

Tips – ハンケル変換 (Hankel Transform)

目次

Tips – ハンケル変換 (Hankel Transform)	1
概要	1
例	2
精度に関する注意事項	3
他に含まれるもの	3
参考文献	3
HankelTransform 関数のヘルプ	4
HankelTransform (yw, nu[, xw=xw, kw=kw, numK=numK, kMax=kMax, dest=dest])	4

概要

オリジナルの投稿とプロシージャファイルの入手は下記です。

<https://www.wavemetrics.com/forum/general/hankel-transform-package-0>

ハンケル変換 (Hankel Transform) は、次数 nu のハンケル (フーリエ・ベッセル) 変換を計算します。

$$F(k) = f(r) J_{\text{nu}}(k r) r \, dr \text{ の } 0 \text{ から無限大までの積分}$$

ここで、 J_{nu} は第 1 種ベッセル関数です。

これは、円対称関数の 2 次元フーリエ変換の半径方向成分であるため、半径上にデータが存在するあらゆる場面で現れます。

具体的には、回折やビーム伝播、小角散乱、トモグラフィー、アベル型逆問題、そして半径対称な偏微分方程式 (PDE) の研究などが挙げられます。

この変換はそれ自体が逆変換でもあります。

すなわち、 $F(k)$ を同じ順序で同じルーチンに再入力することで、 $f(r)$ を復元することができます。

このパッケージは、純粋な Igor Pro プロシージャコードで構成されています。

インストールが必要な XOP はありません。

このプロシージャには、異なるアプローチを用いた変換の実装がいくつか含まれています。

主な変換関数：

```
Function/WAVE HankelTransform(WAVE yw, Variable nu, [WAVE xw, WAVE kw,  
Variable numK, Variable kMax, String dest])
```

- ・ 'yw' は 1D の実数入力関数 $f(r)$ です。'xw' が指定されていない場合、半径はウェーブの X 方向のスケールに基づいて算出されます。

- ・ 'nu' は順序を表します。基となる 'bessJ' は実数の順序を受け入れるため、'nu' は整数以外や負の値であっても構いません。
- ・ 'xw' は半径を明示的に指定するため、不均一にサンプリングされたデータも直接処理できます。
- ・ 'kw' は任意の出力ウェーブ数を指定します。
- ・ 'kw' が省略された場合、'numK' と 'kMax' によって出力グリッドが設定されます。'kMax' のデフォルト値は π/dr であり、これは入力サンプリングのナイキスト限界です。
- ・ 'dest' は出力ウェーブの名前を指定します。これを指定しない場合はフリーウェーブが返されるため、結果をウェーブ参照に代入してください。

出力グリッドが均一な場合、結果には k に対する X スケールが反映されるため、k を基準として直接プロットおよび評価されます。

積分は直接求積法によって評価されます。

台形重みは実際の半径座標から構築され、カーネル $J_{nu}(k_{ir_j})$ は $m \times n$ 行列として形成され、'MatrixOP' を使って適用されるため、時間とメモリの両方が $m \times n$ に比例します。

512×512 であれば実質的に瞬時に処理されますが、4096×4096 の場合、カーネル用に約 134 MB が必要となります。

被積分関数には r の因子が含まれているため、 $r=0$ では 0 になります。

したがって、入力範囲を f が遠端でも十分に減衰するように選択すれば、台形積分は極めて速く収束します。

滑らかで十分に減衰した入力の場合、結果は機械精度に近い精度で得られます。

解析的な 0 次のガウス関数対

$$\exp(-a r^2) \quad \leftrightarrow \quad \exp(-k^2/(4a))/(2a)$$

に対して、 $n=512$ のとき、順変換と往復変換の結果は、いずれも約 $1e-9$ の精度で一致します。

例

例えば、次のようなコードになります。

```
Variable a = 2, n = 512, rMax = 6
Make/O/D/N=(n) gaussIn
SetScale/P x, 0, rMax/(n-1), "", gaussIn
gaussIn = exp(-a*x^2)
WAVE ht = HankelTransform(gaussIn, 0, numK=n, kMax=20, dest="gaussOut")
Duplicate/O ht, gaussExact
gaussExact = exp(-x^2/(4*a))/(2*a)
```

これを実装したデモ用の関数 'DemoHankelTransform()' はまさにこのチェックを実行し、その結果を r 空間に往復変換して、両方のエラーの数値を履歴に出力します。

プロシージャの実装とデモの実行は次の手順で行います。

1. HankelTransform.ipf を User Procedures フォルダに配置します。
2. プロシージャウィンドウに次の行を追加し、Compile ボタンをクリックします。

```
#include "HankelTransform"
```

3. コマンドウィンドウで次を実行します。

```
DemoHankelTransform()
```

精度に関する注意事項

通常、切り捨て誤差（二乗誤差ではない）が主な誤差要因となります。
f が最後のポイントに達する前に十分に減衰するように入力範囲を選択してください。
これは、切り捨てられた末尾部分が答えに直接影響するためです。

