CONTENTS

Igor Pro ガイドツアー（3 – ヒストグラム&カーブフィッティング）

ホワイトノイズのヒストグラム

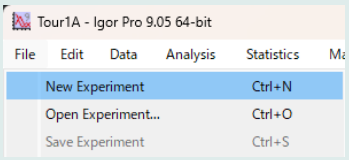
このガイドツアーでは、ヒストグラムの操作を説明し、重みづけを使ったカーブフィッティングを説明します。

オプションの最後の部分では、残差プロットを作成し、履歴のコマンドから有用なプロシージャを作成する方法を示します。

ここでは、均等に分布した「ホワイトノイズ」のヒストグラムを生成します。

1. メニュー File → New Experiment を選択します。

メニュー Misc → Preferences Off を選択して、プリファレンスがオフになっているようにします。

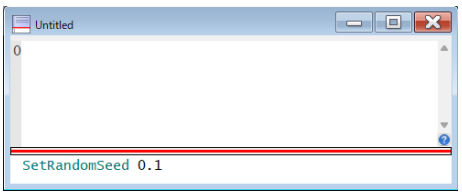


2. 分析に何らかのデータが必要なので、ランダムな値を生成します。

コマンドラインに次を入力して、Enter キーを押します。

```
SetRandomSeed 0.1
```

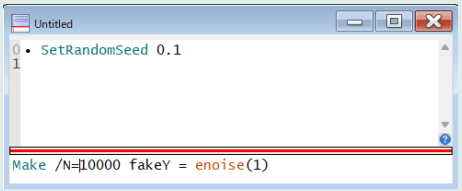
これは乱数ジェネレーターを初期化し、このガイドツアーで使うデータを生成します。



3. コマンドラインに次を入力して、Enter キーを押します。

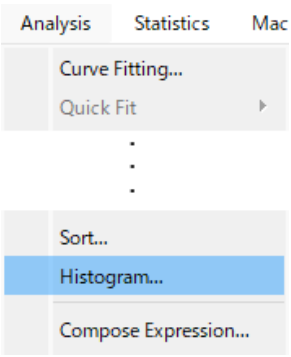
```
Make /N=10000 fakeY = enoise(1)
```

これは -1 から 1 までの均等に分散したランダムな値の 10,000 点のウェーブを生成します。



4. メニュー Analysis → Histogram を選択します。

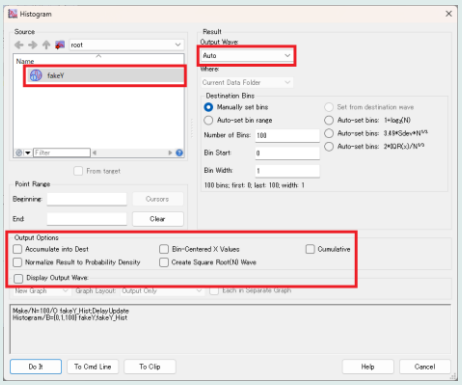
Histogram ダイアログが表示されます。



5. **Sources のリストから fakeY を選択します。**

Output Wave のプルダウンメニューで Auto を選択します。

Display Output Wave を含む、すべてのチェックボックスを外します。

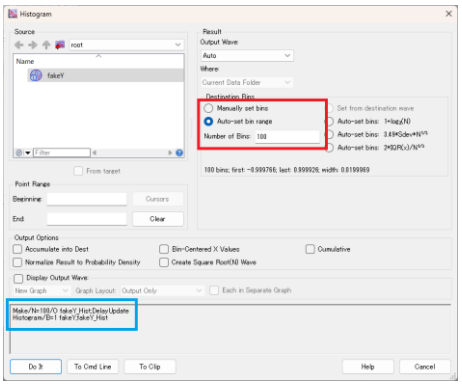


6. **Destination Bins の Auto-set Bin Range ラジオボタンをクリックします。**

画面下のコマンドボックスには次の 2 つのコマンドが表示されているはずです。

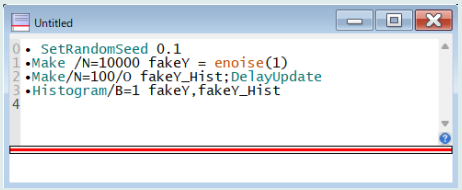
```
Make /N=100/O fakeY_Hist; DelayUpdate
Histogram /B=1 fakeY,fakeY_Hist
```

最初のコマンドは、結果を受け取るウエーブを作成し、2 つ目のコマンドは分析を実行します。
Auto-set bin range モードで Histogram を実行すると、出力ウェーブからビン数を取り出します。



7. **Do It をクリックします。**

ヒストグラムの処理が実行されます。
ここではグラフは表示されません。

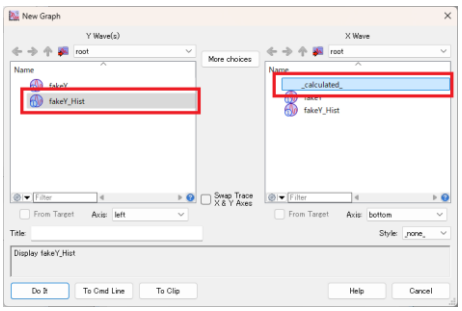


8. 次にヒストグラムを表示します。

メニュー Windows → New Graph を選択します。

New Graph ダイアログで、Y Waves リストで fakeY_Hist、X Wave リストで _calculated_ を選択します。

Do It をクリックします。

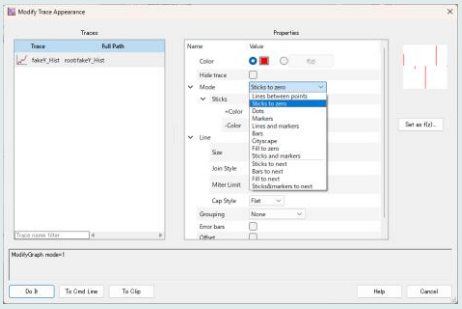


9. 表示されたグラフを少し触ります。

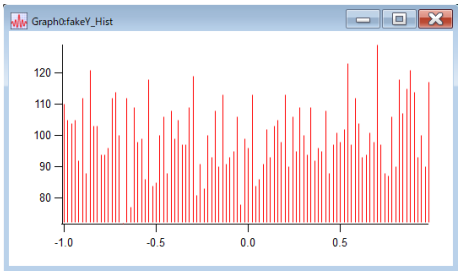
グラフのトレースをダブルクリックしてください。

Modigy Trace Appearance ダイアログが表示されます。

Mode ポップアップメニューから Stick to zero を選択して、Do It をクリックします。

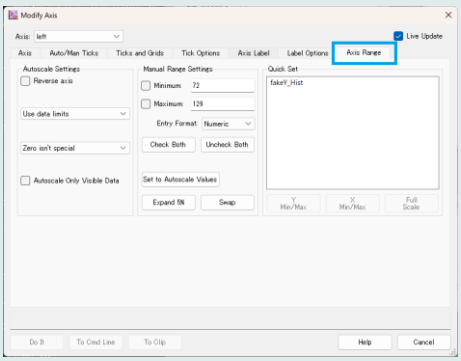


10. グラフが新しいモードで再描画されます。

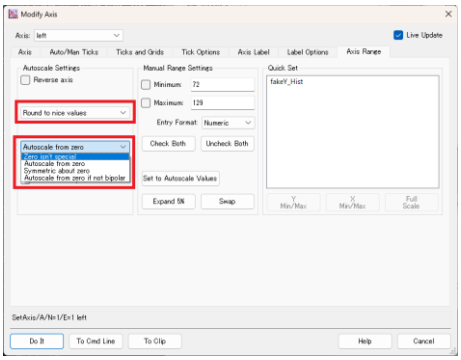


11. Y 軸の目盛の1つ（例えば、100）をダブルクリックします。

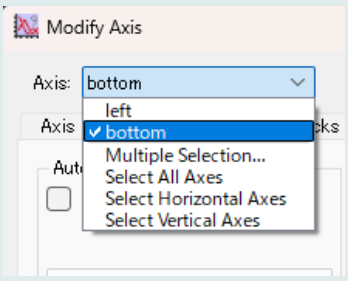
Axis Range タブが表示されている Modify Axis ダイアログが開きます。



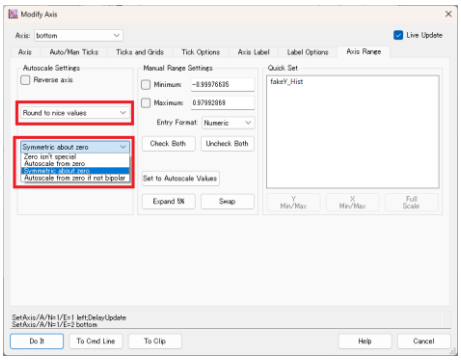
12. Autoscale Settings の2つのポップアップメニューからそれぞれ Round to nice values と Autoscale from zero を選択してください。



