CONTENTS

サンプルの Experiment – Gizmo Waterfall Demo	
クイックノート	2
はじめに	
手順	3
プロシージャの内容	8

クイックノート

メニュー File \rightarrow Example Experiments \rightarrow Visualization \rightarrow Waterfall

この Experiment は、Gizmo でウォーターフォールプロットを作る方法を説明します。 これには、Igor Pro v9.0 以降が必要です。



出発点は、2D 行列(matrixData)に格納されたスペクトルデータです。 各列は、温度や圧力などの特性に対応し、各行は特定の波長に対応します。 matrixData のウェーブスケーリングには、波長と列プロパティの両方が設定されているものと想定します。 データが矩形グリッドでサンプリングされていない場合は、以下に説明する方法を使う必要があります。

はじめに

新規の Experiment からの手順を説明しますが、データはサンプルの Experiment からコピーして使っても構いあません。



2. メニュー Windows \rightarrow Procedure Windows \rightarrow Procedure Window を開き、サンプルの Experiment からプロシージャをすべてコ ピーします。

コピーしたら、ウィンドウ下部の Compile ボタンをクリックします。



xData[][2]

0.958739

1.04713

matrixData[][3]

0817

1.05117 1.14213

0.98730

@ 2

xData[][4] m

905433

957176

95793

902198

0.976462

3. 行列データをトリプレットウェーブに変換します。

Gizmo のパス(Path)オブジェクトには、トリプレットウェーブ(3 列ウェーブで、各行がパスの頂点の XYZ 座標 を含む)が必要です。

各列は、異なるパスオブジェクトとして、または単一のパスオブジェクトの異なるセグメントとして表示されるべき です。

パスのトリプレットに NaN 値の行を挿入することで、セグメントを分割することができます。

4. [Step 1] 入力行列からトリプレット waterfallPathTriplet を作成するには、メインプロシージャファイルの関数 matrixToGizmoPathTriplet(inMat)を使います。

コマンドウィンドウで次を実行します:

matrixToGizmoPathTriplet(matrixData)

waterfallPathTriplet というウェーブ (パスオブジェクト) が作成され、内容を見ると 50 行ごとに NaN の行が挿入されています。

5. [Step 2] パスオブジェクトから Gizmo プロットを作成します。

メニュー Windows → New → 3D Plot を選択し、メニュー Gizmo → Append Path を選択します。

Source Wave が waterfallPathTriplet になっていることを確認してください。

GUI を使わない場合は、以下のコマンドを実行します。

NewGizmo

AppendToGizmo defaultPath=root:waterfallPathTriplet

Durrent Data Foli	der: roo				RUCU	ш	0			63
Display	4	-> -	h 👪 root	~	Row	wate	rfallPathTriple	waterfallPathTriple	waterfallPa	thTriple
Waves			^				0	1	2	
Variables		reame			46		46	0	0.	390079
Strings	17	~ 4	root		47		47	0	0.	917758
🗌 Info		1	Ella matri	Data	48		48	0	0.	954866
Plot			🕖 water	fallPathTriplet	10		10	0		07004
New Data Folder					50					
new Dana Police					31				÷.	102.014
Save Copy					52		1	1	0.	751295
Browse Expt.					53		2	1	0	95254
Delete	11				54		3	1	0.	995128
Deserve					55		4	1	1	.06177
Execute Cind					56		5	1	0.	965963
					57		6	1	0	94166
							-			
Included										

Rath Properties	×
Name: path0	Live Update
Source Wave: vaterfallPathTriplet	
Color	3D Tube Segments
O None	Parimeter Comments, 20
O Fixed	Permeter deginerity. 20
Color Table Bairbon Details	Fixed Radius: 0.05
	Radkas Wave: ▼_none_
Golar Wave ,none	Draw Segment Endceps
AppendToGiamo patherostmatetalPathTopletnamenpathil ModdhGiamo displayLastChard AppendToGiamo auss-defaultAves.name=auss0	
Do R To Ond Line To Olip	Cancel

