

Tips – Super Quick Fit

（オプションの拡充、派生値の出力、ユーザー関数のサポートを備えた「Quick Fit」）

このプロシージャは Igor Pro のヘビーユーザー作成し、フォーラムに投稿したものに、編集にあたって追記したものです。

オリジナルの投稿は下記。

<https://www.wavemetrics.com/node/21866>

概要

このプロジェクトは、標準搭載の「Quick Fit」に代わるものです。「Super Quick Fit」は、「Quick Fit」のすべての機能を網羅し、さらにいくつかの便利な機能を追加しています。グラフまたはトレースを右クリックすると、メニューが表示されます。

（開発者より）

バージョン 1.05 以降の機能改善案を提案し、そのテストを行ってくださった *Uppsala University* の *Oleg Dyachok* 氏に、特に感謝申し上げます。

あらゆる機能を徹底的にテストするよう最善を尽くしましたが、時折バグが見落とされることもあります。

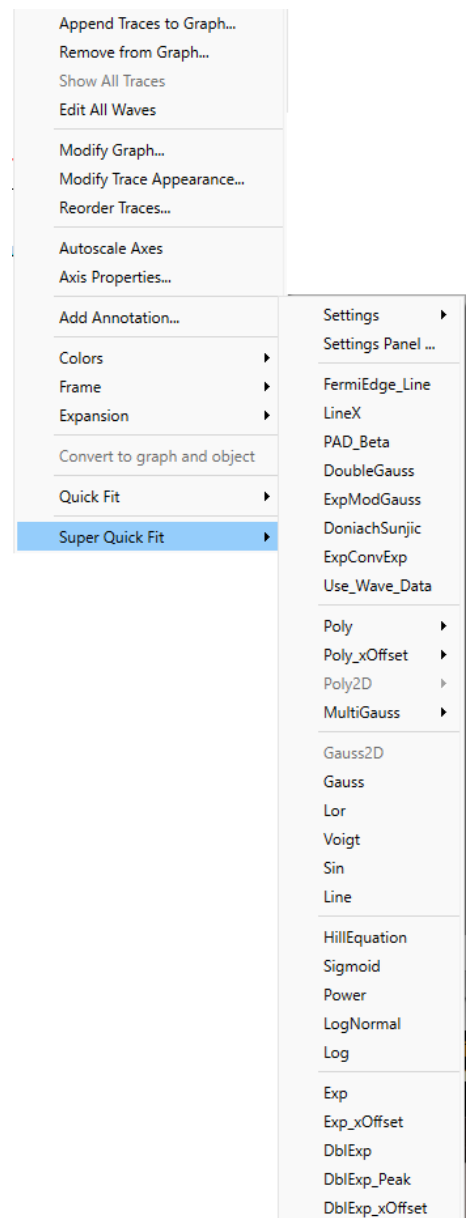
バグ報告や新機能に関するご提案はいつでも歓迎いたします。

フィードバックをいただければ、より便利な設定やピーク形状、派生値などを喜んで追加させていただきます。

現在、2D フィッティングのサポートは完全ではない可能性があります。

これは、すべてをテストするための十分なデータがないためです。

2D フィッティングの改善に関するご提案や、この機能を詳細にテストしていただける方がいらっしゃいましたら、ぜひお知らせください。



インストール

上記のフォーラムの URL からプロシージャファイル Super Quick Fit.ipf をダウンロードします。

1. User Procedures にプロシージャファイルを配置します。
2. Igor Pro を起動し、メニュー Windows→Procedure Windows→Procedure Window を選択します。
3. `#include "Super Quick Fit"`
という行を追加し、下部の Compile ボタンをクリックします。

何らかのグラフを表示し、グラフウィンドウ内で右クリックすると、コンテキストメニューに Super Quick Fit アイテムが追加されていることがわかります。

改良される設定

Fit Between Cursors

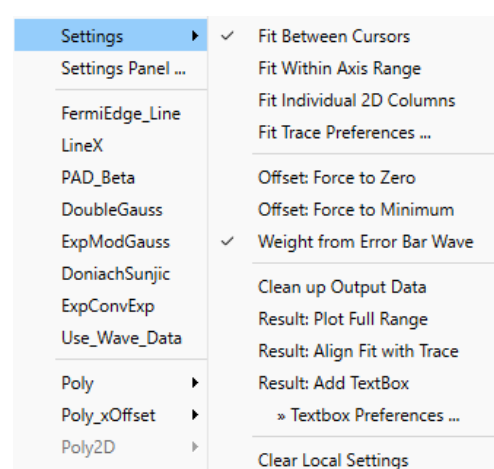
Quick Fit と同じです。フィッティングはカーソル間の範囲に限定されます。