13. 今度は X 軸を設定するために、左上の Axis ポップアップメニューから bottom を選択します。

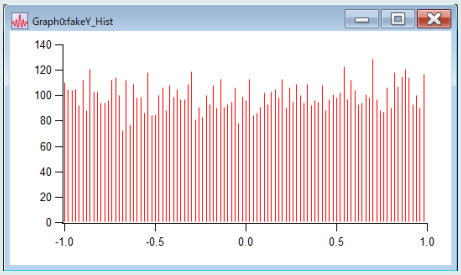


14. Autoscale Settings の2つのポップアップメニューからそれぞれ Round to nice values と Symmetric about zero を選択してください。



15. **Do It をクリックするとグラフは次のようになります。**

軸の表示範囲が変わっていることがわかります。



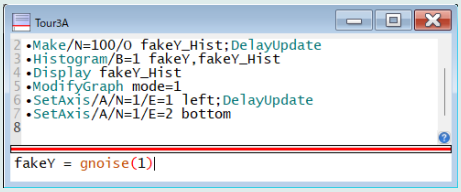
16. **メニュー File → Save Experoment As を選択し、**
"Tour3A.pxp" という名前で保存します。

ガウスノイズのヒストグラム

ここではガウスノイズでヒストグラムを描いてみます。
前のセクションの続きから操作します。
閉じている場合は、Tour3A.pxp を開いたところから始めます。

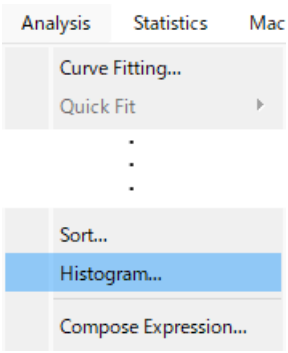
1. コマンドウィンドウに次を入力して、Enter キーを押します。

```
fakeY = gnoise(1)
```



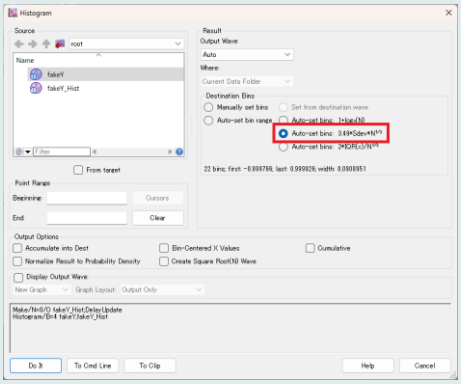
2. メニュー Analysis → Histogram を選択します。

前回設定した状態でダイアログが開きます。



3. Destination Bins のラジオボタンで「Auto-set bins: 3.49*Sdev*N^-1/3」を選択します。

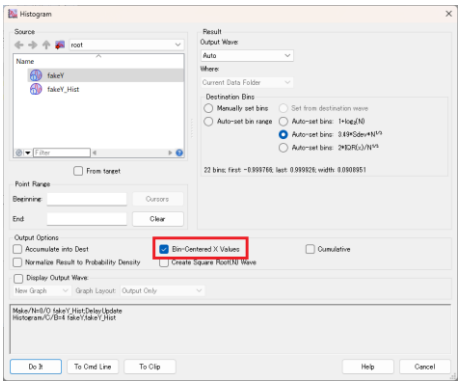
すぐ下の情報が表示される領域に、このヒストグラムは、ここでは 22 のビンを持つことが示されます。
これは、ヒストグラムの適切なビン数を選択する Sturges による方法です（詳細はマニュアルの V-349 Histogram を参照）。



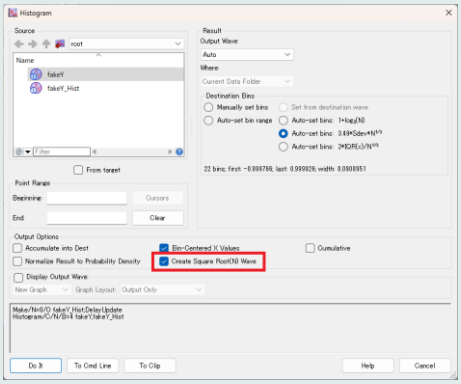
4. 下の Bin-Centered X Values チェックボックスをチェックします。

デフォルトでは、ヒストグラムは、X 値が各ビンの左端にあり、右端が次の X 値によって与えられるように、出力ウェーブの X スケーリングを設定します。

以降の手順で、ヒストグラムにカーブフィッティングするため、X 値が各ビンの中央にします。



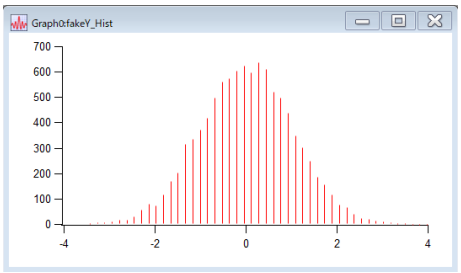
5. その下の Create Square Root(N) Wave チェックボックスをチェックします。



ヒストグラムのようなカウントデータは、通常、ポアソン分布の値を持ちます。
ポアソン分布の推定平均は単純にカウント数(N)で、推定標準偏差は N の平方根です。
これを考慮しないと、カーブフィッティングに偏りが生じます。

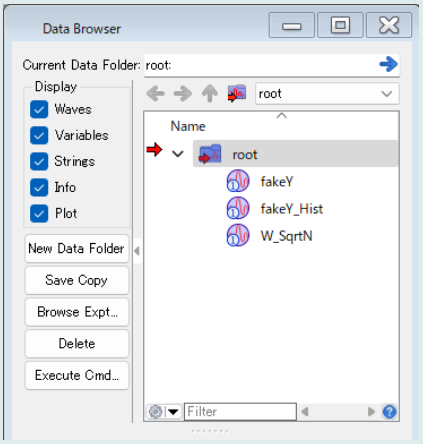
カーブフィッティングを行うときには、この追加のウェーブを重みづけに使います。

6. Do It をクリックします。



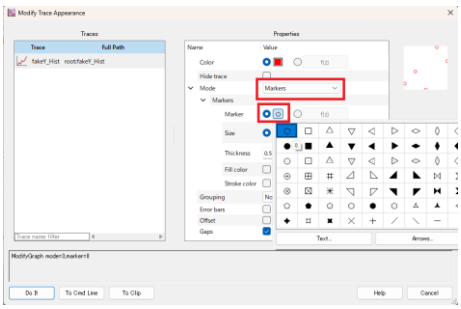
ヒストグラムの入力がガウス分布のノイズであったことから予想できたように、Graph0 に表示されたヒストグラムはガウス分布をしています。

7. Data Browser には3つのウェーブ（fakeY, fakeY_Hist, W_SqrtN）が表示されています。



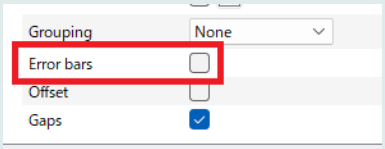
fakeY_Hist はヒストグラムの出力、W_SqrtN は N 個のデータの平方根を受け取るためのヒストグラムによって作成されたウェーブです。

