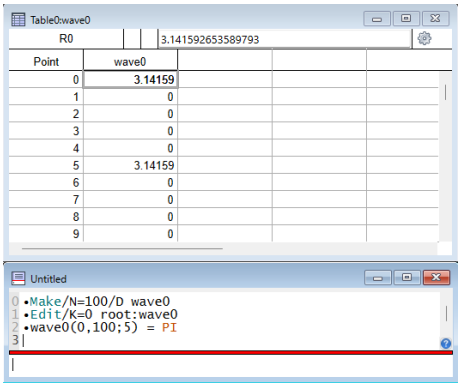
```
wave0[0,98;2] = 1      // wave0 の偶数ポイントを 1 に設定
wave0[1,99;2] = -1     // wave0 の奇数ポイントを-1 に設定
```



セミコロン後の数字は増分です。
開始ポイント番号から始まり、終了ポイント番号まで、増分分ずつスキップしながら進みます。
結果のポイント番号ごとに、ウェーブ代入文の右辺を評価し、それに応じて宛先ポイントを設定します。

増分は、X 値で範囲を指定する場合でも、増分が常にポイント数で指定される場合に使用できます。
例えば、

```
wave0(0,100;5) = PI   // wave0 の指定された X 値を PI に設定
```



ここでは、x = 0 に対応するポイント番号から始まり、x = 100 に対応するポイント番号まで進みます。
ポイント番号は、反復ごとに 5 ずつ増加します。
開始が省略された場合、ポイント番号 0 が使われます。
終了が省略された場合、ウェーブの最後のポイントが使われます。
* 文字または INF を使って、最後の点を指定することもできます。
増分値を省略した場合、デフォルトで 1 ポイントになります。

以下に、これらのショートカットを説明する例をいくつか示します。

```
wave0[ ,50] = 13                    // wave0 のポイント 0 から 50
wave0[51,] = 27                     // wave0 のポイント 51 から最後のポイント
wave0[51,*] = 27                    // wave0 のポイント 51 から最後のポイント
wave0[51,INF] = 27                  // wave0 のポイント 51 から最後のポイント
wave0[ , ;2] = 18.7                 // wave0 のすべての偶数ポイント
wave0[1,*;2] = 100                  // wave0 のすべての奇数ポイント
```

宛先ウェーブのサブレンジ（一部）は、1 つのポイントまたは複数のポイントで構成されますが、ソースウェーブのサブレンジは 1 つのポイントで構成される必要があります。

つまり、次のウェーブ代入文

```
wave1(4,5) = wave2(5,6)             // 正しくない！
```

は正しくありません。
この代入では、x は 4 から 5 の範囲です。
次の方法を使うことで、希望の効果を得ることができます：

```
wave1(4,5) = wave2(x+1)             // OK!
```

左側に指定された範囲により、x は 4 から 5 まで変化します。
したがって、x+1 は 5 から 6 まで変化し、右側の式は wave2 の値を 5 から 6 まで返します。

インデックスウェーブを使ったインデクシング

次の構文を使って、別のウェーブを使ってインデックス値を提供し、宛先ウェーブの特定の要素を設定することができます。

```
destWave[indexWave] = <expression>
```

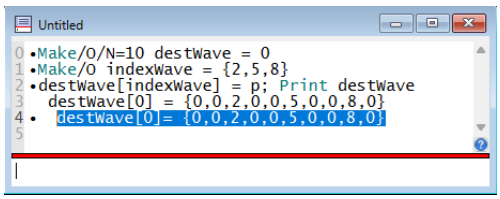
この機能は Igor Pro 8.0 で追加されました。

宛先ウェーブが 1 次元の場合、インデックスウェーブには有効なポイント番号のリストが含まれている必要があります。
例えば：

```
Make/O/N=10 destWave = 0
Make/O indexWave = {2,5,8}
destWave[indexWave] = p; Print destWave
```

出力：

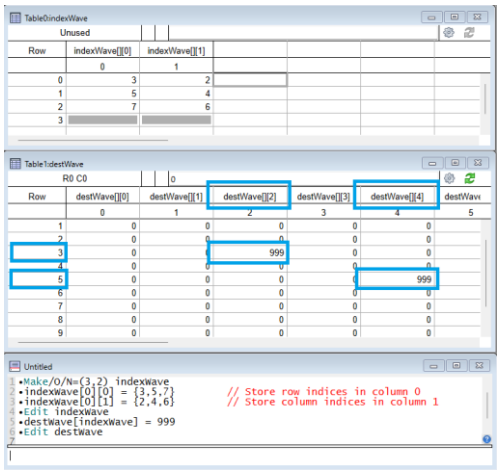
```
destWave[0]= {0,0,2,0,0,5,0,0,8,0}
```



宛先ウェーブが多次元の場合、インデックスウェーブは２次元で、１列目に有効な行インデックス、２列目に有効な列インデックスを指定することができます。

次の例では、宛先ウェーブのすべての要素を 0 に設定し、要素（3,2）、（5,4）、および （7,6） を 999 に設定します。

```
Make/O/N=(10,10) destWave = 0
Make/O/N=(3,2) indexWave
indexWave[0][0] = {3,5,7}       // 列 0 に行インデックスを保存
indexWave[0][1] = {2,4,6}       // 列 1 に列インデックスを保存
Edit indexWave
destWave[indexWave] = 999
Edit destWave
```

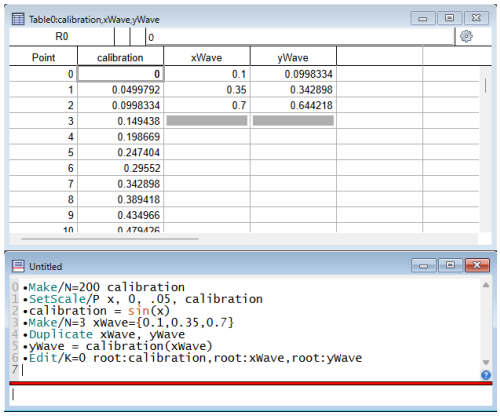


ウェーブ代入における補間

小数点以下の数値または２つのデータポイントの間に位置する X 値を指定すると、直線的に補間したデータ値を返します。
例えば、wave1[1.75] は、ポイント 1 のデータ値からポイント 2 のデータ値までの４分の３の位置にある wave1 の値を返します。
この補間は、１次元ウェーブに対してのみ行われます。
多次元データへの代入については、ヘルプ Multidimensional Wave Assignment（Multidimensional Waves.ihf）を参照してください。

これは強力な機能です。
calibration と呼ばれる等間隔のキャリブレーション曲線があり、xData というウェーブに保存されている特定の X 座標のセットのキャリブレーション値を見つけたい場合を想像してください。
calibration ウェーブの X スケーリングを設定している場合は、次のようなことを実行することができます。

```
Make/N=200 calibration
SetScale/P x, 0, .05, calibration
calibration = sin(x)
Make/N=3 xWave={0.1,0.35,0.7}
Duplicate xWave, yWave
yWave = calibration(xWave)
```



