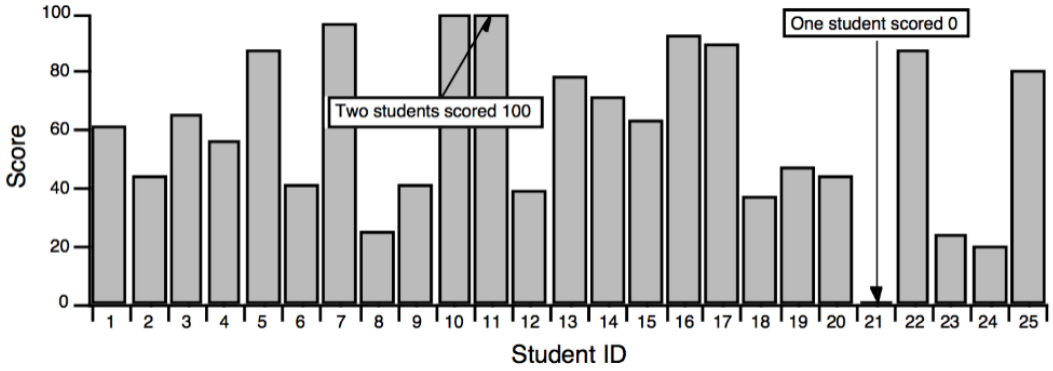
CONTENTS

ビジュアルヘルプ – 分析（2）

ヒストグラム

ヒストグラムは、通常、等間隔に設定された複数の値のビン（範囲）ごとに、そのビン内に収まる入力値の数を合計します。
例えば、ヒストグラムは、0-10、10-20、20-30 などの各ビンに収まるデータ値の数を数えるのに役立ちます。
この計算は、生徒のテストの成績を示すために行われることがよくあります。



この場合、ヒストグラムの一般的な用途は、特定の数値範囲（通常は A、B、C、D の各成績に対応する範囲）に該当する生徒数を把握することです。
教師が 0～100 の範囲を 4 等分し、各成績ごとに 1 ビンずつ割り当てることにしたとします。
ヒストグラムコマンドを使うことで、4 つのビンそれぞれに該当する生徒数を数え、各成績を取得する生徒数を示すことができます。

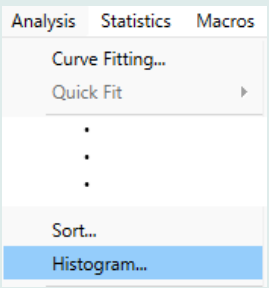
1. サンプルのデータを作成します。

```
Make/N=25/D
scores={60,42,64,52,84,40,92,24,34,100,100,38,80,78,66,90,88,38,44,42,0,80,24,20,78}
```

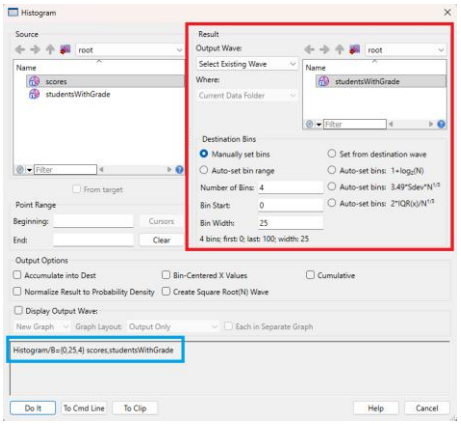
2. ヒストグラムの出力を保持するウェーブを作成します。

```
Make/O/D/N=4 studentsWithGrade
```

3. メニュー Analysis→Histogram を選択します。



4. Histogram ダイアログで次のように設定します。



Output Wave → studentsWithGrade ウェーブ
Manually set bins を選択
Number of bins → 4
Bin Width → 25

コマンドは次のようになっていることを確認してください。

```
Histogram/B={0,25,4} scores,studentsWithGrade
```

/B フラグはヒストグラムに、0 から始まり、ビン幅 25 で 4 つのビンを作成するよう指示します。
最初のビンは 0 から 25 未満までの値をカウントします。

5. Do It をクリックします。

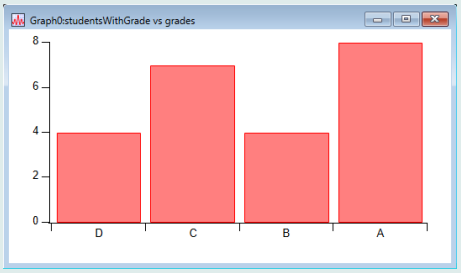
Histogram コマンドはソースウェーブ（scores）を分析し、ヒストグラム結果を宛先ウェーブ（studentsWithGrade）に格納します。

6. 生徒の成績をカテゴリプロットで成績文字対学生の成績としてプロットするためのテキストウェーブを作成します。

```
Make/O/T grades = {"D", "C", "B", "A"}
```

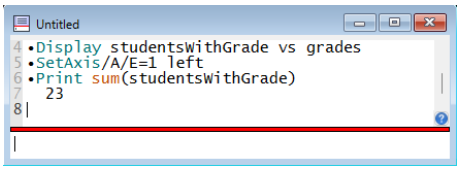
7. カテゴリプロットを作成します。

```
Display studentsWithGrade vs grades
SetAxis/A/E=1 left
```



8. カテゴリプロットは問題なさそうです。
すべての生徒が正しく分類されているか再確認します。

```
Print sum(studentsWithGrade)
       23
```



行方不明の生徒が 2 人います。
彼らはテストで 100 点を取った生徒です。
私たちが定義した 4 つの分類は実際には次のようになります。

Bin 1: 0 - 24.99999
Bin 2: 25 - 49.99999
Bin 3: 50 - 74.99999
Bin 4: 75 - 99.99999

問題は、テストの得点値が実際には 100 ではなく 101 の値を含むことです。
最後のビンに満点を含めるためには、ビンの幅に 0.001 のような小さな数値を加えることができます。

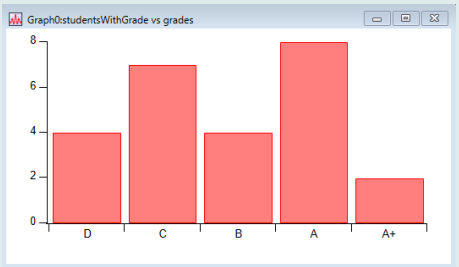
Bin 1: 0 - 25.00999
Bin 2: 25.001 - 49.00199
Bin 3: 50.002 - 74.00299
Bin 4: 75.003 - 100.0399

ただし、25 点、50 点、75 点を取得した生徒は 1 段階下の評価に移行されてしまいます。
おそらく最善の解決策は、満点取得者用の新たな評価ビンを追加することです。

9. コマンドラインで次を実行します。

```
Make/O/T grades= {"D", "C", "B", "A", "A+"}
Histogram/B={0,25,5} scores,studentsWithGrade
```

