はじめに

Igor Pro は、WaveMetrix 社によるソフトウェアで、高機能なグラフ描画、データ解析、プログラミングを備えている。

グラフの関数フィッティング、微分、積分、フーリエ変換、多変量解析を含む行列計算など多数の数値解析機能がある上、大規模データの取り扱いが迅速にできる。本稿では主にスペクトル解析に Igor Pro を活用し始めた方やこれから活用する方を対象に、基本的な使用方法を紹介する。

体系的で網羅的な使用方法はソフトに付属のマニュアルに詳細な記載があるが、辞書的でかつ大量であるがゆえに初心者には読んで使い方を覚えるには非常に厳しい。辞書的ではなく読み物的な入門マニュアルとすることで、「ソフト付属のマニュアルを使いつつ覚えていく」ことができるようになるまでのつなぎとなれば幸いである。

英語版に基づいて書かれているが、日本語版でも機能は変わらない。なお、発行日時点では Ver. 8 が発売されているが、本稿では Ver. 6.2 に準拠している。調べた限り使用方法に大きな変更点はないので、本稿で紹介されている内容は Ver. 8 でも活きるものとの認識である。

Igor Pro 8 の製品情報

https://www.hulinks.co.jp/software/da_visual/igor

日本語版 ダウンロード版	¥238,680 (税込)
日本語版 教育用 ダウンロード版	¥123,120 (税込)
日本語版 学生用 年間ライセンス ダウンロード版	¥19,440 (税込)
英語版 ダウンロード版	¥183,600 (税込)
英語版 教育用 ダウンロード版	¥88,560 (税込)
英語版 学生用 年間ライセンス ダウンロード版	¥14,040 (税込)

目次

1　基礎的な使い方

本章では、基礎的な使用方法について解説する。

1.1　画面の構成

基本的な画面構成を図 1 に示した。

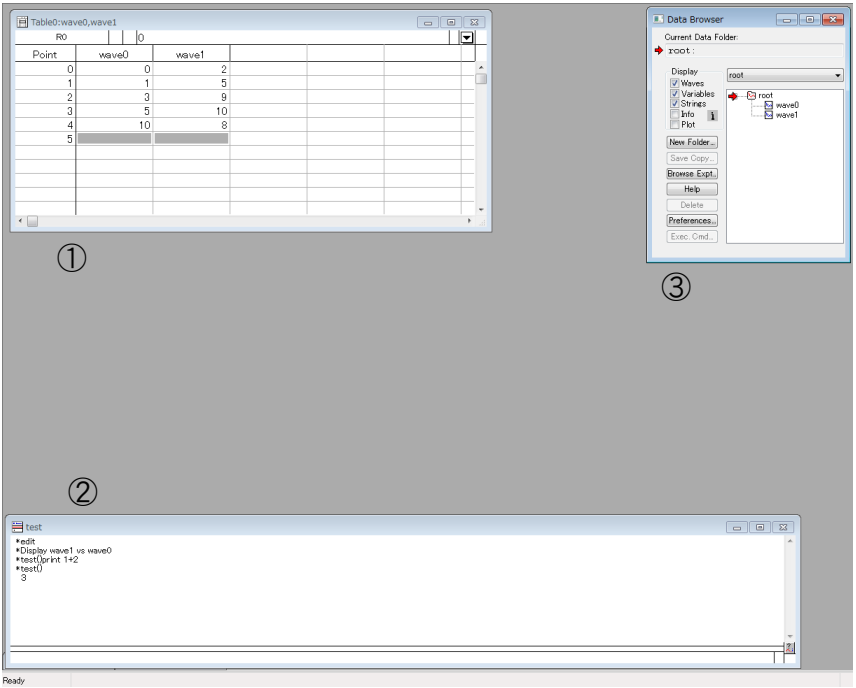


図 1 基本的な画面構成

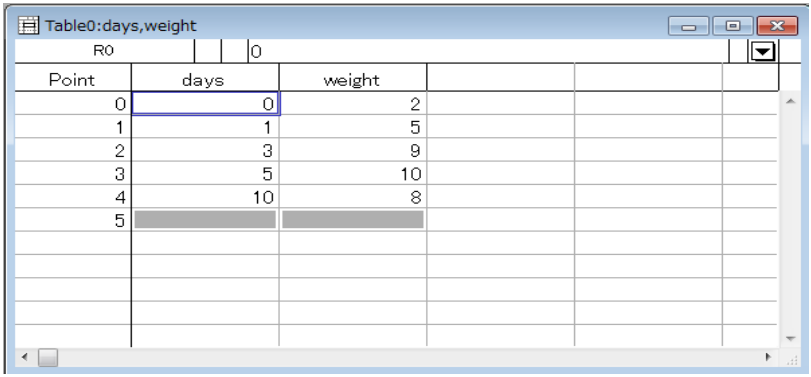
①がデータを入力するための Table、②が操作を行うための Command window、③が格納されたデータの一覧を表示した Data browser である。Data browser はデフォルトでは表示されていないが、メニューバーの Data > Data browser で表示でき、出しておくと便利である。Command window は作業をしているうちに他のウィンドウの下に埋もれがちであるが、「Ctrl + J」で手前に持ってくることができる。

1.2 データ入力とグラフ表示

グラフ表示に必要なデータは数列として扱われており、「Wave」と呼ばれている。Y 軸に対応する Wave と、X 軸に対応する Wave を用意してグラフを表示するというのが基本である。その作業の一例を以下に示す。

時間で経過する重量の変化をプロットしたいとする。

図 2 のように Table に数値を入力していく。何も入力されていない Table[1]に数値を入力していくと、Wave の名前は wave0, wave1, …のように自動で付けられていくが、Table 上で名前を変えたい Wave を右クリックし、「Rename」を選ぶことで名前を変えることができる。



Point	days	weight
0	0	2
1	1	5
2	3	9
3	5	10
4	10	8
5		

図 2 テーブルへの入力例 (経過日数と重量)

メニューバーから Windows > New Graph を選択すると図 3 が表示され、X 軸 (X Wave) として「days」を、Y 軸 (Y Wave) として「weight」を選択する (このとき、X Wave と Y Wave のデータポイント数が一致していない場合は表示できず、エラーとなるので注意)。そこで「Do it」をクリックすると、図 4 のようなグラフが表示される。

[1] 何も入力されていない Table はメニューバーの Windows > New Table で新しく作ることができる。または Command window にて”`edit`”を実行でも OK。