GUI を使わない場合は、以下のコマンドを実行すると一度に処理できます(GUI を使うときはこの手順は不要です)。

ModifyGizmo ModifyObject=axes0,objectType=Axes,property={3,ticks,3}
ModifyGizmo ModifyObject=axes0,objectType=Axes,property={2,visible,0}
ModifyGizmo ModifyObject=axes0,objectType=Axes,property={6,visible,0}
ModifyGizmo ModifyObject=axes0,objectType=Axes,property={11,visible,0}

7. GUI での操作の場合は次のようになります。

Gizmo0 コントロールパネルで axes0 をダブルクリックします。



Axis Type: Box

Axis Labels Grids and Planes Custom Axis

Select All

Cle

 \sim

Name: ayes()

Axis

🔽 X1

🔽 X2

Ticks and Labels

8. 「Axis」タブで、X3、Y1、Z0 のチェックを外します。

「Ticks and Labels」 タブを選択し、Axis ポップアップメニューから Z1 を選択します。

Show Tick Marks チェックボックスと Show Numerical Labels チェックボックスをオンにし、Do It をクリックします。



9. グラフは右図のようになります。

10. [Step 4] リボンオブジェクトを追加して、スペクトルに垂直方向の 塗りつぶしを適用します。

そのためには、パスをリボンに変換する必要があります。

コマンドラインを使う場合は次のように入力します。

Modifygizmo pathToRibbon={waterfallPathTriplet,2,0}

これで、waterfallPathTriplet_R というウェーブが作成されます。

11. Gizmo0 Info パネルの Object List 下の「+」マークで Ribbon を選択します。

Source Wave で waterfallPathTriplet_R を選択します。

Do It をクリックします。

12. ribbon0 を Object List から Display List にドラッグして、 Display List に追加します。

コマンドラインでは次のように入力します。

AppendToGizmo

ribbon=root:waterfallPathTriplet_R,name=ribbon0

13. [Step 4] リボンに色を適用します。

リボン全体に単一の一定した色を適用することもできますし、スペクトルの各平面に異なる色を与えるカラーウェーブを適用して、よりクリエイティブな表現をすることもできます。

ここでは、スペクトル平面のそれぞれに RGBA のエントリーを1つずつ含 むウェーブを作ることから始めます。

コマンドウィンドウで次を実行します。

Make/O/N=(6,4) ribbonColor Edit ribbonColor

このテーブルにサンプルの Experiment からデータをコピーします。







F	R0 C0	1			۲
Row	ribbonColor[][0]	ribbonColor[][1]	ribbonColor[][2]	ribbonColor[][3]	
	0	1	2	3	
0	1	0	0	0.2	
1	0	1	0	0.2	
2	0	0	1	0.2	
3	1	1	0	0.2	
4	0	1	1	0.2	
5	1	0	1	0.2	
6					

14. リボンオブジェクトのカラーウェーブを作成するには、関数 makeColorWaveForRibbon()を使って、各スペクトル平面でトリプレッ トの値をベースとした色を生成します。

コマンドウィンドウで次を実行します。

makeColorWaveForRibbon(waterfallPathTriplet R,ribbonCol or)

	_	0	1	2
ribbonColorWave というウェーフか生成されます。	0	1	0	
	1	1	0	
	2	1	0	
	3	1	0	
	4	1	0	
	5	1	0	1
	6		0	
			0	
15 Cizmoo Info パナルズ ribbono たびブルクレックレス Dibbon				
	Distance D			
	Kibbon Pi	opercies		
Properties の Color セクションで Color Wave ラジオボタンクリッ	Name: ribbor	0		
クレーribbonColorWaya を選択します	Source Wave:	▼waterfallPath	Triplet_R	
	Color			
	0			

