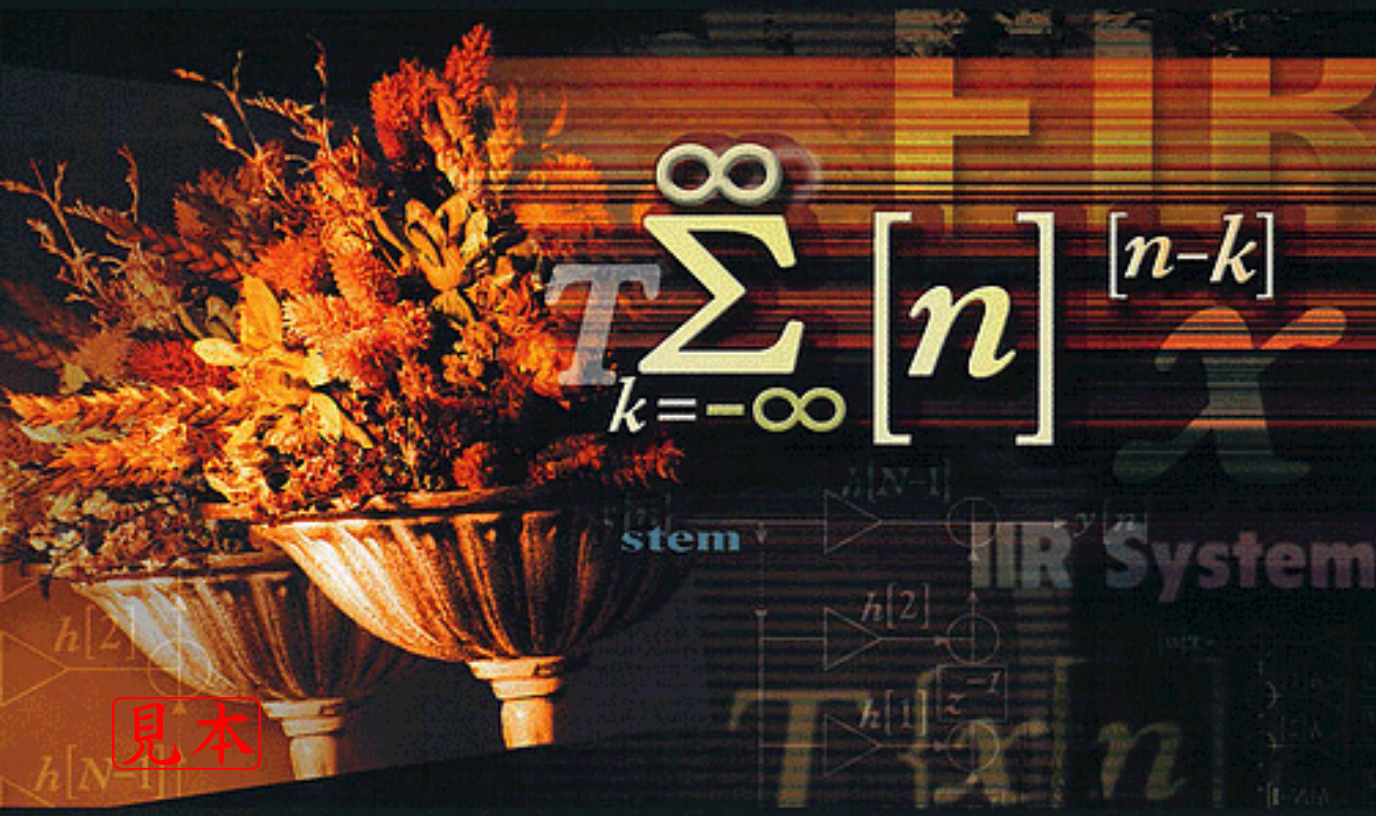
ディジタル信号処理シリーズ

ツールを使った実践的アプリケーション開発

MATLABによる画像&映像信号処理

●村松 正吾 著



CQ出版社

前書き

2011年，日本におけるアナログ・テレビ放送が地上波，BS共に打ち切られる．60年近い歴史が幕を閉じ，代わってディジタル・テレビ放送が主役の座を奪う．ディジタル方式の本格的普及は，CSディジタル放送が開始され，DVDが登場した90年代中旬に始まった．このディジタル化の波は，2000年のBSディジタル放送，2003年の地上デジタル放送の開始によって一気に加速した．これと並行して，写真の世界では大手カメラ・メーカがフィルム・カメラ事業の縮小や新規開発の中止を表明している．アナログ映像情報メディア最後の砦（とりで）といっても良い映画の世界でも，撮影，編集，配給，上映のすべてをディジタルで行うディジタル・シネマが準備を整え，普及を待つばかりである．

ビデオ・カメラの世界では電子ビーム系の時代は終わり，CCDやCMOSなどの固体素子デバイスが世を席巻している．表示機器についてもブラウン管（CRT）はその役割を終え，薄型化に有利なプラズマ・ディスプレイ・パネル（PDP）や液晶ディスプレイ（LCD）が世界の市場をにぎわせ，し烈な争いを繰り広げている．新たな技術はいずれも画素単位で映像を制御するため，ディジタル方式になじみやすい．

HD-DVDやBlu-ray Diskの登場，ハード・ディスクやフラッシュROMの大容量化，ブロードバンド/ワイヤレス・ネットワークの普及は，ディジタル画像・映像情報の発展と相互に作用している．まさに，ディジタル隆盛の時代といえる．一方で，劣化が少なく複製や加工が容易であるというディジタル方式の利点がコンテンツ制作・配給者を悩ます一面もある．著作権やセキュリティの問題である．これらの対策も，重要な課題として活発に研究開発が進められている．

従来，放送やパッケージ・メディアなどの1対多の関係にある応用が画像・映像信号処理技術の発展をけん引してきた．次世代においては，多対1，多対多の関係にあるビデオ・センサ・ネットワークが新しい技術の創造を要求することは間違いない．医療福祉，防災減災，監視や環境モニタリングなど，その応用にも期待が膨らむ．

このように話題に事欠かない状況の中，画像・映像信号処理に携わる研究開発技術者の社会的な役割はますます重要となり，多くの人材が必要とされている．幅広い技術要求に対応するため，各人が基礎知識を身につけ，適切な処理を直感的に設計・選択できる能力を備えることが望まれる．

本書では，ディジタル画像・映像信号処理を学ぶ上で最低限必要な線形システムと周波数解析の理解に重点を置いた．第7章までは，初学者が身に付けるべき基礎知識を簡単な例題と共に解説した．対象とする読者は，高専生や学部学生を主としている．第8章以降は，1次元と多次元の違いを理論と直感で学べるように配慮した．多次元のフィルタ設計やマルチレート信号処理など，高度な内容も含まれる．対象とする読者は，画像・映像信号処理に携わる大学院生や若手研究開発者を主としている．

本文中，解説があいまいにならないようになるべく数式を添えた．さらに，直感的な解釈が可能なようにMATLABプログラムとその実行例を豊富に掲載した．MATLABは初学者でも高度なプログラムを簡単に実現可能とする．反面，知識がなくても結果が得られてしまう恐れがある．そこで，

筆者から読者へお願いがある．本書を読み進める際，結果に対する考察を行い，MATLABを自らの理解を確認するツールとして利用してほしい．

執筆は慎重に行い，推敲（すいこう）も重ねて行った．しかし，筆者の未熟さゆえに至らない点が多々あると思われる．今後のためにも，ご意見やご批判をいただければ幸いである．本書によって多くの読者が画像・映像信号処理に関する基礎知識を共有し，本書が技術の発展に寄与することができれば，筆者にとってはこの上ない喜びである．

なお，本書を執筆するにあたり多くの方々にご協力とご支援をいただいた．まず，本書執筆の機会を与えて下さった九州工業大学の尾知博教授に感謝したい．当初は共著の予定であったが，残念ながらスケジュールの都合で実現に至らなかった．次に，首都大学東京の貴家仁志教授に感謝したい．本書の執筆において多くの励ましの言葉をいただいた．また，新潟大学の菊池久和教授に感謝したい．日頃よりいただいている知見は本書の内容にも反映されている．本書の推敲やプログラムの動作確認に協力してくれた渡邉勤氏，安達充幸氏をはじめ，新潟大学村松研究室所属の大学院生，学部生にも感謝したい．さらに本書執筆中，ジョージア工科大学のJim McClellan教授と2度も夕食を共にする好機に恵まれ，大きな励みとなった．

度重なる構成の見直しや不慣れな対応にも関わらずご辛抱いただき，本書の出版まで導いてくださったCQ出版社Interface編集長の中山俊一氏，毎日コミュニケーションズ（元CQ出版社Interface編集部）の大竹友美氏に感謝したい．最後に，本書執筆を暖かく見守ってくれた友人と家族に感謝したい．

2007年3月29日　村松正吾

目　次

目　次

目　次

第1章

画像・映像信号処理とは

画像・映像情報メディアのディジタル化は，高品質，高効率，高機能，高セキュリティなどの利益をもたらす．これらの技術を実現し，発展させていくためには，画像・映像信号処理についての理解は欠かせない．本章では，本書全体を概観するために，画像・映像信号処理の概要と本書のねらいについて述べる．第2章以降で具体的に解説を進める．なお，本書において「画像」は静止画像を，「映像」は動画像を示す言葉として使用する．

1.1　画像・映像信号処理の基礎

画像・映像信号処理とは，「画像・映像情報を伝送，記録，解析，加工する技術」である．その種類も圧縮，強調，改善，再構成，記述，認識など，多岐にわたる．

● 画像・映像信号処理の流れ

図1.1 に画像・映像信号処理の流れを示す．まず，撮像機器のセンサによって光の分布が電気信号

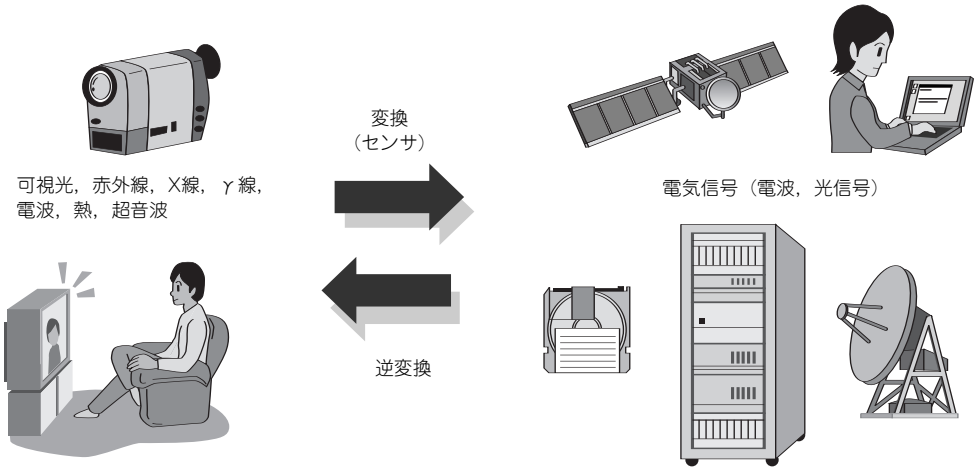


図1.1
画像・映像信号処理の流れ

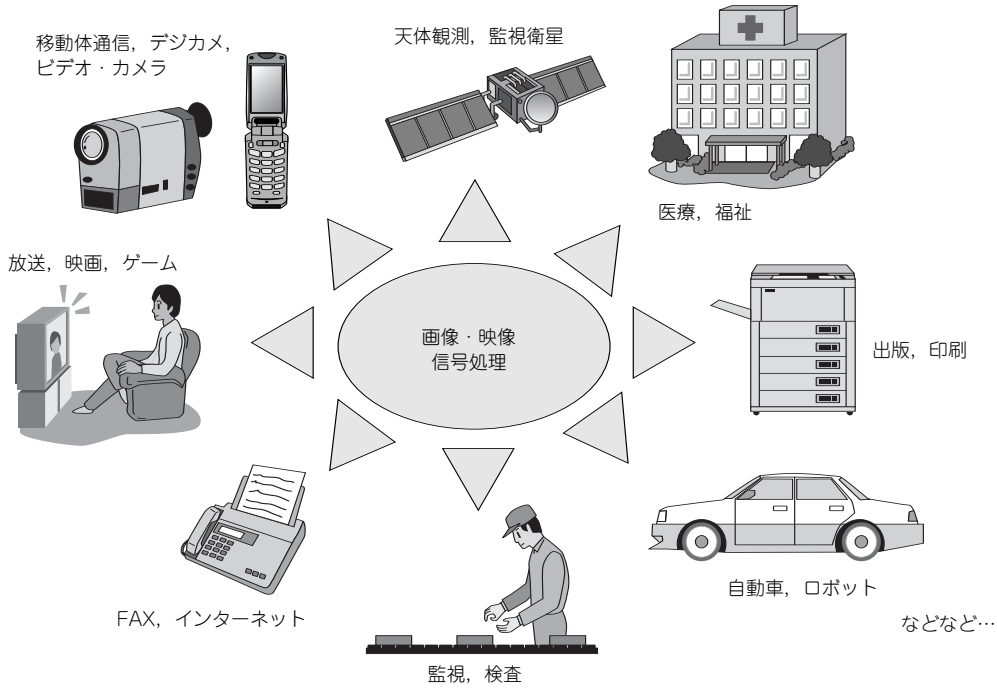


図1.2　画像・映像信号処理の応用例

に変換される．いったん電気信号に変換されれば，その光の分布の様子を伝送，記録，解析，加工することは容易である．さらに標本化・量子化を施せばディジタル信号となり，コンピュータ上での操作が可能となる．高品質・高効率な記録・伝送のほか，暗号化などによりセキュリティを高めることもできる．表示機器によって信号を再び光の分布に戻すことで，原画像・原映像を異なる場所，異なる時刻に再現できる．

なお，画像・映像信号は可視光に限らない．変換するセンサさえ存在すれば，紫外線やX線など，あらゆる物理量の分布が信号処理の対象となる．また，信号が水平方向・垂直方向を軸とする分布や配列として与えられれば「画像処理」，水平方向・垂直方向・時間を軸とする分布や配列として与えられれば「映像処理」とみなせる．

● 画像・映像信号処理の応用例

画像・映像信号処理の応用例は多岐にわたる．**図1.2**に一部の例を掲載する．いずれも画像や映像を扱う応用だが，目的が異なれば撮像の条件や要求される表示の形態も異なる．従って，要求される処理は用途に応じておのずと変わる．

ある用途で優れた方法は，別の用途では役に立たないこともある．複雑な処理を施すよりも，単純な処理の方が視覚的に優れた性能を示すこともある．人間が良いと感じるか否かは，単純な問題ではない．これが画像・映像信号処理の難しい部分であり，また，経験や発想を生かせる楽しい部分でもある．

● 画像・映像信号の入出力

なぜ，用途に応じて画像・映像信号処理への要求に違いが起こるのか．画像・映像の入出力信号に着目して，その例を示そう．

(1) 撮像機器――入力部

撮像機器は光などの分布を電気信号に変換する．すなわち，画像・映像と処理システムの接点にあたる．現在，主流の撮像機器は，CCD(Charge Coupled Device) やCMOS(Complementary Metal Oxide Semiconductor) などの固体素子を採用し，従来の撮像管による撮像機器と比べて，小型化・軽量化されている．

放送や映画，出版などのプロ向け撮像機器では，高価な光学系を構成し，CCD/CMOSセンサも複数利用して，色や解像度などが高品位な画像・映像を取得する．三板式CCD/CMOSセンサ・カメラがこれにあたる．

一方，安価な民生用カメラや携帯電話付属カメラは，CCD/CMOSセンサ1枚のみで構成される場合がほとんどである．空間解像度を犠牲にし，色情報をカラー・フィルタを通して取得するため，1枚のカラー画像を得るために「デモザイキング」と呼ばれる補間処理が必要となる．

監視カメラや車載カメラは，色情報や高解像度を必要としない場合もある．劣悪な環境にカメラが置かれていれば，ノイズ除去や強調処理が必要となるだろう．固定カメラか可動カメラかの違いによっても動きに関する処理が変わるだろう．

身体の断層画像を直接撮影することはできない．CT スキャナは身体の周りをX 線や核磁気共鳴を利用してスキャンする．その情報から断層画像を得るために，再構成処理が必要となる．

(2) 表示機器――出力部

表示機器は画像・映像信号と人間の接点にあたる．現在，映像の表示機器として電子ビーム系のCRT(Cathode Ray Tube) に変わり，液晶ディスプレイ(LCD：Liquid Crystal Display) やプラズマ・ディスプレイ(PDP：Plasma Display Panel)などのフラット・パネル系が勢力を増している．

放送では，インターレース(飛び越し)走査と呼ばれる映像フォーマットが採用されている．垂直時間解像度を保ちつつ1チャネルの伝送に必要となる帯域を半分にするという技術である．放送がディジタル化されてもインターレース走査は健在である．インターレース走査では，空間の標本位置がフィールド周期という時間間隔でズレている．インターレース走査は，画素に空間的広がりや残像のある電子ビーム系のCRTに比べて，発光を画素単位で制御するLCDやPDPにおいて視覚上の問題を生じる．空間的な標本位置のズレを解消するためのインターレース-プログレッシブ(IP)変換(デインターレース処理)という補間技術が必要となる(**図1.3**)．

画像の印刷についても，レーザ記録，静電記録，感熱記録，インクジェット記録など，手法は多岐にわたる．白と黒の2階調[注1-1]，あるいは限られた色の組み合わせで中間の明るさや微妙な色を表現しなければならない．インクかトナーかの違いも考慮しなければならない．視覚的劣化を抑えつつ限られた階調や色へ変換するハーフ・トーニングという技術が必要となる．

注1-1：階調数は，表現できる明るさの種類を示す．

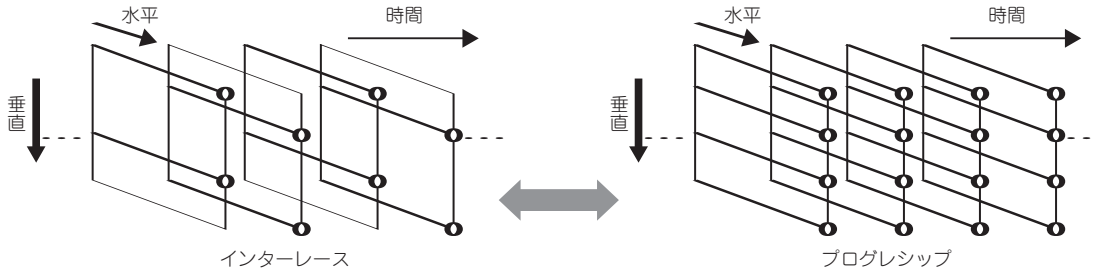


図1.3　インターレース-プログレッシブ変換

最後に，画像・映像信号処理の結果が画像・映像であるとは限らないことにも注意しておこう．特徴を表す数値や内容を示す判別結果が出力となることもある．指紋画像や顔画像による人物認証の結果は，個人を特定する名前やIDで十分である．

1.2　本書のねらい

学ぶべき画像・映像信号処理は用途ごとに変わるのではないか――確かにそうである．しかし，汎用的に役立つ基礎知識や広く利用される要素技術が存在する．本書では，そのような画像・映像信号処理全般において重要と思われる，

- 画素処理
- フィルタリング
- 周波数解析
- 標本化とその変換

について，直感的に理解できることを目標にして解説した．そのため，数値シミュレータMATLABのプログラム例とシミュレーション結果を豊富に掲載した．また，学習に役立つように多くの実習や演習課題も準備した．ぜひとも取り組んで，じっくりと深く理解してもらいたい．なお，**図1.4**に本書の読み進め方の例を示しておく．

● 本書の構成

第2章では，MATLABに慣れていない読者のためにMATLABの基本的な操作について解説した．従って，MATLABの利用経験がある読者は読み飛ばしても差し支えない．

第3章では，画像・映像信号処理について学ぶための準備として，1次元信号処理について解説した．復習という位置付けで，基礎的な内容については詳細の説明を省き，結論を中心にその概要を述べた．MATLABを利用した信号処理シミュレーションの導入も兼ねているので，第4章以降の理解に役立てていただきたい．

第4章から画像・映像信号処理の本格的な解説が始まる．第4章では，MATLABで画像・映像を取り扱うための留意点や機能を解説した．また，ヒストグラム操作のような画素処理や色空間変換といった話題にも触れている．

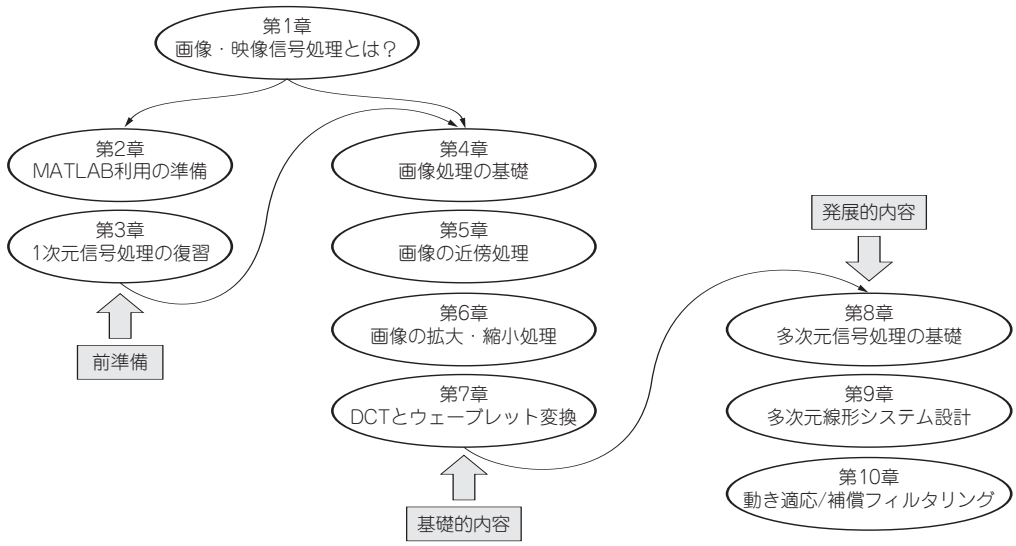


図1.4　本書の読み進め方

第5章では，画像の近傍処理や映像のフレーム間処理について解説した．近傍処理はマスク処理，あるいはフィルタリングとも呼ばれ，もっとも基本的な画像・映像処理技術である．この章では，画像処理の分野で典型的に利用される技法とそれらの特徴について解説した．

第6章では，画像の拡大・縮小について解説した．映像のフレーム・レート変換の例題についても掲載した．マルチレート信号処理の導入も兼ねており，拡大・縮小処理の周波数領域での振る舞いやシステムの設計法，効果的実現法についても解説した．

第7章では，離散コサイン変換(DCT：Discrete Cosine Transform）とウェーブレット変換について解説した．DCTとウェーブレット変換は，符号化をはじめ，画像・映像信号処理において欠かせない要素技術となっている．本章では，第6章で紹介するマルチレート信号処理を用いて，DCTとウェーブレット変換をフィルタリングの観点からも統一的に解説した．

画像・映像信号処理は多次元信号処理である．第8章以降では，1次元信号処理にはない多次元特有の問題について扱った．多くの読者にとっては，第7章までの内容で十分だろう．ただし，面白くなるのはここからである．第8章は，多次元信号処理の基礎的事項についてまとめている．特に，変数のベクトル表記を導入し，周波数解析と標本化について1次元信号処理との類似点と相違点をまとめている．

第9章は「多次元線形システム設計」と題して，主に非分離型のフィルタ設計法を紹介している．各手法についてじっくりと理解するというよりも，多次元フィルタを設計したいとき，リファレンスとして利用すると良いだろう．後半では，非分離型のマルチレート信号処理についても紹介した．これは発展的内容といえよう．

第10章は，映像信号処理における動き情報の重要性をインターレース-プログレッシブ(IP)変換を題材に解説している．固定係数，動き適応処理，動き補償処理という順序でシミュレーションを行

い，性能の向上について確認できるように構成している．

● 表記法

本書では次のような表記を利用する．

- 丸かっこ：連続変数．$x(t)$，$X(j\Omega)$，$X(z)$など
- 四角かっこ：離散変数．$x[n]$，$X[k]$など
- 太字の小文字：ベクトル．$\boldsymbol{x}$，$\boldsymbol{a}$など
- 太字の大文字：配列もしくは行列．$\boldsymbol{X}$，$\boldsymbol{A}$など

● MATLABの環境

本書に掲載したMATLAB のソース・コードは，次の環境で動作確認を行った．

- MATLAB R2006a
- Signal Processing Toolbox
- Image Processing Toolbox

本書に掲載するソース・コードは，基本的にMATLAB本体とSignal Processing Toolboxのみで動作する．ただし，Image Processing Toolboxがある場合についてのコメントも記載し，必要に応じてソース・コードも併記した．

例題で紹介したソース・コードはすべて実習でも活用できるように，`practiceXX_X.m`というファイル名で用意してある．`XX_X`の部分は章番号と各章における実習番号となっている．一部，Image Processing Toolboxの機能を用いた`practiceXX_X_ip.m`も合わせて配布している．以下のWebサイト，

`http://www.cqpub.co.jp/haibai/books/39/39xxx.htm`

よりダウンロードして実際に試してほしい．

参考文献

(1) 吹抜 敬彦；画像・メディア工学，コロナ社，2002年．
(2) Rafael C. Gonzalez and Richard E. Woods；Digital Image Processing, 2nd Ed.，Prentice Hall，2001.
(3) 今村 元一；ビデオ信号の基礎とその操作法，CQ出版社，2003年．
(4) Interface編集部 編；ディジタル放送の基礎技術入門，TECH Iシリーズ，CQ出版社，2002年．
(5) Bahadir K. Gunturk, John Glotzbach, Yucel Altunbasak, Ronald W. Schafer, and Russel M. Mersereau；Demosaicking: Color Filter Array Interpolation，IEEE Signal Proc. Magazine，Vol.44，Jan. 2005.
(6) Zhaoui Sun；Video Halftoning，IEEE Trans. on Image Proc.，Vol.15，No.3，pp.678-686，Mar. 2006.
(7) Gerard De Haan, and Erwin B. Bellers；Deinterlacing . An Overview，Proc. of IEEE，Vol.86，No.9，pp.1839-1857，Sep. 1998.

第2章

MATLAB利用の準備

本書ではMATLABのプログラム例を多数掲載し，実際に実行することで，画像・映像信号処理に関する理解を深めてもらうことを目的としている．まず準備として，MATLAB の基本操作について概説したい．本書に掲載するプログラム例を理解して確かめるのに十分な機能のみ，解説する．従って，インストール方法や詳細な利用法については，MATLABのマニュアルなどを参照されたい．なお，MATLABを実行する環境がなくても，ソース・コードを読み解くことができれば，ほかのプログラミング言語による学習も可能であろう．すでにMATLAB利用の経験がある読者は，本章を読み飛ばし，第3章以降に進んでも差し支えない．

2.1　基本操作

以下では，MATLABバージョンR2006aの利用を前提として解説を進める．

● 立ち上げ方と終わり方

(1) 立ち上げ方

Windows系のOSを使用している場合，デスクトップ上にあるアイコン(**図2.1**)をダブルクリックすればよい．すると，**図2.2**のような画面が現れ，すぐに利用できる状態になる．Unix系のOS を使用している場合は，ターミナル上で，

```
matlab
```

と入力すればよい．実際は，使用環境により立ち上げ方が異なることもあるため，使用するコンピュータの管理者に問い合わせるのが良いだろう．

図2.1
MATLABのアイコン

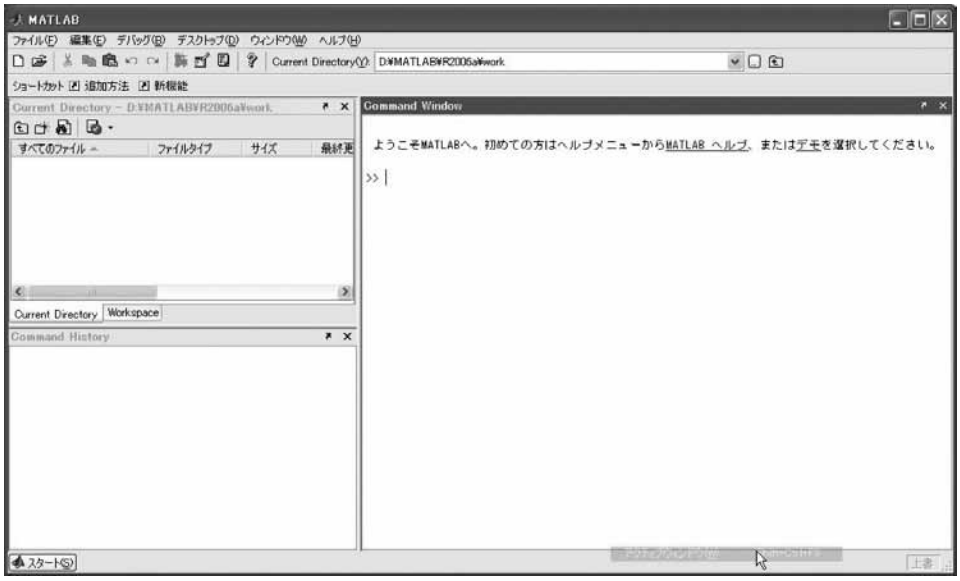


図2.2　起動直後の画面

(2) 終わり方

MATLABのCommand Window上(**図2.2**では右側)のプロンプト記号>>に続いて

```
>> exit
```

と入力することで，MATLABを終了することができる．

● オンライン・ヘルプ

MATLABには豊富な数の関数があらかじめ用意されている．そこで，各関数の利用法の理解にはオンライン・ヘルプ機能は欠かせない．オンライン・ヘルプ機能の使い方について解説を進めよう．

MATLABを終了していたら，立ち上げ直そう．まず，`fft`関数が何を行う関数なのかを調べてみよう．

Command Window上で，

```
>> help fft
```

を実行する．すると，**図2.3**のような表示が現れる．

ヘルプ文が長く，画面をはみ出してしまう場合には，あらかじめ`more on`と設定しておくと便利である．すると，ヘルプ文をはじめとしたCommand Window上の出力結果が1画面分の長さに区切られて表示される．

```
>> more on
>> help fft
```

続きを表示する際には，スペース・キーなどを押せばよい．`more`の機能を停止したければ，

```
>> more off
```

とすればよい．

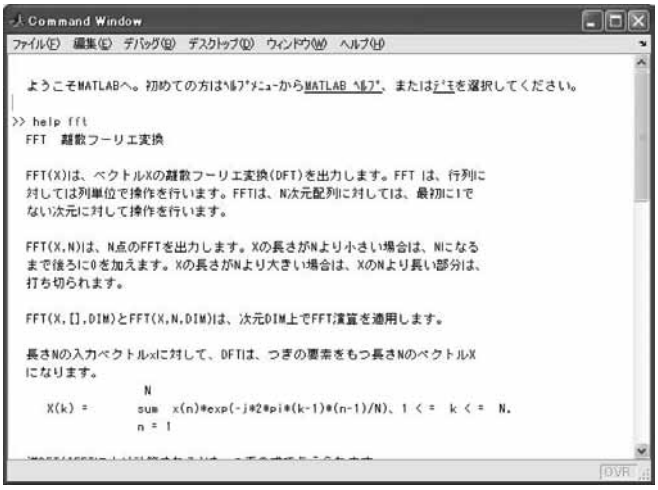


図2.3
オンライン・ヘルプの使用例

例題2.1　conv関数

conv関数は何を行う関数か？

解

conv関数は，畳み込み演算や多項式の積演算を行う関数である．

例題2.2　help .

help .(ヘルプ・ドット)というコマンドを実行すると，何が表示されるか？

解

help . を実行すると，演算子や特殊記号の説明が表示される．

例題2.3　help i

help i(ヘルプ・アイ)というコマンドを実行すると，何が表示されるか？

解

help i を実行すると，複素数の説明が表示される．help j(ヘルプ・ジェイ)，help pi(ヘルプ・ピーアイ)も確かめてみよう．

関数をはじめとするMATLABの操作方法がわからなくなったら，このオンライン・ヘルプ機能を利用すると便利である．より詳細な説明が必要になったら，docコマンドを利用すればよい．例えば，Command Window上で，

```
>> doc fft
```

を実行すると，**図2.4**に示すようなオンライン・ドキュメントを閲覧することができる．

● 行列やベクトルの扱い

MATLABの大きな利点の一つに，行列操作が簡便であることが挙げられる．MATLABにおいて

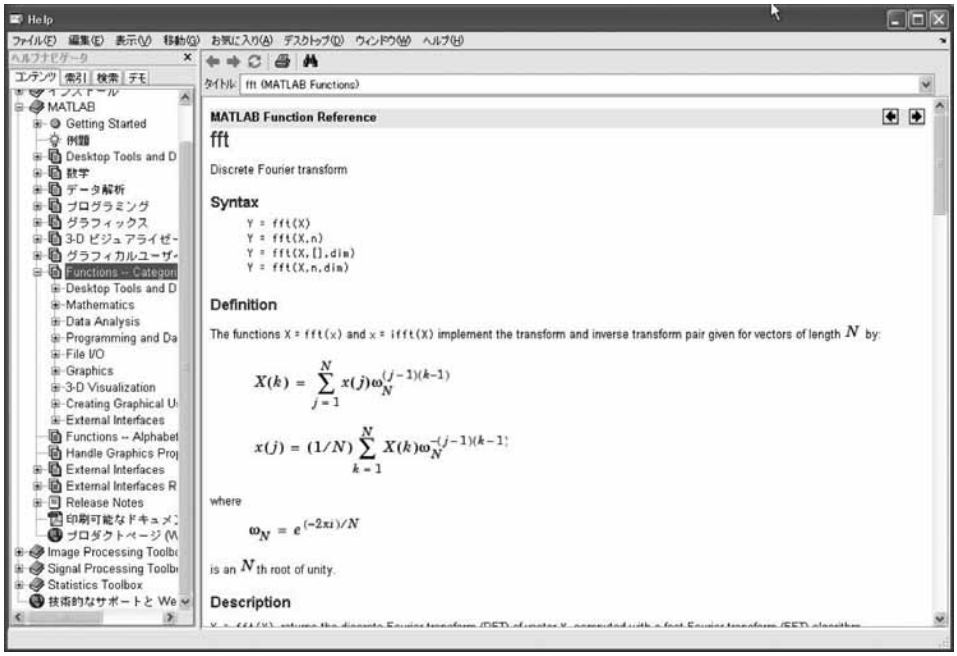


図2.4　オンライン・ドキュメントの使用例

行列やベクトルを扱う方法について解説しよう．

まず，

$$\mathbf{A}=\begin{pmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{pmatrix} \quad \cdots\cdots (2.1)$$

$$\mathbf{B}=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{pmatrix} \quad \cdots\cdots (2.2)$$

という行列の積$\mathbf{C}=\mathbf{AB}$を求めることを例として，解説を進める．

行列$\mathbf{A}$，$\mathbf{B}$の定義は，Command Window上で

```
>> A = [ 1 2 ;
         3 4 ;
         5 6 ];
>> B = [ 1 2 3 ;
         4 5 6 ];
```

と入力する．すなわち，行方向は要素間をスペースで区切り，列方向は各行ベクトル間を;(セミコロン)で区切って並べる．

また，行列積$\mathbf{C}=\mathbf{AB}$は，

```
>> C = A * B;
```

表2.1　行列(あるいはベクトル)に対する基本操作

コマンド	内　容
X+Y, X-Y, X*Y	加算，減算，乗算
X.*Y, X./Y	要素ごとの乗算，要素ごとの除算
kron(X,Y)	クロネッカー・テンソル積
X.', X'	転置，エルミート転置
fliplr(X), flipud(X), rot90(X)	行反転，列反転，90度回転
X(:)	列ベクトル化
X(:,iCol), X(iRow,:)	iCol番目列ベクトル，iRow番目行ベクトル
X(:,iCol:jCol)	[X(:,iCol), X(:,iCol+1), ..., X(:,jCol)]
X(iRow:jRow)	[X(iRow,:); X(iRow+1,:); ...; X(jRow,:)]

と実行する．すると，Cに演算結果が代入される．Command Window上で

```
>> C
```

のように末尾に;(セミコロン)を付けずに変数名Cを入力することで，

```
C =
     9 12 15
    19 26 33
    29 40 51
```

のように表示され，行列**C** の内容を確かめることができる．

ベクトルは列数が1，あるいは行数が1の行列と考えられるので，同じように操作できる．

行列(あるいはベクトル)X，Yに対する基本的な演算操作を**表2.1**にまとめる．

例題2.4　行列転置

行列**B**の転置と行列**A**の転置の積**D**を計算してみよう．

解

Command Window上で

```
>> D = B.' * A.';
>> D
```

と入力すると，

```
D =
     9 19 29
    12 26 40
    15 33 51
```

のように表示される．

例題2.5　行列要素の選択

行列**A**の上の2本の行ベクトルからなる正方行列と，行列**B**の左にある2本の列ベクトルからなる正方行列との和**E**を計算してみよう．

解

Command Window 上で

```
>> E = A(1:2,:) + B(:,1:2);
>> E
```

と入力すると，

```
E =
     2  4
     7  9
```

と表示される．

なお，MATLABでは行列要素のインデックスは1から始まることに注意されたい．

2.2 ループ処理と条件分岐処理

MATLABでは，ループ処理のためにfor文とwhile文が利用できる．また，条件分岐のためにif文とswitch文が利用できる．for文とif文の使用法を，簡単な例を用いて解説しよう．

● ループ処理

まず，for文を用いたループ処理の例を見てみよう．例えば，

$$s=\sum_{n=0}^{10} n \quad \cdots\cdots (2.3)$$

という演算は，Command Window 上で

```
>> s = 0; % s の初期値設定
>> for n = 0:10 % n を0 から10 まで繰り返す
     s = s + n; % s にn の値を累積加算
   end
```

と実行できる．

```
>> s
```

と入力すると

```
s =
    55
```

と表示される．for文は，forの直後に記述された変数の仕様に従い，forとendの間の処理を繰り返す．for文，while文の詳細については，help for，help whileを参照されたい．なお，%から行末まではコメント文となり，処理には影響しない．

例題2.6　for文

m行n列目の要素が，

$$[\mathbf{C}]_{k,n} = \cos\left(\frac{k(2n+1)\pi}{8}\right), \quad k,n = 0,1,2,3 \quad \cdots\cdots (2.A)$$

と与えられる4×4行列**C**を生成しよう．cos関数，定数piを用いてもよい．ただし，MATLABでは行列のインデックスが1から始まることに注意されたい．

解

Command Window上で

```
>> for iCol=1:4
       n = iCol - 1;
       for iRow=1:4
           k = iRow - 1;
           C(iRow,iCol) = ...
               cos(k*(2*n+1)*pi/8);
       end
   end
```

と実行すればよい．

```
>> C
```

と入力すると，

```
C =
    1.0000    1.0000    1.0000    1.0000
    0.9239    0.3827   -0.3827   -0.9239
    0.7071   -0.7071   -0.7071    0.7071
    0.3827   -0.9239    0.9239   -0.3827
```

と表示される．

なお，複数行にまたがるコマンドは，...の後に改行しなければならない．

● 条件分岐処理

次に，if文を用いた条件分岐処理について解説しよう．例えば，$n = 0$ならば$d = 1$，$n \neq 0$ならば$d = 0$，それ以外ではdは値をとらないという処理

$$d = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \\ \phi & \text{その他} \end{cases} \quad \cdots\cdots (2.4)$$

は，あらかじめ変数nに値を代入した上で，

```
>> if n==0 % n が0 ならば
       d = 1; % d は1.
   elseif n~=0 % n が0 ではなければ，
       d = 0; % d は0.
```

```
    else % それ以外では，
      d = []; % d は空．
    end
```

のように実行できる（例えばあらかじめn=[]が設定されていれば，dは空となる）．if文はifの直後にある条件式に従い，条件分岐を行う．条件が満たされた場合，ifに続く処理が実行され，条件が満たされなかった場合，else以下の処理が実行される．elseifは，条件付きのelse文である．if文の詳細についてはhelp ifを参照されたい．また，条件式として使用される比較演算子，論理演算子については，help relopを参照されたい．

例題2.7　条件分岐

例題2.6で与えられた4×4行列**C**の1行目の行ベクトルに係数1/2を乗じ，残りの行ベクトルに係数$1/\sqrt{2}$を乗じて，4×4行列**D**を生成してみよう．sqrt関数を用いてよい．

解

例題2.6の解答に示したコマンドの実行直後に

```
>> for iRow=1:4
     k = iRow - 1;
     if k==0
       D(1,:) = C(1,:)/2;
     else
       D(iRow,:) = ...
         C(iRow,:)/sqrt(2);
     end
   end
```

と実行すればよい．

```
>> D
```

と入力すると，

```
D =
   0.5000   0.5000   0.5000   0.5000
   0.6533   0.2706  -0.2706  -0.6533
   0.5000  -0.5000  -0.5000   0.5000
   0.2706  -0.6533   0.6533  -0.2706
```

と表示される．なお，この行列**D**は，第7章で詳しく解説する離散コサイン変換（DCT：Discrete Cosine Transform）行列と呼ばれる直交行列である．

if文のほかにもswitch文によって処理の分岐を実現できる．詳細についてはhelp switchを参照されたい．

2.3　グラフ表示

MATLABには，便利なグラフ表示機能が多数用意されている．以下では，信号の周波数特性表示を例として，グラフ表示に使うstem関数とplot関数の使い方を解説しよう．なお，周波数特性については次章においてあらためて解説を行うので，ここではグラフ表示の一例として読み進めてもらいたい．

● stem関数

まず準備として，rand関数を使って適当な8×1ベクトルxを生成しよう．Command Window上で

```
>> x = rand(8,1);
```

と入力すればよい．直後に

```
>> x
```

と入力すると，例えば

```
x =
    0.9501
    0.2311
    0.6068
    0.4860
    0.8913
    0.7621
    0.4565
    0.0185
```

のように表示される．rand関数は一様分布の疑似乱数を生成する関数なので，実際には上記のものと異なることがある．

では，ベクトルxのstem表示を行ってみよう．Command Window上で

```
>> stem(x)
```

と入力すると，**図2.5**のような表示が得られる．stem表示は，離散時間信号(数列)を表すのに便利な機能である．

● plot 関数

では次に，plot関数を利用してベクトルxの周波数振幅特性を見てみよう．Command Window上で

```
>> A = 20*log10(abs(fft(x,512)));
>> plot(A)
```

と入力すると，**図2.6**のような表示が得られる．1行目では，xの振幅特性([dB])を512点FFT(Fast Fourier Transform)を用いて求めている．2行目が実際にグラフ表示を行っている部分である．plot表示はデータ間を直線で補間してグラフを表示する．オプションの指定によって，点のみの表示も

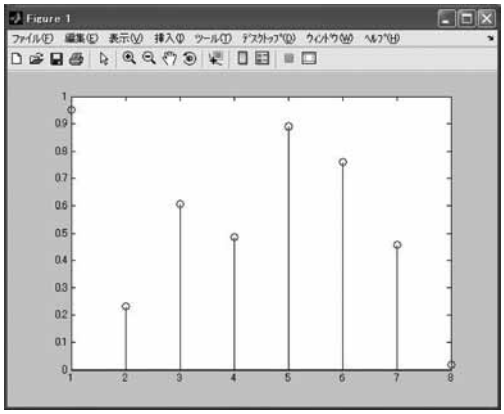


図2.5　stem 表示の例

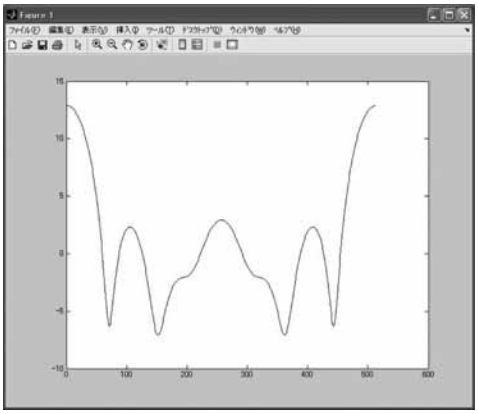


図2.6　plot 表示の例

可能である．詳細については`help plot`を参照されたい．

例題2.8　**plot表示**

同じサイズのベクトル変数x，yを用いて`plot(x,y)`を実行すると，*x*-*y*プロット表示がなされる．変数tを0.0から6.3まで0.1刻みで変えながら，関数$f(t) = \cos(t)$を*x*-*y*プロットしてみよう．

解

Command Window上で

```
>> for i = 0:63
       t(i+1) = i/10;
       f(i+1) = cos(t(i+1));
   end
>> plot(t,f)
```

もしくは，

```
>> t = 0.0:0.1:6.3;
>> f = cos(t);
>> plot(t,f)
```

と入力すればよい．

後者の解について解説しよう．

- 1行目は，0.0 ～6.3まで0.1刻みの値を要素とするベクトル変数**t**を生成する．
- 2行目は，ベクトル変数**t**の各要素に対するcos関数の演算を行い，各要素に対応する演算結果をベクトル変数**f**へ代入する．
- 3行目で，ベクトル変数**t**，**f**を用いた*x*-*y*プロット表示を行う．

このように，MATLABの表現をうまく利用することで，for文を省いた簡潔な記述が可能となる．

例題2.9　**plot表示**

ディジタル信号は周期的な周波数特性を有する．横軸の1周期を2πと正規化して，先のベクトルxの周波数振幅特性を表示してみよう．

解

Command Window上で

```
>> f = 0:2/512:2-2/512;
>> A = 20*log10(abs(fft(x,512)));
>> plot(f,A)
>> xlabel(['Normalized Frequency ',...
           '(¥times¥pi rad)'])
>> ylabel('Magnitude (dB)')
```

と実行すればよい〔Unix系OS上では，¥は\(バックスラッシュ)に読み換える〕．

2.4　Mファイルの作成

ここまで読み進めると，MATLABの使い方に慣れると同時に，Command Window上での作業に煩わしさを感じるだろう．この煩わしさは，次に解説するMファイルの作成法と使用法を覚えることで解消される．

● スクリプト

MATLABでは，Mファイルと呼ばれるスクリプト・ファイルによって一連の処理をまとめることができる．ここでは，MATLAB付属のエディタなどを使用し，ampres.mというファイルを作成してみよう．

MATLABのエディタを利用する場合は，Command Window上で

```
>> edit ampres
```

と実行すればよい．また，使い慣れたテキスト・エディタを利用してもよい．ただし，ファイルの拡張子は .m とすること．

以下の内容をファイルampres.mに記述しよう．

```
x = rand(8,1);
f = 0:2/512:2-2/512;
A = 20*log10(abs(fft(x,512)));
plot(f,A)
xlabel(['Normalized Frequency ',...
        '(¥times¥pi rad)'])
ylabel('Magnitude (dB)')
```


続いてCommand Window上で

```
>> ampres
```

と実行してみよう．すると，横軸を正規化した周波数振幅特性が表示される．

● 関数定義

先のスクリプト・ファイルでは，入力xやFFT点数を変えたいときに，いちいちファイルを編集しなければならない．MATLABでは，スクリプト・ファイルをさらに発展させて，独自の関数を定義することができる．先ほど作成したファイルampres.mを以下のように編集しよう．

```
function A = ampres(x,nPoints)
% AMPRES: Display Amplitude Response
% Copyright (c), 20XX, (Your name),
%              All rights reserved
f = 0:2/nPoints:2-2/nPoints;
A = 20*log10(abs(fft(x,nPoints)));
plot(f,A)
xlabel(['Normalized Frequency ',...
        '(¥times¥pi rad)'])
ylabel('Magnitude (dB)')
```

続いて，Command Window上で

```
>> x = rand(8,1);
>> A = ampres(x,512)
```

と実行してみよう．すると周波数振幅特性が表示される．入力ベクトルxやFFT点数nPointsを変えることができるので，試してみよう．

例題2.10　**Mファイル**

周波数特性は複素数として与えられ，位相特性も持つ．位相特性を表示するphsres関数も作成してみよう．

解

ファイル名：phsres.m

```
function P = phsres(x,nPoints)
% PHSRES: Display Phase Response
% Copyright (c), 20XX, (Your name),
%              All rights reserved
f = 0:2/nPoints:2-2/nPoints;
X = fft(x,nPoints);
P = 180*angle(X)/pi;
plot(f,P)
xlabel(['Normalized Frequency ',...
```

```
        '(¥times¥pi rad)'])
  ylabel('Phase (degrees)')
```

angle関数についてはオンライン・ヘルプを参照されたい．

例題2.11　Mファイル

subplot関数を用いると，一つのグラフ内に複数のグラフを表示できる(help subplotを参照)．振幅特性と位相特性を同時に表示するfreqres関数を作成してみよう．

解

ファイル名：freqres.m

```
function [A,P] = freqres(x,nPoints)
% FREQRES: Display Frequancy Response
% Copyright (c), 20XX, (Your name),
%              All rights reserved
f = 0:2/nPoints:2-2/nPoints;
X = fft(x,nPoints);
A = 20*log10(abs(X));
P = 180*angle(X)/pi;
subplot(2,1,1);plot(f,A)
xlabel(['Normalized Frequency ',...
        '(¥times¥pi rad)'])
ylabel('Magnitude (dB)')
subplot(2,1,2);plot(f,P)
xlabel(['Normalized Frequency ',...
        '(¥times¥pi rad)')
ylabel('Phase (degrees)')
```

例題2.12　オンライン・ヘルプ

オンライン・ヘルプ機能を使ってhelp ampres，help phsres，help freqresを調べてみよう．

解

functionに続く，コメントアウトされた行の内容が表示される．

2.5　変数の保存と読み込み

Command Window上で利用した変数は，ワークスペース(Workspace)上に保持されている．しかし，ワークスペース上のデータはMATLABを終了するたびに消えてしまう．以下では，ワークス

ペース上におけるデータのファイル保存とその読み込みに関する機能を紹介しよう．

● 変数の表示

ワークスペース上に保持されている変数を確認してみよう．Command Window上ではコマンドwhoを利用できる．例えば，

```
>> who
```

と実行すると

```
Your variables are:
A x
```

のように表示される．この例では，保持されている変数がAとxであることがわかる．より詳しく調べたいときは，コマンドwhosを利用する．データ型やサイズなど，より詳細に表示される．また，Workspaceウィンドウでも確認できる．

● 変数の保存

変数Aとxを保存したい場合，Command Window上では

```
>> save mysignal A x
```

と実行すればよい．変数Aとxはファイルmysignal.matに保存される．これはMATファイルと呼ばれる．拡張子 .matは自動的に付加される．

実際に保存されているかを確かめてみよう．Command Window上で

```
>> ls
```

と実行することにより，ファイルmysignal.matの有無を確認できる．

● MATファイルの読み込み

では次に，ファイルmysignal.matを読み込んでみよう．その前に，Command Window上で

```
>> clear
```

を実行しよう．コマンドclearは，ワークスペース上の変数をすべて消去する．この時点で，コマンドwhoを用いて現在保持している変数を確認しても何も表示されないはずである．clearは変数名を指定して消去することも可能である．詳細はhelp clearを参照されたい．

すべての変数が消去されているようならば，以下のコマンドを実行してみよう．

```
>> load mysignal
>> who
```

変数Aとxがワークスペース上に読み込まれていることが確認できる．

ここまでの内容をマスタすれば，MATLABを利用した高度なシミュレーションも可能だろう．なお、次章以降でMATLABのコマンド例を示す際には，スクリプトとしてMファイルに記述することも考慮し，プロンプト>>は省略する．

参考文献

(1) サイバネットのMATLABホームページ，http://www.cybernet.co.jp/matlab/

第3章

1次元信号処理の復習

第4章以降で，画像・映像信号処理について解説する．本章では，その理解に必要な1次元信号処理について，MATLABを利用しながら復習を行おう．内容は，フーリエ解析，標本化定理，窓関数，Z 変換，線形時不変システム，畳み込み演算，伝達関数，フィルタ設計の概説である．復習という位置づけのため，詳細な定義や議論，証明などは割愛している．1次元信号処理についてより詳しく学習したい読者は，文献(1)などを参照してほしい．既に1次元信号処理について熟知している読者は，本章を読み飛ばして第4章以降に進んでも差し支えない．

3.1 標本化と量子化

ディジタル信号処理システムの基本構成を**図3.1**に示す．アナログ-ディジタル変換器(ADC：Analog to Digital Converter)には，あらかじめアナログ・フィルタ(Analog Low-pass Filter)によって帯域制限された信号$x(t)$が入力され，標本化と量子化が施される．結果としてディジタル信号$x[n]$が出力される．$x[n]$には，フィルタ処理や変調，圧縮などのディジタル処理が施される．その出力$y[n]$は，必要に応じて再度ディジタル-アナログ変換器(DAC：Digital to Analog Converter)とアナログ・フィルタによりアナログ信号$y_R(t)$として復元される．

● 標本化

図3.2に標本化(Sampling)の概略を示す．標本化は，あらかじめ定められた標本化周期(Sampling Period)ごとにアナログ信号$x(t)$の瞬時値$x(nT_S)$を取得し，離散時間信号，すなわち数列$x_C[n]$へと変換

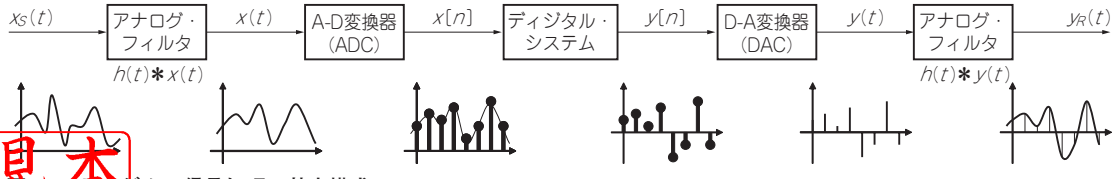


図3.1 ディジタル信号処理の基本構成

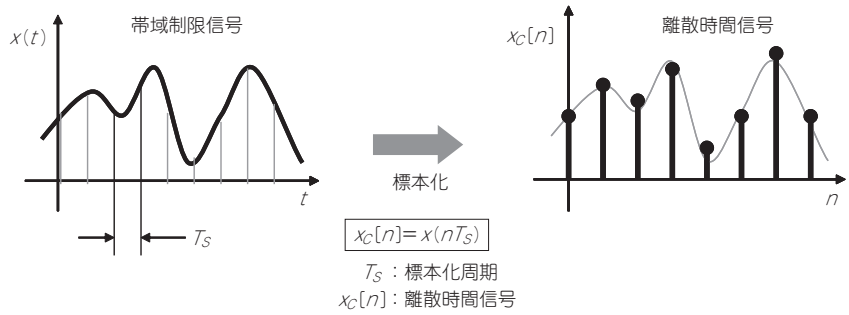


図3.2　標本化

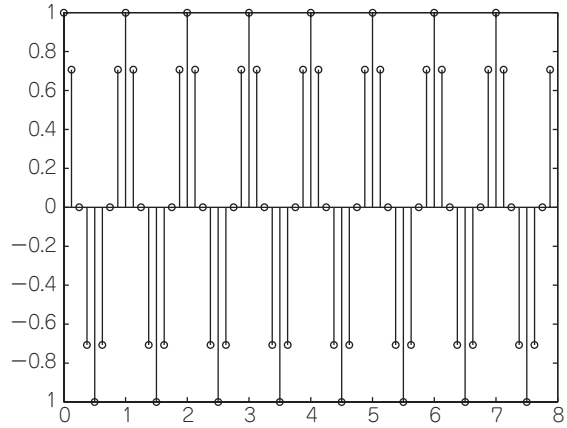


図3.3　余弦波の標本化

する．各標本の値は標本値と呼ぶ．

例題3.1　**標本化**

周波数1Hzの余弦波$x(t) = \cos 2\pi t$に対し，標本化周期$T_S = 1/8$［sec］の標本化を施し，離散時間信号$x_C[n]$を生成してみよう．

解

以下に，MATLABによるシミュレーション例を示す．なお，時間幅を8［sec］とした．

```
f = 1;
Ts = 1/8;
t = 0:Ts:8-Ts;
xc = cos(2*pi*f*t);
stem(t,xc)
```

図3.3に結果を示す．

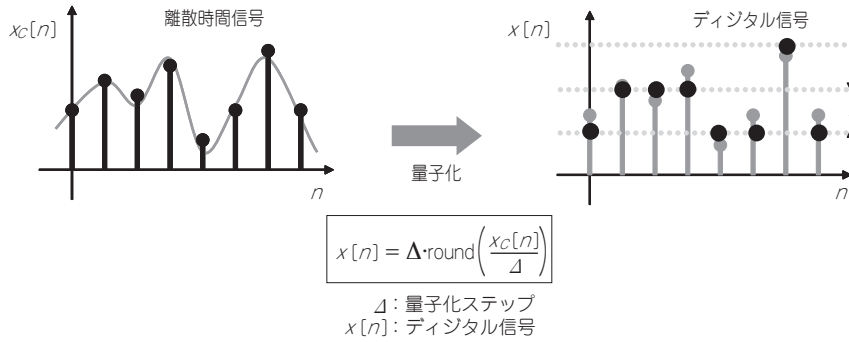


図3.4　量子化

実習3.1　**標本化**

M-file：practice03_1.m

標本化周期T_Sを半分(1/16 [sec])，または倍(1/4 [sec])にして，同様の手順を試してみよう．そのほかの条件についても試してみよう．

● 量子化

一般に，離散時間信号$x_C[n]$は時間的には離散であるが，振幅は連続な値をもつ．このため，コンピュータによる処理には適していない．そこで，振幅を離散化するために量子化を施す．**図3.4**に量子化(Quantization)の概略を示す．あらかじめ定められた量子化ステップ(Quantization Step)ごとに離散時間信号$x_C[n]$の振幅を離散化し，ディジタル信号$x[n]$を得ている．線形量子化ではΔを一定に設定するが，必ずしも一定である必要はない[注3-1]．

線形量子化の一例を示そう．

$$x[n] = \Delta \cdot \mathrm{round}\left(\frac{x_C[n]}{\Delta}\right) \quad \cdots\cdots (3.1)$$

ただし，roundは四捨五入による整数化を意味する．

例題3.2　**量子化**

例題3.1で与えられた離散時間信号$x_C[n]$に対して，$\Delta = 1/4$の線形量子化を施してみよう．

解

以下に，MATLABによるシミュレーション例を示す．ただし，xcには余弦波を標本化して得られた離散時間信号$x_C[n]$が保持されているものとする．

```
delta = 1/4;
x = delta * round(xc/delta);
```


注3-1：非線形量子化や適応量子化など．

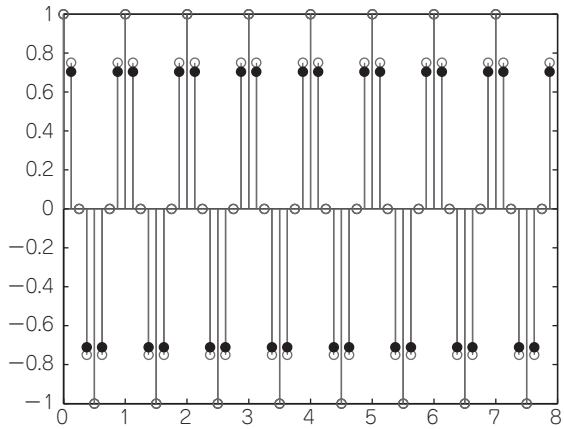


図3.5　余弦波の離散化

```
stem(t,xc,'filled')
hold on
stem(t,x)
hold off
```

図3.5に結果を示す．黒丸(●)が量子化前，白丸(○)が量子化後の数列を表している．量子化による誤差を確認することができる．

実習3.2　**量子化**

M-file：practice03_2.m

量子化ステップΔを倍(1/2)にして同様の手順を試してみよう．また，そのほかの条件でも試してみよう．

3.2　周波数解析

図3.6に示すように，アナログ信号$x(t)$はさまざまな周波数の(複素)正弦波の重み付け和に展開できる．フーリエ解析は，ある信号がどのような正弦波の成分から構成されるのかを解析する手段を与える．