デフォルトの 'kMax' を超える k を指定すると、入力のサンプリングデータには含まれていない詳細情報が要求されることになります。

不連続な入力は、 $O(dr)$ のオーダーでしか収束せず、リング状の特性を示します。
変換が $R J_{-1}(kR)/k$ となる円形のトップハット関数が、その典型的な例です。
このような関数に対しては、大幅にオーバーサンプリングを行うか、あるいは J_{-nu} の零ポイントでサンプリングを行う手法を使います。

他に含まれるもの

'HankelTransformFFT()' は、Siegman の準高速ハンケル変換（Opt. Lett. 1, 13 (1977)）を実装したもので、対数グリッド（r と k）上で $O(N \log N)$ の計算量を持つ手法です。

これは、 $O(mn)$ の直接ルーチンが処理速度として遅すぎる場合に有用ですが、対数グリッドに通常伴う注意点があります。

すなわち、 $r=0$ には到達できず、 $kMax/kMin$ は $rMax/rMin$ に強制的に等しくなり、位相ステップがサンプルあたり約 1 ラジアンを超えると、振動カーネルがエイリアシングを起こします。
その状況が発生した時には、ルーチンが警告を出します。

'HankelTransformFFTLog()' は、FFTLog（Talman 1978; Hamilton, MNRAS 312, 257 (2000)）の現在開発中の実装です。

これは参考用として、その診断ルーチンとともに含まれていますが、まだ検証が完了しておらず、信頼すべきではありません。

本番環境での作業には 'HankelTransform()' を使ってください。

参考文献

Bracewell, R.N., *"The Fourier Transform and Its Applications"*, 3rd ed., McGraw-Hill, 2000
第 13 章で、ハンケル変換と、円対称関数の 2 次元フーリエ変換との関係について解説しています。

Gradshteyn, I.S., and I.M. Ryzhik, "Table of Integrals, Series, and Products", 7th ed., Academic Press, 2007

第 6.5 節で、検証に役立つハンケル変換のペアを一覧表にまとめています。

Siegman, A.E., "Quasi fast Hankel transform", *Opt. Lett.* **1**, 13 (1977)

Hamilton, A.J.S., "Uncorrelated modes of the non-linear power spectrum", *MNRAS* **312**, 257 (2000).

HankelTransform 関数のヘルプ

HankelTransform.ifn ファイルを開くと HankelTransform 関数のヘルプを見ることができます。

HankelTransform(yw, nu[, xw=xw, kw=kw, numK=numK, kMax=kMax, dest=dest])

HankelTransform 関数は、入力ウェーブ yw の次数 nu のハンケル変換を含むウェーブを返します。これは次のように定義されます。

$$F_n(k) = \int_0^{\infty} f(r) J_n(kr) r dr$$

ここで、Jn は次数 n の第 1 種ベッセル関数です。

半径座標 r は、xw が指定されていない限り、yw の X 軸スケーリングから算出されます。

この関数はウェーブ参照を返します。

dest が指定されない場合、結果はフリーウェーブとなるため、使用するには戻り値をウェーブ参照に代入する必要があります。

パラメーター

yw	f(r) を含む 1D の実数ウェーブです。このウェーブには少なくとも 2 つのポイントが含まれていなければなりません。
nu	変換の順序を表します。基となる bessJ 関数は実数の順序を受け入れるため、nu は整数以外や負の値であっても構いません。
xw	均一にサンプリングされていないデータ用の、オプションの半径ウェーブです。これは yw と同じポイント数を持つ必要があり、非負であり、かつ昇順でなければなりません。xw を省略した場合、半径は pnt2x を使って yw の X スケールから取得されます。
kw	出力の横軸のウェーブを表すオプションです。これを指定すると、numK と kMax の設定が上書きされ、出力値の間隔は任意に設定できます。この場合、出力グリッドは不均一となるため、結果には X 軸のスケーリングは適用されません。
numK	出力ウェーブのデータポイント数です。デフォルトは numpnts(yw) です。値は 2 以上である必要があります。kw が指定された場合は無視されます。
kMax	出力における最大のウェーブ数です。デフォルト値は pi/dr です。ここで、dr は入力の平均半径間隔であり、これは入力のサンプリングにおけるナイキスト限界に相当します。kw が指定された場合は無視されます。
dest	出力ウェーブに付けるオプションの名前です。このウェーブは現在のデータフォルダー内に作成され、同名の既存のウェーブがある場合は上書きされます。dest を省略した場合は、代わりにフリーウェーブが返されます。

詳細

積分は直接求積法によって評価されます。

半径座標から台形重み w_j が形成され、変換は次のように計算されます。

$$F(k_i) = \sum_j w_j r_j f(r_j) J_n(k_i r_j)$$

カーネルは $m \times n$ の行列として構築され、MatrixOp によって適用されます。

ここで、 n は入力ポイントの数、 m は出力ポイントの数です。

したがって、実行時間とメモリ使用量はともに $m \times n$ に比例します。

512×512 の変換は実質的に瞬時に行われますが、4096×4096 の場合、カーネル用に約 134 MB が必要です。

重みは、均一であると仮定するのではなく、実際の半径方向の間隔に基づいて計算されるため、 xw を通した不均一にサンプリングされたデータも正しく処理されます。

被積分関数には r の因子が含まれているため、 $r=0$ のポイントで 0 になります。

$rMax$ を、 f が遠端でも 0 まで減衰するように選択すれば、被積分関数は両端で 0 になり、台形法による求積は極めて速く収束します。

滑らかで十分に減衰した入力の場合、結果は機械精度に近い精度で得られます。

出力グリッドが均一な場合、返されるウェーブの X 軸のスケーリングは k の値に設定されるため、結果を k を軸として直接プロットしたり評価したりすることができます。

