Tips – 3D 三角分割と補間についての考察

これは、三角分割と補間に関するブログポストを手順を中心に再編集したものです。
（ https://www.wavemetrics.com/news/3d-interpolation ）

グリッド化されたデータ

データが長方形のグリッド上でサンプリングされている場合、用途に応じて線形補間のさまざまな手法を利用できます。
以下の例では、次の同じデータセットを使います。

新しいエクスペリメントを作成したところからの手順で確認します。

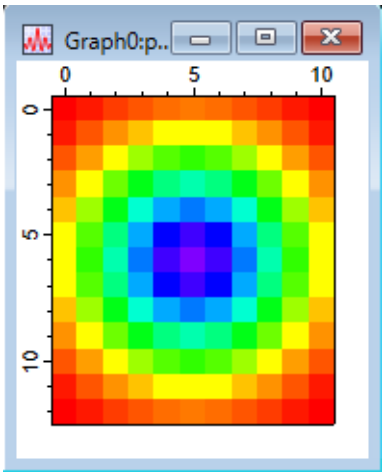
1. コマンドウィンドウで次を入力してデータセットを作成します。

```
Make/B/U/N=(11,13,15) ddd=gauss(x,5,3,y,6,3,z,7,3)*1e5
```

2. z=10 の位置で XY 平面をサンプリングし、画像として表示します。

```
MatrixOp p10=ddd[][][10]
NewImage p10
ModifyImage p10 ctab= {*,*,Rainbow,0}
```

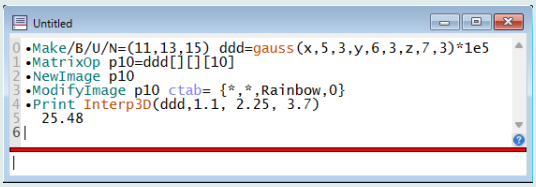
8 ビットの整数データという低解像度のサンプルであるため、画像が粗く表示されます。



グリッド化されたデータに対する補間の例

3. ddd で定義されたボリューム内の任意の XYZ 座標（例：(x=1.1, y=2.25, z=3.7)）について、１つの補間値を取得したい場合は、次のようにします。

```
Print Interp3D(ddd,1.1, 2.25, 3.7)

  25.48
```

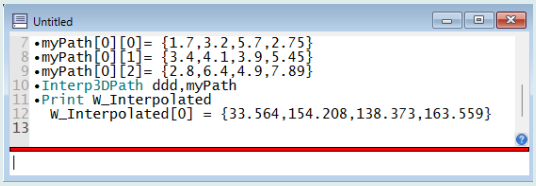


4. 長方形のグリッド上にサンプリングされていない一連のポイントを補間する必要がある場合は、それらの XYZ 座標をトリプレットウェーブに読み込み、Interp3DPath を使います。
例えば、コマンドウィンドウで４つの XYZ トリプレットを含むパスを作成します。

```
Make/N=(4,3) myPath
myPath[0][0]= {1.7,3.2,5.7,2.75}
myPath[0][1]= {3.4,4.1,3.9,5.45}
myPath[0][2]= {2.8,6.4,4.9,7.89}
```

5. 補間を実行して、結果を表示します。

```
Interp3DPath ddd,myPath
Print W_Interpolated

  W_Interpolated[0]= {33.564,154.208,138.373,163.559}
```



データを（100,150,200）の次元を持つ新しい 3D ウェーブに補間する必要がある場合は次の手順で行います。

6. コマンドウィンドウでウェーブを作成します。

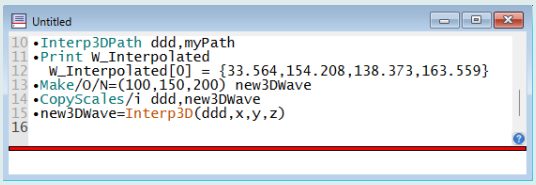
```
Make/O/N=(100,150,200) new3Dwave
```

