

CONTENTS

サンプルの Experiment – Spectral Deconvolution Demo	2
クイックノート	2
手順.....	3
プロシージャの内容.....	4

サンプルの Experiment – Spectral Deconvolution Demo

クイックノート

メニュー File → Example Experiments → Analysis → Spectral Deconvolution Demo

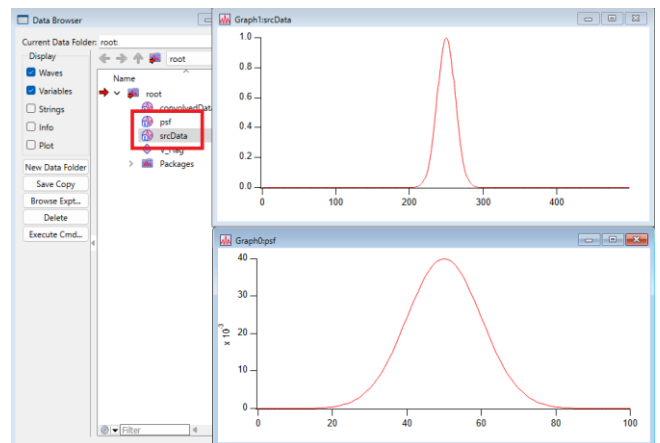
この Experiment は、スペクトルのデコンボリューションを行うための簡単なプロシージャの使用例です。このアルゴリズムは、Peter A. Jansson の著書「Deconvolution With Applications in Spectroscopy」(Academic Press、1984 年、ISBN 0-12-380220-2) に基づいています。これは同書の初版です。

デモはデコンボリューションパネルでコントロールされます。

まず、ソースと PSF を選択します。

デモには、ウェーブ srcData を、点像分布関数ウェーブ PSF で畳み込んで、先に作成した convolvedData ウェーブがすでに存在しています。

Data Browser のグラフ領域の関連するプロファイルを確認すると、srcData と PSF の両方にガウスプロファイルが確認できます。



ここで使うアルゴリズムは、現在の推定値を緩和関数と入力スペクトルと推定値の PSF との畳み込みの差の積を用いて修正し、その修正値を次の推定値として反復的に推定するプロセスです。

$$\text{nextSpectrum} = \text{currentEstimate} + \text{relaxationFunction} * (\text{currentEstimate} ** \text{psf})$$

** は畳み込みを表します。

初期推定値は入力データ（畳み込みスペクトル）です。

このデモでは、PSF が既知であり、PSF と畳み込みスペクトルが同じスケールを持つものと仮定しています。

緩和関数の 3 つの選択肢は、Jansson が提案した以下の通りです : Jansson 1970、Blass と Halsey、およびガウス関数 (Jansson の 183 ページを参照) 。

3 つの機能は、滑らかな/ガウス分布のような PSF の場合、いずれも十分な性能を発揮しているようです。

しかし、例えば $\text{Rect}(x/a)$ のような PSF を選択した場合、アルゴリズムが収束するために異なる緩和関数が必要になる場合があります。

手順

デモ Experiment を開きます。

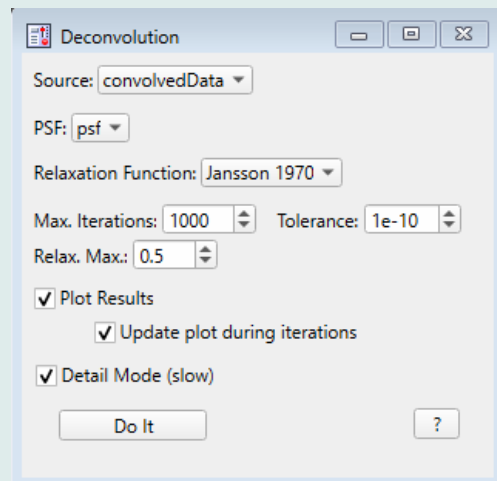
Source で convolvedData を、PSF で psf を選択します。