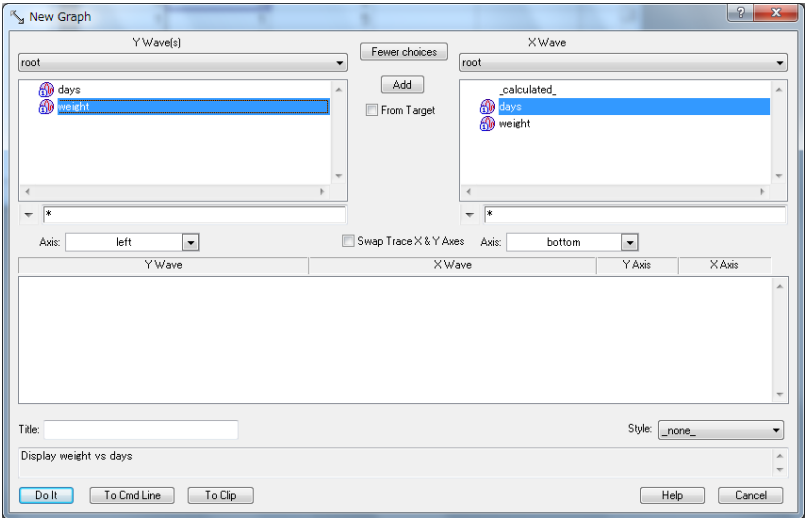


図 3 New Graph 選択後のウィンドウ

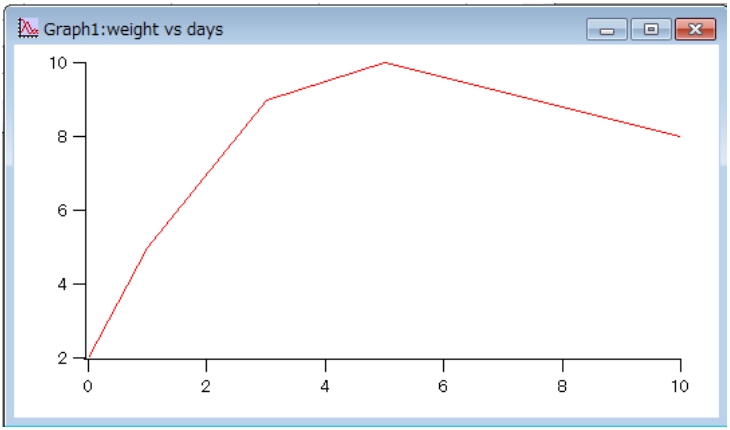


図 4 表示されるグラフ

このグラフの Trace (グラフ中のデータポイントは Trace と表現される) 箇所で右クリック > Modify weight (wave 名)、もしくは Trace のない箇所で右クリック > Modify Trace Appearance で、Trace の表示について編集することができる。

また、軸部分をダブルクリック、もしくは Trace のない箇所で右クリック > Axis Properties で、軸を編集することができる。

編集例を図 5 に示した。詳細な編集手順は割愛するが、様々な表示方法があることを各自確認されたい。

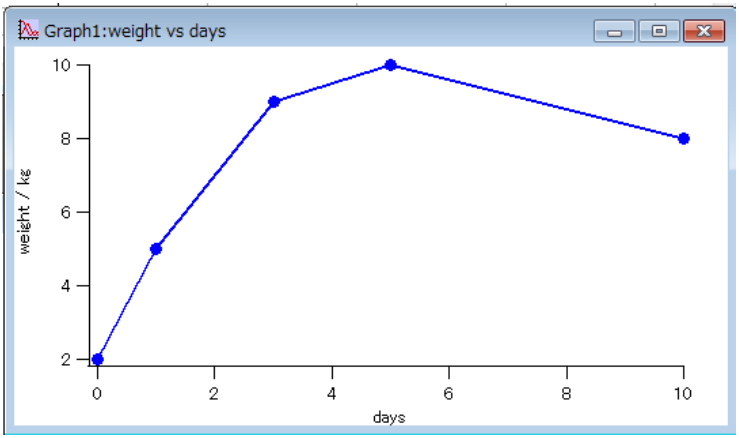


図 5 編集後のグラフ

データ入力では、テーブルに数値を打ち込むことよりも測定データなどがファイルになったものを読み込ませたいことが実際には多いだろう。txt ファイル (タブ区切り、スペース区切りなど) や CSV ファイルなどはドラッグ&ドロップで読み込める上、エクセルファイルからコピー&ペーストでテーブルに貼りつけることもできる。

このようなグラフを、エクセルと比較して非常に高速に、かつ綺麗に表示できることが Igor の大きなメリットである。この例のような 5 点のグラフではその恩恵が得られないが、1,000 点を超えるデータサイズのものを表示するときに効いてくる。

1.3 右の縦軸と左の縦軸を併用する方法

通常は左の縦軸が使用されるが、それとは独立の縦軸を右側で使用したい場合。

例えば図 5 のグラフに右の縦軸を使用したグラフを追加する場合、グラフ中の Trace のない箇所で右クリックして、Append Traces to Graph を選択する。

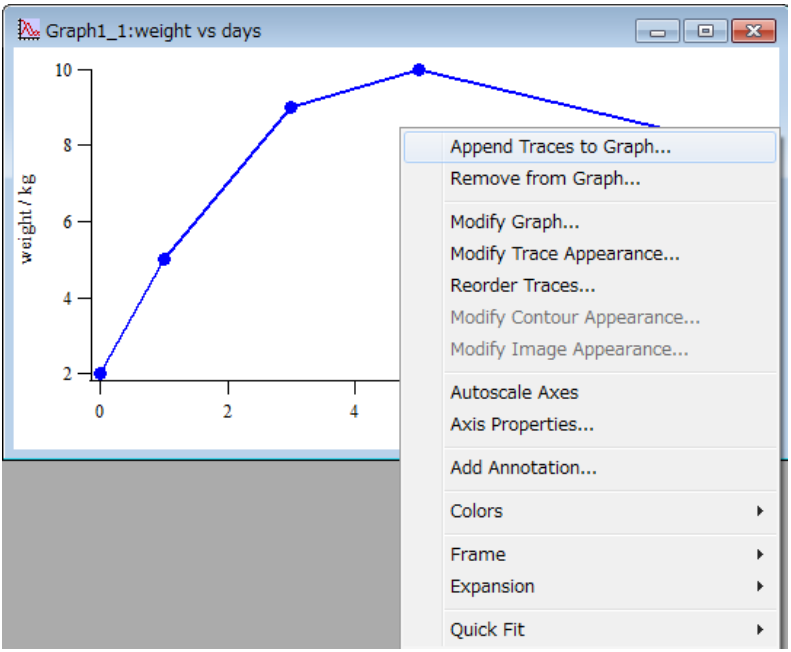


図 6 グラフへのトレースの追加

すると図 7 のような Append Traces 画面が表示されるので、追加したい YWave と対応する XWave を選択して、Axis のプルダウンから”right”を選択する。