Fit Within Axis Range

フィッティング範囲は、グラフの上限／下限（画像の場合は左／右）の軸スケールに制限されます。

Fit Individual 2D Columns

グラフに 2D ウェーブのトレースが含まれている場合、各フィッティング済み列には、「fit_XX_」という接頭辞が付いた独自の出力ウェーブ名が割り当てられます。
XX はフィッティング済み列を表し、桁数はウェーブの y 方向のサイズによって決まります（例えば、200 列のウェーブの場合、最初の列の出力名は「fit_000_wavename」となります）。



Fit Trace Preferences

プレファレンスによるフィッティングトレースのスタイルを選択するためのパネルを開きます（以下を参照）。

Offset: Force to Zero

フィッティング関数の「オフセット値」（多くの場合、基準値 y0）はゼロに固定されます。

Offset: Force to Minimum

「オフセット値」は最小値に保たれます。この最小値は、配置されたカーソルによる最小値、またはデータの最小値のいずれかによって決定されます（フィッティング関数によっては、x 軸または y 軸の最小値のいずれかとなる場合があります）。

Weight from Error Bar Wave

Quick Fit と同じです。重みウェーブは、トレースの誤差ウェーブから取得されます。

Clean up Output Data

フィッティング終了後、「fit_」ウェーブ以外出力ウェーブ、文字列、変数を削除します。履歴出力やグラフのテキストボックスのみを参照したい場合、W_coef や W_sigma などのウェーブが蓄積されるのを避けたい場合に、これを使ってください。

Result: Plot Full Range	グラフの軸の全範囲にわたって出力を生成します。
Result: Align Fit with Trace	フィッティングトレースのスケールとオフセットを、ソースと同じ方法に調整します（つまり、フィッティングをソースに合わせるようにします）。
Result: Add TextBox	Quick Fit と同じです。グラフに、結果値を含む標準のテキストボックスを追加します。
Textbox Preferences	Quick Fit と同じです。結果のテキストボックスに表示させたい内容を選択してください。

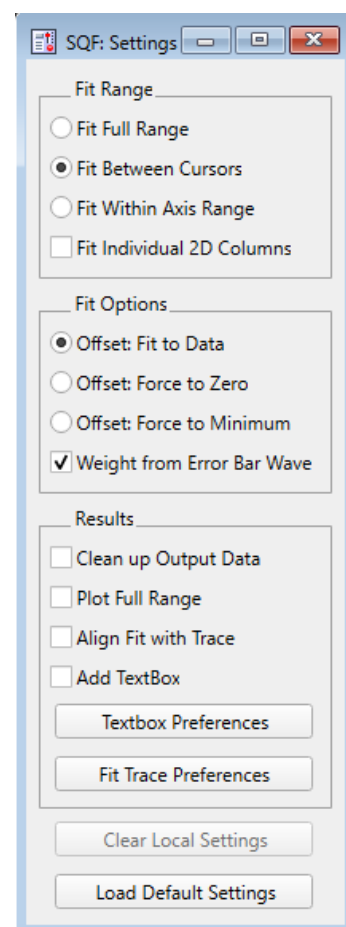
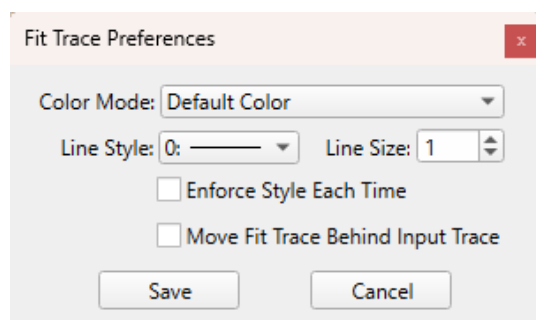
設定は現在のエクスペリメント（root:QuickFitMenu_settings として）に保存され、各エクスペリメントファイル間で使った設定の一貫性が保たれます。

また、最後に使った設定は Igor Pro のセッションをまたいで保持されるため、お好みの作業スタイルをデフォルトとして設定することが可能です。Clear Local Settings メニューを選択すると、現在のエクスペリメントの設定（つまり QuickFitMenu_settings）がすべて削除され、代わりに永続的なグローバル設定が読み込まれます。

メニュー Settings Panel を選択すると、右のダイアログが表示され、メニューと同様の設定を GUI で行えます。

Fit Trace Preferences ボタンをクリックすると、以下のダイアログが表示されます。

Fit Trace Preferences では、新しいフィットトレースに適用する色や線のスタイルを設定できます。



これにより、フィッティングトレースを元のトレースとより明確に区別できるようになります。Enforce Style Each Time を有効にしない限り、既存のフィットトレースには影響しません。