これは、ウェーブ代入文の補間機能を使って、xData ウェーブの各 X 座標について、キャリブレーションウェーブから直線補間された値を見つけます。

値のリスト

ウェーブまたはウェーブの一部（サブレンジ）に、中括弧で囲んだ値のリストを使って値を割り当てることができます。
行や列を追加することもできます。

値の 1D リスト

このセクションでは、1D ウェーブの要素を設定し、値のリストを使って行を追加する方法を説明します。
以下に示すように、1D 値のリストは、中括弧で囲まれた行の値のリストで構成されます。

```
Make/O/N=5 wave0 = NaN                  // 5 ポイントのウェーブを作り、テーブルで表示
Edit/W=(5,45,450,350) wave0

wave0 = {0, 1, 2}                       // wave0 を 3 行に次元を変え、Y 値を 0,1,2 に設定

Make/O/N=5 wave0 = NaN                  // 5 ポイントに戻す

wave0[1,3]= {1, 2, 3}                   // ポイント 1 から 3 を 1,2,3 に設定

wave0 = NaN
wave0[1]= {1, 2, 3}                     // ポイント 1 から 3 を 1,2,3 に設定（前の例と同じ）

wave0 = NaN
wave0[0,4;2]= {0, 2, 4}                 // ステップサイズ 2 を使ってポイント 0,2,4 を 0,2,4 に設定

// wave0 の行数を 5 行から 8 行に拡張し、新しい Y 値を設定
// インデックス 5 がウェーブ行の数と同じであるため、行が追加される
wave0 = NaN
wave0[5]= {5, 6, 7}

Make/O/N=5 wave0 = NaN                  // 5 ポイントに戻す

// wave0 を 1D から 2D にするために列を追加
// インデックス 1 がウェーブ列の数と同じであるため、列が追加される
wave0[][1] = {10,11,12,13,14}
```

値の 2D リスト

このセクションでは、2D ウェーブの要素を設定し、値のリストを使って行と列を追加する方法を説明します。
以下の通り、2 次元リストは中括弧で囲まれたリストのネストされたリストから構成されています。
各内側のリストは、1 つの列の行の値を定義します。

```
Make/O/N=(4,3) w2D = NaN                // 4 行 3 列のウェーブを作成
Edit/W=(225,45,850,350) w2D

// w2D の次元を変更し、要素を設定
// 列 0 を {1,2,3} に、列 1 を {10,11,12} に設定
w2D = {{0,1,2}, {10,11,12}}

Make/O/N=(4,3) w2D = NaN                // 4 行 3 列に戻す

// 列 0 を {0,1,2,3} に、列 1 を {10,11,12,13} に設定
// [] は「すべての行」と読み替えることが可能
// つまり、次は列 0 から 1 までのすべての行を設定することを意味する
w2D[][0,1] = {{0,1,2,3}, {10,11,12,13}}
```

```
// w2D の行数を４行から５行に拡張し、新しい Y 値を設定
// 行インデックス 4 がウェーブ行の数と等しいため、行を追加
// [] は「すべての列」と読み替えることが可能
// つまり、次は行 4、すべての列をを意味する
w2D[4][] = {{4},{14},{24}}

// w2D の行数を 5 行から 7 行に拡張し、新しい Y 値を設定
w2D[5][] = {{5,6},{15,16},{25,26}}

Make/O/N=(4,3) w2D = NaN              // ４行３列に戻す

// w2D の列数を３列から４列に拡張し、新しい Y 値を設定
// 列インデックス 3 がウェーブ列の数と等しいため、列を追加
w2D[][3] = {{30,31,32,33}}

// w2D の列数を４列から６列に拡張し、新しい Y 値を設定
w2D[][4] = {{40,41,42,43},{50,51,52,53}}
```

ウェーブの初期化

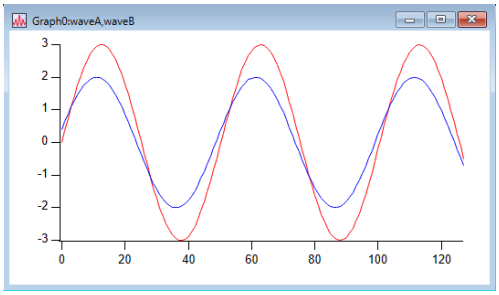
コマンドラインまたはプロシージャから、次の例に示すように、１つのコマンドでウェーブを作成し、それを初期化することができます。

```
Make wave0=sin(p/8)          // wave0 はデフォルトのポイント数を持つ
Make coeffs={1,2,3}          // coeffs は３ポイントだけを持つ
```

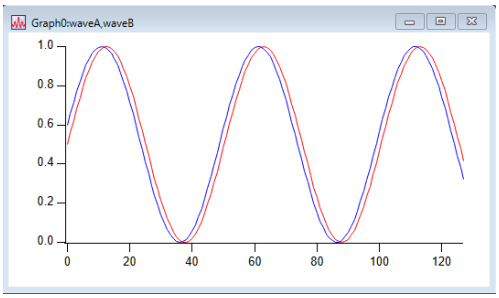
例：ウェーブの正規化

複数のウェーブの形を比較する場合、それらを共通の範囲に正規化することをお勧めします。
例えば、

```
// サンプルデータの作成
Make waveA = 3*sin(x/8)
Make waveB = 2*sin(pi/16 + x/8)

// ウェーブの表示
Display waveA, waveB
ModifyGraph rgb(waveB)=(0,0,65535)
```




```
// ウェーブの正規化
Variable aMin = WaveMin(waveA)
Variable bMin = WaveMin(waveB)
waveA -= aMin
waveB -= bMin
Variable aMax = WaveMax(waveA)
Variable bMax = WaveMax(waveB)
waveA /= aMax
waveB /= bMax
```



一時変数 aMin と bMin の使用に注目してください。
これらは 2 つの理由から必要です。
まず、`waveA -= WaveMin(waveA)` と書くと、waveA のポイントごとに WaveMin が呼び出され、これは時間の無駄になります。
さらに悪いことに、ウェーブフォームの代入文の実行中に waveA の最小値が変化し、誤った結果になります。

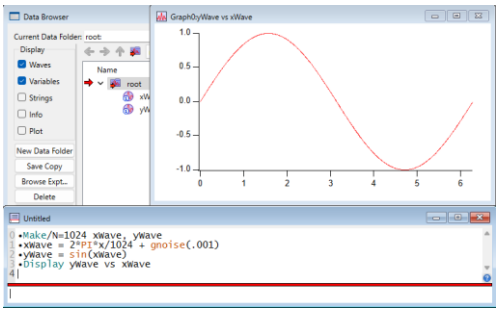
ウェーブフォームの演算を行うより高速な方法があります。
大きなウェーブの場合、FastOp および MatrixOp 演算を使うと速度が向上します。

```
waveA -= aMin                          // FastOp はウェーブ変数をサポートしていない
FastOp waveA = (1/aMax) * waveA

MatrixOp/O waveA = waveA - aMin
MatrixOp/O waveA = waveA / aMax
```

例：XY データをウェーブフォームデータに変換

XY データを等間隔のウェーブフォームデータに変換したい場合があります。
例えば、高速フーリエ変換（FFT）は、均一に間隔が空いたデータが必要です。
時間ドメインで XY データを測定した場合、FFT を行う前にこの変換を行う必要があります。