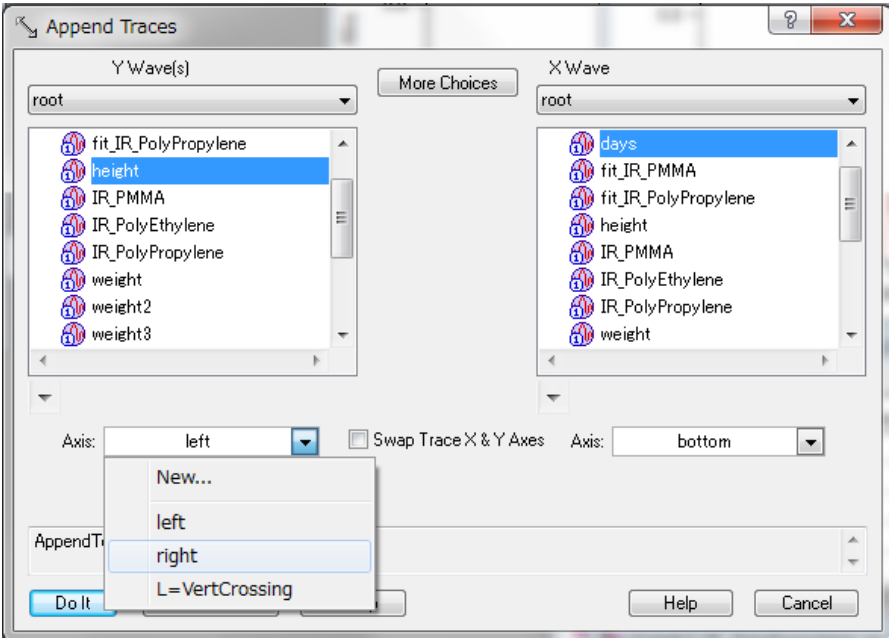


図 7 Append Traces 画面

図 8 のように新たな Trace と、それに対応した右側の縦軸が現れる。

このように右側の縦軸の併用は、Trace を追加するときにしかできないので注意が必要である。これと同じ要領で、上側の横軸を併用することもできる。

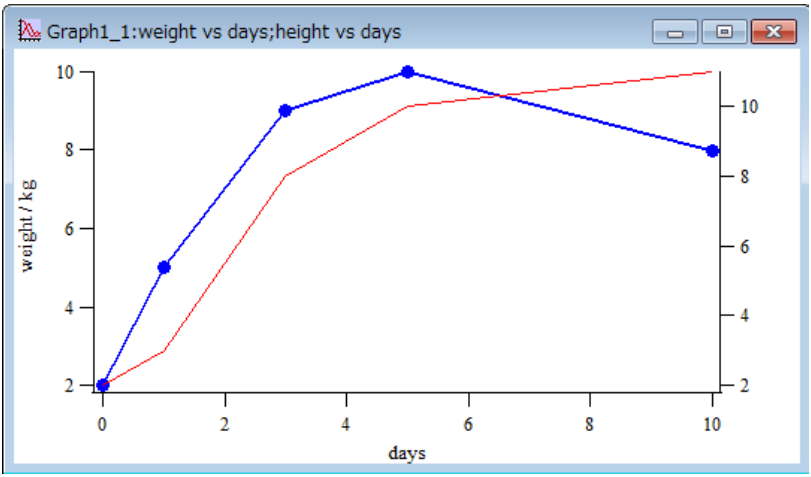


図 8 Trace 追加後のグラフ

1.4 Command window の活用

1.2 で紹介したような操作は、基本的に Command window に文字列を入力することで再現できる。

例えば、図 4 を表示するには、

```
Display weight vs days
```

と入力してエンターを押せばよい。`Display` が命令を下すコマンドで、その後に wave 名 (Y 軸) vs wave 名 (X 軸) の順番で記述されている。各単語の間は半角スペースで区切る必要がある。

入力において、コマンドも wave 名も、大文字と小文字を区別しないことに注意が必要である。`Display` を `DISPLAY` や `display` と書いても同じであるし、wave 名は作った時点で `weight` であったとしても、`Weight` や `WEIGHT` と書いても同じ wave を指しているとみなされる。このことからわかるとおり、`weight` という wave と `WEIGHT` という wave を別のものとして作ることはできない。

さらに、1.2 で紹介したような操作でグラフを表示したときも、Command window には上

記のコマンドが表示されている。これを活用して、メニューバーを使って操作したときにどのようなコマンドが使われているかを簡単に知ることができるので、マクロを組む上で参考になる。