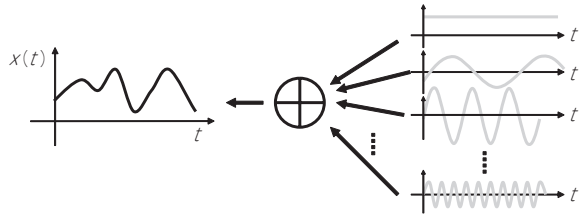


図3.6　フーリエ解析

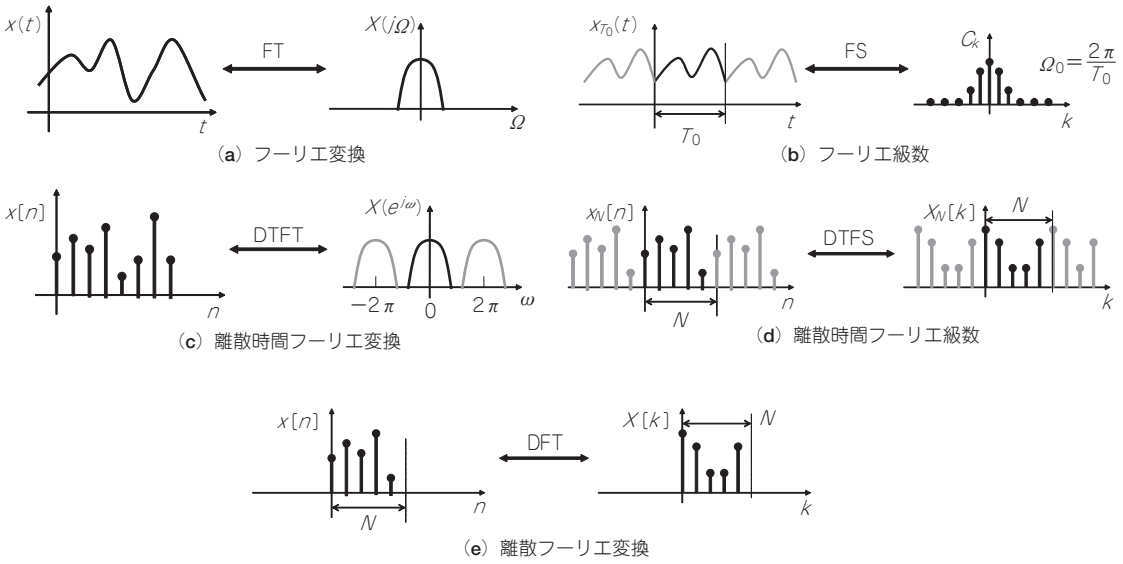


図3.7　五つのフーリエ解析

● フーリエ解析のバリエーション

信号の周期性の有無，連続・離散の違いから，フーリエ解析には以下の五つのバリエーションがある（以下，Rは実数集合，zは整数集合を意味する）．

(1) フーリエ変換（FT：Fourier Transform）

フーリエ変換は，非周期・連続な信号$x(t)$の周波数解析に用いられる．周波数スペクトラム$X(j\Omega)$もまた，非周期・連続となる（**図3.7**(**a**)を参照）．順変換，逆変換は以下のとおり．

$$X(j\Omega)=\int_{-\infty}^{\infty}x(t)e^{-j\Omega t}dt,\quad \Omega\in R \qquad \cdots\cdots (3.2a)$$

$$x(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}X(j\Omega)e^{j\Omega t}d\Omega,\quad t\in R \qquad \cdots\cdots (3.2b)$$

(2) フーリエ級数（FS：Fourier Series）

フーリエ級数は，周期・連続な信号$x_{T_0}(t)$の周波数解析に用いられる．周波数スペクトラムC_kは，非周期・離散となる（**図3.7**(**b**)を参照）．順変換，逆変換は以下のとおり．

$$C_k=\frac{1}{T_0}\int_0^{T_0}x_{T_0}(t)e^{-jk\Omega_0 t}dt,\quad k\in Z \qquad \cdots\cdots (3.3a)$$

$$x_{T_0}(t)=\sum_{k=-\infty}^{\infty}C_k e^{jk\Omega_0 t},\quad t\in R \qquad \cdots\cdots (3.3b)$$

(3) 離散時間フーリエ変換（DTFT：Discrete-Time Fourier Transform）

離散時間フーリエ変換は，非周期・離散な信号$x[n]$の周波数解析に用いられる．周波数スペクトラム$X(e^{j\omega})$は，周期・連続となる（**図3.7**(**c**)を参照）．順変換，逆変換は以下のとおり．

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}, \quad \omega \in R \quad \cdots\cdots (3.4a)$$

$$x[n] = \frac{1}{2\pi}\int_0^{2\pi} X(e^{j\omega})e^{j\omega n}, \quad n \in Z \quad \cdots\cdots (3.4b)$$

(4) 離散時間フーリエ級数(DTFS：Discrete-Time Fourier Series)

離散時間フーリエ級数は，周期・離散な信号$x_N[n]$の周波数解析に用いられる．周波数スペクトラム$X_N[k]$は，周期・離散となる(**図3.7(d)**を参照)．順変換，逆変換は以下のとおり．

$$X_N[k] = \sum_{n=0}^{N-1} x_N[n]W_N^{nk}, \quad k \in Z \quad \cdots\cdots (3.5a)$$

$$x_N[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_N[k]W_N^{-nk}, \quad n \in Z \quad \cdots\cdots (3.5b)$$

ただし，$W_N = e^{-j\frac{2\pi}{N}}$．

(5) 離散フーリエ変換(DFT：Discrete Fourier Transform)

離散フーリエ変換は，有限・離散な信号$x[n]$の周波数解析に用いられる．周波数スペクトラムX[k]は，有限・離散となる(**図3.7(e)**を参照)．順変換，逆変換は以下のとおり．

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk}, \quad k = 0,1,\cdots,N-1 \quad \cdots\cdots (3.6a)$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-nk}, \quad n = 0,1,\cdots,N-1 \quad \cdots\cdots (3.6b)$$

これは有限信号に周期性を仮定して，DTFSによる解析を行うことと等価である．

DFTには高速アルゴリズムが存在する．総称して**高速フーリエ変換(FFT：Fast Fourier Transform)**と呼ばれる．MATLABでは，`fft`関数，`ifft`関数としてDFTの順変換と逆変換が提供されている．

FFTは積分を必要とせずに点で高速な演算が可能なため，有限・離散ではない信号に対してもよく利用される．この場合，信号の周期性を暗に仮定していることに注意されたい．

例題3.3　周波数特性

例題3.1で与えられた余弦波の離散時間信号$x_C[n]$の周波数特性を512点DFTを用いて観察してみよう．

解

以下に，MATLABにおけるコマンド例を示そう．なお，`xc`にはあらかじめ例題3.1で与えられた数列が保持されているものとする．

```
nPoints = 512;
figure(1) % 正規化周波数
freqz(xc,1,nPoints)
figure(2) % アナログ周波数
```

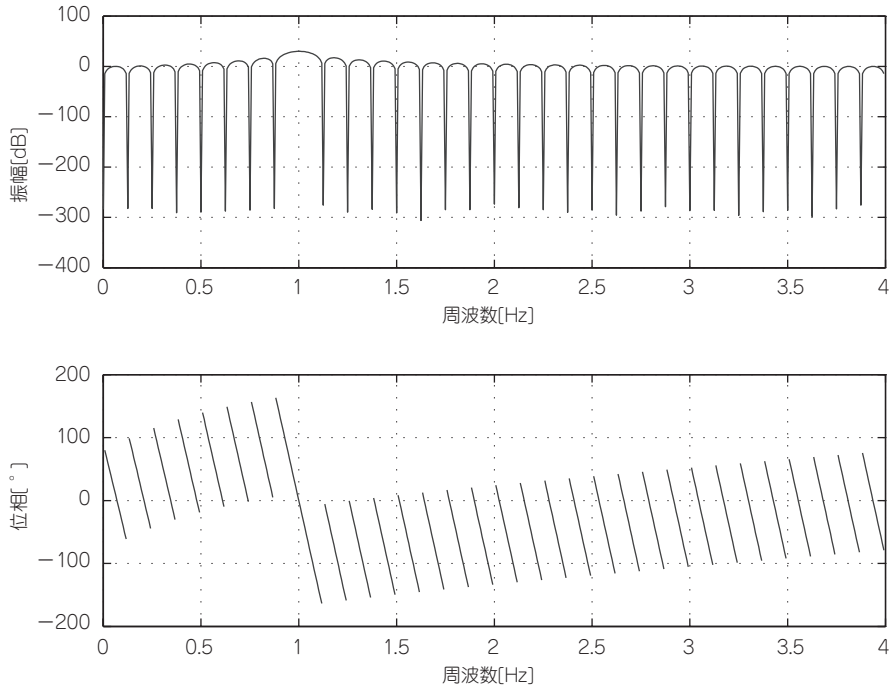


図3.8　周波数特性

```
freqz(xc,1,nPoints,1/Ts)
```

周波数特性の表示には`freqz`関数を用いた．**図3.8**にアナログ周波数表示の結果を示す．

実習3.3　**周波数特性**

M-file：practice03_3.m

次のことを試してみよう．ただし，DFT点数は512点で一定とする．

- 標本化周期 $T_S = 1/16$ [sec]，$1/4$ [sec] の場合についても周波数特性を確かめてみよう．どのように変化するか？
- 時間幅を4 [sec]，または16 [sec] にして周波数特性を確かめてみよう．どのように変化するか？

● 標本化定理

最大周波数 Ω_{MAX} の帯域制限信号 $x(t)$は，$\Omega_S = 2\pi/T_S > 2\Omega_{\mathrm{MAX}}$ の標本化周波数で標本化した標本化列 $x[n]$から再構成可能である．これを**標本化定理**という．逆に，この条件を満たさなければ再構成は不可能である．このことをMATLABで確かめてみよう．

例題3.4　**標本化定理の確認**

周波数 $F = \Omega/2\pi = 3$ [Hz] の余弦波 $x_3(t) = \cos 6\pi t$ と，周波数 $F = \Omega/2\pi = 5$ [Hz] の余弦波 $x_5(t) =$

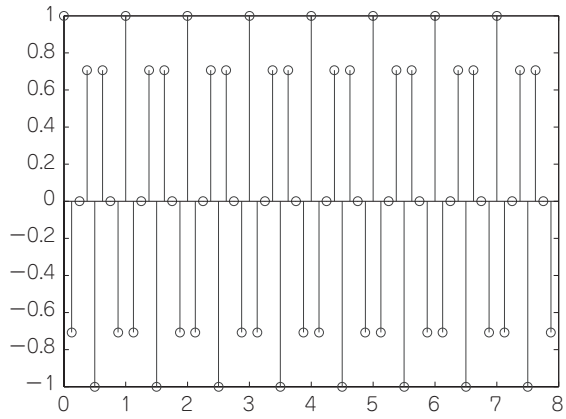


図3.9　標本化定理の確認

$\cos 10\pi t$を，共に標本化周期$T_S = 1/8$［sec］で標本化し，結果として得られる離散時間信号$x_3[n]$と$x_5[n]$を比較してみよう．

解

以下にMATLAB上でのコマンド例を示す．

```
% DFT点数
nPoints = 512;
% 標本化周期
Ts = 1/8;
% 時間幅
duration = 8;
% 標本点
t = 0:Ts:duration-Ts;
% 標本データ
x3 = cos(6*pi*t);
x5 = cos(10*pi*t);
% x3の表示
figure(1) % 数列
stem(t,x3)
figure(2) % 周波数特性
freqz(x3,1,nPoints,1/Ts)
% x5の表示
figure(3) % 数列
stem(t,x5)
figure(4) % 周波数特性
```

表3.1　代表的な窓関数

名　称	定　義
方形窓	$w_R = \begin{cases} 1 & 0 \leqq n \leqq M-1 \\ 0 & \text{それ以外} \end{cases}$
ハニング窓	$w_N = \frac{1}{2} w_R \left(1 - \cos \frac{2\pi n}{M}\right)$
ハミング窓	$w_M = \frac{1}{2} w_R \left(d - (1-d)\cos \frac{2\pi n}{M}\right),\ d = \frac{25}{46}$
ブラックマン・ハリス窓	$w_B = w_R \left(0.423 - 0.498 \cos \frac{2\pi n}{M} + 0.0792 \cos \frac{4\pi n}{M}\right)$

```
freqz(x5,1,nPoints,1/Ts)
```

x_3とx_5がまったく同じになることが確認できる(**図3.9**)．$F = 5$［Hz］の余弦波は標本化定理を満たしていないため，$F_S/2 = 4$［Hz］で折り返し，$F = 3$［Hz］と区別できなくなる．

実習3.4　**標本化定理の確認**

M-file：practice03_4.m

$\Omega > \Omega_S/2$となるほかの余弦波でも試してみよう．また，標本化周期を変化させるとどうなるかを試してみよう．

● 窓関数

理論上，余弦波(あるいは正弦波)の周波数特性は線スペクトラムとなるはずである．しかし，例えば**図3.8**のように，DFT(FFT)の結果は線スペクトラムとなっていない．

DFTは有限長離散信号のためのフーリエ解析である．先の例では余弦波に有限な時間範囲を定め，DFT点数に足りない部分をゼロ値のサンプルで埋めた．その影響が周波数特性の計算結果に現れている．一般に信号の周期性は不明であり，切り取った信号の周波数特性を計算すると，同じような問題が起こりうる．

窓関数は，信号の切り取りの影響を抑える役割をもつ．短時間フーリエ変換(MATLABではspecgram関数)やフィルタ設計(MALTABでは`fir1`関数や`fir2`関数)に利用される．**表3.1**に代表的な窓関数の例を挙げておこう．

例題3.5　**窓関数**

例題3.1の離散時間信号$x_C[n]$に対してハニング窓を掛けてみよう．ただし，窓長Mを64点(8［sec］)としよう．また，DFT点数を512点として周波数特性も観察し，**図3.8**と比較してみよう．

解

以下にMATLABによるコマンド例を示す．


```
% 窓長
nPoints = 64;
```

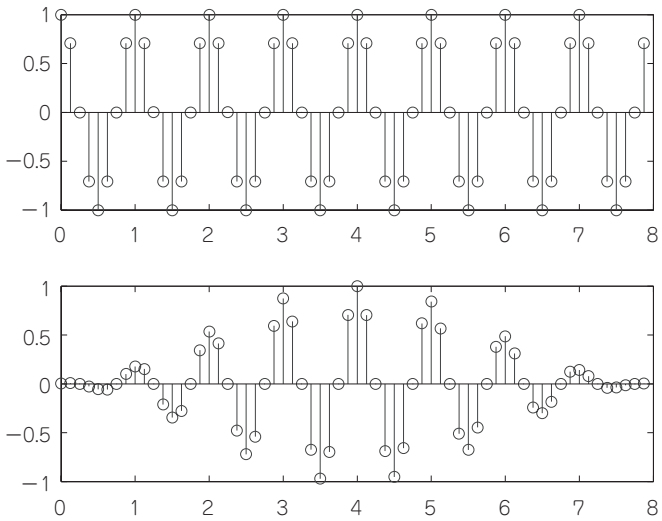
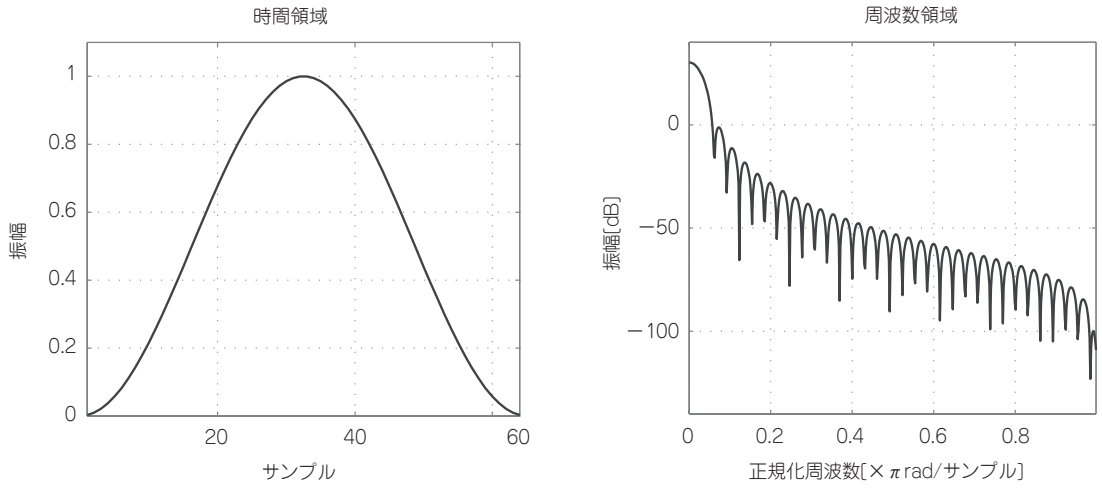
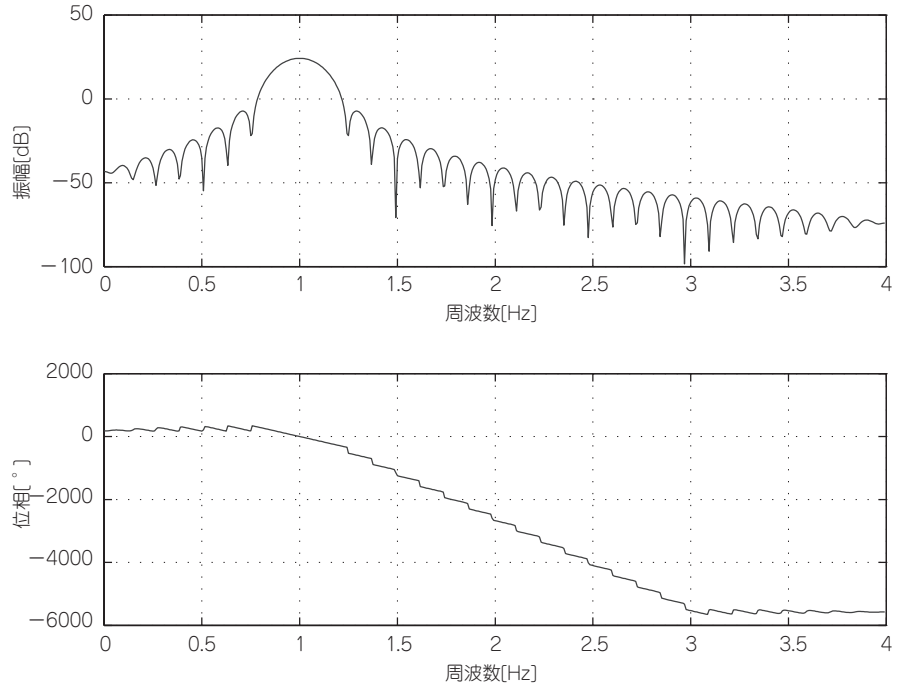


図3.10　窓関数適用前（上）と適用後（下）

```
% 標本化周期
Ts = 1/8; % 1/Fs
% 時間幅
duration = 8;
% 標本点
t = 0:Ts:duration-Ts;
% 標本データ
xc = cos(2*pi*t);
% 窓関数
w = window(@hanning, nPoints);
% 窓関数の適用
xw = w(:).*xc(:);
xr = xc(:);
% 時間領域表現
figure(1)
subplot(2,1,1)
stem(t,xc)
subplot(2,1,2)
stem(t,xw)
% 周波数領域表現
figure(2)
```



（**a**）ハニング窓の例



（**b**）窓関数適用後の周波数特性

図3.11　窓関数の例

```
freqz(xw,1,512,1/Ts)
hold on
freqz(xr,1,512,1/Ts)
subplot(2,1,1)
axis([0 4 -100 50])
% 窓関数
wvtool(w);
```

図3.10，**図3.11**に結果を示す．

実は，**図3.8**は64点の方形窓を掛ける操作そのものである．

実習3.5　**窓関数**

M-file：practice03_5.m

ハミング窓はhamming関数として用意されている．@hanningの代わりに@hammingも試してみよう．窓長，DFT点数も変えて観察してみよう．

3.3　Z変換

信号処理では，時間領域表現，周波数領域表現に加えて，Z変換領域表現の三つの表現を使いこなすことが要求される．Z変換領域の表現は，システムの安定性の判別や設計，実現において重要な役割を果たす．

● インパルスの重み付け和表現

任意の離散時間信号$x[n]$（以降，量子化の影響を無視して$x_C[n]$という表記を$x[n]$に変える）は，インパルスの重み付け和として，

$$x[n]=\sum_{k=-\infty}^{\infty}x[k]\delta[n-k] \quad \cdots\cdots (3.7)$$

と表現できる．ここで，

$$\delta[n]=\begin{cases}1 & n=1\\ 0 & n\neq 0\end{cases}$$

である．例えば，**図3.12**の数列は**図3.13**のように表現できる．これはZ変換にとっても便利な表現である．

● 定義

Z変換は，

$$X(z)=\sum_{n=-\infty}^{\infty}x[n]z^{-n},\quad z\in C \quad \text{（以下，}C\text{は複素数集合を意味する）} \quad \cdots\cdots (3.8)$$

と定義され，

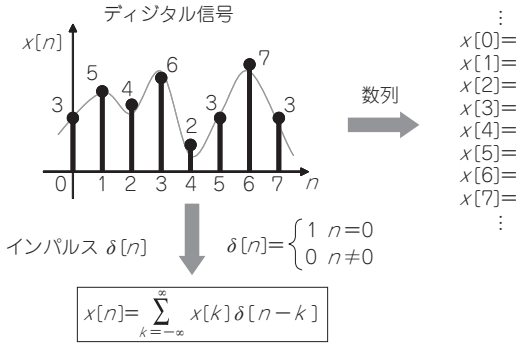


図3.12　インパルスの重み付け和表現(1)

$$\cdots,\ x[0]=3,\ x[1]=5,\ x[2]=4,\ \cdots$$

$$\downarrow$$

$$x[n]=\cdots+3\,\delta[n-0]+5\,\delta[n-1]+4\,\delta[n-2]+\cdots$$

図3.13　インパルスの重み付け和表現(2)

$$x[n]=\cdots+3\,\delta[n-0]+5\,\delta[n-1]+4\,\delta[n-2]+\cdots$$

$$\updownarrow Z$$

$$X(z)=\cdots+3\,z^{-0}+5\,z^{-1}+4\,z^{-2}+\cdots$$

図3.14　Z変換の例

$$Z\{x[n]\}=X(z)$$

とも表現される．

例題3.6　**インパルスのZ変換**

nを変数，kを定数として$\delta[n-k]$のZ変換

$$Z\{\delta[n-k]\}$$

を求めてみよう．

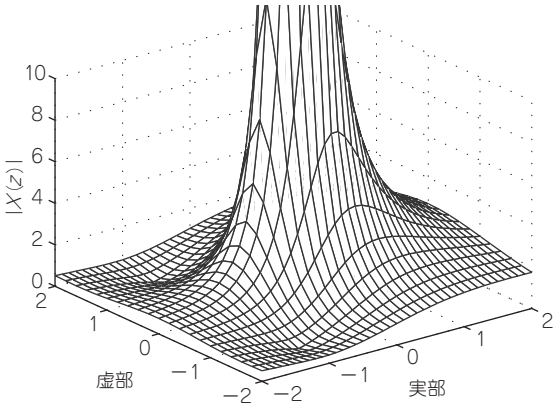
解

$$Z\{\delta[n-k]\}=\sum_{n=-\infty}^{\infty}\delta[n-k]z^{-n}=\sum_{n=-\infty}^{\infty}\delta[n]z^{-(n+k)}=z^{-k}$$

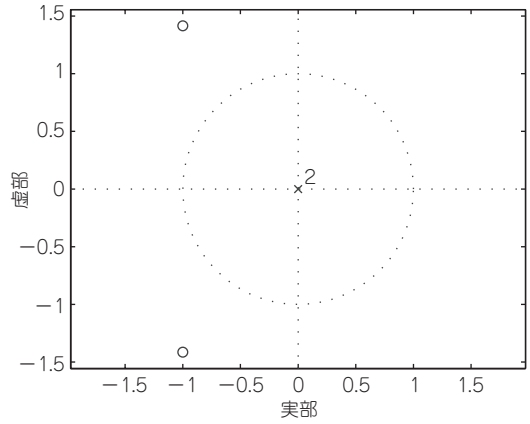
Z変換は線形であるため，**図3.13**の数列の$\delta[n-k]$をz^{-k}に置き換えることで，**図3.14**のようにZ変換領域による表現が与えられる．

例題3.7　**Z変換と複素平面**

数列



(a) Z変換



(b) 零極点配置

図3.15　Z変換と零極点配置

$x[n] = \delta[n] + 2\delta[n-1] + 3\delta[n-2]$

のZ変換$X(z)$を求めて，$|X(z)|$のグラフを表示してみよう．また，$X(z) = 0$となる複素平面上でのz(零点)を確かめよう．

解

MATLABによるコマンド例を以下に示そう．

```
% 数列
x = [ 1 2 3 ];
% z変換
[re,im]=meshgrid(-2:.1:2,-2:.2:2);
z = re+i*im;
X = 0;
for iSample=1:length(x)
  X = X + x(iSample) * z.^(-iSample+1);
end
% メッシュ表示
figure(1)
mesh(re,im,abs(X))
xlabel('Real')
ylabel('Imaginary')
zlabel('|X(z)|')
axis([-2 2 -2 2 0 10])
% 零極点配置
figure(2)
```

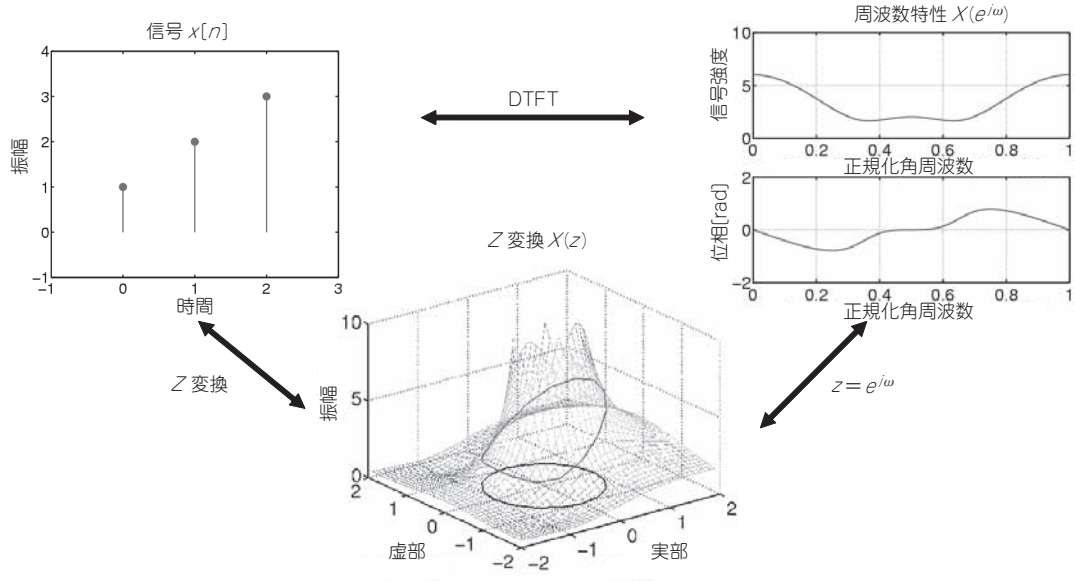


図3.16　時間，周波数，Z変換領域の関係

```
zplane(x)
```

図3.15に処理結果を示す．meshgrid関数は複素平面上の標本位置を生成している．zplane関数は**図3.15(b)**のように複素平面上の零極点配置を表示する．通常は零極点配置を知るだけで十分なため，**図3.15(a)**のような表示までは必要とされない．

実習3.6　Z変換と複素平面

M-file：practice03_6.m

ほかの数列でも試してみよう．

● 離散時間フーリエ変換(DTFT)との関係

信号$x[n]$のZ変換$X(z)$に$z = e^{j\omega}$を代入すると，

$$X(z)\big|_{z=e^{j\omega}} = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} = X(e^{j\omega}),\quad \omega \in R \qquad \cdots\cdots (3.9)$$

という関係が得られ，離散時間フーリエ変換(DTFT)に帰着する．すなわちDTFTは，Z平面の単位円周上$z = e^{j\omega}$で$X(z)$を評価していることにほかならない．

$$x[n] = \cdots + 3\,\delta[n-0] + 5\,\delta[n-1] + 4\,\delta[n-2] + \cdots$$

$$\updownarrow \text{ DTFT}$$

$$X(e^{j\omega}) = \cdots + 3\,e^{-j0} + 5\,e^{-j\omega} + 4\,e^{-j2\omega} + \cdots$$

図3.17　周波数特性の例

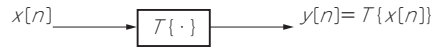


図3.18　システム

信号$x[n]$，周波数特性$X(e^{j\omega})$，Z変換$X(z)$の間には，**図3.16**に示されるような深い関係がある．結局，**図3.13**の数列は，$\delta[n-k]$を$e^{-j\omega k}$に置き換えることで，**図3.17**のような周波数領域による表現となる．

3.4　線形時不変システム

以下では，信号処理システムにおいて，最も基本的な**線形時不変（LTI：Linear Time-invariant）システム**について解説しよう．LTIシステムは，そのインパルス応答$h[n]$をもって性質を特徴付けることができる．$h[n]$のDTFTは**周波数応答**と呼ばれ，周波数領域のシステムの振る舞いを表す．また，$h[n]$のZ変換は**伝達関数**と呼ばれ，システムの安定判別や実現などに重要な役割を果たす．

● 線形性と時不変性

図3.18に，一般的な信号処理システム$T\{\cdot\}$を示そう．信号処理システムは，入力信号$x[n]$が与えられると別の信号$y[n]=T\{x[n]\}$に変換して出力する．

(1) 線形性とは

システム$T\{\cdot\}$の線形性は，定数a，bに対して，

$$y_1[n]=T\{x_1[n]\},\ y_2[n]=T\{x_2[n]\}$$
$$\Rightarrow y[n]=T\{ax_1[n]+bx_2[n]\}=aT\{x_1[n]\}+bT\{x_2[n]\} \quad \cdots\cdots (3.10)$$

と表現することができる．

例題3.8　**線形性**

次のシステムは線形か．

1. $T_1\{x[n]\}=2x[n]+x[n-1]$
2. $T_2\{x[n]\}=x2[n]+x[n-1]$
3. $T_3\{x[n]\}=x[2n]+x[n-1]$

解

$T_1\{\cdot\}$，$T_3\{\cdot\}$は線形，$T_2\{\cdot\}$は非線形である．

(2) 時不変性とは

システム$T\{\cdot\}$の時不変性は，

$$y=T\{x[n]\}\Rightarrow y[n-k]=T\{x[n-k]\} \quad \cdots\cdots (3.11)$$

と表現することができる．

例題3.9　時不変性

次のシステムは時不変か．

1. $T_1\{x[n]\} = 2x[n] + x[n-1]$
2. $T_2\{x[n]\} = x2[n] + x[n-1]$
3. $T_3\{x[n]\} = x[2n] + x[n-1]$

解

$T_1\{\cdot\}$，$T_2\{\cdot\}$は時不変，$T_3\{\cdot\}$は時変である．

● 畳み込み演算

線形時不変システムは，線形性と時不変性を同時に満たす．また，その特徴はインパルス$\delta[n]$に対する応答$h[n]$で表現され，システムの振る舞いは，以下に示すとおり入力信号$x[n]$とインパルス応答$h[n]$の畳み込み演算となる．

$$y[n] = T\{x[n]\} = T\left\{\sum_{k=-\infty}^{\infty} x[k]\delta[n-k]\right\}$$

$$= \sum_{k=-\infty}^{\infty} x[k]T\{\delta[n-k]\} \quad (\because \text{線形性})$$

$$= \sum_{k=-\infty}^{\infty} x[k]h[n-k] \quad (\because \text{時不変性}) \qquad \cdots\cdots (3.12)$$

ただし，$h[n] = T\{\delta[n]\}$とする．

どのような線形時不変システムも，そのインパルス応答$h[n]$をもって特徴付けられることが分かる．なお，

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \qquad \cdots\cdots (3.13)$$

が成立することは，畳み込み演算の重要な性質の一つである．

例題3.10　線形時不変システム

入力された信号について，連続するサンプル3点の平均を取りながら出力信号を与えるシステムを考えよう．これは，$h[n] = T\{\delta[n]\} = 1/3(\delta[n] + \delta[n-1] + \delta[n-2])$の線形時不変システムとして解釈できる．入力信号$x[n] = \delta[n] + 2\delta[n-1] + 3\delta[n-2]$の応答を確かめてみよう．

解

MATLABでは，畳み込み演算を行う関数としてconv関数が用意されている．以下に，MATLABによる処理例を示そう．

```
% 入力x[n]
x = [1 2 3];
% インパルス応答h[n]
```

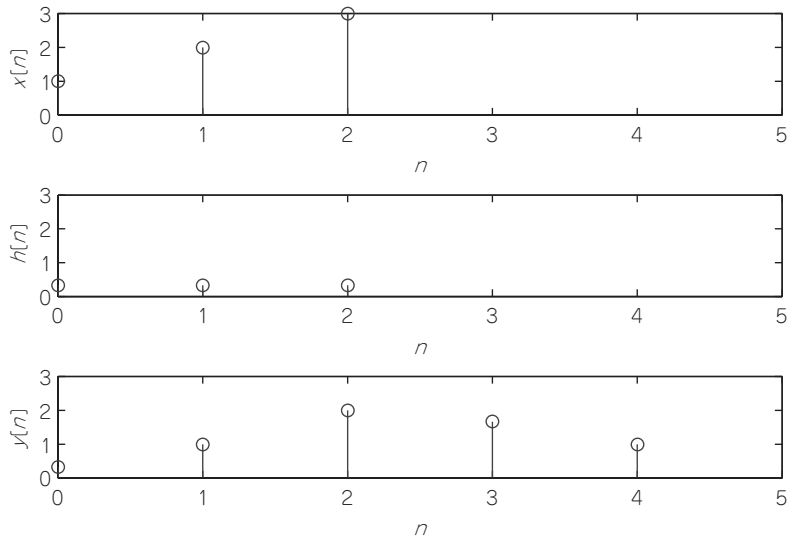


図3.19　線形時不変システムの例(3点移動平均フィルタ)

```
h = [1 1 1]/3;
% 出力y[n]
y = conv(h,x);
% x[n] の表示
subplot(3,1,1)
stem(0:length(x)-1,x)
axis([0 length(y) 0 3])
xlabel('n');ylabel('x[n]')
% h[n] の表示
subplot(3,1,2)
stem(0:length(h)-1,h)
axis([0 length(y) 0 3])
xlabel('n');ylabel('h[n]')
% y[n] の表示
subplot(3,1,3)
stem(0:length(y)-1,y)
axis([0 length(y) 0 3])
xlabel('n');ylabel('y[n]')
```

図3.19にMATLABによる演算結果の例を示す.

実習3.7　線形時不変システム

M-file：practice03_7.m

入力信号$x[n]$やインパルス応答$h[n]$をいろいろ変えて試してみよう．

● 伝達関数と周波数応答

インパルス応答$h[n]$のZ変換

$$H(z)=\sum_{n=-\infty}^{\infty}h[n]z^{-n} \quad \cdots\cdots (3.14)$$

をそのシステムの**伝達関数**という．伝達関数$H(z)$は，入力信号と出力信号のZ変換$X(z)$，$Y(z)$より，

$$H(z)=\frac{Y(z)}{X(z)} \quad \cdots\cdots (3.15)$$

と表現される．また，

$$H(e^{j\omega})=H(z)\big|_{z=e^{j\omega}} \quad \cdots\cdots (3.16)$$

はインパルス応答$h[n]$のDTFTであり，周波数応答と呼ばれる．式(3.15)より，

$$Y(e^{j\omega})=H(e^{j\omega})X(e^{j\omega}) \quad \cdots\cdots (3.17)$$

が成立する．すなわち，時間領域の畳み込み演算は周波数領域における積演算と等価である．

例題3.11　周波数応答

例題3.10の内容を周波数領域で確認してみよう．

解

以下にMATLABによる処理例を示そう．ただし，x，h，yは既に計算され，保持されているものとする．

```
% X(z) の周波数特性
X=freqz(x,1,512);
% H(z) の周波数応答
H=freqz(h,1,512);
% Y(z) の周波数特性
[Y,w]=freqz(y,1,512);
% 周波数特性（応答）表示
plot(w/pi,abs(X),'b')
hold on
plot(w/pi,abs(H),'g:')
hold on
plot(w/pi,abs(Y),'r-.')
legend('X(e^{j¥omega})',...
```

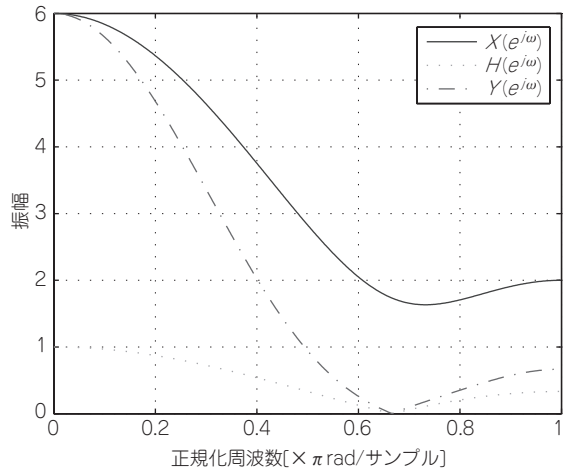


図3.20　周波数積の例(3点移動平均フィルタ)

```
    'H(e^{j¥omega})','Y(e^{j¥omega})')
  grid on
  hold off
```

図3.20にMATLABによる演算結果の例(振幅のみ)を示す．周波数応答$H(e^{j\omega})$による入力信号の周波数成分が除去されている様子がよく分かる．

実習3.8　**周波数応答**

M-file：practice03_8.m

入力信号$x[n]$やインパルス応答$h[n]$をいろいろ変えて試してみよう．

● FIRフィルタとIIRフィルタ

インパルス応答$h[n]$が無限に広がるものを**IIR(Infinite Impluse Response)システム**，有限の範囲に収まるものを**FIR(Finite Impluse Response)システム**と呼ぶ．また，インパルス応答$h[n]$が非ゼロの係数を正の時刻($n \geqq 0$)でのみもつ場合，そのシステムを**因果的**(causal)であるという．逆に$n < 0$で非ゼロ係数をもつ場合，そのシステムを**非因果的**(non-causal)であるという．

(1) FIRシステム

因果的なFIRシステムの畳み込み演算は，

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \quad \cdots\cdots (3.18)$$

のように表現でき，伝達関数も

$$H(z) = \sum_{n=0}^{N-1} h[n]z^{-n} \quad \cdots\cdots (3.19)$$

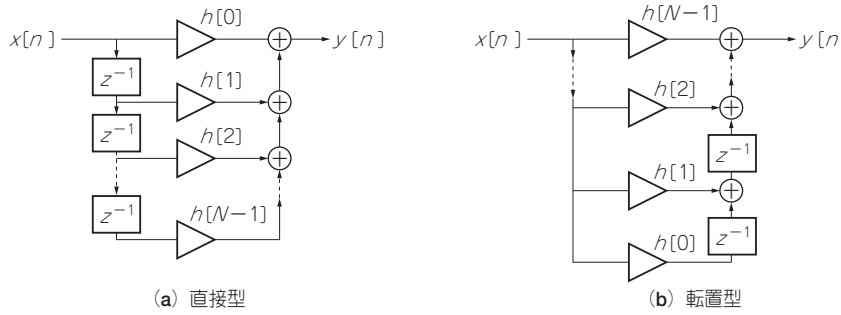


図3.21　FIRシステムの構成

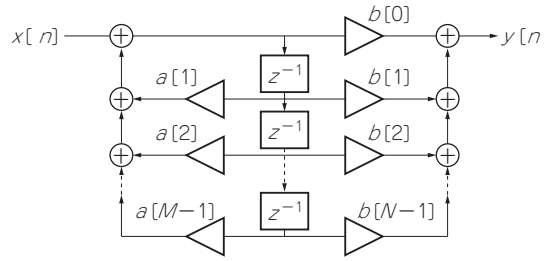


図3.22　IIR(再帰型)システムの構成(便宜上，$N = M$とした)

のように有限な数の項からなる多項式として表現される．

図3.21に因果的なFIRシステムの構成図を示す．z^{-1}は1サンプルの遅延器を意味する．定数乗算器と加算器の組み合わせによって実現できる．

(2) IIRシステム

IIRシステムは無限のインパルス応答をもつため，畳み込み演算による実現は困難である．そこで，畳み込み演算の代わりに差分方程式

$$y[n] = \sum_{k=0}^{N-1} b[k]x[n-k] - \sum_{k=0}^{M-1} a[k]y[n-k] \quad \cdots\cdots (3.20)$$

など，異なる表現を用いて実現される．

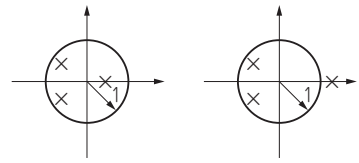
差分方程式で表現されるシステムでは，その伝達関数$H(z)$は

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B(z)}{A(z)} = \frac{b[0] + b[1]z^{-1} + \cdots + b[N-1]z^{-(N-1)}}{1 + a[1]z^{-1} + \cdots + a[M-1]z^{-(M-1)}} \quad \cdots\cdots (3.21)$$

と与えられる．有限な数の係数$b[n]$，$a[n]$からIIRシステムを構築することができる．

図3.22に差分方程式で表現される因果的IIRシステムの構成図を示す．FIRシステムと異なりフィードバック(再帰型の構成)が必要になる．

フィードバックは，そのループ利得に気をつけないと発散を起こし，システムが不安定になる恐れがある．IIRシステムは，FIRシステムに比べて特性の良いフィルタを少ない資源(リソース)で実現できる利点があるが，安定性に注意しなければならない．



（a）安定　　（b）不安定

図3.23　安定判別の概要（×印は極の位置）

(3) 安定判別

IIRシステムの安定性に関する定義の一つにBIBO（Bounded-in Bounded-out）安定性がある．BIBO安定なシステムでは，有界な入力$|x[n]| < \infty$に対し，有界な出力$|y[n]| < \infty$を得る．BIBO安定性の必要十分条件は以下のとおり．

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty \quad \cdots\cdots (3.22)$$

また，システムがBIBO判定か否かは，伝達関数$H(z)$の極の位置を見ればよい．

- すべての極（pole）が単位円内にあれば安定（**図3.23（a）**）
- 極が単位円外にあれば不安定（**図3.23（b）**）

例題3.12　**安定判別**

次の係数をもつシステムが安定か否かを判定してみよう．

1. $b[n] = 1/2(\delta[n] + \delta[n-1])$，$a[n] = \delta[n] - \delta[n-1]$
2. $b[n] = 1/2(\delta[n] + \delta[n-1])$，$a[n] = \delta[n] - 2\delta[n-1]$
3. $b[n] = 1/2(\delta[n] + \delta[n-1])$，$a[n] = \delta[n] - 1/2\delta[n-1]$

解

3. のみ安定なシステムであり，1.，2. は不安定なシステムである．以下に，3. の場合についてMATLABによる確認例を示そう．

```
% フィルタ係数
b = [1 1]/2;
a = [1 -1/2];
% インパルス応答の表示
figure(1)
impz(b,a)
% z平面表示
figure(2)
zplane(b,a)
% 周波数応答の表示
figure(3)
```

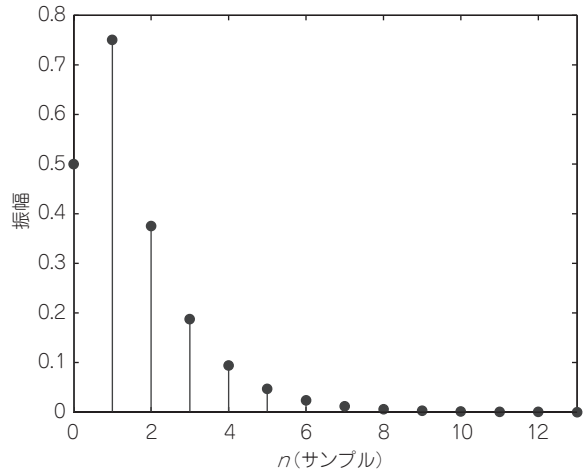
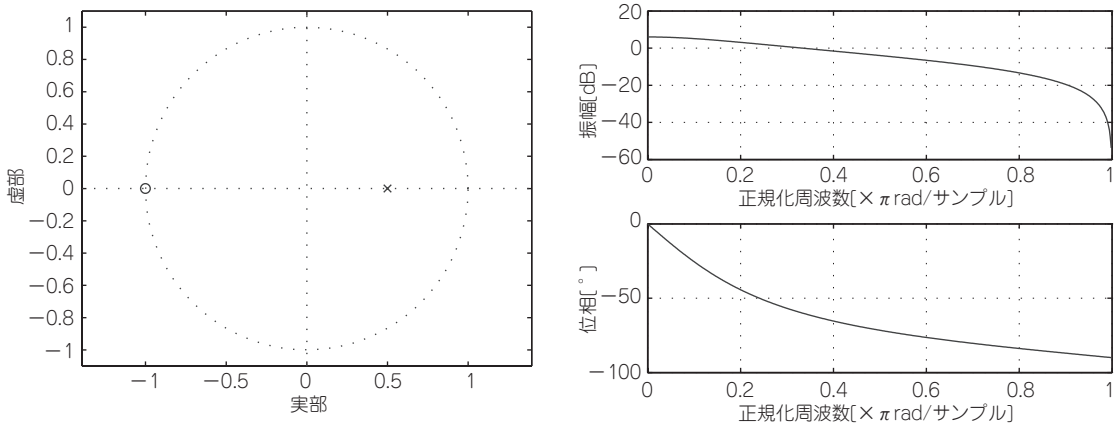


図3.24　インパルス応答



（**a**）極配置　　（**b**）周波数応答

図3.25　安定判別の例

```
freqz(b,a)
```

図3.24にインパルス応答を，**図3.25**に極配置と周波数応答を示す．impz関数はシステムのインパルス応答を表示する機能を提供する．impz関数，zplane関数，freqz関数にはいずれも分母多項式$A(z)$，分子多項式$B(z)$の係数を引き数として与えることができる．

実習3.9　**安定判別**

M-file：practice03_9.m

1.，2. の場合，そのほかのIIRシステム，FIRシステム（$a[n]=\delta[n]$）についても判定判別を試してみよう．

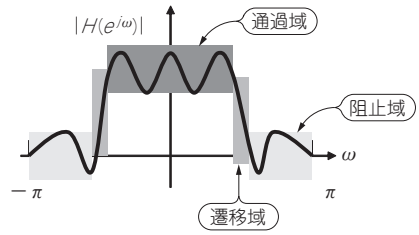


図3.26　周波数応答における主な仕様

3.5　フィルタ設計

ここまでの解説では，システムが既に与えられていることを前提としてきた．実際は，各種問題に合わせてシステムを設計する必要もあるだろう．以下では，よく用いられるフィルタ設計法の一つとして等リプルFIR設計(firpm関数)を紹介しよう．

フィルタ設計では，時間領域で仕様を与える場合，周波数領域で仕様を与える場合，両方の領域で仕様を与える場合の三つがある．以下では，周波数領域で仕様を与える例を紹介する．

周波数領域で仕様を与える場合は，通常，**図3.26**に示すように通過域，阻止域，遷移域を設定する．通過域は信号を通す周波数帯域，阻止域は信号を阻止する周波数帯域である．遷移域は通過域と阻止域の間の帯域で，設計の評価に含めない．各帯域における誤差(揺らぎ)をリプルと呼ぶ．等リプル設計法は，各帯域の揺らぎの大きさが一定となるフィルタを与える．なお，設計の原理については文献(1)などを参照されたい．

例題3.13　**フィルタ設計**

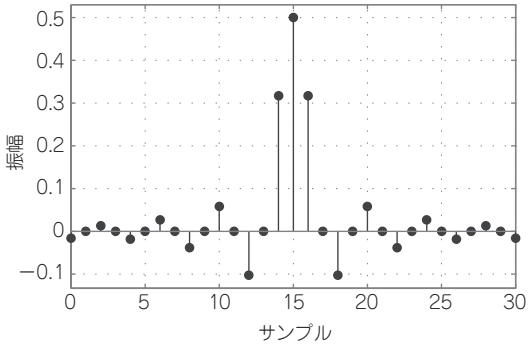
以下の仕様を満たすフィルタを設計しよう．

- 伝達関数の次数：30(タップ数：31)
- 通過域：$0 \leqq \omega < \pi/2 - \pi/20$
- 遷移域：$\pi/2 - \pi/20 \leqq \omega \leqq \pi/2 + \pi/20$
- 阻止域：$\pi/2 + \pi/20 \leqq \omega < \pi$
- 通過域利得：1

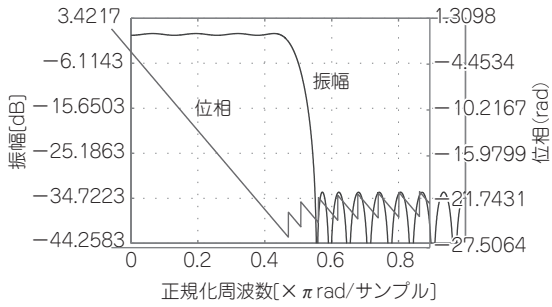
解

firpm関数を用いて与えられた仕様を満たすフィルタを設計しよう．以下にMATLABによるコマンド例を示す．

```
% フィルタ次数
nOrder = 30;
% 通過域端
edgePass = .45;
% 通過域利得
```



（a）インパルス応答



（b）周波数応答

図3.27　FIRフィルタの設計例

```
gainPass = 1;
% 阻止域端
edgeStop = .55;
% フィルタ設計（インパルス応答）
h = firpm(nOrder,...
[0 edgePass edgeStop 1], ...
[gainPass gainPass 0 0]);
% FIR 直接型構成
Hd = dfilt.dffir(h);
% インパルス応答表示
impz(Hd)
% 周波数応答表示
freqz(Hd)
```

firpm関数の第2引き数[0 .45 .55 1]の要素は，それぞれ，正規化角周波数[0, $(\pi/2 - \pi/20)$, $(\pi/2 + \pi/20)$, π]に対応している．この各角周波数点における振幅の理想値を次の引き数［1 1 0 0］で与えている．詳しくは，ヘルプまたはドキュメントを参照されたい．**図3.27**に設計したフィルタのインパルス応答と周波数応答を示す．なお，impz関数，freqz関数には直接，インパルス応答hを引き数に渡してもよい．

この例題ではfirpm関数に与える次数を明示的に与え，通過域と阻止域のリプルを最適化したが，MATLABでは，あらかじめリプルを仕様として与えてフィルタの次数を推定するためのfirpmord関数が提供されている．例えば，次のように利用できる．


```
% 通過域端
edgePass = .45;
```

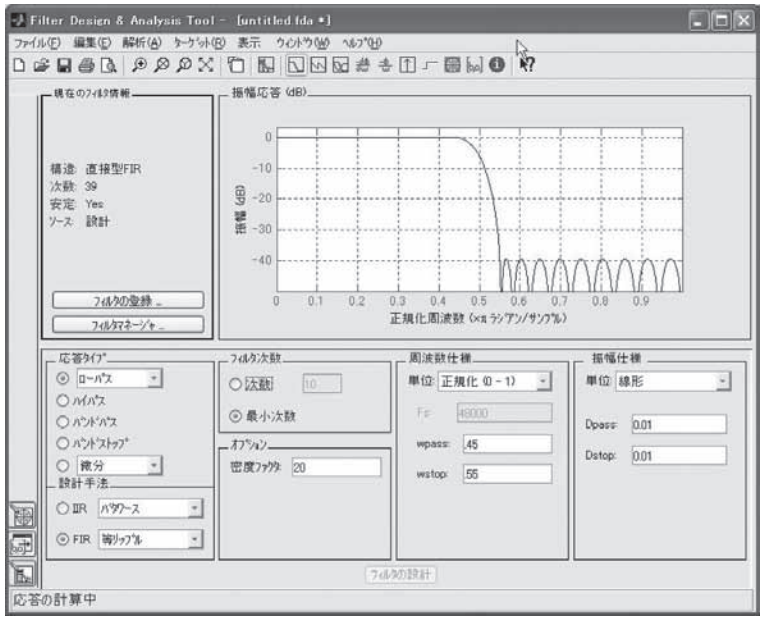


図3.28　fdatoolの使用例

```
% 通過域利得
gainPass = 1;
% 通過域許容リプル
deltaPass = .01;
% 阻止域端
edgeStop = .55;
% 阻止域許容リプル
deltaStop = .01;
% 次数推定
[nOrder,F,A] = ...
  firpmord([edgePass edgeStop],...
  [ gainPass 0 ],[deltaPass deltaStop]);
% フィルタ設計（インパルス応答）
h = firpm(nOrder, F, A);
```

また，ほかのFIRフィルタ設計関数として，firls，fir1，fir2，maxflatが提供されている．IIRフィルタの設計には，butter関数，cheby1関数，cheby2関数，ellip関数，maxflat関数を利用できる．

だが，何よりもまずfdatoolの利用が簡単である（**図3.28**）．そのほか，fvtool，sptoolといったツール群が用意されているので，ヘルプやドキュメントで確かめてほしい．

実習3.10　**フィルタ設計**

M-file：practice03_10.m

フィルタの仕様を変えて，さまざまなフィルタを設計してみよう．

章末問題

問題3.1　**音の周波数解析**（MATLAB演習）

次のコマンドを確かめてみよう．

```
load gong
wavplay(y,Fs)
figure(1)
freqz(y,1,length(y),Fs)
figure(2)
specgram(y,128,Fs)
```

さらに，1行目で`load chirp`として，結果を比較してみよう．

問題3.2　**窓関数の設計**（MATLAB演習）

MATLABの`wintool`関数で，各種窓関数を設計してみよう．

問題3.3　**窓関数の応用**（MATLAB演習）

連続信号$x(t) = \cos(2\pi t + \pi/2) + 2\cos(4\pi t)$に対し，標本化周期1/8［sec］で標本化し，離散時間信号を与えよう．また，適当な窓を掛けてスペクトラムを観察してみよう．興味があれば，窓長，DFT点数などを変えてみよう．

問題3.4　**移動平均フィルタ**（MATLAB演習）

$N(>3)$点移動平均フィルタ

$$h[n] = \frac{1}{N},\quad n = 0,1,\cdots,N-1$$

の振幅応答と位相応答を確かめてみよう．

問題3.5　**フィルタの利得**（MATLAB演習）

$h[n] = \delta[n] + \delta[n-1] + \delta[n-2]$で与えられるフィルタと例題3.10の3点移動平均フィルタを，振幅応答と位相応答を用いて比較してみよう．また，同じ入力$x[n]$に対する処理結果の違いも比較してみよう．

問題3.6　**フィルタ処理**（MATLAB演習）

連続信号$x(t) = \cos(2\pi t + \pi/2) + 2\cos(4\pi t)$に対し，サンプリング周期$T_S = 1/8$［sec］で標本化を施し，離散時間信号を生成しよう．ただし，時間幅は8［sec］とする（64点）．さらに，4点の移動平均フィルタを掛けてスペクトラムを確認してみよう．

問題3.7　**フィルタ設計**（MATLAB演習）

次のように生成した信号x

```
load gong
x1 = y;
load chirp
x2 = y;
x = x1(1:length(x2)) + x2;
```

に対して，例題3.13で設計したフィルタを使ってフィルタリングを施し，例題3.10のようにフィルタ前後の信号の周波数特性を比べてみよう．また，wavplay関数を用いてそれぞれの信号を聞き比べてみよう．

参考文献

(1) 尾知博；シミュレーションで学ぶディジタル信号処理，CQ 出版社，2001年．
(2) 佐川雅彦，貴家仁志；高速フーリエ変換とその応用，昭晃堂，1992年．

第4章

画像処理の基礎

本章では，MATLABにおける画像処理の基本的な操作法や便利な関数について紹介し，ガンマ補正やヒストグラム均等化などの画素処理や，色空間変換などの色情報処理について簡単に解説しよう．

4.1　画像・映像表現の概要

画像・映像信号処理について述べる前に，まず，これらの信号がどのように標本化されているかを考えておかなければならない．

多次元信号の標本化における自由度については，第8章以降に解説することとし，ここでは，1次元信号処理の拡張から与えられる方形標本化について概説する．また，今後のために多次元信号の配列表現を確認しておく．

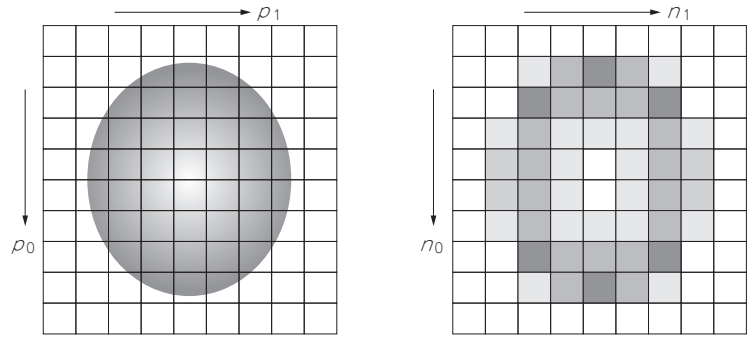
● 方形標本化の概要

(1) 方形標本化とは

センサを配列状に並べたCCDやCMOSなどのセンサ・アレイを用いることで，センサ上に写像された光の強さの分布，すなわち画像を標本化できる．このような過程を経て得られる画像信号は，**図4.1**に示されるように空間的に離散的な信号となる．ここで取得された一つ一つの値は，1次元の場合と同様に標本値と呼ばれるが，画像の場合は特に**画素値**と呼ばれることが多い．

センサ上に映る光の強さの分布は，垂直方向の連続変数p_0と水平方向の連続変数p_1を用いて，$x(p_0, p_1)$と表現される．一方，標本化後の離散信号は垂直方向の離散変数n_0と水平方向の離散変数n_1を用いて$x[n_0, n_1]$と表現され，$x(p_0, p_1)$とは$x[n_0, n_1] = x(n_0P_0, n_1P_1)$のように関係づけることができる．ここで，$P_0$，$P_1$は空間的な標本化周期を表している．このように各連続変数p_0，p_1の方向に対して独立な標本化周期を用いると，**図4.1**に示されるように標本位置が各変数軸と直角に交わる格子上に位置することとなる．このような標本化は**方形標本化**と呼ばれ，もっとも素直な方法である．

なお，変数p_0，p_1および標本化周期P_0，P_1には空間的な単位［dpi(dot per inch)など］があるが，画像の場合，センサの大きさや画素数によってこれらの値は異なり，不明なままでも問題にならな



（**a**）標本化前　　（**b**）標本化後

図4.1　センサ・アレイによる画像の標本化

いことが多い．音声信号の標本化周期 T［sec］と異なり，大型テレビや携帯端末などで，空間標本化周期を無視した信号再構成が日常的に行われていることを想像してもらえればよい．むしろ，画像の場合は画素数が重視される．

(2) 配列表現

標本化後の信号は離散的な変数を持つため，配列として表現される．画像の場合，**図4.2**に示されるように，左上を原点として水平方向を右向きに，垂直方向を下向きに座標軸をとることが多い．配列を行列で表現しやすいこと，従来の撮像管などの電子ビームの走査に対応することが主な理由である．例えば，垂直解像度×水平解像度＝$N_0 \times N_1$の画像の場合，各画素値を用いて$N_0 \times N_1$行列$\mathbf{X}$として

$$\mathbf{X} = \begin{pmatrix} x[0,0] & x[0,0] & \cdots & x[0,N_1-1] \\ x[1,0] & x[1,1] & \cdots & x[1,N_1-1] \\ \vdots & \vdots & \ddots & \vdots \\ x[N_0-1,0] & x[N_0-1,1] & \cdots & x[N_0-1,N_1-1] \end{pmatrix}$$

のように表現される．

(3) 量子化

なお，ディジタル信号として画像を取得するためには，標本化後に量子化を施し，画素値の離散化を行う必要がある．1画素当たりb［bpp］[注4-1]の画像の階調数は$L = 2^b$となる．したがって，モノクロ画像の場合，総ビット数Sは，

$$S = N_0 \times N_1 \times b[\text{bits}]$$

となる．

アニメのような限定色画像の場合，**インデックス方式**を用いることでデータ量を減らすことができる．インデックス方式とは，色情報をもったカラー・マップとそのインデックスからなる配列によってカラー画像を表現する方式である．一方，RGBの画素値をもつ配列によって表現する方式を

注4-1：bits per pixel.