ウェーブのスケーリングを、元の 3D ウェーブの範囲に合わせて設定します。

```
CopyScales/i ddd,new3Dwave
```

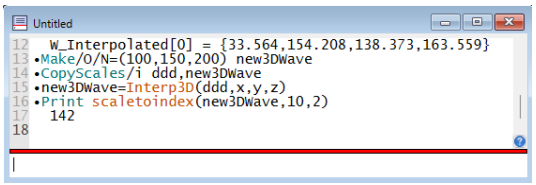
最後に、補間値を計算します。

```
new3DWave=Interp3D(ddd,x,y,z)
```



7. ddd[][][10] に対応する補間平面を求めるには、まず次を使って新しいインデックスを計算します。

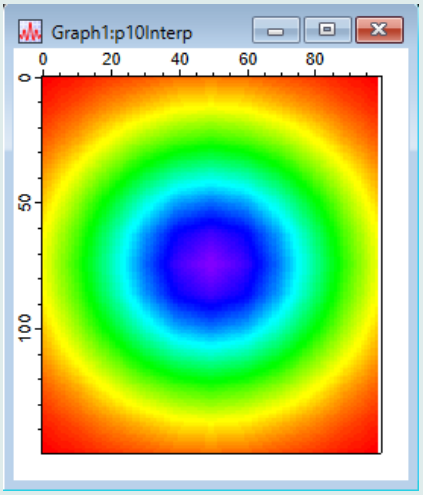
```
Print scaletoindex(new3DWave,10,2)

  142
```



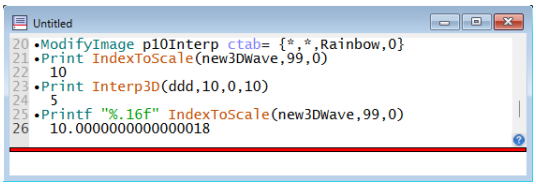
8. 続いて次を実行して画像を表示します。

```
MatrixOP/O p10Interp=new3DWave[][][142]
NewImage p10Interp
ModifyImage p10Interp ctab= {*,*,Rainbow,0}
```

オリジナルデータのレイヤー 10 を補間した new3DWave のレイヤー142 の画像が表示されます。



9. p10Interp の最後の行には NaN 値が含まれていることに注意してください（上の図では右端に白色で表示されています）。
通常、補間ルーチンは、補間ポイントがデータの定義域外にある場合、NaN を返します。
この場合、最後の行は 99 であり、これは x 値に対応します。




```
Print IndexToScale(new3DWave,99,0)

 10
```

しかし、x=10 はデータの定義域に含まれるべきです。
例えば、

```
Print Interp3D(ddd,10,0,10)

 5
```

代わりに次を実行してみると、NaN となる理由が明らかになります。

```
Printf "%.16f" IndexToScale(new3DWave,99,0)

 10.0000000000000018
```

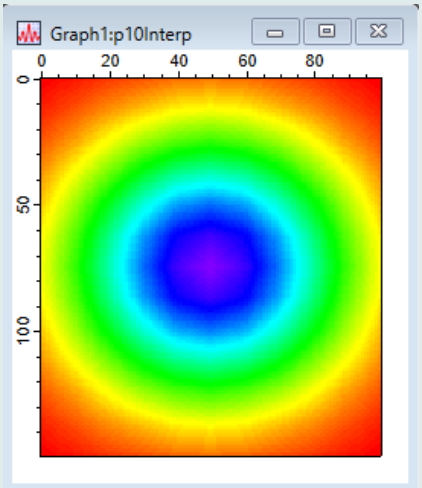
つまり、Igor Pro ではウェーブのスケーリングが「start」と「offset」という 2 つのパラメーターとして保存されているため、スケーリングをコピーすると start と offset が調整されますが、これらは浮動小数点数の丸め誤差の影響を受けます。
その結果、new3DWave の 99 行目が、補間範囲を超過してしまうほどわずかに大きな x 値に対応することになります。

10. この問題に対する現実的な解決策は、オフセットパラメーターの値を小さくすることです。
例えば、

```
SetScale/P x 0,0.1010101010101,"", new3DWave
```

を実行して、次に再度補間を行います。

```
new3DWave=Interp3D(ddd,x,y,z)
MatrixOP/O p10Interp=new3DWave[][][142]
```