テキストウェーブに ”A+” を追加し、Histogram ダイアログでは、Number of bins を 5 にします。



ヒストグラムの作図に関する情報は、ヘルプ Category Plots と「ヒストグラム結果のグラフ化」のセクションを参照してください。

この例は、ヒストグラムのビン設定を選択する時に必要な注意点を指摘することを目的としていました。
今回の例では手動のビン設定を使いました。

Histogram コマンドでは、5 通りの方法でビン設定が可能です。
これらは Histogram ダイアログの Destination Bins ラジオボタンに対応します。

ビンモード	動作
Manually set bins	明示的に指定したパラメーターに基づいて、宛先ウェーブのポイント数と X 方向のスケーリングを設定します。
Auto-set bin range	宛先ウェーブの X 方向スケーリングを設定し、入力ウェーブの値の範囲をカバーします。 宛先ウェーブにおけるポイント数（ビン数）は変更されません。したがって、ヒストグラムを計算する前にポイント数を自身で設定する必要があります。 Histogram ダイアログを使う時、Output Wave メニューから Make New Wave または Auto を選択すると、新規ウェーブに設定するポイント数を指定する必要があります。この場合、指定用の Number of Bins ボックスが表示されます。
Set from destination wave	宛先ウェーブにおける X 方向のスケーリングやポイント数は変更されません。したがって、ヒストグラムを計算する前に、X 方向のスケーリングとポイント数を自身で設定する必要があります。 Histogram ダイアログを使う時、Output Wave メニューから Select Existing Wave を選択した場合にのみ、Set from destination wave ラジオボタンが利用可能になります。
Auto-set bins: $1+\log_2(N)$	入力データを検証し、入力データポイントの数に基づいてビンの数を設定します。Auto-set bin range を選択した場合と同様のビン範囲を設定します。
Auto-set bins: $3.49*Sdev*N^{-1/3}$	入力データを検証し、データポイントの数とデータの標準偏差に基づいてビン数を設定します。Auto-set bin range を選択した場合と同様のビン範囲を設定します。
Auto-set bins: $2*IQR(x)/N^{-1/3}$ (Freedman-Diaconis method)	最適なビン幅を次のように設定します。 $2*IQR(x)/N^{-1/3}$ ここで IQR は四分位範囲（StatsQuantiles 参照）であり、区間は最小値と最大値の間で均等に分布しています。 これは Igor Pro 7.0 で追加されました。

ヒストグラムに関する注意点

Set bins from destination wave モードを使う場合、ヒストグラムを計算する前に、Make コマンドを使って宛先ウェーブを作成する必要があります。
また、SetScale コマンドを使って宛先ウェーブの X 軸スケーリングを設定する必要があります。

Histogram コマンドは、1D ウェーブと多次元ウェーブを区別しません。
多次元ウェーブをソースウェーブとして使った場合、そのウェーブは 1D であるかのように解析されます。
それでも次のように有用な場合があります。
ソースウェーブのデータ値がビンに分類される回数を示すヒストグラムが得られます。

2D または 3D 画像データのヒストグラムを作成したい場合は、画像専用の機能をサポートする ImageHistogram コマンドを使うことをお勧めします。

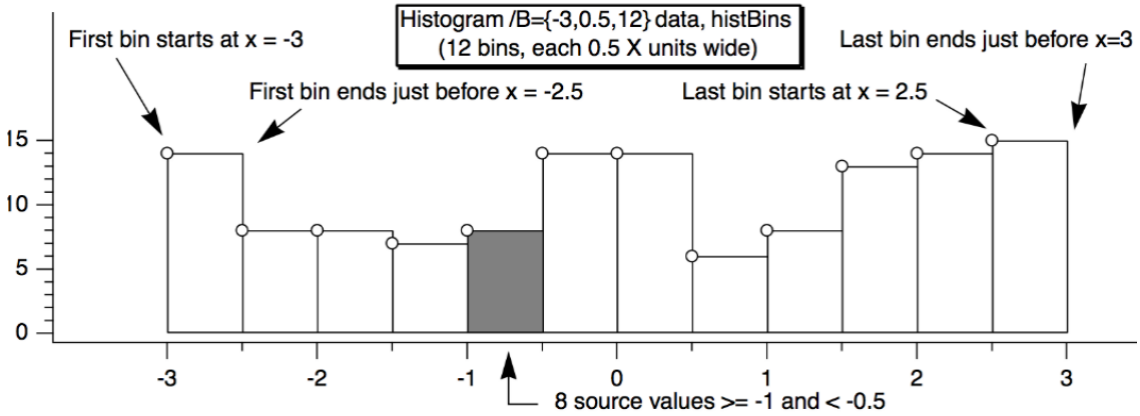
ヒストグラム結果のグラフ化

上記の例では、ヒストグラムの結果がカテゴリプロットとして表示されました。
これは、ビンがテキスト値に対応していたためです。
ヒストグラムのビンは、数値軸上に表示されることがよくあります。
この場合、Igor がヒストグラム結果をどのように表示するかを知る必要があります。

例えば、この histBins 宛先ウェーブは 12 ポイント（ビン）を持ち、最初のビンは -3 から始まり、各ビンの幅は 0.5 です。
X 軸のスケーリングは右の表に示されています。

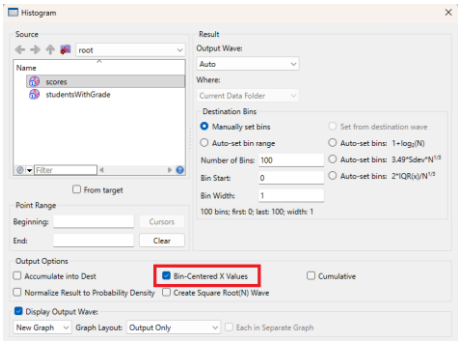
histBins を棒グラフとマーカーモードの両方でグラフ化すると、次のようになります。

Point	histBins.x	histBins.d
0	-3	14
1	-2.5	8
2	-2	8
3	-1.5	7
4	-1	8
5	-0.5	14
6	0	14
7	0.5	6
8	1	8
9	1.5	13
10	2	14
11	2.5	15



マーカーはバーの開始位置に配置されることに注意してください。
マーカーのトレースをビンの中心に表示したい場合は、マーカーをビンの幅の半分だけオフセットできます。

あるいは、Bin-Centered X Values オプションを使って 2 つ目のヒストグラムを作成することもできます。
Histogram ダイアログで、Bin-Centered X Values チェックボックスをオンにします。