1.5 Command Help の使い方

コマンドの名前から、その使い方を調べることができる。メニューバーの Help > Command Help を選択すると、図 9 の画面が表示される。

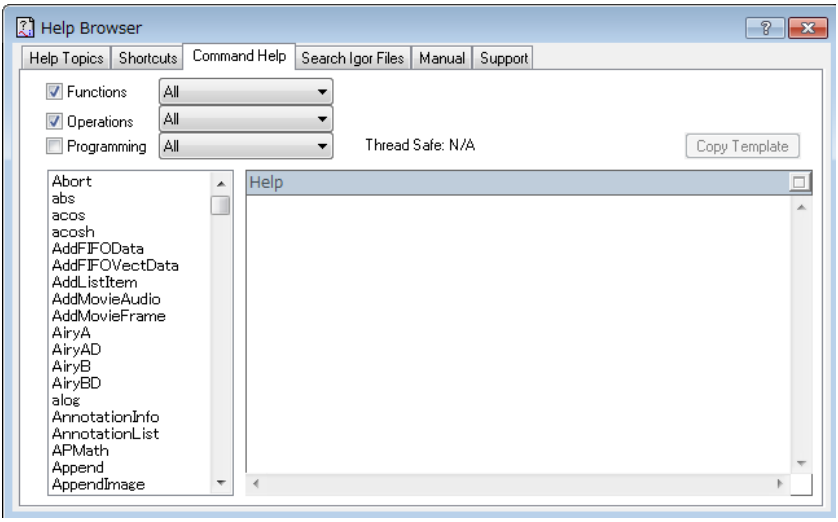


図 9 Command Help 画面

例えば”Rename”を調べると図 10 が表示される。

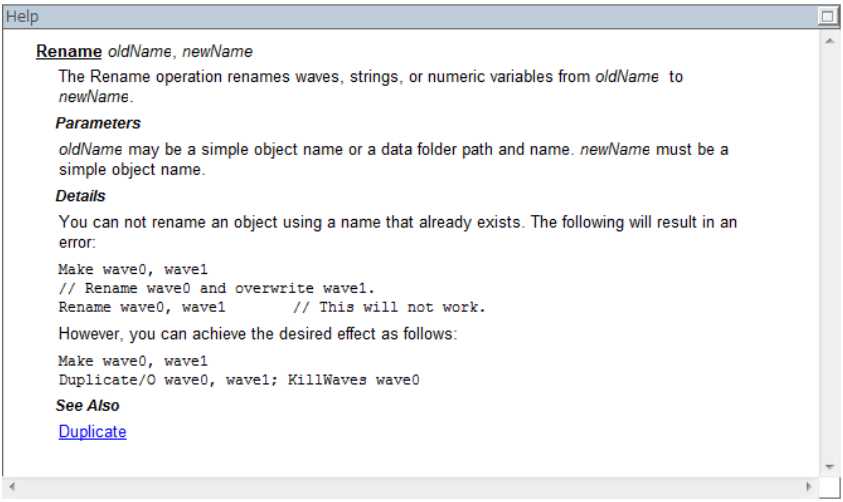


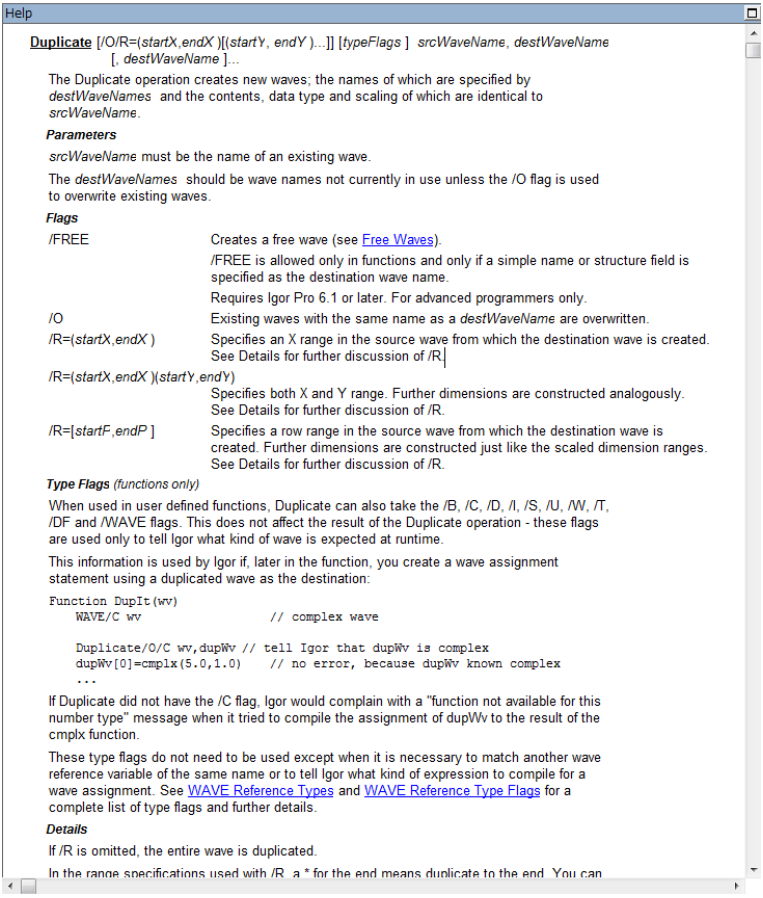
図 10 Rename コマンドのヘルプ

Wave 名、文字列名、変数名を *oldName* から *newName* に変更する、とある。1.2 で示したような wave 名の変更も、このコマンドで実行できる。例えば wave1 の名前を weight に変えるときは、

```
Rename wave1, weight
```

と入力して実行すればよい。

もう一つ、例として”Duplicate”のコマンドを紹介する。ヘルプ画面を図 11 に示した。


Help

Duplicate [/O/R=(*startX*,*endX*)[(*startY*, *endY*)...]] [*typeFlags*] *srcWaveName*, *destWaveName* [, *destWaveName*]...

The Duplicate operation creates new waves; the names of which are specified by *destWaveNames* and the contents, data type and scaling of which are identical to *srcWaveName*.

Parameters

srcWaveName must be the name of an existing wave.

The *destWaveNames* should be wave names not currently in use unless the /O flag is used to overwrite existing waves.

Flags

/FREE	Creates a free wave (see Free Waves). /FREE is allowed only in functions and only if a simple name or structure field is specified as the destination wave name. Requires Igor Pro 6.1 or later. For advanced programmers only.
/O	Existing waves with the same name as a *destWaveName* are overwritten.
/R=(*startX*,*endX*)	Specifies an X range in the source wave from which the destination wave is created. See Details for further discussion of /R.
/R=(*startX*,*endX*)(*startY*,*endY*)	Specifies both X and Y range. Further dimensions are constructed analogously. See Details for further discussion of /R.
/R=[*startP*,*endP*]	Specifies a row range in the source wave from which the destination wave is created. Further dimensions are constructed just like the scaled dimension ranges. See Details for further discussion of /R.

Type Flags *(functions only)*

When used in user defined functions, Duplicate can also take the /B, /C, /D, /I, /S, /U, /W, /T, /DF and /WAVE flags. This does not affect the result of the Duplicate operation - these flags are used only to tell Igor what kind of wave is expected at runtime.

This information is used by Igor if, later in the function, you create a wave assignment statement using a duplicated wave as the destination:

```
Function DupIt(wv)
    WAVE/C wv                   // complex wave

    Duplicate/O/C wv,dupWv // tell Igor that dupWv is complex
    dupWv[0]=cmplx(5.0,1.0)     // no error, because dupWv known complex
    ...
```

If Duplicate did not have the /C flag, Igor would complain with a "function not available for this number type" message when it tried to compile the assignment of dupWv to the result of the cmplx function.

These type flags do not need to be used except when it is necessary to match another wave reference variable of the same name or to tell Igor what kind of expression to compile for a wave assignment. See WAVE Reference Types and WAVE Reference Type Flags for a complete list of type flags and further details.

Details

If /R is omitted, the entire wave is duplicated.

In the range specifications used with /R, a * for the end means duplicate to the end. You can

図 11 Duplicate コマンドのヘルプ

Duplicate は、ある wave をコピーして新しい wave を作るコマンドであり、Rename と並んで非常によく使う。図で示した範囲よりもまだ下に説明が続いている。

Flags という項目があることに注目されたい。コマンドの直後にフラッグを付けることでコマンドの性質をアレンジすることができる。ここでは”/O”のフラッグが重要である。/O を付けておくと、コピーしたあとの wave 名が既に使われているとき、上書き処理してくれる。逆に付けないと、「この wave 名は既に使われているから実行不可」というエラーが出る。

つまり、

```
Duplicate/O wave1, weight
```

と入力･実行すると、wave1 がコピーされて weight という wave が作成される。このとき、weight というwave が既に存在していたとしても、/O のフラッグを付けているため、上書きしてくれる。

1.6 関数によるグラフのフィッティング

回帰分析は Igor にてよく用いられる解析機能の一つである。直線をはじめとする多項式、ガウス関数、ローレンツ関数、指数関数、対数関数、sin 関数などがデフォルトで用意されており、簡単に行える。

例として、IR 吸収スペクトルのバンドフィッティングの例を紹介する。

図 12 に、ポリメチルメタクリレート (PMMA) の IR 吸収スペクトルを示した。Y Wave 名は" IR_PMMA"で、X Wave 名は"wn_PMMA"である。1730 cm^{-1} あたりにあるバンドのトップ位置や幅を知るために、ガウス関数によるフィッティングを行う。

フィッティング領域をカーソルで指定する。カーソルはグラフウィンドウをアクティブにして「Ctrl + i」で出すことができる (もしくは Command window にて"ShowInfo"を実行)。丸カーソルと四角カーソル[2]で挟むことで、領域を指定する。

