MATLAB 瞬時周波数 Toolbox*

○矢田部浩平, 升山義紀, 草野翼, 及川靖広 (早大理工)

1 まえがき

音響信号処理の多くは，時間周波数領域における手続きとして実現されている．従来は振幅のみに基づく処理が主流であったが，近年では位相に対する処理の重要性が認識され，スペクトログラムの位相に関する様々な研究が盛んに行われている．

かつてない位相ブームが世界的に訪れている中，2019 年 3 月に発行された日本音響学会誌 75 巻 3 号では「位相情報を考慮した音声音響信号処理」と題した小特集が組まれ，スペクトログラムの位相と瞬時周波数に関して我々も執筆を行った [1]．小特集記事のうち 3 件については Acoustical Science and Technology の 40 巻 3 号に英語版も掲載され，我々の記事の英語版 [2] 出版に合わせて MATLAB コードを公開した [3] (`https://doi.org/10/c3qb`)．このコードは，読みやすいように特にドキュメンテーションに力をいれて頑張って執筆し，個人的にはそれなりに良く書けたと思っているが，説明文が全て英語なので敷居が高いと感じられることも承知している．

そこで本稿では，公開した MATLAB コード一式の説明書を日本語で残すことを目的とし，具体的なコードを用いてスペクトログラムや瞬時周波数の計算方法を説明する．また，解説記事 [1] の本題である，瞬時周波数を用いた位相修正 [4–6] を計算する方法についても，具体例を用いて説明する．

2 ツールボックスの内容

ここでは，公開したコード (`https://doi.org/10/c3qb`) の内容について説明する．ただし，(本稿の題目は申し込み時に適当につけたもので) このコード自体に名前は付けていないので，これ以降コード一式を指して「ツールボックス」と呼ぶことにする．

2.1 ツールボックスで計算できること

本ツールボックスには以下の機能が実装されている．

1. 実数信号からスペクトログラムへの変換
2. スペクトログラムから実数信号への逆変換
3. 完全再構成のための合成窓関数の計算
4. 瞬時周波数計算のための微分窓関数の計算
5. 各ビンにおける瞬時周波数の計算
6. 瞬時周波数を用いた位相修正の計算

すなわち，「時間領域と時間周波数領域の相互変換」「瞬時周波数の計算と利用」「それらに必要な窓関数の計算」が簡単に実行できるツールボックスである．コードの可読性および数式との対応を重視し，(例えば入力は複素信号に対応せず実数信号のみなど) 機能を限定することでシンプルで簡潔な実装を目指した．

2.2 コード一覧

本ツールボックスは以下のファイルを含んでいる．

2.2.1 デモンストレーション用コード

各関数の `help` を読まずとも実行できるように，関数の使用例を示すデモコードが入っている．中身をコピーすれば各機能を実装できるように書かれている．

- `demo1_DGTusage.m` 時間領域から時間周波数領域に変換し，時間領域に逆変換するデモ
- `demo2_windowUsage.m` 完全再構成のための窓関数を計算し，再構成できることを確認するデモ
- `demo3_IFandIPC.m` 瞬時周波数を計算し，それを用いて位相修正を行い，時間領域に逆変換するデモ
- `demo_FDGTusage.m` 時間領域と時間周波数領域の相互変換の計算時間を短縮するデモ

2.2.2 窓関数計算用コード

様々な窓関数を簡単に呼び出して利用できるように，窓関数コレクションと計算用コードが入っている．

- `generalizedCosWin.m` 一般化コサイン窓 (Hann, Blackman, Nuttall 等 19 種類) とその微分窓の生成
- `calcCanonicalDualWindow.m` 完全再構成のための双対窓関数 (dual window) の計算
- `calcCanonicalTightWindow.m` 完全再構成のためのタイトな窓関数 (tight window) の計算
- `numericalDiffWin.m` 上記の一般化コサイン窓以外の窓関数に対する微分窓の数値的な計算

2.2.3 時間周波数変換 (離散 Gabor 変換) 用コード

時間領域と時間周波数領域の相互変換用コードが入っている．ただし，文献 [1] と用語を合わせるために，短時間 Fourier 変換と呼ぶかわりに，ここでは離散 Gabor 変換の頭文字を取って DGT と呼んでいる．

- `DGT.m` 実信号からスペクトログラムへの変換
- `invDGT.m` スペクトログラムから実信号への逆変換
- `zeroPaddingForDGT.m` DGT に必要な前処理

*MATLAB toolbox for calculating instantaneous frequency. By Kohei YATABE, Yoshiki MASUYAMA, Tsubasa KUSANO and Yasuhiro OIKAWA (Waseda University).