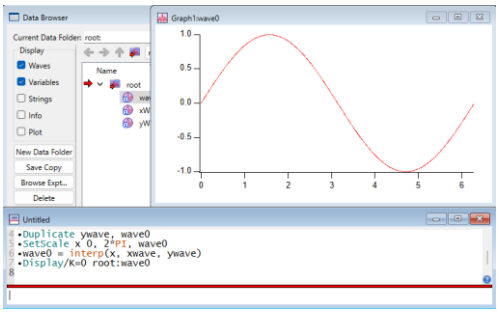
次のように XY データのサンプルを作成します。



```
Make/N=1024 xWave, yWave
xWave = 2*PI*x/1024 + gnoise(.001)
yWave = sin(xwave)
```

xWave の値は 0 から 2π の範囲にあり、わずかなノイズが含まれています。
このデータは x 軸方向で等間隔に配置されていませんが、単調増加しています（この場合、常に増加しています）。
もし単調でなければ、XY ペアを並べ替えることができます。

XY データに相当するウェーブフォームは、次のように作成できます。



```
Duplicate ywave, wave0
SetScale x 0, 2*PI, wave0
wave0 = interp(x, xwave, ywave)
```

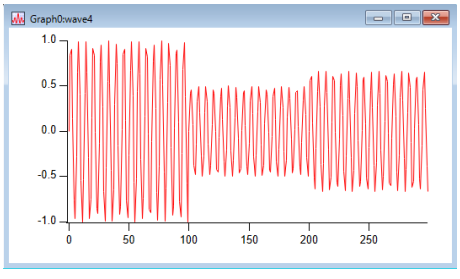
SetScale コマンドは、wave0 のスケーリングを設定し、その X 値が 0 から 2π までになるようにします。
そのデータ値は、各 X 値において ywave vs xwave の曲線から線形補間を使って値を選択することで生成されます。

例：ウェーブの連結

ウェーブの連結は、Concatenate コマンドを使うとより簡単に行うことができます。
次の簡単な例は、主にウェーブ代入文の使い方を説明するためのものです。

100 ポイントの 3 つのウェーブ、wave1、wave2、wave3 があるとします。
3 つの元のウェーブをつなぎ合わせた 4 番目のウェーブ、wave4 を作成したいとします。
以下のコマンドの順序で実行してください。

```
Make/N=100 wave1, wave2, wave3
wave1=sin(x)
wave2=1/2*sin(x)
wave3=2/3*sin(x)

Make/N=300 wave4
wave4[0,99] = wave1[p]              // wave4 の最初の 1/3
wave4[100,199] = wave2[p-100]       // wave4 の中央の 1/3
wave4[200,299] = wave3[p-200]       // wave4 の最後の 1/3
```

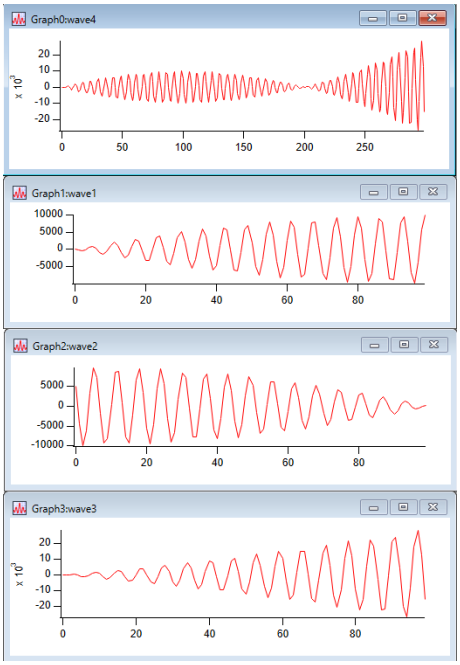


この例では、wave4 のサブレンジ（一部）をウェーブ代入文の宛先として使っています。

右側の式は、wave1、wave2、および wave3 の適切な値をインデックス付けします。
p は、宛先で評価されるポイントの範囲を指定します。
したがって、p は最初の代入では 0 から 99 まで、2 番目の代入では 100 から 199 まで、3 番目の代入では 200 から 299 までです。
各代入において、右側のウェーブはポイント 0 からポイント 99 までの 100 ポイントしかありません。
そのため、ソースウェーブの 100 個の値を選択するには、右側の p をオフセットする必要があります。

例：ウェーブの分解

300 ポイントの wave4 を、それぞれ 100 ポイントの 3 つのウェーブ（wave1、wave2、wave3）に分解したいとします。
これを行うためのコマンドの順序は次のとおりです。




```
Make/N=300 wave4
wave4 = (x*(x-200))*sin(x)
Make/N=100 wave1, wave2, wave3
wave1 = wave4[p]                    // wave4 の最初の 1/3
wave2 = wave4[p+100]                // wave4 の中央の 1/3
wave3 = wave4[p+200]                // wave4 の最後の 1/3
```

この例では、wave4 のサブレンジ（一部）をデータのソースとして使います。
ポイント番号インデックスを使って、wave4 の目的のセグメントにインデックスを付けます。
wave1、wave2、wave3 それぞれが 100 ポイントずつあるため、p は 0 から 99 の範囲です。
最初の代入では、wave4 の 0 から 99 までのポイントにアクセスします。
2 回目の代入では、wave4 の 100 から 199 までのポイントにアクセスします。
3 回目の代入では、wave4 の 200 から 299 までのポイントにアクセスします。

また、Duplicate コマンドを使って、別のウェーブのセクションからウェーブを作成することもできます。

例：複素数ウェーブの計算

複素数や複素数ウェーブを操作するための組み込み関数が多数用意されています。
これらの関数は、次の例で説明します。

時間ドメインのウェーブフォームを作成し、それに FFT を実行して複素数ウェーブを生成します。
この例での関数は、複素数ウェーブの実部と虚部を取り出し、二乗和を求め、直角座標から極座標に変換する方法を示しています。
周波数ドメインでの処理に関する詳細については、ヘルプ Fourier Transforms（Signal Processing.ihf）を参照してください。