追加のフィッティング関数

（Curve Fitting ダイアログでは、EMG 以外の関数も使用可能です）

FermiEdge_Line Fermi-Dirac 分布と線形勾配を組み合わせたものです。

$$\text{式: } f(E) = y_0 + (m \cdot E + A) / (\exp((E - E_f)/kT) + 1)$$

LineX X 切片（x0）を直接の係数として用いた直線近似です。特に傾きの急な直線において、不確実性の少ない x 切片を求めるのに有用です。デフォルトの直線近似は Y 切片に基づいており、Y 切片を求めるのには適していません。

$$\text{式: } f(x) = m \cdot (x - x_0)$$

PAD_Beta 角度分布関数です。例えば、光電子エクスペリメントの解析などに用いられます。

式 : $f(\theta) = \text{Scale} * (1 + \text{Beta} * \text{legendreA}(2, 0, \cos((\theta - \theta_0) * (\pi/180))))$

DoubleGauss

2つのガウス関数を組み合わせてダブルレットにフィッティングさせます。2つの明確な「ピーク」特徴がない入力データに対しては、うまく機能しません。

式 : $f(x) = y_0 + \text{Area1} * \text{Gauss}(x, \text{Loc1}, \text{Width1}) + \text{Area2} * \text{Gauss}(x, \text{Loc2}, \text{Width2})$

MultiGauss

最大 10 個のガウス関数を組み合わせることができます。メニューから関数の数を選択してください。ただし、選択した関数の数に対して、ピークパラメーターの自動推定が成功するとは限りません。

ExpModGauss

指数修正ガウス (EMG) のピーク形状です。「Multipeak Fit」パッケージで提供されているものと同じです。

ExpConvExp

2つの指数関数的減衰関数の畳み込みです。Multipeak Fit パッケージで提供されているものと同じです。

Doniach-Šunjić

X線光電子分光法で観測される、よく知られた非対称なピーク形状です。Multipeak Fit パッケージで提供されているものと同じです。

Use_Wave_Data

スケール調整および位置ずれを行ったウェーブをフィッティングさせます。

導出値の出力

FWHM、面積、高さなどの追加の導出パラメーター（該当する場合）は、履歴とグラフのテキストボックスに表示されます。

現在、以下の関数が派生パラメーターを出力します : LineX、DoubleGauss、MultiGauss、ExpModGauss、ExpConvExp、Line、Gauss、Lor(entz)、Voigt、Doniach-Šunjić。

また、フィッティング後に追加の派生パラメーターを出力するように独自の関数を設定することも可能です（以下を参照）。

ExpConvExp の代替派生パラメーター

ExpConvExp 関数は、Multipeak Fit パッケージの標準的な関数形式を使いますが、別の関数形式に関する派生パラメーター（仮想振幅、立ち上がり時間の逆数、立ち下がり時間の逆数）も併せて示されています。

この別の形式は、細胞生物学などの一部の分野ではより一般的に用いられています。

以下に、両方の関数形式を比較します。

$$f_{std}(x) = y_0 + \text{Amp} \frac{\tau_1}{\tau_2 - \tau_1} (e^{-\tau_1(x-x_0)} - e^{-\tau_2(x-x_0)})$$

$$f_{alt}(x) = y_0 + \text{Amp}_{virt} (1 - e^{-\tau_{rise}(t-\mu)}) e^{-\tau_{decay}(t-\mu)}$$

コマンドラインまたは独自のコードから「Super Quick Fit」を呼び出すには

Super Quick Fit は、以下のコードを実行することで実行することもできます。

```
SuperQuickFit(string graphName, string traceName, string fitFuncName [,int quiet])
```

‘graphName’ には、開いているグラフの任意の名前を指定するか、最上位のグラフの場合は単に “” を指定できます。

‘traceName’ も空 (“”) のままにしておくと、画像プロットまたはグラフの最初のトレースにフィッティングされます。

‘fitFuncName’ は、有効な組み込み関数またはユーザー関数の名前でなければなりません。

SuperQuickFit() はエラーコードを返します（フィッティングが成功した場合は 0）。

直近のフィッティング設定が使われ、現時点ではコードから変更することはできません。

この関数を実行する前に、まず右クリックメニューですべての設定を行ってください。

単一のグラフ内の複数のトレース、または複数のグラフ内のトレースをフィッティングさせる場合に最も有用です。

オプションの ‘quiet’ フラグを使うと、履歴へのすべての出力を抑制できます。

エラーコードは以下の通りです（例えば、関数呼び出しで print を使う場合など）。

- 1 = 有効なターゲットまたは関数がない
- 2 = グラフが見つからない
- 3 = 指定されたグラフが存在しない
- 4 = グラフ上にトレースがない
- 5 = 無効な関数形式
- 6 = 無効なコンタートレース
- 10 = フィッティング関数の推定に失敗した
- 100 = 画像の X 軸または Y 軸が別々になっている（現時点では未対応）
- 999 = フィッティングの実行に失敗した