🗌 Info Ph New Data F

Save Con

R0 C0

クし、ribbonColorWave を選択します。 Do It をクリックします。

Name: ribbon0	Live Update
Source Wave: waterfallPathTriplet_R	
Color	
O None	
Fixed	
Color Table Rainbow Details	
O Color Wave ribbonColorWave	
Calculate Normals	
Do It To Cmd Line To Clip	Cancel
Do It To Cmd Line To Clip	Cancel

waterfallPathTriplet R

83 2

0.2

1

16. [Step 6] さらに次に進みます。 リボンが不透明な塗りつぶしを使っているため、スペクトルの表示は理想 的ではありません。

リボンを半透明にして表示するには、Gizmo メニューから「Enable Transparency Blend」を選択します。



17. かなり良くなりましたが、表示にまだ問題があることは明らかであり、 これはスペクトル平面が描画される順序と関係があります。 Z 軸を中心に表示を回転させるとそのことがわかります。 Z 軸が左側に来るように表示を回転させると、リボンウェーブのデータを 簡単に反転させることができます。 ここでは、コマンドで反転させてみます。



18. コマンドウィンドウで次を実行します。

MatrixOP/O

waterfallPathTriplet_R=reverseCols(waterfallPathTriplet_R)

一般的には、リボンカラーウェーブも反転させるべきですが、このケースで はすべてのセグメントが同じサイズなので、カラーウェーブの順序を反転 させる必要はありません。



例えば、コマンドウィンドウで次を実行します。

makeSampleScatterFromMatrix(matrixData,10)

scatterTriplet というウェーブが生成されます。 10 は間引きの度合いです(表示するポイントの数)。

20. Gizmo0 Info パネルの Object List の下の「+」マークで、Scatter を選択します。

Scatter Wave で、scatterTriplet を選択し、Shape セクションの Fixed Share を Sphere、Size セクションの Fixed を 0.3 にして Do It をクリックします。

21. scatter0 を Object List から Display List にドラッグします。



- D X

Gizmo0:waterfallPathTriplet,waterfallPathTriplet_R

Scatter Properties	×
Name: scatter0	Live Update
Scatter Wave: 🖝 scatter Triplet	
Shape and Size Color Rotation Drop Lines	
Shape Fixed Shape Sphere	
Shape Wave none_	
Size	
Fixed 0.3 Marker Size Wave scatterTriplet	
Do It To Cmd Line To Clip	Cancel



ポップアップメニューから GL_FRONT_AND_BACK を選択し、色を黄 色にします。 🐹 Shininess

25. Object List \mathcal{O} light 0, Attribute List \mathcal{O} specular 0, shininess0、blendFunc0 を Display List に移動します。

適用される順番を示すため、Display List 内の順番は重要です。 上から liaht0 specular0 shininess0 axes0… のように並べてください。

結果は右図のようなグラフになります。

プロシージャの内容

メニュー Windows \rightarrow Procedure Windows \rightarrow Procedure Window 内のコードを掲載します。

// x と y のコンポーネントにウェーブスケーリングを、z のコンポーネントにマトリックス要素を使用して、 // マトリックスデータを表すトリプレットウェーブを作成。 // トリプレットウェーブでは、列は NaN 行によって分離されているため、Gizmo にパスオブジェクト // として表示された場合、連続するカラムは連結されない。 Function matrixToGizmoPathTriplet(inMat)

Wave inMat

光を当てます。 Gizmo0 Info パネルの Object List の下の「+」マークで、Light を選 択します。

22. 散布図のポイントはソリッドな Sphere なので、サンプルのように

ライトの位置は適当に決めます(例では Azimuth は 70、Elevation は -45)。

Gizmo0 Info パネルの Attribute List の下の「+」マークで、Specular

23. 次に、ライトの属性(Attribute)を決めます。

を選択します。

24.	次に、	同様に	Shi	nine	ss を	選択しま	す。			
Fron	t Shini	iness	を 5	に、	Back	Shinine	ess を	20	に設定し	ます。