各項目では、緩和関数、反復数、許容値などを設定します。

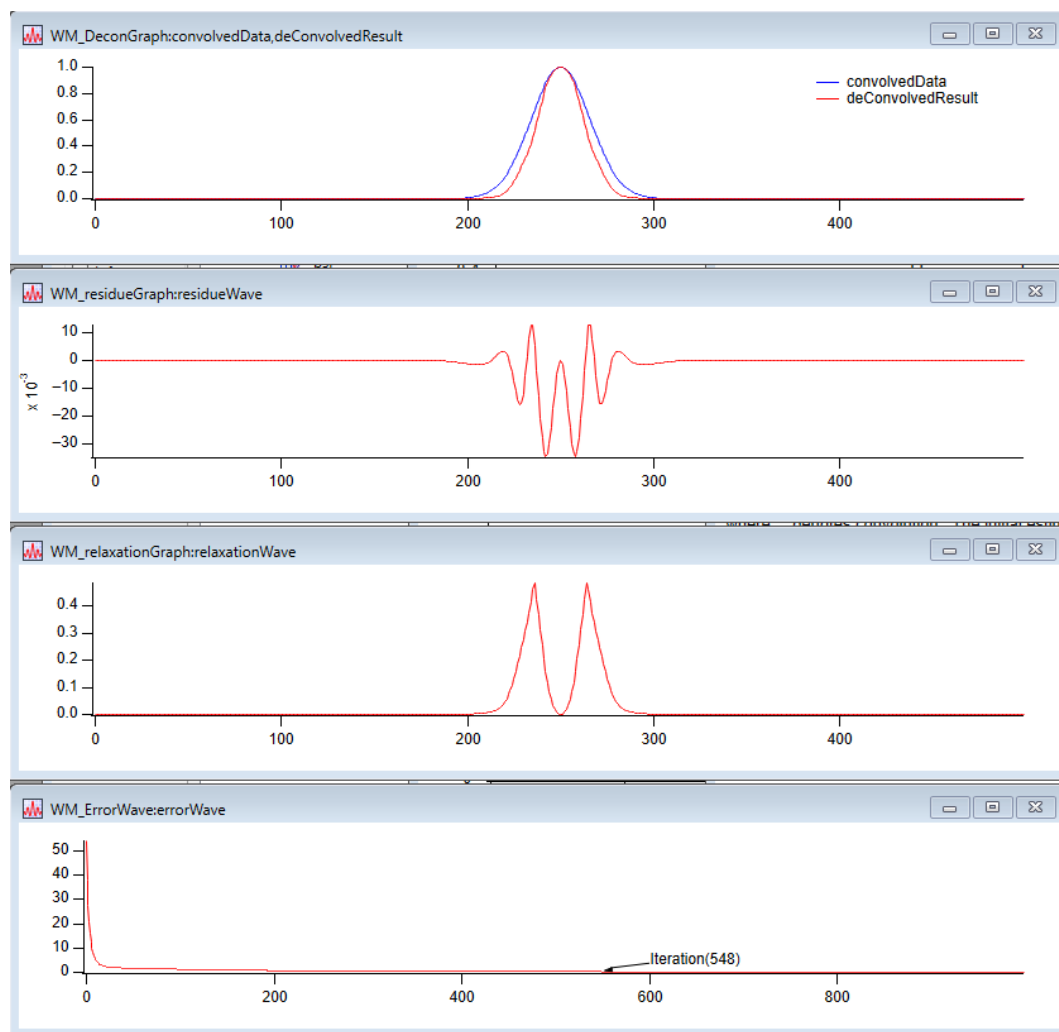
Plot Result をチェックすると結果がグラフに表示されます。

Update plot during iterations をチェックすると反復ごとにグラフが更新されます。

Detail Mode をチェックすると、より詳細に処理されますが、動作が遅くなります。



Do It をクリックすると、処理中のグラフが更新され、結果が表示されます。



プロシージャの内容

```
// これは 1 次元デコンボリューションを行うためのメインの呼び出しです。入力は :
// convolvedData -- デコンボリューションするデータを含むウェーブ
// psfData -- 点拡散関数を表すウェーブ
// iterations -- ソリューションにおける反復回数
// relaxFunction -1: Jansson, 2: Blass, 3: Gaussian.
// relaxMax -- 操作の内容を十分に理解している場合を除き、1 に設定してください

// 一般的な注意事項 :
// convolvedData および psfData は、[0,1] の値を持つ正規化されたウェーブであることが予想されます。
// 2 つのウェーブには、ウェーブスケールリングを一切適用しないでください。
// このプロシージャではウェーブスケールリングは削除されていますが、必要に応じて追加してください。
// このコードは "Deconvolution With Applications in Spectroscopy", Peter A. Jansson (1st ed.) に基づいています。
// アルゴリズムは 183 ページに記載されており、以下で使われる 3 つの緩和関数を説明しています。
//
```

```
Function WM_Deconvolve1DPanel()
    DoWindow/F WM_DeconPanel
    if(V_Flag==1)
        return 0
    endif

    String oldDF=GetDataFolder(1)
    SetDataFolder root:
    NewDataFolder/O/S Packages
    NewDataFolder/O/S WM_1DDeconvolution
    Variable/G numIterations=1000
    Variable/G relaxationMax=0.5
    Variable/G convergenceEpsilon=1e-10

    // コントロールパネル
    NewPanel /K=1 /W=(431,140,737,411) as "Deconvolution"
    PopupMenu WM_DeconSource,pos={7,7},size={83,20},title="Source:"
    PopupMenu WM_DeconSource,mode=1,popvalue="psf",value= #"WaveList(¥"*¥", ¥";¥",
¥"DIMS:1¥" )"
    PopupMenu WM_DeconPSF,pos={7,38},size={69,20},title="PSF:"
    PopupMenu WM_DeconPSF,mode=1,popvalue="psf",value= #"WaveList(¥"*¥", ¥";¥",
¥"DIMS:1¥" )"
    SetVariable WM_DeconIterations,pos={7,98},size={145,15},title="Max. Iterations:"
    SetVariable WM_DeconIterations,value=
root:Packages:WM_1DDeconvolution:numIterations
    CheckBox WM_DeconPlotCheck,pos={8,150},size={72,14},title="Plot Results"
    CheckBox WM_DeconPlotCheck,value= 0
    CheckBox WM_DeconUpdateCheck,pos={47,172},size={146,14},title="Update plot
during iterations"
    CheckBox WM_DeconUpdateCheck,value= 0