自分のフィッティング関数を追加する

このような関数の追加は簡単です。

通常の（例えば、Curve Fitting ダイアログで作成されたような）フィッティング関数、あるいは一括フィッティング関数を記述し、その後に初期推定値を計算するための 2 つ目の関数を追加します。

パッケージに含まれている LineX 関数のような簡単な例を見えます。

```
function LineX(WAVE w, Variable x) : FitFunc
    //CurveFitDialog/ Equation:
    //CurveFitDialog/  $f(x) = m \cdot (x - x_0)$ 
    //CurveFitDialog/ End of Equation
    //CurveFitDialog/ Independent Variables 1
    //CurveFitDialog/ x
    //CurveFitDialog/ Coefficients 2
    //CurveFitDialog/  $w[0] = x_0$ 
    //CurveFitDialog/  $w[1] = m$ 
    return w[1]*(x-w[0])
end
```

初期推定値を計算するには（また、この関数を「Super Quick Fit メニュー」の項目として表示させるには）、SuperQuickFitStruct 構造体を入力として受け取る、YourFitFuncName+“_prepareCoef” という名前の関数を用意する必要があります。

したがって、例えば、関数の名前を LineX_prepareCoef とする必要があります。

```
function LineX_prepareCoef(STRUCT SuperQuickFitStruct &s)
    Make/D/O/N=2 W_coef = {0,1}
    WAVE s.cw = W_coef
```

```

        return 0
    end

```

この関数は、正しい数の係数（LineX の場合は 2 つ必要）を持つ係数ウェーブを作成し、フィッティングが成功する可能性が高い初期推定値を設定する必要があります。

作成された係数ウェーブは、ウェーブ代入 s.cw に渡す必要があります。

ちなみに、このウェーブを W_coef と名付ける必要はありません。

ただし、この標準的な名前を選択した場合、「クリーンアップ」設定が有効になっていると、係数ウェーブは適切に削除されます。

これで、Super Quick Fit でその関数を選択できるようになります。

上記の例では、初期推定値は単に $x_0=0$ 、 $m=1$ と設定されています。

これは、適切な推定値を得るには単純すぎるかもしれません。

入力データにアクセスして、より妥当な推定値を算出することも可能です。

LineX の完全な関数は次のようになります。

```

function LineX_prepareCoef(STRUCT SuperQuickFitStruct &s)
    WAVE temp_yw = SQF_getTraceWave(s)                // 関連するトレースデータを抽出
    WaveStats/Q temp_yw
    variable Wmax = V_max                             // データの最小値と最大値を取得
    variable Wmin = V_min
    Differentiate temp_yw/D=temp_yw
    WaveStats/Q temp_yw
    variable slope = ((Wmax-Wmin)>0) ? V_max : V_min    // 導関数から傾きを求める

    Make/D/O/N=2 W_coef = {(s.holdMode==1 ? 0 : s.xBase),slope} // 係数ウェーブ(2 ポイント)を準備
    WAVE s.cw = W_coef
    return 0
end

```

ここでは、入力データの導関数から傾き m を抽出し、フィッティングカーネルから提供されたデータからオフセット x_0 を算出します。

なお、入力ウェーブは s.data に格納されていますが、これは必ずしも実際にプロットされたデータポイントを表すわけではありません（s.data の一部のみがプロットされ、フィッティングに使われる場合があります）。

関連するデータポイントのみを使うようにするため、SQF_getTraceWave(s) が呼び出されます。

この関数は、関連するデータポイントのみを含む、フリーの 1 次元ウェーブを返します。

必要に応じて、SQF_getTraceXWave(s) を使って、対応する x ウェーブを取得してください。

s.holdMode は現在のオフセットモード（1 の場合はオフセットが強制的に 0 に設定される）であり、s.xBase は現在の x 軸の最小値（カーソル位置か、入力データの最小値のいずれか）です。