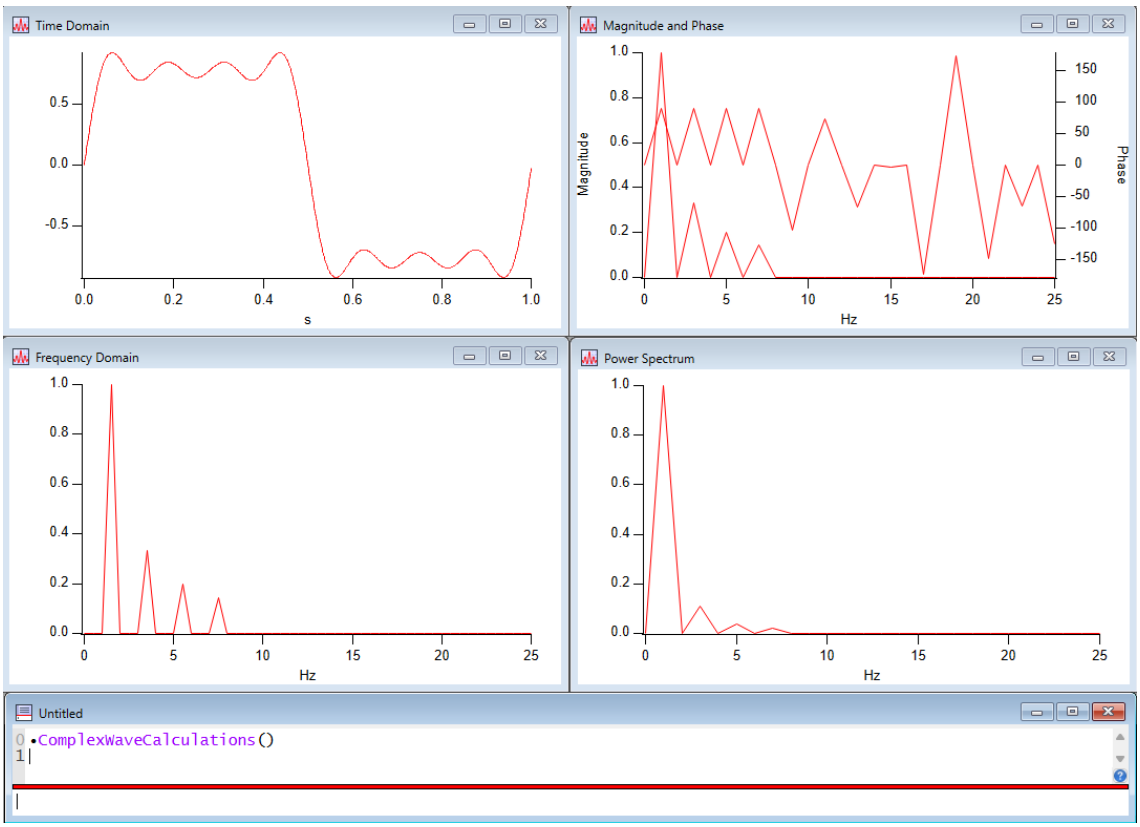
```
Function ComplexWaveCalculations()
        // 時間ドメインのウェーブフォームを作成
        Make/O/N=1024 wave0
        SetScale x 0, 1, "s", wave0                    // 0 秒から 1 秒へスケーリング
        wave0 = sin(2*PI*x) + sin(6*PI*x)/3 + sin(10*PI*x)/5 + sin(14*PI*x)/7
        Display wave0 as "Time Domain"

        // FFT を実行
        FFT/DEST=cwave0 wave0                  // 複素数ウェーブの cwave0 が生成，513 ポイント
```

```
        cwave0 /= 512;cwave0[0] /= 2                    // 振幅を正規化
        Display cwave0 as "Frequency Domain";SetAxis bottom, 0, 25

        // マグニチュードと位相を計算
        Make/O/N=513 mag0, phase0, power0               // これらは実数ウェーブ
        CopyScales cwave0, mag0, phase0, power0
        mag0 = real(r2polar(cwave0))
        phase0 = imag(r2polar(cwave0))
        phase0 *= 180/PI                                // 度に変換
        Display mag0 as "Magnitude and Phase"; AppendToGraph/R phase0
        SetAxis bottom, 0, 25
        Label left, "Magnitude";Label right, "Phase"

        // パワースペクトルを計算
        power0 = magsqr(cwave0)
        Display power0 as "Power Spectrum";SetAxis bottom, 0, 25
End
```

上記のコードを Procedure Window にコピーし、コンパイルします。
コマンドラインで

```
ComplexWaveCalculations()
```

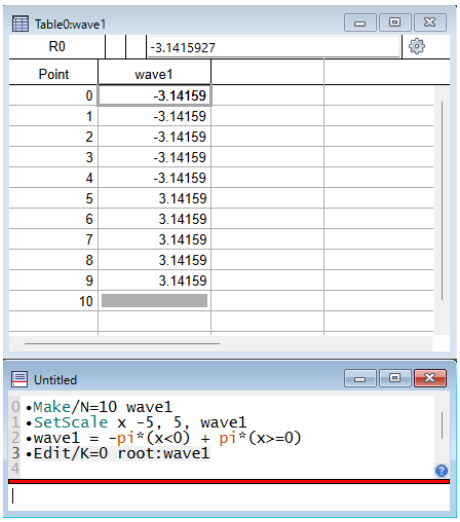
を実行すると、次の結果が得られます。



例：比較演算子とウェーブの合成

比較演算子 ==、>=、>、<=、および < は、ウェーブの合成に役立ちます。

x<0 の場合、データ値が -π となり、x>=0 の場合、データ値が +π となるようにウェーブを設定したいとします。
次のウェーブ代入文でこれを実現できます。




```
Make/N=10 wave1
SetScale x -5, 5, wave1
wave1 = -pi*(x<0) + pi*(x>=0)
```

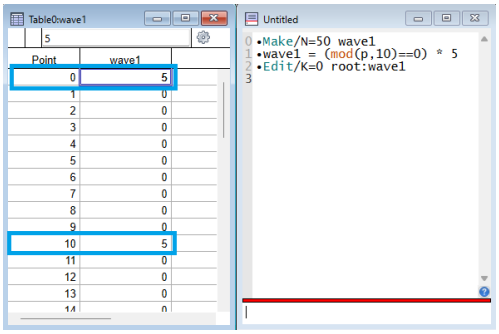
これは、条件式が真のときに 1 を返し、偽のときに 0 を返すため、乗算が実行されるためです。

条件演算子（？ ：）を使って、次のような代入を行うこともできます。

```
wave0 = (x>0) ? pi : -pi
```

mod 関数と==演算子を使って、一連のパルスを生成することができます。

次のウェーブ方程式は、ポイント 0 から 10 ポイントごとに 5 を、その他のポイントには 0 を割り当てます。




```
Make/N=50 wave1
wave1 = (mod(p,10)==0) * 5
```

例：ラベルを使ったウェーブの代入とインデックス付け

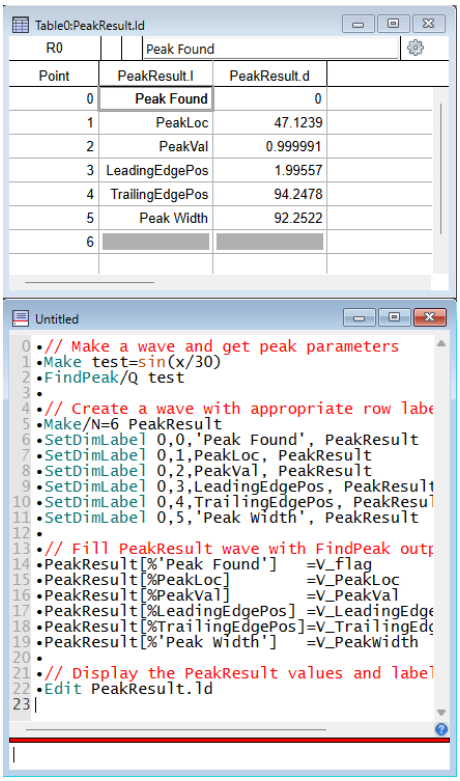
次元ラベルを使うと、ウェーブ値を意味のある名前で参照することができます。
例えば、係数の値を格納するウェーブを作成し、混乱を招き、意味もあまりない数値インデックス（coef［1］ など）の代わりに、係数の名前（coef［%Friction］ など）でこれらの値を直接参照することができます。
ウェーブの値とラベルをテーブルで表示することもできます。