8. Graph0 のトレースをダブルクリックして、Modify Trace Appearance ダイアログを開きます。



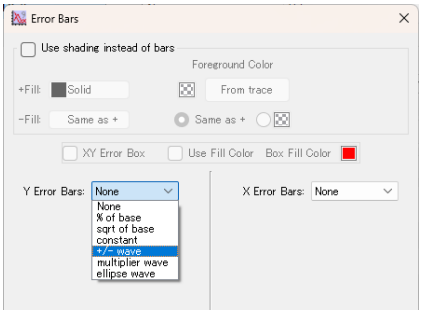
Mode メニューで markers を選択し、「O」を選択します。

9. Error Bars チェックボックスをクリックします。



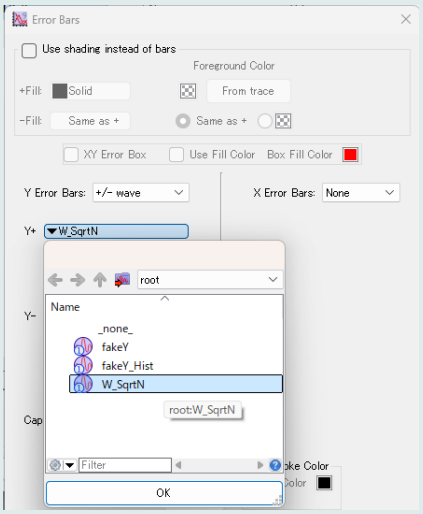
Error bars ダイアログが開きます。

10. **Y Error Bars ポップアップメニューで "+/- wave" を選択します。**



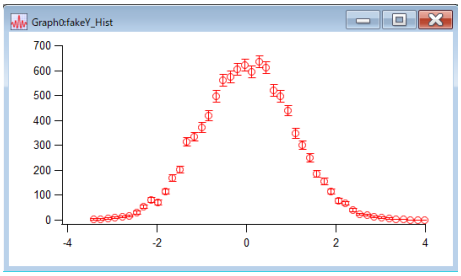
11. **Y+ ポップアップメニューで W_SqrtN を選択します。**

Y- メニューは Same as Y+ に設定されます。
もし、非対称のエラーバーが必要な場合は他の Y- の他の項目を選択します。



12. **OK をクリックし、Do It をクリックします。**

グラフは次のようになります。

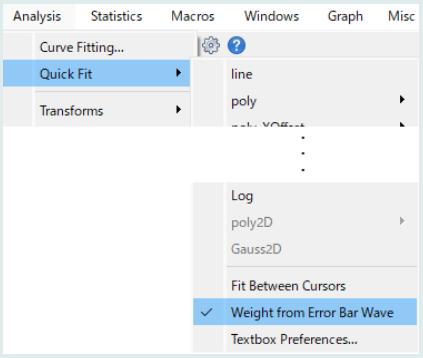


13. **メニュー File → Save Experoment As を選択し、"Tour3B.pxp" という名前で保存します。**

ヒストグラムのカーブフィッティング

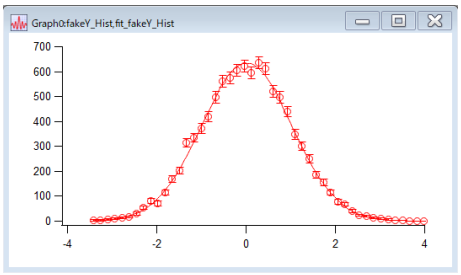
前のセクションの続きから始めます。
閉じている場合は、Tour3B.pxp を開いたところから始めます。

1. Graph0 のウィンドウがアクティブになっているかを確認してください。



メニュー Analysis → Quick Fit を選択し、Weight from Error Bar Wave がチェックされているか確認してください。
チェックされていない場合は、選択してチェックを入れてください。

2. **メニュー Analysis → Quick Fit → gauss を選択します。**



グラフは次のようになります。

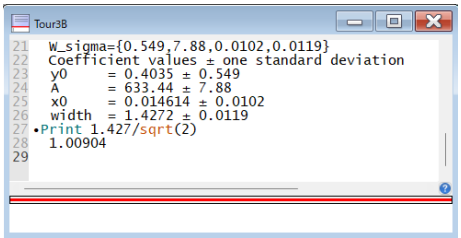
3. 履歴領域に見られるように、フィッティングの結果は次のようになります。

```
Coefficient values ± one standard deviation
y0      = 0.4035 ± 0.549
A       = 633.44 ± 7.88
x0      = 0.014614 ± 0.0102
width   = 1.4272 ± 0.0119
```

もとのデータは標準偏差 1 で生成されています。では、なぜ幅が約 1.4 になっているのでしょうか？
Igor の Gaussian フィッティング関数の定義では、幅は sigma*$2^{1/2}$ です。

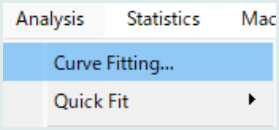
4. **コマンドラインに次を入力して、Enter キーを押します。**

```
Print 1.427/sqrt(2)
```



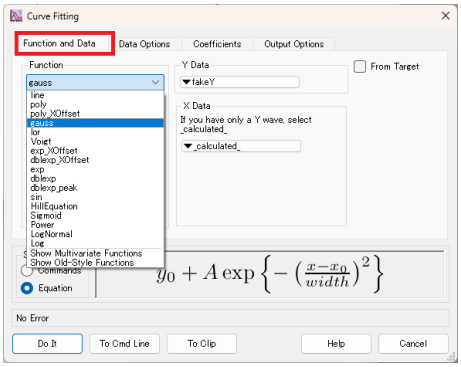
結果は非常に 1.0 に近くなっています。

5. いろいろな問題点をチェックするためにフィッティングからの残差をプロットすることは、しばしば役に立ちます。
これを行うには、Curve Fitting ダイアログを使う必要があります。



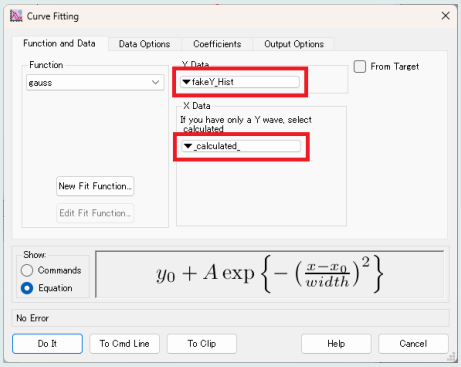
メニュー Analysis → Curve Fitting を選択します。

6. **Function and Data タブを選択し、Function ポップアップメニューから gauss を選択します。**



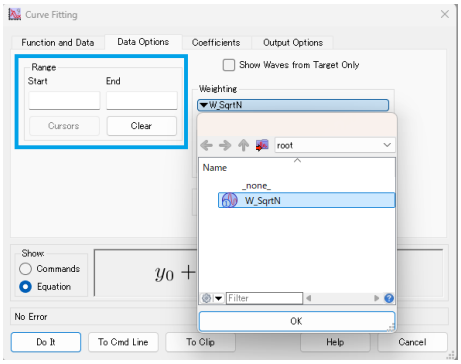
7. **Y Data ポップアップメニューから fakeY_Hist（fakeY ではない）を選択します。**

X Data ポップアップメニューから _calculated_ を選択します。

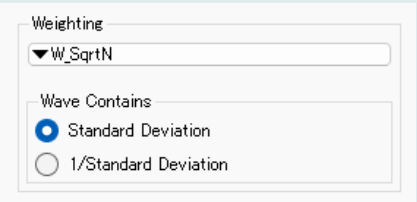


8. **Data Options タブをクリックして、Range の Start と End に値がある場合は、Clear ボタンを押して、空にしてください。**

Weighting のポップアップメニューから W_SqrtN を選択してください。



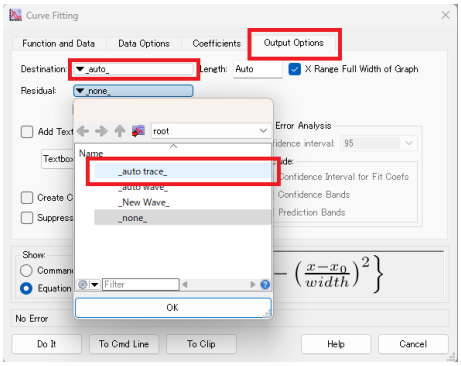
9. **Wave Contains で Standard Deviation ラジオボタンを選択してください。**



10. **Output Options タブをクリックし、Destination ポップアップメニューから _auto_ を選択してください。**

Residual ポップアップメニューから _auto trace_ を選択します。

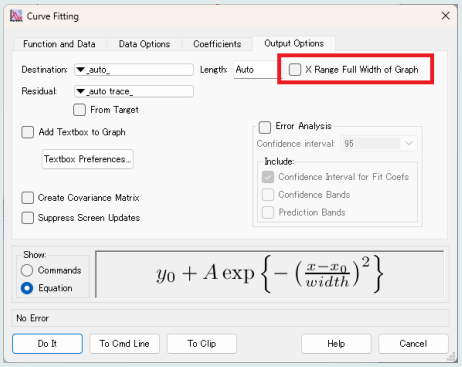
残差は自動的に計算され、グラフのカーブフィッティングに追加されます。



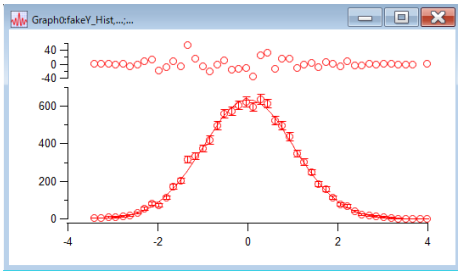
11. X Range Full Width of Graph チェックボックスをオフにします。

これは、ソースのウェーブの一部をフィッティングする場合にのみ使います。

Do It をクリックします。



12. カーブフィッティングが始まり、完了したら OK をクリックします。



13. メニュー File → Save Experoment As を選択し、"Tour3C.pxp" という名前で保存します。

上記では取り上げていない小さな問題が１つあります。
ビンの１つにはゼロが入っています。ゼロの平方根はもちろんゼロです。
そのため、重み付けウェーブにはゼロが含まれており、カーブフィッティングはそのデータポイントを無視することになります。
この問題の解決に最適なアプローチは明らかではありません。
ゼロを１に置き換える人もいます。
次のコマンドは、重み付けウェーブ内のゼロを置き換え、フィッティングをやり直します。

```
W_SqrtN = W_SqrtN[p] == 0 ? 1 : W_SqrtN[p]
CurveFit/NTHR=0 gauss fakeY_Hist /W=W_SqrtN /I=1 /D /R
```

これは結果にあまり影響しません。というのも、ヒストグラムにゼロが１つしかなかったためです。

```
Coefficient values ± one standard deviation
    y0 =-0.40357 ± 0.464
    A =644.76 ± 7.98
    x0 =-0.0014186 ± 0.00996
    width =1.4065 ± 0.0115
```