これにより、 x_0 の係数が適切に設定され、直線がカーソル位置または 0 に固定されるようになります。

'force offset' 設定は、デフォルトでは最初の係数に適用されます。

他の係数を固定したい場合は、s.holdStr に適切な固定文字列を指定する必要があります（例：s.holdStr = 「00001」 とすると、5 番目の係数が固定されます）。

構造体の完全な定義は次のようになります。

```

Structure SuperQuickFitStruct

    string fitFunc        // 使われるフィッティング関数名
    WAVE cw               // 係数ウェーブ
    WAVE sw               // 共分散行列 (M_Covar)
    WAVE data             // データウェーブ (1D または 2D)
    WAVE fit              // フィッティングウェーブ (フィッティングが成功した場合のみ)
    WAVE xw               // x ウェーブ
    WAVE yw               // y ウェーブ (画像とコンター用)
    int32 pntMin          // フィッティング範囲 (画像の x 方向): 開始ポイント
    int32 pntMax          // フィッティング範囲 (画像の x 方向): 終了ポイント
    int32 pntMinY         // 画像の y 方向: 開始ポイント
    int32 pntMaxY         // 画像の y 方向: 終了ポイント
    int32 xw_Min          // トレースの x ウェーブのフィッティング範囲

```

```

int32 xw_Max
int32 yw_inc          // プロットされたトレースのインクリメント
int32 xw_inc
int16 yw_trDim        // トレース -> y ウェーブで使われる次元
int16 xw_trDim        // 使われる次元 -> x ウェーブ (存在すれば)
int32 yw_dims[4]      // トレースの y ウェーブの次元 => 各次元の開始点を記録
int32 xw_dims[4]      // トレースの x ウェーブの次元
double xBase          // x ベースライン / 開始オフセット
double yBase          // y ベースライン / 開始オフセット
int16 holdMode        // 現在の Hold 設定: [0] no hold, [1] force o zero, [2] force to min
int16 plotFull        // 全範囲にわたるプロットの設定
int16 doTextbox       // テキストボックス描画の設定
string holdStr        // フィッティングに対する Hold 文字列
EndStructure

```

独自の派生値関数を追加する

Super Quick Fit では、フィッティング完了後に、フィッティング結果から派生値を計算するなど、追加のデータ処理を行うことができます。

そのためには、YourFitFuncName+"_derivedVals" という名前の関数を作成します。

原則として、ここではどのような処理でも行えますが、主な目的は、履歴に出力するための文字列を返すことです。再び例を見てみます (LineX からの例です)。

```

function/S LineX_derivedVals(STRUCT SuperQuickFitStruct &s)
    variable y_c = -s.cw[1]*s.cw[0]
    variable y_e = y_c * sqrt( (s.sw[0][0]/s.cw[0]^2) + (s.sw[1][1]/s.cw[1]^2) +
2*s.sw[0][1]/(s.cw[0]*s.cw[1]))
    string str
    sprintf str, "Derived values:\r\tY-intercept\t= %g ± %g", y_c, (numtype(y_e) ? 0 : y_e)
    if (s.doTextbox)
        AppendText "\t" + str
    endif
    return str
end

```

この関数は、フィッティング係数 (s.cw) と共分散行列 (s.sw) から直線の y 切片を計算し、その結果を出力文字列 printStr に格納します。

ここでは、結果がデフォルトのテキストボックスにも追加されますが、もちろんこの手順を省略することも可能です。

利用可能なすべてのデータは、やはり同じ SuperQuickFitStruct から受け取られます。

結果の追跡分析や、データのディスクへの保存など、任意の処理を行うコードを追加することも可能です。

基本的に、メニューからフィッティングを実行した後 (もちろん、フィッティングが成功している必要がありますが)、実行したい処理はどのようなものでも追加できます。

汎用の派生値関数

もし、Super Quick Fit が正常に実行されるたびに後処理を行いたい場合は、汎用的な (フィッティング関数に依存しない) 関数を記述することもできます。

この関数の名前は「SuperQuickFit_derivedVals」でなければなりませんが、それ以外は、前述のフィッティング関数固有のバージョンとまったく同じように動作します。

なお、汎用的な後処理関数とフィッティング関数固有の後処理関数の両方が存在する場合、汎用的な関数が最後に実行されることに注意してください。

以下の関数は、その仕組みを示しています。

この関数は、結果の注釈に短い文章を追加するだけのものです。

繰り返しになりますが、Super Quick Fit 自体の正常な完了を Abort で中断しないこと以外、制限はありません。

```
function/S SuperQuickFit_derivedVals(STRUCT SuperQuickFitStruct &s)
    if (s.doTextbox)
        AppendText "This is a test."
    endif
    return ""
end
```