2.2.4 瞬時周波数と位相修正計算用コード

微分窓を用いた瞬時周波数計算用のコードと，それを用いた位相修正の計算用コードが入っている．

- `calcInstFreq.m` 瞬時周波数の計算
- `instPhaseCorrection.m` 位相修正の計算
- `invInstPhaseCorrection.m` 位相修正の逆変換

2.2.5 時間周波数変換の計算時間短縮用コード

反復計算のために，可読性を犠牲にして (MATLAB の範囲内での) 高速化を施したコードが入っている．

- `FDGT.m` 上記 `DGT.m` の高速版 (Faster DGT)
- `invFDGT.m` 上記 `invDGT.m` の高速版
- `precomputationForFDGT.m` 高速化用の事前計算

2.3 ツールボックスの特徴

他の時間周波数解析ツールと異なる点として，本ツールボックスは以下のような特徴を有している．

まず，スペクトログラムの計算において，位相の定義が四種類実装されている．文献 [1] で述べたように，スペクトログラムの位相は時間原点の選び方によって異なり，どの定義で計算するかによって大幅に性質が変わる．それらをフラグで切り替えられるようになっており，全種類を手軽に試せるコードとなっている．また，位相の定義は時間領域のインデックス操作のみで実装され，定義を変えても計算精度が落ちないように工夫されている．さらに，数学的に適切な周期境界条件で計算されるようになっており，そのための前処理用関数も実装している．

窓関数まわりには草野によるこだわりが反映され，19 種類ものコサイン級数窓が実装されている．河原英紀先生の 5-/6-term 窓や Albrecht の 7-/11-term 窓も含んでおり，解析的な微分窓も一緒に計算される．また，広く使われる双対窓 (最適合成窓) のみでなく，処理に頑健なタイト窓の計算も含み，さらに任意の窓に対する微分窓の近似計算も実装されている．

瞬時周波数の計算は，位相の定義に応じて相対瞬時周波数と (絶対) 瞬時周波数が両方実装されている．また，位相修正とその逆変換も実装されている．

3 使用方法

デモコードに沿って各関数の使用方法を説明する．なお，本ツールボックスを利用するには，MATLAB 9.1 (2016b) 以降のバージョンが必要である．

3.1 スペクトログラムの計算

時間領域の実数信号 `sig` をスペクトログラム `spec` に変換するには，関数 `DGT` を用いる．窓関数 `win`，窓のシフト幅 `shiftLen`，FFT 長 `fftLen` を定義し，

```
spec = DGT(sig,win,shiftLen,fftLen);
```

と書けばスペクトログラムが計算される．ただし，信号と窓関数は縦ベクトル，シフト幅と FFT 長は自然数であり，`fftLen` は窓長以上でなければならない[1]．窓は任意だが，後述する `generalizedCosWin` を用いれば様々な窓を簡単に生成できる．

数学的に適切な取り扱いのために，`DGT` は周期境界条件で実装されている．文献 [1] の **2.3** で述べたように，そのためには信号の拡張が必要な場合がある．本ツールボックスでは，周期境界のための事前計算を

```
sig = zeroPaddingForDGT(sig,shiftLen,fftLen);
```

によって自動的に実行でき，周期境界条件を満たす場合は何も実行されない．`DGT` 実行前にこの計算を行わないと `DGT` 内部でエラーを返すことがある．

スペクトログラムから時間信号への逆変換も同様に

```
sig = invDGT(spec,win,shiftLen,fftLen);
```

で計算できる．ただし，`DGT` と `invDGT` で適切な窓の組み合わせを用いないと，信号は完全再構成されない．後述のように，そのための「適切な窓」を自動的に生成する関数も用意されている．

3.2 スペクトログラムの位相の定義の選択

`DGT` の第五・第六引数には，位相の定義を変更するフラグを入力できる (デフォルトは両方とも `false`)．

第五引数 `rotateFlag` は，`DGT` の定義自体を変更する．`rotateFlag = true` の場合，信号と複素正弦波が時間原点を共有し，窓のみがシフトするように計算される．対して，`rotateFlag = false` の場合，窓と複素正弦波が時間原点を共有し，窓のみでなく複素正弦波もシフトする．詳細は文献 [1, 2] を参照[2]．