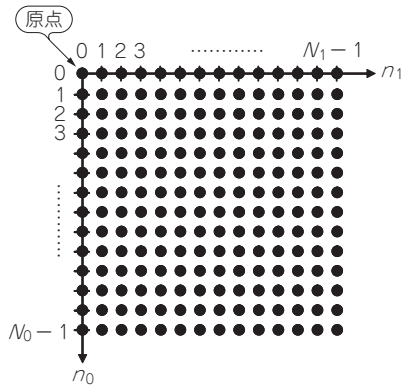


図4.2　画像の配列表現

トゥルー・カラー方式と呼ぶ．以降，断りなくトゥルー・カラー方式を前提とする．

(4) 映像信号

センサ・アレイ上へ写像された光の強さの分布の時間変化を取得することで，映像信号(動画像)を得ることができる．もとの連続信号は，空間変数p_0，p_1に時間変数tを加えることで，$x(p_0, p_1, t)$と表現される．一方，標本化後の離散信号は空間変数n_0，n_1に時間変数kを加えることで，$x[n_0, n_1, k]$と表現され，$x(p_0, p_1, t)$とは$x[n_0, n_1, k] = x(n_0P_0, n_1P_1, kT)$のように関係づけられる．ここで，$T$は標本化周期を表している．この関係では，標本化周期が各軸に独立に与えられているため，3次元の方形標本化ということができる．1画素当たりbビット(b bpp)のモノクロ映像の場合，1秒当たりのビット数(ビット・レート)Rは，

$$R = N_0 \times N_1 \times \frac{1}{T} \times b \ [\text{bps}]$$ 注4-2

となる．

4.2　MATLABでの画像処理

ここでは，MATLABで画像信号を取り扱うための基本的な操作についてまとめる．画像の表現や入出力がMATLAB上による処理に与える影響が大きいため，特に画像データのハンドリングに焦点を当てよう．

● 画像の表現と入出力

(1) 画像の表現

MATLABはさまざまなデータ・タイプを定義しているが，画像データに対しては特に次のデータ・タイプがよく利用される．

- `logical`：バイナリ　{true, false}

注4-2：bits per second.

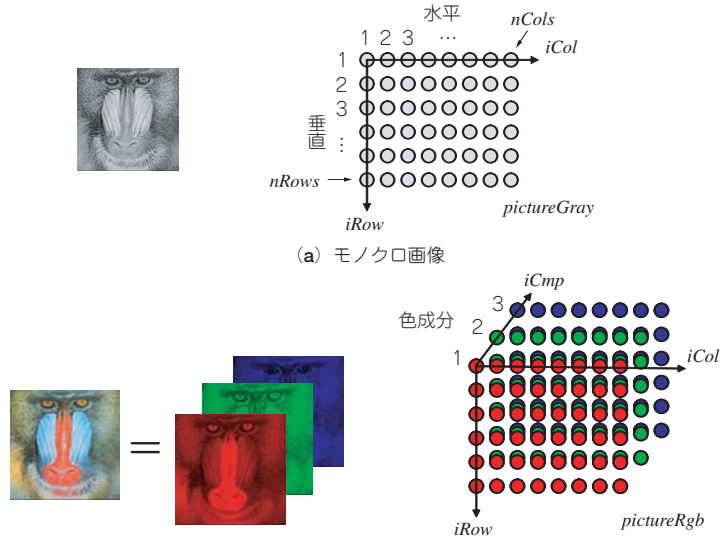


図4.3　MATLAB上の画像表現

- `uint8`：符号なし8ビット整数［0 255］
- `uint16`：符号なし16ビット整数［0 65535］
- `double`：倍精度浮動小数点数［0.0 1.0］

データ量は，`logical` = `uint8` < `uint16` < `double`の順で大きくなるが，処理の制約は`double` < `uint16` < `uint8` < `logical`の順で大きくなる．

モノクロ画像は，輝度成分のみの配列として表現できる．また，カラー画像は，原則として赤(R)成分，緑(G)成分，青(B)成分からなる配列として表現できる．**図4.3(a)**に示すように，MATLAB上では，モノクロ画像は左上を頂点とする2次元配列として表現される．また，**図4.3(b)**に示すように，カラー画像はR成分の左上を頂点とする配列として表現される．ここで，MATLAB上の配列のインデックス番号は1から始まることに注意されたい．

以降，MATLABのプログラム例では，

- *iRow*：行インデックス
- *iCol*：列インデックス
- *iCmp*：色成分インデックス

という変数を用いて，すべて1から開始するものとする．一方，文章中の説明ではほかの文献との整合性をとるため，**図4.2**に示すとおり，配列の原点は0から開始するものとする．変数の対応関係を**表4.1**に示す．なお，$nRows = N_0$，$nCols = N_1$という関係が成り立つ．モノクロ画像*pictureGray*

表4.1　MATLABプログラムと解説の変数の対応

iRow	*iCol*	*iCmp*
$n_0 + 1$	$n_1 + 1$	1(R)，2(G)，3(B)

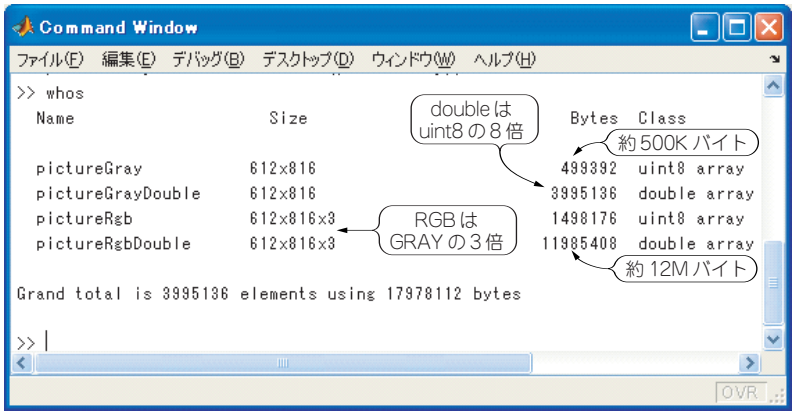


図4.4　MATLAB上のデータ・サイズの比較

の*iRow*行*iCol*列の画素値は，

```
pictureGray(iRow,iCol)
```

と表現され，カラー画像*pictureRgb*の*iRow*行*iCol*列*iCmp*成分の画素値は，

```
pictureRgb(iRow,iCol,iCmp)
```

と表現される．

例題4.1　**データ・サイズ**

さまざまな表現形式の配列について，データ・サイズを比較してみよう．

解

次のコマンドを実行した結果を**図4.4**に示す．

```
clear all;
nRows = 612;
nCols = 816;
pictureGray = ... % モノクロuint8
zeros(nRows,nCols,'uint8');
pictureGrayDouble = ... % モノクロdouble
zeros(nRows,nCols,'double');
pictureRgb = ... % カラーuint8
zeros(nRows,nCols,3,'uint8');
pictureRgbDouble = ... % カラーdouble
zeros(nRows,nCols,3,'double');
clear nRows nCols
whos
```


結果より，pictureRgbDoubleはpictureGrayの24倍もの記憶容量を要することが分かる．

メモリの有効利用という観点から，表現形式の選択が重要であるといえる．

実習4.1　**データ・サイズ**

M-file：practice04_1.m

大きさや形式を変えてデータ・サイズを比較してみよう．また，uint16型やlogical型についても調べてみよう．バイナリ画像は次のように与えると良い．

```
% バイナリlogical
pictureBinary = ...
  logical(zeros(nRows,nCols));
```

● 画像の入力

画像ファイルの入力には，imread関数とfread関数が利用できる．それぞれの機能を以下にまとめる．

- imread：JPEG，TIFFなど，標準形式に対応したファイル入力
- fread：RAW(生)など，一般的なバイナリ・データのファイル入力

(1) imread

imread関数は，その引き数にファイル名を指定するだけで，画像ファイルを配列として読み込む．カラー画像の場合，通常，RGBカラー空間上の3次元配列(サイズ$N_0 \times N_1 \times 3$，uint8型)としてワークスペースに取り込まれる．詳細については，ドキュメントを参照されたい．なお，カラー画像のままではメモリの無駄遣いとなるので，モノクロ画像に変換して処理したい場合がある．この場合，次の変換式で各画素ごとの輝度成分Yを計算することができる．

$$x_Y = 0.2989x_R + 0.5870x_G + 0.1140x_B \quad \cdots\cdots (4.1)$$

なお，Image Processing Toolboxでは，この変換のためにrgb2gray関数が用意されている．

例題4.2　**imreadの使用例**

firenzeRgb.jpgというカラー画像を読み込み，モノクロ画像に変換しよう．

解

以下に，MATLABによる処理例を示す．

```
% カラー画像の読み込み
pictureRgb = imread('./data/firenzeRgb.jpg');
% モノクロ画像への変換
pictureGray = uint8( ...
0.2989 * pictureRgb(:,:,1) + ... % R
0.5870 * pictureRgb(:,:,2) + ... % G
0.1140 * pictureRgb(:,:,3) % B
);
```

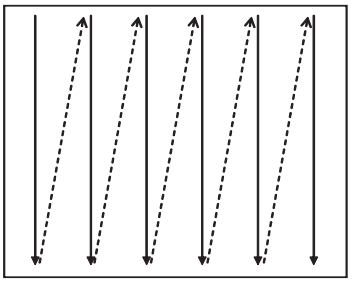


図4.5
fread 関数による読み込み/fwrite による書き込み順序

```
% メモリ節約のため利用しない配列はクリアする
clear pictureRgb;
```

(2) fread

fread関数は，汎用的なファイル読み込みの関数である．ヘッダ情報のないRAW画像や独自のフォーマットで記録された画像データの読み込みに使える．以下に，fread関数の使用例を示そう．

```
% 画像サイズ　高さ×幅
frameSize = [612 816];
nPixels = prod(frameSize);
% 読み込み精度をuint8 に設定
precision = 'uint8=>uint8';
% 読み込みモード'r' にてファイルをオープン
fileId = fopen('./data/firenze.yyy', 'r');
% 画像データを列ベクトルとして読み込む
pictureGray = ...
fread(fileId, nPixels, precision);
fclose(fileId);
% 列ベクトルを配列に変換
pictureGray = ...
reshape(pictureGray, frameSize);
```

fread関数は，データを列ベクトルとして読み込むことに注意してほしい．今，ファイルfirenze.yyyは，1画素8ビットの画像データを左上から**図4.5**のように走査したバイト列が記録されていると仮定している．これは行列操作を基本とするMATLABの特徴的な点で，水平走査に慣れた画像処理技術者は注意が必要である．水平走査(行ベクトル)を優先して保存されたファイルの場合，reshape後に配列を転置する必要がある．

● 画像の表示

ワークスペースに取り込んだ画像や処理を施した画像を実際に目で確かめたい場合，image関数を使って画像を表示することができる．Image Processing Toolbox では，image関数の代わりに表示

の最適化を行うimshow関数や，さらに高機能なimtool関数を利用できる．

uint8型の場合，画素値の範囲が0～255，double型の場合，画素値の範囲が0.0～1.0に設定されていれば，画面上に適切に表示される．

例題4.3　画像表示

ファイルfirenzeRgb.jpgを読み込んで，カラー画像とモノクロ画像の表示を行ってみよう．また，double型に変換したカラー画像についても表示してみよう．

解

以下に，MATLABによる処理例を示す(Image Processing Toolboxがある場合，image関数をすべてimshow関数で置き換えることをお勧めする)．

```
% 画像ファイルの読み込み
pictureRgb = imread('./data/firenzeRgb.jpg');
% カラー画像(uint8) の表示
figure(1);
image(pictureRgb);
axis image; axis off;
% モノクロ画像への変換
pictureGray = ...(省略)% 例題4.2 を参照
% モノクロ画像(uint8) の表示
figure(2);
image(pictureGray); colormap(gray(256));
axis image; axis off;
% 倍精度への変換
pictureRgbDouble = ...
double(pictureRgb)/255.0;
% カラー画像(double) の表示
figure(3);
image(pictureRgbDouble);
axis image; axis off;
```

imshow関数とimtool関数では，モノクロ画像とカラー画像の違いを意識せずに利用できる．一方，image関数については，モノクロ画像表示のときにカラー・マップの指定と軸(axis)の設定を行わないと，適切に表示されない．

実習4.2　画像表示

M-file：practice04_2.m

例題において，double型のスケーリング(255.0による除算)を行わないとどのように表示されるかを確かめてみよう(Image Processing Toolboxには，画像データを double型に変換するim2dobule関数が用意されている．この関数を利用すると，スケーリングも同時に自動で行われる)．

● 画像の出力

ワークスペース上の画像データをファイルに出力するため，imwrite関数とfwrite関数が利用できる．それぞれの機能を以下にまとめる．

- imwrite：JPEG，TIFF など，標準形式に対応したファイル出力
- fwrite：RAW(生)など，一般的なバイナリ・データのファイル出力

(1) imwrite

imwrite関数は，カラー/モノクロの判別やuint8/doubleの判別も自動で行い，ワークスペース上の画像データを指定されたフォーマットでファイルに書き込む．

例題4.4　画像出力

firenzeRgb.jpgを読み込み，TIFF形式にて保存しよう．また，double型のモノクロ画像に変換し，JPEG形式にて保存しよう．

解

以下に，MATLABでの処理例を示す．

```
% 画像ファイルの読み込み
pictureRgb = imread('./data/firenzeRgb.jpg');
% カラー画像(uint8) のTIFF 形式での書き出し
imwrite(pictureRgb, 'firenzeRgb.tif');
% モノクロ画像(double) への変換
pictureGrayDouble = ...
double(rgb2gray(pictureRgb))/255.0;
% モノクロ画像(double) のJPEG 形式での書き出し
imwrite(pictureGrayDouble, ...
'data/firenzeGray.jpg');
```

実習4.3　画像出力

M-file：practice04_3.m

JPEG形式，TIFF形式以外のフォーマットについても試してみよう．また，double型への変換の際のスケーリングを行わないと，どのように保存されるかを確かめてみよう．

(2) fwrite

fwrite関数の使用例を以下に示そう．

```
% バイナリ書き込みモード(wb) でファイルをオープン
fileId = fopen('firenze.yyy','wb');
% pictureGray をuint8 にて保存
fwrite(fileId, pictureGray, 'uint8');
% ファイルをクローズ
fclose(fileId);
```

fwrite関数は，指定されたファイルに配列を記録する．上記の例では，バイナリ書き込みモードとuint8の精度が選択されているので，配列はバイト列として保存される．このとき列ベクトルが優先され，**図4.5**に示す順序となることに注意されたい．水平走査(行ベクトル)を優先したい場合，配列をあらかじめ転置してからfwrite関数に渡す必要がある．

4.3 MATLABによる映像処理

ここでは，MATLABで映像データを取り扱うための基本的な操作についてまとめる．

● 映像の表現

MATLABでは，映像データをマルチフレーム方式かムービー(MOV)方式のいずれかで取り扱うと便利である．なお，本書のMATLABのプログラム例では，フレーム番号を表すために

☒*iFrame*：フレーム・インデックス

という変数を用い，1から開始するものとする．文章中の説明では，ほかの文献との整合性をとるため，フレーム時刻kを0から開始するものとする．すなわち，$iFrame = k + 1$と関係する．

(1) マルチフレーム方式

マルチフレーム方式は，映像を4次元配列として表現する方法である．RGBカラー画像が3次元配列として表現されるが，これに時間軸を加えた3次元配列の列(画像シーケンス)である．マルチフレーム方式では輝度成分のみのモノクロ映像も4次元配列として表現する．RGBカラー映像*multiFramesRgb*の*iRow*行*iCol*列*iCmp*成分*iFrame*フレームの画素値は

```
multiFramesRgb(iRow,iCol,iCmp,iFrame)
```

と表現され，モノクロ映像*multiFramesGray*の*iRow*行*iCol*列*iFrame*フレームの画素値は

```
multiFramesGray(iRow,iCol,1,iFrame)
```

と表現される．

(2) ムービー方式

ムービー(MOV)方式では，**図4.6**に示すように2次元(モノクロ)配列，あるいは3次元(RGBカラー)配列をメンバとしてもつ*frame*構造体の列として映像を表現する．*frame*構造体は，*cdata*と*map*というメンバからなり，それぞれが画像配列とカラー・マップを保持する．モノクロ映像*framesSeqGray*の*iFrame*フレーム*iRow*行*iCol*列の画素値は

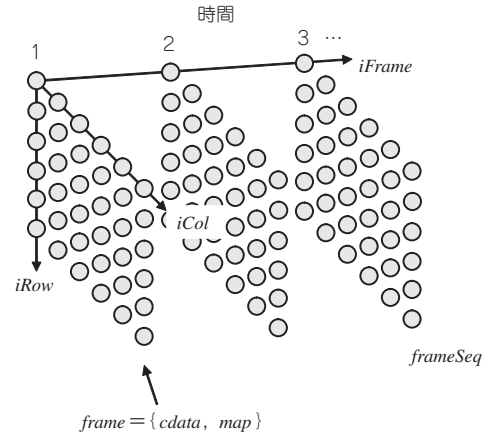


図4.6
ムービー（MOV）方式

```
frameSeqGray(iFrame).cdata(iRow,iCol)
```

と表現され，RGBカラー映像*frameSeqRgb*の*iFrame*フレーム*iRow*行*iCol*列*iCmp*成分の画素値は

```
frameSeqRgb(iFrame).cdata(iRow,iCol,iCmp)
```

と表現される．RGBカラー画像の場合，*colormap*は無視されるので必要ないが，モノクロ画像の場合は，適切な*colormap*として`gray`を与える必要がある（すなわち，インデックス画像として扱われる）．

● 映像の入力

映像ファイルの入力には，`aviread`関数の利用が便利である．場合によっては，`imread`関数や`fread`関数を利用した映像ファイルの入力も必要となる．それぞれの機能を以下にまとめる．

- `aviread`：AVI形式に対応したファイル入力．ワークスペース・データとしてMOVを作成．
- `imread`：JPEG，TIFFなど，標準形式のフレーム画像ファイル入力に利用可能．`im2frame`関数と組み合わせてMOV への変換も可能．
- `fread`：YCbCr（生）など，一般的なバイナリ・データのファイル入力．

`fread`関数の利用例については，後ほど例題4.9で紹介する．

（1）aviread

`imread`関数は，その引き数にAVI形式のファイル名を指定するだけで，映像をムービー（MOV）データとして読み込む．`frame2im`関数と合わせて利用すると，フレーム画像（*frame*構造体）から画像データ（*cdata*）を配列として抽出できる．以下に使用例を示そう．

```
% AVI 形式ファイルを読み込み
frameSeq = aviread('calcio.avi');
% 最初のフレームから画像データを抽出
pictureRgb = frame2im(frameSeq(1));
% 抽出した画像データを表示
```

```
image(pictureRgb);
```

(2) imread

imread関数は，フレームごとに分かれて保存されている映像データの読み込みに利用できる．im2frame関数と組み合わせて利用することでMOVへの変換も可能である．Image Processing Toolboxがある場合，いったんマルチフレーム方式で映像を保存するとimmovie関数によってMOVへ変換できる．

例題4.5　フレーム画像列の読み込み

フレーム画像列calcio000.jpg，calcio001.jpg，…，calcio149.jpgをMOVデータに変換しよう．

解

以下に，MATLABによる処理例を示す．

```
nFrames = 150;
for iFrame = 1:nFrames
  fileName = ...
    sprintf('calcio%03d.jpg', iFrame-1);
  pictureRgb = imread(fileName);
  frameSeq(iFrame) = ...
    im2frame(pictureRgb);
end
```

Image Processing Toolboxがある場合は，次のように処理することも可能である．

```
nFrames = 150;
for iFrame = 1:nFrames
  fileName = ...
    sprintf('calcio%03d.jpg', iFrame-1);
  pictureRgb = imread(fileName);
  multiFramesRgb(:,:,:,iFrame) = ...
    pictureRgb;
end
frameSeq = immovie(multiFramesRgb);
```

● 映像の表示

ワークスペース上のMOVデータは，movie関数を使って映像を表示できる．

例題4.6　映像表示

フレーム・レートが30 [fps] [注4-3]の映像ファイルcalcio.aviを読み込み，movie関数により3

回繰り返して表示してみよう．

解

以下に，MATLABでの処理例を示す．

```
% ファイル名
fileName = 'calcio.avi';
% AVI ファイル情報の取得
fileInfo = aviinfo(fileName);
% フレーム・レート情報の取得
frameRate = fileInfo.FramePerSecond;
% AVI ファイルの読み込み
frameSeq = aviread(fileName);
% 繰り返し回数
nRepeats = 3;
% 映像表示
movie(frameSeq, nRepeats, frameRate);
```

大きなデータはメモリ領域を圧迫するので，次に紹介するAVI形式による出力を行い，ほかのツールで表示することをお勧めする．movieは簡易ビューワとして利用するとよい．なお，image関数とdrawnow関数の併用でもフレーム画像列を映像として表示できる．ただし，フレーム・レートは制御できない．

実習4.4　映像表示

M-file：practice04_4.m

フレーム画像列calcio000.jpg，calcio001.jpg，…，calcio149.jpgを読み込んで，movie関数により表示してみよう．繰り返し回数やフレーム・レートを変えて表示を確かめてみよう．

● 映像の出力

ワークスペース上の映像データをファイルに出力するため，movie2avi関数，およびavifile関数とaddframe関数の組み合わせを利用できる．imwrite関数によるフレーム画像列の出力や，汎用のfwriteも利用できる．それぞれの機能を以下にまとめる．

- movie2avi：MOVデータをAVI形式でファイル出力
- avifile，addframe：フレーム画像列をAVI形式にてファイル出力
- imwrite：JPEG，TIFFなど，標準形式のフレーム画像ファイル出力に利用可能．frame2im

注4.3：frames per second.

関数と組み合わせてMOVからの変換も可能.

- fwrite：YCbCr(生)など，一般的なバイナリ・データのファイル出力

以下では，avifile関数とaddframe関数を併用する方法について解説する.

avifile関数は，MATLABワークスペース上にAVI形式のファイル・オブジェクトを生成する．addframe関数はこのAVI形式のファイルにフレーム画像(*frame*構造体)を追加する機能をもつ．オブジェクト指向プログラミングになじみのある読者には，avifile関数はコンストラクタ，addframe関数はメンバ関数(メソッド)と説明すると理解しやすいだろう．以下では，映像信号の入出力の具体例として，フレーム画像列calcio000.jpg，calcio001.jpg，…，calcio149.jpgを読み込んでAVIファイルを作成するプログラムを掲載する.

```
nFrames = 150;
frameRate = 30;
% 出力ファイル
fileNameOut = 'calcio.avi';
clear mex;
aviObj = avifile(fileNameOut,...
  'FPS',frameRate,...
  'COMPRESSION','None');
% フレームごとの処理
for iFrame = 1:nFrames
% 現フレームの読み出し
fileNameIn = sprintf('calcio%03d.jpg', ...
  iFrame-1);
picture = imread(fileNameIn);
% フレームの出力
aviObj = addframe(aviObj,...
  im2frame(picture));
end
aviObj = close(aviObj);
```

4.4 画素処理

ある用途に適した画像を得るために原画を処理することを**画像強調**という．画像強調は，用途ごとに適切に選択しなければならない．例えば，X線画像に適した処理が，火星探索機で撮影した画像に適するとは限らない．画像強調処理は，大きく分けて**空間領域処理**と**周波数領域処理**に分類できる．ここで解説する**画素処理**は，空間領域処理の一種で原画の各画素ごとの輝度値や色度の変換を行う．空間領域処理には，ほかに**近傍処理**がある．近傍処理はマスク処理，あるいはフィルタ処理

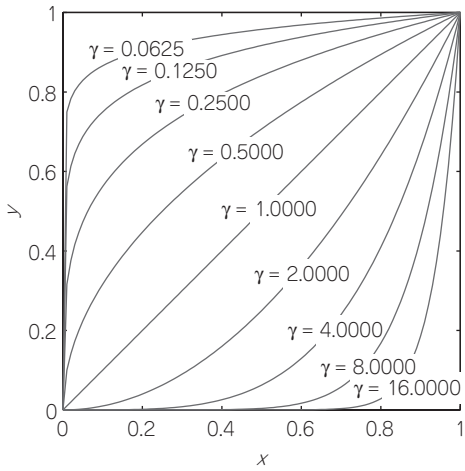


図4.7
累乗則変換

とも呼ばれる．近傍処理については次章で詳しく解説する．

輝度値変換を行う画素処理は，

$$y = T(x) \quad \cdots\cdots (4.2)$$

と表現できる．ここでxは入力輝度値，yは出力輝度値であり，Tが変換を表す．変換Tとしてネガポジ変換やしきい値処理など，さまざまな関数を選択できる．Tとして階調数を減らす関数を選べば量子化となるが，これも一種の画素処理といえる．以下では，画素処理の代表例として**累乗則変換**と**ヒストグラム均等化**について解説しよう．

● 累乗則変換，ガンマ補正

(1) 累乗則変換

いま，話を簡単にするため画素値の範囲を［0.0 1.0］と仮定する．累乗則変換は

$$T(x) = x^{\gamma} \quad \cdots\cdots (4.3)$$

と表現できる．**図4.7**に式(4.3)で与えられる関数のグラフを示そう．γはパラメータで，この選択によって以下の機能が与えられる．

- $\gamma < 1$：暗部を明るくする効果
- $\gamma > 1$：明部を暗くする効果

例題4.7　**累乗則変換**

ファイル`firenzeRgb.jpg`を読み込んでモノクロ画像に変換し，$\gamma = 0.25$を用いて累乗則変換を施してみよう．

解

ファイル読み込みとモノクロ画像への変換の後のMATLABによる処理例を，以下に示す．

```
parameterGamma = 0.25;
:(省略)
```

（a）処理前

（b）処理後

図4.8　累乗則変換の例

```
% モノクロ画像をdouble に変換
pictureGrayDouble = ...
  double(pictureGray)/255.0;
% 累乗則変換
pictureCorrected = ...
  pictureGrayDouble.^parameterGamma;
  :（省略）
```

モノクロの原画像を**図4.8**(**a**)に，$\gamma = 0.25$の累乗則変換を施した画像を**図4.8**(**b**)に示す．明るい画像に変換されることが確認できる．

Image Processing Toolbox がある場合，累乗則変換にimadjust関数を以下のように用いることができる．

```
pictureCorrected = ...
  imadjust(pictureGray,[],[],...
           parameterGamma);
```

imadjust関数では，double型への変換は必要ない．

実習4.5　**累乗則変換**

M-file：practice04_5.m

$\gamma = 0.25$以外の値でも累乗則変換を試してみよう．

(2) ガンマ補正

累乗則変換は**ガンマ補正**にも利用される．**図4.9**にガンマ補正の概略図を示す．一般に，表示機器において信号電圧v_dと表示輝度B_dの間には，おおよそ

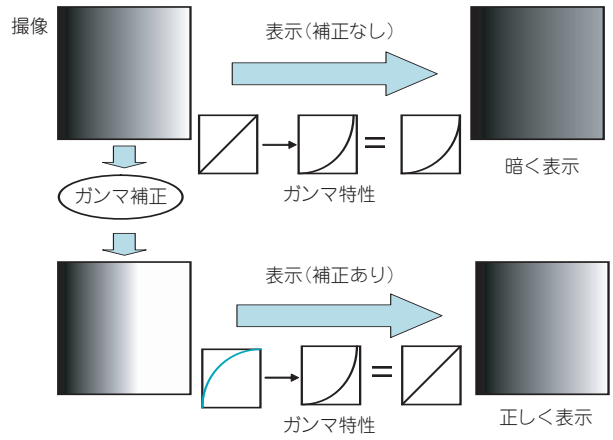


図4.9　ガンマ補正

$$B_d \propto v_d^{\gamma_d}$$

という関係が成り立つ．CRTの場合，$\gamma_d \simeq 2.2 \sim 2.5$といわれている．このような特性を**ガンマ特性**と呼ぶ．一方，撮像機器においても照度B_cと信号電圧v_cの間に

$$v_c \propto B_c^{\gamma_c}$$

という関係が成り立つ．CCDの場合は$\gamma_c \simeq 1.0$，ビジコン（撮像管）の場合は$\gamma_c \simeq 1.7$，といわれている．

このような表示機器と撮像機器を接続しても，正しい階調による表示は不可能である．この系全体の入出力が比例するように施す処理をガンマ補正とよぶ．なお，ITU-R BT.709 勧告ではガンマ補正を

$$y = \begin{cases} 4.5x & x < 0.018 \\ 1.099x^{0.45} - 0.099 & x \geq 0.018 \end{cases} \quad \cdots\cdots (4.4)$$

と規定している．表示機器のガンマ値γ_dを約2.2と仮定し，立ち上がり部分に若干の線形性を与えた関数となっている．

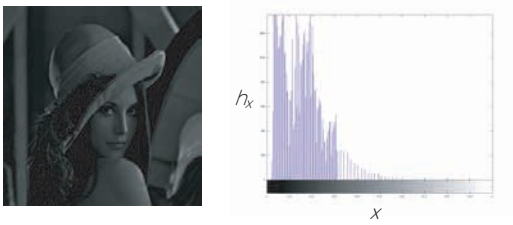
● ヒストグラム処理

累乗則変換は，γの選択によって画像の明暗を制御でき，単純ではあるが視覚的な効果が大きい．ただし，個々の画像について適切なγを設定する必要がある．明るい画像の明度を落とす，暗い画像の明度を上げる，といった処理を自動で適応的に行えれば，パラメータ設定の煩わしさがなく便利である．**ヒストグラム均等化**は，与えられた画像に適した画素値変換$T(x)$を作成し，どのような明るさの画像でも類似した明るさの処理結果を与える．

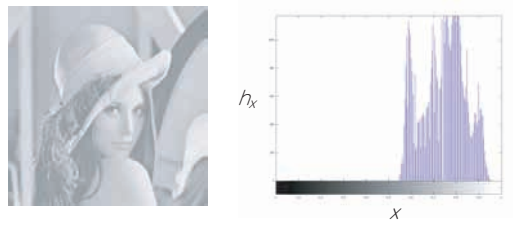
(1) ヒストグラム

話を簡単にするため，モノクロ画像を考える．ヒストグラムとは，画像内の輝度値xの度数h_xをグラフ表示したものである．8ビット・モノクロ画像の場合，xは256種類の値を取りうる．**図4.10**に8ビット・モノクロ画像のヒストグラムの例を示そう．

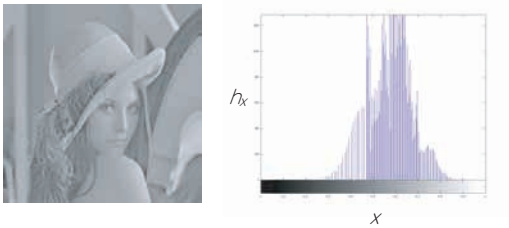
xの画素数をh_xとすると，正規化度数p_xが



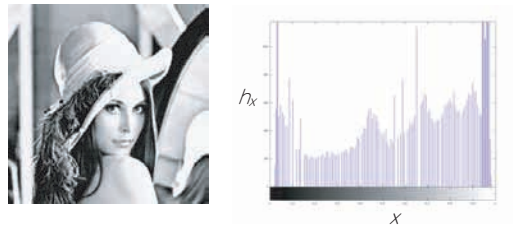
(a) 暗い画像



(b) 明るい画像



(c) 低コントラスト画像



(d) 高コントラスト画像

図4.10　画像のヒストグラムの例

$$p_x = \frac{h_x}{N} \quad \cdots\cdots (4.5)$$

と定義される．ここで，Nは総画素数を意味する．正規化度数p_xは，必ず0以上1以下の値となる．

(2) ヒストグラム均等化

ヒストグラム均等化は，処理後のヒストグラムが一様になるような画素値変換$T(x)$を作成する．結果として，明るい画像に対しても，暗い画像に対しても同じようなヒストグラムをもつ出力が得られる．

ヒストグラム均等化の変換$T(x)$は，正規化度数p_xから

$$y = T(x) = (L-1)\sum_{u=0}^{x} p_u, \quad x = 0,1,2,\cdots,L-1 \quad \cdots\cdots (4.6)$$

と与えればよい．Lは画素値の階調数であり，8ビットの場合は256となる．必要に応じてyの整数化を行う．ここではxを符号なしの整数値であると仮定したが，実数値に対する変換$T(x)$を得ることは容易である．

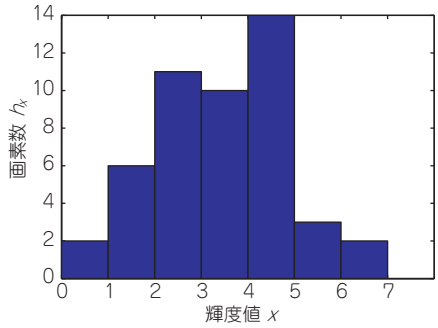
式(4.6)は，xが連続で，その確率密度関数$p(x)$が与えられたとき，その累積密度関数$c(x) = \int_0^x p(u)\,du$が一様分布を与える変換となることに基づいている．ディジタル画像では，xが離散であるため正当性はないが，実用的である点から広く利用されている．

例題4.8　ヒストグラム均等化

図4.11(a)の配列に対してヒストグラム均等化を施してみよう．ただし，$L = 8$としよう．また，出力yを小数点以下第1位で四捨五入して整数化しよう．

4	3	3	5	5	3	3	2
2	1	2	4	6	6	4	1
3	3	4	4	4	4	3	1
4	4	4	5	3	2	4	2
4	4	2	4	3	2	3	1
2	2	1	2	2	1	0	0

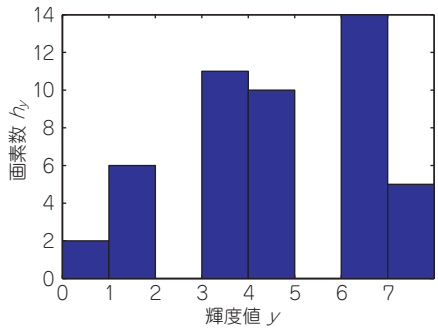
（**a**）処理前の配列



（**b**）処理前のヒストグラム

6	4	4	7	7	4	4	3
3	1	3	4	7	7	6	1
4	4	6	6	6	6	4	1
6	6	6	7	4	3	6	3
6	6	3	6	4	3	4	1
3	3	1	3	3	1	0	0

（**c**）処理後の配列



（**d**）処理後のヒストグラム

図4.11
ヒストグラム均等化の例

表4.2　度数分布表

x	0	1	2	3	4	5	6	7
h_x	2	6	11	10	14	3	2	0
p_x	2/48	6/48	11/48	10/48	14/48	3/48	2/48	0

表4.3　ヒストグラム均等化の画素値変換表

x	0	1	2	3	4	5	6	7
y	0	1	3	4	6	7	7	7

解

与えられた配列より総画素数Nは，$6 \times 8 = 48$である．よって，正規化度数p_xは**表4.2**に示す結果となる．

表4.2より，ヒストグラム均等化を行う画素値の変換式が次のように与えられる．

$$T(0) = 7 \times \frac{2}{48} = 0.2917$$

$$T(1) = T(0) + 7 \times \frac{6}{48} = 1.1667$$

$$\vdots$$

$$T(7) = T(6) + 7 \times 0 = 7.000$$

上式より，画素値を小数点以下第1位で四捨五入して変換表を作成すると，**表4.3**が得られる．

なお，Image Processing Toolboxがある場合，`histeq`関数でヒストグラム均等化を施すことができる．また，`imhist`関数で画像のヒストグラムを表示できる．

実習4.6　**ヒストグラム均等化**

M-file：practice04_6.m

ファイル`firenzeRgb.jpg`を読み込んでモノクロ画像に変換し，ヒストグラム均等化を施してみよう．

4.5　色空間

ここでは，カラー画像を扱う際に重要な色空間について概説しよう．色空間は座標軸系と部分空間から定義され，各色は色空間内の一つの点として表現される．たとえ，読者の皆さんと筆者が同じデータ配列を共有していても，色空間を共有していなければ同じ画像を共有していることにはならない．また，カラー画像を処理する場合，人間の心理的感覚に近い色空間を選択しないと，期待した効果を得られないこともある．

色空間にはRGB空間以外にもさまざまなものが存在する．幸い，広く利用されている色空間の間では相互変換の手段が与えられるので，それらの間を往来できる．以下では，RGB空間とそのほかの例としてYCbCr空間，およびHSI空間を紹介しよう．

● RGB空間

人間の眼球の奥深くの網膜上に中心窩（ちゅうしんか）という部分がある．中心窩には，約575nm，約535nm，約445nmの光にピークを持つ3種類の錐体と呼ばれる感覚細胞が集中して存在している．眼球が，ある物体に焦点を合わせることとは，中心窩付近にその物体の像を結ぶことである．人間は，この像のスペクトラム分布をおおよそ赤，緑，青の組み合わせとして感覚し，色を知覚する．

RGB空間は心理物理的な色表現を与える空間として，撮像機器，表示機器共に広く利用されている．**図4.12**に3次元座標上のRGB空間と色の関係を示す．**図4.12**では，各成分の値域を［0.0 1.0］と正規化しているため，立方体の形をしている．

なお，R，G，Bの定義が異なれば別の色空間と考えなければならない．RGB三原色の選択やガンマ補正の有無などの詳細については，ほかの文献に譲る．

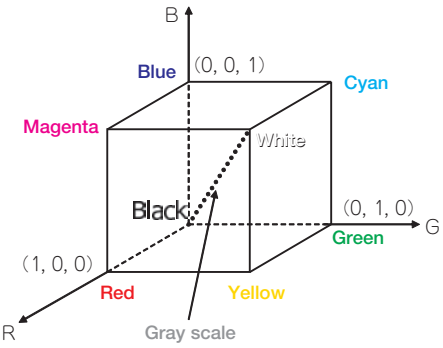


図4.12　RGB 空間

● YCbCr空間

YCbCr空間は，輝度成分Yと二つの色差成分Cb，Crで色を表現する．輝度成分Yはモノクロ映像を与え，二つの色差信号Cb，Cr はカラー化に足りない色度情報を補う．

(1) 変換式

JPEGで広く利用されている色空間の変換と逆変換の例を以下に示そう．

$$\begin{pmatrix} x_Y \\ x_{Cb} \\ x_{Cr} \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & -0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{pmatrix} \begin{pmatrix} x_R \\ x_G \\ x_B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} \quad \cdots\cdots (4.7)$$

$$\begin{pmatrix} x_R \\ x_G \\ x_B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.402 \\ 1 & -0.344 & -0.714 \\ 1 & 1.772 & 0 \end{pmatrix} \begin{pmatrix} x_Y \\ x_{Cb} - 128 \\ x_{Cr} - 128 \end{pmatrix} \quad \cdots\cdots (4.8)$$

ここでx_R，x_G，x_Bは，それぞれガンマ補正がなされた値域［0 255］のR，G，B成分の値である．映像(動画像)の場合は，ITU-R BT.601 勧告で与えられている

$$\begin{pmatrix} x_Y \\ x_{Cb} \\ x_{Cr} \end{pmatrix} = \begin{pmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{pmatrix} \begin{pmatrix} x_R \\ x_G \\ x_B \end{pmatrix} + \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix} \quad \cdots\cdots (4.9)$$

という変換が広く利用されている．ここでx_R，x_G，x_Bは，先ほどと同じように，それぞれガンマ補正がなされた値域［0 255］のR，G，B成分の値であり，x_Y，x_{Cb}，x_{Cr}は，それぞれ［16 235］，［16 240］，［16 240］という範囲で与えられる．Image Processing Toolboxで提供される `rgb2ycbcr`関数は，式(4.9)を実装している．

なお，線形RGB空間に対してガンマ補正を加えた空間をR' G' B' 空間と表現し，R' G' B' 空間から式(4.9)によって与えられる空間をY' Cb' Cr'(もしくは，Y' CbCr)空間と表現することがある．Y' 成分は*Luma*，Cb'，Cr' 成分は*Chroma*と呼ばれ，線形RGB 空間から与えられる輝度成分*Luminance*や色差成分*Chrominance*と区別される．本書ではR' G' B' 空間，Y' Cb' Cr' 空間を簡単にRGB空間，YCbCr空間と表記している．

実習4.7　YCbCr空間

M-file：practice04_7.m

ファイル`firenzeRgb.jpg`を読み込み，式(4.7)の定義に従ってYCbCr空間へ変換し，各成分を画像として表示してみよう．`uint8`型は負の値を扱えないので，`double`型に変換して計算しよう．

(2) 色差サブサンプリング

色差成分Cb，Crの変化に対する人間の視覚特性が輝度成分Yよりも鈍感であることから，色差成分の解像度を落とすことでデータ量の圧縮を実現できる．この考え方は，画像や映像の伝送・蓄積を行う際に，伝送路や蓄積媒体の効率的利用のために用いられる．

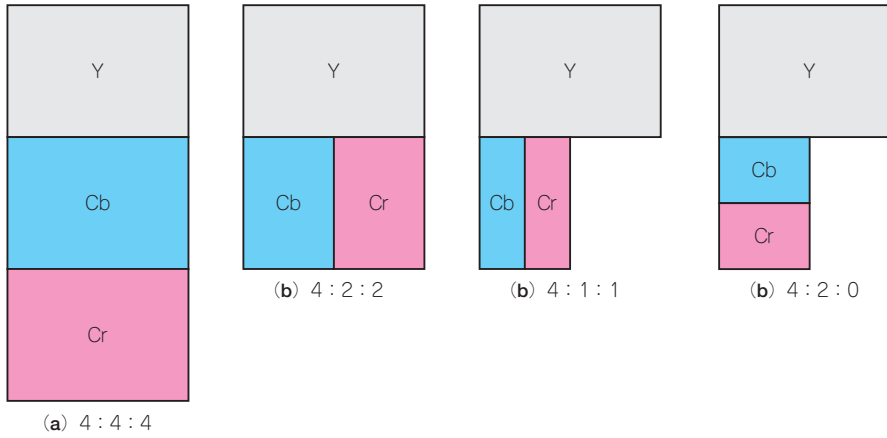


図4.13　色差サブサンプリング

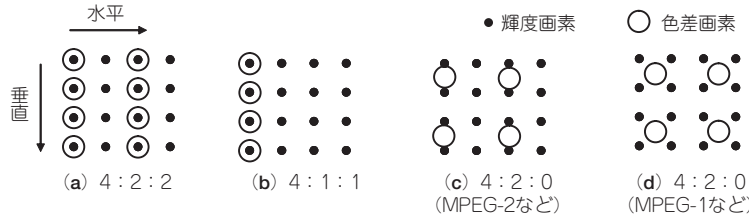


図4.14　色差成分の標本位置

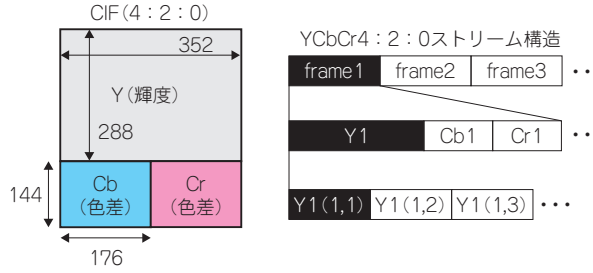


図4.15　CIF映像のビット・ストリーム構造の例

色差信号に対する解像度の低い標本化を**色差サブサンプリング**と呼ぶ．色差サブサンプリングの概略を**図4.13**に示そう．4：4：4は色差サブサンプリングを行わない形式を表している．水平の解像度を半分に減らした形式を4：2：2，1/4に減らした形式を4：1：1と呼ぶ．水平，垂直それぞれの解像度を半分に減らした形式を4：2：0と呼ぶ．4：4：4に比べて，4：2：2は2/3，4：1：1と4：2：0は1/2のデータ量で画像や映像を表現している．

色差成分の標本位置を**図4.14**に示そう．MPEG-1とMPEG-2では異なるなど，フォーマットごとに標本位置に違いがあるので注意が必要である．CIF形式の映像は，MPEG-1と同じ標本位置の4：

2：0となっている．4：4：4と4：2：2，4：1：1，あるいは4：2：0の間の変換については，拡大・縮小処理について解説している第6章を参照されたい．

以下に，CIFの生データからAVIファイルを作成するMATLABのコマンド例を掲載しよう．

例題4.9　CIF→AVI変換

図4.15に示されるようなビット・ストリーム構造をもつCIF映像(ファイル名mobile.cif)をAVI形式(ファイル名mobile.avi)に変換してみよう．ただし，

- 水平走査されている(列優先ではない)
- フレームレートは30［fps］
- フレーム数は300

とする．

解

以下に，MATLABによる処理例を示す．

```
frameSizeY = [288 352];
frameSizeC = [144 176];
frameRate = 30;
nFrames = 300;
nPixelsY = prod(frameSizeY);
nPixelsC = prod(frameSizeC);
precision = 'uint8=>uint8';
fileId = fopen('mobile.cif','r');
% avifile オブジェクトのクリア
clear mex;
% avifile オブジェクトの生成
aviObj = avifile('mobile.avi',...
                 'FPS', frameRate);
for iFrame = 1:nFrames
  % 輝度(Y) 成分を列ベクトルとして読み込み
  pictureTpd= ...
    fread(fileId, nPixelsY, precision);
  % 転置に注意して配列化
  pictureYCbCr(:,:,1) = ...
    reshape(pictureTpd,...
      frameSizeY(2), frameSizeY(1)).';
  for iCmp = 2:3
    % 色差(Cb,Cr) 成分を列ベクトルとして読み込み
```

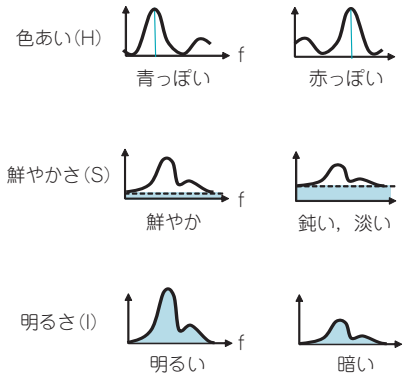


図4.16
スペクトラム分布と色の心理的特徴の関係

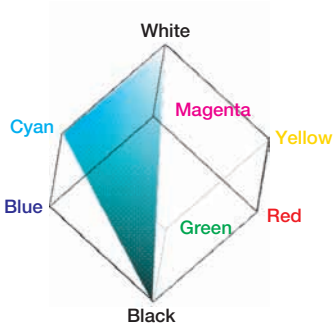
```
      pictureTpd = ...
        fread(fileId, nPixelsC, precision);
      % 転置に注意して配列化
      pictureC = reshape(pictureTpd, ...
        frameSizeC(2), frameSizeC(1)).';
      % 縦横それぞれ2 倍に拡大(Image Proc. TB)
      pictureYCbCr(:,:,iCmp) = ...
        imresize(pictureC,2);
    end
    % YCbCr をRGB へ変換(Image Proc. TB)
    pictureRgb = ycbcr2rgb(pictureYCbCr);
    % フレームの追加
    aviObj = ...
      addframe(aviObj, im2frame(pictureRgb));
  end
  fclose(fileId);
  % avifile オブジェクトのクローズ
  aviObj = close(aviObj);
```

ycbcr2rgb関数とimresize関数はImage Processing Toolboxで提供される関数で，それぞれ画像の色空間変換，拡大・縮小を行う．Image Processing Toolboxを持っていない読者のために，ycbcr2rgbcqとimresizecqという簡易版の関数を用意した．また，ファイルmobile.cifは以下のサイトから入手できるので，試していただきたい．

http://trace.eas.asu.edu/yuv/

HSI空間

RGB空間は，撮像機器や表示機器に適した色空間である．また，YCbCrは伝送や蓄積に適してい



（a）対角線を垂直に立てた図　　（b）（a）を上部から見た図

図4.17　RGB空間とHSI空間の関係

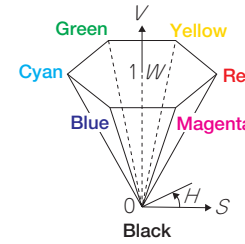


図4.18　6角錐モデル

る．しかし，これらの空間は心理的な色の特徴量を必ずしも表現していないため，色そのものの操作には適していない．ここで紹介するHSI空間は，色操作に適した色空間の一種である．

(1) 色の心理的特徴量

人間は色あい(Hue)，鮮やかさ(Saturation)，明るさ(Intensity)から色を特徴付けるといわれている．これらの頭文字を列挙したものがHSIである．スペクトラム分布とそれぞれの特徴量の関係を**図4.16**に示そう．色あいは，占有するスペクトラムの周波数に関連する．鮮やかさはスペクトラムに含まれる白色の量，明るさはスペクトラムの囲む面積(エネルギ)に対応する．

(2) RGB空間との関係

RGB空間との直感的な関係を示すために，**図4.17(a)**にRGB空間の黒白対角軸を垂直に立ち上げた図を示そう．黒白間の対角軸は，明るさIに対応する．また，**図4.17(a)**を上部から見た図を**図4.17(b)**に示す．色あいHが角度に，鮮やかさSが中心からの距離に対応することがわかる．すなわち，H，S，Iの値をもって色を表現できる．

(3) HSV空間

HSI空間の具体的な定義にはさまざまなものが存在する．ここではA. R. SmithによるHSV (H，S，Value)空間の定義を紹介しよう．このHSV空間は**図4.18**に示すような6角錐の形をしている．なお，MATLABではRGB空間とHSV空間の間の変換に`rgb2hsv`関数と`hsv2rgb`関数を利用することができる．

RGB空間からHSV空間への変換は，x_R，x_G，x_Bの値域が［0.0 1.0］として，

$$x_H = \begin{cases} \dfrac{1}{6}\left(\dfrac{x_G - x_B}{x_{\max} - x_{\min}}\right), & \text{if } x_{\max} = x_R \\ \dfrac{1}{6}\left(2 + \dfrac{x_B - x_R}{x_{\max} - x_{\min}}\right), & \text{if } x_{\max} = x_G \\ \dfrac{1}{6}\left(4 + \dfrac{x_R - x_G}{x_{\max} - x_{\min}}\right), & \text{if } x_{\max} = x_B \end{cases}$$

$$x_S = \frac{x_{\max} - x_{\min}}{x_{\max}}$$

$$x_V = x_{\max} \qquad \cdots\cdots (4.10)$$

と与えられる．ただし，$x_{\max} = max(x_R, x_G, x_B)$，$x_{\min} = min(x_R, x_G, x_B)$である．$x_{\max} = x_{\min}(x_S = 0)$のとき$x_H$は定義されない．また，$x_{\max} = 0(x_V = 0)$のとき$x_S$は定義されない．

例題4.10　HSV空間

ファイルfirenzeRgb.jpgを読み込み，HSV空間に変換して，それぞれの成分を表示してみよう．

解

以下に，MATLABによる処理例を示す．

```
pictureRgb = imread('./data/firenzeRgb.jpg');
% RGB->HSV 変換
pictureHsv = rgb2hsv(pictureRgb);
figure(1); % 原画の表示
imshow(picuterRgb);
figure(2); % H 成分の表示
imshow(picuterHsv(:,:,1));
figure(3); % S 成分の表示
imshow(picuterHsv(:,:,2));
figure(4); % V 成分の表示
imshow(picuterHsv(:,:,3));
```

Image Processing Toolboxがない場合はimshow関数が使えないので，代わりに用意した簡易版のimshowcq関数を利用していただきたい．

カラー画像の明るさと鮮やかさを調整することを想像してみよう．RGB空間では，明るさの調整は良いとして，鮮やかさの調整は直感的に思いつかないだろう．一方，HSV空間では，V成分とS成分を操作すればよいことが分かる．

実習4.8　HSV空間での処理

M-file：practice04_8.m

ファイルfirenzeRgb.jpgを読み込み，RGB各成分に対して累乗則変換を行ってみよう．また，HSV空間へ変換し，V成分のみに累乗則変換を施し，再度RGB空間に戻した画像との比較を行ってみよう．

● その他

(1) ロスレス変換

静止画像国際標準方式JPEG2000では，ひずみを許す非可逆圧縮(ロッシ)モードとひずみを許さない可逆圧縮(ロスレス)モードが規程されている．ロッシ・モードでは先に紹介したYCbCr空間が用

いられるが，有限語調実現ではビット精度で元のRGB空間表現を復元できない．そこで，ロスレス・モード用に可逆のコンポーネント変換(RCT：Reversible Component Transform) が定義されている(章末問題4.9を参照)．

(2) CIE色空間

Image Processing Toolboxでは，CIE(国際照明委員会)が定める色空間を扱うために，色空間変換を作成するmakecform関数と色空間変換を適用するapplycform関数が用意されている．

章末問題

問題4.1　総ビット数

高さ$N_0 = 256$，幅$N_1 = 256$で，R，G，B各成分の1画素あたりのビット数がそれぞれ$b_R = 8$ [bpp]，$b_G = 8$ [bpp]，$b_B = 8$ [bpp] のカラー画像データの総ビット数S [bits] を求めてみよう．

問題4.2　ビット・レート

高さ$N_0 = 720$，幅$N_1 = 1280$，フレーム間隔$T = 1/30$ [sec]，YCbCr4:2:0のHDTV(High Definition Television)のビット・レート [bps] を求めてみよう．ただし，Y，Cb，Cr各成分の1画素当たりのビット数を8 [bpp] とする(ちなみに，圧縮を前提とした地上デジタル放送の1チャネル当たりのビット・レートは14 [Mbps] 程度である)．

問題4.3　RAW画像の読み込み(MATLAB演習)

高さ$N_0 = 612$，幅$N_1 = 816$のカラー画像のR，G，B成分をそれぞれバイト列として保存したfirenze.rrr，firenze.ggg，firenze.bbbというファイルを用意した．これらを読み込んで，RGBカラー画像として表示しよう．なおそれぞれのファイルは，列ベクトルを優先して走査している．

問題4.4　MPEG-1/2の入力(MATLAB演習)

MPEG-2デコーダのライブラリlibmpeg2が以下のサイトから入手できる．

```
http://libmpeg2.sourceforge.net/
```

同サイトより，このライブラリを使ったmpeg2decというツールも入手できる．このツールはMPEG-1/2 の画像をデコードし，各フレーム画像のYCbCr成分を1枚のPGM画像(モノクロ画像)にまとめて出力する機能がある．このPGM画像列からAVI形式の映像を作成してみよう．

問題4.5　フレーム画像列の出力(MATLAB演習)

映像ファイルcalcio.aviをframe2im関数とimwrite関数を使ってJPEG形式のフレーム画像列に変換してみよう．

問題4.6　**ガンマ補正**

ガンマ値$\gamma_c = 1.0$のCCDカメラで撮影した画像(輝度値[0.0 1.0])を，ガンマ値$\gamma_d = 2.0$のモニタで表示したい．ガンマ補正の式を与え，このグラフを描いてみよう(立ち上がりの線形近似は無視する)．

問題4.7　**ヒストグラム均等化**(MATLAB演習)

適当なモノクロ画像に対して，$\gamma < 1$と$\gamma > 1$の累乗則変換を施し，明るくした画像と暗くした画像を用意しよう．さらに，それぞれに対してヒストグラム均等化を施し，その結果を比較してみよう．

問題4.8　**HSV空間における処理**(MATLAB演習)

ファイル`firenzeRgb.jpg`を読み込み，S成分のみに累乗則変換を施した画像を表示してみよう．累乗則変換のパラメータγをいろいろと変えて画像の変化を見てみよう．

問題4.9　**色空間変換**(MATLAB演習)

JPEG2000のロスレス・モードで利用されるRCTと逆RCTを施して，RGB空間における画像表現が完全に復元されることを確かめてみよう．なお，RCTは，

$$x_{Y_0} = \left\lfloor \frac{x_R + 2x_G + x_B}{4} \right\rfloor$$
$$x_{Y_1} = x_B - x_G$$
$$x_{Y_2} = x_R - x_G$$

と与えられ，逆RCTは，

$$x_G = x_{Y_0} - \left\lfloor \frac{x_{Y_1} + x_{Y_2}}{4} \right\rfloor$$
$$x_R = x_{Y_2} - x_G$$
$$x_B = x_{Y_1} - x_G$$

と与えられる．ただし，$\lfloor \cdot \rfloor$は小数点以下切り捨て(整数化)を意味する

参考文献

(1) サイバネットのMATLABホームページ，`http://www.cybernet.co.jp/matlab/`

(2) Rafael C. Gonzalez and Richard E. Woods；*Digital Image Processing, 2nd Ed.*，Prentice Hall，2001.

(3) Yao Wan, J örn Ostermann Ya-Qin Zhang；*Video Processing and Communications*，Prentice Hall，2001.

(4) 小野定康，鈴木純司；わかりやすいJPEG2000の技術，オーム社，2003年.

(5) David S. Taubman and Michael W. Marcellin；JPEG2000, Image Compression Fandamentals, Standards and Practice，Kluwer Academic Publishers，2002.