ウェーブ次元ラベルは、SetDimLabel コマンドを使って作成します。
詳細については、ヘルプ Dimention Labels（Multidimentional Waves.ihf）を参照してください。
次元ラベルの長さは 255 バイトまでです。
スペースを含むような長い名前を使う場合は、その名前を単一引用符で囲んでください。

Igor Pro 8.0 以前では、次元ラベルは 31 バイトに制限されていました。
長い次元ラベルを使う場合、ウェーブファイルおよび Experiment には Igor Pro 8.0 以降が必要です。

次の例では、ウェーブを作成し、FindPeak コマンドを使ってウェーブのピークパラメーターを取得します。
次に、適切なラベルを持つ出力パラメーターウェーブを作成し、そのラベルを使って FindPeak の結果を出力ウェーブに代入します。

```
// ウェーブを作成し、ピークパラメーターを取得
Make test=sin(x/30)
FindPeak/Q test

// 適切な行ラベルを持つウェーブを作成
Make/N=6 PeakResult
SetDimLabel 0,0,'Peak Found', PeakResult
SetDimLabel 0,1,PeakLoc, PeakResult
SetDimLabel 0,2,PeakVal, PeakResult
SetDimLabel 0,3,LeadingEdgePos, PeakResult
SetDimLabel 0,4,TrailingEdgePos, PeakResult
SetDimLabel 0,5,'Peak Width', PeakResult

// FindPeak 出力変数で PeakResult を埋める
PeakResult[%'Peak Found']   =V_flag
PeakResult[%PeakLoc]        =V_PeakLoc
PeakResult[%PeakVal]        =V_PeakVal
PeakResult[%LeadingEdgePos] =V_LeadingEdgeLoc
PeakResult[%TrailingEdgePos]=V_TrailingEdgeLoc
PeakResult[%'Peak Width']   =V_PeakWidth

// テーブルに ReakResult 値とラベルを表示
Edit PeakResult.ld
```



上記の方法に加えて、ウェーブをテーブルで表示し、データと一緒に次元ラベルを表示することで、次元ラベルを作成および編集することもできます。
ラベル付きテーブルの使い方の詳細については、ヘルプ Showing Dimension Labels（Tables.ihf）を参照してください。

一致しないウェーブ

ほとんどの実際の使用においては、異なる長さのウェーブを混合する必要はありません。
実際、これを意図的に行うことはほとんどなく、誤って行われてしまう場合がほとんどです。
ただし、実際の使用で混合が必要な場合は、これをどのように処理するかを知る必要があります。

次のような、１つのウェーブの値を別のウェーブに代入する場合を考えてみましょう。

```
wave1 = wave2
```

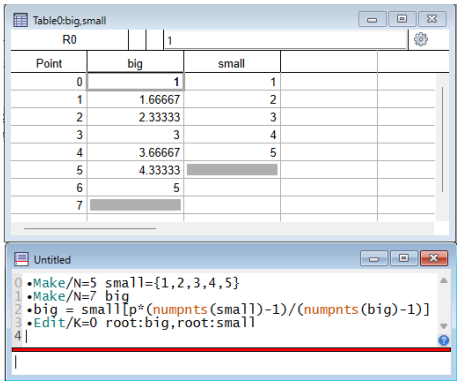
この代入では、明示的なインデックス指定がないため、次のように記述した場合と同じように式を評価します。

```
wave1 = wave2[p]
```

wave2 のポイント数が wave1 のポイント数よりも多い場合、余分なポイント数は代入に影響しません。
これは、p が 0 から n-1 の範囲を取るためです。
ここで、n は wave1 のポイント数です。

wave2 のポイント数が wave1 よりも少ない場合、wave2[p] の値が wave2 の長さよりも大きい p の値について評価を試みます。
この場合、ウェーブ代入文では wave2 の最後のポイントの値が返されますが、「Index out of range（インデックス範囲外）」エラーも発生します（古いバージョンでは発生しないことがあります）。

wave1 の値を wave2 の値の範囲に補間して、wave2 の値の範囲に wave1 の値を拡張したいと考えているかもしれません。
この操作を実行させるには、右側に適切な X 値を明示的にインデックス指定する必要があります。
たとえば、2 つの長さの異なるウェーブがある場合、次の例のように指定します。




```
Make/N=5 small={1,2,3,4,5}
Make/N=7 big
big = small[p*(numpnts(small)-1)/(numpnts(big)-1)]
```

NaN、INF、欠損値

浮動小数点数値ウェーブのデータの値は、通常は有限の数値ですが、NaN または INF である場合もあります。
NaN は「数値ではない（not a number）」という意味です。
式が数学的に意味を成さない場合、その式は NaN を返します。
例えば、log(-1) は NaN の値を返します。
また、テーブルまたはウェーブ代入文を使って、欠損値を表すために NaN をポイントに設定することもできます。
式が数学的に意味を成すが有限値を持たない場合、その式は INF を返します。
例えば、log(0) は -INF を返します。

IEEE の浮動小数点の規格は、NaN 値の表現方法と動作を定義しています。
NaN を整数ウェーブで表現する方法はありません。
NaN を整数ウェーブに格納しようとすると、ゴミ値が格納されます。

比較演算子は、NaN パラメーターでは機能しません。
これは、定義上、NaN は他の値（別の NaN であっても）と比較すると false となるためです。
値が NaN であるかどうかを調べるには、numtype を使ってください。

Igor は、カーブフィッティングおよびウェーブ統計演算をする時には NaN および INF を無視します。
NaN および INF は、グラフのスケールには影響を与えません。
プロットをする時には、NaN および INF を、それぞれ欠損値および無限大値として適切に処理します。

Igor は、他の多くの演算、特に FFT などの DSP 関連の演算では、NaN および INF を無視しません。
一般に、ウェーブのデータポイントのすべてまたは大部分を数値的に結合する操作は、1 つ以上のポイントが NaN または INF の場合、意味のない結果になります。
主な例としては、領域関数と平均関数、および積分演算と FFT 演算などがあります。
Smooth（スムージング）や Differentiate（微分）など、一部のポイントのみを結合するコマンドは、NaN または INF の近傍にあるポイントのみを「contaminate（汚す）」します。
Interpolate コマンド（Analysis メニュー）を使って、NaN を含まないウェーブを作成することができます。

area や mean などの関数、Convolve などの演算、またはウェーブ内のポイントを合計するその他の関数や演算で NaN が返された場合、ウェーブ内のポイントの一部が NaN であることを示しています。
カーブフィッティングの結果から NaN が得られた場合は、カーブフィッティングが失敗したことを示しています。

NaN の取り扱いについては、ヘルプ Dealing with Missing Values（Analysis.ihf）を参照してください。

ソースウェーブとして宛先ウェーブを使わないでください

ウェーブ代入文の宛先も右側の式に現れる場合、予期しない結果になることがあります。
次の例を考えてみましょう。

```
wave1 -= wave1(5)
wave1 -= vcsr(A)                    // カーソル A は wave1 上にある
```

これらの各例は、wave1 の特定のポイントにおける wave1 の値を、wave1 のすべてのポイントから差し引こうとするものです。
この動作は期待通りになりません。
というのも、その特定のポイントでの wave1 の値が代入中に変更されるためです。
代入のどこかの時点で、wave1(5) または vcsr(A) が 0 を返すため、そのポイントでの wave1 の値が変更されるからです。