[2] 丸カーソルは"pcsr(A)"、四角カーソルは"pcsr(B)"という名前が付いている。マクロを組む上で使うこともできる。

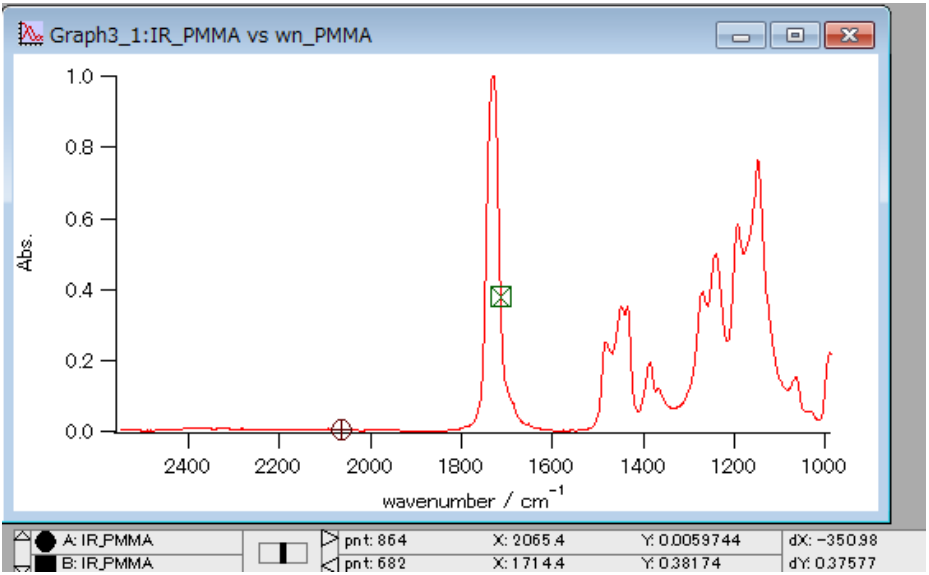


図 12 フィッティング箇所の指定

次に、メニューバーから Analysis > Curve Fitting を選択する。Function and Data タブ (図 13) の Function から関数 (ここでは gauss) を選択する。Y_Data と X_Data をプルダウンから指定する。フィッティング対象のウィンドウがアクティブになっていれば[3]、From Target にチェックを入れればウィンドウ上に描かれている wave から自動で選択してくれる。

[3] ウィンドウが最後にクリックされ、最も手前に配置されているとき、ウィンドウがアクティブであるという。アクティブなウィンドウに適用されるコマンドも多々あるので注意。

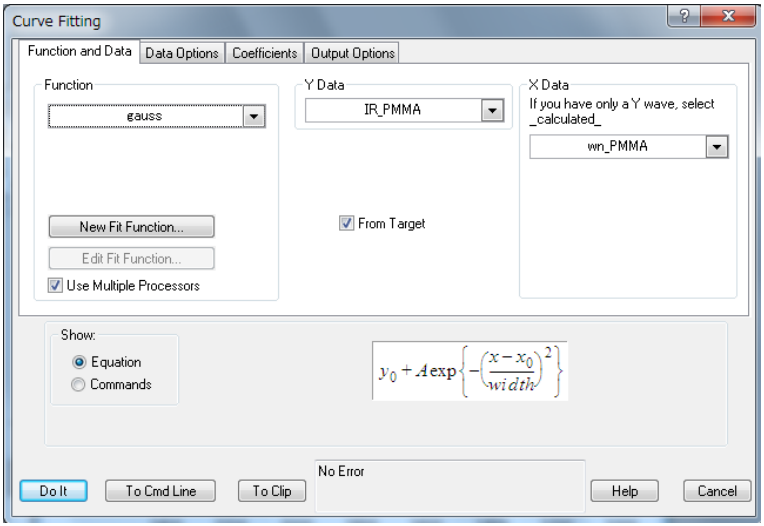


図 13 Curve Fitting の Function and Data タブ

次に、Data Options タブ (図 14) にて Range (フィッティングを適用する範囲) を指定する。今回のようにフィッティング範囲をカーソルで挟んで指定する場合は、”Cursors” ボタンをクリックすれば自動で指定される。

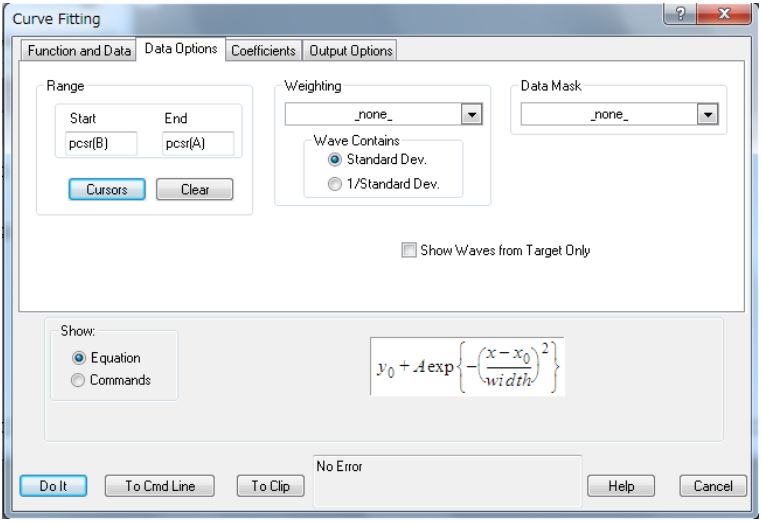


図 14 Curve Fitting の Data Options タブ

ここで”Do it” ボタンをクリックするとフィッティングが実行され、図 15 のように表示される。ただし、図 15 では見やすいようにフィッティング関数を青線に変更してある。

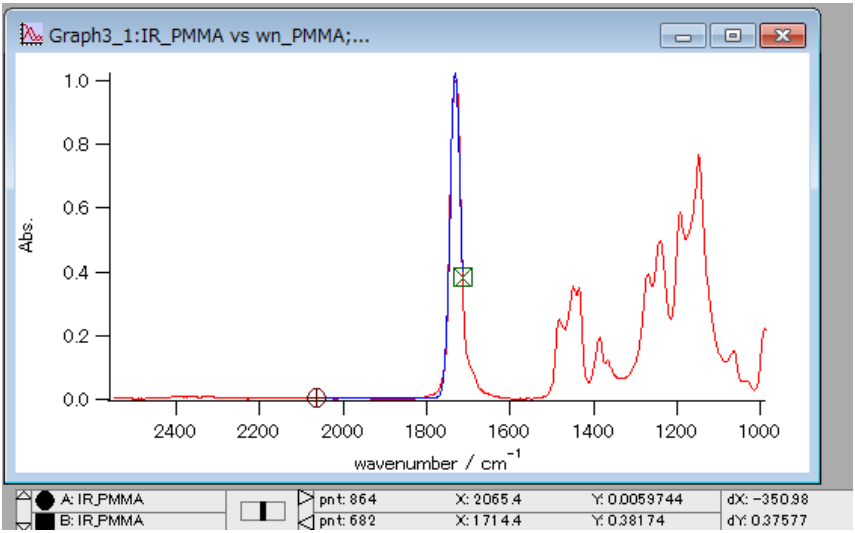


図 15 フィッティングされた関数

フィッティング実行後のコマンドラインの履歴部分は図 16 のように表示され、フィッティング結果がわかる。

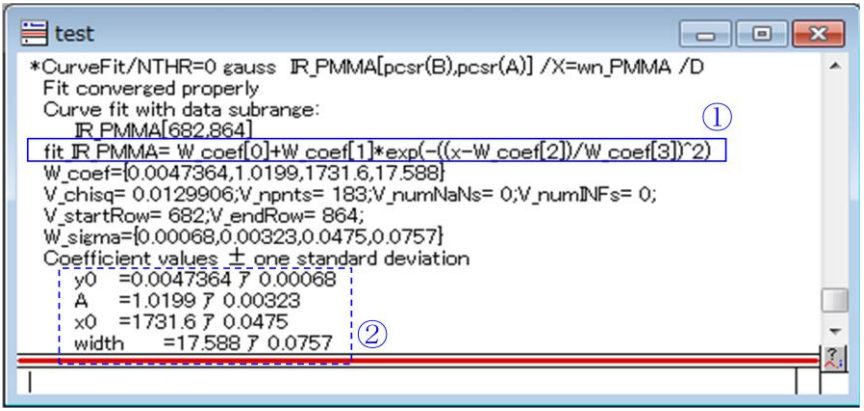


図 16 フィッティング実行後のコマンドライン

①の実線部は、フィッティング関数の係数とX軸の関係を示している。係数は”W_coef”という名前の wave に格納される形で扱われる[4]。そして②の破線部は、フィッティング結果によって導かれた係数を示している。上から順に、y0 → W_coef[0]、A → W_coef[1]、x0 → W_coef[2]、width → W_coef[3]と対応している。例では、ガウス関数の中央位置が 1731.6 cm^{-1}、幅が 17.588 cm^{-1} であることがわかる。