    Button
WM_DeconDoIt,pos={24,231},size={96,20},proc=WM_DeconDoitButtonProc,title="Do It"
    PopupMenu WM_DeconRelFuncPop,pos={7,67},size={201,20},title="Relaxation
Function:"
    PopupMenu WM_DeconRelFuncPop,mode=1,popvalue="Jansson 1970",value= #"¥"Jansson
```

```

1970;Blass & Halsey;Gaussian¥""
    Button
WM_DeconHelpButton,pos={254,230},size={30,20},proc=WM_DeconHelpButtonProc,title=" ? "
    SetVariable WM_DeconRMxSetVar,pos={7,121},size={120,15},title="Relax. Max.:"
    SetVariable WM_DeconRMxSetVar,value=
root:Packages:WM_1DDeconvolution:relaxationMax
    SetVariable setvar0,pos={165,98},size={120,15},title="Tolerance:"
    SetVariable setvar0,value= root:Packages:WM_1DDeconvolution:convergenceEpsilon
    CheckBox WM_DeconDebugCheck,pos={9,200},size={102,14},title="Detail Mode (slow)"
    CheckBox WM_DeconDebugCheck,value= 0

    DoWindow/C WM_DeconPanel

    SetDataFolder oldDF
End
//
// Do It ボタンを押したときの処理
Function WM_DeconDoitButtonProc(ctrlName) : ButtonControl
    String ctrlName

    // チェックボックスオプション
    Variable wantsDisplay=0
    Variable wantsUpdate=0
    Variable debugMode=0

    NVAR numIterations=root:Packages:WM_1DDeconvolution:numIterations
    NVAR relaxationMax=root:Packages:WM_1DDeconvolution:relaxationMax
    NVAR convergenceEpsilon=root:Packages:WM_1DDeconvolution:convergenceEpsilon

    Variable relaxFunction

    // 選択したウェーブを取得し、それらが有効であることを確認
    ControlInfo WM_DeconSource
    Wave convolvedData=$S_Value
    if(WaveExists(convolvedData)==0)
        Abort "Bad source wave specification."
        return 0
    endif

    ControlInfo WM_DeconPSF
    Wave psfData=$S_Value
    if(WaveExists(psfData)==0)
        Abort "Bad PSF wave specification."
        return 0
    endif

    ControlInfo WM_DeconDebugCheck
    debugMode=V_Value

    if(debugMode)
        Wave targetWave=root:srcData
    endif

    // ウェーブの設定が完了したので、データフォルダーをプライベート DF に設定
    String oldDF=GetDataFolder(1)
    SetDataFolder root:Packages:WM_1DDeconvolution