特定のポイントで wave1 の値を格納するために変数を使うことで、希望の結果を得ることができます。

```
Variable tmp
tmp = wave1(5); wave1 -= tmp
tmp = vcsr(A); wave1 -= tmp
```

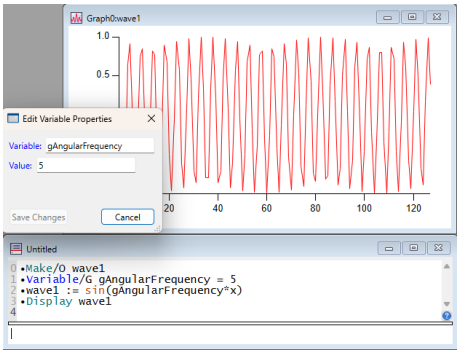
ウェーブ依存式

ウェーブの代入文を「ウェーブに固定」するには、文中の「=」を「:=」に置き換えます。
これにより、ウェーブは式で参照されるオブジェクトに依存するようになります。

例えば、

```
Make/O wave1
Variable/G gAngularFrequency = 5
wave1 := sin(gAngularFrequency*x)   // ":="に注意
Display wave1
```

ここで「gAngularFrequency = 8」を実行すると、ウェーブが自動的に更新されます。



同様に、SetScale コマンドを使ってウェーブの X スケーリングを変更すると、ウェーブは新しい X 値の範囲に合わせて自動的に再計算されます。

依存関係は、できるだけ使用を控えるべきです。
過剰な使用は、理解が困難でデバッグが難しい相互依存の関係の網の目を生み出します。
必要に応じて、明示的に対象を更新する方が望ましいです。

詳細な説明はヘルプ Dependencies（Dependencies.ihf）を参照してください。

ウェーブノートを使う

ウェーブの特性の一つに、ウェーブノートがあります。
これは、Igor が各ウェーブとともに保存するプレーンテキストです。
ウェーブを作成すると、メモは空の状態になります。
メモの長さに制限はありません。

Data Browser を使って、ウェーブノートを表示、編集することができます。
Note コマンドまたは note 関数を使って、プロシージャからウェーブノートの内容を設定または取得することができます。

当初、ウェーブノートは、ユーザーがウェーブに関する非公式なコメントを保存するための場所として考えられていましたが、その目的には問題ありません。
しかし、時間の経過とともに、ウェーブノートは、ウェーブの追加のユーザー定義プロパティを構造化された形で保存するのに便利な場所でもあることがわかりました。

これらの追加プロパティは、Data Broswer を使って編集できますが、プロシージャでも使用、操作することができます。

これを行うには、キーワードと値のペアをウェーブノートに保存します。
例えば、ノートは次のようになります。

```
CELLTYPE:rat hippocampal neuron
PATTERN:1VN21
TREATMENT:PLACEBO
```

次に、Igor の関数を使って、ウェーブの CELLTYPE、PATTERN、および TREATMENT プロパティを設定することができます。
これらのプロパティは、StringByKey 関数を使って取得することができます。

整数ウェーブ

Igor は、主にデータ取得プロジェクトを支援するために、整数ウェーブのサポートを提供しています。
これにより、ハードウェアと接続するユーザーは、整数ウェーブに直接書き込み/読み取りを行うことができます。
ライブ表示がやや高速化され、XOP プログラマが整数を浮動小数点数に変換する手間が省けます。

整数ウェーブは、画像の保存にも適しています。

メモリの考慮すること以外には、整数ウェーブを使う理由はありません。
整数ウェーブが宛先の場合、ウェーブ代入文の評価がより高速になることを期待するかもしれません。
しかし、Igor は代入には浮動小数点を使っていて、保存のために整数に変換しているだけなので、そうはなりません。

日付／時刻ウェーブ

日付は、Igor 日付形式（1904 年 1 月 1 日午前 0 時からの経過秒数）で表されます。
それ以前の日付は、負の値で表されます。
Igor は、西暦 -32768 年から 32767 年までの日付をサポートしています。

Igor 7 以降、Igor は 0 年がないグレゴリオ暦を使っています。
1 年の前の年は-1 年となります。

日付は、単精度または整数のウェーブのデータ値には正確に保存できません。
日付と時刻を保存するには、必ず倍精度を使ってください。

日付または日付/時刻データを含むウェーブのデータ単位は「dat」に設定する必要があります。
これにより、Igor はそのウェーブに日付/時刻データが含まれていることを認識し、グラフの軸や表の列のデフォルトの表示を変えます。

次の例は、日付/時刻データの使い方を示しています。
まず、日付/時刻データを作成し、テーブルで表示します：

```
Make/D/N=10/O testDate = date2secs(2011,4,p+1)
// ウェーブを日付/時刻データに設定
SetScale d 0, 0, "dat", testDate
```

SetScale d を使って、ウェーブのデータ単位を「dat」に設定しました。

次に、このウェーブをテーブルで表示します。

```
Edit testDate
ModifyTable width(testDate)=120      // 列の幅を広げる
```

データは日付として表示されていますが、実際には数値として保存されています。
具体的には、1904 年 1 月 1 日からの秒数です。

このことは、列の形式を変更することで確認できます：

```
ModifyTable format=1          // 整数で表示
```

日付形式に戻します。

```
ModifyTable format(testDate)=6      // 日付に戻す
```

次に、いくつかの時間データを作成します。
このウェーブは日付を保存しないため、倍精度である必要はありません。

```
Make/N=10/O testTime = 3600*p       // データは秒数で保存
AppendToTable testTime
```

次に、日付データに時刻データを追加して、日付/時刻のウェーブを作成します。

このウェーブは日付を格納するため、倍精度であり、データ単位は「dat」である必要があります。
Duplicate コマンドを使って元の日付ウェーブを複製することで実現します。

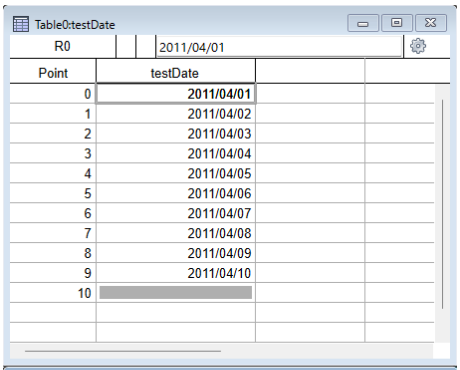
```
Duplicate/O testTime, testDateTime
AppendToTable testDateTime
ModifyTable width(testDateTime)=120          // 幅を広げる
Redimension/D testDateTime
```

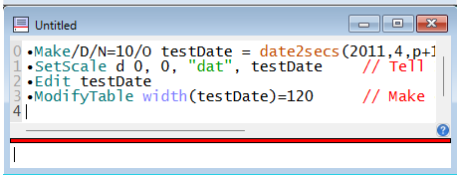
単位が「dat」であり、すべての時刻値が 00:00:00 であるため、日付/時刻のウェーブを日付形式で表示します。
ここで、列の形式を日付/時刻に変更します。