W_coef に限らず、wave 内の n 番目の数字は”wave 名[n]”で表される[5]。これはマクロを組む際にも多用するため重要である。

[4] W_coef は、フィッティングすると自動で作られる。フィッティングするたびに W_coef の中身は上書きされていってしまうことに注意。

[5] Igor では”0 番目”がスタートであることに注意。

1.7 グラフの自動での色変更

意外に知られていない機能であるが、複数本の wave がグラフ上に表示されているとき、それぞれの色をまとめて変更することができる。

グラフをアクティブした状態で、メニューバーから Graph > Packages > Make Traces Different を選択すると、図 17 のようなウィンドウが表示される。

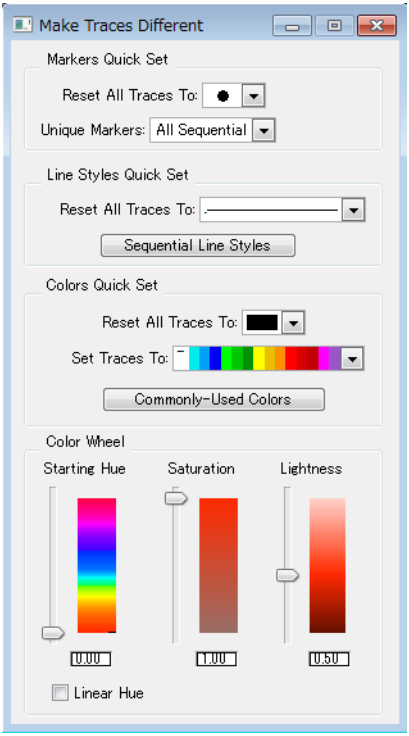


図 17 Make Traces Different のウィンドウ

その後、Set Traces To のプルダウンを選択すると図 18 が表示されるため、色の組み合わせを選択することができる。

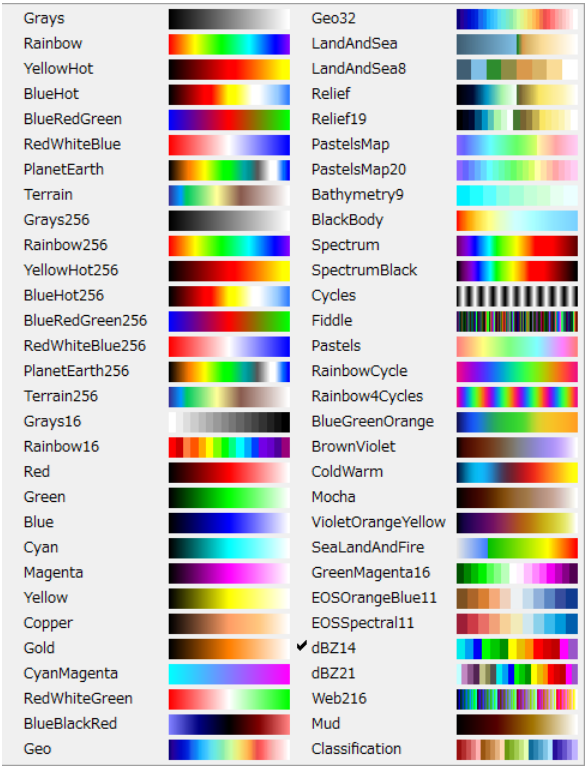


図 18 Set Traces To のプルダウン選択時

1.8 ショートカットキー

本章の最後に、使用頻度の高いショートカットキーを紹介する (ここまでで紹介してきたものも含む)。

“Ctrl + a”→グラフをアクティブにした状態では、Autoscale する。テキストなどのウィンドウでは、全選択する。

“Ctrl + d”→アクティブなグラフを複製する。

“Ctrl + i”→アクティブなグラフの下部にカーソルを表示する。

“Ctrl + t”→アクティブなグラフに図形などを追加できるようになる。

2　マクロの作り方

マクロを用いることで、コマンドラインに直接打ち込んでできることに加えて変数･関数を使用した数値処理や、グラフ描画、グラフレイアウト変更などを自動で行うことができる。コマンドは C 言語に近い形で構築されている。初心者にはハードルが高く感じられるかもしれないが、この機能が Igor の肝と呼べるもので、少し使ってみるだけでも十分にその便利さを実感できるはずなので触れてみてほしい。

2.1　マクロの入力画面

新しくマクロの入力画面 (Procedure ウィンドウ) を作製するには、Windows > New > Procedure を選択する。すると、図 19 のようなウィンドウが表示される。

一番上の行に

```
#pragma rtGlobals=3        //Use modern global method and strict wave access.
```

とデフォルトで入力されているが、これはコンパイラへのメッセージである。変更する必要のない部分であり、消さずにこの部分の下の行から使っていけばよい。

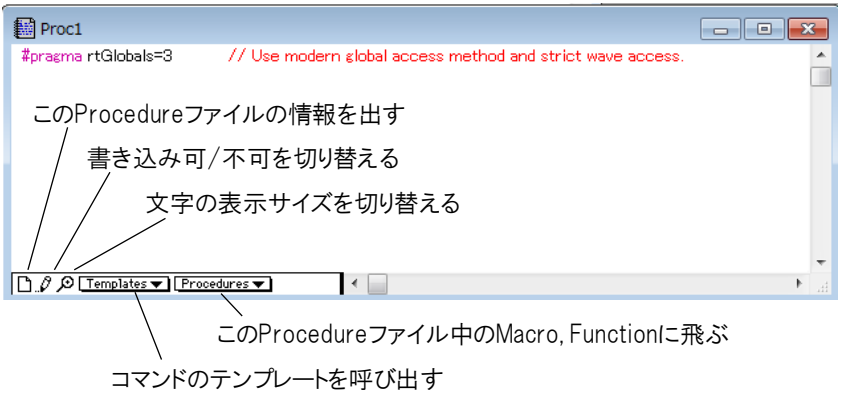


図 19 Procedure ウィンドウと操作ボタンの説明

マクロの開始と終了の宣言は、Functionと End を用いる。Functionと End で挟まれた部分の内容が実行される。

Function/End の代わりに Macro/End を用いることもできる。Macro では上から順に一行ずつ逐次的に処理していくのに対して、Function はコンパイルすることで中身全体を変換してから実行するため、Function の方が処理速度が速いというメリットがある。Function を用いることで困ったことは特に生じないので本稿では全て Function で紹介するが、出来上がったものについては「マクロ」と表記している。