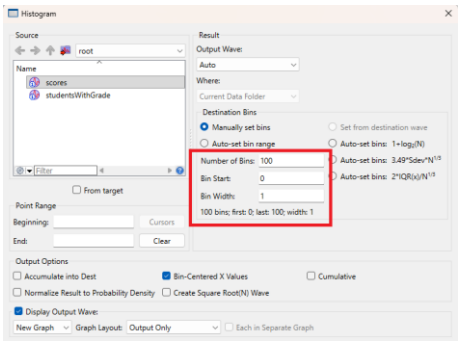


Histogram ダイアログ

Histogram コマンドを使うには、Analysis メニューから Histogram を選択します。

Manually set bins または Set from destination wave のビンモードを使うには、ヒストグラムがカバーするソースウェーブ内のデータ値の範囲を決定する必要があります。
ソースウェーブをグラフ化して視覚的に行うか、WaveStats コマンドを使って正確な最小値と最大値を特定できます。

ダイアログでは、開始ビン値とビン幅の入力が必要です。
開始ビン値と終了ビン値が分かっている場合は、ビン幅を計算するためにいくつかの演算を行う必要があります。



Destination Bins ボックスの下にあるテキスト行には、最初の値と最後の値、およびビンの幅と数が表示されます。
この情報は試行錯誤による設定に役立ちます。

Manually set bins またはいずれかの Auto-set モードを使う場合、Igor は宛先ウェーブの X 単位をソースウェーブの Y 単位と同じに設定します。

Accumulate チェックボックスを有効にすると、ヒストグラムは宛先ウェーブをクリアせず、既存の値にカウントを追加します。
複数のヒストグラムの結果を 1 つの宛先に累積する場合に使います。
この操作を行う場合は、途中でビンの設定を変更することは意味がないため、Auto-set bins オプションを使わないでください。
代わりに、Set from destination wave モードを使ってください。
Accumulate オプションを使うには、宛先ウェーブが倍精度または単精度の実数である必要があります。

Bin-Centered X Values と Create Square Root(N) Wave オプションは、ヒストグラムへのカーブフィッティングに役立ちます。
Bin-Centered X Values を使わない場合、フィッティング関数内の X 位置パラメーターはすべて、半ビン幅分シフトされます。
Square Root(N) Wave は、ヒストグラムデータの標準偏差を推定するウェーブを生成します。
これは、カウントデータがポアソン分布に従うという事実に基づいています。
このオプションで生成されるウェーブは、カウント値がゼロのビンに対して特別な処理を行いません。
したがって、Square Root(N) Wave を使ってカーブフィッティングの重み付けを行う場合、これらのゼロカウントビンはフィッティングから除外されます。
ゼロ値を適切な値で置き換える必要があるかもしれません。

古いバージョンの Igor では、accumulate オプションには 2 つの影響がありました：

- Igor が宛先ウェーブをクリアしない
- Igor が Set bins from destination wave モードを効果的に使う

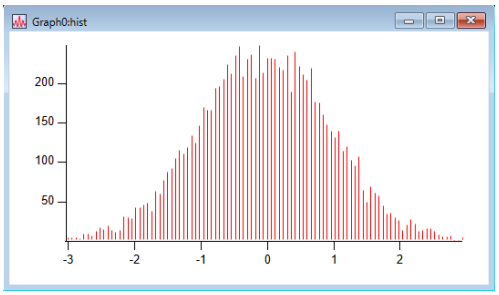
後方互換性を維持するため、累積フラグ（ /A ）が使われ、ビンフラグ（ /B ）が使われていない場合、Histogram コマンドは依然としてこの動作をします。
このダイアログは常にビンフラグを生成します。
したがって、累積フラグは単に累積を強制するだけで、ビン作成には影響しません。

多次元ウェーブに対して Histogram コマンドを使用できますが、データは 1 つの 1D ウェーブに属するものとして扱われます。
2D または 3D のウェーブを扱う場合は、ImageHistogram コマンドの使用が適しています。
このコマンドは 1 レイヤーずつヒストグラムを計算します。

ヒストグラムの例

以下のコマンドは、ヒストグラムコマンドの簡単なテストを示しています。

```
// 再生可能なランダムの設定
SetRandomSeed 0.2
// 生データを作成
Make/O/N=10000 noise = gnoise(1)
// 宛先ウェーブを作成
Make/O hist
// Histogram コマンドを実行
Histogram/B={-3, (3 - -3)/100, 100} noise, hist
Display hist; Modify mode(hist)=1
```



生データは、標準偏差 1.0、平均 0 のガウス分布から抽出された 10,000 個のサンプルで構成されます。
ヒストグラムは幅 6/100＝0.06 の 100 個のビンで作成されます。
ヒストグラムの大きさは以下のように予測されます。

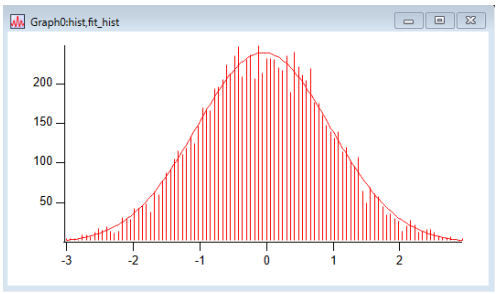
```
A = (N*dx)/sqrt(2*pi)/sigma = (10000*0.06)/sqrt(2*pi)/1 = 239.365
```

ウェーブヒストグラムの X スケーリングは、ヒストグラムのデフォルト設定であるビンの左側を使います。

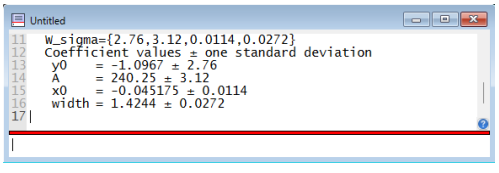
ヒストグラムへのカーブフィッティング

前のセクションのソースウェーブの値は gnoise 関数で生成されたガウス分布の乱数であるため、ヒストグラムはおなじみのガウス分布のベル型曲線となるはずです。
データにガウス曲線をフィッティングさせることで、サンプルが抽出された母集団の特性を推定できます。
まず、サンプルヒストグラムにガウス曲線をフィッティングさせてみます。

```
// ヒストグラムへのカーブフィッティング
CurveFit gauss hist /D
```



カーブフィッティングによる解は右のようになります。

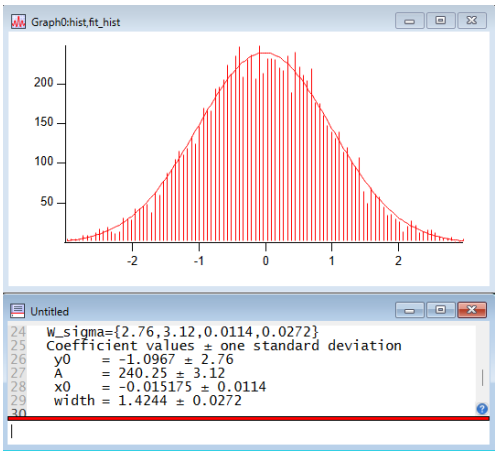


Igor の組み込みガウスフィッティング関数の定義により、幅係数は √2×σ、すなわち 1.4142 となることが予想されます。
このフィットから得られた係数は、ピーク位置 x0 を除き、期待値から 1 標準偏差以内に収まっています。