第5章

画像の近傍処理

前章では，画素処理や色空間変換などの基本的な画像・映像の操作をMATLABの処理例と共に解説した．本章では，画素処理や色空間変換と並んで画像・映像処理の基本となる**近傍処理**について，MATLABによる処理例を交えながら解説しよう．

5.1　近傍処理の一般的操作

画素処理と同じように，近傍処理は空間領域(映像では時空間領域)の処理であり，画像・映像のノイズ除去やエッジ抽出など，画質の調整や強調に用いられる．単純な処理だが，符号化やパターン認識の前処理・特徴量抽出，人工的効果など，その応用範囲は幅広い．

図5.1に3×3のマスクを用いた画像$x[n_0, n_1]$に対する近傍処理の様子を示そう．近傍処理では，各画素とその近傍画素を用いて出力画像$y[n_0, n_1]$の画素値を計算する．この変換をTとすると，出力画像と入力画像は，

$$y[n_0, n_1] = T\{x[n_0, n_1]\} \quad \cdots\cdots (5.1)$$

のように関係付けられる．

いま，**図5.1**に示すように，ある位置$[n_{0c}, n_{1c}]$に着目しよう．式(5.1)は，出力画像の画素$y[n_{0c}, n_{1c}]$

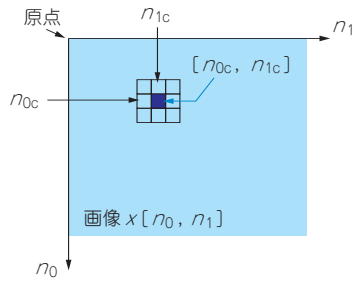


図5.1　近傍処理(3×3マスクの例)

が，$x[n_{0c}, n_{1c}]$とその近傍の画素値の処理から求められることを意味する．この処理をすべての画素に対して行い，出力画像を得る．

画素処理が，着目する画素の値xのみで出力値yを決定したのに対して，近傍処理は，出力値$y[n_{0c}, n_{1c}]$を着目する画素$x[n_{0c}, n_{1c}]$とその周辺の画素値で決定する．マスクの大きさとして，3×3以外を選択することも可能である．1×1のマスクを選択する場合は，画素処理そのものに相当する．また，時間軸や色空間も考慮したより高次元のマスクも存在する．

5.2　近傍処理の具体例

本節では近傍処理の具体例として，以下に示す平滑化処理と先鋭化処理について解説しよう．

- 平滑化処理
  - 加重移動平均フィルタ（線形）
  - ガウス平滑化フィルタ（線形）
  - 順序統計フィルタ（非線形）
- 先鋭化処理
  - ラプラシアン・フィルタ（線形）
  - 高域強調フィルタ（線形）
  - ガウスのラプラシアン（LoG）フィルタ（線形）
  - 勾配（こうばい）フィルタ（非線形）

● 平滑化処理

平滑化処理は，大きなオブジェクトの抽出の前処理や，直線や曲線上の小さな溝を埋める補間，ノイズ除去などに利用される．代表例として，加重移動平均フィルタ（線形）や順序統計フィルタ（非線形）が挙げられる．これらについて解説しよう．

(1) 加重移動平均フィルタ（線形）

加重移動平均フィルタは，最も基本的な近傍処理といえる．その変換は，

$$y[n_0, n_1] = T\{x[n_0, n_1]\}$$
$$= \sum_{k_0=K_T}^{K_B} \sum_{k_1=K_L}^{K_R} w[k_0, k_1]\, x[n_0 + k_0, n_1 + k_1] \quad \cdots\cdots (5.2)$$

のように加重移動平均操作として表現される．ここで，$w[k_0, k_1]$は，各画素にかかる重み係数であり，K_T，K_B，K_L，K_Rは，それぞれ近傍処理の上，下，左，右の範囲を示している．係数配列$w[k_0, k_1]$そのものを本書では**マスク係数**と呼ぶことにする．**図5.2**に3×3マスクの例（$K_T = -1$，$K_B = 1$，$K_L = -1$，$K_R = 1$）を二つ示す．また，**図5.3**に移動平均フィルタを用いた近傍処理の例を示す．なお，配列の境界部では，外側にゼロ値を仮定している．

例題5.1　**移動平均（矩形（くけい））フィルタ**

図5.4(a)に示すノイズの加わった画像に対し，**図5.2**(a)に示すマスクを用いた平滑化処理を施

$\frac{1}{9}\times$

1	1	1
1	1	1
1	1	1

（a）移動平均（矩形）フィルタ

$\frac{1}{16}\times$

1	2	1
2	4	2
1	2	1

（b）加重移動平均フィルタ

図5.2　マスクの例

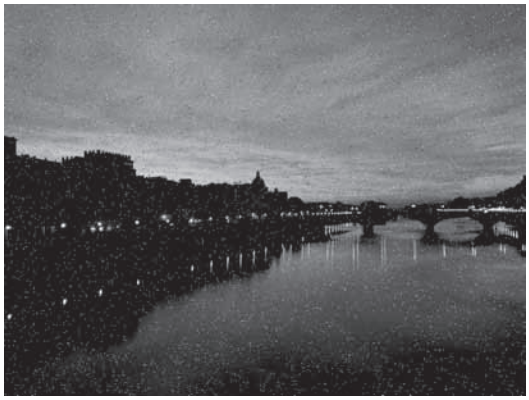
18	9	9	9
27	9	9	9
36	9	9	9

3×3移動平均（矩形）フィルタの処理例

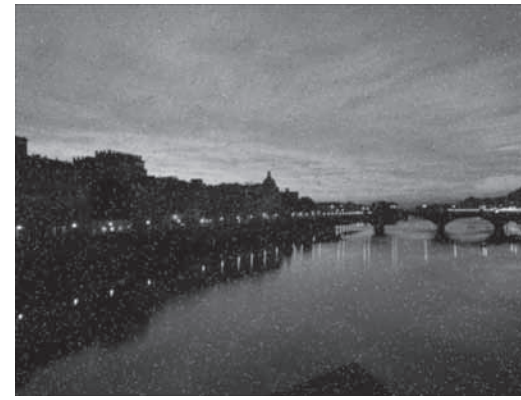
7	9	6	4
12	15	9	6
9	11	6	4

(18+9+9+27+9+9+36+9+9)/9
=135/9
=15

図5.3　近傍処理の例



（a）ノイズの加わった画像



（b）移動平均フィルタで処理した画像

図5.4　画像平滑化の例

してみよう（Image Processing Toolboxでは，`imnoise`関数に，ノイズ・タイプとして`solt & pepper`を指定することで，このごま塩ノイズを重畳できる）．

解

以下に，MATLABによる処理例を示す．`pictureNoisy`は**図5.4（a）**のワークスペース上の参照とし，`uint8`型であるとする．

```
% マスクの設定
movAveMask = [1 1 1;
              1 1 1;
              1 1 1 ] / 9;
% 移動平均フィルタ処理
```

```
  pictureFiltered = ...
    uint8(filter2(movAveMask,pictureNoisy));
```

近傍処理にfilter2関数を利用した．**図5.4(b)**に処理結果を示す．

図5.4(b)を見ると，平均化の効果により輝度の急峻な変化が抑えられ，ノイズが低減されていることがわかる．なお，Image Procssing Toolboxでは， fspecial関数にフィルタ・タイプとして引き数averageを与えることで，移動平均フィルタのマスクを生成できる．さらに，imfilter関数を利用して以下のように近傍処理を実行できる．

```
% マスクの設定
movAveMask = fspecial('average',[3 3]);
% フィルタ処理
pictureFiltered = ...
  imfilter(pictureNoisy,movAveMask);
```

(2) ガウス平滑化フィルタ(線形)

図5.2(a)に示す移動平均フィルタは，中心画素と近傍画素の重みが同じである．そこで，**図5.2(b)**のマスクのように中心に重みを置いたフィルタがしばしば利用される．また，標準偏差σをパラメータとして重み付けを制御できるガウス平滑化フィルタもよく利用される．**ガウス平滑化フィルタ**のマスク係数$w[k_0, k_1]$は，

$$w[k_0,k_1] = \frac{g_\sigma(k_0,k_1)}{\sum_{k_0=K_T}^{K_B}\sum_{k_1=K_L}^{K_R} g_\sigma(k_0,k_1)} \quad \cdots\cdots (5.3)$$

と与えられる．ただし，

$$g_\sigma(p_0, p_1) = \frac{1}{2\pi\sigma^2} e^{-\frac{p_0^2+p_1^2}{2\sigma^2}}$$

Image Processing Toolboxでは， fspecial関数にフィルタ・タイプとして引き数gaussianを与えることで，ガウス平滑化フィルタのマスクを生成できる．

実習5.1　加重移動平均フィルタ

M-file：practice05_1.m

図5.4(a)に示すノイズの加わった画像に対して，**図5.2(b)**に示すマスクを用いた平滑化処理を施して見よう．また，より大きなサイズのマスクを用いた移動平均フィルタについても試してみよう．

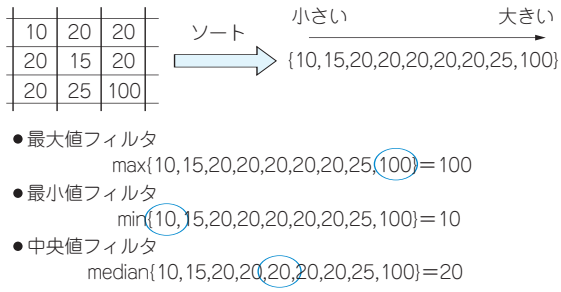


図5.5　順序統計フィルタによる近傍処理

(3) 順序統計フィルタ(非線形)

先に述べた移動平均フィルタ，加重移動平均フィルタは，ノイズを低減すると同時にノイズ以外の輪郭部分もぼかしてしまう．原画像の輪郭をぼかさずにノイズだけを低減する有効な方法として，**メディアン(中央値)フィルタ**がよく知られている．これは順序統計フィルタというクラスに分類される．

順序統計フィルタでは，まず，マスク内の画素をその大きさに応じてソートする．**図5.5**に3×3マスクの処理例を示す．次に，最大値，最小値，あるいは中央値など，ソート後の画素の位置によって出力画素を決定する．例えば，インパルス状の大きな値や小さな値の画素は，マスク内において中央値とはなり難く，中央値を出力するメディアン・フィルタで効果的に除去できる．

例題5.2　**メディアン・フィルタ**

図5.4(a)に示すノイズの加わった画像に対して，3×3のメディアン・フィルタ処理を施してみよう．

解

以下に，MATLABによる処理例を示す．pictureNoisyは**図5.4(a)**のワークスペース上の参照とし，uint8型であるとする．

```
[nRows nCols] = size(pictureNoisy);
% 境界部にゼロ値を外挿
pictureNoisy = ...
  [zeros(nRows,1) ...
   pictureNoisy zeros(nRows,1)];
pictureNoisy = ...
  [zeros(1,nCols+2) ; ...
  pictureNoisy ; ...
  zeros(1,nCols+2)];
%フィルタ処理
```

図5.6　メディアン・フィルタで処理した画像

```
for iCol = 1:nCols
  for iRow = 1:nRows
    % 3x3 領域の切り出し
    % (水平垂直の1画素のずれに注意)
    subDomain = ...
      pictureNoisy(iRow:iRow+2, ...
        iCol:iCol+2);
    % 中央値の出力
    pictureMedian(iRow,iCol) = ...
      median(subDomain(:));
  end
end
```

この例では，中央値の抽出にmedian関数を利用した．**図5.6**に処理結果を示す．

図5.6より，メディアン・フィルタは，移動平均フィルタに比べてインパルス状のノイズをより低減することが分かる．Image Processing Toolboxでは，メディアン・フィルタを適用する関数としてmedfilt2関数が用意されている．さらに，より一般的な順序統計フィルタを適用する関数としてordfilt2関数が用意されている．

実習5.2　**順序統計フィルタ**

M-file：practice05_2.m

最大値フィルタ，最小値フィルタも試してみよう．

0	1	0
1	−4	1
0	1	0

（a）4近傍

1	1	1
1	−8	1
1	1	1

（b）8近傍

図5.7　マスクの例

18	9	9	9
27	9	9	9
36	9	9	9

4近傍ラプラシアン・フィルタの処理例

−36	0	−9	−18
−45	18	0	−9
−108	18	−9	−18

0+9+0+27−4・9+9+0+9+0
=18

図5.8　先鋭化処理の例

● 先鋭化処理

平滑化フィルタと同じくらいよく利用されるフィルタとして，先鋭化フィルタがある．先鋭化フィルタは，画像の詳細部の抽出や強調のために利用される．印刷，医療用画像，産業製品検査，軍事システムなど，応用も幅広い．デジタル・カメラで撮影した画像の“ぼけ”の補正にも利用できる．以下では，この代表例として，ラプラシアン・フィルタと勾配フィルタを紹介しよう．

(1) ラプラシアン・フィルタ(線形)

ラプラシアン・フィルタは，2次元の2次微分操作

$$\nabla^2 x(p_0, p_1) = \frac{\partial^2 x}{\partial p_0^2} + \frac{\partial^2 x}{\partial p_1^2}$$

を意味しており，2次差分演算

$$\frac{\partial^2 x}{\partial n^2} \simeq x[n+1] - 2x[n] + x[n-1]$$

による近似を利用している．微分とはいっても変数が離散であるため，数学的な意味は持たない．ただし，以下の点で2次微分に類似している．

- 一定区間で出力がゼロ
- ステップ/ランプの始点・終点で出力が非ゼロ
- ランプに沿って出力がゼロ

ラプラシアン・フィルタは，式(5.2)で示した加重移動平均(差分)操作によって実現できる．**図5.7**にラプラシアン・フィルタのマスクの例を示そう．**図5.7(a)**は，1次元の2次微分を水平垂直に適用した4近傍ラプラシアン・フィルタのマスクである．**図5.7(b)**は，さらに対角方向も考慮した8近傍ラプラシアン・フィルタのマスクである．**図5.8**に4近傍ラプラシアン・フィルタの処理例を示す．

例題5.3　**4近傍ラプラシアン・フィルタ**

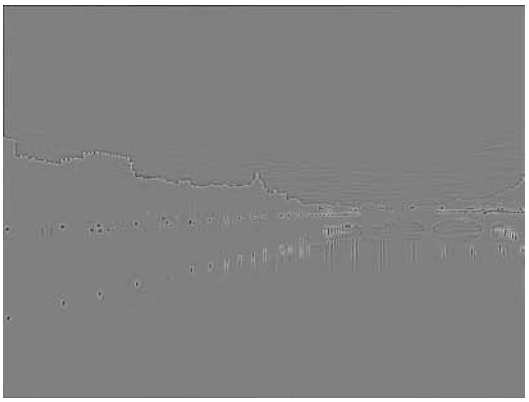
図5.9(a)に示す画像に対して，**図5.7(a)**に示すマスクを用いた先鋭化処理を施してみよう．

解

以下に，MATLABによる処理例を示す．`pictureGray`は**図5.9(a)**のワークスペース上の参照とし，`uint8`型であるとする．

% マスクの設定

（a）原画像



（b）処理画像

図5.9　4近傍ラプラシアン・フィルタによる画像先鋭化の例

```
laplacianMask = [0  1 0;
                 1 -4 1;
                 0  1 0 ];
% ラプラシアン・フィルタ処理
pictureFiltered = uint8(...
  filter2(laplacianMask,pictureGray));
```

マスク係数が異なる以外，加重移動平均フィルタと処理は変わらない．**図5.9**(**b**)に処理結果を示す．なお，出力は負値も含むため，ゼロ値を128にシフトして[0 255]の輝度値範囲で表示している．

図5.9(**b**)より，輝度の変化部が抽出されていることがわかる．なお，Image Processing Toolboxがある場合は，`fspecial`関数にフィルタ・タイプとして`laplacian`を，パラメータαとして0.0を与えることで，4近傍ラプラシアン・フィルタを設計できる．パラメータαを0.5と与えることで8近傍ラプラシアン・フィルタも設計できるが，本書で示したフィルタとは利得が異なるので注意されたい．

実習5.3　8近傍ラプラシアン・フィルタ

M-file：practice05_3.m

8近傍ラプラシアン・フィルタも試してみよう．

(2) 高域強調フィルタ(線形)

先のラプラシアン・フィルタは，画像の輝度変化を抽出した．この結果を原画像に加える(実際は引く)ことで，輪郭を強調するフィルタとなる．このようなフィルタは**高域強調フィルタ**(**High-boost filter**)と呼ばれる．**図5.10**に4近傍および8近傍の高域強調フィルタのマスクを示す．なお，

0	−1	0
−1	A+4	−1
0	−1	0

（**a**）4近傍

−1	−1	−1
−1	A+8	−1
−1	−1	−1

（**b**）8近傍

図5.10　マスクの例

ラプラシアン演算を∇^2とすると，入出力関係は，

$$y[n_0, n_1] = Ax[n_0, n_1] - \nabla^2 x[n_0, n_1]$$

と表現される．ただし，$A \geqq 1$である．この演算は原画像から平滑化画像を差し引く**アンシャープ（unsharp）演算**に対応することから，**アンシャープ・マスキング**とも呼ばれる．

例題5.4　**4近傍高域強調フィルタ**

図5.9（**a**）に示す画像に対して，**図5.10**（**a**）に示すマスクを用いた高域強調処理を施してみよう．ただし，$A = 1$とする．

解

以下に，MATLABによる処理例を示す．pictureGrayは**図5.9**（**a**）のワークスペース上の参照とし，uint8型であるとする．

```
% マスクの設定
unsharpMask = [ 0 -1 0;
               -1 5 -1;
                0 -1 0 ];
% 高域強調フィルタ処理
pictureFiltered = uint8(...
  filter2(unsharpMask,pictureGray));
```

図5.11に処理結果を示す．

図5.11より，フィルタ処理か輪郭部分を強調していることが分かる．なお，Image Processing Toolboxでは，fspecial関数にフィルタ・タイプとしてunsharpを，パラメータαとして0.0を指定することで，本稿で示す4近傍高域強調フィルタを設計できる．パラメータαを0.5と与えることで8近傍高域強調フィルタを設計できるが，本書で示したフィルタとは利得が異なるので注意されたい．

実習5.4　**8近傍高域強調フィルタ**

M-file：practice05_4.m

図5.11　4近傍高域強調フィルタによる画像先鋭化の処理例

8近傍高域強調フィルタも試してみよう．

(3) ガウスのラプラシアン(LoG)フィルタ(線形)

ラプラシアン・フィルタは，画像の輪郭を抽出するために利用できるが，同時に細かいノイズも強調してしまう．そこで，ガウス平滑化フィルタとラプラシアン・フィルタを同時にかける**ガウスのラプラシアン(LoG)フィルタ**もポピュラなフィルタの一つとして利用される．

$$\nabla^2 g_\sigma(p_0, p_1) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{p_0^2 + p_1^2}{2\sigma^2}\right) e^{-\frac{p_0^2+p_1^2}{2\sigma^2}}$$

より，LoGフィルタのマスクは，

$$w[k_0, k_1] = \frac{k_0^2 + k_1^2 - 2\sigma^2}{2\pi\sigma^6} e^{-\frac{k_0^2+k_1^2}{2\sigma^2}}$$

と与えられる．Image Processing Toolbox の`fspecial`関数では，ファイル・タイプに`log`を指定することで，LoGフィルタを設計できる．上式とスケーリングが異なるので，詳細はマニュアルを参照してほしい．

(4) 勾配フィルタ(非線形)

勾配フィルタは，式(5.2)で示した加重移動平均(差分)操作と絶対値和を組み合わせて実現される非線形フィルタである．以下に手続きをまとめる．

1．勾配を求める．

$$\nabla x = \begin{pmatrix} G_0 \\ G_1 \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial p_0} \\ \frac{\partial x}{\partial p_1} \end{pmatrix}$$

2．勾配の大きさを求め出力とする．

$$\|\nabla x\| = \sqrt{G_0^2 + G_1^2} \sim |G_0| + |G_1|$$

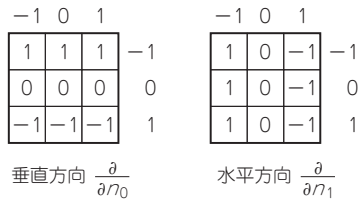


図5.12　Prewitt マスクの例

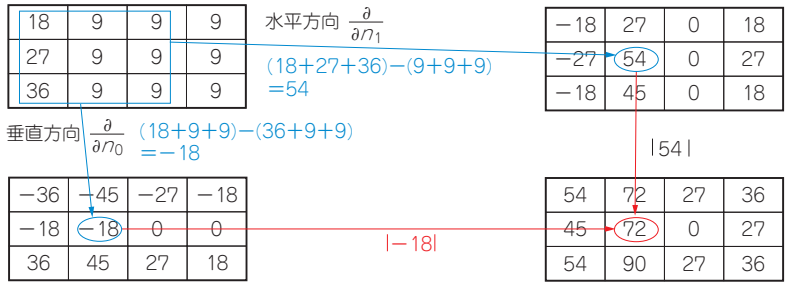


図5.13　Prewitt マスクによる処理例

なお，離散の変数に対応させるため1次微分演算は差分に置き換えられることに注意する．また，勾配の大きさを求める際に絶対値和を取るため，非線形な処理となる．結果が非負の値をとるため，2値化処理と組み合わせてエッジ検出に利用されることが多い．

図5.12に1次微分フィルタのマスク（Prewittマスク）の例を示し，**図5.13**にその処理の具体例を示す．

例題5.5　**Prewittフィルタ**

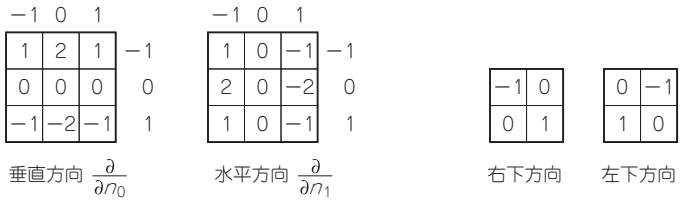
図5.9(a)に示す画像に対して，**図5.12**に示すマスクを用いたPrewittフィルタ処理を施してみよう．

解

以下に，MATLABによる処理例を示す．`pictureGray`は**図5.9(a)**のワークスペース上の参照とし，`uint8`型であるとする．

```
% マスクの設定
prewittMask = [ 1  1  1;
                0  0  0;
               -1 -1 -1 ];
% フィルタ処理と結果の表示
pictureVerFiltered = ...
  filter2(prewittMask, pictureGray);
```

図5.14　Prewitt マスクによる輪郭抽出例



(a) Sobel　　(b) Robarts(Forsen)

図5.15　マスクの例

```
pictureHorFiltered = ...
  filter2(prewittMask.', pictureGray);
pictureFiltered = uint8(...
  abs(pictureVerFiltered) ...
  + abs(pictureHorFiltered) );
```

図5.14に処理結果を示す．輪郭が輝度の変化部として抽出されていることが分かる．

Image Processing Toolboxでは，`fspecial`関数に，フィルタ・タイプとして`prewitt`を指定することで垂直方向のPrewittフィルタのマスクを生成できる．

実習5.5　**Sobel/Robartsフィルタ**

M-file：practice05_5.m

勾配を計算する際に，**図5.15**に示すようなマスクも広く利用されている．**図5.15**(**a**)のマスクを用いた勾配フィルタをSobelフィルタと呼び，**図5.15**(**b**)のマスクを用いた勾配フィルタをRobartsあるいはForsenフィルタと呼ぶ．SobelフィルタおよびRobartsフィルタを用いた輪郭抽出も試してみよう．なお，Image Processing Toolboxでは，`fspecial`関数に，フィルタ・タイプ

としてsobelを指定することで垂直方向のSobelフィルタ・マスクを生成できる．また，egde関数によってSobel，Prewitt，Robartsフィルタを用いたエッジ検出が行える．

5.3　線形シフト不変システム

画像に対する近傍処理は，2次元の信号処理システムとして解釈できる．その最も基本的なクラスとして線形シフト不変(LSI：Linear Shift-invariant) システムがある．加重移動平均(差分)操作によって実現される加重移動平均フィルタ，ラプラシアン・フィルタ，高域強調フィルタはその典型例といえる．以下，線形シフト不変システムについて解説しよう．

● インパルス応答

1次元の線形時不変システムと同じように，2次元の線形シフト不変システムも，そのインパルス応答によってシステムの性質が表現される．なお，2次元の単位インパルス信号 $\delta[n_0, n_1]$は，1次元の単位インパルス信号 $\delta[n_d]$により

$$\delta[n_0, n_1] = \delta[n_0]\delta[n_1] = \begin{cases} 1 & n_0 = n_1 = 0 \\ 0 & \text{その他} \end{cases}$$

と定義される．

また，2次元システム $T\{\cdot\}$のインパルス応答は，

$$h[n_0, n_1] = T\{\delta[n_0, n_1]\} \quad \cdots\cdots (5.5)$$

と定義される(**図5.16**)．

● 畳み込み演算

システムが線形かつシフト不変である場合，その性質は式(5.5)で定義されるインパルス応答 $h[n_0, n_1]$によって決定され，出力 $y[n_0, n_1]$は $h[n_0, n_1]$と入力 $x[n_0, n_1]$の演算によって求められる．このことを示そう．

まず，入力 $x[n_0, n_1]$を $\delta[n_0, n_1]$の重み付け和として

$$x[n_0, n_1] = \sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} x[k_0, k_1]\delta[n_0 - k_0, n_1 - k_1] \quad \cdots\cdots (5.6)$$

と表現しよう．すると，入力 $x[n_0, n_1]$に対するシステムの応答が次のように表現される．

$$\begin{aligned} y[n_0, n_1] &= T\{x[n_0, n_1]\} \\ &= T\left\{\sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} x[k_0, k_1]\delta[n_0 - k_0, n_1 - k_1]\right\} \\ &= \sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} x[k_0, k_1]T\{\delta[n_0 - k_0, n_1 - k_1]\} \quad (\because \text{線形性}) \end{aligned}$$

(a) 一般的な応答　　(b) インパルス応答

図5.16　システムの応答とインパルス応答

$$= \sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} x[k_0, k_1] h[n_0 - k_0, n_1 - k_1] \quad (\because \text{シフト不変性}) \cdots\cdots (5.7)$$

この誘導では，$T\{\cdot\}$の線形性とシフト不変性を利用した．最後の結果は2次元の畳み込み演算となっている．

これは，変数変換により

$$y[n_0, n_1] = \sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} h[k_0, k_1] x[n_0 - k_0, n_1 - k_1] \cdots\cdots (5.8)$$

と表現することもできる．マスクの範囲外にゼロ値の係数を仮定して式(5.2)を式(5.8)と比べると，マスク係数$w[k_0, k_1]$はインパルス応答$h[k_0, k_1]$と

$$h[k_0, k_1] = w[-k_0, -k_1]$$

のように関係していることが分かる．

MATLABでは，2次元の畳み込み演算を行う関数としてconv2関数が用意されている．Image Processing Toolboxの`imfilter`関数も，convオプションを指定することで畳み込み演算を行う．これは，係数配列をマスク係数とするか，180°回転したインパルス応答とするかという違いにほかならない．

例題5.6　インパルス応答

MATLABで，次のコマンドを試してみよう．

```
ires = [ 1 2 3;
         4 5 6;
         7 8 9 ];
conv2(ires,1,' full' )
```

次も試してみよう．

```
mask = [ 9 8 7;
         6 5 4;
         3 2 1];
filter2(mask,1,' full' )
```

解

いずれもシステムのインパルス応答が得られ，同じ結果となる．

システムの性質

線形シフト不変システムでは，インパルス応答$h[n_0, n_1]$の性質によってシステムを分類すること

ができる．以下に分類の例を示そう．

(1) サポート領域

インパルス応答が無限に広がるか有限の範囲に収まるかによって，1次元の場合と同じようにFIRかIIRかに分類される(**図5.17**)．例えば3×3マスクを用いた近傍処理は，インパルス応答の広がりが3×3のマスク・サイズに限定されるので，FIRフィルタと分類される．

(2) 因果性

インパルス応答$h[n_0, n_1]$が，$n_0 < 0$もしくは$n_1 < 0$の範囲において非ゼロの係数を持たなければ，そのシステムは**因果的システム**と呼ばれる(**図5.18**)．

ただし，1次元の場合と異なり，空間的な因果性はあまり問題にならない．画像処理においては，むしろゼロ位相システムなど，非因果的なシステムを積極的に利用する．これまで紹介した各種近傍処理はすべてゼロ位相をもつ**非因果的システム**である．

位相特性については，次章で再度取り上げる．なお，多次元の場合，ある特定の変数，例えば垂直変数n_0のみ因果性を満たすシステムもあり得る．このようなシステムを**準因果的システム**という．

(3) 可分離性

インパルス応答が，$h[n_0, n_1] = h_0[n_0]h_1[n_1]$と変数ごとの1次元インパルス応答に分離して表現できるシステムを**可分離システム**と呼ぶ．分離できないシステムの方が一般的であり，これを**非分離システム**と呼ぶ[**図5.19(a)**]．

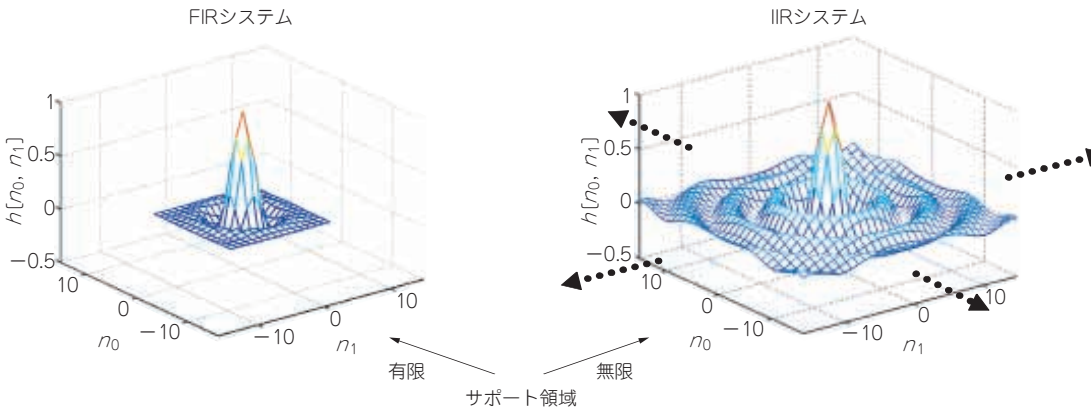


図5.17　FIRシステムとIIRシステム

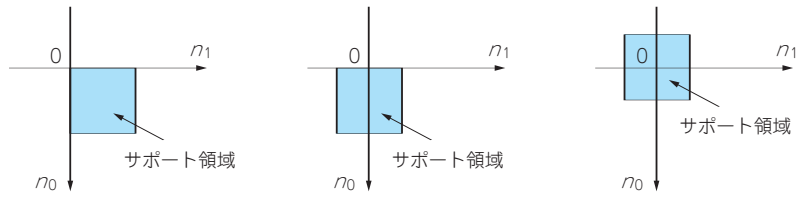
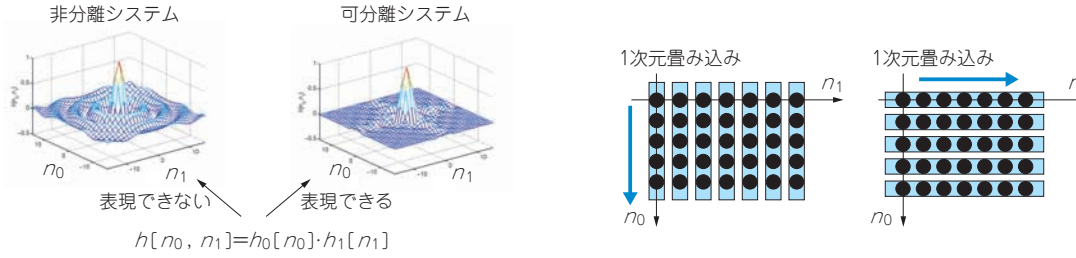


図5.18　システムの因果性



（a）インパルス応答　　（b）分離処理

図5.19　非分離システムと可分離システム

可分離システムは，

$$u[n_0,k_1]=\sum_{k_0=-\infty}^{\infty}x[k_0,k_1]h_0[n_0-k_0]$$
$$=h_0[n_0]*x[n_0,k_1]$$
$$y[n_0,n_1]=\sum_{k_1=-\infty}^{\infty}u[n_0,k_1]h_1[n_1-k_1]$$
$$=h_1[n_1]*u[n_0,n_1]$$

のように次元ごとの独立な畳み込み演算によって効果的に実現できる利点をもつ[**図5.19(b)**]．

例題5.7　**インパルス応答の分離**

3×3の移動平均フィルタのインパルス応答が分離可能であることを示そう．

解

3×3の移動平均フィルタのインパルス応答は

$$h[n_0,n_1]=h_0[n_0]\cdot h_1[n_1]$$

ただし，

$$h_d[n]=\begin{cases}\dfrac{1}{3} & -1\le n\le 1\\ 0 & \text{その他}\end{cases}$$

と表現できる．

実習5.6　**分離処理**

M-file：practice05_6.m

3×3の移動平均フィルタを用いた画像の平滑化を分離処理によって行ってみよう．

5.4　時間方向フィルタ

これまで，近傍処理として静止画像を対象とした空間処理の例を紹介した．近傍処理は映像の各

フレームに適用できることはもちろんのこと，**図5.20**からも分かるように，時間軸方向についても適用できる．以下では，フレーム間平均処理およびフレーム間差分処理について紹介しよう．

● フレーム間平均処理

フレーム間平均は，時間方向に平均をとることでノイズの除去や急に変化する動作を除去する．2フレーム間の平均をとるフィルタのインパルス応答は

$$h[k] = \frac{1}{2}\delta[k] + \frac{1}{2}\delta[k-1] \quad \cdots\cdots (5.9)$$

と与えられる．

例題5.8　フレーム間平均処理

AVI形式の映像ファイルを読み込み，フレーム間平均処理を施してAVI形式の映像ファイルに出力してみよう．

解

MATLABによる処理例を以下に示す．ただし，`fileNameIn`は入力ファイル名，`fileNameOut`は出力ファイル名を保持する文字列変数とする．

```
% 入力ファイル
fileInfo = aviinfo(fileNameIn);
frameRate = fileInfo.FramesPerSecond;
nFrames = fileInfo.NumFrames;
% 出力ファイル
clear mex;
```

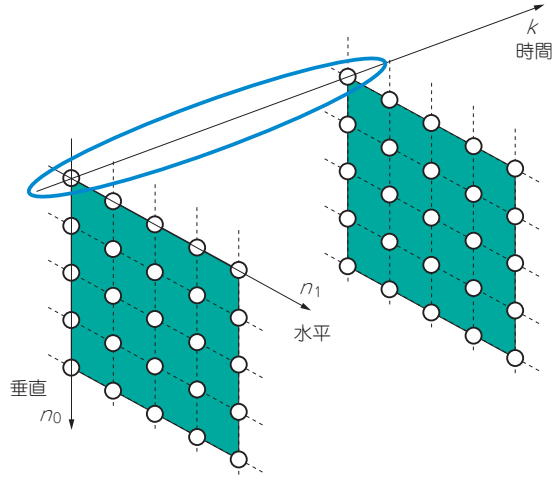


図5.20　時間方向フィルタ

```
aviObj = avifile(fileNameOut,...
  'FPS',frameRate,...
  'COMPRESSION','None');
% フレーム間平均処理準備
framePre = aviread(fileNameIn,1);
picturePre = framePre.cdata;
frameFiltered.colormap = [];
% フレーム間平均処理
for iFrame = 1:nFrames
  % 現フレームの読み出し
  frameCur = aviread(fileNameIn,iFrame);
  pictureCur = frameCur.cdata;
  % フィルタ処理
  pictureFiltered = uint8(...
    0.5*(double(pictureCur)...
      + double(picturePre)));
  % 処理フレームの出力
  frameFiltered.cdata = pictureFiltered;
  aviObj = addframe(aviObj,frameFiltered);
  % 前フレームの更新
  picturePre = pictureCur;
end;
aviObj = close(aviObj);
```

Image Processing Toolboxがある場合，フィルタ処理の部分でimlincomb関数を利用できる．

● フレーム間差分処理

フレーム間差分処理は，時間方向に差分をとることで時間的変化分を抽出する．2フレーム間の差分をとるフィルタのインパルス応答は

$$h[k] = \delta[k] - \delta[k-1] \quad \cdots\cdots (5.10)$$

と与えられる．なお，処理結果が非負の値となるよう絶対値をとることもある．

実習5.7　フレーム間差分絶対値処理

M-file：practice05_7.m

AVI形式の映像ファイルを読み込み，フレーム間差分処理を施し，絶対値をとってAVI形式の映像ファイルに出力してみよう（Image Processing Toolboxがある場合，差分処理にはimsubtract関数を，差分絶対値を求めたいときにはimabsdiff関数を利用できる）．

5.5　境界処理

画像や映像は，空間領域において有限な明確に定義されたサポート領域をもつ．すなわち，解像度や画素数と表現されるものである．画像にフィルタ処理を施す際には，入力画像のサポート領域外にある未知の画素値が必要となる．畳み込み演算を行った場合は，出力の画素数増加問題も生じる．このことは，境界部分で1次元音声処理とは異なった対策が要求されることを意味する．以下では，こうした問題の解決法として，**ゼロ値拡張法**，**周期拡張法**，**対称拡張法**について解説する．

● ゼロ値拡張法

ゼロ値拡張法は，**図5.21**（**a**）に示すように領域外の画素値にゼロ値を仮定する．すなわち，画像$x[n_0, n_1]$のサイズが垂直×水平$= N_0 \times N_1$のとき，

$$x[n_0, n_1] = 0, \quad n_0 < 0 \text{ or } n_0 > N_0 - 1 \text{ or } n_1 < 0 \text{ or } n_1 > N_1 - 1$$

のように仮定する．これまで示してきた処理例は，すべてこのゼロ値拡張を行っていた．このゼロ値拡張は簡便だが，平滑化処理では処理画像に黒い縁が発生し，先鋭化処理でも処理画像に白や黒

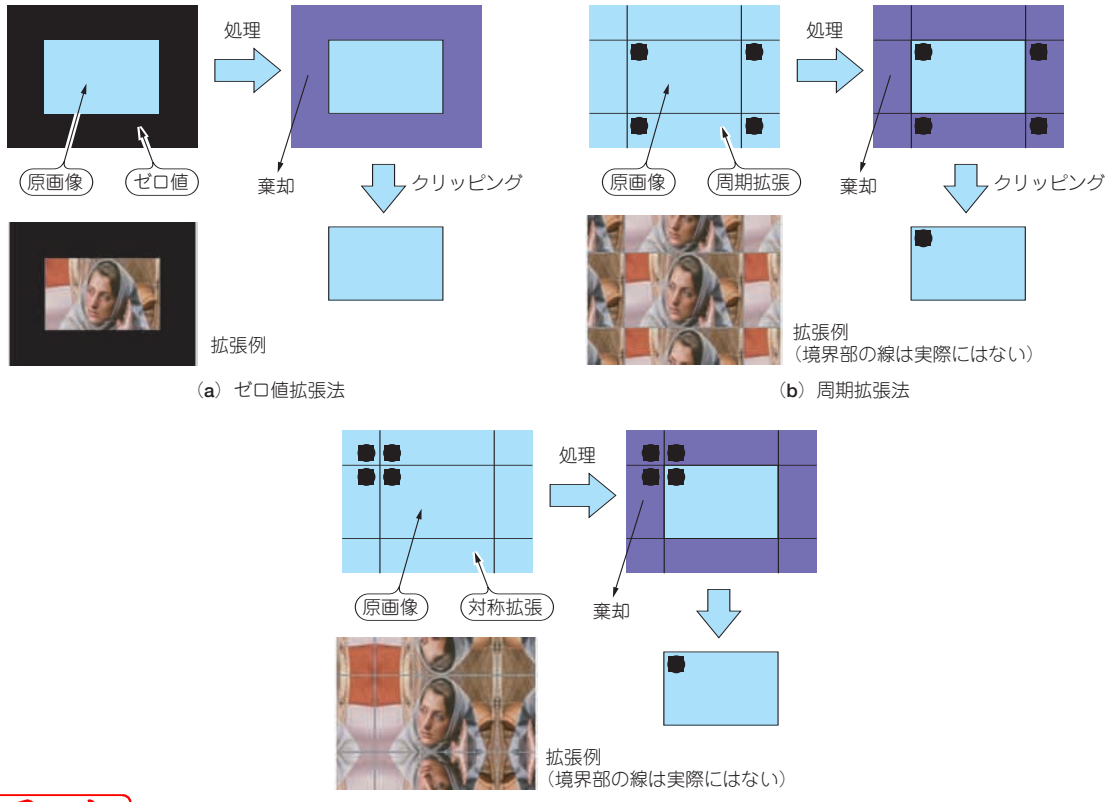


図5.21　境界の拡張法

（a）ゼロ値拡張　（b）周期拡張　（c）対称拡張

図5.22　各種拡張法による処理結果（左下境界部拡大図）

の縁が発生する問題がある（**図5.22**）．

● 周期拡張法

周期拡張法は，**図5.21**（**b**）に示すように画像に周期性を仮定する．すなわち，画像$x[n_0, n_1]$のサイズが垂直×水平＝$N_0 \times N_1$のとき，次式のように仮定する．

$$x[n_0, n_1] = x[n_0 - \ell_0 N_0, n_1 - \ell_1 N_1]$$

周期信号に対する畳み込み演算の結果は，同じ周期をもつ周期信号となる．すなわち，1周期に着目すれば画素数増加の問題が生じない．また，DFT領域での積演算と等価である．しかし，他方の境界からの色の回り込みが問題となる（**図5.22**）．

● 対称拡張法

対称拡張法は，**図5.21**（**c**）に示すように画像の境界部に対称性を仮定する．すなわち，

$$x[n_0, n_1] = x[-n_0 + 2c_0, n_1] = x[n_0, -n_1 + 2c_1]$$

のような関係を仮定する．c_0，c_1は，それぞれの軸における対称の中心である．例えば，$c_d = 0$，$N_d - 1$と選択することは，**図5.23**（**a**）のWs（Whole-sample Symmetry）の対称性を，$c_d = -1/2$，$N_d - 1/2$と選択することは**図5.23**（**b**）のHS（Half-sample Symmetry）の対称性を仮定することに相当する．対称拡張法は境界部が良好に仕上がる利点をもつ（**図5.22**）．

例題5.9　**ゼロ値拡張法と対称拡張法**（1次元）

次のインパルス応答$h[n]$をもつ1次元の線形システムに，次の1次元信号$x[n]$を入力したときに得られる出力を，ゼロ値拡張法，対称拡張法（HS）により求めてみよう．

$$h[0] = \frac{1}{2}, h[-1] = h[1] = \frac{1}{4}$$

$$x[0] = 1, x[1] = 2, x[2] = 4, x[3] = 2$$

解

ゼロ値拡張法の場合，

（a）サンプル上に対称軸　　（b）サンプル間に対称軸

図5.23　二つの対称性

$$yz[0]=\frac{1}{4}\cdot 0+\frac{1}{2}\cdot 1+\frac{1}{4}\cdot 2=1$$

$$yz[1]=\frac{1}{4}\cdot 1+\frac{1}{2}\cdot 2+\frac{1}{4}\cdot 4=2.25$$

$$yz[2]=\frac{1}{4}\cdot 2+\frac{1}{2}\cdot 4+\frac{1}{4}\cdot 2=3$$

$$yz[3]=\frac{1}{4}\cdot 4+\frac{1}{2}\cdot 2+\frac{1}{4}\cdot 0=2$$

対称拡張法（HS）の場合，

$$ys[0]=\frac{1}{4}\cdot 1+\frac{1}{2}\cdot 1+\frac{1}{4}\cdot 2=1.25$$

$$ys[1]=\frac{1}{4}\cdot 1+\frac{1}{2}\cdot 2+\frac{1}{4}\cdot 4=2.25$$

$$ys[2]=\frac{1}{4}\cdot 2+\frac{1}{2}\cdot 4+\frac{1}{4}\cdot 2=3$$

$$ys[3]=\frac{1}{4}\cdot 4+\frac{1}{2}\cdot 2+\frac{1}{4}\cdot 2=2.5$$

Image Processing Toolbox のimfilter関数では，デフォルトでゼロ値拡張法を使用する．境界に関するオプションとしてcircularを指定すると周期拡張法が適用され，symmetricを指定すると対称拡張法（HS）が適用される．また，配列を拡張するためのpadarray関数も用意されている．

実習5.8　**境界処理**

M-file：practice05_8.m

9×9の移動平均フィルタなど，大きなマスクを用いた平滑化を行い，ゼロ値拡張法，周期拡張法，対称拡張法の結果を比較してみよう．

章末問題

問題5.1　**エンボス・フィルタ**（MATLAB演習）

2次のマスクを用いて画像の輝度成分に線形フィルタ処理を施すと，浮き彫り（エンボス）効果が得

られる．試してみよう．ただし，処理結果に負の値が現れるのでdoubleで処理し，0.5を加算する．

1	0	0
0	0	0
0	0	−1

問題5.2　**フィルタの係数配列**（MATLAB演習）

さまざまなガウシアン・フィルタ，LoGフィルタをサイズ，標準偏差パラメータを変えながら設計し，係数配列をmesh関数で観察してみよう．

問題5.3　**加重移動平均フィルタの分離処理**（MATLAB演習）

図5.2(b)に示した加重移動平均フィルタのインパルス応答が分離可能であることを示し，画像の平滑化を分離処理によって行ってみよう．

問題5.4　**フレームごとの輪郭抽出**（MATLAB演習）

映像データを読み込み，フレームごとに勾配フィルタをかけて結果を映像として表示してみよう．

問題5.5　**境界処理**（MATLAB演習）

次の配列に対して，4近傍ラプラシアン・フィルタ処理およびPrewittフィルタ処理を施してみよう．ただし，境界処理として対称拡張法(HS)を適用すること．

18	9	9	9
27	9	9	9
36	9	9	9

問題5.6　**境界処理**（MATLAB演習）

画像firenzeRgb.jpgの輝度成分に対して，4近傍ラプラシアン・フィルタ処理およびPrewittフィルタ処理を施してみよう．ただし，境界処理として対称拡張法(HS)を適用すること．

参考文献

(1) 貴家仁志；よくわかるディジタル画像処理，CQ出版社，1996年．
(2) Rafael C. Gonzalez and Richard E. Woods；Digital Image Processing, 2nd Ed., Prentice Hall, 2001.
(3) John, W. Woods；Multidimensional Signal, Image and Video Processing and Coding, Academic Press, 2006.

第6章

画像の拡大・縮小処理

本章では，画像の拡大処理と縮小処理をフィルタリングの観点から解説しよう．拡大・縮小を行うシステムは，複数の異なる標本化周期のディジタル信号を扱うマルチレート・システムとして解釈できる．マルチレート・システムの構成要素，基本的なレート変換器，設計手法を簡単にまとめる．また，映像のフレーム・レート変換についても触れる．

6.1　画像のマルチレート信号処理

画像の拡大・縮小処理の問題は，マルチレート信号処理の問題として取り扱うことができる．マルチレート信号処理については後述することにして，まずは簡単な例を試してみよう．

● 間引きによる縮小処理

例題6.1　**間引きによる縮小処理**

図6.1に示す画像に対して間引きによる縮小処理を施してみよう．間引き処理は，**図6.2**に示すように，2×2ブロックの4 画素のうち，○印1画素を残し，×印の3画素を捨てることで実現しよう．

解

MATLABによる処理例を以下に示す．pictureGrayは原画像を保存する配列である．

```
pictureGrayDownsampled = ...
  pictureGray(1:2:end, 1:2:end);
```

もしくは，

```
pictureGrayDownsampled = ...
```

図6.1　原画像

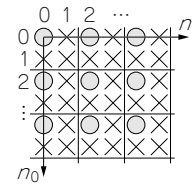


図6.2　間引き処理

図6.3　間引きによる縮小画像

```
    downsample(...
      downsample(pictureGray,2).',2).';
```

図6.3に処理結果を示す．結果として，画像の大きさが1/4(垂直1/2×水平1/2)のサイズに縮小される．

図6.3より，細かいしま模様の部分に**図6.1**と異なる方向の荒いしま模様が現れていることを確認できる．これは，間引き処理による高周波成分から低周波成分への折り返し成分(エリアジング)に起因する．後ほど詳しく解説しよう．

なお，Image Processing Toolboxでは，

```
sizeNew = size(pictureGray)./[2 2];
pictureGrayDownsampled = ...
  imresize(pictureGray,sizeNew);
```

とimresize関数を利用しても同じ処理を実現できる．

実習6.1　**間引きによる縮小処理**

M-file：practice06_1.m

間引き処理によって画像を$1/D_0D_1$(垂直$1/D_0$×水平$1/D_1$)のサイズに縮小しよう．

● 平均による縮小処理

では，間引き処理による縮小処理に代わって，平均による縮小処理を試してみよう．**図6.4**に，4×4の配列を平均操作によって1/2×1/2に縮小する処理例を示す．この処理は，画像配列を大きさ2×2のブロックに分け，それぞれの平均値を求めて出力画素値としている．より一般的な$1/D_0$×$1/D_1$の縮小処理の場合，大きさ$D_0 \times D_1$のブロックに分割し，それぞれの平均値を出力画素とすれば

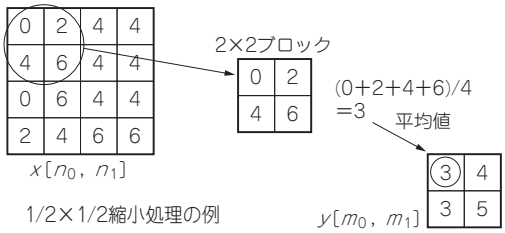


図6.4　平均による縮小処理の例

図6.5　平均による縮小画像

よい．

例題6.2　平均による縮小処理

図6.1に示す画像を2×2の大きさのブロックに分解し，それぞれのブロックの平均値を出力することによって，画像を1/4(垂直1/2×水平1/2)のサイズに縮小してみよう．

解

MATLABによる処理例を以下に示す．pictureGrayは原画像を保存するuint8型の配列とする．

```
pictureGrayDecimated = uint8(( ...
  double(pictureGray(1:2:end,1:2:end)) + ...
  double(pictureGray(2:2:end,1:2:end)) + ...
  double(pictureGray(1:2:end,2:2:end)) + ...
  double(pictureGray(2:2:end,2:2:end)) )/4);
```

図6.5に平均による縮小画像を示す．この図では分かりにくいが，先の間引き処理の結果(**図6.3**)において生じたエリアジングが，**図6.5**の画像では目立たなくなることを確認できる．

エリアジング回避のための方法は平均処理に限らず，より一般的なものを選択できる．画像の縮小処理の設計については後述する．

なお，Image Processing Toolboxでは，

```
sizeNew = size(pictureGray)./[2 2];
pictureGrayDecimated = ...
  imresize(pictureGray, sizeNew, ...
           'bilinear',ones(2,2)/4);
```


などimresize関数を利用しても同じ処理を実現できる．

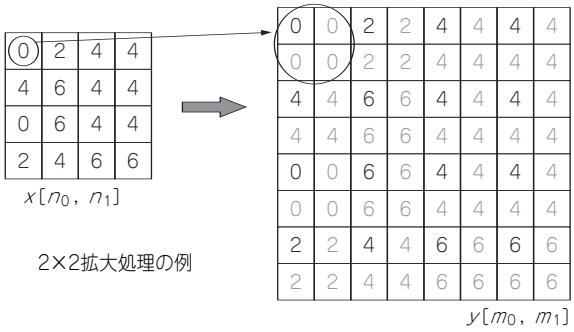


図6.6　ゼロ次ホールドによる拡大処理の例

図6.7　原画像

実習6.2　**平均による縮小処理**

M-file：practice06_2.m

画像を$D_0 \times D_1$の大きさのブロックに分解し，それぞれのブロックの平均値を出力することによって画像の大きさを$1/D_0 D_1$(垂直$1/D_0$×水平$1/D_1$)のサイズに縮小しよう．

● ゼロ次ホールドによる拡大処理

では次に，拡大処理についても試してみよう．**図6.6**に，4×4の配列を**ゼロ次ホールド**によって2×2倍に拡大する処理例を示す．この処理は，各画素値を2×2のブロックに拡大し，出力画像に配している．より一般的な$U_0 \times U_1$の拡大処理では，各画素を大きさ$U_0 \times U_1$のブロックに拡大し，出力画像に配してやればよい．なお，ゼロ次ホールド法は，最近傍(Nearest Neighbor)法とも呼ばれる．

例題6.3　**ゼロ次ホールドによる拡大処理**

図6.7に示す画像の各画素を，同じ画素値を一様にもつ2×2の大きさのブロックに置き換える(ゼロ次ホールド処理を施す)ことで，画像を4倍(2×2)のサイズに拡大しよう．

解

以下に，MATLABによる処理例を示す．pictureGrayは原画像を保存するuint8型の配列とする．

```
pictureGrayIntpd = ...
  zeros(size(pictureGray).*[2 2],'uint8');
pictureGrayIntpd(1:2:end,1:2:end) = ...
  pictureGray;
pictureGrayIntpd(2:2:end,1:2:end) = ...
  pictureGray;
pictureGrayIntpd(1:2:end,2:2:end) = ...
  pictureGray;
```

図6.8　ゼロ次ホールドによる拡大画像

```
pictureGrayIntpd(2:2:end,2:2:end) = ...
  pictureGray;
```

図6.8にゼロ次ホールドによる拡大画像を示す．2×2の大きさのブロックごとに画素値が一様な値をもつため，ブロック状の効果が知覚されてしまう．

Image Processing Toolboxでは，

```
sizeNew = size(pictureGray).*[2 2];
pictureGrayIntpd = imresize(...
  pictureGray, sizeNew, 'nearest');
```

とimresize関数を利用しても同じ処理を実現できる．

実習6.3　**ゼロ次ホールドによる拡大処理**

M-file：practice06_3.m

画像の各画素を，同じ画素値を一様にもつ$U_0 \times U_1$の大きさのブロックに置き換えることで，画像の大きさを$U_0 \times U_1$倍のサイズに拡大しよう．

6.2　マルチレート信号処理とは？

ディジタル信号は，アナログ信号のサンプリングと量子化によって与えられる．ここで，**図6.9**を眺めながら標本化の操作を思い出してほしい．

● 標本化の復習

標本化とは，連続的に変化するアナログ信号について，ある一定の間隔T_Sでその瞬時値を取得し，数列に置き換える操作である．例えば音声信号の場合，1/8000秒という標本化間隔がしばしば利用される．この数値は，元のアナログ信号に戻すための情報として，以後もその数列につきまとうことになる．標本化間隔T_Sの代わりに，その逆数である標本化周波数F_Sを用いることも多い．標本化周波数は，サンプリング・レートあるいは単にレートとも呼ばれる．

いま，異なる標本化間隔T_{S0}，T_{S1}で標本化された二つの数列$x_0[n]$，$x_1[m]$を仮定しよう．この二つの数列の足し算$x_0[\ell]+x_1[\ell]$にはどのような意味があるのだろうか？

少なくとも，各瞬時値

- $x_0[\ell]=x_0(\ell T_{S0})$
- $x_1[\ell]=x_1(\ell T_{S1})$

をとらえた時刻T_{S1}とT_{S0}が常に一致することはなく，その結果の有用性は乏しい．そこで，標本化間隔をそろえることが重要となる．これは可能だろうか？

答えはYesである．この実現方法は，大きく以下の二つの方式に分類することができる(**図6.10**)．

- いったんアナログ信号に復元し，同じ標本化間隔にて再標本化を施す方式
- ダウン・サンプラ，アップ・サンプラ，ディジタル・フィルタからなるオール・ディジタル方式

特に後者は，経年変化やノイズの影響を受けにくいといったディジタルの利点を保ったまま実現できる．このように，複数の異なる標本化間隔あるいはサンプリング・レートを取り扱う信号処理を**マルチレート信号処理**と呼ぶ．

● レート変換の応用例

マルチレート信号処理の応用例は多岐に亘り，現在の情報通信技術の根幹をなす重要なものが多数存在する．音声・音響データなどのフォーマット変換器はそのよい例である．例えば，44.1kHzのレート(標本化間隔の逆数)をもつCDデータと96kHzのレートをもつDVD Audioデータの間のフォーマット変換がある．画像の縮小処理や拡大処理は多次元のマルチレート信号処理として解釈

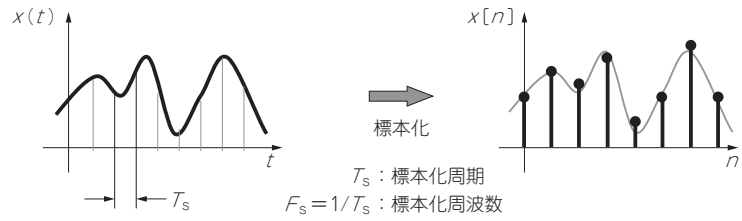


図6.9
標本化の復習

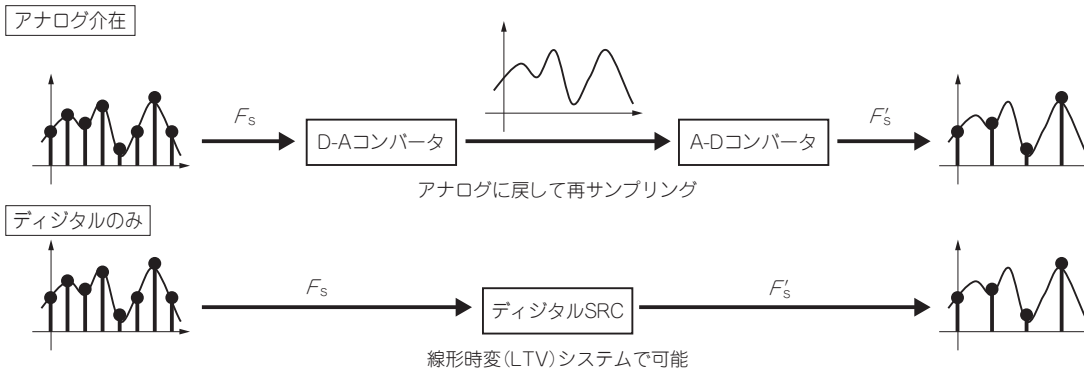


図6.10　サンプリング・レート変換器（SRC）

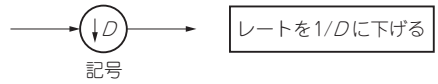


図6.11　ダウン・サンプラ（間引き器）

でき，後述するように，より一般的なシステムが存在する．

まずは準備として，二つの1次元の処理要素を導入しよう．一つは**ダウン・サンプラ**であり，もう一つは**アップ・サンプラ**である．これらの処理そのものは非常に単純である．しかし，このたった二つの処理要素を加えるだけで，線形時不変システムのみでは実現不可能だったさまざまな処理が実現可能となる．

● ダウン・サンプラ

ダウン・サンプラは，サンプリング・レートを整数分の1に下げる処理要素である．その動作から**間引き器**とも呼ばれる．また，この処理そのものを**ダウン・サンプリング**あるいは**間引き処理**と呼ぶ．Dを正の整数としてサンプリング・レートを$1/D$に下げる場合，ダウン・サンプラはDサンプルごとに1点のサンプルを残し，ほかの$D-1$個のサンプルを単純に棄却する（捨ててしまい，以後の処理で用いない）．

(1) 入出力関係

ダウン・サンプラの入出力関係を式で表現すると

$$y[m] = x[Dm] \quad \cdots\cdots (6.1)$$

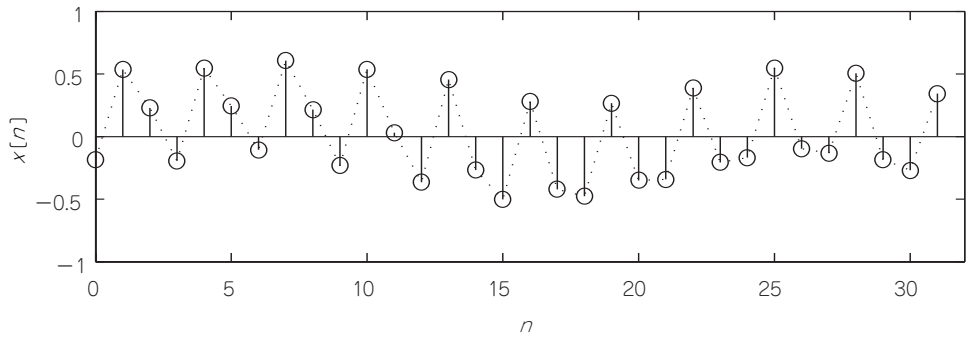
となる．また，本稿におけるダウン・サンプラの記号を**図6.11**に示す．

例題6.4　**ダウン・サンプラ**

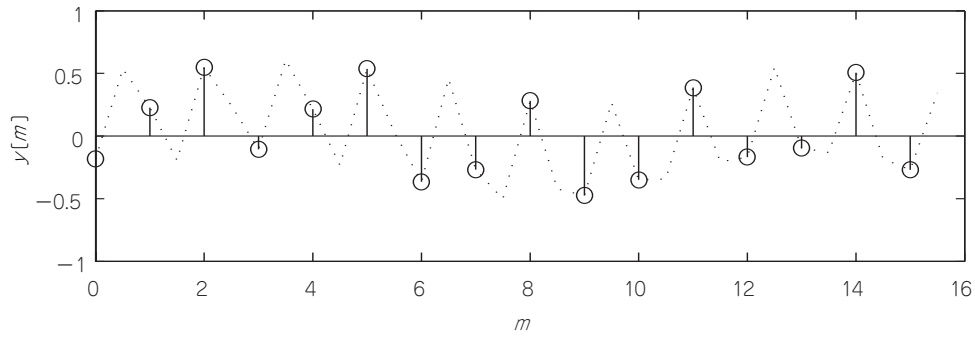
間引き率2 のダウン・サンプリングを実行しよう．

解

以下にMATLABによる処理例を示す．`inputSeq`は入力信号を保存する配列である．



（**a**）入力数列



（**b**）出力数列

図6.12　間引き率2のダウン・サンプリングの例

```
dFactor = 2; % 間引き率
outputSeq = downsample(inputSeq, dFactor);
```

処理結果の例を**図6.12**に示す．なお，点線は入力数列の値を直線で結んだものである．奇数番目のサンプルが単に捨てられるだけの結果となる．

実習6.4　**ダウン・サンプラ**

M-file：practice06_4.m

間引き率2以上の，より一般的なダウン・サンプリングを実行しよう．

(2) ダウン・サンプラと周波数スペクトラム

ダウン・サンプラのみでもレートを下げるという目的は達成される．しかし，これは単にサンプリング・レートを下げるというだけである．元の信号がもつ周波数スペクトラムは折り返し(エリアジング)成分によるひずみをまともに受けてしまう．