```

```

Variable dataSize=DimSize(convolvedData,0)
if(DimSize(convolvedData,1)>0)
    abort "this is a 1D deconvolution."
    return 0
endif

// 約束通り、ローカルコピーからウェーブスケーリングをすべて消去
Duplicate/O convolvedData,srcWave,deConvolvedResult
Duplicate/O psfData, psfWave
SetScale/P x,0,1, srcWave,psfWave

// 2つのウェーブを正規化
normalize(srcWave)
normalize(psfWave)

Variable psfSize=DimSize(psfWave,0)
Make/O/N=(dataSize) relaxationWave=0,tmpWave,prevConvolution,oldEstimate
if(debugMode)
    Make/O/N=(dataSize) residueWave
endif

Variable i,integratedChange

Controlinfo WM_DeconRelFuncPop
relaxFunction=V_value

// ユーザーがプロットの表示を希望しているかどうかを確認：
ControlInfo WM_DeconPlotCheck
if(V_Value)
    wantsDisplay=1
    ControlInfo WM_DeconUpdateCheck
    wantsUpdate=V_Value
    if(WinType("WM_DeconGraph")==0)
        Display/K=1 convolvedData,deConvolvedResult
        string name=NameOfWave(convolvedData)
        ModifyGraph rgb($name)=(0,0,65535)
        Legend/C/N=text0/A=RT/F=0
        DoWindow/C WM_DeconGraph

        if(debugMode)
            Display/K=1 residueWave
            DoWindow/C WM_residueGraph
            Display/K=1 relaxationWave
            DoWindow/C WM_relaxationGraph
        endif
    endif
endif

if(debugMode)
    Make/O/N=(numIterations) errorWave=0
    if(WinType("WM_ErrorWave")==0)
        Display/K=1 errorWave
        DoWindow/C WM_ErrorWave
        Tag/N=iterationTag/F=0/L=2/X=20/Y=20 errorWave, 0,"Iteration¥¥OX"
    endif
endif