カーブフィッティングの残差（オプションツアー）

このセクションは、主に Igor プログラミングを使ってタスクを自動化したい人たちのためのものです。
ここでは、履歴領域をプロシージャを生成するコマンドのソースとして使う方法を説明します。
例として、グラフに残差を追加するプロシージャを作成します。

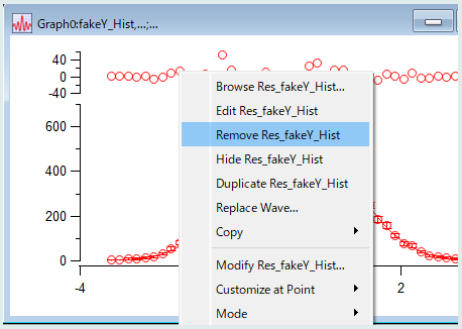
前のセクションで、Igor がカーブフィッティングからの残差を自動的に表示できることを説明しました。
したがって、ここで書くプロシージャはあえて必要はありません。
しかし、プロシージャを作成するプロセスを示しておきます。そこで、プロシージャを書く準備として、残差を手動で追加します。

ガウス関数へのカーブフィッティングがうまくいき、gnoise 関数が本当にガウス分布のノイズを生成するのであれば、ヒストグラムデータとフィッティングした関数の差をプロットしても、湾曲は見られないはずです。

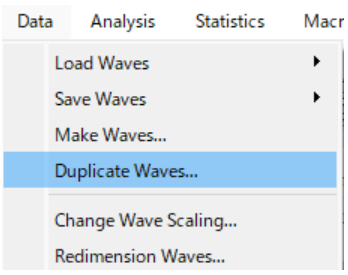
1. 前のセクションの続きから始めます。

Gaussian フィッティングから、自動的に生成された残差を取り除くには、グラフ上部の残差のトレースを直接、右クリックして、ポップアップメニューから Remove Res_fakeY_Hist を選択します。

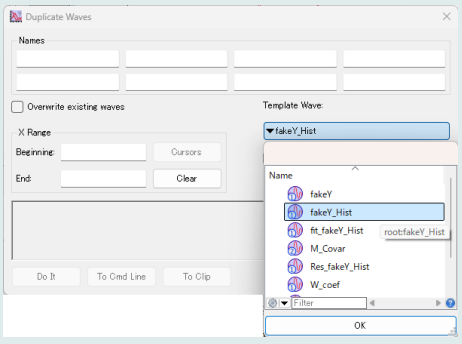
グラフから残差のトレースが削除されます。



2. **メニュー Data → Duplicate Waves を選択します。**



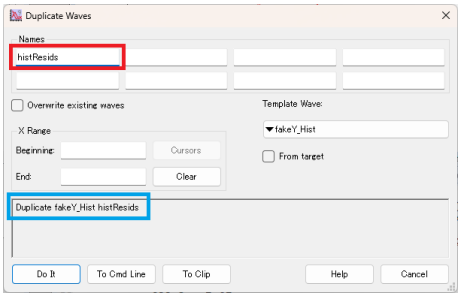
3. **Template Wave ポップアップメニューから fakeY_Hist を選択します。**



4. **最初の名前の入力ボックスに histResids と入力します。**

Do It をクリックします。

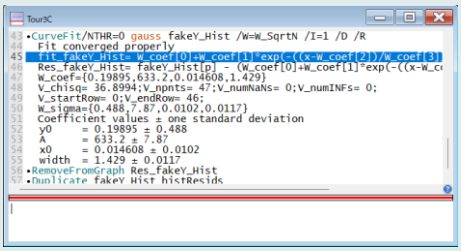
これで、残差を含むのに適したウェーブが作成されました。



5. コマンドウィンドウの履歴領域から、次の行を探してください。

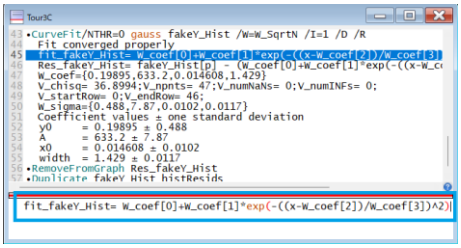
```
fit_fakeY_Hist= W_coef[0]+W_coef[1]*exp(-((x-W_coef[2])/W_coef[3])^2)
```

`W_coef` はフィッティングパラメーターを含む CurveFit 操作で作成されたウェーブです。
`W_coef[0]` は y0 パラメーター、`W_coef[1]` は A パラメーター、`W_coef[2]` は x0 パラメーター、`W_coef[3]` は幅パラメーターです。
この行は、CurveFit 操作がフィッティング先のウェーブのデータを設定するために、どのようなことをしたかを概念的に示しています。



6. **この行を１回クリックして、Enter キーを押してください。**

コマンドラインにコピーされます。



7. **コマンドラインのテキストを次のように書き換えます。**

```
histResids = fakeY_Hist -(W_coef[0]+W_coef[1]*exp(-((x-W_coef[2])/W_coef[3])^2))
```

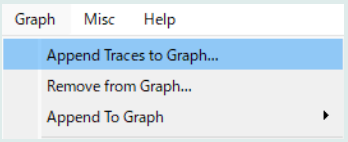
言い換えると、`fit_fakeY_Hist` を `histResids` に書き換え、"=" の後ろに "`fakeY_Hist -`" を追加し、そこから文末までをかっこで囲みます。
追加したかっこの中にある式は、フィッティングによって決定されたパラメーターを使ったモデル値を表しています。
このコマンドは、フィッティングが実行されたデータ値からモデル値を引くことによって、残差を計算します。
もし、フィッティングが計算された X 値ではなく、X ウェーブを使っていた場合は、式の "x" の代わりに X ウェーブの名前に置き換える必要があります。

8. **Enter キーを押します。**

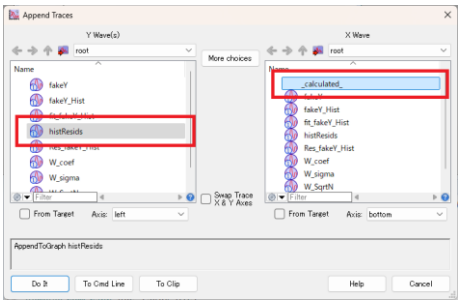
このウェーブへの代入文は、測定データ（Histogram 操作の出力）と理論的なガウス（CurveFit 操作で決定）の差を計算します。

9. 次に、現在の内容の上に残差のグラフを積み重ねます。

メニュー Graph → Append Traces to Graph を選択します。

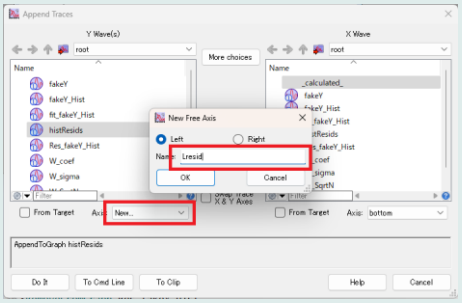


10. **Y Waves のリストから histResids を選択し、X Wames のリストから _calculated_ を選択します。**



11. **Y Waves リストの下の Axis ポップアップメニューから New を選択し、Name のボックスに "Lresid" と入力し、OK をクリックします。**

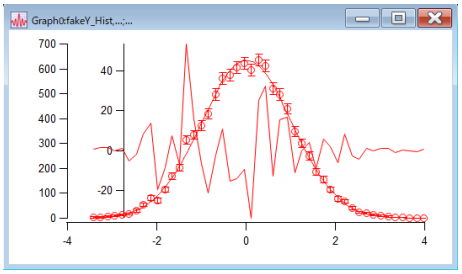
Do It をクリックします。



12. 新しいトレースと軸が追加されました。

次に軸をアレンジする必要があります。

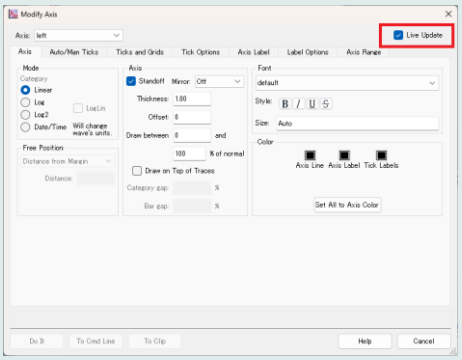
Left 軸と Lresid 軸の間の空いているスペースを分割することでこれを行います。



13. **左の Y 軸をダブルクリックします。**

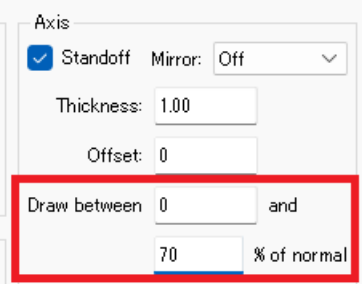
Modify Axis ダイアログが表示されます。
他のダイアログが表示される場合は、キャンセルして、カーソルの位置を確認してもう一度試してください。