例題6.1で示した画像の縮小処理も，エリアジングの影響を受けていると解釈できる．この点を理

解するために，ダウン・サンプリングと周波数スペクトラムとの関係を見てみよう．

ダウン・サンプラの入力信号$x[n]$と出力信号$y[m]$は共に数列である．それぞれのz変換を$X(z)$，$Y(z)$として，これらの間の関係を見てみよう．

例題6.5　ダウン・サンプラの入出力関係

間引き率2のダウン・サンプラの入出力関係をz変換領域で表現してみよう．

解

まず，入力数列$x[n]$のうち，棄却する奇数番目のサンプルをゼロ値に置き換えよう．この数列は，

$$u[n]=\frac{1}{2}\{x[n]+(-1)^n x[n]\}$$

のように$x[n]$とそのπ変調数列$e^{j\pi n}x[n]=(-1)^n x[n]$の平均操作で与えられる．$u[n]$の$z$変換$U(z)$は，

$$U(z)=\frac{1}{2}\{X(z)+X(-z)\}$$

と表現される．結局，$U(z)$を介して，$X(z)$と$Y(z)$の関係が

$$\begin{aligned}Y(z)&=\sum_{m=-\infty}^{\infty}y[m]z^{-m}\\&=\sum_{m=-\infty}^{\infty}u[2m]z^{-m}\\&=\sum_{m=-\infty}^{\infty}u[n]z^{-\frac{n}{2}}\\&=U(z^{\frac{1}{2}})=\frac{1}{2}\{X(z^{\frac{1}{2}})+X(-z^{\frac{1}{2}})\}\end{aligned}$$

と導かれる．

より一般的に間引き率がDの場合は，$2\pi k/D$変調成分間の平均$U(z)=\frac{1}{D}\sum_{k=0}^{D-1}X(e^{-j\frac{2\pi k}{D}}z)$を用いて

$$Y(z)=\frac{1}{D}\sum_{k=0}^{D-1}X\left(W_D^k z^{\frac{1}{D}}\right) \quad \cdots\cdots (6.2)$$

という関係が導かれる．ただし，$W_D=e^{-j\frac{2\pi}{D}}$である．

(3) 周波数スペクトラム

式(6.2)に$z=e^{j\omega}$を代入すると，離散時間フーリエ変換領域での入出力関係が

$$Y(e^{j\omega})=\frac{1}{D}\sum_{k=0}^{D-1}X\left(e^{j\frac{\omega-2\pi k}{D}}\right) \quad \cdots\cdots (6.3)$$

のように導かれる．$k=0$以外の変調成分は，ナイキスト周波数以下の低いサンプリング・レートの標本化で生じるエリアジングそのものである(**図6.13**)．

■ $D=2$のとき

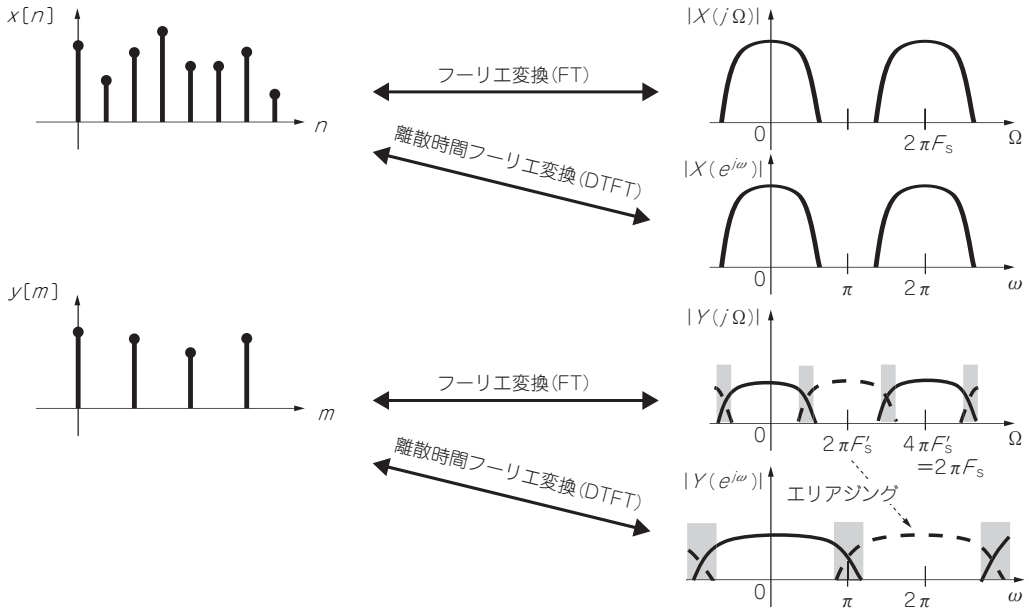


図6.13　間引き率2のダウン・サンプリングの周波数スペクトラム

例題6.6　**ダウン・サンプラと周波数スペクトラム**

間引き率2のダウン・サンプリングを実行し，その周波数スペクトラムの変化を見てみよう．

解

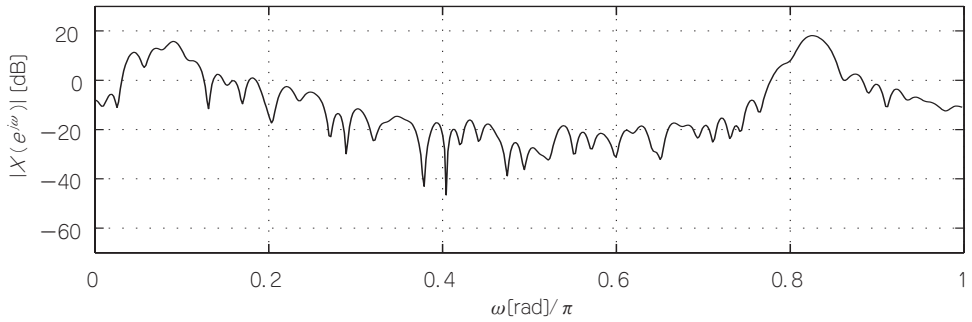
間引き率2の場合，ダウン・サンプル前後の信号間の周波数スペクトラムは

$$Y(e^{j\omega})=\frac{1}{2}\{X(e^{j\frac{\omega}{2}})+X(-e^{j\frac{\omega}{2}})\}$$

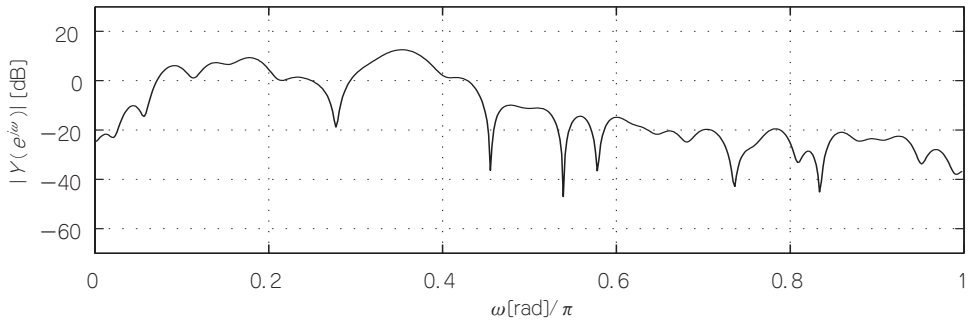
のように関係する．以下はMATLABによる処理例である．例題6.4の解で示した処理の後に実行する．

```
subplot(2,1,1);
[H,W] = freqz(inputSeq);
plot(W/pi,20*log10(abs(H)));
subplot(2,1,2);
[H,W] = freqz(outputSeq);
plot(W/pi,20*log10(abs(H)));
```

図6.14に処理結果の例を示す．



（**a**）入力数列



（**b**）出力数列

図6.14　間引き率2のダウン・サンプリングの周波数スペクトラム例

図6.14より，ダウン・サンプリングによってスペクトラムの形状が引き伸ばされた上，ひずむことが確認できる．

実習6.5　**ダウン・サンプルと周波数スペクトラム**

M-file：practice06_5.m

間引き率2以上のより一般的なダウン・サンプリングを実行し，その周波数スペクトラムの変化を見てみよう．

● アップ・サンプラ

アップ・サンプラは，サンプリング・レートを整数倍に上げる処理要素である．その動作から**ゼロ値挿入器**とも呼ばれる．また，この処理そのものを**アップ・サンプリング**，あるいは**ゼロ値挿入処理**と呼ぶ．Uを正の整数としてサンプリング・レートをU倍に上げる場合，アップ・サンプラは連続する二つのサンプル間に$U-1$個のゼロ値サンプルを挿入する．

(1) 入出力関係

アップ・サンプラの入出力関係を式で表すと，

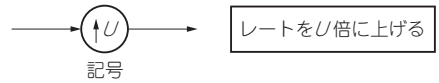


図6.15　アップ・サンプラ（ゼロ値挿入器）

$$y[m] = \begin{cases} x\left[\dfrac{m}{U}\right] & m = Un \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (6.4)$$

となる．また，本稿におけるアップ・サンプラの記号を**図6.15**に示す．

例題6.7　**アップ・サンプラ**

補間率2のアップ・サンプリングを実行しよう．

解

以下にMATLABによる処理例を示す．`inputSeq`は入力信号を保存する配列である．

```
uFactor = 2;
outputSeq = upsample(inputSeq,uFactor);
```

処理結果の例を**図6.16**に示す．なお，点線は入力数列の値を直線で結んだものである．奇数番目のサンプルとして単にゼロ値が挿入されるだけの結果となる．

実習6.6　**アップ・サンプラ**

M-file：practice06_6.m

補間率3以上の，より一般的なアップ・サンプリングを実行しよう．

(2) アップ・サンプラと周波数スペクトラム

アップ・サンプラのみでもレートを上げるという目的は達成できる．しかし，単にゼロ値を挿入するだけではなく，新たに挿入されたサンプル点がもともとのサンプル点の間を滑らかに補間することが望まれる．例題6.3で示したゼロ次ホールド処理はその一例である．

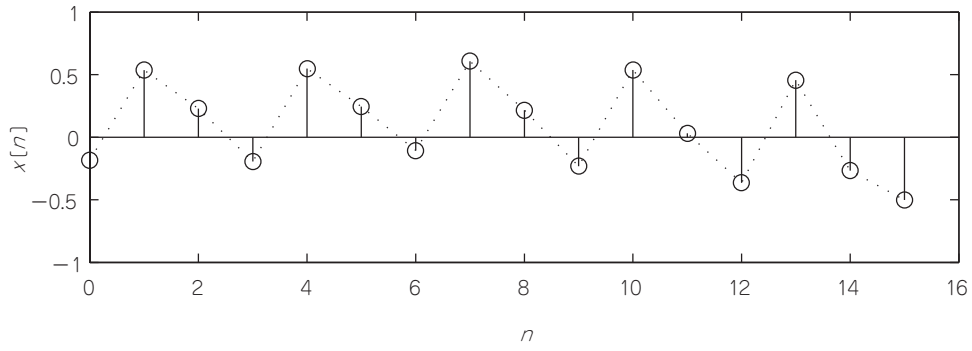
後述するように，アップ・サンプリングは周波数スペクトラムに**イメージング**を生ずる．ゼロ次ホールド処理は，このイメージングを抑圧する効果を有している．この点を理解するために，アップ・サンプリングと周波数スペクトラムとの関係を見てみよう．

例題6.8　**アップ・サンプラの入出力関係**

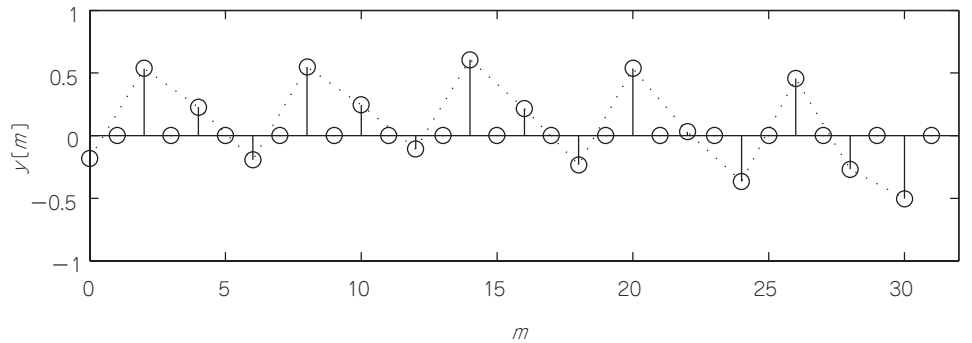
補間率2のアップ・サンプラの入出力関係をz変換領域で表現してみよう．

解

アップ・サンプラの出力$y[m]$のz変換$Y(z)$は，



（**a**）入力数列



（**b**）出力数列

図6.16　補間率2のアップ・サンプリングの例

$$
\begin{aligned}
Y(z) &= \sum_{m=-\infty}^{\infty} y[m]z^{-m} \\
&= \sum_{n=-\infty}^{\infty} y[2n]z^{-2n} \quad (\because y[2n+1]=0) \\
&= \sum_{n=-\infty}^{\infty} x[n]z^{-2n}
\end{aligned}
$$

のように展開される．従って，

$Y(z) = X(z^2)$

という関係が導かれる．

より一般的に，補間率がUの場合には，

$$Y(z) = X(z^U) \quad \cdots\cdots (6.5)$$

という関係が導かれる．

(3) 周波数スペクトラム

式(6.5)に$z = e^{j\omega}$を代入すると，離散時間フーリエ変換領域の入出力関係が

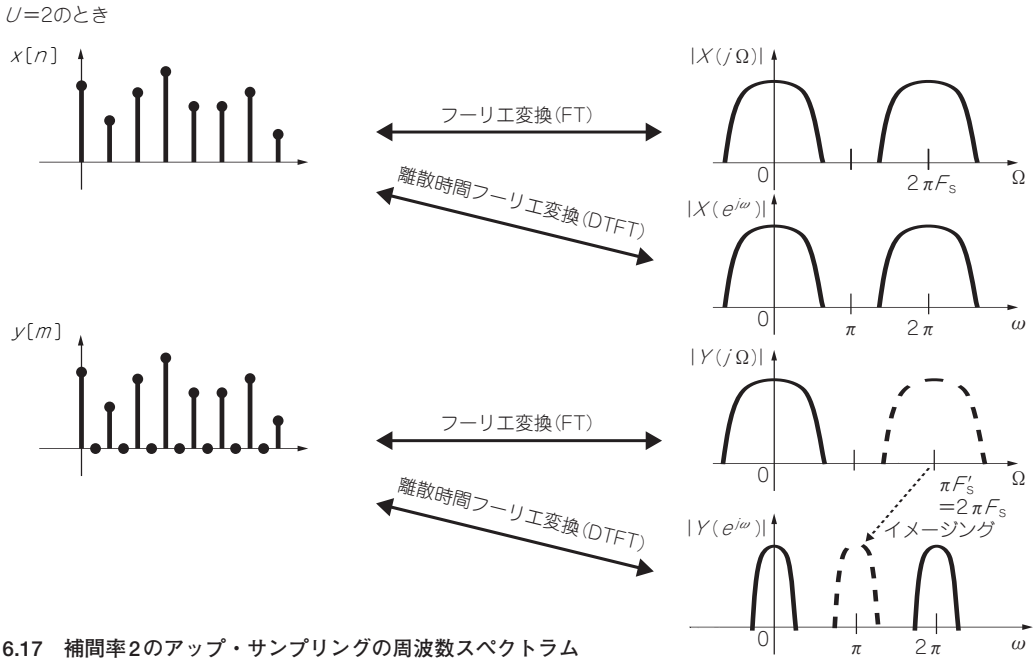


図6.17　補間率2のアップ・サンプリングの周波数スペクトラム

$$Y(e^{j\omega}) = X(e^{jU\omega}) \quad \cdots\cdots (6.6)$$

のように導かれる(**図6.17**)．上式は，正規化周波数軸を$1/U$倍に縮めることを意味する．いま，$x[n]$のサンプリング・レートが$F_S = 8$[kHz]であったと仮定しよう．$x[n]$を補間率$U = 2$でアップ・サンプリングすると，$F'_S = UF_S = 2 \times 8$[kHz] $= 16$[kHz]が新しいサンプリング・レートとなる．もともとは低周波成分のコピーであった8[kHz]($\omega = 2\pi$)付近のスペクトラムが高周波成分($\omega = \pi$)として存在することになる．これがイメージング成分となる．補間率Uのアップ・サンプリングでは，0から2π(すなわち，0からF'_S[Hz])の間に$U-1$個のイメージングが生じる．

例題6.9　**アップ・サンプラと周波数スペクトラム**

補間率2のアップ・サンプリングを実行し，その周波数スペクトラムの変化を見てみよう．

解

補間率2の場合，アップ・サンプル前後の信号間の周波数スペクトラムは

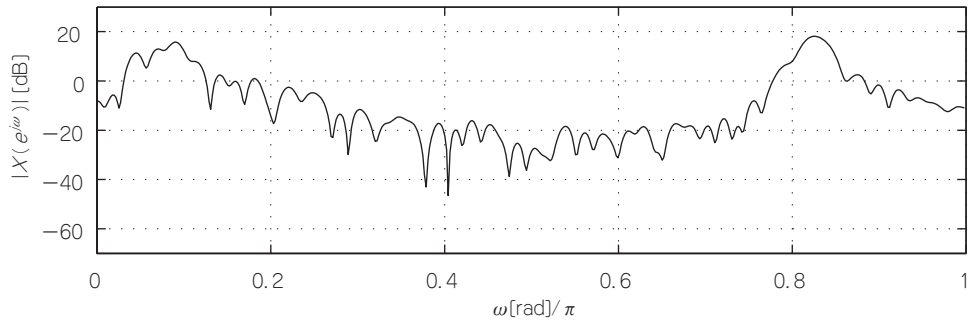
$$Y(e^{j\omega}) = X(e^{j2\omega}) \quad \cdots\cdots (6.7)$$

のように関係する．**図6.18**にMATLABによる処理結果の例を示す．

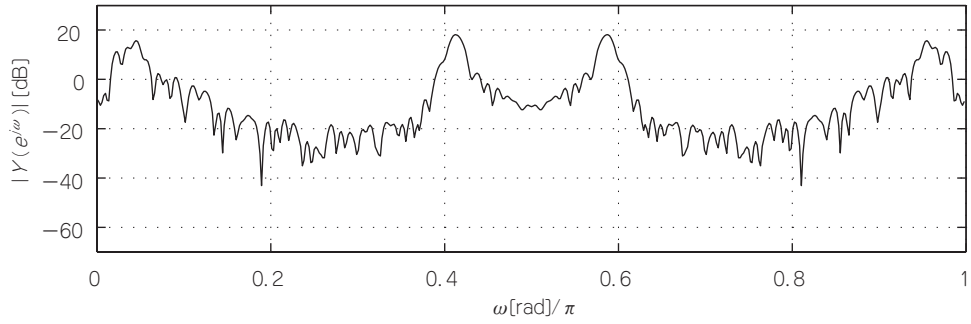
実習6.7　**アップ・サンプルと周波数スペクトラム**

M-file：practice06_7.m

補間率3以上の，より一般的なアップ・サンプリングを実行し，その周波数スペクトラムの変化を見てみよう．



(**a**) 入力数列



(**b**) 出力数列

図6.18　補間率2のアップ・サンプリングの周波数スペクトラム例

6.3　レート変換器

本節では，ダウン・サンプラやアップ・サンプラを使ったレート変換器の基本構成について概説しよう．以下では，整数分の1にレートを下げる**デシメータ**，整数倍にレートを上げる**インターポレータ**，そしてこれらの組み合わせによる**有理数比レート変換器**について，順に説明する．

● デシメータ

デシメータは，整数分の1にレートを下げるレート変換器である．その基本構成は，**図6.19**に示されるようにフィルタとダウン・サンプラからなる．このフィルタは，ダウン・サンプリングによって生じるエリアジングをあらかじめ回避する役割をもち，**デシメーション・フィルタ**と呼ばれる．

(1) 理想フィルタ

ダウン・サンプリングによって生じるエリアジングは，$2\pi k/D$変調成分の折り返しによって生じ

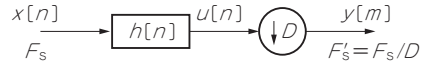


図6.19　デシメータの基本構成

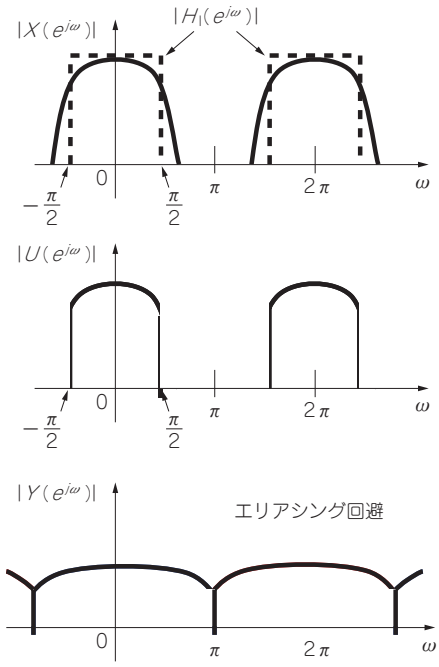


図6.20
間引き率2のデシメータの周波数スペクトラム(DTFT)

る．ダウン・サンプラへの入力信号があらかじめ$\pm\pi/D$で帯域制限されていれば，スペクトラムの重なりが生じない．従って，デシメーション・フィルタ$h[n]$の理想周波数応答$H_1(e^{j\omega})$は

$$H_1(e^{j\omega}) = \begin{cases} 1 & |\omega| < \dfrac{\pi}{D} \\ 0 & \dfrac{\pi}{D} \le |\omega| < \pi \end{cases} \quad \cdots\cdots (6.8)$$

のように与えられる(**図6.20**)．

(2) 平均フィルタ

式(6.8)の理想特性を持つフィルタはsinc関数を標本化したものとして与えられ，無限のインパルス応答をもち，実現不可能である．そこで実際には，この理想特性を近似したフィルタを利用する．

$$h[n] = \begin{cases} \dfrac{1}{D} & 0 \le n < D \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (6.9)$$

と表現されるフィルタは間引き処理を平均操作で与えるが，これは近似フィルタの一種と解釈できる．

例題6.10　平均フィルタの周波数応答

$D=2$の場合について，式(6.9)で与えられる平均フィルタの周波数振幅応答を見てみよう．

解

以下にMATLABのコマンド例を示す．

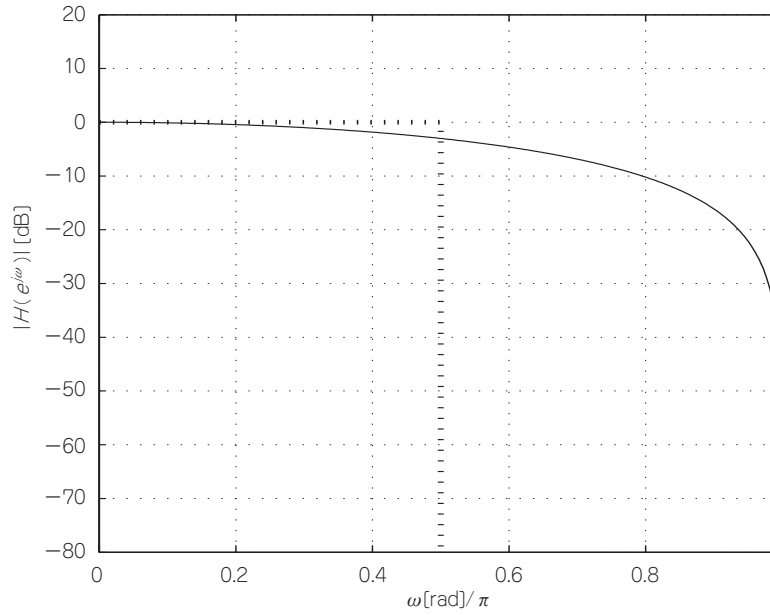


図6.21
平均フィルタ($D=2$)の周波数振幅応答

```
dFactor = 2; % 間引き率
averageFilter = ones(dFactor,1)/dFactor;
[H,W] = freqz(averageFilter);
plot(W/pi,20*log10(abs(H)));
```

図6.21に結果を示す．大ざっぱではあるが，$\pi/2$よりも高い周波数成分をカットすることが分かる．

実習6.8　**平均フィルタの周波数応答**

M-file：practice06_8.m

$D>2$の場合について，式(6.9)で与えられる平均フィルタの周波数振幅応答を見てみよう．

● インターポレータ

インターポレータは，整数倍にレートを上げるレート変換器である．その基本構成は，**図6.22**に示されるようにアップ・サンプラとフィルタからなる．このフィルタは，**インターポレーション・フィルタ**と呼ばれ，アップ・サンプリングによって生じるイメージングを抑圧し，サンプル間を滑らかに補間する役割をもつ．

(1) 理想フィルタ

アップ・サンプリングによって生じるイメージングは，$2\pi/U$ごとに現れる．アップ・サンプリン

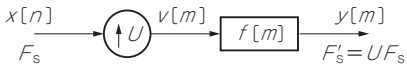


図6.22　インターポレータの基本構成

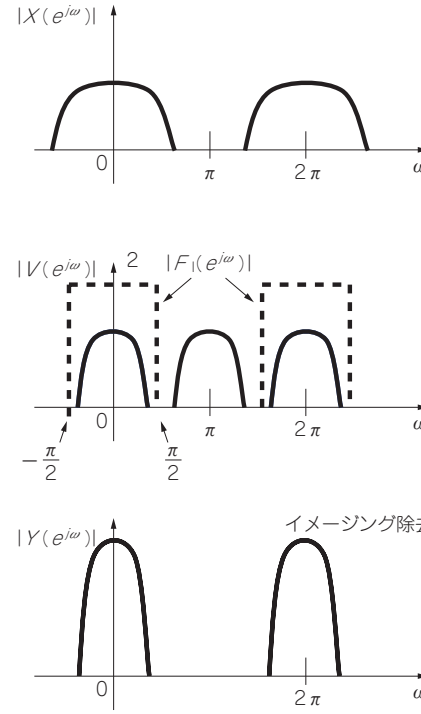


図6.23
補間率2のインターポレータの周波数スペクトラム（DTFT）

グ後の信号$v[m]$を$\pm\pi/U$で帯域制限すれば，2π（すなわち$F'_S = UF_S$[Hz]）ごとのスペクトラムのみを取り出し，イメージング成分を除去できる．もともとF'_Sで標本化された信号を得ることが目的なので，振幅を調整するための利得Uをかけることに注意すると，インターポレーション・フィルタ$f[m]$の理想周波数応答は，

$$F_I(e^{j\omega}) = \begin{cases} U & |\omega| < \dfrac{\pi}{U} \\ 0 & \dfrac{\pi}{U} \le |\omega| < \pi \end{cases} \quad \cdots\cdots (6.10)$$

と与えられる（**図6.23**）．

(2) ゼロ次ホールド・フィルタ

式(6.10)の理想特性をもつフィルタは，理想デシメーション・フィルタと同じように実現不可能である．そこで実際には，理想特性を近似したフィルタを利用することになる．

$$f[m] = \begin{cases} 1 & 0 \le m < U \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (6.11)$$

と表現されるフィルタはゼロ次ホールド処理を与える．これは一種の近似となっている．

(3) 線形補間フィルタ

$$f[m] = \begin{cases} 1 - \dfrac{|m|}{U} & |m| < U \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (6.12)$$

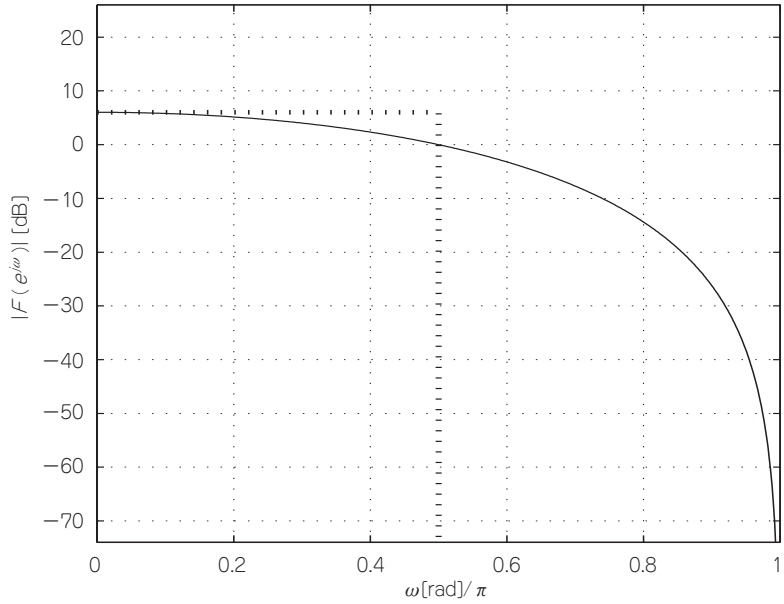


図6.24
線形補間フィルタ($U=2$)の周波数振幅応答

と表現されるフィルタは，線形補間処理を与える．これも一種の近似となっている．

例題6.11　**線形補間フィルタの周波数応答**

$U=2$の場合について，式(6.12)で与えられる線形補間フィルタの周波数振幅応答を見てみよう．

解

以下にMATLABのコマンド例を示す．

```
% 補間率2 の線形補間フィルタ
lineIntpFilter = [ 1 2 1 ]/ 2;
% 周波数振幅応答表示
[H,W] = freqz(lineIntpFilter);
plot(W/pi,20*log10(abs(H)));
```

図6.24に結果を示す．直流利得が2(約6[dB])であり，$\pi/2$よりも高い周波数成分をカットしている．

実習6.9　**線形補間フィルタの周波数応答**

M-file：practice06_9.m

$U=2$の場合について，式(6.12)で与えられる線形補間フィルタの周波数振幅応答を見てみよう．

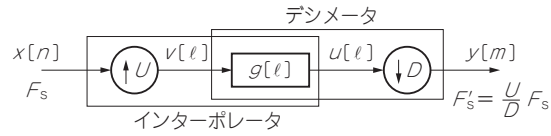


図6.25
有理数比レート変換器の基本構成

● 有理数比レート変換

前節で紹介したデシメータは，整数分の1にレートを下げるレート変換器であり，インターポレータは，整数倍にレートを上げるレート変換器である．これらを組み合わせることで，有理数比レート変換を実現できる．例えば，44.1kHzのレートをもつCDデータと96kHzのレートを持つDVD Audioデータの間のフォーマット変換が可能となる．

有理数比レート変換器の基本構成は，**図6.25**に示されるようにインターポレータとデシメータを縦続接続し，インターポレーション・フィルタとデシメーション・フィルタを共有した形となる．通常，演算やフィルタ設計の効率化のため，UとDについて，互いに素の関係のものを選択する．

(1) 理想フィルタ

アップ・サンプリングによって生じるイメージングは，$2\pi/U$ごとに現れる．アップ・サンプリング後の信号v[ł]を$\pm\pi/U$で帯域制限すれば，イメージング成分を除去できる．また，信号u[ł]があらかじめ$\pm\pi/D$で帯域制限されていれば，ダウン・サンプリングによって生じるエリアジングは生じない．従って，インターポレーションで必要となる利得Uに注意すると，有理数比レート変換フィルタg[ł]の理想周波数応答$G_{\mathrm{I}}(e^{j\omega})$は，

$$G_{\mathrm{I}}(e^{j\omega}) = \begin{cases} U & |\omega| < \min\left(\dfrac{\pi}{U}, \dfrac{\pi}{D}\right) \\ 0 & \min\left(\dfrac{\pi}{U}, \dfrac{\pi}{D}\right) \le |\omega| < \pi \end{cases} \quad \cdots\cdots (6.13)$$

と与えられる(**図6.26**)．

(2) 線形補間フィルタと平均フィルタの縦続接続

式(6.13)で表されるフィルタは実現不可能である．そこで，実際は近似フィルタを利用することになる．例えば，式(6.9)で表される平均フィルタと式(6.12)で表される線形補間フィルタを縦続接続することで，簡単な近似フィルタを与えることができる．

例題6.12　**縦続接続フィルタの周波数応答**

$U = 2$，$D = 3$の場合について，式(6.12)と式(6.9)で与えられるフィルタを縦続接続した場合の周波数振幅応答を見てみよう．

解

MATLAB上でのコマンド例を以下に示す．


```
% 直線補間フィルタのインパルス応答
lineIntpFilter = [1 2 1]/2;
```

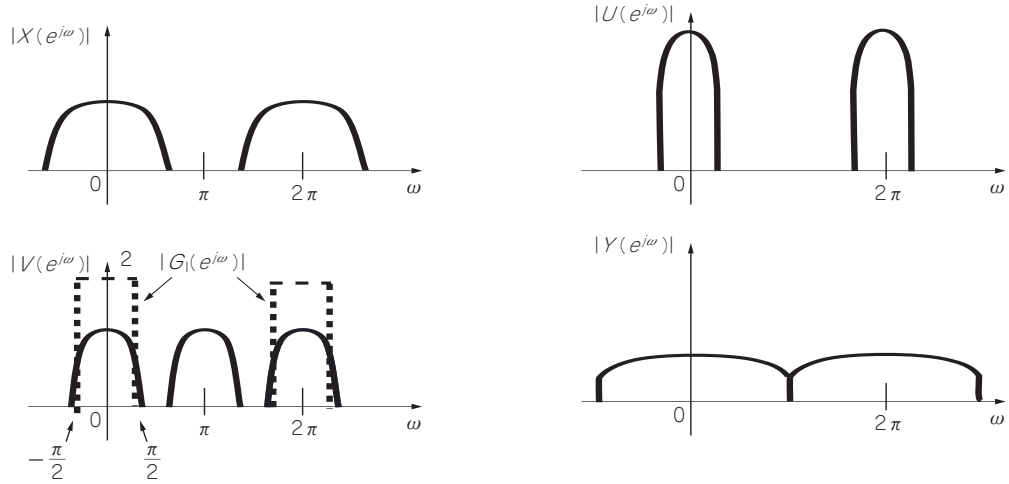


図6.26　倍率*U*/*D*＝2/3の有理数比レート変換器の周波数スペクトラム(DTFT)

```
% 平均フィルタのインパルス応答
averageFilter = [1 1 1]/3;
% 全フィルタのインパルス応答
totalFilter = ...
  conv(lineIntpFilter, averageFilter)
% 周波数振幅特性の表示
[H,W] = freqz(totalFilter);
plot(W/pi,20*log10(abs(H)));
```

図6.27に結果を示す．直流利得が2(約6[dB])であり，$\pi/3$よりも高い周波数成分をカットしている．

実習6.10　**線形補間フィルタの周波数応答**

M-file：practice06_10.m

$U=2$，$D=3$以外の場合についても，平均フィルタと線形補間フィルタで与えられる近似フィルタの周波数振幅応答を見てみよう．

● フレーム・レート変換

日米のテレビ放送方式(NTSC)と欧州のテレビ放送方式(PAL/SECAM)では，映像の解像度やフレーム・レートが異なる．また，テレビ放送と映画の間にも映像フォーマットの違いがある．これら異なるフォーマットの映像データを統一的に扱うためには，互いの変換技術が必要となる．画像の解像度変換については次節で解説することとし，ここではレート変換技術をフレーム・レート変

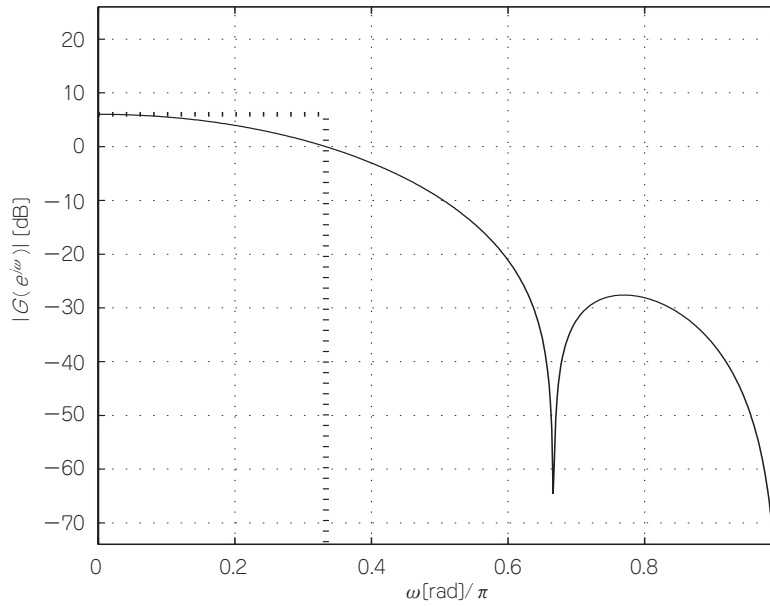


図6.27
線形補間フィルタ（$U=2$）と平均フィルタ（$D=3$）の縦続接続の周波数振幅応答

換に応用する例を紹介しよう．

例題6.13　**フレーム・レート変換**

フレーム・レート30［fps］の映像をフレーム・レート25［fps］の映像に変換しよう．

解

画素ごとに，$U/D=25/30=5/6$の1次元のレート変換を施せばよい．利得が$U=5$となり，カットオフ周波数が$\pm\pi/6$となる近似フィルタが必要となる．例えば，次の近似フィルタを利用できる（例題6.19，章末問題6.7を参照）．

$$g[0]=1,\ g[\pm1]=\frac{7}{8},\ g[\pm2]=\frac{5}{8},\ g[\pm3]=\frac{3}{8},\ g[\pm4]=\frac{1}{8}$$

このフィルタを用いると，**図6.28**のようにフレーム・レート変換を実現できる．

図6.28では，アップ・サンプリングによって挿入されるゼロ値フレームに対する処理やダウン・サンプリングによって棄却されるフレームの計算を省略している．これらは**ポリフェーズ分解**と呼ばれる手法である．次節にて詳しく説明する．

また，実際の映像では動きとスペクトラムの関係を理解する必要がある．動き補償の必要性も含めて，適切なフィルタの選択に関する解説については，次章以降で行う．

実習6.11　**フレーム・レート変換**

M-file：practice06_11.m

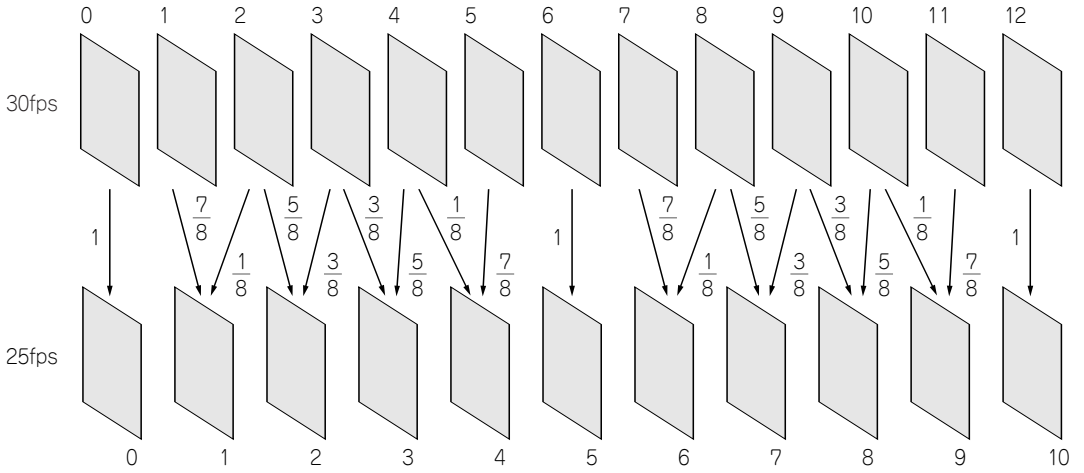


図6.28　フレーム・レート変換(30［fps］→25［fps］)**の例**(矢印の添え字は重み付け和の重み係数)

例題6.13のフレーム・レート変換を実現してみよう．

6.4　効果的実現法

これまで1次元のマルチレート信号処理について概説した．いよいよ画像の拡大縮小処理との関係について解説しよう．また，マルチレート・システムの諸性質を紹介し，レート変換器の効率的な実現を可能とするポリフェーズ構成について説明しよう．

● 分離処理

画像は2次元信号と解釈できる．水平，垂直の軸に対して，独立に1次元のレート変換を施すことで，画像の拡大縮小処理，すなわち解像度変換を実現できる．**図6.29**に垂直間引き率2，水平間引き率2のデシメーションの分離処理の概略を示す．拡大処理も同じように，各軸にインターポレーションを施せばよい．一般に，2次元の畳み込み演算を実行するよりも，分離処理で実現した方が演算量が少なくなる．

(1) 平均操作による縮小処理

例題6.2で示した平均操作による縮小処理は，各次元に対して式(6.9)のデシメーション・フィルタを用いた間引き率2のデシメーションを施していることと等価である．一見，例題6.2では畳み込み

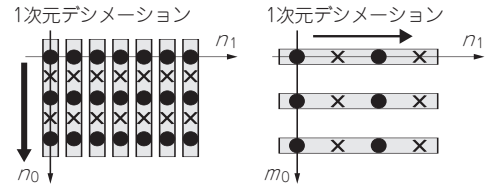
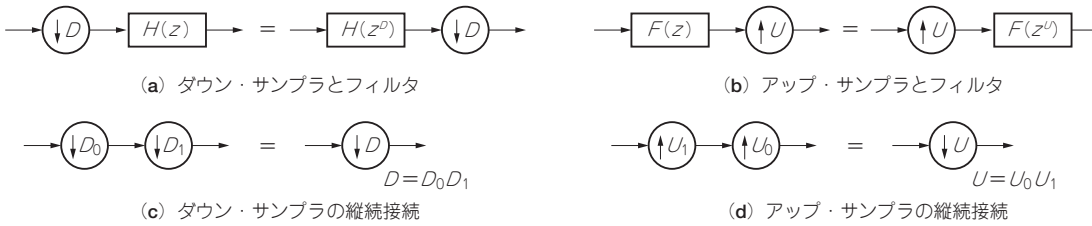


図6.29
垂直間引き率2，水平間引き率2のデシメーションの分離処理



(a) ダウン・サンプラとフィルタ　(b) アップ・サンプラとフィルタ

(c) ダウン・サンプラの縦続接続　(d) アップ・サンプラの縦続接続

図6.30　レート変換の等価関係

演算が行われていないように感じるが，ブロック分割後の平均操作は以下で紹介するデシメータのポリフェーズ分解による実現と解釈できる．

(2) ゼロ次ホールドによる拡大処理

例題6.3で示したゼロ次ホールドによる拡大処理は，各次元に対して式(6.11)のインターポレーション・フィルタを用いた補間率2のインターポレーションを施していることと等価である．例題6.3では畳み込み演算が行われていないように感じるが，各画素をブロックに置き換える操作は以下で紹介するインターポレータのポリフェーズ分解による実現と解釈できる．

● マルチレート・システムの諸性質

マルチレート信号処理システムを実現するに際に有用となるレート変換の等価関係を**図6.30**に示す．特に，**図6.30(a)**，**(b)**は**ノーブル恒等変換**(Noble Identity) と呼ばれており，マルチレート・システムの実現において重要な性質である．

● ポリフェーズ分解

本節では，レート変換器の効率的な実装について解説しよう．デシメーション・フィルタとダウン・サンプラからなるデシメータの基本構成では，ダウン・サンプラによって棄却されるサンプルも演算の対象となり，むだを生じる．一方，アップ・サンプラとインターポレーション・フィルタからなるインターポレータの基本構成においても，アップ・サンプラによって挿入されたゼロ値に対する演算がむだとなる．これらのむだを省く技術として，ポリフェーズ分解がある．

(1) デシメータのポリフェーズ分解

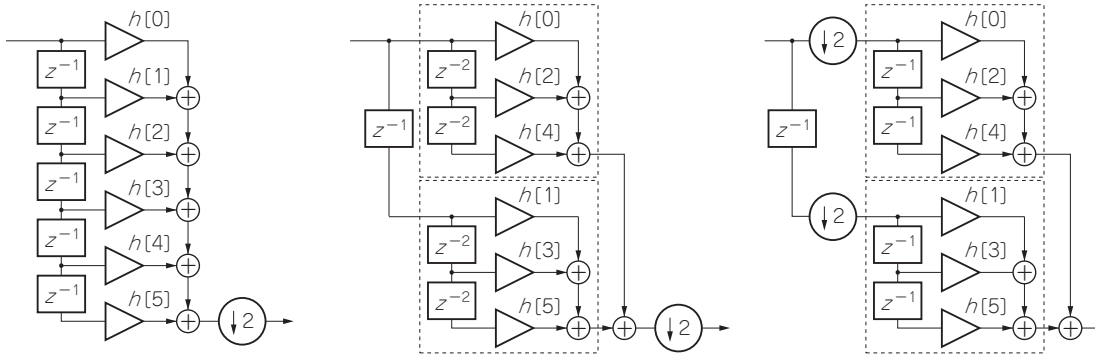
まず，ポリフェーズ構成によるデシメータの効果的実現について説明しよう．

例題6.14　**デシメータの効果的実現**

デシメーション・フィルタのタップ数が6，間引き率$D = 2$のデシメータをポリフェーズ分解により効果的に実現しよう．

解

図6.31(a)に，FIR直接構成でデシメーション・フィルタ$H(z)$を実現した例を示す．同構成では，ダウン・サンプラによって間引かれるサンプルまでも演算の対象となる．まず，**図6.31(b)**のように，偶数遅延と奇数遅延の係数に分けた構成に等価変換しよう．二つのサブフィルタはz^{-2}から構成されることが分かる．結果として，**図6.30(a)**の等価関係よりダウン・サンプラが各サブフィル



（a）FIR直接構成　（b）偶数遅延と奇数遅延の係数に分けた構成　（c）効果的構成

図6.31　デシメータのポリフェーズ構成

タ$E_0(z)$，$E_1(z)$の左へ動き，フィルタ処理の前にあらかじめサンプルが棄却され，むだが省かれる．

各サブフィルタ$E_0(z)$，$E_1(z)$は**タイプⅠのポリフェーズ・フィルタ**と呼ばれる．より一般的に，タイプⅠのポリフェーズ分解は

$$H(z) = \sum_{d=0}^{D-1} E_d(z^D) z^{-d}$$

と表現される．ただし，

$$E_d(z) = \sum_{m=-\infty}^{\infty} h[Dm + d] z^{-m}$$

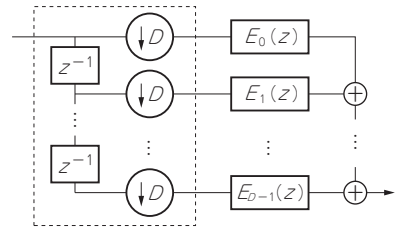
図6.32に一般的な間引き率Dのデシメータのポリフェーズ構成を示す．なお，遅延列とダウン・サンプラからなる構成部は，Dサンプルごとのシリアル-パラレル(S/P)変換器となっている．

(2) インターポレータのポリフェーズ分解

次に，ポリフェーズ分解によるインターポレータの効果的実現について説明しよう．

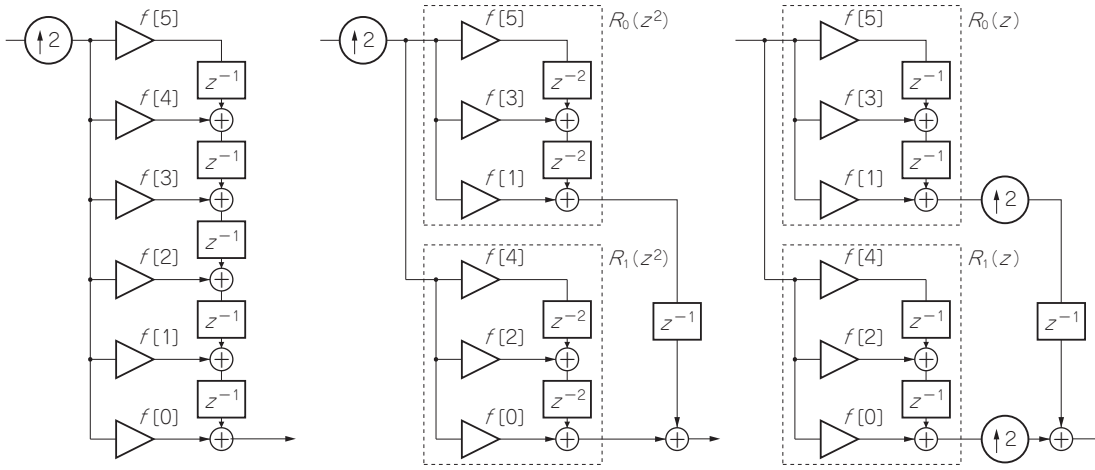
例題6.15　**インターポレータの効果的実現**

インターポレーション・フィルタのタップ数が6，補間率$U = 2$のインターポレータをポリ



シリアル-パラレル変換

図6.32　タイプⅠのポリフェーズ分解



（**a**）FIR反転構成　（**b**）偶数遅延と奇数遅延の係数を分けた構成　（**c**）効果的構成

図6.33　インターポレータのポリフェーズ構成

フェーズ分解により効果的に実現しよう．

解

図6.33(**a**)に，インターポレーション・フィルタ$F(z)$をFIR反転構成で実現した例を示す．同構成では，アップ・サンプラによって挿入されたゼロ値サンプルまでも演算の対象となる．まず，**図6.33**(**b**)のように，偶数遅延と奇数遅延の係数に分けた構成に等価変換しよう．二つのサブフィルタはz^{-2}から構成されることが分かる．結果として，**図6.30**(**b**)の等価関係よりアップ・サンプラが各サブフィルタ$R_0(z)$，$R_1(z)$の右へ動き，ゼロ値に対するフィルタ処理が省略される．

各サブフィルタ$R_0(z)$，$R_1(z)$は**タイプⅡのポリフェーズ・フィルタ**と呼ばれる．より一般的に，タイプⅡのポリフェーズ分解は，

$$F(z)=\sum_{u=0}^{U-1}R_u(z^U)z^{-(U-1-u)}$$

と表現される．ただし，

$$R_u(z)=\sum_{m=-\infty}^{\infty}f[U(m+1)-1-u]z^{-m}$$

図6.34に一般的な補間率Uのインターポレータのポリフェーズ構成を示す．なお，アップ・サンプラと遅延列からなる構成部は，Uサンプルごとのパラレル-シリアル(P/S)変換器となっている．

(3) 有理数倍率レート変換器のポリフェーズ分解

UとDが互いに素である場合，**図6.35**に示すようにアップ・サンプラとダウン・サンプラの順序を交換できる．有理数倍率のレート変換器についても，先に示したポリフェーズ分解と**図6.35**の性質を利用することで，効果的な実現が可能となる．

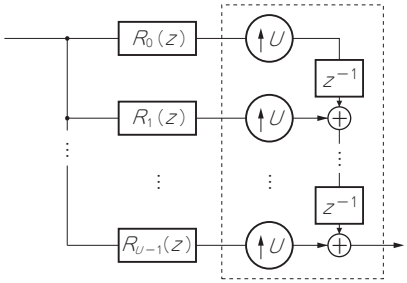


図6.34　タイプⅡのポリフェーズ構成

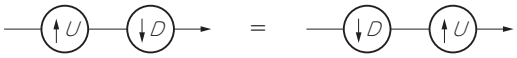


図6.35　アップ・サンプラとダウン・サンプラの交換

例題6.16　**有理数比レート変換の効果的実現**

フィルタのタップ数が6，補間率$U = 2$，間引き率$D = 3$の有理数比レート変換器のポリフェーズ構成を導出しよう．

解

図6.36(a)は，**図6.36(b)**を経て**図6.36(c)**を得る．

6.5　レート変換フィルタ設計

ここでは，レート変換フィルタの設計に有効な**固有フィルタ設計法**を紹介しよう．固有フィルタ設計法は周波数領域の設計仕様に加えて，時間領域の制約を課すことが容易なことが知られている．なお，レート変換フィルタの近似を得るためには，窓関数設計法，線形計画法，チェビシェフ・フィルタなどの一般的なフィルタ設計法が利用できる．これについては，『シミュレーションで学ぶディジタル信号処理』(CQ出版社 刊)を参照していただきたい．

● 固有フィルタ設計

固有フィルタ設計(Eigenfilter Design)法はFIRフィルタ最適化設計手法の一種で，周波数領域の理想応答との自乗誤差を目的関数としている．時間領域の制約を与えやすいという利点を持つため，後述するようにナイキスト・フィルタ設計に利用できる．

(1) 線形位相低域通過固有フィルタ

線形位相FIR低域通過フィルタ$g[\ell]$を考えよう．この伝達関数は$G(z) = \sum_{1=0}^{N} g[\ell]z^{-1}$と表現され，$g[\ell] = g[N-\ell]$を満たす．さらに，フィルタ次数$N$が偶数であるとすると．$G(z)$の周波数振幅応答は，

$$G_R(\omega) = \left|G(e^{j\omega})\right| = \sum_{n=0}^{N/2} b_n \cos\omega n = \mathbf{b}^T\mathbf{c}(\omega) \quad \cdots\cdots (6.14)$$

と与えられる．ただし，

$$\mathbf{b} = \left(b_0 \quad b_1 \quad \cdots \quad b_{\frac{n}{2}}\right)^T$$

$$= \left(2g\left[\frac{N}{2}\right] \quad g\left[\frac{N}{2}+1\right] \quad \cdots \quad g[N]\right)^T \quad \cdots\cdots (6.15)$$

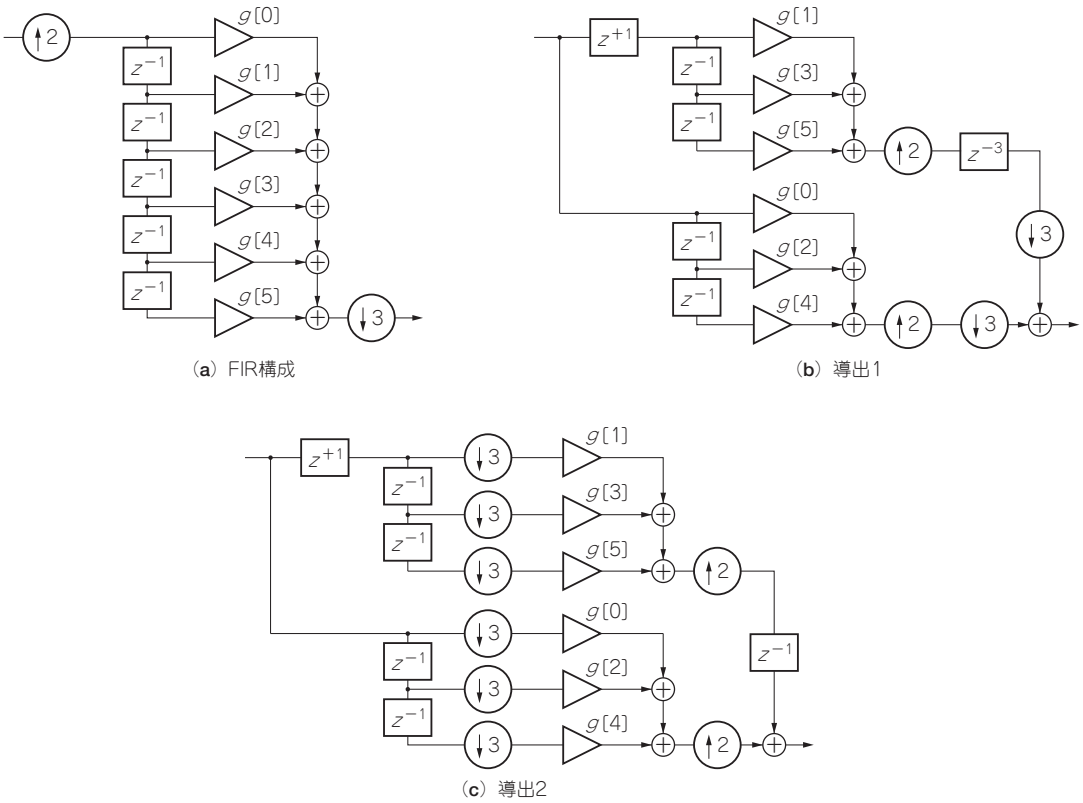


図6.36　有理数比レート変換のポリフェーズ構成

$$\mathbf{c}(\omega) = \left(1 \quad \cos\omega \quad \cdots \quad \cos\frac{N\omega}{2}\right)^T \qquad (6.16)$$

である．

通過域の誤差エネルギE_Pと阻止域の誤差エネルギE_Sは，それぞれ

$$E_P = \int_0^{\omega_P} \left|G_R(0) - G(e^{j\omega})\right|^2 \frac{d\omega}{\pi}$$
$$= \mathbf{b}^T(\mathbf{1} - \mathbf{c}(\omega))(\mathbf{1} - \mathbf{c}(\omega))^T\mathbf{b} = \mathbf{b}^T\mathbf{Q}\mathbf{b} \qquad (6.17)$$

$$E_S = \int_{\omega_s}^{\pi} \left|G_R(e^{j\omega})\right|^2 \frac{d\omega}{\pi}$$
$$= \mathbf{b}^T\mathbf{c}(\omega)\mathbf{c}^T(\omega)\mathbf{b} = \mathbf{b}^T\mathbf{P}\mathbf{b} \qquad (6.18)$$

と与えられる．ただし，**1**は要素がすべて1の$(N/2+1)$次元ベクトルであり，**Q**，**P**は$(N/2+1)\times(N/2+1)$行列である．それぞれの(m, n)番目要素$q_{m,n}$と$p_{m,n}$は，通過域端ω_Pと阻止域端ω_Sから

$$q_{m,n} = \int_0^{\omega_P} (1-\cos m\omega)(1-\cos n\omega)\frac{d\omega}{\pi} \qquad (6.19)$$

$$p_{m,n} = \int_{\omega_S}^{\pi} \cos m\omega \cos n\omega \frac{d\omega}{\pi} \quad \cdots\cdots (6.20)$$

と与えられる．

固有フィルタ設計法では目的関数ϕを

$$\phi = \alpha E_S + (1-\alpha)E_P = \mathbf{b}^T\mathbf{R}\mathbf{b} \quad \cdots\cdots (6.21)$$

と定義する．ただし，αは通過域と阻止域の重みを制御するパラメータで，$0 < \alpha < 1$である．また，

$$\mathbf{R} = \alpha\mathbf{P} + (1-\alpha)\mathbf{Q} \quad \cdots\cdots (6.22)$$

である．

Rは，実で対称な正定値行列であり，ϕを最小化する単位ノルム・ベクトル**b**が，**R**の最小の固有値に対応する固有ベクトルとして与えられ，**べき乗法**(power method)を用いて計算できる．

● デシメーション・フィルタ

早速，固有フィルタ設計によるデシメーション・フィルタを設計してみよう．

例題6.17　**デシメーション・フィルタの設計**

間引き率$D = 2$のデシメーション・フィルタを固有フィルタ設計法により設計し，インパルス応答と周波数振幅応答を見てみよう．ただし，

- 通過域端：$\omega_P = \pi/D \times (1 - 0.2)$
- 阻止域端：$\omega_S = \pi/D \times (1 + 0.2)$
- 誤差エネルギ評価の重み：$\alpha = 0.5$
- フィルタの次数：$N = 30$

とする．

解

図6.37にMATLABによる設計例を示す．**図6.21**に示す平均フィルタの周波数振幅応答と比べてみよう．なお，固有フィルタ設計を実装した`eigLpFir`関数を用意したので参照していただきたい．

```
dFactor = 2; % 間引き率（正整数）
pass = 1/dFactor*(1-0.2); % 通過域端
stop = 1/dFactor*(1+0.2); % 阻止域端
alpha = 0.5; % 重みパラメータ
nOrder = 30; % フィルタ次数
% 固有フィルタ設計
h = eigLpFir(nOrder,pass,stop,alpha);
```

実習6.12　**デシメーション・フィルタの設計**

M-file：practice06_12.m

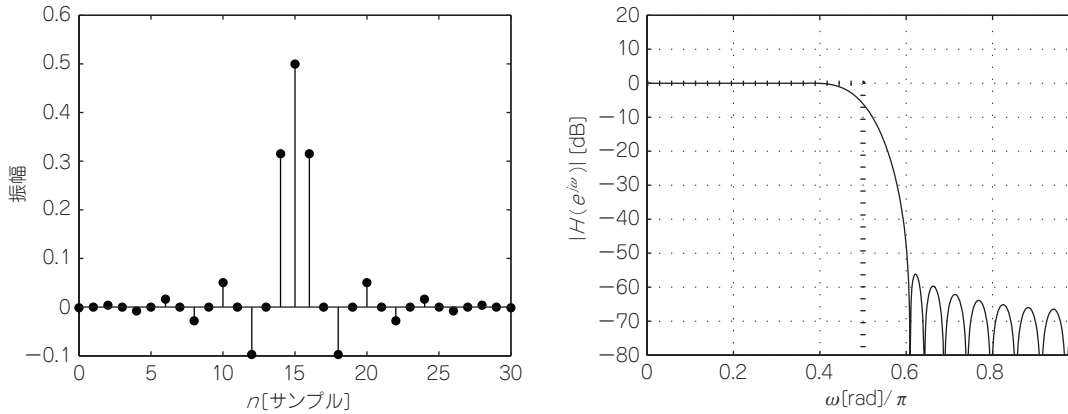


図6.37　固有フィルタ設計法による間引き率2のデシメーション・フィルタの設計例

各種パラメータを変えてフィルタを設計してみよう．

(1) その他のフィルタ設計法

MATLABでは，デシメーションを行う関数として`decimate`が用意されており，デフォルトでタイプIのチェビシェフ・フィルタ(`cheby1`関数)が適用される．関数`decimate`でFIRをオプション指定すると，Hamming窓を用いたFIRフィルタ窓関数法による設計(`fir1`関数)が適用される．