第六引数 `zeroPhaseFlag` は，窓関数の定義を変更する．`zeroPhaseFlag = false` の場合，窓の第一要素 `win(1)` を時刻ゼロとして扱い，その他の要素を正の時刻とみなす．対して，`zeroPhaseFlag = true` の場合，窓の中心 `win(floor(length(win)/2)+1)` を時刻ゼロとして扱い，窓の前半を負の時刻，後半を正の時刻とみなす．これは，窓を `ifftshift` することと同じであり，周波数領域がシフトされるのと同様に時間領域もシフトすることで，窓に起因する線形位相成分を除去できる (文献 [1, 2] の図 2 を参照)．

[1] この制約は，コードを簡潔にするためにそれ以外の場合を実装していないだけであり，一般には窓長未満の FFT 長も設定可能で，完全再構成可能にもできる．`DGT` の第四引数は省略可能で，その場合は FFT 長と窓長が同じとして扱われる．

[2] 文献 [1] と [2] で数式番号が異なることに注意しながら，数式と `rotateFlag` との対応を述べる．文献 [1] の数式番号で言えば，`true` が式 (3)，`false` が式 (13) に対応する．対して，文献 [2] と対応させれば，`true` が Eq. (7)，`false` が Eq. (22) を表す．どちらの定義で計算しても，位相が変わるのみで振幅は共通である．

3.3 窓関数の生成

窓長 wLen と窓の種類 (文字列) wName を定義して

```
[win,dWin] = generalizedCosWin(wLen,wName);
```

と書けば，コサイン級数窓を生成することができる．ここで，戻り値 win は生成された窓 (縦ベクトル) であり，それを解析的に微分した dWin も同時に出力される[3]．窓関数としては表–1 に示した 19 種類が収録されており，それらの特性を知りたい場合は

```
generalizedCosWin(wLen,wName);
```

のように戻り値を省略すれば，生成した窓と周波数特性がプロットされる．一般に，微分可能階数が多いほど，すなわち表–1 で C^m の m が大きいほど，サイドローブの減衰が速い．一方，微分可能階数が少ないほど設計の自由度が高いので，ピークサイドローブレベルを下げたり，メインローブ幅を狭めることを目標にすると，不連続な窓の方が有利な場合がある．

これら窓関数の文献調査および実装は草野による．第二引数には文字列 wName の代わりに，コサイン級数の係数を低次から高次の順に並べた係数ベクトルを直接入力しても良い．すなわち，この関数に収録されていない窓でも，コサイン級数窓であれば同様に窓関数とその微分を生成することができる．

3.4 窓関数に対する計算

窓関数に対して計算を行う関数が 3 種類入っている．

1 つ目は窓関数を数値的 (近似的) に微分する関数で，

```
dWin = numericalDiffWin(win);
```

によって任意の窓を微分することができる．その際，巡回畳み込みの影響を変化させるゼロ埋めの量を第二引数として入力することができる[5]．この関数はコサイン級数窓以外の窓を微分することを想定しており，コサイン級数窓を用いる場合は generalizedCosWin によって解析的に微分する方が好ましいことが多い．

2 つ目は，与えられた窓に対する標準双対窓 (canonical dual window) を計算する関数で，窓とシフト幅を

表–1 generalizedCosWin に収録されている窓関数一覧．ただし，左の列は第二引数に入力される wName を表し，右の列の「n-term」は n 項コサイン級数であることを，「C^m」は m 階微分が連続関数であることを表している．また，C^m の記述がない窓は不連続関数である．

rect	矩形窓 (1-term)
hann	Hann 窓 (2-term, C^1)
hamming	Hamming 窓 (2-term)
blackman	Blackman 窓 (3-term, C^1)
exactBlackman	厳密な Blackman 窓 (3-term)
blackmanHarris	Blackman–Harris 窓 (4-term)
blackmanHarris3term	Blackman–Harris 窓 (3-term)
nuttall3term	Nuttall 窓 (3-term)
nuttall3termC1	Nuttall 窓 (3-term, C^1)
nuttall3termC3	Nuttall 窓 (3-term, C^3)
nuttall4term	Nuttall 窓 (4-term)
nuttall4termC1	Nuttall 窓 (4-term, C^1)
nuttall4termC3	Nuttall 窓 (4-term, C^3)
nuttall4termC5	Nuttall 窓 (4-term, C^5)
flattop	フラットトップ窓 (5-term)
kawahara5term	河原らの窓 (5-term, C^5) [7][4]
kawahara6term	河原らの窓 (6-term, C^7) [7][4]
albrecht7term	Albrecht 窓 (7-term) [8]
albrecht11term	Albrecht 窓 (11-term) [8]

```
dualWin = calcCanonicalDualWindow(win,shiftLen);
```