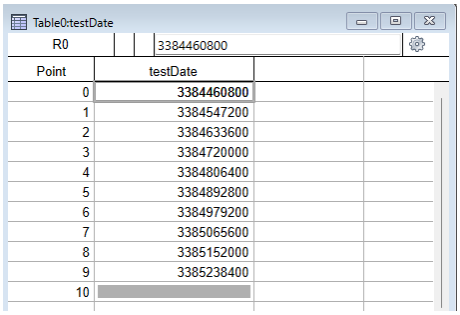
```
// 列を日付/時刻形式に設定
ModifyTable format(testDateTime)=8
```

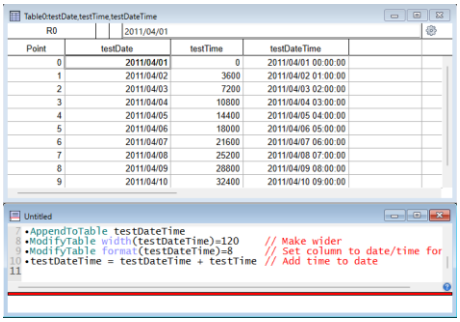
最後に、時間を追加します。

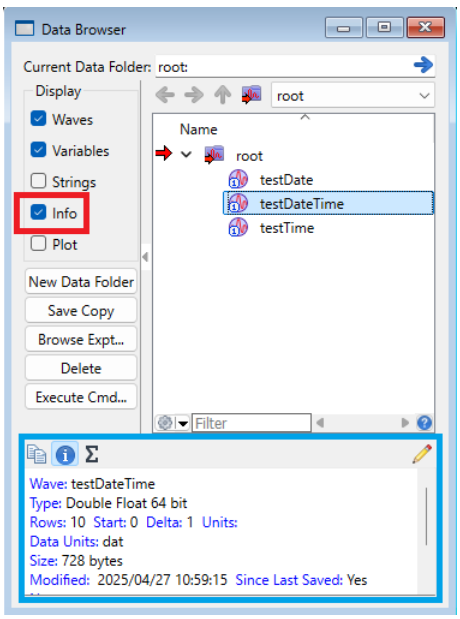
```
// 時刻を日付に追加
testDateTime = testDateTime + testTime
```









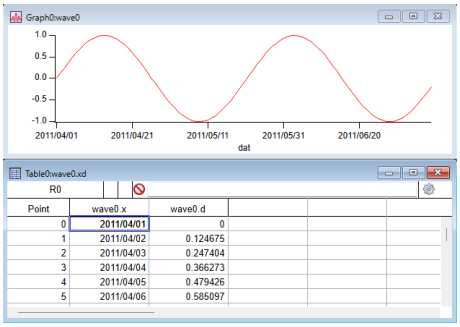


ウェーブのデータ型を確認するには、Data Browser の情報ペインを使います。
日付/時刻のウェーブに表示されるデータ型は「Double Float 64 bit」である必要があります。
そうでない場合は、メニュー Data → Redimension Waves を使って、倍精度に次元を変更してください。

これまで、ウェーブのデータに日付を保存する方法を見てきました。
通常、このような日付ウェーブは、XY ペアの X ウェーブを供給するために使われます。
データがウェーブフォームである場合は、ウェーブフォームとして扱われるウェーブの X 値に日付データを格納します。

例えば、

```
Make/N=100 wave0 = sin(p/8)
SetScale/P x date2secs(2011,4,1), 60*60*24, "dat",
wave0
Display wave0
Edit wave0.id; ModifyTable format(wave0.x)=6
```



ここでは、SetScale コマンドは、これまでのようにデータ単位ではなく、ウェーブの X スケーリングと単位を設定するために使われます。
この場合、ウェーブのデータ型に関係なく常に倍精度を使って X 値を計算するため、ウェーブを倍精度にする必要はありません。

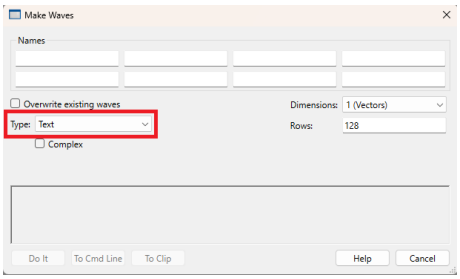
テキストウェーブ

テキストウェーブは、数値ではなくテキストを含む点を除いて、数値ウェーブとまったく同じです。
数値ウェーブと同様、テキストウェーブは 1 次元から 4 次元まで指定できます。

テキストウェーブは次の方法で作成します。

- テーブルの最初の未使用のセルに、数字以外の任意の文字を入力する。
- 区切り文字で区切られたテキストファイルから、数値ではない列を含むデータをインポートする。
- /T フラグを指定して Make コマンドを実行する。

Make Waves ダイアログでは、Type ポップアップメニューから Text を選択して、テキストウェーブを生成することができます。
ほとんどの場合、テキストウェーブは、テーブルにテキストを入力して作成します。
詳細については、ヘルプ Using a Table to Create New Waves（Tables.ihf）を参照してください。



テキストウェーブの要素には、何でも格納することができます。
長さの制限はありません。
テキストウェーブは、テーブルで編集したり、ウェーブ代入文を使ってテキストウェーブの要素に値を代入したりすることができます。

カテゴリプロットでは、テキストウェーブを使って、グラフ内の個々のデータポイントに自動的にラベルを付け（マーカーモードを使い、マーカーポップアップメニューからテキストウェーブを選択）、テーブルにメモを保存することができます。
プログラマは、複雑なプロシージャの入力や出力など、さまざまなデータを保存するのに、テキストウェーブが便利だと感じるかもしれません。

コマンドラインでテキストウェーブを作成および初期化する方法は、次のとおりです。

```
Make/T textWave= {"First element","Second and last element"}
```

テキストのウェーブを表示するには、テーブルを作成します。

```
Edit textWave
```

これで、いくつかのウェーブ代入文を試して、その結果をテーブルで確認することができます。

```
textWave[2] = {"Third element"}          // 新しい行を作成
textWave += "*"                          // 各ポイントの後にアスタリスクを追加
textWave = "*" + textWave                // 各ポイントの前に"*"を挿入
```

テキストウェーブのテキストのエンコーディング

このセクションは、テキストウェーブに ASCII 以外のデータを保存する場合にのみ関係します。
Igor は、テキストウェーブに格納されているバイトを、そのウェーブのテキストコンテンツのテキストエンコーディング設定に基づいて解釈します。
この設定は、Igor 7 以降で作成されたウェーブでは UTF-8、それ以前のバージョンで作成されたウェーブでは通常 MacRoman または Windows-1252 になります。

Igor 7 以降では、作業用テキストのエンコーディングとして UTF-8 形式の Unicode が使われます。
テキストウェーブ要素のコンテンツにアクセスしたり、テキストウェーブテキスト要素にテキストを保存したりすると、Igor は必要なテキストエンコーディングの変換を自動的に行います。
例えば、macRomanTextWave が Macintosh の Igor 6 で作成されたウェーブであると仮定します。
その場合、

```
String str = macRomanTextWave[0]    // Igor は MacRoman から UTF-8 に変換する
str = "Area = 2πr"                  // π は ASCII ではない文字
macRomanTextWave[0] = str           // Igor は UTF-8 から MacRoman に変換する
Make/O/T utf8TextWave               // 新しいウェーブは UTF-8 テキストエンコーディングを使う
utf8TextWave = macRomanTextWave     // Igor は MacRoman から UTF-8 に変換する
macRomanTextWave = utf8TextWave     // Igor は UTF-8 から MacRoman に変換する
```