Specular	×	
Name: specular0		
Face: GL_FRONT_AND_BACK		
AppendToGizmo attribute specular=11,0.999969,0.500008,1,1032]name=specular0		
Do It To Cmd Line To Clip Cancel		

Name: shininess0 Front Shininess: 5 Back Shininess: 20

AppendToGizmo attribute shininess={5,20},name=shininess0

```
// サニティチェック(明らかな不整合がないかチェック):
       Variable type=WaveType(inMat)
       if(type==0)
              doAlert 0, "Bad input wave."
              return 0
       endif
       if(DimSize(inMat,2)>0)
              doAlert 0, "Bad input dims."
              return 0
       endif
       // 固定名の出力を構築:
       Variable rows=DimSize(inMat,0)
       Variable cols=DimSize(inMat,1)
       Variable newRows=(rows+1)*cols
       Make/O/N=(newRows,3) waterfallPathTriplet
       Make/O/FREE/N=(rows) xwave=DimOffset(inMat,0)+p*dimDelta(inMat,0)
       Variable i, j, yy, count=0
       for(i=0;i<cols;i+=1)</pre>
              yy=DimOffset(inMat,1)+i*dimDelta(inMat,1)
               for(j=0;j<rows;j+=1)</pre>
                      waterfallPathTriplet[count][0]=xwave[j]
                      waterfallPathTriplet[count][1]=vv
                      waterfallPathTriplet[count][2]=inMat[j][i]
                      count+=1
              endfor
              // セグメントを切るために NaN を追加:
              waterfallPathTriplet[count][0,2]=NAN
              count+=1
       endfor
// リボンを表すトリプレットウェーブを取り、colData で渡された個々の色を使って、
// 各リボンセグメントに一定の色を提供する。
// セグメントの数が指定された色数を超える場合、残りのセグメントには最後の色が使用される。
Function makeColorWaveForRibbon(inRibbon, colData)
       Wave inRibbon, colData
       // サニティチェック(明らかな不整合がないかチェック):
       Variable rows=DimSize(inRibbon,0)
       if(rows<=0 || dimSize(inRibbon,1)!=3)</pre>
              doAlert 0, "Bad dimensionality for ribbon wave."
              return 0
       endif
       Variable colorRows=DimSize(colData,0)
       if(colorRows<=0 || dimSize(colData,1)!=4)</pre>
              doAlert 0, "Bad dimensionality for colData wave."
              return 0
       endif
       // 出力を作成:
       Make/O/N=(rows,4) ribbonColorWave=0 // NaN の行には alpha をゼロに設定。
       Variable i, segmentCounter=0
       for(i=0;i<rows;i+=1)</pre>
              // 各セグメントごとに色を変える:
              if (segmentCounter<colorRows && numType(inRibbon[i][0])==2)
```

End

End

// 行列データからデータをサンプリングしてウェーブを作成。

```
Function makeSampleScatterFromMatrix(inMat,numScatterPtsPerCol)
       Wave inMat
       Variable numScatterPtsPerCol
       Variable rows=DimSize(inMat,0)
       Variable cols=DimSize(inMat,1)
        if(cols<2 || rows<2)
                doAlert 0, "Bad dimensionality for inMat."
                return 0
       endif
       Variable outRows=cols*numScatterPtsPerCol,i,j,index,counter=0
       Make/O/N=(outRows,3) scatterTriplet
        for(i=0;i<cols;i+=1)</pre>
                for(j=0;j<numScatterPtsPerCol;j+=1)</pre>
                        index=trunc(abs(enoise(rows)))
                        scatterTriplet[counter][0]=DimOffset(inMat,0)+index*dimDelta(inMat,0)
                        scatterTriplet[counter][1]=DimOffset(inMat,1)+i*dimDelta(inMat,1)
                        scatterTriplet[counter][2]=inMat[index][i]
                        counter+=1
                endfor
        endfor
```

End