ピーク位置 x0 は、ゼロより約半ビン下方にシフトしています。
gnoise 関数は平均がゼロの乱数を生成するため、x0 はゼロに近いと予想されます。
x0 のシフト値は、Igor がヒストグラムビンの X 値を保存する方法による結果です。
X 値を左端に設定することは棒グラフの表示には適していますが、カーブフィッティングには不向きです。

デフォルトでは、ヒストグラムのウェーブは X スケーリングが設定されており、X 値はビンの左端の値に対応します。
通常、ヒストグラムに曲線をフィッティングさせる場合、X 値を中央に配置したいものです。
SetScale を使って X スケーリングを中央値に変更できます。
しかし、Histogram コマンドでビンの中心の X 値を持つ出力を生成するオプションを単純に使う方が簡単です。
Histogram コマンドに/C フラグを追加します。

```
// Histogram コマンドを実行
Histogram/C/B={-3, (3 - -3)/100, 100} noise, hist
// ヒストグラムへのカーブフィッティング
CurveFit gauss hist /D
```



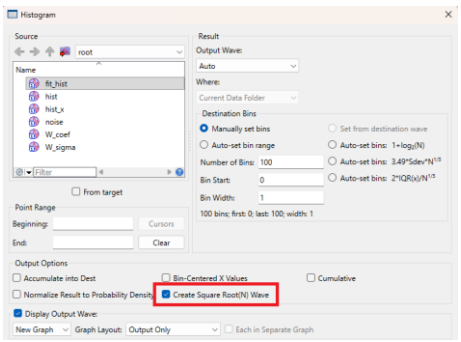
これによりヒストグラムを示すトレースがグラフ上で半ビン分ずれて表示されます。
トレースがマーカーや点で表示されている場合、これが意図した結果である可能性がありますが、バーを使っている場合は表示が不正確です。

別の方法として、ヒストグラムデータに合わせて X ウェーブを作成する方法があります。
この X ウェーブには、半ビン分シフトした X 値が含まれます。
この X ウェーブをカーブフィッティングの入力として使いますが、グラフには表示しないでください。

```
Histogram/B={-3, (3 - -3)/100, 100} noise, hist          // /C を使わないヒストグラム
Duplicate hist, hist_x
hist_x = x + deltax(hist)/2
CurveFit gauss  hist /X=hist_x/D
```

この方法を使うと、X スケーリングを変更せずに元のヒストグラムウェーブをグラフ化できるため、バーを使ったグラフは正しいものになります。
また、中心 X 値を使ったカーブフィッティングも提供され、正しい x0 が得られます。
Histogram コマンドを 2 回使うこともできます。
1 回目は /C フラグを指定してビン中心の X 値を取得し、2 回目はフラグを指定せずにバーに適したシフトされた X スケーリングを取得します。
どちらの方法にも、追跡が必要な余分なウェーブが生成されるという欠点があります。

このフィッティングは統計的に正しくありません。
ヒストグラムはカウントを表すため、ヒストグラムの値はポアソン分布によって記述される不確かさを持つべきです。
ポアソン分布の標準偏差は平均の平方根に等しく、これはヒストグラムのビンの推定不確かさが値の大きさに依存することを意味します。
これはさらに、誤差が一定ではないことを意味し、カーブフィッティングは偏った解を与え、フィッティング係数の不確実性の推定が不正確になる可能性があります。



この問題は重み付けウェーブを用いて近似的に解決できます。
適切な重み付けウェーブは、Histogram コマンドに /N フラグを追加するか、Histogram ダイアログで Create Square Root(N) Wave チェックボックスをオンにすることで生成されます。

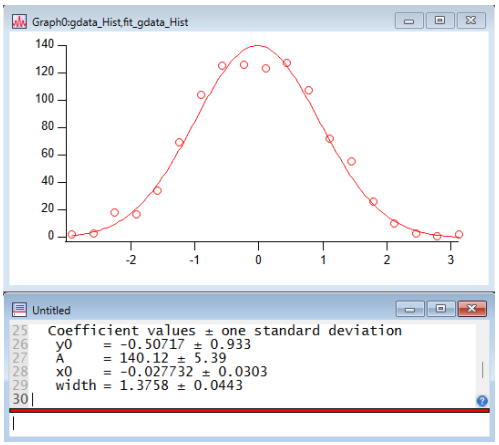
次の例では、gnoise を使ってガウス分布に従う値から新しいデータセットを作成し、ビン中心の X 値と適切な重み付けウェーブを用いたヒストグラムを作成します。
その後、重み付けなしと重み付けありの２つのカーブフィッティングを実行します。

カーブフィッティングから得られる係数値は次の通りと予想されます。

```
y0      =       0
A       =       (1024*deltax(gdata_Hist))/sqrt(2*pi)/1 = 139.863
x0      =       0
width   =       1.4142
```

コマンドラインで次を実行します。

```
// 再生可能なランダムの設定
SetRandomSeed 0.5
Make/O/N=1024 gdata = gnoise(1)
Make/O/N=20 gdata_Hist
Histogram/C/N/B=4 gdata,gdata_Hist
Display gdata_Hist
ModifyGraph mode=3,marker=8
CurveFit gauss  gdata_Hist /D
CurveFit gauss  gdata_Hist /W=W_SqrtN /I=1 /D
```



２番目の CurveFit コマンドに「/W=W_SqrtN /I=1」が追加されている点に注目してください。
これにより、Histogram コマンドで作成された重み付けウェーブを用いた重み付けが追加されます。
また、/B=4 を使うことで、Histogram コマンドが入力データに適したビン数とビン範囲を設定します。

重み付けなし（/W のない CurveFit コマンド）のフィッティング結果：

```
y0      = -3.5134 ± 3.87
A       = 138.82 ± 4.46
x0      = -0.023173 ± 0.03
width   = 1.4922 ± 0.0702
```

重み付けあり（/W のある CurveFit コマンド）のフィッティング結果：

```
y0      = -0.50717 ± 0.933
A       = 140.12 ± 5.39
x0      = -0.027732 ± 0.0303
width   = 1.3758 ± 0.0443
```

結果は明らかに有意に異なります。
偽のデータを作成したため、結果がどうなるか予測可能であり、重み付けフィッティングも予測値により近くなります。

この加重フィッティングに対しては、いくつかの反論が考えられますが、それがあなたにとって重要かどうかは別問題です。

加重フィッティングは、実際の誤差がポアソン分布に従うという事実に対する近似解に過ぎません。
カウントデータをフィッティングする真に正しい方法は、最尤法によるフィッティングです。
Igor は最尤法フィッティングを直接サポートしていません。