● インターポレーション・フィルタ

インターポレーション・フィルタの理想特性を式(6.10)に示したが，さらに時間領域における制約として，オリジナルのサンプル値を保持するナイキスト・フィルタが望ましい．以下で解説しよう．

(1) ナイキスト・フィルタ

補間率Mのインターポレーションを考えよう．もし，インターポレーション・フィルタのインパルス応答が，あるk $(0 \le k \le M-1)$に対して

$$f[Mn+k] = \begin{cases} 1 & n = n_k \\ 0 & n \neq n_k \end{cases} \quad \cdots\cdots (6.23)$$

という条件を満たす任意の整数n_kを持つ場合，補間前のオリジナルのサンプル値が補間後もその値を保つ．$k = 0$に対して$n_k = 0$の場合が単純で理解しやすい．インパルス応答が時刻0で‘1’となり，Mの整数倍の時刻では‘0’となることを意味する．このようなフィルタを**M-thバンド・フィルタ**あるいは**ナイキスト(M)フィルタ**と呼ぶ．

(2) 固有フィルタ設計

固有フィルタ設計法は時間領域の制約を与えやすい．式(6.23)のナイキスト(M)フィルタの条件も例外ではない．実際に，式(6.23)の制約を

$$b_{Mn} = \begin{cases} 1 & n = 0 \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (6.24)$$

のようにb_nに課すことで設計できる．具体的には，$b_n = 0$となる要素を取り除いて得られるベクトル$\hat{\mathbf{b}}$と対応する列ベクトル，行ベクトルを取り除いた行列$\hat{\mathbf{R}}$を新たに定義し，$\hat{\mathbf{b}}^{\mathrm{T}}\hat{\mathbf{R}}\hat{\mathbf{b}}$を$\mathbf{b}^{\mathrm{T}}\mathbf{b} = 1$の制約のもとで目的関数を最小化すればよい．

例題6.18　**インターポレーション・フィルタの設計**

補間率$U = 2$のインターポレーション・フィルタ（ナイキスト(2)フィルタ）を固有フィルタ設計法により設計し，インパルス応答と周波数振幅応答を見てみよう．ただし，以下のとおりとする．

- 通過域端：$\omega_P = \pi/U \times (1 - 0.2)$
- 阻止域端：$\omega_S = \pi/U \times (1 + 0.2)$
- 誤差エネルギ評価の重み：$\alpha = 0.5$
- フィルタの次数：$N = 30$

解

図6.38にMATLABによる設計例を示す．**図6.24**に示す線形補間フィルタの周波数振幅応答と比べてみよう．なお，ナイキスト・フィルタ設計を実装した`eigLpNqFir`関数を用意したので参照してほしい．

```
uFactor = 2; % 補間率（正整数）
pass = 1/uFactor*(1-0.2); % 通過域端
stop = 1/uFactor*(1+0.2); % 阻止域端
alpha = 0.5; % 重みパラメータ
nOrder = 30; % フィルタ次数
% 固有フィルタ設計
f = uFactor * eigLpNqFir(...
nOrder,pass,stop,alpha,uFactor);
```

実習6.13　**インターポレーション・フィルタの設計**

M-file：practice06_13.m

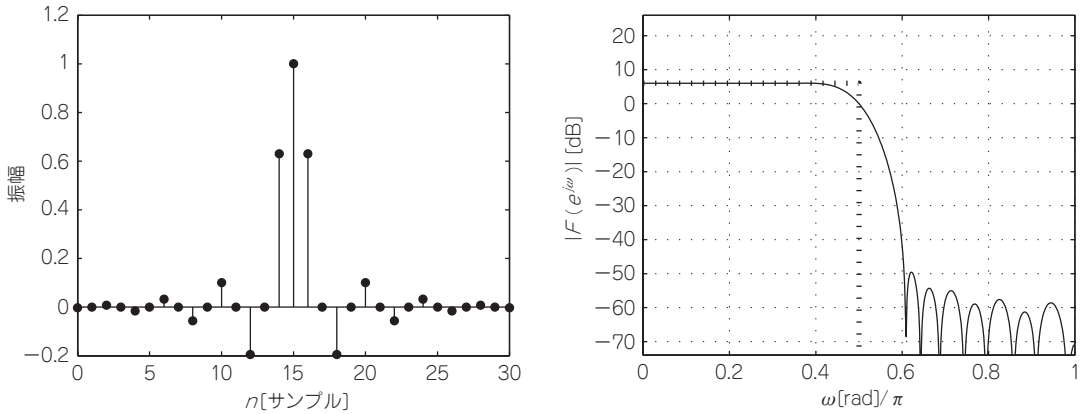
各種パラメータを変えてフィルタを設計してみよう．

(3) その他のフィルタ設計法

MATLABでは，インターポレーション関数として`interp`が用意されており，デフォルトでFIRの最小自乗誤差設計（`firls`関数）によるナイキスト・フィルタ（`intfilt`関数）が適用される．

有理数比レート変換フィルタ

固有フィルタ設計による有理数比レート変換フィルタを設計してみよう．



（a）インパルス応答　　（b）周波数振幅応答

図6.38　固有フィルタ設計法による補間率2のインターポレーション・フィルタの設計例

例題6.19　有理数比レート変換器の設計

レート変換比$U/D = 5/6$のレート変換フィルタを固有フィルタ設計法により設計し，インパルス応答と周波数振幅応答を見てみよう．ただし，以下のとおりとする．

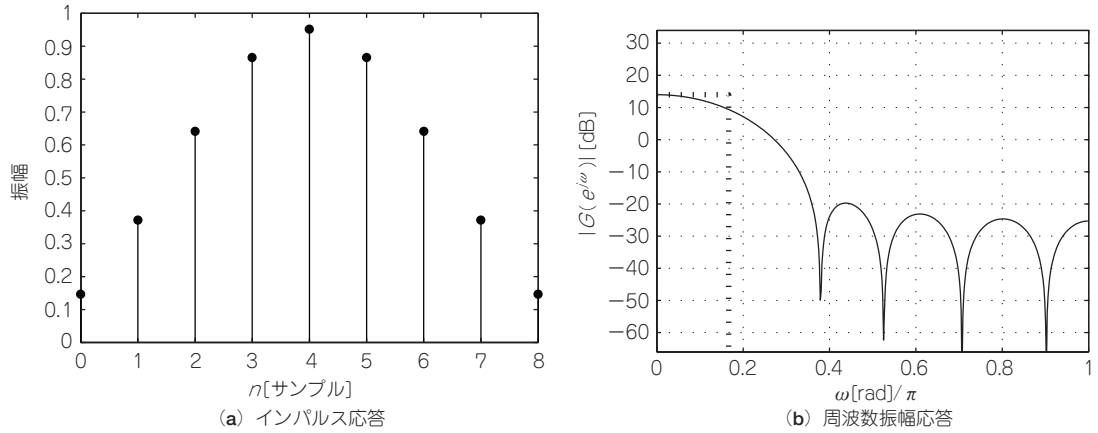
- 通過域端：$\omega_P = min(\pi/U, \pi/D) \times (1 - 1)$
- 阻止域端：$\omega_S = min(\pi/U, \pi/D) \times (1 + 1)$
- 誤差エネルギ評価の重み：$\alpha = 0.5$
- フィルタ次数：$N = 8$

解

図6.39にMATLABによる設計例を示す．

```
uFactor = 5; % 補間率
dFactor = 6; % 間引き率
sFactor = max(uFactor,dFactor);
pass = 1/sFactor * (1 - 1); % 通過域端
stop = 1/sFactor * (1 + 1); % 阻止域端
alpha = 0.5; % 重みパラメータ
nOrder = 8; % フィルタ次数
% 固有フィルタ設計
g = uFactor * eigLpNqFir(...
  nOrder,pass,stop,alpha,uFactor*dFactor);
```


有理数比のレート変換の場合，オリジナルの標本値を残すためには$M = U \times D$としてナイキスト(M)



（a）インパルス応答　（b）周波数振幅応答

図6.39　固有フィルタ設計法による $U=5$，$D=6$ の有理数比レート変換

フィルタを設計すればよい．

実習6.14　**有理数比レート変換フィルタの設計**

M-file：practice06_14.m

各種パラメータを変えてフィルタを設計してみよう．

(1) その他のフィルタ設計法

MATLABでは，有理数比レート変換関数 upfirdn 関数が既に用意されている．また，フィルタの自動設計機能のついた resample 関数も用意されている．resample 関数では，デフォルトでFIRフィルタの最小自乗誤差設計（firls 関数）によるフィルタが適用される．なお，upfirdn は後述するポリフェーズ構成によって実現されている．Image Processing Toolbox がある場合，imresize 関数で直接画像の拡大縮小処理を実行できる．詳細はマニュアルを参照していただきたい．

(2) 境界処理

レート変換にはフィルタリングが伴うため，画像に適用する場合，空間の境界部においてゼロ値拡張や対称拡張などの対処が必要となることに注意されたい．

章末問題

問題6.1　**画像縮小の分離処理**（MATLAB演習）

解像度を垂直1/2，水平1/2にする画像の縮小を分離処理で行う解像度変換器を設計してみよう．decimate 関数を用いてもよい．また，各種パラメータを変えながら比較・検討してみよう．

問題6.2　**画像拡大の分離処理**（MATLAB演習）

解像度を垂直2倍，水平2倍にする画像の拡大を分離処理で行う解像度変換器を設計してみよう．interp関数を用いてもよい．また，各種パラメータを変えながら比較・検討してみよう．

問題6.3　**ゼロ次ホールド処理**（MATLAB演習）

式(6.11)の1次元ゼロ次ホールド・フィルタの周波数振幅応答を見てみよう．

問題6.4　**有理数比解像度変換の分離処理**（MATLAB演習）

垂直1080×水平1920の解像度を持つ画像から垂直480×水平720の解像度を持つ画像を得るための解像度変換器を設計してみよう．また，この逆の変換器も設計してみよう．

問題6.5　**フレーム・レート変換**（MATLAB演習）

例題6.13のフィルタ$g[\mathrm{l}]$の周波数振幅特性を調べてみよう．また，フレーム・レート変換が**図6.28**のように実現できることを確かめてみよう．

問題6.6　**DVD Audio→CD Audioレート変換器**（MATLAB演習）

96.0kHzで標本化されたDVD Audio信号を標本化周波数44.0kHzの信号に変換するレート変換器を設計しよう．なお，実際のCD Audio信号の標本化周波数は44.1kHzである．しかし，このレート変換を用いても約0.2%の誤差で収まる．具体的には，44.0kHzで標本化されている3分の曲を44.1kHzで再生したとしても，再生時間が約0.4秒だけ速くなるに過ぎない．

問題6.7　**フレーム・レート変換フィルタ(1)**（MATLAB演習）

例題6.19で設計したフィルタ係数に8を掛け，四捨五入により整数化し，再度8で割ると，例題6.13のフィルタ$g[\mathrm{l}]$が得られることを確かめてみよう．

問題6.8　**フレーム・レート変換フィルタ(2)**（MATLAB演習）

フレーム・レート25[fps]の映像を30[fps]の映像に変換するフィルタを設計してみよう．

参考文献

(1) P.P. Vaidyanathan；*Multirate Systems and Filter Banks*，Prentice Hall，1993.
(2) 貴家仁志；マルチレート信号処理，昭晃堂，1995年.
(3) 貴家仁志；よくわかるディジタル画像処理，CQ出版社，1996年.
(4) 尾知 博；シミュレーションで学ぶディジタル信号処理，CQ出版社，2001年.

第7章

DCTとウェーブレット変換

本章では，画像・映像符号化に欠かせないツールとなった離散コサイン変換（DCT：Discrete Cosine Transform）と離散時間ウェーブレット変換（DWT：Discrete-time Wavelet Transform）について解説しよう．DCTやDWTに代表される変換技術は，マルチメディア信号処理のほか，TDM（Time Division Multiplexing）-FDM（Frequency Division Multiplexing）変換器やOFDM（Orthogonal Frequency Division Multiplex）などの通信用信号処理にも必要不可欠な要素技術となっている．これら変換技術を使いこなせるように，行列演算とフィルタリングという二つの視点をもって説明を進めよう．

7.1 信号変換の必要性

DCTやDWTの詳細について説明する前に，信号変換が必要とされる理由について具体例を挙げながら概説しよう．

● まずは試してみよう

まずは，1次元信号の変換から復習しよう．既に第2章で解説した離散フーリエ変換（DFT）も信号変換の一種である．DCTの結果と比べてみよう．

例題7.1 DFTとDCT

次の16点の信号$x[n]$の16点DFTと16点DCTの結果を比較してみよう．

$$x[n] = 2\cos\left(\frac{\pi}{8}n\right) + \cos\left(\frac{\pi}{4}n\right),\ 0 \le n < 16$$

解

以下に，MATLABによる処理例を示す．


```
%% 原信号 x[n] の生成
```

```
n=0:15;
x(n+1)= 2*cos(pi/8*n) + cos(pi/4*n);
% 原信号x[n] の表示
figure(1);
stem(n,x);
xlabel('n'); ylabel('x[n]');
title('Original sequence');
% DFT の計算と結果表示
ydft = fft(x);
figure(2);
stem(n,abs(ydft));
xlabel('k'); ylabel('Y_{DFT}[k]');
title('DFT (Magnitude)');
% DCT の計算と結果表示
ydct = dct(x);
figure(3);
stem(n,ydct);
xlabel('k'); ylabel('Y_{DCT}[k]');
title('DCT');
```

図7.1 に原信号とそのDFTおよびDCTの結果を示す．

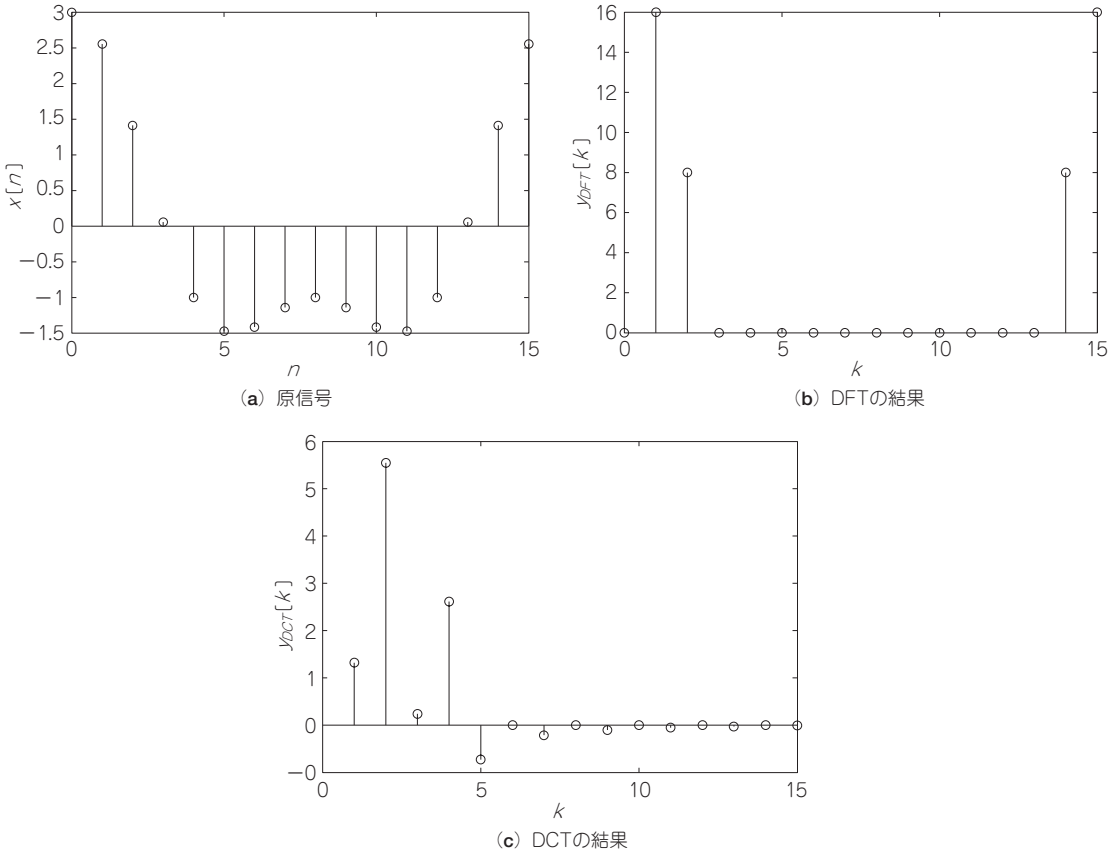
いま，**図7.1**(**a**)の原信号$x[n]$が二つの余弦波から構成されていることをあらかじめ知っている．しかし，**図7.1**(**a**)の波形を見ただけで，この事実に気がつくだろうか？ **図7.1**(**b**)に示すDFTは，$x[n]$が大きさの異なる周波数の低い二つの余弦波(あるいは正弦波)から構成されることを明確に見せている．このように，一見複雑な信号も変換によって特徴が明確になることがある．

また，**図7.1**(**c**)に示すDCTも，DFTの左半分の係数($0 \leqq k < 8$)に類似した結果を与えることに気づくだろう．DFTは，原信号が実数であっても複素数の係数を与える．一方，DCTは，原信号が実数であれば，係数もまた実数となる．周波数解析の能力に加え，実数の結果を与えるという点は，DCTが多くの支持を受けている一つの理由である．DFTとDCTの関係については，後ほど詳しく解説しよう．

実習7.1 **DFTとDCT**

M-file：practie07_1.m

原信号$x[n]$を変えてDFTとDCTの結果を比較してみよう．特に，次の16点の信号



（**a**）原信号　（**b**）DFTの結果

（**c**）DCTの結果

図7.1　DFTとDCT

$$x[n] = 2\cos\left(\frac{\pi}{8}n + \frac{\pi}{16}\right) + \cos\left(\frac{\pi}{4}n + \frac{\pi}{8}\right),\ 0 \le n < 16$$

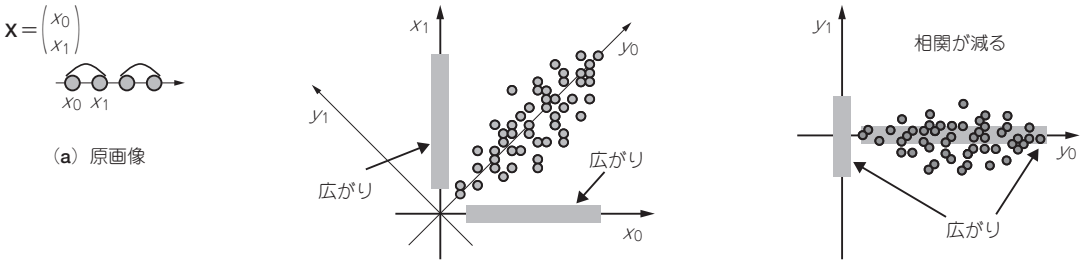
ではどのようになるか？

● 画素間の相関と変換

画像信号の隣接する画素の間には強い相関があるといわれている．画像や映像の圧縮技術では，このような相関をたくみに利用して画質を保ちつつ，データ容量を削減している．

画像の隣接する画素をペアにして**図7.2**(**a**)に示すように2次元のベクトルを構成し，これらを直交平面上にプロットすると，**図7.2**(**b**)に示すような散布図が得られる．すなわち，45度の対角上にデータが集中するような散布図である．これは，x_0とx_1の間の関係(相関)が強く，x_0が与えられるとx_1が予測しやすいことを意味している．予測のしやすさはむだな情報が含まれることを意味する．

ベクトル$x = (x_0, x_1)^T$に，



（**a**）原画像　（**b**）変換前の散布図　（**c**）変換後の散布図

図7.2　画素間の相関と変換

$$\begin{pmatrix}y_0\\y_1\end{pmatrix}=\begin{pmatrix}\cos\left(-\frac{\pi}{4}\right) & -\sin\left(-\frac{\pi}{4}\right)\\ \sin\left(-\frac{\pi}{4}\right) & \cos\left(-\frac{\pi}{4}\right)\end{pmatrix}\begin{pmatrix}x_0\\x_1\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\-1 & 1\end{pmatrix}\begin{pmatrix}x_0\\x_1\end{pmatrix} \quad \cdots\cdots(7.1)$$

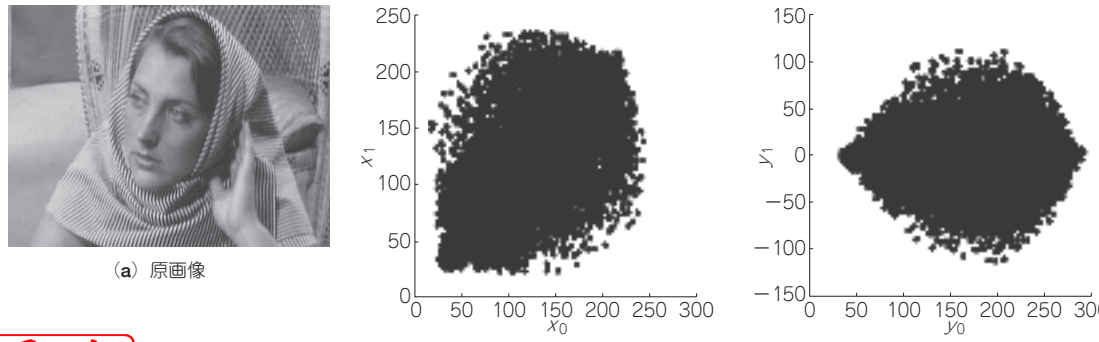
という変換を施してみよう．すると，ベクトル $\mathbf{y}=(y_0, y_1)^T$ の散布図は，**図7.2**（**c**）のように**図7.2**（**b**）を－45度回転させたものとして与えられる．y_0 は x_0 や x_1 の1.4倍程度広がるが，y_1 の広がりは小さくなる．また，y_0 と y_1 の相関が減り，一方から他方を予測することが難しくなる．むだな情報が省かれ，重要な情報が y_0 に集中することを意味する．

例題7.2　**散布図と信号変換**

図7.3（**a**）の画像について，水平方向に画素のペアを組んで2次元ベクトルを構成し，その散布図を描いてみよう．また，式(7.1)の変換を施した後の散布図も描いてみよう．

解

MATLABのコマンド例を以下に示す．`pictureGray` はモノクロ画像のワークスペース上の参照とする．



（**a**）原画像　（**b**）変換前の散布図　（**c**）変換後の散布図

図7.3　画素間の相関と変換

```
% 2次元ベクトル集合の抽出
nPixels = prod(size(pictureGray));
setOfX = ...
    reshape(pictureGray.', 2, nPixels/2);
% 変換前の散布図
figure(1);
scatter(setOfX(1,:),setOfX(2,:),'.');
axis square;
xlabel('x_0'); ylabel('x_1');
title('Scatter plot (before transform)');
% 信号変換
setOfY = 1 / sqrt(2) * [ 1 1 ; -1 1 ] ...
    * double(setOfX);
% 変換後の散布図
figure(2);
scatter(setOfY(1,:),setOfY(2,:),'.');
axis square;
xlabel('y_0'); ylabel('y_1');
title('Scatter plot (after transform)');
```

図7.3(**b**)，(**c**)に変換前と変換後の散布図をそれぞれ示す．**図7.3**(**b**)，(**c**)より，信号変換は，周波数解析のほかにも，統計的な性質(相関)を変えるという重要な機能をもつことが分かる．

実習7.2　**散布図と信号変換**

M-file：practice07_2.m

ほかの画像についても試してみよう．

7.2　信号変換の基礎

信号変換の基本は行列演算である．ここでは行列演算の意味について，数学的に厳密な議論は避けて，直感的な解説を行いたいと思う．線形代数を学習したことのある読者は，簡単な復習と思って読み進めていただきたい．

● 変換の種類

M点の信号$x[n]$，$n=0$，1，2，…，$M-1$を考えよう．$M \times M$行列$\mathbf{A}$を使った$x[n]$の変換は，

$$\underbrace{\begin{pmatrix} y[0] \\ y[1] \\ \vdots \\ y[M-1] \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,M-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,M-1} \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} x[0] \\ x[1] \\ \vdots \\ x[M-1] \end{pmatrix}}_{\mathbf{x}} \quad \cdots\cdots (7.2)$$

もしくは，

$$y[k] = \sum_{n=0}^{M-1} a_{k,n}\, x[n],\ k = 0,1,\cdots,M-1 \quad \cdots\cdots (7.3)$$

と表現される．ただし，$a_{k,n}$は行列$\mathbf{A}$のk行n列目の要素とする．$y[k]$は**変換係数**と呼ばれる．

(1) 逆変換

もし，$\mathbf{A}$に逆行列が存在する場合，すなわち行列式$\det\mathbf{A}$が非ゼロ（非特異）の場合，逆行列$\mathbf{A}^{-1}$が存在し，

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} \quad \cdots\cdots (7.4)$$

のように変換係数ベクトル$\mathbf{y}$から$\mathbf{x}$を再構成可能である．

(2) ユニタリ変換

もし，複素共役転置$\mathbf{A}^{\dagger}$が逆行列の定数倍，すなわち$\mathbf{A}^{\dagger} = c\mathbf{A}^{-1}$ならば，

$$\mathbf{x} = \frac{1}{c}\mathbf{A}^{\dagger}\mathbf{y} \quad \cdots\cdots (7.5)$$

のように変換係数ベクトル$\mathbf{y}$から$\mathbf{x}$を再構成可能である．このような行列を**ユニタリ行列**と呼ぶ．また，ユニタリ行列による変換を**ユニタリ変換**とよぶ．複素共役転置の例を以下に示そう．

$$\begin{pmatrix} 1+j & j \\ 1-j & 2+j2 \end{pmatrix}^{\dagger} = \begin{pmatrix} 1-j & 1+j \\ -j & 2-j2 \end{pmatrix} \quad \cdots\cdots (7.6)$$

ただし，jは虚数単位である．

DFT行列はユニタリ行列の一種であり，$a_{k,n} = W_M^{kn} = e^{-j\frac{2\pi}{M}kn}$と定義され，

$$\mathbf{A}^{\dagger}\mathbf{A} = c\mathbf{A}^{-1}\mathbf{A} = c\mathbf{I} \quad \cdots\cdots (7.7)$$

が成立する．ただし，$\mathbf{I}$は単位行列である．

(3) 直交変換

行列$\mathbf{A}$が実数かつユニタリ行列ならば，転置行列$\mathbf{A}^{T}$が逆行列の定数倍$c\mathbf{A}^{-1}$となり，

$$\mathbf{x} = \frac{1}{c}\mathbf{A}^{T}\mathbf{y} \quad \cdots\cdots (7.8)$$

のように変換係数ベクトル$\mathbf{y}$から$\mathbf{x}$を再構成できる．このような行列を**直交行列**と呼ぶ．特に，$c = 1$の場合，**正規直交行列**と呼ぶ．また，直交行列による変換を**直交変換**と呼ぶ．

式(7.1)の変換行列は正規直交行列であり，

$$\mathbf{A}^{T}\mathbf{A} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \quad \cdots\cdots (7.9)$$

が成立する．

例題7.3　正規直交変換

次の行列$\mathbf{A}$が正規直交行列であることを確かめてみよう．

$$\mathbf{A} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

解

$\mathbf{A}^T\mathbf{A} = \mathbf{I}$が証明できればよい．数学的な証明については読者自身にお任せすることにして，以下ではMATLAB上での確認の例を示そう．

```
theta = pi/3; % 角度θ [rad]
A = [ cos(theta) -sin(theta) ; % 変換行列
sin(theta) cos(theta) ];
A.'*A  % A^T * A の実行
```

角度thetaを変えても結果は常に単位行列となる．

● 基底ベクトルと基底画像

(1) 基底ベクトル

逆行列

$$\mathbf{A}^{-1} = (\mathbf{b}_0 \ \mathbf{b}_1 \ \cdots \ \mathbf{b}_{M-1}) \quad \cdots\cdots (7.10)$$

の各列ベクトル$\mathbf{b}_k$は**基底ベクトル**と呼ばれる．信号ベクトル$\mathbf{x}$は，これら基底ベクトル$\mathbf{b}_k$の線形結合

$$\mathbf{x} = y[0]\mathbf{b}_0 + y[1]\mathbf{b}_1 + \cdots + y[M-1]\mathbf{b}_{M-1} \quad \cdots\cdots (7.11)$$

のように展開でき，重みは変換係数$y[k]$となる．

例題7.4　基底ベクトル

ベクトル$\mathbf{x} = (\sqrt{2}, 2\sqrt{2})^T$を行列

$$\mathbf{A}^{-1} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}^{-1} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

の列ベクトルの線形結合で表せることを確かめよう．

解

数式上の確認については読者自身にお任せすることにして，以下ではMATLAB上での確認の例を示そう．

```
x = sqrt(2)*[ 1 2 ].';      % ベクトルx の定義
A = [ 1 1 ; -1 1 ]/sqrt(2); % 変換行列
y = A * x;                  % 変換係数の計算
```

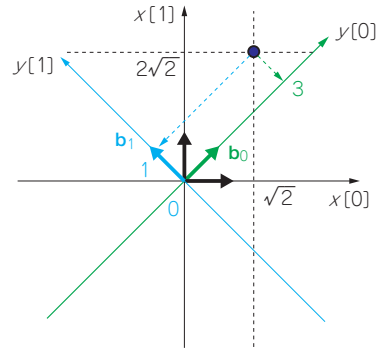


図7.4
信号変換と基底ベクトル

```
invA = inv(A);     % 変換行列の逆行列
b0 = invA(:,1);    % 基底ベクトルb0
b1 = invA(:,2);    % 基底ベクトルb1
y(1)*b0 + y(2)*b1 % 基底ベクトルの線形結合
```

ベクトル**x**が再構成される（**図7.4**を参照）．

(2) 基底画像

1次元の信号変換と同じように，画像のような2次元の信号変換も定義できる．順変換と逆変換はそれぞれ，

$$y[k_0,k_1]=\sum_{n_1=0}^{M_1-1}\sum_{n_0=0}^{M_0-1}a_{k_0,k_1,n_0,n_1}x[n_0,n_1] \quad \cdots\cdots (7.12)$$

$$x[n_0,n_1]=\sum_{k_1=0}^{M_1-1}\sum_{k_0=0}^{M_0-1}b_{n_0,n_1,k_0,k_1}y[k_0,k_1] \quad \cdots\cdots (7.13)$$

と表現される．配列

$$\mathbf{B}_{k_0,k_1}=\begin{pmatrix} b_{0,0,k_0,k_1} & b_{0,1,k_0,k_1} & \cdots & b_{0,M_1-1,k_0,k_1} \\ b_{1,0,k_0,k_1} & b_{1,1,k_0,k_1} & \cdots & b_{1,M_1-1,k_0,k_1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M_0-1,0,k_0,k_1} & b_{M_0-1,1,k_0,k_1} & \cdots & b_{M_0-1,M_1-1,k_0,k_1} \end{pmatrix} \quad \cdots\cdots (7.14)$$

は基底ベクトルに代わり，**基底画像**と呼ばれる．変換前の画像配列を**X**とすると，**X**はこれら基底画像$\mathbf{B}_{k_0,k_1}$の線形結合

$$\mathbf{X}=y[0,0]\mathbf{B}_{0,0}+y[0,1]\mathbf{B}_{0,1}+\cdots+y[M_0-1,M_1-1]\mathbf{B}_{M_0-1,M_1-1} \quad \cdots\cdots (7.15)$$

のように展開でき，重みは変換係数$y[k_0,k_1]$となる．

例題7.5 **基底画像**

図7.5(a)の画像配列**X**を**図7.5**(b)に示す基底画像$\mathbf{B}_{k_0,k_1}$の線形結合で表せることを確かめよう．

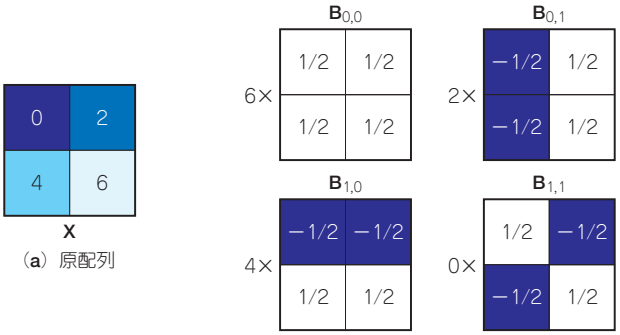


図7.5
画像配列の展開

解

数式上の確認については読者自身にお任せすることにして，以下ではMATLAB上での確認の例を示そう．

```
B00 = [ 1 1 ; 1 1 ]/2;    % 基底画像B00
B01 = [ -1 1 ; -1 1 ]/2; % 基底画像B01
B10 = [ -1 -1 ; 1 1 ]/2; % 基底画像B10
B11 = [ 1 -1 ; -1 1 ]/2; % 基底画像B11
6*B00 + 2*B01 + 4*B10 + 0*B11 % 線形結合
```

配列**X**が再構成される．

図7.5(**a**)の画像配列を**図7.5**(**b**)のように展開する方法については，以下で解説しよう．

● 分離処理

2次元の信号変換を各軸に独立な1次元の信号変換で実行する方法を，2次元信号変換の**分離処理**という．分離処理による2次元信号の順変換と逆変換はそれぞれ，

$$\mathbf{Y} = \mathbf{A}_0 \mathbf{X} \mathbf{A}_1^T \quad \cdots\cdots (7.16)$$

$$\mathbf{X} = \mathbf{A}_0^{-1} \mathbf{Y} \mathbf{A}_1^{-T} \quad \cdots\cdots (7.17)$$

と表現される．ただし，$\mathbf{Y}$は，k_0行k_1列目に$y[k_0, k_1]$を要素としてもつ係数配列であり，$\mathbf{A}_d$は各軸の信号変換を行う任意の非特異行列である．

分離処理の基底画像$\mathbf{B}_{k_0, k_1}$は，

$$\mathbf{B}_{k_0, k_1} = \mathbf{b}_{0, k_0} \otimes \mathbf{b}_{1, k_1}^T \quad \cdots\cdots (7.18)$$

と与えられる．ただし，$\mathbf{b}_{d, k}$は，行列$\mathbf{A}_d^{-1}$のk番目の列ベクトルであり，⊗はテンソル積(MATLABでは`kron`関数)を表す．

例題7.6　**分離処理の基底画像**

$$A_0 = A_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

と選択したときの基底画像を求めてみよう．

解

数式上の確認については読者自身にお任せすることにして，以下ではMATLAB上での確認の例を示そう．

```
A = [ 1 1 ; -1 1 ]/sqrt(2); % 変換行列
invA = inv(A);  % 逆行列（転置も可）
b0 = invA(:,1); % 基底ベクトルb0
b1 = invA(:,2); % 基底ベクトルb1
B00 = kron(b0,b0.') % 基底画像B00
B01 = kron(b0,b1.') % 基底画像B01
B10 = kron(b1,b0.') % 基底画像B10
B11 = kron(b1,b1.') % 基底画像B11
```

図7.5(b)に示す基底画像が与えられる．なお，行列**A**は直交行列なので，逆行列を求める際には単なる転置でもよい．

実習7.3　**分離処理の基底画像**

M-file：practice07_3.m

式(7.16)によって，**図7.5(b)**に示す線形結合の重み(変換係数)$y[k_0, k_1]$が与えられることを確かめてみよう．

● 変換符号化

信号変換の重要な応用に**変換符号化**がある．変換符号化は，静止画像符号化国際標準方式JPEGやJPEG2000に採用されるなど，今日のマルチメディア技術を支えている．

変換符号化はおおむね**図7.6**に示される構成をとる．上段が符号化部，下段が復号部を表している．前処理の部分では，色空間変換やレベル・シフト，ブロック分割などが行われる．順変換では，2次元の信号変換が施される．例えばJPEGの場合，サイズ8×8のブロックに分割された後，それぞれのブロックに対して8×8の2次元DCTが施される．量子化部では，変換係数ごとに異なる量子化が施される．ハフマン符号化や算術符号化などのエントロピ符号化が続き，係数間やブロック間に残る冗長性が取り除かれる．復号側では符号化と逆の手続きが取られる．

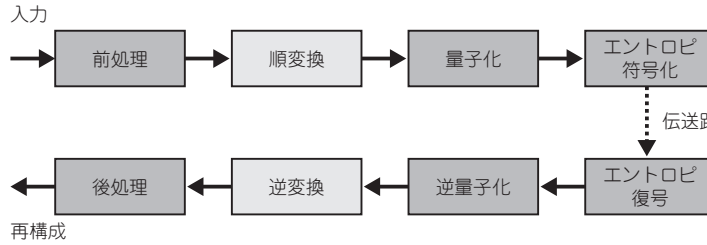


図7.6
変換符号化の概略図

2	2	3	1
2	2	3	1
3	3	2	0
1	1	0	2

(a) 画像配列

1	2
2	4

(b) 量子化テーブルQ

2	2	3	1
2	2	3	1
3	3	1	1
1	1	1	1

(c) 再構成画像

図7.7　変換符号化の例

例題7.7　変換符号化

図7.7(**a**)に示す4×4の画像配列をサイズ2×2のブロックに分割して，次のことを試してみよう．ただし，$[\cdot]_{k_0,k_1}$は配列のk_0行k_1列目の要素を意味する．

1. 例題7.6の2次元変換を施し，その結果として得られる変換係数配列$\mathbf{Y}$を求めてみよう．
2. **図7.7**(**b**)の量子化テーブル$\mathbf{Q}$を用いて，各係数に対する線形量子化を施してみよう．ただし，小数点以下第1位を切り捨てて，整数化を施す．

$$[\mathbf{S}]_{k_0,k_1} = \mathrm{round}\left(\frac{[\mathbf{Y}]_{k_0,k_1}}{[\mathbf{Q}]_{k_0,k_1}}\right)$$

3. 量子化後の各係数に対して逆量子化

$$[\overline{\mathbf{Y}}]_{k_0,k_1} = [\mathbf{Q}]_{k_0,k_1} \cdot [\mathbf{S}]_{k_0,k_1}$$

を行い，2次元の逆変換を施して，原画像を再構成してみよう．

解

以下にMATLABによる処理例を示す．なお，pictureOrg，A，Qにはそれぞれ画像配列，変換行列，量子化テーブルが保存されているものとする．

```
% ブロック処理
nRows = size(pictureOrg,1);
nCols = size(pictureOrg,2);
for iRow = 1:2:nRows
  for iCol = 1:2:nCols
    % ブロック抽出
    X = pictureOrg(iRow:iRow+1,...
```

```
        iCol:iCol+1);
      Y = A*X*A.';     % 順変換
      S = round(Y./Q); % 量子化
      Y = Q.*S;        % 逆量子化
      X = A.'*Y*A;     % 逆変換
      % ブロック配置
      pictureRec(iRow:iRow+1,...
                 iCol:iCol+1) = X;
    end
  end
```

図7.7(c)に再構成の結果pictureRecを示す．垂直水平の高周波成分を含む右下ブロックの画素値が平均値に置き換えられている様子が分かる．

Image Processing Toolboxがある場合は，blkproc関数を用いて

```
% ブロック処理
fun = inline(...
  'A.''*(Q.*(round((A*X*A.'')./Q)))*A',...
  'X','A','Q');
blkSize = [2 2];
pictureRec = ...
  blkproc(pictureOrg,blkSize,fun,A,Q);
```

としても，同じ処理を実行できる．

実習7.4　**変換符号化**

M-file：practice07_4.m

例題7.7の手続きを，量子化テーブルを変えながら実際の画像でも試してみよう．また，原画像と再構成画像を比較してみよう．

7.3　離散コサイン変換(DCT)

いよいよ，直交変換の代表例として離散コサイン変換(DCT)について解説しよう．DCTの特徴は以下のようにまとめられる．

- 変換核がコサイン関数で与えられる

図7.8
4×4画像配列

6	5	3	2
6	5	3	2
2	3	5	6
2	3	5	6

- 基底ベクトルが対称性をもつ
- フーリエ解析と密接な関係がある
- 準最適変換である
- 高速アルゴリズムが存在する

● 定義

1次元M点DCTの変換行列$\mathbf{C}_M$は

$$[\mathbf{C}_M]_{k,n} = \sqrt{\frac{2}{M}}\, c_k \cos\frac{k\left(n+\frac{1}{2}\right)\pi}{M} \qquad k,n = 0,1,\cdots,M-1 \quad \cdots\cdots (7.19)$$

と定義される．ただし，

$$c_k = \begin{cases} \dfrac{1}{\sqrt{2}} & k = 0 \\ 1 & k = 1,2,\cdots,M-1 \end{cases}$$

であり，$[\cdot]_{k,n}$はk行n列目の要素を意味する．直交行列であるため，逆行列$\mathbf{C}_M^{-1}$は転置行列$\mathbf{C}_M^T$で与えられる．なお，2次元DCTは分離処理される．

例題7.8　2次元離散コサイン変換

図7.8に示す4×4の画像配列に対して4×4の2次元DCTを施し，変換係数を求めてみよう．

解

以下に，MATLABによる処理例を示す．なお，xには**図7.8**に示す画像配列が保存されているものとする．

```
nPoints = 4; % DCT 点数
% DCT 行列の計算
[n,k] = meshgrid(0:nPoints-1,0:nPoints-1);
scaling = sqrt(2/nPoints)*...
  diag([1/sqrt(2) ones(1,nPoints-1)]);
C = scaling * cos((k.*(n+0.5)*pi)/nPoints);
% 2次元DCT
Y = C*X*C.'
```


結果として，次の変換係数が得られる．

```
Y =
      16.0000         0         0         0
            0    5.8284         0   -0.4142
            0         0         0         0
            0   -2.4142         0    0.1716
```

なお，2次元DCTの順変換と逆変換(IDCT)は，dct関数，idct関数を利用して

```
Y = dct(dct(X.').');
X = idct(idct(Y.').');
```

もしくは，Image Processing Toolboxのdct2関数，idct2関数を利用して，

```
Y = dct2(X);
X = idct2(Y);
```

のようにも計算できる．

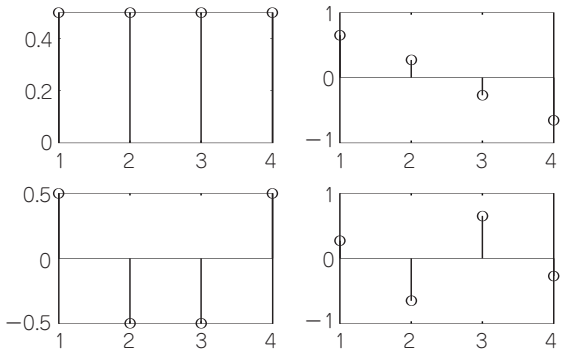
例題7.9　基底ベクトルと基底画像

4点1次元DCTの基底ベクトルと4×4の2次元DCTの基底画像を確認してみよう．

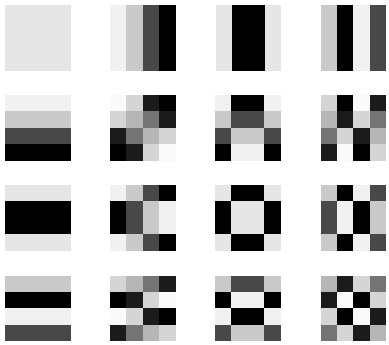
解

以下に，MATLABによる処理例を示す．ただし，Cには4点DCT行列が保存されているものとする．

```
% 基底ベクトルの表示
nRows = size(C,1);
figure(1);
for iRow = 1:nRows
  subplot(ceil(nRows/2),2,iRow);
  stem(C(iRow,:));
end
% 基底画像の表示
figure(2);
adj = max(abs(C(:)))^2; % 輝度調整用
for iRowV=1:nRows
```



（a）基底ベクトル



（b）基底画像

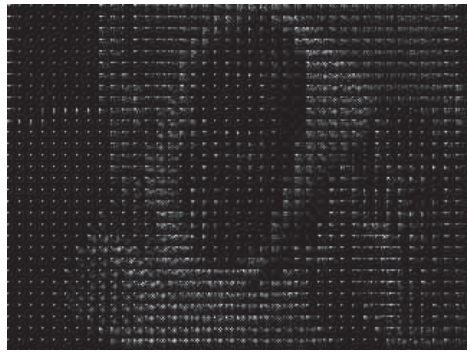
図7.9　4点1次元DCTの基底ベクトルと4×4 2次元DCTの基底画像

```
    for iRowH=1:nRows
      % 基底画像の計算
      basisVecV = C(iRowV,:).';
      basisVecH = C(iRowH,:).';
      B=kron(basisVecV,basisVecH.');
      % 表示のための輝度調整
      B=uint8(127*(B/adj+0.5));
      subplot(nRows,nRows,...
        nRows*(iRowV-1)+iRowH);
      imshow(B); % もしくは，imshowcq(B);
    end
  end
```

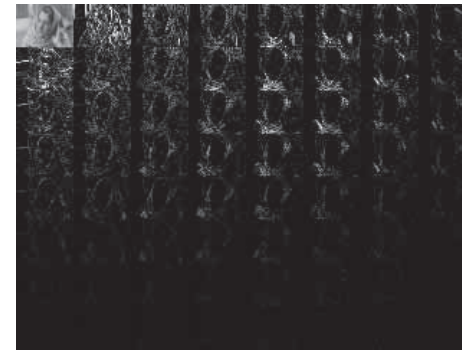
図7.9に基底ベクトルと基底画像を示す．

図7.9より，DCTは異なる波の成分をもつ基底から構成されていることが分かる．左上が直流(DC)成分，そのほかは交流(AC)成分に対応する．

また，**図7.2**(**a**)を重複のない8×8画素のブロックに分割し，それぞれに対して8×8の2次元DCT(ブロックDCT)を施した結果を**図7.10**に示す．直流(DC)係数にエネルギが集中していることが分かる．また，スカーフの細かい模様の部分では交流(AC)係数も大きな値をもつことが分かる．なお，Image Processing Toolboxがある場合，`dctdemo`コマンドによって2次元DCTによる簡単な画像圧縮のデモンストレーションを見ることができる．



（a）ブロックごと



（b）係数ごと

図7.10　8×8ブロックDCTの処理例

実習7.5　**基底ベクトルと基底画像**

M-file：practice07_5.m

$M \neq 4$，特に$M = 8$の場合についても，基底ベクトルと基底画像を確認してみよう．

● DFTとの関係

DCTが信号処理の世界で広く利用されている理由の一つにDFTとの密接な関係がある．周波数積と畳み込み演算の関係が，ある条件下において，DCTでも成立することが証明されている．また，別の理由としてカルーネン・レーベ変換（KLT）との密接な関係がある．これらの関係について解説しよう．

(1) 一般化DFTの定義

まず，一般化DFT（GDFT：Generalized DFT）の紹介から始めよう．信号$x[n]$のM点GDFTは，

$$X_M^{(a,b)}[k] = \sum_{n=0}^{M-1} x[n] W_M^{(k+a)(n+b)} \qquad k = 0, 1, \cdots, M-1 \quad \cdots\cdots (7.20)$$

と定義される．ただし，$W_M = e^{-j\frac{2\pi}{M}}$であり，$a$と$b$は任意の実数である．GDFTはDFTと同じようにユニタリ変換である．$a = b = 0$の場合，すなわち，$X_M^{(0,0)}[k]$はDFTに帰着する．a，bは，それぞれ周波数サンプル点，時間サンプル点のシフト量を意味する．

$a = 0$，$b = 1/2$と選択したGDFTはDCTと深い関係にある．このGDFTは特にOdd-time DFT（OTDFT）と呼ばれる．

M点DFTの変換行列を$\mathbf{W}_M^{(0,0)}$とし，M点OTDFTの変換行列を$\mathbf{W}_M^{\left(0,\frac{1}{2}\right)}$と表現すると，

$$\mathbf{W}_M^{\left(0,\frac{1}{2}\right)} = \mathbf{\Lambda}_M \mathbf{W}_M^{(0,0)} \quad \cdots\cdots (7.21)$$

という関係が成立する．ただし，$\mathbf{\Lambda}_M$は$M \times M$対角行列であり，そのk行k列目要素は$[\Lambda_M]_{k,k} = W_M^{\frac{k}{2}} = W_{2M}^{k} = e^{-j\frac{\pi}{M}k}$と与えられる．

例題7.10　**OTDFT**

8点OTDFTがユニタリ変換であることを確かめてみよう．

解

$\left(\mathbf{W}_M^{\left(0,\frac{1}{2}\right)}\right)^{\dagger}\cdot\mathbf{W}_M^{\left(0,\frac{1}{2}\right)}=c\mathbf{I}$ が証明できればよい．数学的な証明については読者自身にお任せすることにして，以下ではMATLAB上における確認の例を示そう．

```
nPoints = 8; % 変換点数
Wdft = dftmtx(nPoints); % DFT 行列
k=0:nPoints-1;
Lambda = diag(exp(-j*pi/nPoints*k));
Wotdft = Lambda * Wdft; % OTDFT 行列
Wotdft' * Wotdft % ユニタリ性の確認
```

結果は$c=8$倍の単位行列となる．

(2) OTDFTとDCT

いま，M点の信号$x[n]$から，$2M$点の対称な実信号

$$u[n]=\begin{cases}x[n] & 0,1,\cdots,M-1\\ x[2M-1-n] & M,M+1,\cdots,2M-1\end{cases}\qquad\cdots\cdots(7.22)$$

を定義し，$u[n]$の$2M$点OTDFTを考えよう．

$$X_{2M}^{\left(0,\frac{1}{2}\right)}[k]=\sum_{n=0}^{2M-1}u[n]W_{2M}^{k\left(n+\frac{1}{2}\right)}=\sum_{n=0}^{M-1}x[n]\left(W_{2M}^{k\left(n+\frac{1}{2}\right)}+W_{2M}^{-k\left(n+\frac{1}{2}\right)}\right)$$

$$=2\sum_{n=0}^{M-1}x[n]\cos\frac{k\left(n+\frac{1}{2}\right)\pi}{M}\qquad k=0,1,\cdots,2M-1\qquad\cdots\cdots(7.23)$$

変換係数$X_{2M}^{\left(0,\frac{1}{2}\right)}[k]$はcos関数の性質により対称となり，独立な係数の数は$k=0$，1，…，$M-1$のM点となる．すなわち，計算する変換係数は半分でよい．実は，正規化のためのスケーリングを除けば，式(7.23)の変換はDCTそのものであり，式(7.19)は，

$$[\mathbf{C}_M]_{k,n}=\frac{c_k}{\sqrt{2M}}\left[\mathbf{W}_{2M}^{\left(0,\frac{1}{2}\right)}\mathbf{E}_M\right]_{k,n}\qquad k,n=0,1,\cdots,M-1\qquad\cdots\cdots(7.24)$$

と等価である．ここで，$\mathbf{E}_M$は，$2M\times M$の対称拡張行列

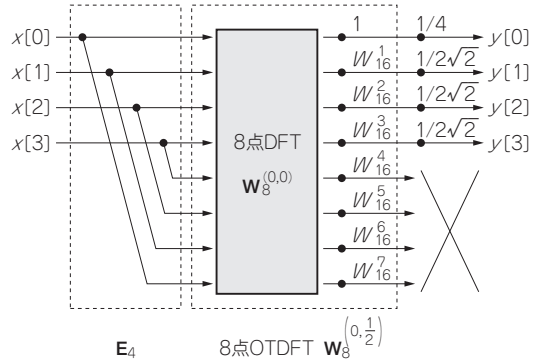


図7.11　8点OTDFTによる4点DCTの計算

$$\mathbf{E}_M = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \end{pmatrix}$$

である．

例題7.11　**OTDFTとDCT**

8点OTDFTにより，4点DCTが与えられることを確かめてみよう．

解

図7.11に処理フローを示し，以下にMATLAB上におけるコマンド例を示す．ただし，8点OTDFT行列が`Wotdft`に保存されていると仮定する（例題7.10を参照）．

```
nPoints = 4;
E = [ eye(nPoints) ; ...
      fliplr(eye(nPoints)) ];
C = 1/sqrt(2*nPoints)*...
      diag([1/sqrt(2) ones(1,nPoints-1)])*...
      Wotdft(1:nPoints,:)*E;
```

Cは`dct(eye(4))`と同じ結果を与える．

実習7.6　**OTDFTとDCT**

M-file：practice07_6.m

$M \neq 4$，特に$M = 8$の場合についても式(7.24)の関係を確かめてみよう．

● KLTとの関係

KLTは，それそのものが信号に依存するため，符号化などへの実用例はDCTに比べるとあまり見られない．しかし，定常確率過程に対して無相関化，エネルギ寄与率最大化，符号化利得最大化などの点で最適な変換である．

まずは，KLT行列の定義を示そう．

(1) KLTの定義

以下では，実数の定常確率過程の連続するM点信号からなるM次元ベクトルを$\mathbf{x}$とする．また，$\mathbf{R}_{xx}$を$\mathbf{x}$の共分散行列$\mathbf{R}_{xx} = E\{\mathbf{x}\mathbf{x}^T\} - E\{\mathbf{x}\}E\{\mathbf{x}^T\}$と定義する．ただし，$E\{\cdot\}$は期待値である．KLT行列はベクトル$\mathbf{x}$に依存し，その$k$番目の基底ベクトルは，

$$\mathbf{R}_{xx}\phi_k = \lambda_k \phi_k \quad \cdots\cdots (7.25)$$

を満たす$\mathbf{R}_{xx}$の正規化固有ベクトルϕ_kとして与えられる．なお，λ_kは$\mathbf{R}_{xx}$の固有値である．結果としてKLT行列$\mathbf{\Phi}$は，

$$\Phi = \left(\phi_0, \phi_1, \cdots, \phi_{M-1}\right)^T \quad \cdots\cdots (7.26)$$

と与えられる．通常，$\lambda_0 > \lambda_1 > \cdots > \lambda_{M-1}$となるように並べられる．共分散行列$\mathbf{R}_{xx}$が実対称行列であるため，行列$\mathbf{\Phi}$は正規直交行列として与えられる．

例題7.12　**KLT行列**

図7.3(a)の画像について，水平方向に画素のペアを組んで構成した2次元ベクトル集合の共分散行列$\mathbf{R}_{xx}$を求め，KLT行列を求めてみよう．

解

以下に，MATLABのコマンド例を示す．ただし，`setOfX`には，例題7.2と同じ，水平方向の画素ペアで構成された2次元ベクトルの集合が保存されていると仮定する．

```
Rxx = cov(double(setOfX.')); % 共分散行列
[V,D] = eig(Rxx);        % 固有値分解
[Y,I] = sort(diag(D)); % 固有値のソート
% 固有ベクトルを並び換えと転置
Phi = V(:,nPoints-I+1).';
```

KLT行列が`Phi`として与えられ，基底ベクトルの符号の違いを除けば，式(7.1)の変換行列と類似した行列となる．垂直方向の画素ペアやそのほかの画像について確認してみよう．

KLTによって与えられる係数ベクトルを$\mathbf{y}=\mathbf{\Phi x}$とし，この共分散行列を$\mathbf{R}_{yy}$とすると，

$$\mathbf{R}_{yy}=\mathbf{\Phi R}_{xx}\mathbf{\Phi}^{T} \quad \cdots\cdots (7.27)$$

と表現される．$\mathbf{\Phi}$が$\mathbf{R}_{xx}$の固有ベクトルから構成されるため，$\mathbf{R}_{yy}$は固有値を対角要素としてもつ対角行列にほかならない．KLTは共分散行列の非対角要素が0，すなわち係数間に相関のない（無相関な）信号へ変換することが分かる．

実習7.7　KLT行列

M-file：practice07_7.m

$M\neq 2$，特に$M=8$の場合についてKLT行列を求めてみよう．

(2) 1次自己回帰過程モデル

隣り合うサンプル間の相関の強い信号は1次自己回帰（AR(1)）過程としてしばしばモデル化される．AR(1)過程は期待値ゼロのホワイト・ノイズ（白色雑音）$w[n]$と相関係数ρにより，

$$x[n]=w[n]+\rho x[n-1] \quad \cdots\cdots (7.28)$$

のように定義される．ただし，$|\rho|<1$である．相関係数ρが1に近いほど，隣り合うサンプル間で似た値をとる．自然画像では，水平垂直方向共に相関係数ρは0.95程度といわれている．

AR(1)過程のM次元ベクトル

$$\mathbf{x}=(x[0],x[1],\cdots,x[M-1])^{T}$$

の共分散行列$\mathbf{R}_{xx}$は，

$$\mathbf{R}_{xx}=\begin{pmatrix} 1 & \rho & \rho^{2} & \cdots & \rho^{M-1} \\ \rho & 1 & \rho & \cdots & \rho^{M-2} \\ \rho^{2} & \rho & 1 & \cdots & \rho^{M-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{M-1} & \rho^{M-2} & \rho^{M-3} & \cdots & 1 \end{pmatrix} \quad \cdots\cdots (7.29)$$

のようにToeplitz行列として与えられる．

(3) KLTとDCT

AR(1)過程に対してKLTを求め，$\rho\to 1$という極限操作を行うと，KLT行列はDCT行列に収束することが知られている．このことから，ρが1に近いAR(1)過程でモデル化できる自然画像に対して，DCTは統計的に準最適な変換であると裏付けられる．

例題7.13　KLTとDCT

$\rho=0.95$のAR(1)過程の信号に対する4点KLTの変換行列を求めDCT行列と比較してみよう．

解

以下に，MATLAB上でのコマンドの例を示す．


```
nPoints = 4; % ベクトルの次元
```

```
rho = 0.95;  % 相関係数  |ρ|<1
% AR(1) 過程モデルの共分散行列
Rxx = toeplitz(power(rho,0:nPoints-1));
% 固有ベクトルと固有値の計算
[V,D] = eig(Rxx);
% 固有値のソート
[Y,I] = sort(diag(D));
% KLT 行列
Phi = V(:,nPoints-I+1).'
% DCT 行列
C = dct(eye(nPoints))
```

基底ベクトルの符号の違いを除けば，KLTとDCTは類似した結果となる．

実習7.8　**KLTとDCT**

M-file：practice07_8.m

$M \neq 4$，特に$M = 8$の場合についても比較してみよう．ρの値も変えながら確認してみよう．

● 実現技術

DFTとの関係，KLTとの関係に加えて，DCTの普及に欠かせない特徴が，高速アルゴリズムの存在である．以下では，スパース行列分解による高速アルゴリズムの一例を紹介し，さらに，H.264/MPEG-4 AVCで採用されている整数精度変換を紹介しよう．

(1) 高速アルゴリズム

M点DCTを行列演算により実現するとM^2回の乗算と$M(M-1)$の加算を要する．一方で，DFTとの関係を利用して高速フーリエ変換(FFT)により演算回数を減らす方法がある．FFT法は既存ライブラリなどが使える利点はあるが，複素演算を伴うため，Mが小さいときはあまり効果を期待できない．以下では，複素演算を用いずに高速化するWangのアルゴリズムを紹介しよう．

準備として，タイプI～IVの4種類のDCT行列を紹介しよう．なお，これまで本書で扱ってきたDCTはタイプIIそのものである．

$$[\mathbf{C}^{\mathrm{I}}_{M+1}]_{k,n} = \sqrt{\frac{2}{M}}\, c_k c_n \cos\frac{kn\pi}{M} \qquad k,n = 0,1,\cdots,M \quad \cdots\cdots (7.30a)$$

$$[\mathbf{C}^{\mathrm{II}}_{M}]_{k,n} = \sqrt{\frac{2}{M}}\, c_k \cos\frac{k\left(n+\frac{1}{2}\right)\pi}{M} \qquad k,n = 0,1,\cdots,M-1 \quad \cdots\cdots (7.30b)$$

$$[\mathbf{C}_M^{\mathrm{III}}]_{k,n} = \sqrt{\frac{2}{M}}\, c_n \cos\frac{\left(k+\frac{1}{2}\right)n\pi}{M} \qquad k,n = 0,1,\cdots,M-1 \quad \cdots\cdots (7.30c)$$

$$[\mathbf{C}_M^{\mathrm{IV}}]_{k,n} = \sqrt{\frac{2}{M}} \cos\frac{\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}\right)\pi}{M} \qquad k,n = 0,1,\cdots,M-1 \quad \cdots\cdots (7.30d)$$