詳細については、ヘルプ Text Encodings（Text Encodings.ihf）および Wave Text Encodings（Text Encodings.ihf）を参照してください。

バイナリデータを保存するためにテキストウェーブを使う

数値ウェーブは各要素に固定数のバイトを格納しますが、テキストウェーブは各ポイントに異なるバイト数を格納することができます。
これにより、テキストウェーブは、可変長の可変数の Blob（Binary Large Object）を格納するのに便利です。
これは、高度なプログラマが、洗練されたプロシージャパッケージで実行するような操作です。

この操作を行う場合は、テキストウェーブにバイナリを含むことを指定する必要があります。
そうしないと、Igor はそれをテキストとして解釈しようとし、エラーが発生します。
この問題の詳細については、ヘルプ　Text Waves Containing Binary Data（Text Encodings.ihf）を参照してください。

ホームウェーブと共有ウェーブ

パックされた Experiment ファイル、またはパックされていない Experiment のデータフォルダーに対応するディスクフォルダーに保存されているウェーブは、「ホーム」ウェーブです。
それ以外のウェーブは「共有」ウェーブです。

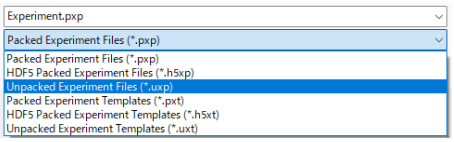
ホームウェーブは通常、その所有者の Experiment のみに使うことを目的としています。
共有ウェーブは通常、複数の Experiment で使うことを目的としています。

パックされた Experiment でウェーブを作成すると、そのウェーブはデフォルトで、パックされた Experiment ファイルに保存され、ホームウェーブになります。
スタンドアロンファイルに明示的に保存した場合のみ、共有ウェーブになります。

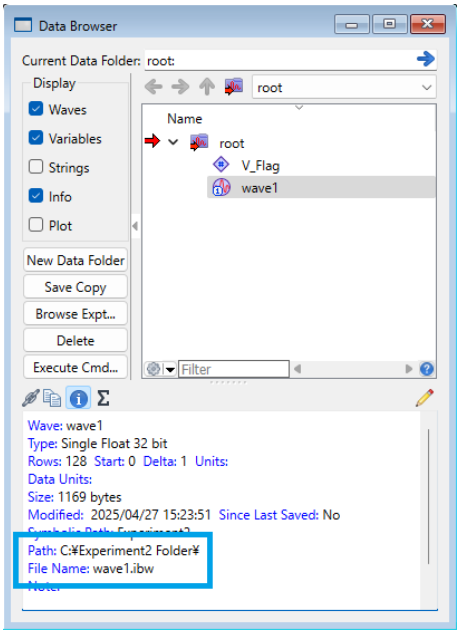
パックされていない Experiment でウェーブを作成すると、そのデータフォルダーに対応するディスクフォルダーにデフォルトで保存され、ホームウェーブになります。

ルートにあるウェーブの場合、ディスクフォルダーは Experiment フォルダーです。
他のデータフォルダーにあるウェーブの場合、ディスクフォルダーは Experiment フォルダーのサブフォルダーです。
ウェーブは、デフォルトのディスクフォルダー以外の独立したファイルに明示的に保存した場合にのみ、共有ウェーブになります。

パックされた Experiment のパックをパックしないで保存すると、ホームウェーブは Experiment フォルダー内のデフォルトのディスクフォルダーに保存されます。



パックされていない Experiment をパックして保存すると、ホームウェーブはパックされた Experiment ファイルに保存されます。



Data Browser を使って、ウェーブが共有されているかどうかを確認できます。
共有ウェーブの場合、Data Browser の情報ペインには、ディスク上のウェーブファイルのパスとファイル名が表示されます。
ホームウェーブの場合、この情報は省略されます。

共有ウェーブは、取り入れることでホームウェーブに変換することができます。
詳細については、ヘルプ Adopting Files（Experiments, Files and Folders.ihf）を参照してください。

ウェーブのプロパティ

Igor が各ウェーブについて保存するプロパティは次のとおりです。

プロパティ	説明
名前	コマンドやダイアログからウェーブを参照するために使います。 1 から 255 バイト。 標準的な名前は文字で始まります。 文字、数字、またはアンダーバーを含むことができます。 Igor Pro 8.0 以前では、ウェーブ名は 31 バイトに制限されていました。 長いウェーブ名を使う場合は、ウェーブファイルおよび Experiment に Igor Pro 8.0 以降が必要です。 リベラル名にはほぼすべての文字を使用できますが、単一引用符で囲む必要があります。 「ウェーブ名」セクションを参照してください。

	名前は、ウェーブを作成するときに指定します。 名前を変更するには、Rename コマンドを使います。 Data Browser でも変更できます。
データ形式	数値、テキスト、または参照データ型。 「ウェーブのデータ形式」セクションを参照してください。 ウェーブを作成するときに設定します。 Redimension コマンドを使って変更することができます。
長さ	ウェーブ内のデータポイントの数。 また、多次元ウェーブの場合は各次元のサイズ。 ウェーブを作成するときに設定します。 Redimension コマンドを使って変更することができます。
X スケーリング (x0 と dx)	ポイント番号から X 値を計算するために使います。 多次元ウェーブの Y、Z、T スケーリングにも使います。 ポイント p の X 値は、X = x0 + p*dx の式で計算されます。 SetScale コマンドで設定します。
X 単位	軸に自動的にラベルを付けるために使います。 多次元ウェーブの Y、Z、T 単位にも使います。 SetScale コマンドで設定します。
データ単位	軸に自動的にラベルを付けるために使います。 SetScale コマンドで設定します。
データのフルスケール	ドキュメント目的のみ。 計算などには使われません。 SetScale コマンドで設定します。
ノート	ウェーブに関連する任意のテキストを保持します。 Note コマンドまたは Data Browser 経由で設定できます。 note 関数で確認できます。
次元ラベル	各次元インデックスおよび各次元に対して、最大 255 バイトの長さのラベルを保持します。 ヘルプ Dimension Labels（Multidimensional Waves.ihf）を参照してください。 Igor Pro 8.0 以前では、次元ラベルは 31 バイトに制限されていました。 長い次元ラベルを使う場合は、ウェーブファイルおよび Experiment に Igor Pro 8.0 以降が必要です。
依存式	ウェーブが依存関係にある場合、右側の式を保持します。 := または SetFormula 関数を使って依存関係の代入を実行する場合に設定します。 プレーンの = を使って関数を実行するとクリアされます。
作成日付/時刻	ウェーブが作成された日付と時刻です。
修正日付/時刻	ウェーブが修正された日付と時刻です。
ロック	ウェーブのロック状態です。 ロックされたウェーブは修正できません。 SetWaveLock コマンドで設定します。
ソースフォルダー	ウェーブのソースファイルを含むフォルダーを識別します（存在する場合）。
ファイル名	ウェーブのソースファイル名です（存在する場合）。

テキストエンコーディング	ヘルプ Wave Text Encoding（Text Encodings.ihf）を参照してください。