ポアソン分布の標準偏差に対する平方根近似を使う場合、フィッティングモデルは個々の元のデータポイントよりも実際のカウントをより良く近似します。
しかし Igor には現在の反復に基づく重み付けを置き換える方法がないため、それを実現する唯一の方法は、フィッティングを実行し、重み付けウェーブを再計算し、再度フィッティングを行うことです。
変化がほとんど見られなくなるまで十分に長く繰り返します。

ポアソン分布の形状は、カウント数が非常に多い場合にのみガウス分布でよく近似されます。
実際には、「非常に多い」とは「5程度以上」を意味する場合があります。
今回の例では、尾部にある5ポイントのみが 5 以下のカウント数を持ちますが、もとは 1000 ポイント以上から始まっています。

反復加重フィッティング

このセクションは上級ユーザーのみ対象としています。
補正済み重み付けウェーブを用いた反復フィッティングの例を示します。

```
Function FitGaussHistogram(Wave histwave, Wave InSqrtNwave)
	// 入力ウェーブを変更しないようにコピーを作成する
	Duplicate/FREE InSqrtNwave, sqrtNwave

	// Histogram が提供する sqrtNwave を使って、正しいフィッティングの初期値を取得する
	CurveFit/Q gauss  histwave /W=sqrtNwave /I=1
	Wave W_coef

	// ループ内での比較用にフィッティング解を保存する
	Duplicate/FREE W_coef, lastCoef

	// 停止条件に使う初期解の長さを計算する
	MatrixOp/FREE/O length = sqrt(sum(magSqr(W_coef)))
	Variable initialLength = length[0]

	// ループを実行し、各反復ごとに重み付けウェーブを再計算する
	// 新しい重み付けで新しいフィッティングを行う
	do
		// 新しい重みは、フィッティングから得られる期待値 Y の平方根
		sqrtNwave = sqrt(Gauss1D(W_coef, pnt2x(histwave, p)))

		// 新しいフィッティングを行う
		CurveFit/Q gauss  histwave /W=sqrtNwave /I=1
		Wave W_coef

		// このフィッティングと前回のフィッティングの差の長さを計算する
		MatrixOp/FREE/O delta = sqrt(sum(magSqr(W_coef - lastCoef)))

		// 進行状況を表示したい場合は、以下の行のコメントを外してください
		// Print delta
		// Print W_coef

		// 少し恣意的な停止基準
		if (delta[0] < 1e-8*initialLength)
			break
		endif

		lastCoef = W_coef
	while(1)

	Wave W_sigma
	Print "y0¥t=¥t",W_coef[0], "±", W_sigma[0]
	Print "A¥t=¥t",W_coef[1], "±", W_sigma[1]
	Print "x0¥t=¥t",W_coef[2], "±", W_sigma[2]
```

```
        Print "width¥t=¥t",W_coef[3], "±", W_sigma[3]
End
```

前のセクションのエクスペリメントの続きで試すには、コードを Procedure ウィンドウに貼り付け、コンパイルし、コマンドラインで以下を実行します。

```
FitGaussHistogram(gdata_Hist, W_SqrtN)
```

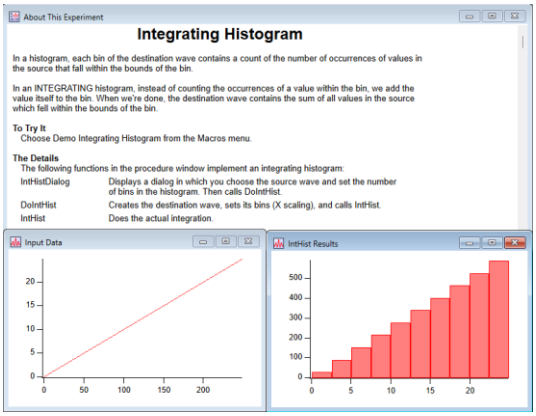
結果は：

```
y0      =       0.181056  ±  0.891992
A       =       139.693  ±  5.4702
x0      =       -0.0296566  ±  0.0310874
width   =       1.38552  ±  0.0444198
```

この例の結果は、ヒストグラムによって生成された重み付けウェーブを用いた重み付けフィッティングの 1 標準偏差の範囲内にあります。

「積分」ヒストグラムの計算

ヒストグラムでは、宛先ウェーブの各ビンには、そのビンの範囲内に収まるソース値の出現回数がカウントされます。
積分ヒストグラムでは、ビン内の値の出現回数をカウントする代わりに、その値自体をビンに加算します。
処理完了時、宛先ウェーブにはそのビンの範囲内に収まるソース値の合計値が含まれます。



Igor には「Integrating Histogram」というサンプルエクスペリメントが付属しており、ユーザー関数を用いたこの手法の実装例を示しています。
このサンプルを確認するには、File→Example Experiments→Analysis→Integrating Histogram を選択してください。

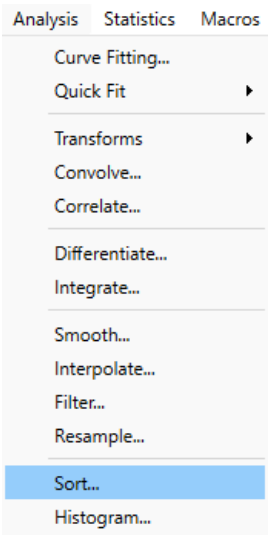
ソート

Sort コマンドは、 1 つ以上の 1D 数値またはテキストウェーブを昇順または降順でソートします。

Sort コマンドは、後続の解析のためにウェーブまたは XY ペアを準備するためによく使われます。
例えば、interp 関数は、入力ウェーブ X が単調増加であると想定しています。

その他のソート関連コマンドとして、MakeIndex と IndexSort があります。
これらはまれなケースで使われ、「MakeIndex と IndexSort」のセクションで説明されています。
SortColumns コマンドは多次元ウェーブの列をソートします。
文字列リストのソートについては、SortList 関数も参照してください。

Sort コマンドを使うには、Analysis メニューから Sort を選択します。

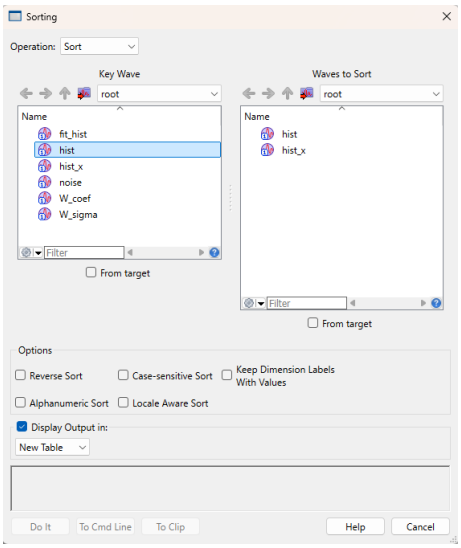


ソートキーウェーブはポイントの再順序付けをコントロールします。

ただし、キーウェーブ自体が Waves to Sort リストで宛先ウェーブとして選択されない限り、再順序付けされません。

宛先ウェーブのポイント数は、キーウェーブと同じでなければなりません。
ダイアログの Key Wave リストからウェーブを選択すると、Igor は Waves to Sort リストにポイント数が同じウェーブのみを表示します。