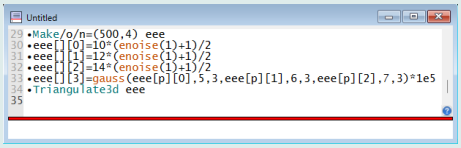


3D 散布データの補間

ここにおける散布データは、４列のウェーブデータとして表され、最初の３列にはデータがサンプリングされた XYZ 座標が、最後の列には各座標に関連付けられたスカラー値が含まれています。

3D 散布データの補間により、XYZ 座標の集合によって定義される凸包内の任意のポイントについて、連続的なスカラー値が得られます。
Igor Pro は、凸包外のポイントについては NaN を返します。

11. ここでは、前の例で使ったのと同じデータセットから、500 ポイントのランダムなサンプルを作成します。
コマンドウィンドウで次を実行します。

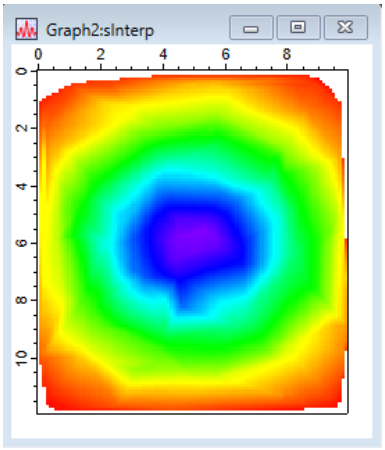



```
Make/o/n=(500,4) eee
eee[][0]=10*(enoise(1)+1)/2
eee[][1]=12*(enoise(1)+1)/2
eee[][2]=14*(enoise(1)+1)/2
eee[][3]=gauss(eee[p][0],5,3,eee[p][1],6,3,eee[p][2],7,3)*1e5
```

次に、データの 3D 三角分割を計算します。
この場合、ボリュームは三角形ではなく四面体に分割されます。

```
Triangulate3d eee
```

12. Triangulate3D のデフォルトの結果は、ウェーブ M_3dVertexList に格納されます。



z=10 に対応する平面のみを補間したい場合は、次のように実行します。

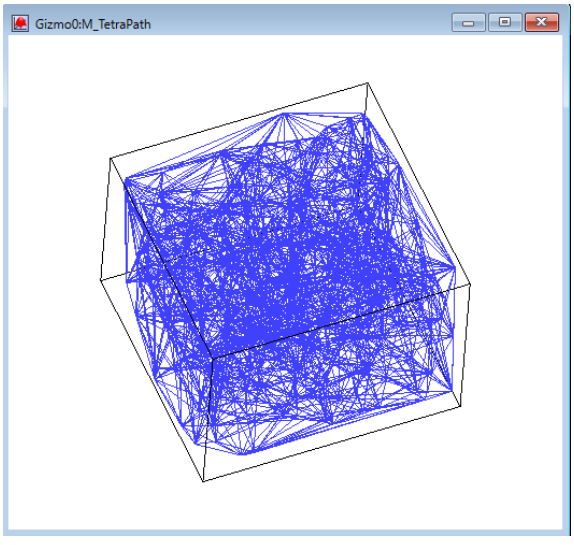
```
Interpolate3d/rngx={0,0.1,100}/rngy={0,0.1,120}
        /rngz={10,0.1,1} /Dest=sInterp
        triangulationwave=M_3dVertexList ,srcwave=eee
NewImage sInterp
ModifyImage sInterp ctab= {*,*,Rainbow256,0}
```

隅に NaN が含まれていることに注意してください。
これは、500 ポイントのランダムサンプリングに、この平面の端点が含まれていなかったことを示唆しています。
2D の場合と同様に、補間は、補間点が属する四面体を特定し、重心座標を用いて補間値を計算することで行われます。

計算の詳細を確認したい場合は、以下のコマンドを実行してください。

```
Triangulate3d/out=2 eee
```

そして、通常のパスオブジェクトを使用して、Gizmo に四面体を表示します。




```
NewGizmo
AppendToGizmo path=root:M_TetraPath,name=path0
ModifyGizmo ModifyObject=path0,objectType=path,
        property={pathColor,0.250019,0.250065,1,1}
AppendToGizmo Axes=BoxAxes,name=axes0
```

ここでは、ポイントの数（500 個）が多いため内部の詳細は確認しづらいですが、角の部分にデータが存在しないことは明らかです。