簡単なマクロを書いてみよう。

```
Function TestPlus()
Variable Ans
Ans=3+5
Print Ans
End
```

以上のマクロは足し算を実行するものである。上の通りに入力し、Compile を押してエラーが出なければ正常にマクロが作られたことになる。

Function の後ろの” TestPlus”はマクロの名前であり、その後ろの”()”には、マクロの実行に必要なパラメータ「引数」を入力する。ここでは引数を必要としないものを作ったので()の中身には何も入れていない。

“Variable Ans”は「Ans という名前の変数を使用する」という宣言であり、ここでは計算結果の入れ物として Ans を用いる。”Ans=3+5”で 3+5 の計算結果を Ans に代入し、Print Ans で Ans に入れられた数をコマンドラインに表示する。

コマンドラインに” Function TestPlus()”と入力しエンターを押すと、次の行に”8”と表示される。

上の方法では足し算の中身をマクロの中に入れ込んでしまっているが、コマンドライン上で足し算に使う数字を指定するようにするには、以下のように作成する。

```
Function TestPlus2(v1, v2)
Variable v1, v2
Variable Ans
Ans=v1+v2
Print Ans
End
```

今回作ったマクロ「TestPlus2」では、2 つの引数 v1, v2 を使用して足し算を実行する。Ans の他にも、v1 と v2 も変数として宣言している点に注目されたい。

このマクロを実行するには、TestPlus2(3, 5)とコマンドラインに入力してエンターを押せばよい。すると次の行に 8 が表示される。実行後のコマンドラインを図 20 に示した（破線枠部分がマクロ実行によるもの)。もちろん、(3, 5)の部分を変更すれば任意の数での足し算が実行される。

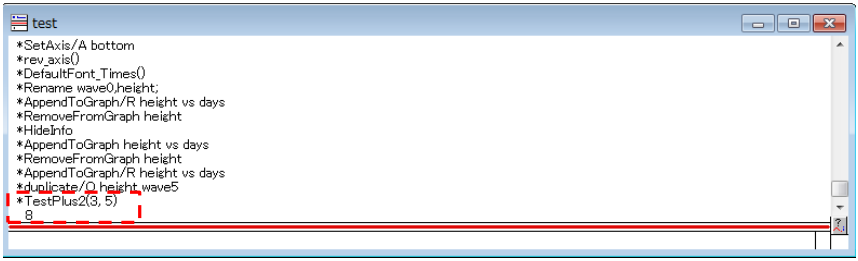


図 20 TestPlus2 のマクロ実行後のコマンドライン

2.2 手軽なマクロの作り方

実行したほとんどの作業内容はコマンドラインに表示されるため、そのコマンド内容をProcedureに貼り付けてしまってFunction/Endで挟んでしまうというのが、ほとんど知識の要らない、それでいて便利な作り方である。

例えば、横軸を反転させる作業を実行すると、コマンドラインに次のように表示される(図 21の破線枠も参照)。

```
SetAxis/A/R bottom
```

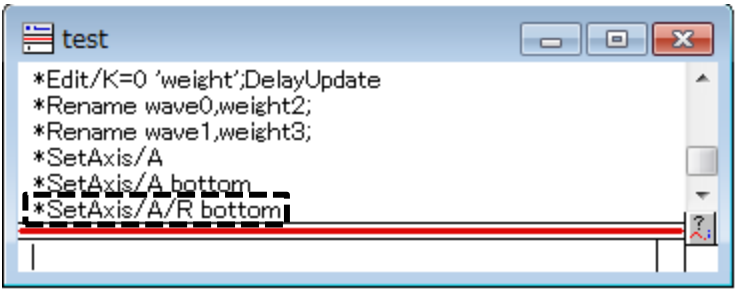


図 21 横軸を反転させた直後のコマンドライン

これを下のようにマクロにしてやる。

```
Function Rev_Axis()
SetAxis/A/R bottom
End
```

そして、横軸を反転させたいグラフをアクティブにしてからコマンドラインでRev_Axis()を実行するだけで、そのグラフの横軸を反転させることができる。

2.3 マクロをメニューから実行する

マクロをコマンドラインから実行せずとも、メニューバーから選択して実行できるようにす

ることができる。例えば 2.2 で作ったマクロに、以下のように Menu/End コマンドを用いる。

```
Function Rev_Axis()
SetAxis/A/R bottom
End

Menu "Macros"
"横軸を反転させる", Rev_Axis()
End
```

すると、メニューバーの”Macros”に「横軸を反転させる」が現れる (図 22) ので、グラフをアクティブにしてから実行することで横軸を反転させることができる。クリックだけでマクロが実行できるので、よく使う操作はこのようにしておくとラクができる。

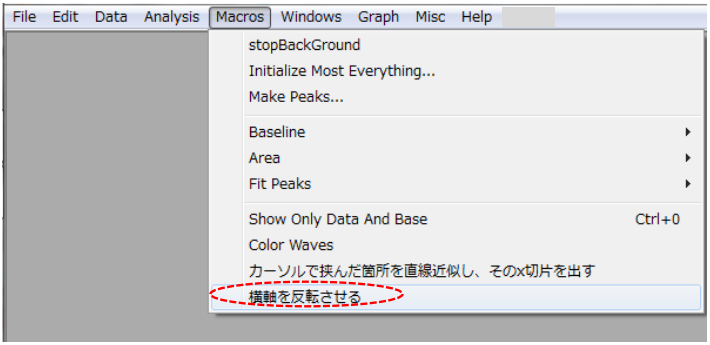


図 22 メニューバーにマクロを追加した例

オススメの使い方は、以下のようにデフォルトフォントを変更することでグラフのすべてのフォントを一瞬で統一するというものである。

```
Function DefaultFont_Times()
DefaultFont/U "Times New Roman"
End

Menu "Macros"
"デフォルトのフォントを Times New Roman にする", DefaultFont_Times()
End
```

2.4 関数の定義

1.6 で示した手順にて、デフォルトで用意された関数でグラフをフィッティングすることができるが、Procedure において任意の関数を定義して、フィッティング関数として用いることができる。

1 次関数をベースラインとしたガウス関数の定義を下に示す。

```
function gau(W,X)        //名前を付ける
wave W
variable X
variable ans
                //CurveFitDialog/ These comments were created by the Curve Fitting dialog. Alteri
                //CurveFitDialog/ make the function less convenient to work with in the Curve Fit
                //CurveFitDialog/ Equation:
                //CurveFitDialog/ variable ans
                //CurveFitDialog/ ans=a+b*X+ height*exp (-ln(2)*(X-X0) ^2/ width^2)
```

```
                    //CurveFitDialog/ f(X) = ans
                    //CurveFitDialog/ End of Equation
                    //CurveFitDialog/ Independent Variables 1
                    //CurveFitDialog/ X
                    //CurveFitDialog/ Coefficients 5 //係数の数
                    //CurveFitDialog/ W[0]=a  //この辺りは係数に名前を付
```