キーウェーブは数値またはテキストウェーブにできますが、複素数であってはなりません。
宛先ウェーブはテキスト、実数、または複素数にできますが、MakeIndex コマンドの場合は宛先がテキストまたは実数でなければなりません。



宛先ウェーブの数には、Igor のコマンドバッファの 2,500 バイト制限が適用されます。
非常に多くのウェーブをソートするには、複数の Sort コマンドを連続して使い、キーとなるウェーブは最後の最後にソートするように注意してください。

デフォルトでは、テキストのソートは大文字小文字を区別しません。
大文字小文字を区別するソートを行うには、Sort コマンドに /C フラグを指定してください。

シンプルなソート

最も単純なケースでは、１つのウェーブをソースと宛先の両方として選択します。
そうすれば、ソートはそのウェーブだけをソートするだけです。

XY ペアを X ウェーブが順序通りになるように並べ替えるには、X ウェーブをソースとして選択し、X ウェーブと Y ウェーブの両方を宛先として選択します。

中央値を見つけるためのソート

以下のユーザー定義関数は、ウェーブの中央値を求めるための Sort コマンド操作の簡単な使用例を示しています。

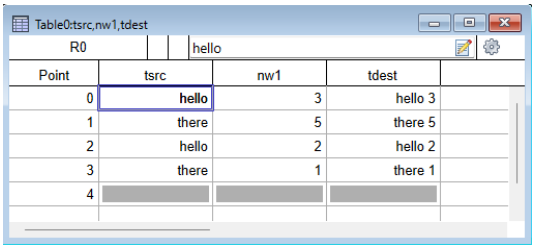
```
Function FindMedian(w, x1, x2)                          // ウェーブ w の中央値を返す
        Wave w
        Variable x1, x2                                 // 興味のある範囲

        Variable result

        Duplicate/R=(x1,x2)/FREE w, medianWave          // ウェーブのクローンを作成
        Sort medianWave, medianWave                     // クローンをソート
        SetScale/P x 0,1,medianWave
        result = medianWave((numpnts(medianWave)-1)/2)

        return result
End
```

ウェーブ内の中心値を見つけるには、組み込みの median 関数を使う方が簡単で高速です。

複数のソートキー

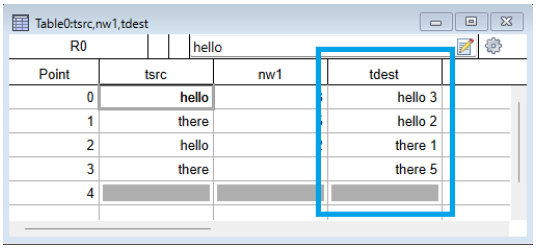
キーウェーブに同一値が複数存在する場合は、二次ソースを使って宛先における対応するポイントの順序を決定する必要があるかもしれません。
これには複数のソートキーを使う必要があります。
Sorting ダイアログでは複数のソートキーを指定できませんが、Sort コマンドでは可能です。
以下に、単一キーと複数キーによるソートの違いを示す例を示します。
ソート後のウェーブ（tdest）がテキストウェーブであり、ソートキーがテキスト（tsrc）と数値（nw1）である点に注意してください。

```
Make/O/T tsrc={"hello","there","hello","there"}
Duplicate/O tsrc,tdest
Make nw1= {3,5,2,1}
tdest= tsrc + " " + num2str(nw1)
Edit tsrc,nw1,tdest
```



単一キーテキストソートを行います。

```
Sort tsrc,tdest       // nw1 は使わない
```

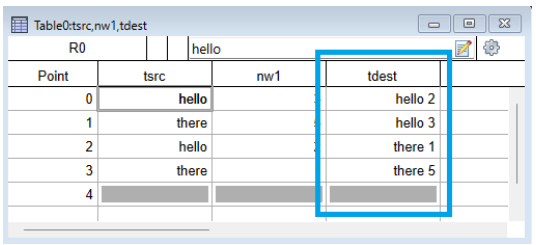


次を実行して、tdest を再度スクランブルします。

```
tdest= tsrc + " " + num2str(nw1)
```

次を実行すると、２つのキーによるソート（nw1 が同じ値を分ける）が表示されます：

```
Sort {tsrc,nw1},tdest
```



「hello 3」が「hello 2」の後にソートされる理由は、nw1[0]=3 が nw1[2]=2 より大きいためです。

中括弧内に２つ以上のウェーブを指定することで、２つ以上のキーでソートできます。

MakeIndex と IndexSort

MakeIndex と IndexSort コマンドは、あまり使われません。
通常は Sort コマンドを使います。

MakeIndex と IndexSort の応用例には以下のようなものがあります。

- 大量のデータの仕分け
- グループ内の個々のウェーブを１つずつ選別
- データを実際に再配置することなく、ソートされた順序でアクセス

- データを元の順序に復元

MakeIndex コマンドは一連のインデックス番号を作成します。
IndexSort はその後、これらのインデックス番号を使ってデータを並べ替え順に再配置できます。
これらを組み合わせることで、Sort コマンドと同様に並べ替えが可能ですが、追加の手順と追加のウェーブを伴います。

利点は、インデックスウェーブを取得すれば、いつでも特定のウェーブのセットからデータを素早くソートできることです。
例えば、数百のウェーブがある場合、コマンドライン１行で通常のソート操作を実行することはできません。
また、プロシージャを記述する時には、複数のウェーブを含む１行のコマンドラインを生成しようとするよりも、ウェーブのセットを１つずつループ処理する方が便利な場合があります。

インデックス値を使えば、IndexSort コマンドを使わずにデータをソート順でアクセスすることも可能です。
例えば、wave1 と wave1index という名前のデータウェーブとインデックスウェーブがある場合、ウェーブ代入の右側で次のようにデータをソート順でアクセスできます。

```
wave1[wave1index[p]]
```

Sort コマンドと同様に、MakeIndex コマンドは複数のソートキーを処理できます。

MakeIndex からの出力ウェーブには、入力ウェーブの要素をソート順にアクセスするためのインデックスが含まれています。
これらのインデックスは後で IndexSort と共に使用し、入力ウェーブをソートしたり他のウェーブの順序を変更したりできます。
例えば：

```
Make/O data0 = {1,9,2,3}
Make/O/N=4 index
MakeIndex data0, index
Print index           // 0, 2, 3, 1 を出力
```

出力値 0、2、3、1 は、data0 をソートする時に、要素 0、次に要素 2、次に要素 3、次に要素 1 にアクセスすることを意味します。
例えば、これは data0 の値を順に表示します。

```
Print data0[0], data0[2], data0[3], data0[1]
```

そのアクセス順序を IndexSort を使って適用できるようになりました。

```
IndexSort index, data0
Print data0           // 1, 2, 3, 9 を出力
```