十分な画面スペースがあれば、ダイアログで設定を変更するたびにグラフが変化するのを見ることができます。
ダイアログの右上にある Live Update チェックボックスが選択されていることを確認してください。

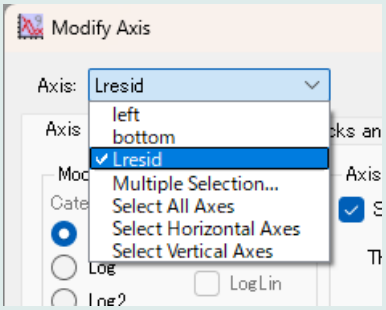


14. **Axis タブをクリックしてください。**

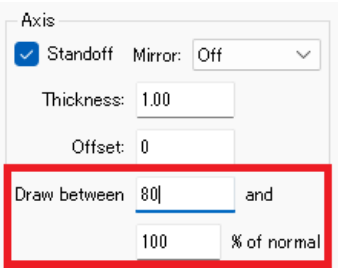
左上の Axis ポップアップメニューですでに Left が選択されているはずです。
ウィンドウの中央部の Axis で Draw between 0 and 70% of normal となるように入力してください。



15. **左上の Axis ポップアップメニューで Lresid を選択します。**

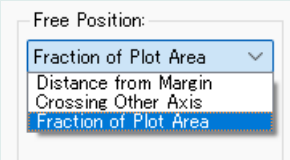


16. **ウィンドウの中央部の Axis で Draw between 80 and 100% of normal となるように入力してください。**

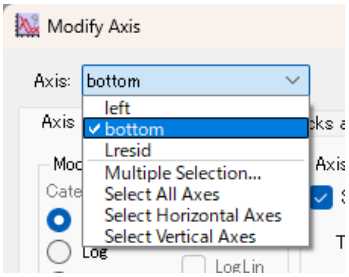


17. **Free Position メニューで Fraction of Plot Area を選択します。**

Lresid 軸は "free" 軸です。これは、水平方向に動き、Left 軸と一直線になります。



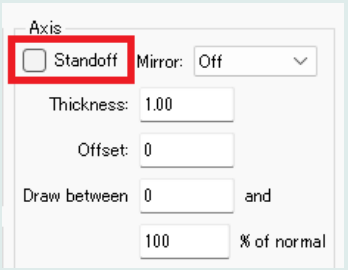
18. **左上の Axis ポップアップメニューで bottom を選択します。**



19. **ウィンドウの中央部の Axis で Standoff チェックボックスをオフにしてください。**

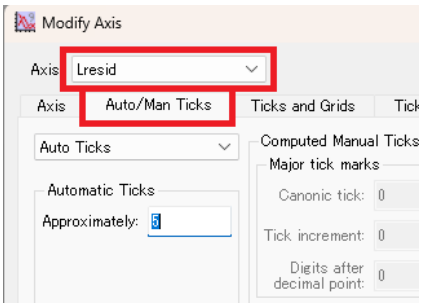
もう少しで完成です。
Lresid 軸の目盛は改良する必要があります。

また、残差データはドット・モードであるべきです。

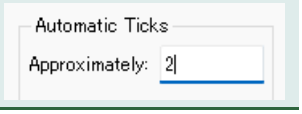


20. **左上の Axis ポップアップメニューで Lresid を選択します。**

Auto/Man Ticks タブをクリックします。



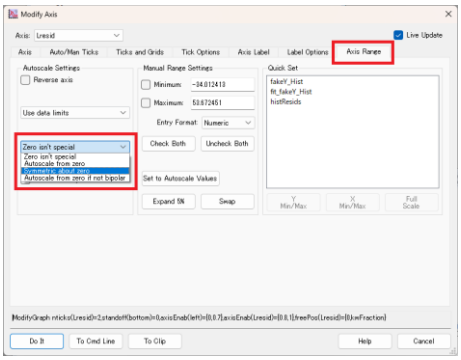
21. **Automatic Ticks の Approximately に 2 を入力します。**



22. **Axis Range タブをクリックします。**

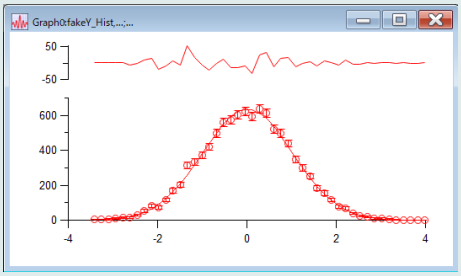
Autoscale Settings 領域で、現在 Zero isn't special と表示されているポップアップメニューを Symmetric about zero に変えます。

Do It をクリックします。



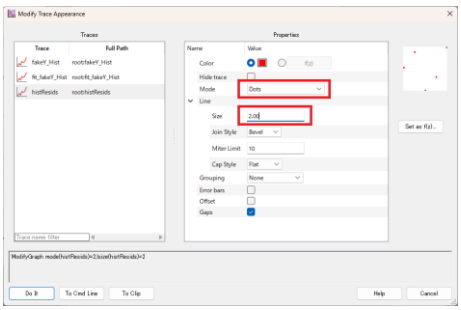
23. **現在折れ線グラフで表示されている histResids トレースをダブルクリックします。**

Modify Trace Appearance ダイアログが表示されます。
histResids がすでにリストで選択されています。



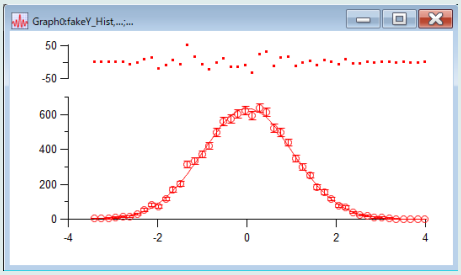
24. **Mode ポップアップメニューで Dots を選択します。**

Line の Size を 2.00 にします。



25. **Do It をクリックします。**

グラフは次のようになります。

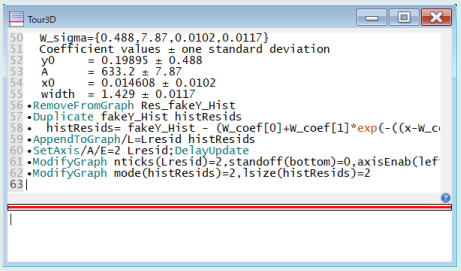


26. **メニュー File → Save Experoment As を選択し、"Tour3D.pxp" という名前で保存します。**

プロシージャの記述（オプションツアー）

このセクションでは、グラフに残差を追加するときに作られたコマンドを集めます。
それらを使って、グラフに残差プロットを追加するプロシージャを作成します。

1. コマンドウィンドウの上端を上にドラッグして、過去 10 行を表示できるぐらいの大きさにします。

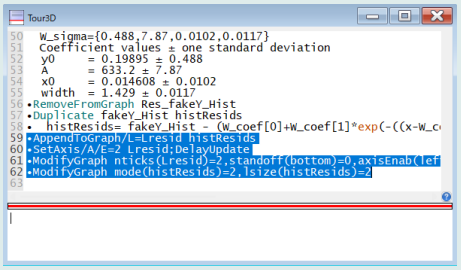


2. 次の行を探します。

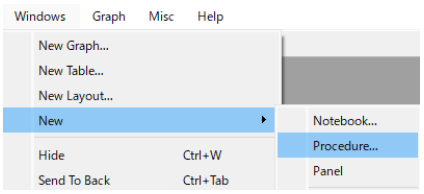
```
•AppendToGraph/L=Lresid histResids
```

3. この行とそれより下の行をすべて選択して、クリップボードにコピーします。

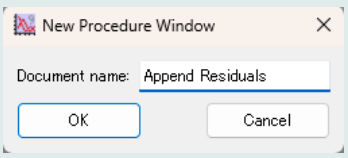
※前のセクションでの、画面での設定や操作順などによって、各行の内容に違いがある場合があります。ここでは、プロシージャの作成手順を示しているので、あまり気にしないで進みましょう。



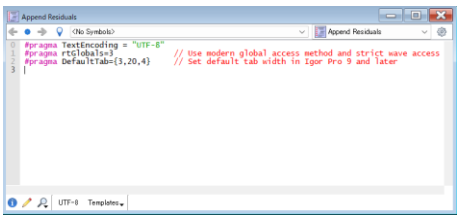
4. **メニュー Windows → New → Procedure を選択します。**



5. **Append Residuals と入力し、OK をクリックします。**



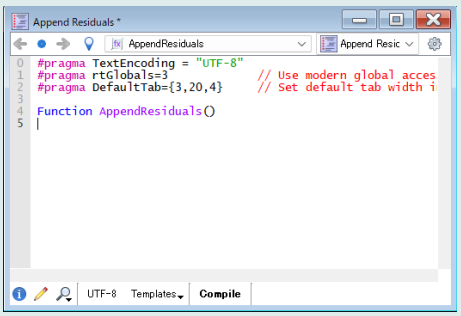
6. 新しいプロシージャウィンドウが表示されます。
常に表示されるビルトインのプロシージャウィンドウを使うこともできますが、この新しいプロシージャウィンドウをスタンドアロンファイルとして保存し、他の Experiment でも使えるようにします。