のように入力すれば計算できる．標準双対窓は最適合成窓とも呼ばれ，DGT と invDGT が信号を完全再構成するために必要な窓である[6]．win を用いて DGT した後に dualWin を用いて invDGT すれば，元の時間信号が復元される．逆に，dualWin を用いて DGT した後に win を用いて invDGT しても復元可能である．具体的な例は demo2_windowUsage.m に示してある．

3 つ目は，標準タイト窓 (canonical tight window) を計算する関数で，標準双対窓の計算と同様に

```
tightWin = calcCanonicalTightWindow(win,shiftLen);
```

と書けば計算できる．タイトな窓とは自分自身が双対な窓であり，すなわち tightWin を用いて DGT した後に tightWin を用いて invDGT すれば，元の時間信号が復元される．タイトな窓は，時間周波数領域での処理の誤差に頑健であることが知られている [9].

3.5 瞬時周波数の計算

瞬時周波数 IF は，窓 win を用いて DGT したスペクトログラム spec と，微分窓 dWin で DGT した dSpec を

```
IF = calcInstFreq(spec,dSpec,fftLen,wLen);
```

のように入力することで計算できる．デフォルトでは rotateFlag = true に対応した相対瞬時周波数が

[3]「解析的に微分」とは，コサイン級数窓の微分を，コサインを微分した結果のサイン級数によって計算することを指す．ただし，端点の微分はゼロとしているので，元の窓が不連続関数の場合は帯域制限された微分の巡回畳み込みによる誤差が生じるが，不連続のギャップが小さい場合は数値的に問題ないことも多い．

[4]河原らの 5-term 窓と 6-term 窓は，設計段階ではそれぞれ C^5 と C^7 になるように制約されているので [7]，表–1 でも C^5 と C^7 で表記したが，実際は係数を 10 桁に丸めた際に高階の微分可能性が失われている．その意味でどちらも C^1 であると言うのが正しいが，単精度浮動小数点数の意味では (つまり有効桁を 5〜6 桁とすれば) 丸め誤差の範囲内でそれぞれ C^5 と C^7 だとは言える．

[5]numericalDiffWin はスペクトル法を採用しており，微分フィルタが巡回畳み込みされる．第二引数に自然数を入力すると，その分だけゼロ埋めをしてから数値微分を行い，その後に付け加えたゼロを削除する．ゼロ埋めによる窓長の変化を保持したい場合は，第一引数に入力する窓自体を予めゼロ埋めしておく必要がある．

[6]DGT が冗長であれば，ある窓に対する双対窓は無数に存在する．その中で最もパワーの小さい窓が「標準」双対窓である．

出力され，正弦波成分が周波数ビン何個分ズレているかがわかる[7]．第五引数には rotateFlag を入力でき，絶対瞬時周波数を計算するには false を入力すれば良い[8]．また第六引数には，ゼロ割を防ぐための正のフロアリング係数を入力でき，小さい値を入力するほど理論的な数式に忠実になるが，数値誤差の影響を受けやすくなる．入力を省略した際のデフォルトの値には，音声信号などの特徴を考慮し，パワースペクトログラムの最大値の 10^{-10} 倍を設定している．

3.6 位相修正と逆変換の計算

信号の瞬時周波数を利用し，位相スペクトログラムの時間的変動を打ち消す方法を「位相修正」と我々は呼んでいる [1–6]．本ツールボックスでは，計算したスペクトログラム spec と瞬時周波数 IF を入力して

```
iPCspec = instPhaseCorrection(spec,IF,shiftLen,fftLen);
```

とするだけで，瞬時位相修正済 (iPC: instantaneous-phase-corrected) スペクトログラム iPCspec を計算することができる．瞬時周波数は各時間周波数ビンにおける局所的な位相の情報なので，関数の内部で数値積分することで瞬時位相に変換してから，複素スペクトログラムの位相をその分だけ回している．位相修正の具体例は，demo3_IFandIPC.m を実行することで図として表示することができる．

文献 [1, 2] で述べられているように，位相修正は簡単に逆変換することができる．順変換と同様に

```
spec = invInstPhaseCorrection(iPCspec,IF,shiftLen,fftLen);
```