ただし，

$$c_k = \begin{cases} \dfrac{1}{\sqrt{2}} & k = 0, M \\ 1 & k = 1,2,\cdots,M-1 \end{cases}$$

Wangはこれら4種類のDCTの行列分解を

$$\mathbf{C}_{M+1}^{\mathrm{I}} = \frac{1}{\sqrt{2}}\mathbf{B}_{M+1}^{T}\begin{pmatrix} \mathbf{C}_{\frac{M}{2}+1}^{\mathrm{I}} & \mathbf{O} \\ \mathbf{O} & \mathbf{C}_{\frac{M}{2}}^{\mathrm{III}}\mathbf{J}_{\frac{M}{2}} \end{pmatrix}\begin{pmatrix} \mathbf{I}_{\frac{M}{2}} & \mathbf{o} & \mathbf{J}_{\frac{M}{2}} \\ \mathbf{o} & \sqrt{2} & \mathbf{o} \\ \mathbf{J}_{\frac{M}{2}} & \mathbf{o} & -\mathbf{I}_{\frac{M}{2}} \end{pmatrix} \quad \cdots\cdots (7.31a)$$

$$\mathbf{C}_M^{\mathrm{II}} = \frac{1}{\sqrt{2}}\mathbf{B}_M^{T}\begin{pmatrix} \mathbf{C}_{\frac{M}{2}}^{\mathrm{II}} & \mathbf{O} \\ \mathbf{O} & \mathbf{C}_{\frac{M}{2}}^{\mathrm{IV}}\mathbf{J}_{\frac{M}{2}} \end{pmatrix}\begin{pmatrix} \mathbf{I}_{\frac{M}{2}} & \mathbf{J}_{\frac{M}{2}} \\ \mathbf{J}_{\frac{M}{2}} & -\mathbf{I}_{\frac{M}{2}} \end{pmatrix} \quad \cdots\cdots (7.31b)$$

$$\mathbf{C}_M^{\mathrm{III}} = \frac{1}{\sqrt{2}}\begin{pmatrix} \mathbf{I}_{\frac{M}{2}} & \mathbf{J}_{\frac{M}{2}} \\ \mathbf{J}_{\frac{M}{2}} & -\mathbf{I}_{\frac{M}{2}} \end{pmatrix}\begin{pmatrix} \mathbf{C}_{\frac{M}{2}}^{\mathrm{III}} & \mathbf{O} \\ \mathbf{O} & \mathbf{J}_{\frac{M}{2}}\mathbf{C}_{\frac{M}{2}}^{\mathrm{IV}} \end{pmatrix}\mathbf{B}_M \quad \cdots\cdots (7.31c)$$

$$\mathbf{C}_M^{\mathrm{IV}} = \frac{1}{\sqrt{2}}\mathbf{T}_M\begin{pmatrix} \mathbf{C}_{\frac{M}{2}}^{\mathrm{III}} & \mathbf{O} \\ \mathbf{O} & \boldsymbol{\Gamma}_{\frac{M}{2}}\mathbf{C}_{\frac{M}{2}}^{\mathrm{III}} \end{pmatrix}\mathbf{B}_M\mathbf{V}_M \quad \cdots\cdots (7.31d)$$

と与えている．ただし，Mを偶数とする．Wangの分解法では，すべてのタイプのDCT行列がサイズが半分のDCTによって表現されるため，Mが2のべき乗のときに分解を繰り返すことができる．

行列$\mathbf{I}_M$，$\mathbf{J}_M$，$\mathbf{B}_M$は，それぞれ$M \times M$の単位行列，左(右)から掛けて行(列)を反転する反転行列，左から掛けて偶数行と奇数行を上下に分ける置換行列である．また，行列$\boldsymbol{\Gamma}_M$は，対角要素が$[\boldsymbol{\Gamma}_M]_{k,k} = (-1)^k$と定義される$M \times M$の対角行列である．$\mathbf{I}_4$，$\mathbf{J}_4$，$\mathbf{B}_4$，$\boldsymbol{\Gamma}_4$，また，$\mathbf{T}_4$，$\mathbf{V}_4$の具体例を以下に示そう．

$$\mathbf{I}_4 = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix},\ \mathbf{J}_4 = \begin{pmatrix} 0&0&0&1 \\ 0&0&1&0 \\ 0&1&0&0 \\ 1&0&0&0 \end{pmatrix},\ \mathbf{B}_4 = \begin{pmatrix} 1&0&0&0 \\ 0&0&1&0 \\ 0&1&0&0 \\ 0&0&0&1 \end{pmatrix},\ \Gamma_4 = \begin{pmatrix} 1&0&0&0 \\ 0&-1&0&0 \\ 0&0&1&0 \\ 0&0&0&-1 \end{pmatrix},$$

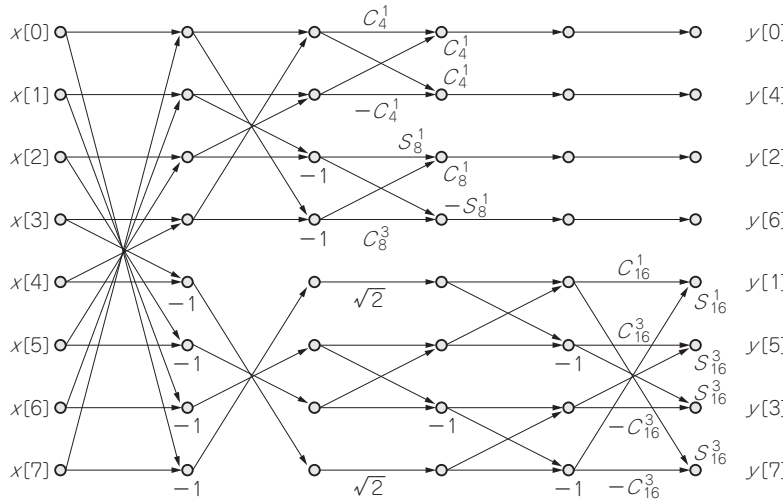


図7.12　Wangの高速DCT（タイプII，$M = 8$）

$$\mathbf{T}_4 = \begin{pmatrix} C_{16}^1 & 0 & 0 & S_{16}^1 \\ 0 & C_{16}^3 & S_{16}^3 & 0 \\ 0 & S_{16}^3 & -C_{16}^3 & 0 \\ S_{16}^1 & 0 & 0 & -C_{16}^1 \end{pmatrix}, \quad \mathbf{V}_4 = \begin{pmatrix} \sqrt{2} & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & \sqrt{2} \end{pmatrix}$$

ただし，$C_\ell^i = \cos\dfrac{i\pi}{\ell}$，$S_\ell^i = \sin\dfrac{i\pi}{\ell}$である．

図7.12にWangによる8点DCT（タイプII）の実現法を示す．なお，$1/\sqrt{2}$によるスケーリングは図中では省略している．IDCTの高速実現については，例えば**図7.12**の構成で，流れ図の矢印の向きを右から左へとれば実現できる．また，以下の性質も利用できる．

$$C_{M+1}^{\mathrm{I}\,-1} = C_{M+1}^{\mathrm{I}} \quad \cdots\cdots (7.32a)$$

$$C_M^{\mathrm{II}\,-1} = C_M^{\mathrm{III}} \quad \cdots\cdots (7.32b)$$

$$C_M^{\mathrm{III}\,-1} = C_M^{\mathrm{II}} \quad \cdots\cdots (7.32c)$$

$$C_M^{\mathrm{IV}\,-1} = C_M^{\mathrm{IV}} \quad \cdots\cdots (7.32d)$$

WangのアルゴリズムによるM点DCT（タイプII）の乗算回数$\mu(\mathbf{C}_M^{\mathrm{II}})$と加算回数$\alpha(\mathbf{C}_M^{\mathrm{II}})$を示そう．

$$\mu(C_M^{\mathrm{II}}) = \frac{M}{2}\log_2 M + 1 \quad \cdots\cdots (7.33a)$$

$$\alpha(C_M^{\mathrm{II}}) = \frac{3M}{2}\log_2 M - M + 1 \qquad M \geq 2 \quad \cdots\cdots (7.33b)$$

この演算量は$\mathbf{T}_M$に現れる平面回転を

$$\begin{pmatrix} C_\ell^i & S_\ell^i \\ S_\ell^i & -C_\ell^i \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} C_\ell^i + S_\ell^i & 0 & 0 \\ 0 & -C_\ell^i + S_\ell^i & 0 \\ 0 & 0 & S_\ell^i \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \end{pmatrix} \quad \cdots\cdots (7.34)$$

と変形し，3回の乗算と3回の加算で実現するものと仮定している．$M = 8$のとき，乗算回数は13回，加算回数は29回となり，行列演算の64回，56回に比べて効率的であることが分かる．

例題7.14　Wangのスパース行列分解

$M = 8$の場合について，式(7.31b)の関係を確かめてみよう．

解

数学的な証明については読者自身にお任せすることにして，以下ではMATLABにおけるコマンド例を示そう．

```
I4 = eye(4);      % 4 × 4 単位行列
J4 = fliplr(I4); % 4 × 4 反転行列
O4 = zeros(4);    % 4 × 4 ゼロ値行列
I8 = eye(8);      % 8 × 8 単位行列
B8 = [ I8(1:2:end,:) ;
       I8(2:2:end,:) ]; % 8 × 8 置換行列
CII4 = dct(eye(4));      % 4 × 4DCT-II 行列
CIV4 = dctiv(eye(4));    % 4 × 4DCT-IV 行列
% 8× 8DCT-IV 行列
CII8 = 1/sqrt(2) * B8.' ...
    * [ CII4 O4 ;
        O4 CIV4*J4 ] ...
    * [ I4 J4 ;
        J4 -I4 ]
```

`dct(eye(8))`と同じ結果が与えられる．なお，DCT(タイプIV)を計算するために`dctiv`関数を別に用意した．

Wangのアルゴリズムの実装例として`wadcti`，`wadctii`，`wadctiii`，`wadctiv`の各関数も用意したので内容を参照されたい．

(2) 整数精度DCT

地上デジタルのワンセグ放送で採用されている動画像符号化標準H.264/MPEG-4 AVCでは，動き補償予測後あるいはイントラ予測後の残差データに対して4×4の2次元DCTをベースとした2次元変換を採用している．ただし，例題7.8の2次元DCTと以下の点で異なっている．

- 整数精度変換である
- 逆変換ミスマッチがない
- 主要な計算は加算とシフトのみで実現できる

● スケーリングは量子化に統合され，乗算回数が低減される

配列$\mathbf{X}$の2次元4×4 DCTは，

$$\mathbf{Y}=(\mathbf{CXC}^T)\circledcirc\mathbf{E}=\left(\begin{pmatrix}1&1&1&1\\1&d&-d&-1\\1&-1&-1&1\\d&-1&1&-d\end{pmatrix}\mathbf{X}\begin{pmatrix}1&1&1&d\\1&d&-1&-1\\1&-d&-1&1\\1&-1&1&-d\end{pmatrix}\right)\circledcirc\begin{pmatrix}a^2&ab&a^2&ab\\ab&b^2&ab&b^2\\a^2&ab&a^2&ab\\ab&b^2&ab&b^2\end{pmatrix}\quad\cdots\cdots(7.35)$$

と等価表現できる．ただし，◉は要素ごとの掛け算を意味し，

$$a=\frac{1}{2},\ b=\sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right),\ d=\frac{\cos\left(\frac{3\pi}{8}\right)}{\cos\left(\frac{\pi}{8}\right)}$$

である．H.264/MPEG-4 AVCでは，これを

$$a=\frac{1}{2},\ b=\sqrt{\frac{2}{5}},\ d=\frac{1}{2}$$

と近似し，

$$\mathbf{Y}=(\mathbf{C}_f\mathbf{XC}_f^T)\circledcirc\mathbf{E}_f=\left(\begin{pmatrix}1&1&1&1\\2&1&-1&-2\\1&-1&-1&1\\1&-2&2&-1\end{pmatrix}\mathbf{X}\begin{pmatrix}1&2&1&1\\1&1&-1&-2\\1&-1&-1&2\\1&-2&1&-1\end{pmatrix}\right)\circledcirc\begin{pmatrix}a^2&\frac{ab}{2}&a^2&\frac{ab}{2}\\\frac{ab}{2}&\frac{b^2}{4}&\frac{ab}{2}&\frac{b^2}{4}\\a^2&\frac{ab}{2}&a^2&\frac{ab}{2}\\\frac{ab}{2}&\frac{b^2}{4}&\frac{ab}{2}&\frac{b^2}{4}\end{pmatrix}\quad\cdots\cdots(7.36)$$

という変換を定義している．変換核$\mathbf{C}_f\mathbf{XC}_f^T$は整数の加算とシフトのみで実現され，スケーリング$\mathbf{E}_f$は量子化に含められる．

例題7.15　整数精度変換

図7.8に示す4×4の画像配列に対して4×4の整数精度変換を施し，変換係数を求めてみよう．

解

以下に，MATLABによる処理例を示す．なお，xには**図7.8**に示す画像配列が保存されているものとする．

```
Cf = [ 1  1  1  1 ;
       2  1 -1 -2 ;
       1 -1 -1  1 ;
       1 -2  2 -1 ];
a = 1/2; b = sqrt(2/5);
e00 = a^2; e01 = a*b/2; e11 = b^2/4;
Ef = [ e00 e01 e00 e01 ;
```

```
        e01 e11 e01 e11 ;
        e00 e01 e00 e01 ;
        e01 e11 e01 e11 ];
% 2次元DCT
Y = (Cf*X*Cf.').*Ef
```

以下のように，例題7.8の結果と若干異なる結果が与えられる．

```
Y =
      7.0000   1.2649  -1.0000   0.6325
      1.2649  -0.2000  -1.2649   1.4000
     -1.0000  -1.2649   1.0000  -0.6325
      0.6325   1.4000  -0.6325   0.2000
```

なお，逆変換は，

$$\mathbf{X} = \mathbf{C}_i^T(\mathbf{Y} \circledcirc \mathbf{E}_i)\mathbf{C}_i = \begin{pmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{pmatrix} \left(\mathbf{Y} \circledcirc \begin{pmatrix} a^2 & ab & a & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{pmatrix} \right) \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{pmatrix} \quad \cdots\cdots\cdots\cdots (7.37)$$

と与えられる．

実習7.9　整数精度変換

M-file：practice07_9.m

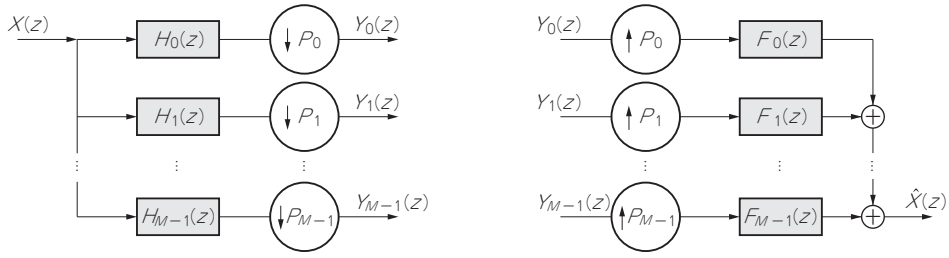
例題7.15で得られた係数$\mathbf{Y}$を逆変換し，元の配列$\mathbf{X}$が再構成されることを確かめてみよう．また，異なる配列$\mathbf{X}$に対しても確かめてみよう．

7.4　フィルタ・バンク

フィルタ・バンクは，情報圧縮や信号解析，適応信号処理など，広い分野で利用される要素技術である．以下では，信号変換行列とフィルタ・バンクの関係について説明する．DCTが，ある特殊なフィルタ・バンクであることも示そう．また，フィルタ・バンクは**離散時間ウェーブレット変換**(DWT)においても重要な役割を果たすことを解説しよう．

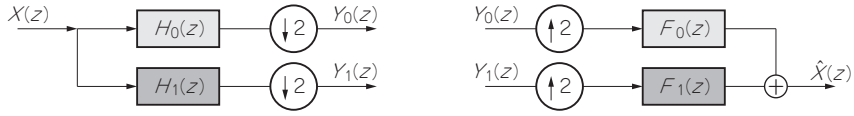
● 並列構成

フィルタ・バンクは，**図7.13**に示されるように分析器と合成器から構成される．分析器は複数のデシメータ，合成器は複数のインターポレータから構成される．分析器から入力された信号$X(z)$は，



（a）分析器　　（b）合成器

図7.13　一般的なフィルタ・バンク



（a）分析器　　（b）合成器

図7.14　2分割フィルタ・バンク

分析器で複数の帯域に分割され，何らかの処理が施される．圧縮などの応用では，量子化やエントロピ符号化が行われる．分割された信号$Y_k(z)$は**サブバンド信号**と呼ばれる．Mは分割数を，P_kはkチャネルの間引き率を表している．

(1) 完全再構成フィルタ・バンク

分析器と合成器は，完全再構成条件

$$\hat{X}(z) = cz^{-d}X(z) \quad \cdots\cdots (7.38)$$

を満たすように設計できる．ここで，cは任意の複素定数，dは任意の整数である．この条件は，復元信号$\hat{X}(z)$が，スケーリングと遅延を除いて原信号$X(z)$と一致することを意味している．このような性質をもつフィルタ・バンクを特に，**完全再構成フィルタ・バンク**と呼ぶ．

(2) 2分割フィルタ・バンク

図7.14は，$M = P_0 = P_1 = 2$という構成の2分割フィルタ・バンクを示している．通常，$H_0(z)$，$F_0(z)$は低域通過フィルタとして，$H_1(z)$，$F_1(z)$は高域通過フィルタとして設計される．後述するように，この構成のフィルタ・バンクは離散時間ウェーブレット変換において重要な役割を果たす．

図7.14の構成の入出力関係は，ダウン・サンプラ，アップ・サンプラ，フィルタの入出力関係より，

$$\hat{X}(z) = T(z)X(z) + A(z)X(-z) \quad \cdots\cdots (7.39)$$

と表現できる．ただし，

$$T(z) = \frac{1}{2}\{H_0(z)F_0(z) + H_1(z)F_1(z)\}$$

$$A(z) = \frac{1}{2}\{H_0(-z)F_0(z) + H_1(-z)F_1(z)\}$$

である．$A(z)$はエリアジング成分$X(-z)$にかかる伝達関数であり，完全再構成を保証するためには

$$A(z) = 0 \quad \cdots\cdots (7.40)$$

でなければならない．さらに，この条件の下で

$$T(z) = cz^{-d} \quad \cdots\cdots (7.41)$$

となれば，このときのみ式(7.38)が満たされる．式(7.40)，(7.41)は，式(7.38)の必要十分条件となっている．

例題7.16　完全再構成フィルタ・バンク

次のフィルタ・バンク($M = P_0 = P_1 = 2$)が完全再構成条件を満たすことを確認してみよう．また，$H_0(z)$，$H_1(z)$の周波数特性も確かめてみよう．

$$H_0(z) = \frac{1}{2}(1 + z^{-1}),\ H_1(z) = \frac{1}{2}(1 - z^{-1})$$

$$F_0(z) = z^{-1}2H_0(z^{-1}),\ F_1(z) = z^{-1}2H_1(z^{-1})$$

解

$A(z) = 1$，$T(z) = z^{-1}$となることが確認できる．数式上での確認は読者自身にお任せすることにして，以下ではMATLABにおける確認の例を示そう．

```
% 分析フィルタ
h0 = [ 1  1 ]/2;
h1 = [ 1 -1 ]/2;
% 合成フィルタ
f0 = 2*fliplr(h0); % 時間反転して2 倍
f1 = 2*fliplr(h1); % 時間反転して2 倍
% A(z)
h0m = h0 * diag([ 1 -1 ]); % H0(-z)
h1m = h1 * diag([ 1 -1 ]); % H1(-z)
A = ( conv(h0m,f0) + conv(h1m,f1) )/2
A =
  0 0 0
% T(z)
T = ( conv(h0,f0) + conv(h1,f1) )/2
```

次の結果が得られる．

```
T =
  0 1.0000 0
```


$H_0(z)$，$H_1(z)$の周波数振幅特性を**図7.15**に示す．

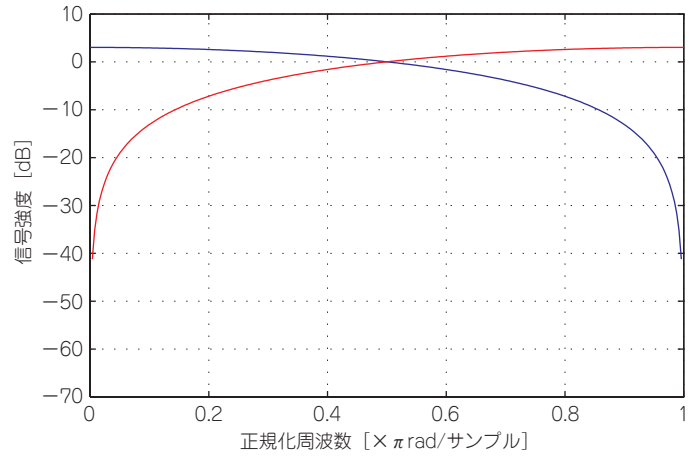


図7.15　2分割フィルタ・バンクの周波数振幅特性の例

実習7.10　**完全再構成フィルタ・バンク**

M-file：practice07_10.m

次のフィルタ・バンク（$M = P_0 = P_1 = 2$）が完全再構成条件を満たすことを確認してみよう．また，$H_0(z)$，$H_1(z)$の周波数振幅特性も確かめてみよう．

$$H_0(z) = \frac{1}{8}\left\{(1+\sqrt{3}) + (3+\sqrt{3})z^{-1} + (3-\sqrt{3})z^{-2} + (1-\sqrt{3})z^{-3}\right\}$$

$$H_1(z) = \frac{1}{8}\left\{(1-\sqrt{3}) - (3-\sqrt{3})z^{-1} + (3+\sqrt{3})z^{-2} - (1+\sqrt{3})z^{-3}\right\}$$

$$F_0(z) = z^{-3}2H_0(z^{-1})$$

$$F_1(z) = z^{-3}2H_1(z^{-1})$$

● 行列演算との関係

フィルタ・バンクは，信号を変換し，再び復元する機能をもつ．これは行列演算による順変換と逆変換に対応する．

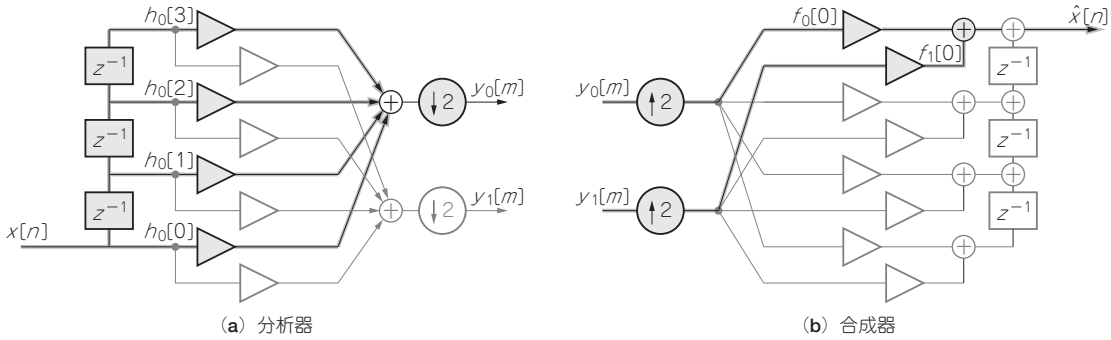
いま，議論を簡単にするために，**図7.14**に示される最大間引き2等分割フィルタ・バンクを考え，各フィルタの伝達関数を，$k = 0$，1に対して，

$$H_k(z) = h_k[0] + h_k[1]z^{-1} + h_k[2]z^{-2} + h_k[3]z^{-3} \quad \cdots\cdots (7.42)$$

$$F_k(z) = f_k[0] + f_k[1]z^{-1} + f_k[2]z^{-2} + f_k[3]z^{-3} \quad \cdots\cdots (7.43)$$

とおく．このとき**図7.14**の分析器と合成器はそれぞれ**図7.16**（**a**），（**b**）のように表現でき，全体は**図7.17**のように等価表現できる．

まず，分析器に着目しよう．これは，2×4行列による行列演算を繰り返すことにほかならない．



（a）分析器　　（b）合成器

図7.16　2分割フィルタ・バンクでの内積演算

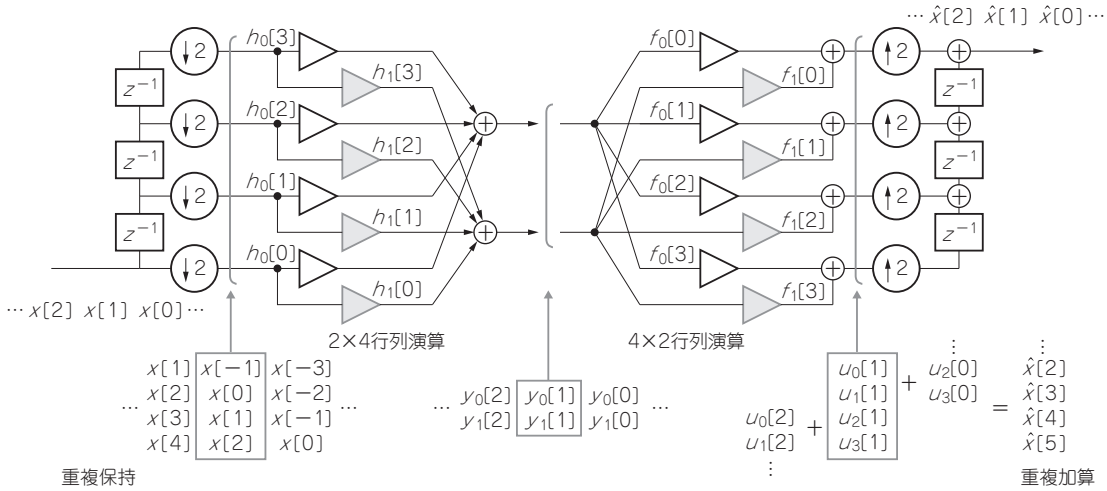


図7.17　2分割フィルタ・バンクと行列変換

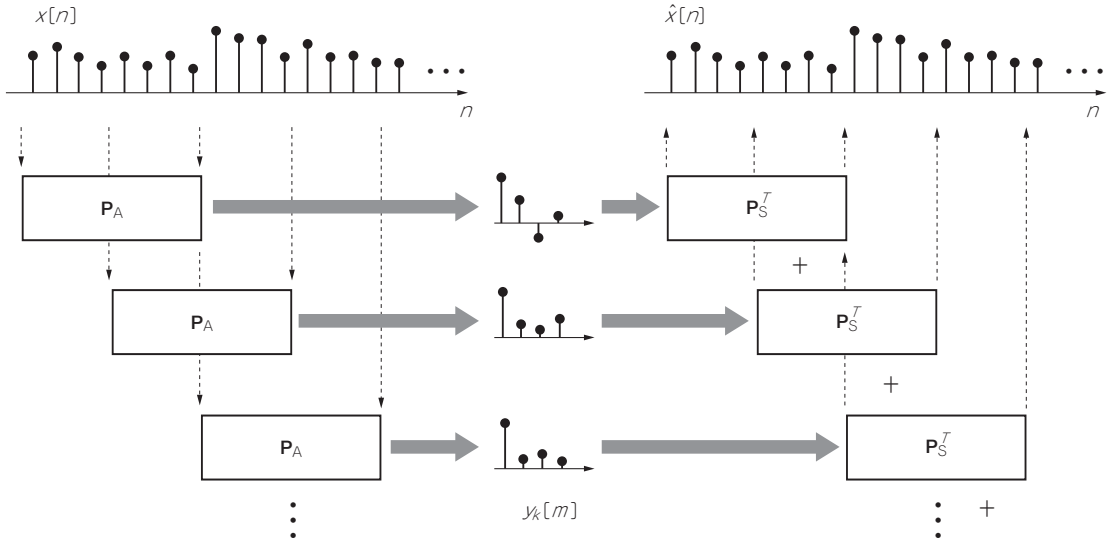
$$\begin{pmatrix} y_0[m] \\ y_1[m] \end{pmatrix} = \underbrace{\begin{pmatrix} h_0[3] & h_0[2] & h_0[1] & h_0[0] \\ h_1[3] & h_1[2] & h_1[1] & h_1[0] \end{pmatrix}}_{\mathbf{P}_A} \begin{pmatrix} x[2m-3] \\ x[2m-2] \\ x[2m-1] \\ x[2m] \end{pmatrix} \quad \cdots\cdots (7.44)$$

一方，合成器は，

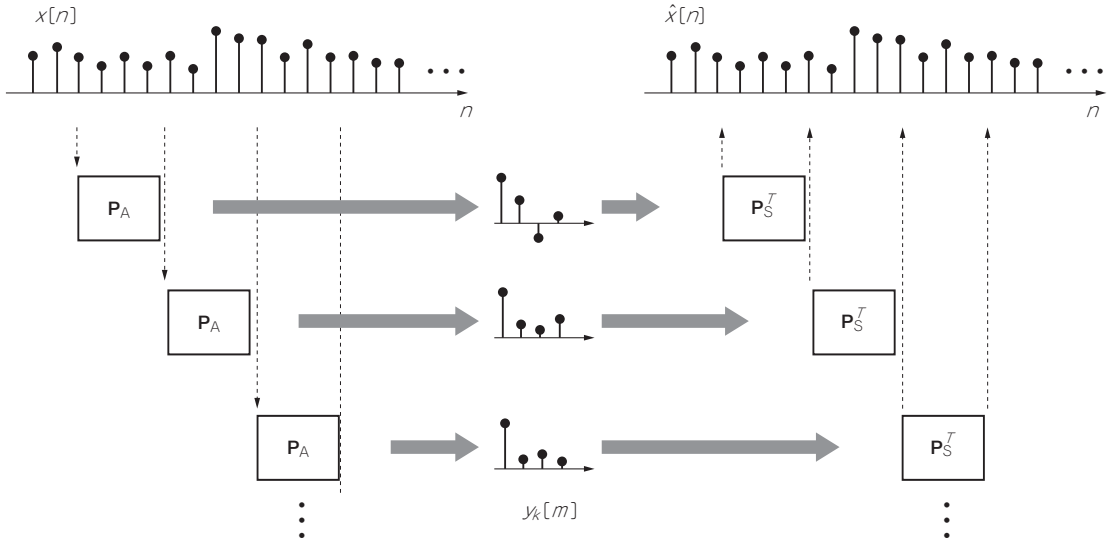
$$\begin{pmatrix} u_0[m] \\ u_1[m] \\ u_2[m] \\ u_3[m] \end{pmatrix} = \underbrace{\begin{pmatrix} f_0[0] & f_1[0] \\ f_0[1] & f_1[1] \\ f_0[2] & f_1[2] \\ f_0[3] & f_1[3] \end{pmatrix}}_{\mathbf{P}_S^T} \begin{pmatrix} y_0[m] \\ y_1[m] \end{pmatrix} \quad \cdots\cdots (7.45)$$

4×2行列による行列演算を繰り返すことにほかならない．最終出力$\hat{x}[n]$は，

$$\hat{x}[2m+k] = u_{k+2}[m-1] + u_k[m], k = 0,1 \quad \cdots\cdots (7.46)$$



(a) 重複ブロック処理



(b) 非重複ブロック処理

図7.18　重複ブロック処理と非重複ブロック処理

の関係より，要素を半分ずらしながら重複加算することによって与えられる．

(1) DCTとの関係

議論は，より一般的なM等分割フィルタ・バンクに容易に拡張できる．分析フィルタ$H_k(z)$のタップ数をL_Aとすると，分析器は$M \times L_A$行列を用いた演算に帰着する．一方，合成フィルタ$F_k(z)$のタップ数をL_Sとすると，合成器は$M \times L_S$行列を用いた演算に帰着する．

$L_A = L_S = 2M$の場合，**図7.18(a)**に示すように重複のあるブロック処理となり，$L_A = L_S = M$の場合，

図7.18(**b**)に示すように独立したブロック処理となる．変換行列とフィルタ係数は，

$$[\mathbf{P}_{\mathrm{A}}]_{k,n} = h_k[L_A - 1 - n] \quad \cdots\cdots (7.47)$$

$$[\mathbf{P}_{\mathrm{S}}]_{k,n} = f_k[n] \quad \cdots\cdots (7.48)$$

のように関係する．

N点DCT(1次元)によるブロックDCTは，分割数$M = N$，フィルタ・タップ数$L_A = L_S = N$の最大間引き非重複型フィルタ・バンクによる処理と解釈できる．

例題7.17　**DCT基底の周波数特性**

4点DCT(1次元)の基底ベクトルを分析フィルタ$h_k[n]$と解釈して，周波数振幅特性を確認してみよう．

解

以下にMATLAB上でのコマンド例を示そう．

```
% DCT 点数
nPoints = 4;
% DCT 行列
E = dct(eye(nPoints));
% 周波数振幅特性
fftPoints = 512;
analysisFilters = fliplr(E);
for iChannel = 1:nPoints
  [H(:,iChannel),W] = ...
    freqz(analysisFilters(iChannel,:),1,...
      fftPoints);
end
plot(W/pi,20*log10(abs(H)));
```

図7.19に4点DCTの周波数振幅特性を示そう．

実習7.11　**DCT基底の周波数特性**

M-file：practice07_11.m

8点DCT基底の周波数特性も確かめてみよう．

(2) ポリフェーズ実現

図7.17は，マルチレート信号処理の性質を用いると**図7.20**(**a**)に示すポリフェーズ・フィルタによる構成と等価変換できる．同じように，より一般的なM等分割フィルタ・バンクも，**図7.20**(**b**)のように構成することができる．ただし，$\mathbf{E}(z)$と$\mathbf{R}(z)$は，それぞれ分析フィルタ$H_k(z)$のタイプIのポリ

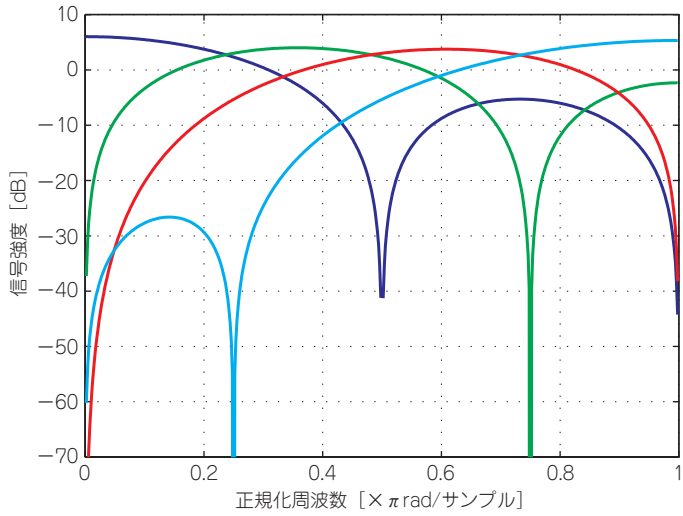
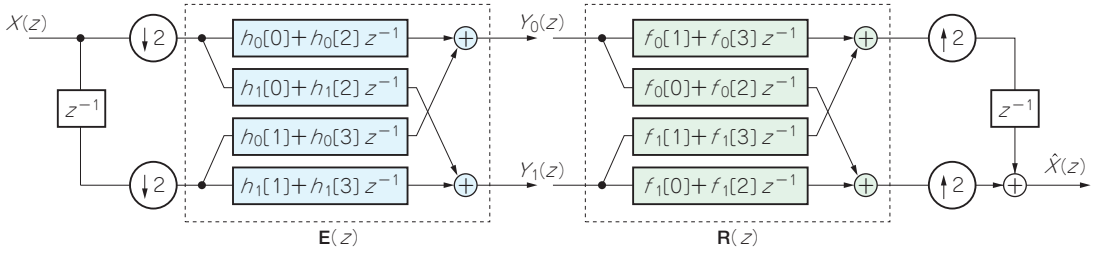
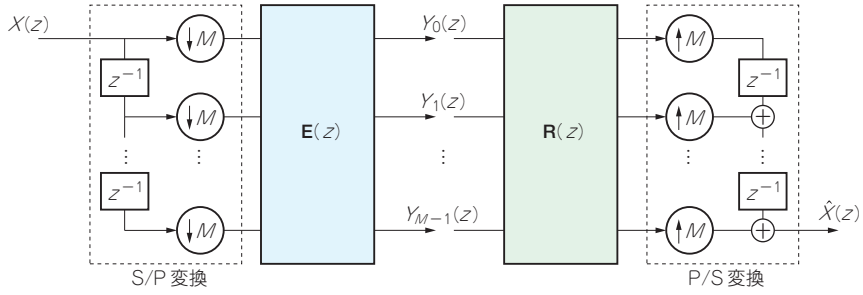


図7.19　4点DCTの周波数振幅特性



（**a**）**図7.17**の等価表現



（**b**）M等分割フィルタ・バンク

図7.20　フィルタ・バンクのポリフェーズ実現

フェーズ・フィルタおよび合成フィルタ$F_k(z)$のタイプⅡのポリフェーズ・フィルタからなる$M \times M$行列多項式行列である．その行列要素は，

$$[\mathbf{E}(z)]_{k,\ell} = E_{k,\ell}(z) \quad \cdots\cdots (7.49)$$

$$[\mathbf{R}(z)]_{\ell,k} = R_{\ell,k}(z) \quad \cdots\cdots (7.50)$$

のように定義される．図中，S/P変換はシリアル-パラレル変換，P/S変換はパラレル-シリアル変換

となっており，$E_{k,1}(z)$，$R_{1,k}(z)$は，それぞれ$H_k(z)$の1番目のタイプⅠポリフェーズ・フィルタ（位相数M），$F_k(z)$の1番目のタイプⅡポリフェーズ・フィルタ（位相数M）である．すなわち，

$$H_k(z) = \sum_{\ell=0}^{M-1} E_{k,\ell}(z^M) z^{-\ell} \quad \cdots\cdots (7.51)$$

$$F_k(z) = \sum_{\ell=0}^{M-1} R_{\ell,k}(z^M) z^{-(M-1+\ell)} \quad \cdots\cdots (7.52)$$

である．$\mathbf{E}(z)$，$\mathbf{R}(z)$は，**ポリフェーズ行列**と呼ばれる．

なお，任意の整数m_0に対して，

$$\mathbf{P}(z) = \mathbf{R}(z)\mathbf{E}(z) = cz^{-m_0}\mathbf{I} \quad \cdots\cdots (7.53)$$

が成立すると，フィルタ・バンクは完全再構成条件（十分条件）を満たし，$T(z) = cz^{-(Mm_0+M-1)}$となる．ポリフェーズ行列による表現は，フィルタ・バンクの設計や実現において大変重要である．

例題7.18　ポリフェーズ実現

実習7.10のフィルタ・バンクをポリフェーズ実現してみよう．

解

以下に，MATLABによる実現のコマンド例を示す．

```
% 入力信号の生成
xOrg = 0:5;
% 分析フィルタ
h0 = [ (1+sqrt(3)) (3+sqrt(3)) ...
          (3-sqrt(3)) (1-sqrt(3)) ]/8;
h1 = [ (1-sqrt(3)) -(3-sqrt(3)) ...
          (3+sqrt(3)) -(1+sqrt(3)) ]/8;
% タイプⅠのポリフェーズ・フィルタ（分析器）
e00 = h0(1:2:end);
e01 = h0(2:2:end);
e10 = h1(1:2:end);
e11 = h1(2:2:end);
% 合成フィルタ
f0 = 2*fliplr(h0); % 時間反転して2倍
f1 = 2*fliplr(h1); % 時間反転して2倍
% タイプⅡのポリフェーズ・フィルタ（合成器）
r00 = f0(2:2:end);
r10 = f0(1:2:end);
r01 = f1(2:2:end);
```

```
r11 = f1(1:2:end);
% S/P 変換
x0 = [xOrg(1:2:end) 0];
x1 = [0 xOrg(2:2:end)];
% 分析処理（ポリフェーズ実現）
y0 = conv(e00,x0) + conv(e01,x1);
y1 = conv(e10,x0) + conv(e11,x1);
% 合成処理（ポリフェーズ実現）
u0 = conv(r00,y0) + conv(r01,y1);
u1 = conv(r10,y0) + conv(r11,y1);
% P/S 変換
xRec = upsample(u0,2,1)+upsample(u1,2);
```

変数 xOrg と xRec を比べると，遅延を除いて信号が完全に復元されることを確認できる．

実習7.12　**ポリフェーズ実現**

M-file：practice07_12.m

ほかの入力信号でも完全再構成を確かめてみよう．

● リフティング手法

フィルタ・バンクの設計・実現の有効な手法としてリフティング手法がある．完全再構成を容易に実現でき，静止画像符号化の国際標準方式JPEG2000で採用されるなど，実用性が高い．

図7.21に5/3変換の，**図7.22**に9/7変換のリフティング構成を示す．JPEG2000 Part Ⅰでは，5/3変換をロスレス（無歪）モードに，9/7変換をロッシ（有歪）モードに採用している．

(1) 5/3変換

5/3変換の分析・合成フィルタのインパルス応答は，**表7.1**のように5および3タップのフィルタから与えられる．

JPEG2000では，**図7.21**のQ_0，Q_1の位置に丸め処理を挿入し，演算結果の整数化を行っている．

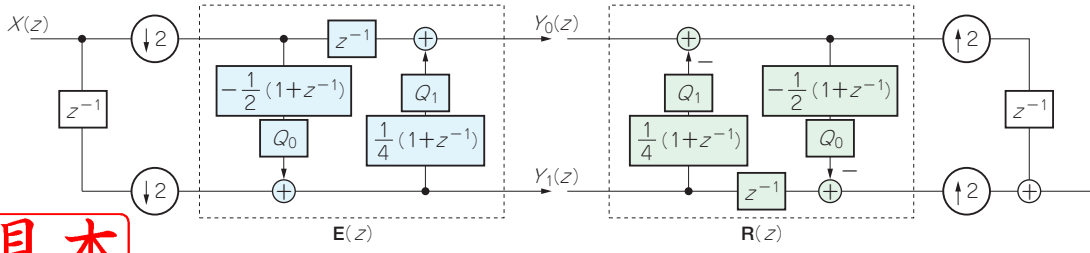


図7.21　リフティング5/3変換

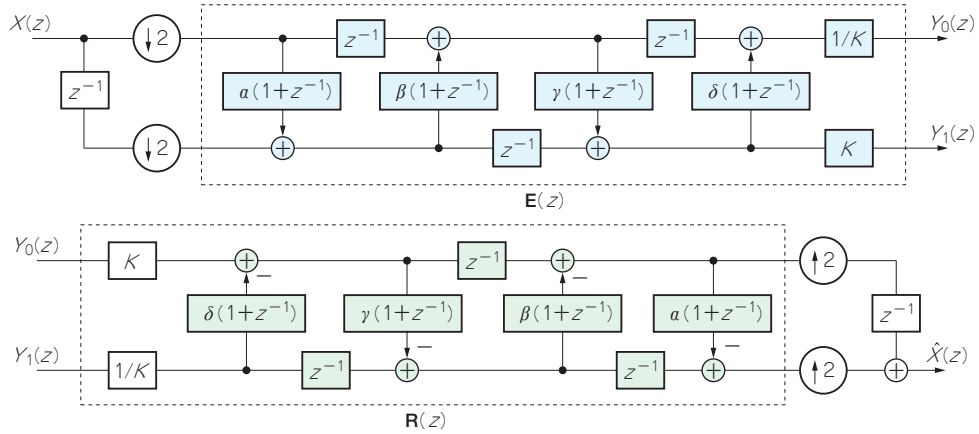


図7.22　リフティング9/7変換

表7.1
5/3変換のインパルス応答（cは対称の中心）

n	$h_0[c \pm n]$	$h_1[c \pm n]$	$f_0[c \pm n]$	$f_1[c \pm n]$
0	6/8	1	1	6/8
1	2/8	−1/2	1/2	−2/8
2	−1/8			−1/8

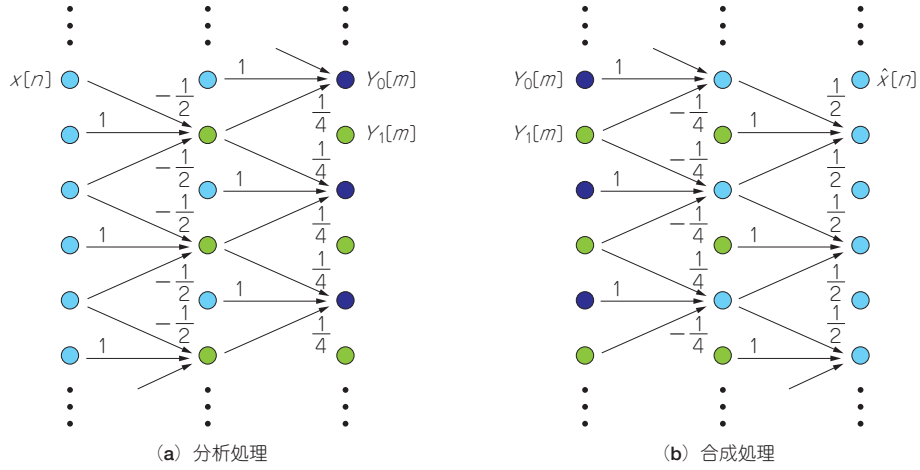


図7.23　5/3変換のインプレース演算実現

リフティング手法は，非線形処理も含めて，順変換と逆変換の各ステップの処理が一致している限り，どのような処理を施しても1ビットの狂いもなく完全再構成を保証する．JPEG2000のロスレス・モードでは，リフティング手法のこの利点を利用している．

なお，リフティング手法の別の利点として，**インプレース(in-place)演算**が可能な点を挙げられる．図7.23は，5/3変換のインプレース演算手順を示している．

例題7.19　リフティング手法

図7.21 に示す5/3変換が，完全再構成条件を満たすことを確認してみよう．また，各フィルタ $H_0(z)$，$H_1(z)$，$F_0(z)$，$F_1(z)$のインパルス応答を確認してみよう．ただし，Q_0，Q_1の影響は無視してよい．

解

$$\mathbf{E}(z)=\begin{pmatrix}1 & \frac{1}{4}(1+z^{-1})\\ 0 & 1\end{pmatrix}\begin{pmatrix}z^{-1} & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0\\ -\frac{1}{2}(1+z^{-1}) & 1\end{pmatrix}$$

$$\mathbf{R}(z)=\begin{pmatrix}1 & 0\\ \frac{1}{2}(1+z^{-1}) & 1\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & z^{-1}\end{pmatrix}\begin{pmatrix}1 & -\frac{1}{4}(1+z^{-1})\\ 0 & 1\end{pmatrix}$$

より，明らかに

$$\mathbf{P}(z)=\mathbf{R}(z)\mathbf{E}(z)=z^{-1}\mathbf{I}$$

となる．よって，式(7.53)より5/3変換は完全再構成を保証する．また，各フィルタのインパルス応答を確認するMATLABコマンドの例を以下に示そう．

```
% タイプⅠのポリフェーズ・フィルタ（分析器）
u00 = [1 0]; u01 = [1 1]/4; % 上向きステップ
u10 = [0 0]; u11 = [1 0];
ud00 = [0 u00]; ud01 = [u01 0]; % 遅延操作
ud10 = [0 u10]; ud11 = [u11 0];
p00 = [1 0]; p01 = [0 0]; % 下向きステップ
p10 = -[1 1]/2; p11 = [1 0];
e00 = conv(ud00,p00)+conv(ud01,p10); % 各要素
e01 = conv(ud00,p01)+conv(ud01,p11);
e10 = conv(ud10,p00)+conv(ud11,p10);
e11 = conv(ud10,p01)+conv(ud11,p11);
% 分析フィルタ
h0 = upsample(e00,2)+upsample(e01,2,1);
h1 = upsample(e10,2)+upsample(e11,2,1);
fvtool(h0,1,h1,1);
% タイプⅡのポリフェーズ・フィルタ（合成器）
p00 = [1 0]; p01 = [0 0]; % 下向きステップ
p10 = [1 1]/2; p11 = [1 0];
pd00 = [p00 0]; pd01 = [0 p01]; % 遅延操作
```

```
pd10 = [p10 0]; pd11 = [0 p11];
u00 = [1 0]; u01 = -[1 1]/4; % 上向きステップ
u10 = [0 0]; u11 = [1 0];
r00 = conv(pd00,u00)+conv(pd01,u10); % 各要素
r01 = conv(pd00,u01)+conv(pd01,u11);
r10 = conv(pd10,u00)+conv(pd11,u10);
r11 = conv(pd10,u01)+conv(pd11,u11);
% 合成フィルタ
f0 = upsample(r00,2,1)+upsample(r10,2);
f1 = upsample(r01,2,1)+upsample(r11,2);
fvtool(f0,1,f1,1);
```

(2) 9/7 変換

9/7変換の分析・合成フィルタは，**表7.2**のように9および7タップのフィルタから与えられる．JPEG2000 Part Iのロッシ・モードで利用される9/7変換は，**図7.22**のα，β，γ，δ，Kに近似値

$\alpha \approx -1.586134342059924$

$\beta \approx -0.052980118572961$

$\gamma \approx 0.882911075530934$

$\delta \approx 0.443506852043971$

$K \approx 1.230174104914001$

を与えればよい．なお，5/3変換と同じように9/7変換もインプレース演算が可能である．

実習7.13　**リフティング手法**

M-file：practice07_13.m

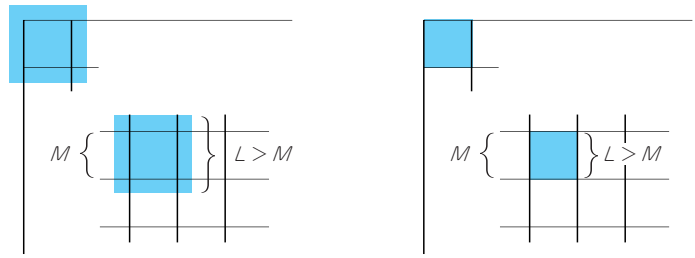
9/7変換についても完全再構成条件と各フィルタのインパルス応答を確認してみよう．

● 画像処理応用

1次元のフィルタ・バンクは，DCTと同じように，分離処理として画像の分析・合成に容易に適用することができる．ただし，DCTと異なり，重複ブロック処理となる．また，画像は空間的に有限な広がりをもつため，重複ブロック処理では，画像の境界において特別な処理を必要とする．

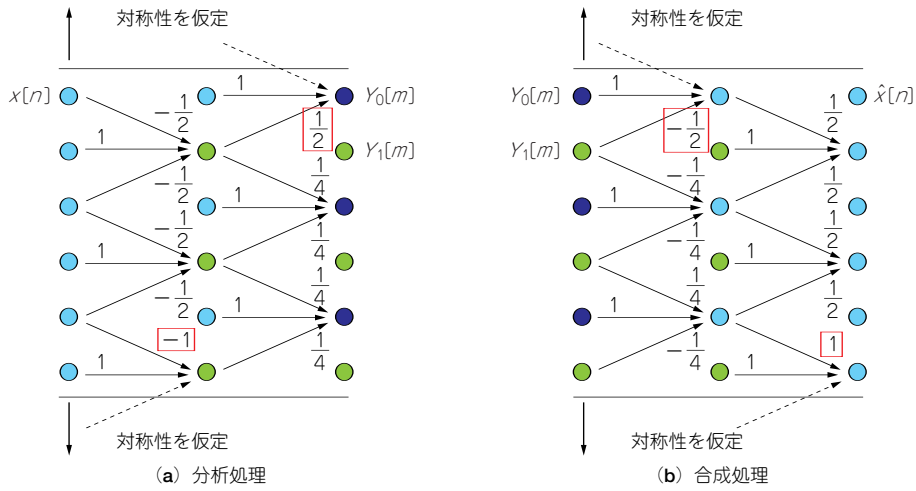
表7.2
9/7変換のインパルス応答（cは対称の中心）

n	$h_0[c \pm n]$	$h_1[c \pm n]$	$f_0[c \pm n]$	$f_1[c \pm n]$
0	0.602949	1.115087	1.115087	0.602949
1	0.266864	−0.591272	0.591272	−0.266864
2	−0.078223	−0.057544	−0.057544	−0.078223
3	−0.016864	0.091272	−0.091272	0.016864
4	0.026749			0.026749



（a）重複ブロック処理　　（b）非重複ブロック処理

図7.24　重複ブロック処理と非重複ブロック処理



（a）分析処理　　（b）合成処理

図7.25
対称拡張法を適用した5/3変換のインプレース演算実現（赤線で囲った部分は，対称拡張法と等価な処理を行うために変更した係数値）

図7.24にフィルタのタップ数Lが分割数Mよりも長い場合と，タップ数Lと分割数Mが等しい場合の2次元ブロック処理の様子を示す．**図7.18**と比べてみよう．画像の場合，境界における処理が重要となる．

分析フィルタ，合成フィルタのインパルス応答が対称性（直線位相性）をもつ場合，対称拡張法により完全再構成を保証しながら，フィルタリングによる点数増加を回避できる．ただし，出力も対称となるように，入力とフィルタの対称性，入力信号の長さ，間引き率あるいは補間率の関係に留意しなければならない．詳細な説明は割愛する．

図7.25に，偶数長の信号に対する境界を考慮した5/3変換のインプレース演算実現を示そう．9/7変換の場合は，係数を変えて同じステップを繰り返せばよい．

例題7.20　**画像の5/3変換**

図7.3(a)のモノクロ画像に対して，5/3変換後のサブバンド画像を確かめてみよう．また，5/3逆変換により原画像を復元してみよう．

図7.26　図7.3(a)に対する5/3変換後のサブバンド画像

解

図7.26にサブバンド画像を示す．また，MATLAB上でのコマンド例，5/3順変換・逆変換関数の例を以下に示す．ただし，画像は`pictureGray`に保存されているものとする．

```
% 5/3 順変換
[subLL,subHL,subLH,subHH] = im53trnscq(pictureGray);
% サブバンド画像の表示
picturesSub = [ subLL 4*abs(subHL);
              4*abs(subLH) 4*abs(subHH) ];
figure(1); imshow(uint8(picturesSub));
title('Subband pictures');
% 5/3 逆変換
pictureRec = ...
  im53itrnscq(subLL,subHL,subLH,subHH);
% 復元画像の表示
figure(2); imshow(uint8(pictureRec));
title('Reconstructed picture');
```

関数`im53trnscq`は，画像の5/3順変換を行う関数で，その内容は以下のとおりである．

```
function [subLL,subHL,subLH,subHH] = im53trnscq(fullPicture);
% 倍精度に変換
fullPicture = double(fullPicture);
% 垂直変換（インプレース演算）
```

```
  fullPicture = ...
    predictionStep(fullPicture,-1/2);
  fullPicture = ...
    updateStep(fullPicture,1/4);
  % 水平変換（インプレース演算）
  fullPicture = ...
    predictionStep(fullPicture.',-1/2);
  fullPicture = ...
    updateStep(fullPicture,1/4).';
  % 係数並べ替え
  subLL = fullPicture(1:2:end,1:2:end);
  subHL = fullPicture(1:2:end,2:2:end);
  subLH = fullPicture(2:2:end,1:2:end);
  subHH = fullPicture(2:2:end,2:2:end);
```

関数im53itrnscqは，画像の5/3逆変換を行う関数で，その内容は以下のとおりである．

```
  function fullPicture = im53itrnscq(subLL,subHL,subLH,subHH);
  % 配列の準備
  fullSize = size(subLL)+size(subHH);
  fullPicture = zeros(fullSize);
  % 係数並べ替え
  fullPicture(1:2:end,1:2:end) = subLL;
  fullPicture(1:2:end,2:2:end) = subHL;
  fullPicture(2:2:end,1:2:end) = subLH;
  fullPicture(2:2:end,2:2:end) = subHH;
  % 水平変換（インプレース演算）
  fullPicture = ...
    updateStep(fullPicture.',-1/4);
  fullPicture = ...
    predictionStep(fullPicture,1/2).';
  % 垂直変換（インプレース演算）
  fullPicture = ...
    updateStep(fullPicture,-1/4);
  fullPicture = ...
    predictionStep(fullPicture,1/2);
```

なお，関数predictionStepと関数updateStepは垂直方向にリフティング演算を行う関数で，その内容は以下のとおりである．

```
function picture = predictionStep(picture,p);
%
picture(2:2:end,:) = picture(2:2:end,:) ...
  + p * ( ...
    picture(1:2:end,:) ...
    + [picture(3:2:end,:); ...
      picture(end-1,:)]);
% end of predictionStep
```

および，

```
function picture = updateStep(picture,u);
%
picture(1:2:end,:) = picture(1:2:end,:) ...
  + u * ( ...
    [picture(2,:); ...
    picture(2:2:end-1,:)] ...
    + picture(2:2:end,:));
% end of updateStep
```

上記predictionStep関数，updateStep関数は対称拡張法を適用している．偶数長のみに対応しているが，奇数長への対応も容易である．なお，JPEG2000 Part Iのロスレス・モードのような整数化のための丸め処理は行っていない．

実習7.14　画像の9/7変換

M-file：practice07_14.m

9/7変換についてもサブバンド画像と復元画像を確認してみよう．

7.5　離散時間ウェーブレット変換(DWT)

ウェーブレット変換は，フランス人学者Morletにより石油探索のための人工地震波解析に導入さ

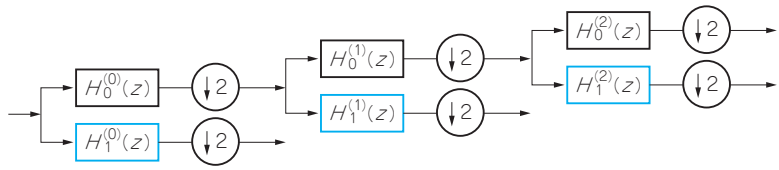
れて以来，多くの物理学者，数学者，工学者たちにより理論的基礎作りや応用が試みられてきた．以下では，ディジタル信号の圧縮や解析に利用される離散時間ウェーブレット変換（DWT）について解説しよう．

● ツリー構成

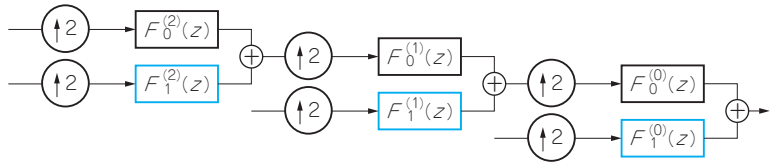
DWTは，オクターブ分割と呼ばれる2分割フィルタ・バンクのツリー構成によって実現される．**図7.27**にレベル3のDWTの構成例を示す．**図7.14**のフィルタ・バンクの低域のチャネルを繰り返し分割し，ツリー構造に配置することで，DWTは構成される．前章で解説したマルチレート信号処理の性質を利用すると，**図7.27**は**図7.28**のように並列構成として等価表現できる．

図7.29に，DCTのような等分割フィルタ・バンク（分割数8）とオクターブ分割のDWT（レベル3）の時間周波数分解能の概略図を示そう．等分割フィルタ・バンクが，周波数帯域によらず時空間分解能が一定であるのに対して，DWTは周波数が高いほど帯域分解能が低く，時間分解能が高くなる．

1次元DWTは分離処理により2次元に拡張できる．**図7.30**に示すように，DWTの基底画像はブロックDCTの基底画像と異なり，互いに重複し，周波数に応じてそのサイズが変わる．このような性質は，人間の視覚特性に適しており，ブロックDCTに比べて，ブロックひずみやモスキート・ノイズの少ない画像圧縮を実現できる．