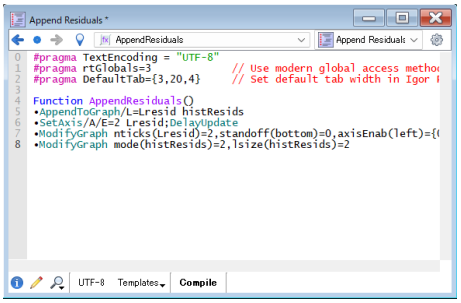
7. **ウィンドウに空白行を追加し、**

```
Function AppendResiduals()
```

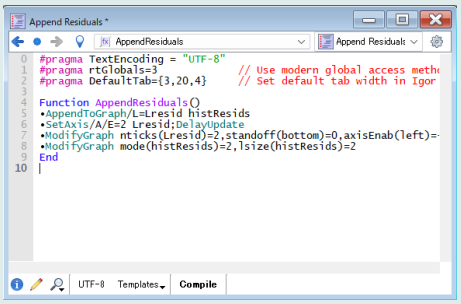
と入力して、Enter キーを押します。



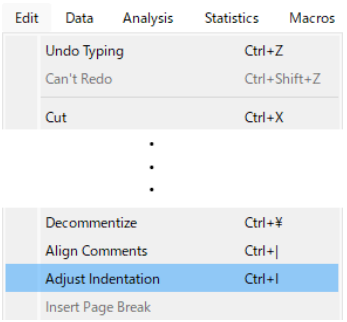
8. **新しいウィンドウへ、クリップボードからコマンドを貼り付けます。**



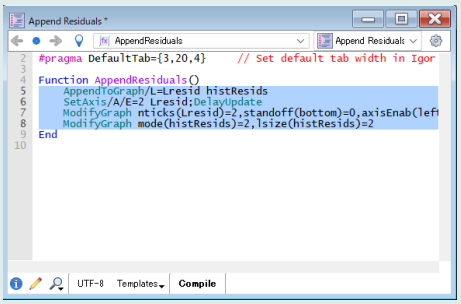
9. **`End` と入力して、Enter キーを押します。**



10. **プロシージャウィンドウに貼り付けた行を選択し、Edit → Adjust Indentation を選択します。**



11. これにより、履歴からコピーされたドットが削除され、プロシージャの通常のインデントを適用するために先頭にタブが追加されます。



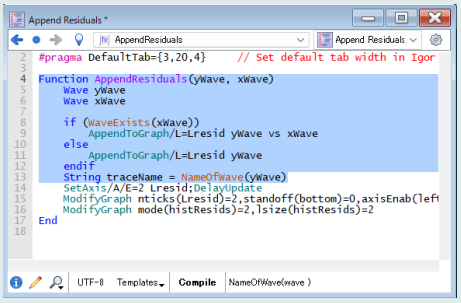
12. **行の終わりに、「;DelayUpdate」がある場合は、これらを削除します。**

DelayUpdate は、プロシージャ内ではなにも影響与えません。
ほぼ機能するプロシージャが出来上がりましたが、大きな制限があります。
それは、残差ウェーブが「histResids」と名付けられている場合のみ動作するというものです。

次のステップでは、等間隔のウェーブフォームデータだけでなく、任意のウェーブや XY ペアでも使えるように機能を変更します。

13. プロシージャの最初の 2 行（SetAxis の行より上）を次の内容に書き換えます。

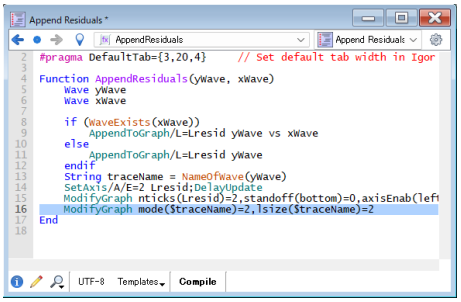
```
Function AppendResiduals(yWave, xWave)
    Wave yWave
    Wave xWave // X wave または x wave がないときは null

    if (WaveExists(xWave))
        AppendToGraph/L=Lresid yWave vs xWave
    else
        AppendToGraph/L=Lresid yWave
    endif
    String traceName = NameOfWave(yWave)
```



14. プロシージャ内の最後の ModifyGraph コマンドで、両方の「histResids」を「$traceName」に変更します。

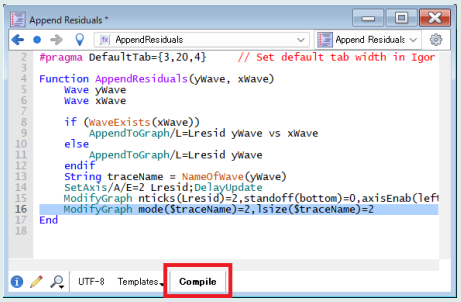
「$」演算子はそれに続く文字列を Igor オブジェクトの名前に変換します。

それでは、動作を試してみます。



15. プロシージャウィンドウの下部にある Compile ボタンをクリックして、コマンドをコンパイルします。

エラーが発生した場合は、上記のリストに一致するようにテキストを編集してください。

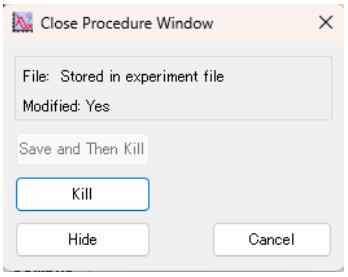


16. 残差のプロシージャウィンドウの閉じるボタン（ウィンドウの右上端の「X」）をクリックします。

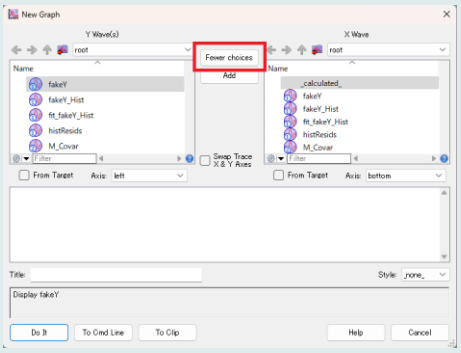
ダイアログが表示され、ウィンドウを終了するか非表示にするか聞いています。

Hide をクリックします。

Shift キーを押しながら閉じるボタンをクリックすると、ダイアログを表示せずにウィンドウが隠されます。

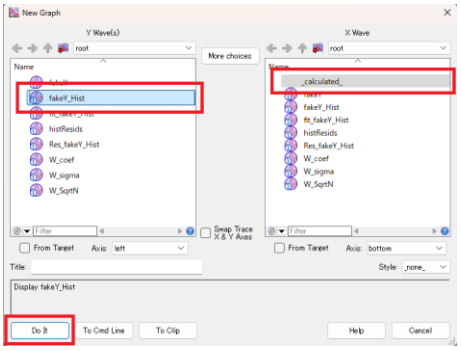


17. メニュー Windows → New Graph を選択して、Fewer Choices ボタンが表示されている場合はそれをクリックします。



18. **Y Wave リストから fakeY_Hist を選択し、X Wave リストから _calculated_ を選択して、Do It ボタンをクリックします。**

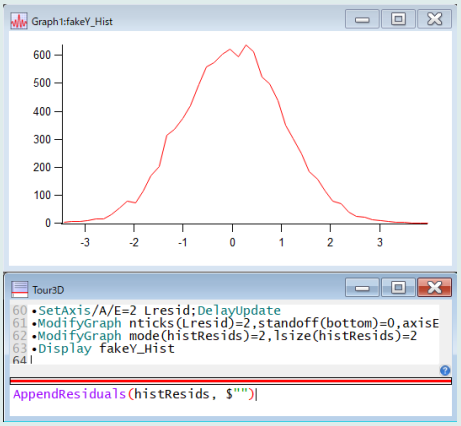
残差のないグラフが作成されます。



19. コマンドウィンドウで次のコマンドを実行します。

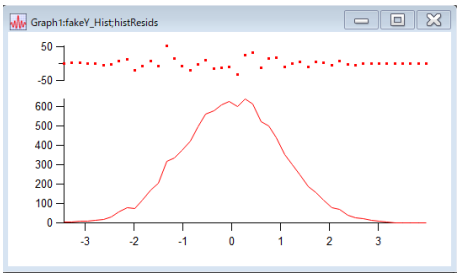
```
AppendResiduals(histResids, $"")
```