```

```

        Execute "TileWindows/W=(6,24,624,792)/A=(4,1)
WM_DeconGraph,WM_residueGraph,WM_relaxationGraph,WM_ErrorWave"
    endif

    try          // ユーザーによる中止で、適切なデータフォルダーに戻れることを確認
        for (i=0;i<numIterations;i+=1)
            Duplicate/O deConvolvedResult,oldEstimate
            // 緩和は、現在の推定値の関数
            updateRelaxation (relaxationWave,deConvolvedResult,
relaxFunction,relaxationMax)
            // 現在の推定値と psf の畳み込みを計算
            Duplicate/O deConvolvedResult,prevConvolution
            Convolve/A psfWave,prevConvolution
            normalize (prevConvolution)

            // 新しい推定
            MultiThread
tmpWave=deConvolvedResult[p]+relaxationWave[p]*(srcWave[p]-prevConvolution[p])
            // 正値性を適用し、その後正規化
            WM_applyConstraints (tmpWave,deConvolvedResult)
            normalize (deConvolvedResult)

            if (debugMode)
                residueWave=targetWave-deConvolvedResult
                errorWave[i]=SUM(residueWave,-inf,inf)^2
            endif

            if (wantsUpdate*wantsDisplay && mod(i,2)==0)
                DoUpdate
            endif

            if (debugMode)
                // "Iteration=",i を出力
                Tag/C/X=10/Y=10/N=iterationTag
errorWave,i,"Iteration(¥¥0X) "
            endif

            // 「収束」のテストを実行
            oldEstimate-=deConvolvedResult
            integratedChange=Sum(oldEstimate)^2/dataSize
            if (integratedChange<convergenceEpsilon)
                Print "Reached Convergence level"
                break
            endif
        endfor
    catch
        Print "User Break"
    endtry

    if (debugMode)
        Printf "Iteration change=%g after %d iterations¥r",integratedChange,i-1
    endif

    normalize (deConvolvedResult)

    if (debugMode==0)
        KillWaves/Z

```

```

tmpWave,prevConvolution,relaxationWave,srcWave,psfWave,oldEstimate
endif

SetDataFolder oldDF

End

//
// 次の関数は、デコンボリューションされたデータの最新の値に対して、緩和ウェーブの
// 値を更新する。deConvolvedResult は [0,1] の範囲で正規化することには注意。

Function updateRelaxation(relaxationWave,deConvolvedResult,relaxFunction,relaxMax)
Wave relaxationWave,deConvolvedResult
Variable relaxFunction,relaxMax

// 緩和関数の切り替え
switch(relaxFunction)
case 1: // Jansson
MultiThread relaxationWave=relaxMax*(1-
2*abs(deConvolvedResult[p]-.5))
break

case 2: // Blass and Halsey (1981) using 1 for exponent.
MultiThread relaxationWave=relaxMax*(deConvolvedResult[p]*(1-
deConvolvedResult[p]))
break

case 3: // Gaussian relaxation
MultiThread relaxationWave=relaxMax*exp(-(
(deConvolvedResult[p]-.5)^2/0.11)
break
endSwitch
End

//
// 前のコンボリューションを取得

Function getPrevConvolution(prevEstimate,psfWave,outWave)
Wave prevEstimate,psfWave,outWave

Variable psfSize=numpts(psfWave)
Variable dataSize=numpts(prevEstimate)
Variable i,k,theSum

// out[i]=SUM{f(k)s(i-k)}
for(i=0;i<dataSize;i+=1)
theSum=0
for(k=0;k<psfSize+i;k+=1)
if(k>i || k>=dataSize)
break
endif
theSum+=prevEstimate[k]*psfWave[i-k]
endfor
outWave[i]=theSum
endfor
End

```

```
//
// 正規化

Function normalize(inWave)
    wave inWave

    Variable theMin,theMax
    [theMin,theMax]=WaveMinAndMax(inWave) // Igor Pro 9 以降が必要
    Variable nor=1/(theMax-theMin)
    MultiThread inWave=(inWave[p]-theMin)*nor
End
```

```
//
// デコンボリューションを試すためのいくつかのサンプルデータ
```

```
Function test(n)
    Variable n

    Make/O/N=101 psf=0
    psf[50-n,50+n]=1
    Make/o/n=500 srcData
    srcData=exp(-(x-250)^2/300)
    duplicate/O srcData, convolvedData
    Convolve/A psf,convolvedData
    normalize(convolvedData)
End
```

```
//
// ヘルプボタンを押したときの処理
```

```
Function WM_DeconHelpButtonProc(ctrlName) : ButtonControl
    String ctrlName

    notebook WM_DecNotebook visible=1
    DoWindow/F WM_DecNotebook
End
```

```
//
Function WM_applyConstraints(tmpWave,deConvolvedResult)
    Wave tmpWave,deConvolvedResult

    MatrixOP/O/S deConvolvedResult=tmpWave*(1+sgn(tmpWave))/2
End
```

```
//
```