けているだけなので、自分がわかりやすいようにつけると良い。

```
                    //CurveFitDialog/ W[1]=b
                    //CurveFitDialog/ W[2]=height
                    //CurveFitDialog/ W[3]=X0
                    //CurveFitDialog/ W[4]=width
  ans=W[0]+W[1]*X+W[2]*exp(-ln(2)*(X-W[3])^2/W[4]^2)        //関数
```

の形を書く

```
  return ans
  end
```

”//”に続く文字列はコメント扱いでありマクロには影響を与えないが、”//CurveFitDialog/”についてはその限りではない。何故このような仕様なのかはわからない。初心者には複雑なことをしているように見えるかもしれないが、上で示したものをそのまま使いまわして必要箇所をアレンジ (下線で示した箇所だけ編集すれば良い)して、様々な関数を作ると良い。ガウス関数においては、ベースラインが定数であるものはデフォルトで用意されているが、傾きを持った１次関数がベースラインのものは用意されていないため、このような関数を作っておくと有用である。

ガウス関数を示したついでに、ローレンツ関数も下に示しておく。

```
  function single_lor(W,X)
  wave W
  variable X
  variable ans
                //CurveFitDialog/ These comments were created by
the Curve Fitting dialog. Alteri
                //CurveFitDialog/ make the function less convenient
to work with in the Curve Fit
                //CurveFitDialog/ Equation:
                //CurveFitDialog/ variable ans
                //CurveFitDialog/  ans=a+b*X  +  (height*width^2)
/((X-X0)^2 +width^2)
                //CurveFitDialog/ f(X) = ans
                //CurveFitDialog/ End of Equation
                //CurveFitDialog/ Independent Variables 1
                //CurveFitDialog/ X
                //CurveFitDialog/ Coefficients 5
                //CurveFitDialog/ W[0]=a
                //CurveFitDialog/ W[1]=b
                //CurveFitDialog/ W[2]=heightA
                //CurveFitDialog/ W[3]=widthA
                //CurveFitDialog/ W[4]=X0A
  ans = W[0] + W[1]*X + (W[2]*W[3]^2)/((X-W[4])^2+W[3]^2)
  return ans
  end
```

2 つのローレンツ関数の裾が重なり合っている曲線についても、下のようにローレンツ関数の和を作っておくことでフィッティングすることができる。

```
function double_lor(W,X)
wave W
variable X
variable ans
                //CurveFitDialog/ These comments were created by
the Curve Fitting dialog. Alteri
                //CurveFitDialog/ make the function less convenient
to work with in the Curve Fit
                //CurveFitDialog/ Equation:
                //CurveFitDialog/ variable ans
                //CurveFitDialog/ ans=a+b*X + (heightA*widthA^2)/
 ((X-X0A)^2+widthA^2) + (heightB*widthB^2)/((X-X0B)^2+widthB^2)
                //CurveFitDialog/ f(X) = ans
                //CurveFitDialog/ End of Equation
                //CurveFitDialog/ Independent Variables 1
                //CurveFitDialog/ X
                //CurveFitDialog/ Coefficients 8
                //CurveFitDialog/ W[0]=a
                //CurveFitDialog/ W[1]=b
                //CurveFitDialog/ W[2]=heightA
                //CurveFitDialog/ W[3]=widthA
                //CurveFitDialog/ W[4]=X0A
                //CurveFitDialog/ W[5]=heightB
                //CurveFitDialog/ W[6]=widthB
                //CurveFitDialog/ W[7]=X0B
```

```
  ans  =  W[0]  +  W[1]*X  +  (W[2]*W[3]^2)/((X-W[4])^2+W[3]^2)  +
(W[5]*W[6]^2)/ ((X-W[7])^2+ W[6]^2)
  return ans
  end
```

2.5 定義した関数をフィッティングで使う方法

2.4 で作成した関数はデフォルトのままではフィッティングで使用することができない。フィッティングの手順は 1.6 で示したとおりだが、そのとき Curve Fitting ウィンドウの Function and Data のタブにて、図 23 に示したように Function のプルダウンから”Show Old-Style Functions”にチェックを入れる。

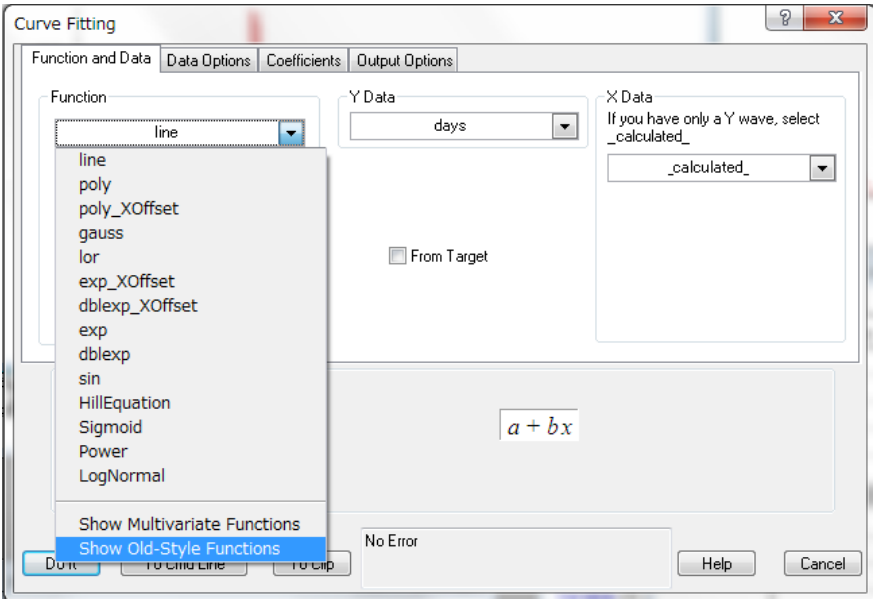


図 23 定義した関数をフィッティング関数候補に追加

チェックを入れると、Function のプルダウンに作った関数が表示されるようになる。

あとがき

最も基本的な使い方から、Procedure ウィンドウの基本的な使い方までを紹介した。もちろんほんの導入部なので、次巻を執筆する機会があればより高度な使い方を示していきたい。

マニュアル作成が初めてであることもあり、至らない点も多々あったことと思われる。本稿に関するご意見、ご質問、ご要望などについてはお気軽に分光解析研究会のメールアドレス bunkaiken@yahoo.co.jp か、twitter アカウント: @bunkaiken へのメッセージにてお聞かせ願いたい。

筆者は大学の学部 4 年にて分光分野の研究室に配属され Igor Pro を使い始め、今もメーカー勤務で使い続けている。研究室の先輩方は非常に Igor Pro に詳しい方が多かったので何でも訊ける環境にあり、恵まれていた。諸先輩方の知識には及ばないが、本稿が理解への一助となれば幸いである。

タイトル **Igor Pro 入門**

発行日 2018 年 8 月 12 日発行
編集/発行者 分光解析研究会
連絡先 bunkaiken@yahoo.co.jp
印刷/製本 有限会社 ねこのしっぽ