X ウェーブがないため、xWave パラメーターの AppendResiduals に NULL を渡すために $"" を使います。

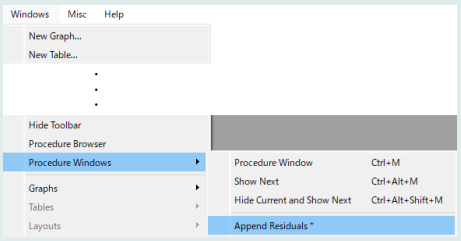


20. AppendResiduals コマンドは、元のヒストグラムデータのグラフ上に残差を表示します。

次に、コマンドラインにウェーブ名を入力しなくてもよいように、ダイアログを表示する機能を追加します。

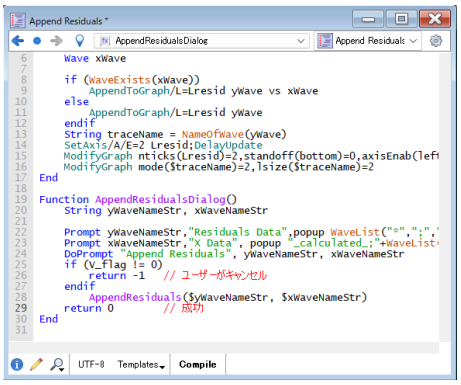


21. **メニュー Windows → Procedure Windows → Append Residuals を選択して、Append Residuals プロシージャウィンドウを表示します。**



22. **Append Residuals ファンクションの定義の下に、次のコマンドを入力します。**

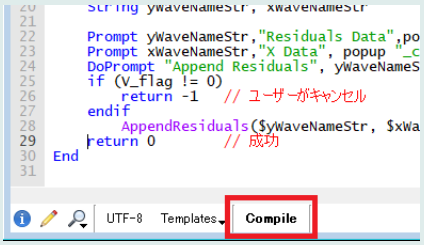
```
Function AppendResidualsDialog()
    String yWaveNameStr, xWaveNameStr

    Prompt yWaveNameStr,"Residuals Data",popup
WaveList("*",";","")
    Prompt xWaveNameStr,"X Data", popup
"_calculated_;"+WaveList("*",";","")
    DoPrompt "Append Residuals", yWaveNameStr,
xWaveNameStr
    if (V_flag != 0)
        return -1   // ユーザーがキャンセル
    endif
    AppendResiduals($yWaveNameStr, $xWaveNameStr)

    return 0        // 成功
End
```



このコマンドは、ユーザーからパラメーターを取得するためのダイアログを表示し、その後 AppendResiduals コマンドを呼び出します。

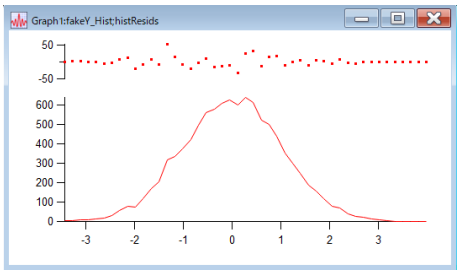
23. プロシージャをコンパイルするには、プロシージャウィンドウの下部にある Compile ボタンをクリックします。

エラーが発生した場合は、上記のリストに一致するようにテキストを編集してください。



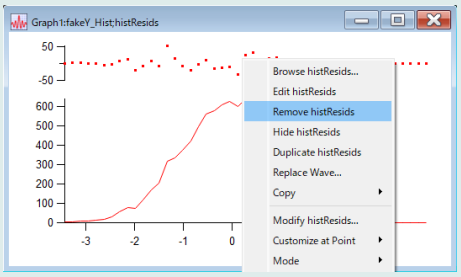
24. Shift キーを押しながらクローズボタンをクリックして、プロシージャウィンドウを非表示にします。

そして、Graph1 ウィンドウをアクティブにします。



25. グラフの上部にある残差トレースを右クリックして、ポップアップメニューから Remove histResids を選択します。

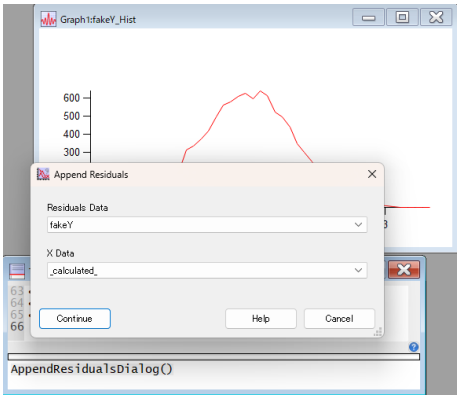
軸に対する最後のウェーブが削除されたため、その軸も削除されます。



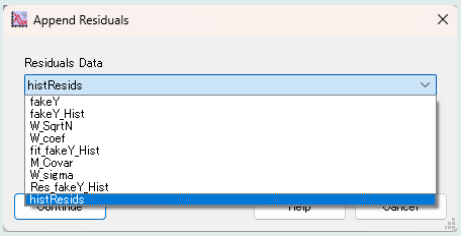
26. コマンドウィンドウで次のコマンドを実行します。

```
AppendResidualsDialog()
```

AppendResidualsDialog コマンドは、パラメーターを選択するためのダイアログを表示します。



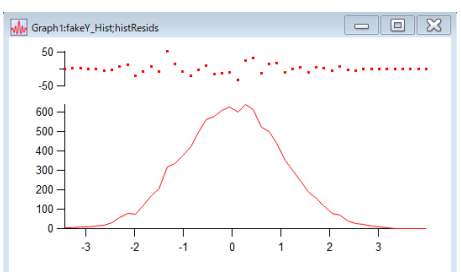
27. Residuals Data ポップアップメニューから histResids を選択します。



28. X Wave ポップアップメニューは、_calculated_ のままにしておきます。

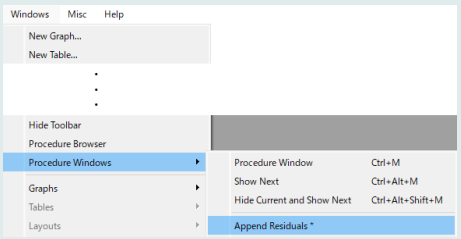
Continue ボタンをクリックします。

グラフには、メインのデータの上に残差のプロットが再び表示されるはずです。



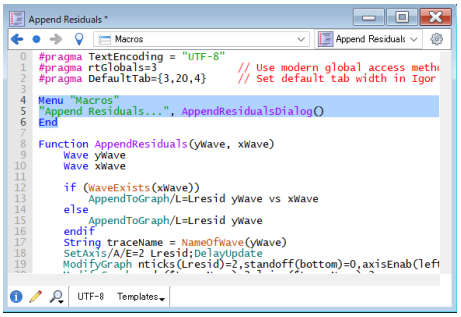
29. 次に、Macros メニューにメニュー項目を追加します。

Windows → Procedure Windows → Append Residuals を選択して、Append Residuals プロシージャウィンドウを表示します。



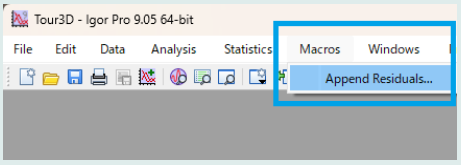
30. **AppendResiduals ファンクションの前に次のコードを入力します。**

```
Menu "Macros"
    "Append Residuals...", AppendResidualsDialog()
End
```



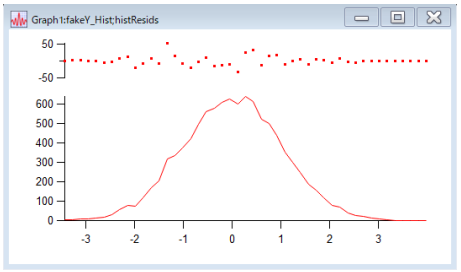
31. **Compile ボタンをクリックします。**

Igor は、プロシージャをコンパイルし、メニュー項目に Macros メニューを追加します。

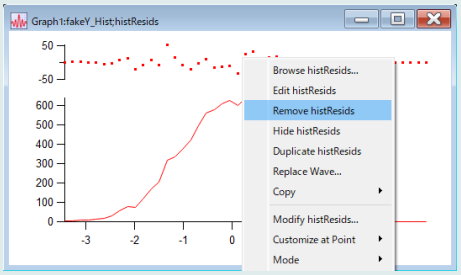


32. **Shift キーを押しながらクローズボタンをクリックして、プロシージャウィンドウを非表示にします。**

そして、Graph1 ウィンドウをアクティブにします。

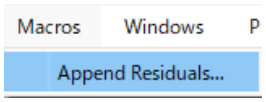


33. **グラフの上部にある残差トレースを右クリックして、ポップアップメニューから Remove histResids を選択して、残差のトレースを削除します。**