IndexSort を使って、別のウェーブに対しても同じ並べ替えを適用できます。

```
Make/O data1 = {0,1,2,3}
IndexSort index, data1
Print data1           // 0, 2, 3, 1 を出力
```

デシメーション

大規模なデータセットを扱う場合、より小規模ながらも代表的なポイント数で処理する方が便利な場合があります。
特に、数百万のポイントを持つグラフがある場合、そのグラフを描画または印刷するのに長い時間がかかるでしょう。
グラフを大きく変えずに、多くのデータポイントを省略することが可能です。
デシメーションはこの目的を達成する１つの方法です。

データを減らす方法は少なくとも 2 つあります。

1. n 番目ごとにデータ値を保持します。例えば、最初の値を保持し、9 つを破棄し、次の値を保持し、さらに 9 つを破棄する、というようにです。これを「省略によるデシメーション（Decimation by Omission）」と呼びます。

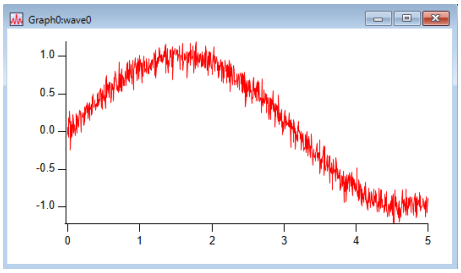
2. n 番目ごとにデータ値を、平均化やフィルタリングなどの計算結果で置き換えます。これを「平滑化によるデシメーション（Decimation by Smoothing）」と呼びます。

省略によるデシメーション

省略によるデシメーションを行うには、より小さな出力ウェーブフォームを作成し、単純なウェーブフォーム算術演算と代入文を使ってその値を設定します。
例えば、10 分の 1 のデシメーション（10 個の値のうち 9 個を省略）を行う場合、入力ウェーブフォームの 10 分の 1 のポイント数を持つ出力ウェーブフォームを作成します。

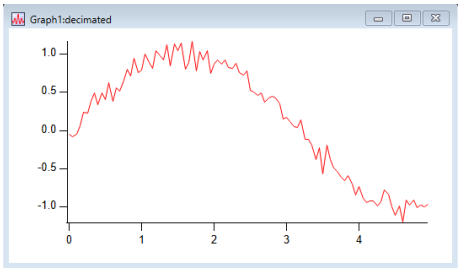
例えば、1000 ポイントのテストウェーブフォームを作成します。

```
Make/O/N=1000 wave0
SetScale x 0, 5, wave0
wave0 = sin(x) + gnoise(.1)
Display wave0
```



デシメーションの結果を格納する 100 ポイントのウェーブフォームを作成します。

```
Make/O/N=100 decimated
SetScale x 0, 5, decimated  // X 範囲を保持
decimated = wave0[p*10]     // for(p=0;p<100;p+=1)
decimated[p]= wave0[p*10]
Display decimated
```



補間係数 1、減算係数（この場合）10、フィルター長 1 を設定して、Resample コマンドとダイアログを使うことで、より容易に省略によるデシメーションを実現できます。

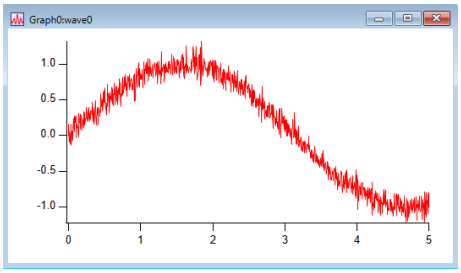
```
Duplicate/O wave0, wave0Resampled
Resample/DOWN=10/N=1 wave0Resampled
```

平滑化によるデシメーション

省略によるデシメーションは一部のデータを完全に破棄するのに対し、平滑化によるデシメーションは全てのデータをデシメーション後の結果に組み合わせます。
平滑化には様々な形態、単純な平均化から様々な種類のローパスデジタルフィルタリングまであります。

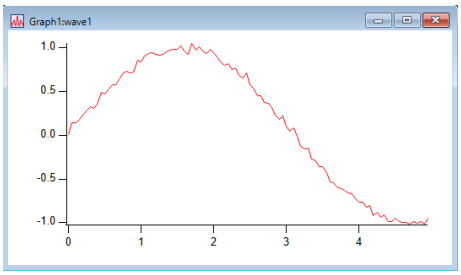
最も単純な平滑化は平均化（ボックスカー平滑化とも呼ばれる）です。
元のデータセットの特定のデータポイントを平均化することで、サンプリング間隔を狭めることができます。
1000 ポイントのデータがあれば、10 ポイントごとに 1 ポイントに平均化することで 100 ポイントの表現を作成できます。
例えば、1000 ポイントのウェーブフォームを作成します。

```
Make/O/N=1000 wave0
SetScale x 0, 5, wave0
wave0 = sin(x) + gnoise(.1)
Display wave0
```



デシメーションの結果を格納する 100 ポイントのウェーブフォームを作成します。

```
Make/O/N=100 wave1
SetScale x 0, 5, wave1
wave1 = mean(wave0, x, x+9*deltax(wave0))
Display wave1
```



出力ウェーブである wave1 のデータポイントは、入力ウェーブの 10 分の 1 であることに注目してください。

ウェーブフォーム代入によって平均化が行われます。

```
wave1 = mean(wave0, x, x+9*deltax(wave0))
```

これは右辺の式を 100 回評価します。
wave1 の各ポイントごとに 1 回ずつです。
シンボル「x」は、評価対象のポイントにおける wave1 の X 値を返します。
右辺の式は、評価対象の wave1 のポイントに対応する区間における wave0 の平均値を返します。

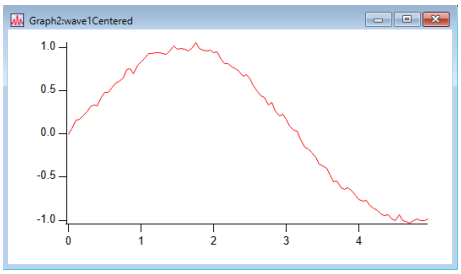
出力ウェーブの X 値は入力範囲の X 値と同じ範囲をカバーすることが不可欠です。
この単純な例では、SetScale コマンドがこの要件を満たしています。

上記の例と同様の結果は、Resample コマンドとダイアログを使うことでより簡単に得られます。

Resample は、一般的なサンプリングレート変換アルゴリズムに基づいており、オプションで補間、ローパスフィルター処理を行い、その後オプションでデータの省略によるデシメーションを行います。
ローパスフィルターは「なし」に設定でき、これは保持されたデータポイントを中心とする奇数個の値を平均化します。
したがって、10 分の 1 のデシメーションでは、10 番目のポイントを中心とする 11 個の値を平均化することになります。