と書けば，位相修正を逆変換することができる．丸め誤差の範囲内で逆変換できるので，invDGT によって時間領域信号を復元することができる．

3.7 スペクトログラムの計算の高速化

本ツールボックスの DGT と invDGT は，コード簡略化のためにインデックスを多用している．具体的には，信号の周期化，信号の短時間セグメントへの分割，位相の定義変更，窓関数の線形位相除去，そして重畳加算法を全てインデックス操作によって実装している．コードを読みやすくするために，このインデックスを DGT と invDGT の内部で毎回生成するように書いており，Griffin–Lim 法などの反復計算においては無駄が多い．そこで，事前計算可能な情報を先に計算することで高速化を施した FDGT と invFDGT を実装している．事前計算は，信号長を sigLen として

```
precomputationForFDGT(sigLen,wLen,shiftLen,fftLen);
```

によって実行され，5 つの戻り値を FDGT と invFDGT に入力することで無駄な計算を省いている．具体的な使用例は demo_FDGTusage.m に記載されているが，実行速度を計るコードも書かれているので，DGT との比較も簡単にできるようになっている．

3.8 DGT のインデックスに関する補足

MATLAB バージョン 9.1 (2016b) 以降では，暗黙的な拡張 (implicit expansion) によって「縦ベクトルと横ベクトルの和」を計算することができる．縦ベクトル x と横ベクトル y の積と似ていて，各 (n,m) に対して A(n,m) = x(n) + y(m) のように行列を計算結果として出力する．ここで，縦ベクトル x を窓のインデックス，横ベクトル y を時間インデックスとすれば，時間インデックス y の各要素を始点とした窓のインデックス全てを x + y のみで一度に生成することができる．すなわち，信号を短時間セグメントに分割するためのインデックスを生成できる．周期的な巡回は mod によって実装できるので，for 文を用いずとも DGT を実装可能である．位相の定義の違いは，DGT の中にある sigIdx や rotIdx の中身と，代入結果を具体的に見てみるとわかりやすいかもしれない．

[7] 瞬時周波数の単位が「周波数ビンの相対インデックス」となるように計算している．例えば相対瞬時周波数が「2」であれば，そのビンには「2 つ上のビン」の正弦波成分が入っていることを表す．このことを具体的な例で確認するには，demo3_IFandIPC.m を実行し，figure, imagesc(IF), caxis(10*[-1 1]) などで瞬時周波数を可視化して，振幅スペクトログラムと比較するとよい．

[8] 「絶対」瞬時周波数を計算した場合は，「何番目のビンの正弦波成分が入っているか」を表すインデックスが返ってくる．「ビンのインデックス」と「実際の周波数」の対応が取れるようにサンプリング周波数を用いて計算すれば，単位を周波数に変換できる．

参考文献

[1] 矢田部浩平, 升山義紀, 草野翼, 及川靖広, “位相変換による複素スペクトログラムの表現,” 日本音響学会誌, **75**(3), 147–155 (2019).

[2] K. Yatabe, Y. Masuyama, T. Kusano and Y. Oikawa, “Representation of complex spectrogram via phase conversion,” *Acoust. Sci. & Tech.*, **40**(3), 170–177 (2019).

[3] K. Yatabe, T. Kusano and Y. Masuyama, “Phase correction of complex spectrogram based on instantaneous frequency [Source Code],” *Code Ocean*, (2019). https://doi.org/10.24433/CO.2743732.v1

[4] K. Yatabe and Y. Oikawa, “Phase corrected total variation for audio signals,” *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, pp. 656–660 (2018).

[5] Y. Masuyama, K. Yatabe and Y. Oikawa, “Low-rankness of complex-valued spectrogram and its application to phase-aware audio processing,” *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, pp. 855–859 (2019).

[6] Y. Masuyama, K. Yatabe and Y. Oikawa, “Phase-aware harmonic/percussive source separation via convex optimization,” *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, pp. 985–989 (2019).

[7] H. Kawahara, K. Sakakibara, M. Morise, H. Banno, T. Toda and T. Irino, “A new cosine series antialiasing function and its application to aliasing-free glottal source models for speech and singing synthesis,” *Proc. INTERSPEECH*, pp. 1358–1362 (2017).

[8] H. Albrecht, “A family of cosine-sum windows for high-resolution measurements,” *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, pp. 3081–3084 (2001).

[9] T. Kusano, Y. Masuyama, K. Yatabe and Y. Oikawa, “Designing nearly tight window for improving time-frequency masking,” *Int. Congr. Acoust. (ICA)*, (2019).