以下の条件ではエラーが発生します。

yw のポイントが 2 ポイント未満	"input needs at least 2 points"
$numpts(xw)$ と $numpts(yw)$ が一致しない	"x wave length must match the y wave"
半径が負の値	"radii must be non-negative"
半径が増えていない	"radii must be increasing"
$numK$ が 2 未満	"numK must be at least 2"

精度

通常、主な誤差の原因は二乗誤差ではなく、切り捨て誤差です。

f が最後のデータポイントに達する前に確実に減衰するように入力範囲を選択すること。切り捨てられた末尾の部分は、結果に直接影響を与えます。

デフォルトの $kMax$ を超える k を指定すると、入力サンプリングデータに含まれていない詳細情報が要求されることになります。

π/dr のデフォルト値は、データがサポートする最大のウェーブ数です。

不連続な入力は $O(dr)$ のオーダーでしか収束せず、リングング現象を示します。

標準的な例として、 $r < R$ のときは $f(r)=1$ 、それ以外は 0 となる円形のトップハット関数があり、その変換は $R J_1(kR)/k$ となります。

このような関数に対しては、大幅にオーバーサンプリングを行うか、 J_{nu} のゼロポイントでサンプリングを行う手法をつかいます。

例

ガウス関数を変換し、解析的な結果と比較します。

0 次のペア $\exp(-a r^2)$ は、 $\exp(-k^2/4a)/(2a)$ に変換されます。

プロシージャウィンドウで次を入力します。

```
Function HankelTest1()  
Variable a = 2, n = 512, rMax = 6
```

```

Make/O/D/N=(n) gaussIn
SetScale/P x, 0, rMax/(n-1), "", gaussIn
gaussIn = exp(-a*x^2)

WAVE ht = HankelTransform(gaussIn, 0, numK=n, kMax=20, dest="gaussOut")

Duplicate/O ht, gaussExact
gaussExact = exp(-x^2/(4*a))/(2*a)

Variable mn, mx
Duplicate/O ht, gaussErr
gaussErr = ht - gaussExact
[mn, mx] = WaveMinAndMax(gaussErr)
printf "max |error| = %.3e¥r", max(abs(mn), abs(mx))

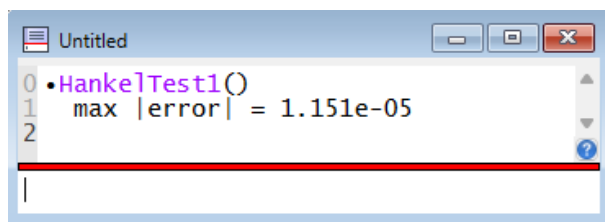
End

```

コマンドウィンドウで

```
HankelTest1()
```

を実行します。

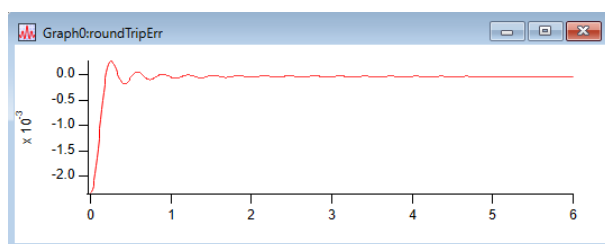


結果に対してもう一度変換を適用することで、その変換がそれ自身の逆変換であることを確認します。プロシージャウィンドウの HankelTest1() の最後に次を追加します。

```

WAVE back = HankelTransform(ht, 0, numK=n, kMax=rMax)
Duplicate/O back, roundTripErr
roundTripErr = back - gaussIn
Display/K=0 roundTripErr

```



以降も定義した関数内で使います。

不均一な放射状グリッド上でサンプリングされたデータを変換し、半径を明示的に指定します。

```
WAVE ht = HankelTransform(yData, 0, xw=rData)
```

呼び出し元が指定した出カウェーブ数範囲に対して、フリーウェーブを返します。

```

Make/O/D kPoints = {0.1, 0.2, 0.5, 1, 2, 5}
WAVE ht = HankelTransform(gaussIn, 1, kw=kPoints)

```

参照

コマンド bessJ, Besselj, Integrate, areaXY, MatrixOp, FFT, Convolve

参考文献

Bracewell, R.N., The Fourier Transform and Its Applications, 3rd ed., McGraw-Hill, 2000. Chapter 13
ハンケル変換と、円対称関数の2次元フーリエ変換との関係について解説しています。

Gradshteyn, I.S., and I.M. Ryzhik, Table of Integrals, Series, and Products, 7th ed., Academic Press, 2007.
第6.5節で、検証に役立つハンケル変換のペアを一覧表にまとめています。