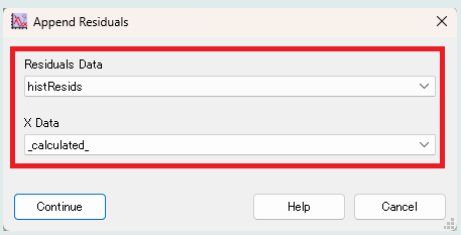


34. **Macros メニューをクリックして、Append Residuals を選択します。**

プロシージャは、パラメーターを選択するためのダイアログを表示します。

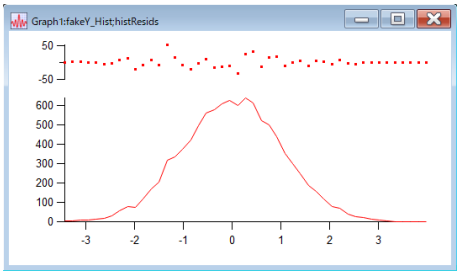


35. **Residuals Data ポップアップメニューから histResids を選択します。**
X Wave ポップアップメニューは _calculated_ のままにしておきます。



36. Continue ボタンをクリックします。

グラフには、メインのデータの上に残差のプロットがこれまでと同じように表示されるはずです。



プロシージャファイルの保存（オプションツアー）

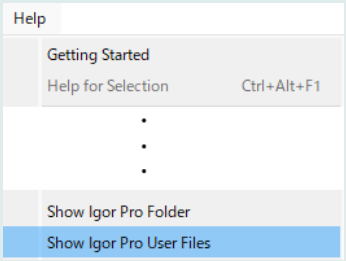
注：Igor Pro のデモ版を 30 日間の試用期間を越えて使っている場合、プロシージャを保存することはできません。

動作するプロシージャが完成したので、今後、使うことができるように保存します。
Igor Pro User Files フォルダーにファイルを保存します。
このフォルダーは Igor によって作られていて、Igor ファイルを保存するために使います。

より正確に言うと、Igor Pro User Files フォルダーの User Procedures サブフォルダーにプロシージャファイルを保存します。

ファイルはハードディスク上のどこに保存しても構いませんが、次のセクションで説明するように、User Procedures サブフォルダーに保存すると、そのファイルにアクセスしやすくなります。

1. メニュー Help → Show Igor Pro User Files を選択します。



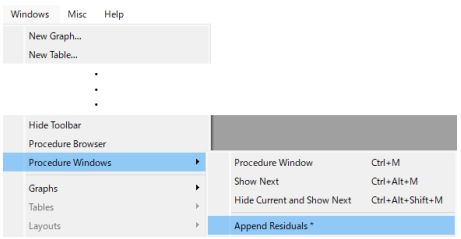
Igor は Igor Pro User Files フォルダーを開きます。
デフォルトでは、このフォルダーは Igor Pro 9 User Files のように、Igor Pro のメジャーバージョン番号が名前に含まれています。
しかし、一般的に Igor Pro User Files フォルダーと呼んでいます。このフォルダーがファイルシステム階層のどこにあるかは、次のステップで必要になるので、メモしておいてください。

デフォルトの場所は：

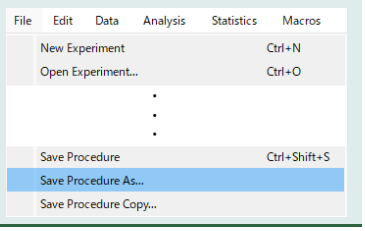
```
C:¥Users¥<user>¥Documents¥WaveMetrics¥Igor Pro 9 User Files
```

Igor Pro User Files フォルダーの User Procedures サブフォルダーに注目してください。
ここにプロシージャファイルを保存します。

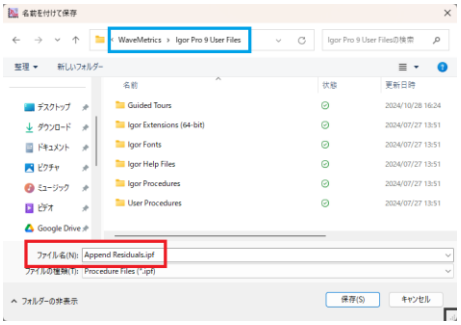
2. Igor に戻り、Windows → Procedure Windows → Append Residuals を選択して、Append Residuals プロシージャウィンドウを開きます。



3. メニュー File → Save Procedure As を選択します。

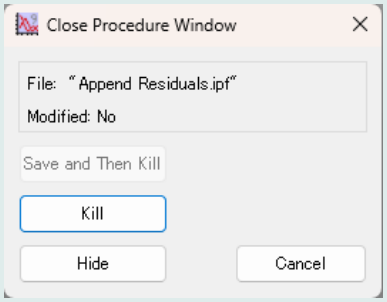


4. ファイル名に「Append Residuals.ipf」と入力します。Igor Pro User Files の User Procedures フォルダーに移動し、「保存」をクリックします。



Append Residuals プロシージャファイルが、スタンドアロンファイルとして保存されます。

5. Append Residuals プロシージャウィンドウのクローズボタンをクリックします。



Igor はファイルを削除するか、隠すかを聞いてきます。

6. Kill ボタンをクリックします。

これにより、現在の Experiment からファイルが削除されますが、ファイルはディスク上に残っているため、必要に応じて開くことができます。

プロシージャファイルを今後使う時には、いくつかの方法があります。

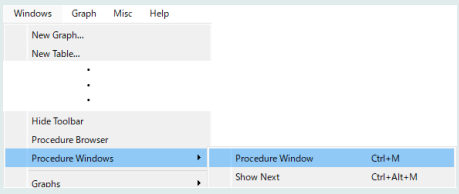
１つはそれをダブルクリックすることです。
もう１つの方法は、File → Open File → Procedure メニューを選択することです。
３つ目は、ビルトインプロシージャウィンドウに `#include` ステートメントを挿入する方法で、次のセクションでこの方法を説明します。

プロシージャファイルのインクルード（オプションツアー）

複数の異なる Experiment で使う予定のプロシージャウィンドウを開くには、`#include` ステートメントを使うのが理想的な方法です。
このセクションでは、その方法を説明します。

注： Igor Pro のデモ版を 30 日間の試用期間を越えて使っている場合、前のセクションで Append Residuals.ipf ファイルを作っていないため、このセクションは実行できません。

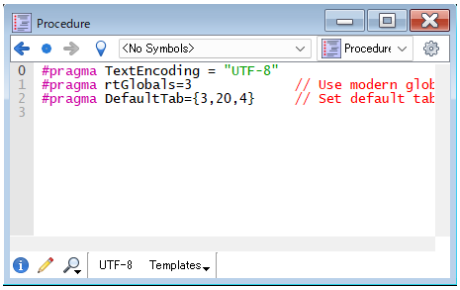
1. Igor では、メニュー Windows → Procedure Windows → Procedure Window を使って、組み込みのプロシージャウィンドウを開きます。



2. ビルトインプロシージャウィンドウの上部付近に、次のような行があることに注目してください。

```
#pragma rtGlobals = 3
```

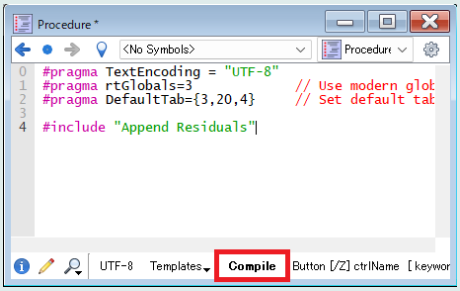
これは専門的で、今は無視できるものです。



3. rtGlobals の行の下に、空白行を挿入し、次のように入力します。

```
#include "Append Residuals"
```

ビルトインプロシージャウィンドウの下部にある、Compile ボタンをクリックします。

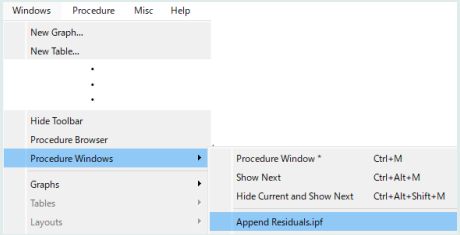


4. Igor はプロシージャウィンドウをコンパイルします。

#include ステートメントを見つけると、Append Residuals.ipf というプロシージャファイルを User Procedures フォルダー内で探し、それを開きます。
非表示で開かれているため、ユーザーには見えません。

（User Procedure フォルダー以外に保存している場合には、そのパスを指定する必要があります）

5. メニュー Windows → Procedure Windows を使って、残差を追加するプロシージャファイルが実際に開いていることを確認します。



Experiment からプロシージャファイルを削除するには、ビルトインプロシージャウィンドウから `#include` ステートメントを削除します。

#include は強力な機能で、プロシージャファイルが他のプロシージャファイルを連続的に含めることを可能にします。
各プロシージャファイルは、必要とする他のプロシージャファイルを自動的に開きます。

Igor が #include ステートメントを満たすためにこのフォルダーを検索するため、User Procedures フォルダーは特別です。

もう 1 つの特別なフォルダーは Igor Procedures です。
Igor Procedures 内のプロシージャファイルは、起動時に Igor によって自動的に開かれ、Igor が終了するまで開いたままになります。
常に開いておきたいプロシージャファイルを置く場所です。
プロシージャファイルをもっとも簡単に使えるようにする方法で、使用頻度の高いファイルに推奨されます。

これでガイドツアー 3 は終了です。