上記の平均化によるデシメーションは、保持されたデータポイントの開始ポイントから 10 個の値ではなく、保持されたデータポイントを中心とする 11 個の値に変更できます。
具体的には以下の方法です。

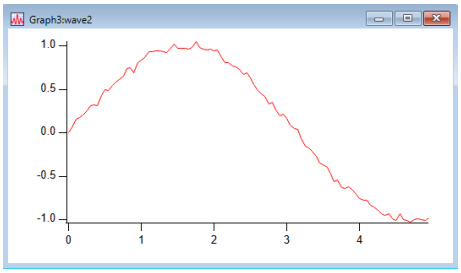
```
Make/O/N=100 wave1Centered
SetScale x 0, 5, wave1Centered
wave1Centered = mean(wave0, x-5*deltax(wave0),
        x+5*deltax(wave0))
Display wave1Centered
```



各デシメーション結果（各平均値）は、wave1 が使った値とは異なる値から形成されますが、元のデータを表すものとして有効性が損なわれるわけではありません。

Resample コマンドを使うと：

```
Duplicate/O wave0, wave2
// /UP がないことは補間がないことを示す
Resample/DOWN=10/WINF=None/N=11 wave2
Display wave2
```



これは wave1Centered = mean(...) の計算とほぼ同じの結果を与えます。
例外は初期値と最終値のみで、これらは単純な端点効果の変動に過ぎません。

Resample の /WINF と /N フラグは、様々な平滑化によるデシメーションに対して単純なローパスフィルタリングオプションを定義します。
デフォルトの /WINF=Hanning ウィンドウは、/WINF=None よりも滑らかな結果を与えます。
これらのウィンドウオプションの詳細については、WindowFunction コマンドを参照してください。

多次元のデシメーションについては、2D、それ以上の次元のウェーブのデシメーションに関する議論を参照してください。

その他のコマンド

WaveTransform の使用

大量のデータ（多数のウェーブまたは複数の大規模なウェーブ）を扱う場合、様々なウェーブ割り当てを、非常に高速に実行でれるウェーブコマンドで置き換えることが役立ちます。
WaveTransform コマンドは、こうした状況での支援を目的として設計されています。
例えば、1D ウェーブのデータを反転させるには、以下のコードを実行します。

```
Function FlipWave(inWave)
        wave inWave

        Variable num=numPnts(inWave)
        Variable n2=num/2
        Variable i,tmp
        num-=1
        Variable j
        for(i=0;i<n2;i+=1)
                tmp=inWave[i]
                j=num-i
                inWave[i]=inWave[j]
                inWave[j]=tmp
        endfor
End
```

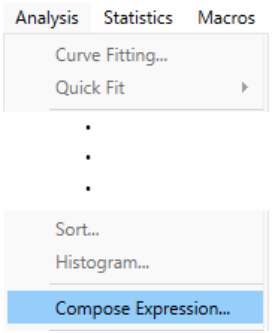
次のコマンドを使うと、同じ結果をはるかに速く得られます。

```
WaveTransform/O flip, waveName
```

「反転」に加え、WaveTransform ではウェーブをポイントインデックスまたは逆ポイントインデックスで埋める、データポイントをシフトする、正規化する、複素共役に変換する、二乗振幅や位相を計算することも可能です。

多次元ウェーブの場合、WaveTransform の代わりに MatrixOp を使ってください。
詳細は「MatrixOp の使用」のセクションを参照してください。

Compose Expression ダイアログ

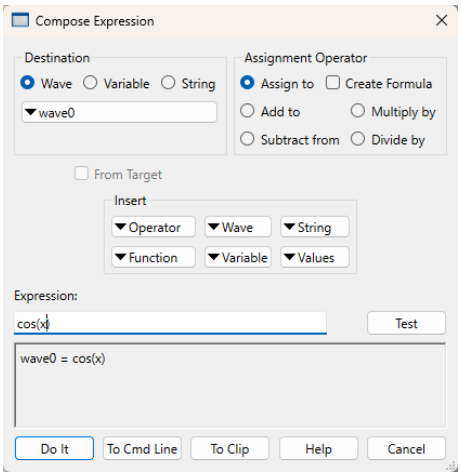


Analysis メニューの Compose Expression を選択すると、Compose Expression ダイアログが表示されます。

このダイアログは、ポイント＆クリックで作成した数値または文字列式に基づいて、ウェーブ、変数、または文字列の値を設定するコマンドを生成します。
ダイアログを使って生成できるコマンドは、コマンドラインに直接入力することも可能です。

Compose Expression ダイアログで生成するコマンドは、３つの部分で構成されます：宛先、代入演算子、式です。
コマンドは数式に似ており、次の形式です。

```
<destination> <assignment-operator> <expression>
```

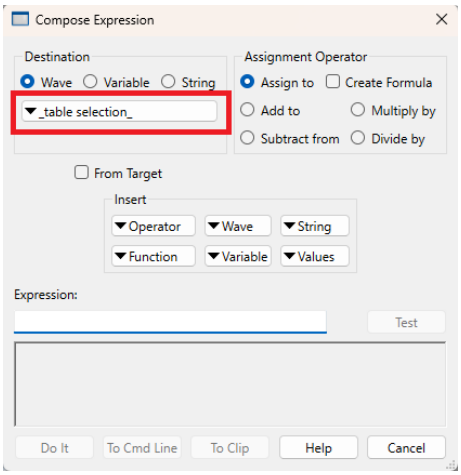


例えば：

```
wave1 = K0 + wave2              // ウェーブの代入コマンド

K0 += 1.5 * K1                  // 変数の代入コマンド

str1 = "Today is" + date()      // 文字列の代入コマンド
```

table selection 項目

Destination Wave ポップアップメニューには「_table selection_」項目が含まれます（テーブルを表示すると選択項目に出てきます）。
table selection を選択すると、Igor は式をテーブル内で選択されている対象に代入します。
これは１つのウェーブ全体、複数のウェーブ全体、あるいは１つ以上のウェーブのサブセットである可能性があります。



この機能を使うには、まずテーブル内で代入したい数値ウェーブを選択します。
次に、Analysis メニューから Compose Expression を選択します。
Destination Wave ポップアップメニューで _table selection_ を選択します。
次に、ウェーブに代入する式を入力します。
ダイアログの下部にあるコマンドボックスに表示される Igor が作成したコマンドに注目してください。
ウェーブのサブセットを選択している場合、Igor はその部分のみのコマンドを生成します。
最後に、Do It をクリックしてコマンドを実行します。

Create Formula チェックボックス

Compose Expression ダイアログの Create Formula チェックボックスをオンにすると、Igor は = 演算子ではなく := 演算子を使ってコマンドを生成します。
:= 演算子は依存関係を確立します。

これにより、代入文の右辺にあるウェーブまたは変数が変更されると、Igor は左辺（代入先）に値を再代入します。
右辺を式（Formula）と呼びます。依存関係と式の詳細については、ヘルプ Dependencies を参照してください。