（**a**）分析器



（**b**）合成器

図7.27　DWTの構成例（3レベル）

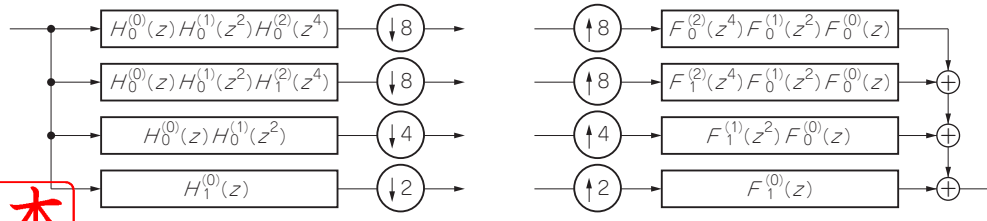


図7.28　DWTの等価並列構成（3レベル）

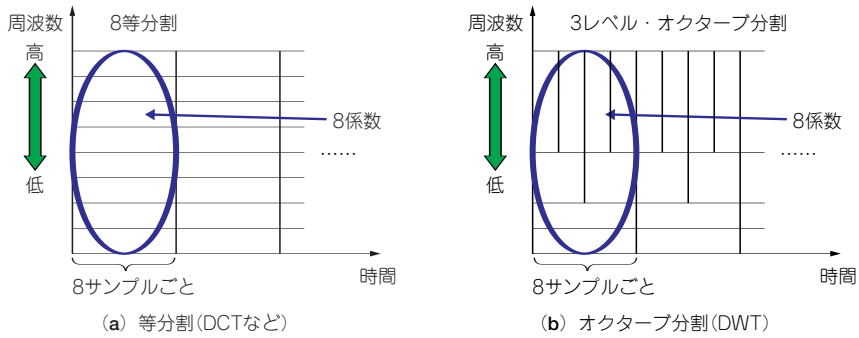


図7.29　時間周波数分解能の比較

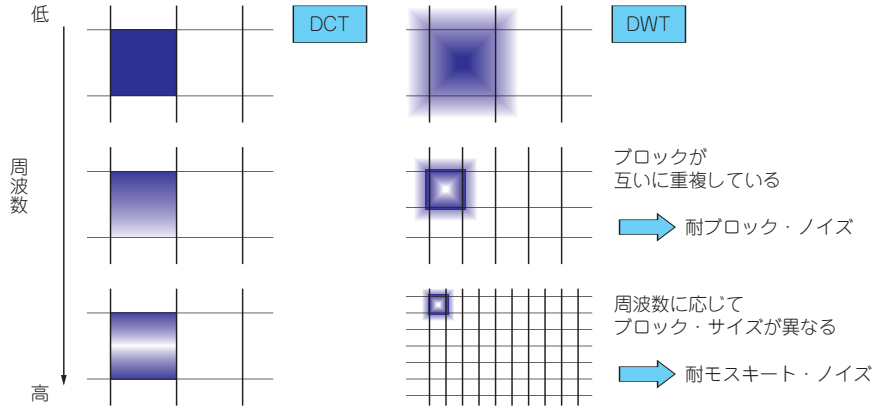


図7.30　周波数帯域と基底画像の比較

例題7.21　**画像の5/3 DWT**

図7.3(a)のモノクロ画像に対して，レベル数3の5/3 DWTを施し，サブバンド画像を確かめてみよう．また，5/3逆DWTにより原画像を復元してみよう．

解

MATLAB上でのコマンド例を以下に示す．ただし，画像は`pictureGray`に保存されているものとする．

```
% 3レベル5/3 DWT
[subLL2, subHL2, subLH2, subHH2] = im53trnscq(pictureGray);
[subLL1, subHL1, subLH1, subHH1] = im53trnscq(subLL2);
[subLL0, subHL0, subLH0, subHH0] = im53trnscq(subLL1);
% サブバンド画像の表示
subband0 = [subLL0 4*abs(subHL0) ;
```

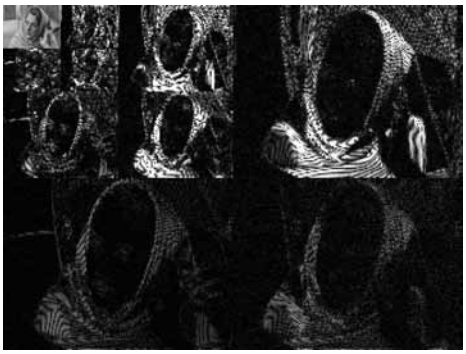


図7.31　図7.3(a)に対する5/3 DWT後のサブバンド画像

```
  4*abs(subLH0) 4*abs(subHH0)];
subband1 = [subband0 4*abs(subHL1) ;
  4*abs(subLH1) 4*abs(subHH1)];
subband2 = [ subband1 4*abs(subHL2);
  4*abs(subLH2) 4*abs(subHH2) ];
figure(1); imshow(subband2);
% 3レベル5/3 IDWT
subLL1 = im53itrnscq(subLL0, subHL0, subLH0, subHH0);
subLL2 = im53itrnscq(subLL1, subHL1, subLH1, subHH1);
pictureRec = im53itrnscq(subLL2, subHL2, subLH2, subHH2);
% 復元画像の表示
figure(2); imshow(pictureRec);
```

図7.31にサブバンド画像を示す．**図7.10(b)**の8×8ブロックDCTの結果と比べてみよう．

実習7.15　**画像の9/7 DWT**

M-file：practice07_15.m

9/7 DWTについてもサブバンド画像と復元画像を確認してみよう．

● レギュラリティ

図7.27のツリー分割を無限に繰り返すことで，連続信号に対する離散ウェーブレット級数(DWS：Discrete-time Wavelet Series)[注7-1]が生成できる．レギュラリティとは，このDWSの既定関数の滑らかさを表す尺度である．元の低域通過フィルタ$H_0(z)$が，ある$Q(z)$を用いて，

注7-1：文献によっては，DWSを離散ウェーブレット変換(DWT)と呼ぶ場合があるので注意が必要．

$$H_0(z) = \left(\frac{1+z^{-1}}{2}\right)^K Q(z) \quad \cdots\cdots (7.54)$$

のように因数分解できるとき，DWSはK次のレギュラリティをもつという[注7-2]．$H_0(z)$が，$z = e^{j\pi} = -1$にK重根をもつことから，Kが大きいほどより滑らかであるといえる．

レギュラリティはDWTにおいても重要である．1次以上のレギュラリティを満たすと，画像圧縮へ応用した際，網目状に模様が入るチェス盤ひずみを回避できる利点がある．例題7.16(ハール基底)，実習7.10(3次Daubechies基底)で紹介した変換，5/3変換，9/7変換のすべてがレギュラリティをもつことが確かめられる．

章末問題

問題7.1　**ユニタリ行列**(MATLAB演習)

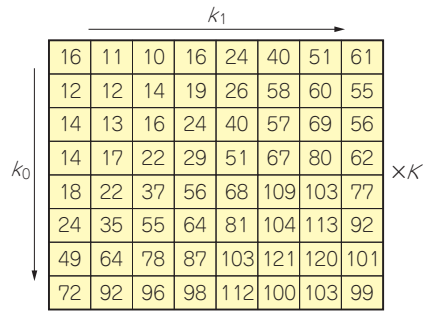
DFT行列がユニタリ行列であることをMATLABで確かめてみよう．M点DFT行列は，`W=dftmtx(M)`で与えられる．また，行列`W`の複素共役転置は`W'`とすればよい．

問題7.2　**直交行列**(MATLAB演習)

DCT行列が直交行列であることをMATLABで確かめてみよう．M点DCT行列は，`C=dct(eye(M))`(Image Processing Toolboxがある場合は `C=dctmtx(M)`)で与えられる．また，行列`C`の転置は`C.'`とすればよい．

問題7.3　**DCT符号化**(MATLAB演習)

図7.3(a)に対して，8×8ブロックDCTを施し，各ブロックの変換係数に対して$K = 1$として量子化テーブル(JPEGの輝度成分用）を使った線形量子化を行い，逆量子化，ブロックIDCTを施して画像を復元し，原画像と比較してみよう．また，$K > 0$の値を変えて試してみよう．



注7-2：厳密には，連続関数への収束条件も必要である．

問題7.4　**実画像に対するKLT行列**（MATLAB演習）

図7.3（**a**）の画像について，垂直方向に連続する8画素で構成した8次元ベクトル集合の共分散行列$\mathbf{R}_{xx}$を求め，KLT行列を求めてみよう．

問題7.5　**KLTの基底画像**（MATLAB演習）

相関係数$\rho = 0.95$のAR(1)過程に対する8点KLTの基底画像を確かめてみよう．また，ほかの相関係数($-1 < \rho < 1$)でも試してみよう．

問題7.6　**ポリフェーズ行列**

実習7.10のフィルタ・バンクをポリフェーズ行列で表現し，式(7.53)の完全再構成条件が満たされることを確認してみよう．

問題7.7　**DWTの基底ベクトル**（MATLAB演習）

3レベルの5/3 DWTおよび9/7 DWTの基底ベクトルを確かめてみよう．**図7.28**を参考にするとよい．また，ほかのレベル数でも確認してみよう．

問題7.8　**DWT符号化**（MATLAB演習）

3レベルの5/3 DWTもしくは9/7 DWTで**図7.3**（**a**）の画像を変換し，いくつかのサブバンド画像をゼロ値に置換してから画像を復元してみよう．また，ほかの画像，ほかのレベル数，量子化なども試してみよう．

問題7.9　**レギュラリティ**（MATLAB演習）

例題7.16（ハール基底），実習7.10（3次Daubechies基底）で紹介した変換，5/3変換，9/7変換の低域通過フィルタ$H_0(z)$が，$z = -1$において，ゼロとなることを複素平面上の零点-極配置をプロットして確かめてみよう．なお，MATLABでは，`zplane`関数もしくは`fvtool`を利用して確認できる．

参考文献

(1) 貴家仁志，村松正吾；マルチメディア技術の基礎DCT入門，CQ出版社，1997年．
(2) 小野定康，鈴木純司；わかりやすいJPEG/MPEG2の実現法，オーム社，1995年．
(3) 藤原洋監修；最新MPEG教科書，アスキー出版局，1994年．
(4) Mohammed Ghanbari；*Standard Codecs: Image Compression to Advanced Video Coding*，The Institution of Electrical Engineers，2003.
(5) Iain E.G. Richardson；*H.264 and MPEG-4 Video Compression*，Wiley，2003.
(6) 小野定康，鈴木純司；わかりやすいJPEG2000の技術，オーム社，2003年．
(7) David S. Taubman and Michael W. Marcellin；*JPEG2000, Image Compression Fundamentals, Standards and Practice*，Kluwer Academic Publishers，2002.

第8章

多次元信号処理の基礎

前章までは，分離処理を中心に多次元信号処理の操作について解説した．本章では，より一般的な多次元信号処理の表現についてまとめ，多次元フーリエ解析や多次元標本化などの基礎理論について概説する．さまざまな例を通して理解を深めよう．

なお，以下ではRは実数，Zは整数，R^DはD次元実数ベクトル，Z^DはD次元整数ベクトルの集合とする．

8.1 多次元信号処理の概要

多次元信号処理の応用は，画像・映像処理をはじめ無線通信，レーダ，ソナー，地震波解析，生体情報処理など，多岐にわたる．その要素技術の多くは，信号解析，ノイズ除去，特徴抽出，レート変換，信号推定，信号復元などであり，システム構成や実現技術において1次元信号処理との共通点も多い．一方で，1次元では存在しない多次元特有の問題もある．

● 多次元信号処理の特徴

1次元信号処理の知識のみで対処しようとすると，多次元の利点を生かせないばかりではなく，思わぬわなにはまることもある．これらの違いの多くは，2次元において既に現れている．以下に例を挙げよう．

- 標本化は標本格子により表現され，標本化密度（レート）以外に標本位置の構造も設定する必要がある．
- 自由度が増え，制約が減る．例えば，標本化定理を満たす信号帯域の形状が無数に存在し，フィルタ仕様の決定が困難な場合がある．
- 多変数多項式は因数分解できるとは限らず，フィルタ設計や安定性の判別が困難な場合がある．

● 多次元信号の表現

ここでは，一般的なD次元の信号について解説を進める．このために，変数のベクトル表記を導入する．

(1) 変数のベクトル表記

D個の連続変数p_dをもつD次元信号

$$x(p_0, p_1, \cdots, p_{D-1}),\ p_d \in R \quad \cdots\cdots (8.1)$$

および，D個の離散変数n_dをもつD次元信号

$$x[n_0, n_1, \cdots, n_{D-1}],\ n_d \in Z \quad \cdots\cdots (8.2)$$

は，変数をD次元ベクトルに置き換えて，それぞれ

$$x(\mathbf{p}),\ \mathbf{p} \in R^D \quad \cdots\cdots (8.3)$$

$$x[\mathbf{n}],\ \mathbf{n} \in Z^D \quad \cdots\cdots (8.4)$$

のように表現できる．静止画像の場合はそれぞれ，

$$x(\mathbf{p}),\ \mathbf{p} \in R^2 \quad \cdots\cdots (8.5)$$

$$x[\mathbf{n}],\ \mathbf{n} \in Z^2 \quad \cdots\cdots (8.6)$$

となる．動画像の場合は，$x(\mathbf{p})$，$\mathbf{p} \in R^3$あるいは$x[\mathbf{n}]$，$\mathbf{n} \in Z^3$となるが，本書では，特に時間変数を強調して，

$$x\begin{pmatrix}\mathbf{p}\\ t\end{pmatrix},\ \mathbf{p} \in R^2, t \in R \quad \cdots\cdots (8.7)$$

$$x\begin{bmatrix}\mathbf{n}\\ k\end{bmatrix},\ \mathbf{n} \in Z^2, k \in Z \quad \cdots\cdots (8.8)$$

と書く場合もある．

例題 8.1　動画像変数のベクトル表記

次の動画像信号について，ベクトル表記を導入して表現してみよう．

$$x[n_0, n_1, k] = \cos(\omega_0(n_0 - v_0 k) + \omega_1(n_1 - v_1 k))$$

解

$\mathbf{n} = [n_0, n_1]^T$，$\omega = (\omega_0, \omega_1)^T$，$\mathbf{v} = (v_0, v_1)^T$とすると，

$$x\begin{bmatrix}\mathbf{n}\\ k\end{bmatrix} = \cos(\omega^T(\mathbf{n} - \mathbf{v}k))$$

という表現を得る．

(2) 単位インパルス信号

変数のベクトル表記を用いると，D次元インパルス信号は

$$\delta[\mathbf{n}] = \begin{cases} 1 & \mathbf{n} = \mathbf{0} \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (8.9)$$

のように簡潔に定義できる．ただし，$\mathbf{n} \in Z^D$であり，$\mathbf{0}$はすべての要素が0となるD次元ベクトルとする．1次元のインパルス信号$\delta[n]$，$n \in Z$を用いて

$$\delta[\mathbf{n}] = \prod_{d=0}^{D-1} \delta[n_d] \quad \cdots\cdots (8.10)$$

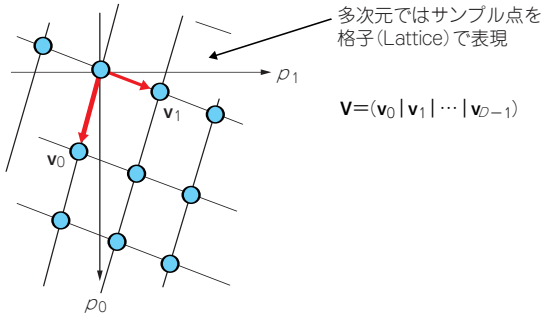


図8.1　多次元信号の標本化点

のように表現できる．

(3) 畳み込み演算

多次元の線形シフト不変システム$T\{\cdot\}$は，そのインパルス応答$h[\mathbf{n}] = T\{\delta[\mathbf{n}]\}$により，その入出力関係を畳み込み演算によって表現できる．いま，変数のベクトル表記を用いると，D次元線形シフト不変システムの入出力関係は

$$y[\mathbf{n}] = T\{x[\mathbf{n}]\} = \sum_{\mathbf{k}\in Z^D} x[\mathbf{k}]\,h[\mathbf{n}-\mathbf{k}] = \sum_{\mathbf{k}\in Z^D} h[\mathbf{k}]\,x[\mathbf{n}-\mathbf{k}] = h[\mathbf{n}] * x[\mathbf{n}] \qquad \mathbf{n}\in Z^D \quad \cdots\cdots (8.11)$$

のように表現される．一般的なD次元の線形シフト不変システムについても，2次元の場合と同じように，インパルス応答$h[\mathbf{n}]$がそのシステムを特徴づけ，IIR/FIR，分離性，因果性，安定性などに性質の違いが現れる．

(4) 標本化

D次元のアナログ信号$x(\mathbf{p})$の標本化は，$D\times D$の非特異行列$\mathbf{V}$によって

$$x[\mathbf{n}] = x(\mathbf{V}\mathbf{n}) \quad \cdots\cdots (8.12)$$

のように表現される．ここで，$\mathbf{V}$は**標本化行列**と呼ばれる．標本化点の集合(標本化格子)は，**図8.1**に示すように行列$\mathbf{V}$の列ベクトル$\mathbf{v}_d$で生成される並行超平面体(2次元の場合は平行四辺形)を標本空間に敷き詰めてできる頂点の集合となる．標本化点の集合はLAT($\mathbf{V}$)と表され，

$$\mathrm{LAT}(\mathbf{V}) = \{\mathbf{V}\mathbf{n}\in R^D \,|\, \mathbf{n}\in Z^D\} \quad \cdots\cdots (8.13)$$

と定義される．

画像表現において最も素直な方形標本化，$x[n_0, n_1] = x(n_0P_0, n_1P_1)$の場合，標本化行列$\mathbf{V}$は

$$\mathbf{V} = (\mathbf{v}_0\,|\,\mathbf{v}_1) = \begin{pmatrix} P_0 & 0 \\ 0 & P_1 \end{pmatrix} \quad \cdots\cdots (8.14)$$

のように与えられる．方形標本化の標本化行列は，次元によらず常に対角行列として与えられる．

例題8.2　インターレース映像の標本化

図8.2に，インターレース映像の標本化格子を示す．この標本化行列を求めよう．

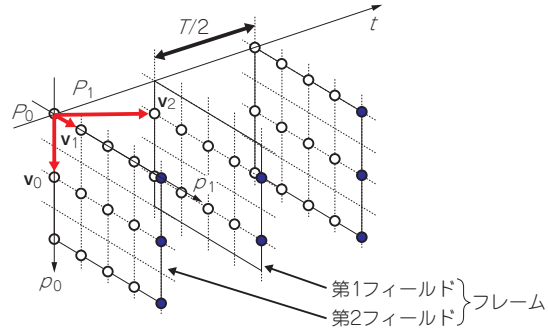


図8.2
インターレース映像の標本化格子

解

標本化格子の基本となる平行六面体を考える．これを構成する三つの列ベクトルからなる3×3行列が標本化行列となるので，**図8.2**より標本化行列**V**は，

$$\mathbf{V} = (\mathbf{v}_0 \mid \mathbf{v}_1 \mid \mathbf{v}_1) = \begin{pmatrix} 2P_0 & 0 & P_0 \\ 0 & P_1 & 0 \\ 0 & 0 & \dfrac{T}{2} \end{pmatrix}$$

と与えられる．

なお，同じ格子を意味する行列は一意ではない．行列式の絶対値が‘1’となる整数行列**E**（ユニモジュラ整数行列）を**V**に右から掛けて得られる$\mathbf{V}' = \mathbf{EV}$もまた同じ格子を示す標本化行列となる．

8.2　多次元信号の周波数と周期性

多次元におけるフーリエ解析への準備として，以下では，多次元信号の周波数と周期性について触れておこう．

● 多次元信号の周波数

1次元余弦波の周波数は，単位時間（1秒）当たりの波の数で与えられる．余弦波の位相の時間微分によって角周波数Ωが与えられ，$F = \Omega/2\pi$として周波数が与えられる．

以下は，2次元の連続変数をもつ余弦波である．

$$x(p_0, p_1) = \cos(\Omega_0 p_0 + \Omega_1 p_1) \quad \cdots\cdots (8.15)$$

各軸の角周波数は，それぞれの変数による位相の偏微分として

$$\frac{\partial(\text{位相})}{\partial p_d} = \Omega_d,\ d = 0, 1 \quad \cdots\cdots (8.16)$$

と与えられる．2次元以上の場合も同じように，各軸の角周波数が各変数による位相の偏微分として与えられる．

多次元の余弦波はベクトル表記によって

$$x(\mathbf{p}) = \cos(\Omega^T \mathbf{p}),\ \ \mathbf{p}, \Omega \in R^D \quad \cdots\cdots (8.17)$$

のように簡潔に表すことができる．さらに，オイラーの公式より，

$$x(\mathbf{p}) = \frac{1}{2}\left(e^{j\Omega^T \mathbf{p}} + e^{-j\Omega^T \mathbf{p}}\right) \quad \cdots\cdots (8.18)$$

なので，多次元余弦波は角周波数$\pm\boldsymbol{\Omega}$の多次元複素正弦波$e^{j\boldsymbol{\Omega}^T\mathbf{p}}$と$e^{-j\boldsymbol{\Omega}^T\mathbf{p}}$から構成されているともいえる．

例題8.3　2次元余弦波

角周波数$\boldsymbol{\Omega} = (4\pi, 2\pi)^T$の2次元余弦波の波形を見てみよう．

解

以下に，MATLAB上でのコマンド例を示そう．

```
% 垂直角周波数
Omega0 = 4*pi;
% 水平角周波数
Omega1 = 2*pi;
% 標本点の定義
[p0,p1] = meshgrid(...
    -0.5:0.01:0.5,-0.5:0.01:0.5);
% 2次元余弦波の標本
x = cos(Omega0*p0 + Omega1*p1);
% 2次元余弦波の表示
surf(p1,p0,x);
xlabel('p_1');
ylabel('p_0');
zlabel('Magnitude');
colormap gray;
shading interp;
axis([-0.5 0.5 -0.5 0.5 -2 2]);
axis square;
```

図8.3に波形を示す．p_0軸に添った波の数がp_1軸のものに比べて倍であることが確かめられる．

実習8.1　**2次元余弦波**

M-file：practice08_1.m

さまざまな角周波数$\boldsymbol{\Omega}$に対する2次元余弦波の波形を確かめてみよう．

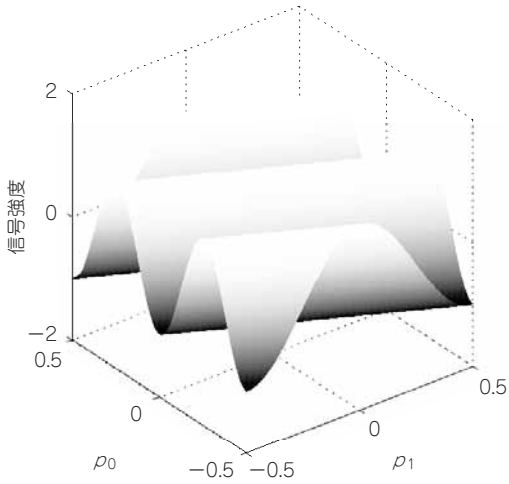


図8.3　2次元余弦波の波形

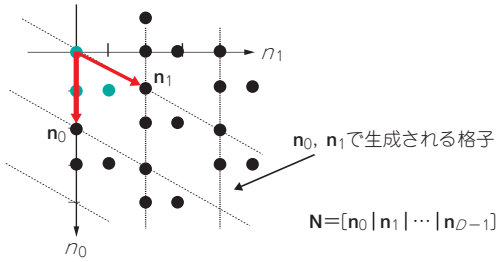


図8.4　非方形周期信号

● 周期信号

続いて，多次元信号における周期性について解説しよう．

(1) 周期行列

多次元周期信号は，連続変数と離散変数共に，それぞれ

$$x[\mathbf{n}]=x[\mathbf{n}-\mathbf{N}\mathbf{l}] \quad \mathbf{l}\in Z^D \qquad \cdots\cdots(8.19)$$

$$x(\mathbf{p})=x(\mathbf{p}-\mathbf{U}\mathbf{q}) \quad \mathbf{q}\in R^D \qquad \cdots\cdots(8.20)$$

という条件を満たす信号として定義される．ここで，$\mathbf{N}$は$D\times D$非特異整数行列，$\mathbf{U}$は$D\times D$非特異実数行列であり，共に**周期行列**と呼ばれる．

2次元の離散信号において，$\mathbf{N}$がdiag(N_0, N_1)のように対角行列(diagonal matrix)である場合を考えよう．すると，周期性の条件は

$$x\begin{bmatrix}n_0\\n_1\end{bmatrix}=x\begin{bmatrix}n_0-N_0\mathrm{l}_0\\n_1-N_1\mathrm{l}_1\end{bmatrix} \quad n_0, n_1, \mathrm{l}_0, \mathrm{l}_1\in Z \qquad \cdots\cdots(8.21)$$

のように表現することができる．

例題8.4　**非方形周期信号**

図8.4に示す非方形周期信号の周期行列を求めよう．

解

2次元信号の周期行列は2×2行列として与えられる．基本周期となる平行四辺形を生成するベクトルを行列の列ベクトルとすればよいので，周期行列は

$$\mathbf{N}=[\mathbf{n}_0|\mathbf{n}_1]=\begin{bmatrix}2&1\\0&2\end{bmatrix}$$

と与えられる．なお，標本化行列の場合と同じように，同じ周期を与える行列は一意に決まらな

いことに注意する．

(2) 基本周期

基本周期内の整数ベクトルの集合がしばしば重要となる．ここで周期行列$\mathbf{N}$の各列ベクトル$\mathbf{n}_d$で生成される平行四辺形(超平面体)内の整数ベクトル集合を$\mathcal{N}(\mathbf{N})$と定義しよう．

例題8.5　基本周期内の整数ベクトル集合

図8.4に示す非方形周期信号の基本周期内の整数ベクトル集合を求めよう．

解

$$\mathcal{N}(\mathbf{N}) = \{\mathbf{n} \in Z^2 \mid \mathbf{n} = \mathbf{N}\mathbf{x}, \quad \mathbf{x} \in [0,1)^2\} = \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}2\\1\end{bmatrix}\right\}$$

ただし，

$$[0,1)^D = \{x \mid 0 \le x_d < 1,\ d = 0,1,\cdots,D-1\}$$

8.3　多次元フーリエ解析

1次元の場合と同じように，信号のスペクトラムやシステムの周波数特性について知ることは，システム設計において必要不可欠である．以下では，多次元におけるフーリエ解析について，例題を交えながら解説しよう．

● フーリエ変換の定義

1次元の場合と同じように，多次元のフーリエ解析においても，取り扱う信号の種類によっていくつかの定義が存在する．以下では，多次元のフーリエ変換，離散空間フーリエ変換，離散フーリエ変換の定義を示そう．

(1) フーリエ変換(FT)

まず，変数が連続かつ非周期(孤立)である信号$x(\mathbf{p})$のフーリエ解析法であるフーリエ変換(FT：Fourier Transform)とその逆変換を以下に示す．

$$X(j\Omega) = \int_{\mathbf{p}\in R^D} x(\mathbf{p}) e^{-j\Omega^T \mathbf{p}} d\mathbf{p} \qquad \Omega \in R^D \quad \cdots\cdots (8.22)$$

$$x(\mathbf{p}) = \frac{1}{(2\pi)^D} \int_{\Omega\in R^D} X(j\Omega) e^{j\Omega^T \mathbf{p}} d\Omega \qquad \mathbf{p} \in R^D \quad \cdots\cdots (8.23)$$

上記の積分は，変数$\mathbf{p}$あるいはΩの各要素を積分変数とする重積分である．ここで注意すべき点は，変数が連続かつ非周期(孤立)である信号$x(\mathbf{p})$のフーリエ変換$X(j\boldsymbol{\Omega})$もまた，変数が連続かつ非周期(孤立)の信号となる点である．

(2) 離散空間フーリエ変換(DSFT)

次に，変数が離散かつ非周期(孤立)である信号$x[\mathbf{n}]$のフーリエ解析法である離散空間フーリエ変換(DSFT：Discrete Space FT)とその逆変換を以下に示す．

$$X(e^{j\omega^T}) = \sum_{\mathbf{n}\in Z^D} x[\mathbf{n}]e^{-j\omega^T n} \qquad \omega\in[R,2\pi)^D \quad \cdots\cdots(8.24)$$

$$x[\mathbf{n}] = \frac{1}{(2\pi)^D}\int_{\omega\in[0,2\pi)^D} X(e^{j\omega^T})e^{j\omega^T\mathbf{n}}d\omega \qquad \mathbf{n}\in Z^D \quad \cdots\cdots(8.25)$$

ただし，$[0,2\pi)^D$は各要素の積分範囲が0～2πであることを意味する．すなわち，

$$[0,2\pi)^D = \{x\in R^D \mid 0 \le x_d < 2\pi, d = 0,1,\cdots,D-1\}$$

である．上記の積分もまた，変数ωの各要素を積分変数とする重積分である．ここで注意すべき点は，変数が離散かつ非周期（孤立）である信号$x[\mathbf{n}]$のDSFT $X(e^{j\omega^T})$は，変数が連続かつ周期信号となる点である．

(3) 離散フーリエ変換（DFT）

最後に，変数が離散かつ有限（周期）である信号$x[\mathbf{n}]$のフーリエ解析法である離散フーリエ変換（DFT：Discrete FT）とその逆変換を以下に示す．

$$X[\mathbf{k}] = \sum_{\mathbf{n}\in\mathcal{N}(\mathbf{N})} x[n]e^{-j\mathbf{k}^T(2\pi\mathbf{N}^{-1})\mathbf{n}} \qquad \mathbf{k}\in\mathsf{N}\ (\mathbf{N}^T) \quad \cdots\cdots(8.26)$$

$$x[\mathbf{n}] = \frac{1}{|\det\mathbf{N}|}\sum_{\mathbf{k}\in\mathcal{N}(\mathbf{N}^T)} X[\mathbf{k}]e^{j\mathbf{k}^T(2\pi\mathbf{N}^{-1})\mathbf{n}} \qquad \mathbf{n}\in\mathcal{N}(\mathbf{N}) \quad \cdots\cdots(8.27)$$

ここで注意すべき点は，変数が離散かつ有限（周期）である信号$x[\mathbf{n}]$のDFT $X[\mathbf{k}]$は変数が離散かつ有限（周期的）な信号となる点である．なお，$|\det\mathbf{N}|$は行列$\mathbf{N}$の行列式の絶対値である．

特に，周期行列$\mathbf{N}$が対角行列の場合，分離処理が可能となり，多次元DFTが各次元ごとの1次元DFTの繰り返しに帰着する．

例題8.6　2次元DFTの分離処理

周期行列$\mathbf{N}$が対角行列$\mathrm{diag}(N_0,N_1)$の場合，この2次元DFTが分離処理可能であることを示そう．

解

$W_N = e^{-j\frac{2\pi}{N}}$として，定義より

$$X[{}^{k_0}_{k_1}] = \sum_{n_0=0}^{N_0-1}\sum_{n_1=0}^{N_1-1} x[{}^{n_0}_{n_1}]e^{-j\left(\frac{2\pi}{N_0}k_0n_0+\frac{2\pi}{N_1}k_1n_1\right)} = \underbrace{\sum_{n_1=0}^{N_1-1}\left(\underbrace{\sum_{n_0=0}^{N_0-1} x[{}^{n_0}_{n_1}]W_{N_0}^{k_0n_0}}_{1\text{-}D\,\mathrm{DFT}}\right)W_{N_1}^{k_1n_1}}_{1\text{-}D\,\mathrm{DFT}} = \underbrace{\sum_{n_1=0}^{N_1-1} X_0[{}^{k_0}_{n_1}]W_{N_1}^{k_1n_1}}_{1\text{-}D\,\mathrm{DFT}}$$

のような表現が得られる．ただし，

$$X_0[{}^{k_0}_{n_1}] = \sum_{n_0=0}^{N_0-1} x[{}^{n_0}_{n_1}]W_{N_0}^{k_0n_0}$$

結果として，各次元ごとの1次元DFTに帰着する．

1次元DFTの計算には，1次元の高速フーリエ変換（FFT）を利用できる．MATLABでは，2次元分離型DFTを行う関数として`fft2`関数が用意されている．また，多次元分離型FFTを行う関数として`fftn`関数が用意されている．

● 周波数特性

フーリエ変換(FT)，離散空間フーリエ変換(DSFT)，離散フーリエ変換(DFT)の結果は，信号が実数でも一般的には複素数となる．従って，しばしば極座標表現により，その振幅と位相を分けて議論する．

(1) 振幅特性と位相特性

今，離散信号$x[\mathbf{n}]$を考えよう．この離散信号$x[\mathbf{n}]$がどのような角周波数の複素正弦波によって構成されているかを知りたければ，そのDSFT $X(e^{j\omega T})$を求めればよい．これを信号$x[\mathbf{n}]$の周波数特性と呼ぶ．また，その振幅を振幅特性，位相を位相特性と呼び，これらは，

周波数特性：$X(e^{j\omega T}) = A(\omega)e^{j\theta(\omega)}$

振幅特性：$A(\omega) = |X(e^{j\omega T})|$

位相特性：$\theta(\omega) = \angle X(e^{j\omega T})$

と関係付けられる．

(2) DSFTとDFTの関係

DFTはDSFTを周波数領域で標本化した結果に一致する．実際，$\mathbf{n} \notin \mathrm{N}\ (\mathbf{N})$において$x[\mathbf{n}] = 0$ならば，$\omega = 2\pi\mathbf{N}^{-\mathrm{T}}\mathbf{k}$をDSFTの定義に代入することで，DFTの定義

$$X[\mathbf{k}] = X(e^{j\omega T})\big|_{\omega=2\pi\mathbf{N}^{-T}\mathbf{k}} \quad \cdots\cdots (8.28)$$

が与えられる．$|\det\mathbf{N}|$を大きくとることで，1周期当たりの周波数標本点が増加し，DSFTの良い近似を与える．具体的には，信号のサポート領域外でゼロ値を想定し，大きな点数のDFT(FFT)を行えばよい．

例題8.7　静止画像のスペクトラム

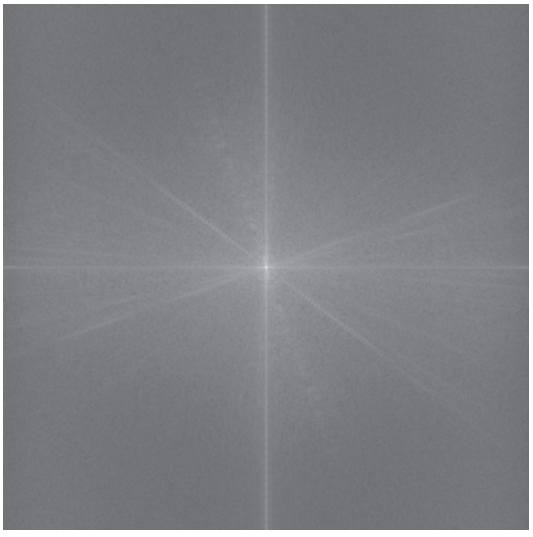
図8.5(a)の画像の振幅特性と位相特性を確認しよう．

解

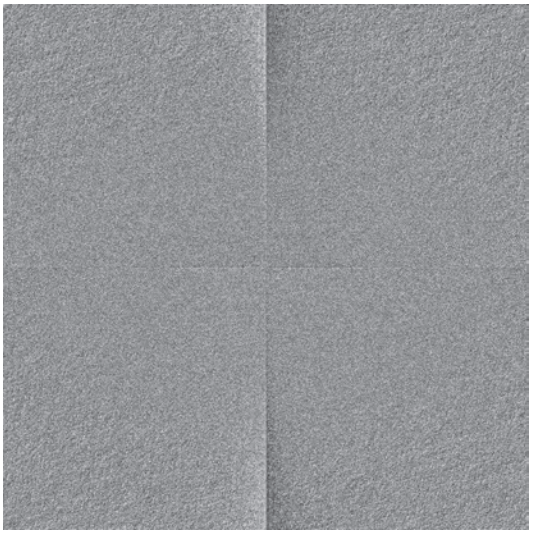
MATLAB上でのコマンド例を以下に示そう．なお，ワークスペースにおける画像の参照をpictureGrayとする．

```
% 2-D FFT 点数
sizeFft = [1024 1024];
% 2-D FFT の計算
freqChar2d = fftshift(...
  fft2(double(pictureGray),...
  sizeFft(1),sizeFft(2)));
% 振幅特性の表示（累乗則変換を利用）
figure(2);
magnitude = abs(freqChar2d);
imshow(...
```

（**a**）原画像



（**b**）振幅特性（$\omega \in [-\pi, \pi)^2$）



（**c**）位相特性（$\omega \in [-\pi, \pi)^2$）

図8.5　静止画像のスペクトラム

```
    (magnitude/max(magnitude(:)))).^0.1);
% 位相特性の表示（0[rad] を中間輝度値とする）
figure(3);
phase = angle(freqChar2d);
imshow((phase+pi)/(2*pi));
```

図8.5（**b**），（**c**）に，それぞれ振幅特性と位相特性を示す．上記の例では，480 × 704の画像に対して，1024 × 1024の2次元FFTを施している．ここでは`fft2`関数と`fftshift`関数を利用した．

実習8.2　**静止画像のスペクトラム**

M-file：practice08_2.m

さまざまな静止画像の2次元スペクトラムを確認してみよう．

例題8.8　**動画像のスペクトラム**

映像ファイルcalcio.aviを読み込み，振幅特性を確認してみよう．

解

MATLAB上におけるコマンド例を以下に示そう．

```
% 3-D FFT 点数（高さ，幅，時間）
sizeFft = [256 256 128];
% 映像データの読み込み
fileName = './data/calcio.avi';
nFrames = 128;
clear array3d;
w = hanning(nFrames);
for iFrame = 1:nFrames
  frameCur = aviread(fileName,iFrame);
  pictureGray = rgb2graycq(frameCur.cdata);
  array3d(:,:,iFrame) = ...
    w(iFrame)*double(pictureGray)/255.0;
end
clear frameSeq pictureGray;
% 3-D FFT の計算
freqChar3d = ...
  abs(fftshift(fftn(array3d, sizeFft)));
clear array3d;
% 3-D スペクトラムの表示（累乗則変換を利用）
freqChar3d = freqChar3d/max(freqChar3d(:));
freqChar3d = freqChar3d.^0.1;
[wt,wh,wv] = meshgrid( ...
  -1:2/sizeFft(3):1-2/sizeFft(3),...
  -1:2/sizeFft(1):1-2/sizeFft(1),...
  -1:2/sizeFft(2):1-2/sizeFft(2));
freqChar3d = permute(freqChar3d,[2 3 1]);
```

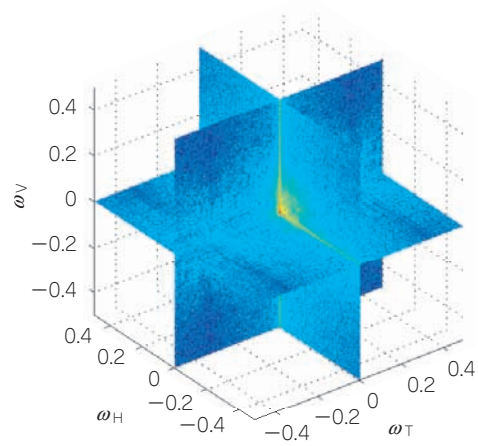


図8.6　動画像のスペクトラム

```
slice(wt,wh,wv,freqChar3d, ...
  0 ,0, 0, 'nearest');
shading flat;
axis vis3d;
colormap(jet);
xlabel('\omega_T (\times\pi rad)');
ylabel('\omega_H (\times\pi rad)');
zlabel('\omega_V (\times\pi rad)');
clear freqChar3d wt wh wv;
```

図8.6に結果を示す．時間方向のみハニング窓を適用した．MATLABの3-D回転機能を利用して，いろいろな方向からこのスペクトラムを観測してみよう．

実習8.3　**動画像のスペクトラム**

M-file：practice08_3.m

さまざまな動画像の3次元スペクトラムを見てみよう[注8-1]．

● 周波数応答

線形シフト不変システムは，インパルス応答によってその性質が決定される．このインパルス応

注8-1：大きなサイズや長時間の映像の場合，使用メモリを節約するために縮小処理やフレーム・レート変換を施した窓関数などを利用してデータの切り出しを行ったりする必要がある．このとき，周波数の帯域や分解能への影響に注意する必要がある．

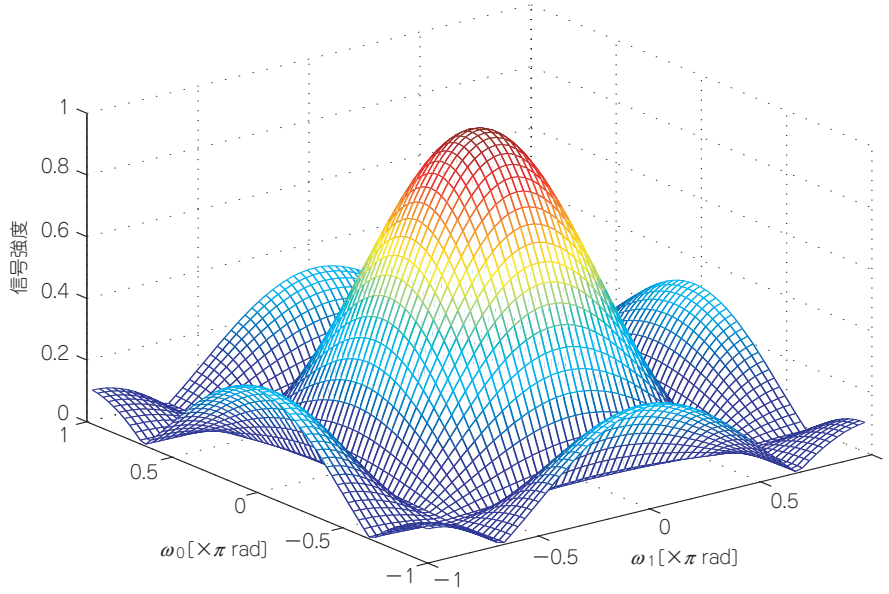


図8.7　平滑化フィルタの周波数応答

答の離散空間フーリエ変換(DSFT)もまた，その特性を知る上で重要である．これをシステムの周波数応答と呼び，また，その振幅を振幅応答，位相を位相応答と呼ぶ．

例題8.9　**平滑化フィルタの周波数応答**

次のインパルス応答$h[\mathbf{n}]$をもつ2次元平滑化フィルタの周波数応答を求めて，グラフ表示してみよう．

$$h[{}^{-1}_{-1}]=\frac{1}{9},h[{}^{-1}_{0}]=\frac{1}{9},h[{}^{-1}_{1}]=\frac{1}{9},h[{}^{0}_{-1}]=\frac{1}{9},h[{}^{0}_{0}]=\frac{1}{9},h[{}^{0}_{1}]=\frac{1}{9},h[{}^{1}_{-1}]=\frac{1}{9},h[{}^{1}_{0}]=\frac{1}{9},h[{}^{1}_{1}]=\frac{1}{9}$$

そのほか，$h[\mathbf{n}]=0$.

解

以下に，MATLAB上でのコマンド例を示そう．

```
% インパルス応答
h = ones(3)/9.0;
% 2-D FFT の計算
sizeFft = [64 64];
ampRes = abs(fftshift(fft2(h,sizeFft(1),sizeFft(2))));
% 周波数応答の表示
[fx,fy] = freqspace(sizeFft,'meshgrid');
mesh(fx,fy,ampRes);
```

```
    xlabel('\omega_1 (\times\pi rad)');
    ylabel('\omega_0 (\times\pi rad)');
    zlabel('Magnitude');
```

図8.7に，$h[\mathbf{n}]$の振幅応答を示す．なお，この特殊な例においては，周波数応答を

$$H(e^{j\omega^T}) = \frac{1}{9}(2\cos\omega_0 + 1)(2\cos\omega_1 + 1)$$

のように解析的に導くことも可能である．

なお，Image Processing Toolboxがある場合には，2次元振幅応答の表示に `freqz2` 関数を利用できる．

実習8.4　**各種フィルタの周波数応答**

M-file：practice08_4.m

そのほかの加重平均フィルタや4近傍および8近傍のラプラシアン・フィルタ，輪郭強調フィルタの周波数応答を求めて，グラフ表示してみよう．

8.4　標本化定理の再確認

1次元アナログ信号の標本化間隔Tによる標本化は，周波数領域における周期$1/T$(角周波数では$2\pi/T$)のスペクトラムの周期化を意味した．多次元における標本化格子LAT($\mathbf{V}$)による標本化もまた，周波数領域におけるスペクトラムの周期化を意味する．

● 標本化とスペクトラム

標本化は周波数領域で何を引き起こすのか？ 標本列を連続信号としてフーリエ変換(FT)を行ってみよう．まず準備として，任意の連続関数$\phi(\mathbf{p})$を

$$\int_{\mathbf{p}\in R^D} \delta(\mathbf{t})\phi(\mathbf{p})d\mathbf{p} = \phi(\mathbf{0}) \quad \cdots\cdots (8.29)$$

のように$\mathbf{p} = \mathbf{0}$のときの値$\phi(\mathbf{0})$と関連付ける連続δ関数を導入する．関数$\delta(\mathbf{p})$を**多次元連続δ関数**と呼ぶ．

すると，連続信号$x(\mathbf{p})$の標本化行列$\mathbf{V}$による標本化は

$$x_{\mathbf{V}}(\mathbf{p}) = x(\mathbf{p})\cdot\left(\sum_{\mathbf{n}\in\mathbb{Z}^D}\delta(\mathbf{p}-\mathbf{V}\mathbf{n})\right) \quad \cdots\cdots (8.30)$$

のように表現される．ここでは，標本化後の信号$x_{\mathbf{V}}(\mathbf{p})$もまた連続変数をもつことに注意していただきたい．途中の計算は割愛するが，この信号のフーリエ変換$X_{\mathbf{U}}(j\boldsymbol{\Omega})$は

$$X_{\mathbf{U}}(j\Omega) = \frac{1}{|\det\mathbf{V}|}\sum_{\mathbf{k}\in\mathbb{Z}^D} X(j(\Omega - \mathbf{U}\mathbf{k})) \quad \cdots\cdots (8.31)$$

のように導出される．ここで，$|\det\mathbf{V}|$は，標本化行列$\mathbf{V}$の列ベクトルが生成する平行超平面体の体積であり，1次元の標本化間隔Tに相当する．また，$X(j\boldsymbol{\Omega})$は，元の連続信号$x(\mathbf{p})$のフーリエ変換である．最も注目すべき点は，$X_{\mathbf{U}}(j\boldsymbol{\Omega})$が，周期$\mathbf{U}$の周期信号となる点である．そしてこの周期行列$\mathbf{U}$は標本化行列$\mathbf{V}$によって

$$\mathbf{U} = 2\pi\mathbf{V}^{-T} \quad \cdots\cdots (8.32)$$

と与えられる．

例題8.10　標本化行列と周期行列

図8.8(a)の周波数サポート領域をもつ2次元のアナログ信号$X(j\boldsymbol{\Omega})$を次の標本化行列

$$\mathbf{V_R} = \begin{pmatrix} P_0 & 0 \\ 0 & P_1 \end{pmatrix},\ \mathbf{V_Q} = \begin{pmatrix} P_0 & \frac{P_0}{2} \\ 0 & P_1 \end{pmatrix}$$

でそれぞれ標本化することを考えよう．この標本化後の信号がもつ周波数スペクトラムのサポート領域をそれぞれ描いてみよう．

解

標本化行列$\mathbf{V}_R$，$\mathbf{V}_Q$のそれぞれに対応する周期行列$\mathbf{U}_R$，$\mathbf{U}_Q$は，式(8.32)より，

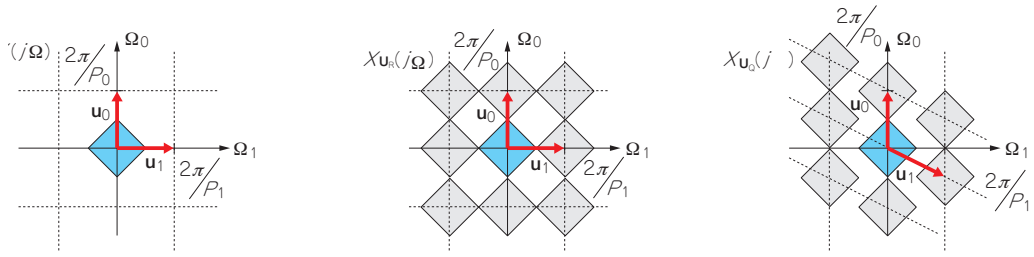
$$\mathbf{U_R} = 2\pi\begin{pmatrix} \frac{1}{P_0} & 0 \\ 0 & \frac{1}{P_1} \end{pmatrix},\ \mathbf{U_Q} = 2\pi\begin{pmatrix} \frac{1}{P_0} & 0 \\ -\frac{1}{2P_1} & \frac{1}{P_1} \end{pmatrix}$$

のように与えれる．よって，周波数スペクトラムのサポート領域は，それぞれ**図8.8(b)**，**(c)**のように描かれる．

(1) FTとDSFTの関係

標本化後の信号のフーリエ変換(FT)とDSFTの間には

$$X(e^{j\omega T}) = X_{\mathbf{U}}(j\Omega)\big|_{\Omega=\mathbf{U}\omega/2\pi} \quad \cdots\cdots (8.33)$$



(a) 2次元アナログ信号の周波数サポート領域

(b) 周波数スペクトラムのサポート領域(1)

(c) 周波数スペクトラムのサポート領域(2)

図8.8　標本化行列と周期行列

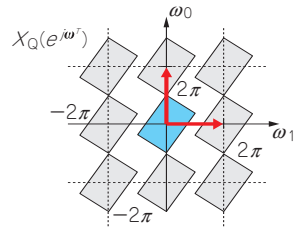


図8.9　非方形標本化とDSFT

のような密接な関係がある．DSFTの変数ωは，FTの周期$\mathbf{U}$を$2\pi\mathbf{I}$に正規化した角周波数と見なせる．ここで注意すべき点は，非方形標本格子による標本化の場合，DSFTの周波数スペクトラム形状がひずんで見える点である．

例題8.11　**非方形標本化とDSFT**

図8.8(c)に対応するDSFTの周波数サポート領域を描いてみよう．

解

周期行列$\mathbf{U}_Q$による周期を$2\pi\mathbf{I}$に正規化するため，**図8.9**のように描かれる．

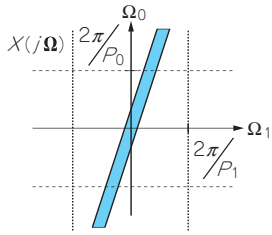
非方形標本化信号のスペクトラムをDSFT(DFT)を用いて観測する場合は，注意が必要である．

(2) エリアジング

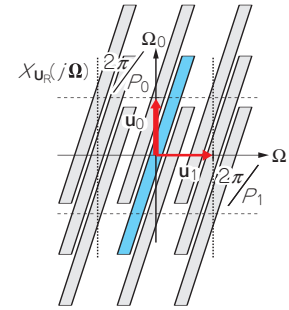
多次元標本化では，1次元標本化の標本化定理を単純に拡張して適用するわけにはいかない．以下は，多次元フィルタを設計する上で非常に重要な事実を例示している．

例題8.12　**多次元標本化の寛容性**

図8.10(a)の周波数サポート領域をもつ2次元のアナログ信号$X(j\mathbf{\Omega})$を例題8.10の標本化行列$\mathbf{V}_R$で標本化することを考える．この標本化後の信号における周波数スペクトラムのサポート領域を



(a) 周波数サポート領域をもつ2次元アナログ信号



(b) 標本化後の信号における周波数スペクトラムのサポート領域

図8.10　多次元標本化の寛容性

描いてみよう．

解

図8.10(b)に結果を示す．

1次元の標本化の考え方を適用すれば，Ω_0の軸についてはまったくエリアジング回避の条件を満たしていない．にもかかわらず，エリアジングをまったく起こすことなく周期化されていることが分かる．すなわち，基本スペクトラム(**図8.10(b)**の中央の薄青色部)を取り出せるフィルタが存在すれば，元のアナログ信号を完全に復元することができる．

● 大域的定速移動モデルのスペクトラム

図8.10(b)に示すような状況が起こりうる身近な例として，簡単な映像のモデルとその周波数特性について紹介しよう．

図8.11(a)に示すように，時刻$t=0$におけるフレーム画像を$x_0(\mathbf{p})$とし，このフレーム全体が一定の速度$\mathbf{v}=(v_0, v_1)^T$で移動する大域的定速移動モデル

$$x(\begin{matrix}\mathbf{p}\\ t\end{matrix}) = x_0(\mathbf{p}-\mathbf{v}t) \qquad \mathbf{p}\in R^2, t\in R \quad \cdots\cdots (8.34)$$

を考える．

$x_0(\mathbf{p})$のフーリエ変換を$X_0(j\boldsymbol{\Omega})$とすると，$x(\begin{matrix}\mathbf{p}\\ t\end{matrix})$のフーリエ変換$X(\begin{matrix}j\boldsymbol{\Omega}\\ j\Omega_t\end{matrix})$は

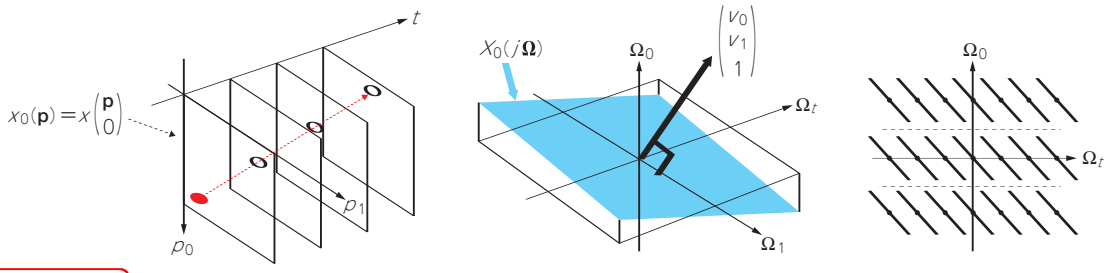
$$X(\begin{matrix}j\boldsymbol{\Omega}\\ j\Omega_t\end{matrix}) = X_0(j\Omega)\cdot 2\pi\delta(\boldsymbol{\Omega}^T\mathbf{v}+\Omega_t) \quad \cdots\cdots (8.35)$$

のように導出される．上式が非ゼロとなる条件が，

$$(\Omega_0, \Omega_1, \Omega_t)\begin{pmatrix} v_0 \\ v_1 \\ 1 \end{pmatrix} = 0 \quad \cdots\cdots (8.36)$$

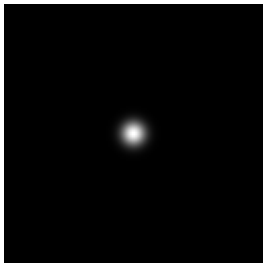
より，**図8.11(b)**に示すようにベクトル$(v_0, v_1, 1)^T$と直交する平面のみに$X_0(j\boldsymbol{\Omega})$が存在するようなスペクトラムが得られる．

では，このような信号を標本化した場合を考えよう．垂直-時間周波数平面上のスペクトラムの例を**図8.11(c)**に示す．垂直方向に動きがある場合，空間的に帯域制限がなされていれば，スペクトラムが傾いても時間方向の標本化周期には影響を与えないことが分かる．むしろ，時間方向に帯域を

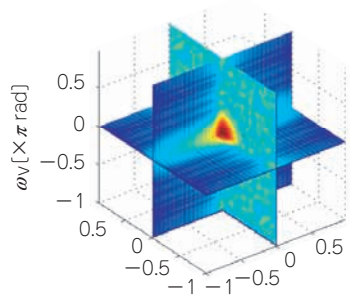


(**a**) 大域的定速移動モデル　(**b**) 周波数スペクトラム(1)　(**c**) 周波数スペクトラム(2)

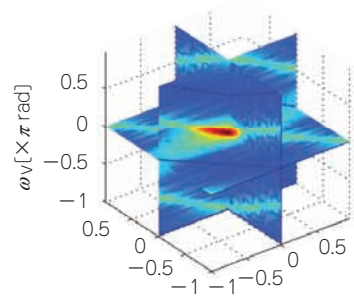
図8.11 大域的定速移動モデルのスペクトラム



(a) フレーム画像



(b) 静止状態の周波数スペクトラム



(c) 動いている状態の周波数スペクトラムの例

図8.12　大域的定速移動モデルのスペクトラムの例

制限してしまうと，空間解像度が犠牲になる．動きの激しい映像でフレーム間平均処理を施すとどうのような映像が出力されるかを想像してほしい．

例題8.13　**大域的定速移動モデルのスペクトラム**

図8.12(a)に示す画像を一定の速度$(v_0, v_1)^T = (2, 1)^T$［画素/フレーム］で動かした動画像を作成し，その周波数スペクトラムを確認してみよう．なお，画像には空間において周期性を仮定しよう．

解

以下に，MATLAB上でのコマンド例を示そう．

```
% 速度ベクトル（垂直，水平）
velocity = [2 1];
% 第0 番目フレームの読み込み
pictureOrg = double(imread('./data/gaussian2d.tif'))/255.0;
% 動画像の生成
nFrames = 32;
picture = pictureOrg;
figure(1);
w = hanning(nFrames);
for iFrame = 1:nFrames
  array3d(:,:,iFrame) = w(iFrame)*picture;
  picture = circshift(picture, velocity);
  imshowcq(picture);
  drawnow;
end
```


図8.12(b)に静止している状態，**図8.12(c)**に動いている状態の周波数スペクトラムを示す．画

像が動くとスペクトラムが斜めに傾くことを確認できる．ここでは，128 × 128 × 128 の 3 次元 FFT を使用した．周波数スペクトラムの表示については，例題8.8を参照されたい．

実習8.5　大域的定速移動モデルのスペクトラム

M-file：practice08_5.m

大域的定速移動モデルの速度(方向，速さ)を変えてスペクトラムを観察してみよう．

8.5　多次元 Z 変換

1次元信号処理と同じように，多次元信号処理においても Z 変換が定義され，システムの解析や実現に大変有用である．

● 単位インパルス信号の重み付け表現

多次元信号もまた単位インパルス信号の重み付け和として表現される．

例題8.14　単位インパルス信号の重み付け表現

次の2次元配列をインパルス信号の重み付けで表現してみよう．

$$\cdots, x\begin{bmatrix}0\\0\end{bmatrix}=3,\ x\begin{bmatrix}1\\0\end{bmatrix}=4,\ x\begin{bmatrix}2\\0\end{bmatrix}=5, \cdots$$

$$\cdots, x\begin{bmatrix}0\\1\end{bmatrix}=6,\ x\begin{bmatrix}1\\1\end{bmatrix}=7,\ x\begin{bmatrix}2\\1\end{bmatrix}=8, \cdots$$

解

次のように表現される．

$$x[\mathbf{n}]=\cdots+3\delta\left[\mathbf{n}-\begin{bmatrix}0\\0\end{bmatrix}\right]+4\delta\left[\mathbf{n}-\begin{bmatrix}1\\0\end{bmatrix}\right]+5\delta\left[\mathbf{n}-\begin{bmatrix}2\\0\end{bmatrix}\right]+$$
$$\cdots+6\delta\left[\mathbf{n}-\begin{bmatrix}0\\1\end{bmatrix}\right]+7\delta\left[\mathbf{n}-\begin{bmatrix}1\\1\end{bmatrix}\right]+8\delta\left[\mathbf{n}-\begin{bmatrix}2\\1\end{bmatrix}\right]\cdots$$

一般的には，

$$x[\mathbf{n}]=\sum_{\mathbf{k}\in Z^D} x[\mathbf{k}]\delta[\mathbf{n}-\mathbf{k}]$$

のように表現される．

● 多次元 Z 変換の定義

多次元の Z 変換は

$$X(\mathbf{z})=\sum_{\mathbf{n}\in Z^D} x[\mathbf{n}]\mathbf{z}^{-\mathbf{n}},\ \mathbf{z}\in C^D \quad \cdots\cdots(8.37)$$

と定義される．ただし，ベクトル $\mathbf{z}$ のベクトル $\mathbf{m}$ 乗は，

$$\mathbf{z}^{\mathbf{m}}=z_0^{m_0}z_1^{m_1}\cdots z_{D-1}^{m_{D-1}},\ \mathbf{z}\in C^D,\ \mathbf{m}\in Z^D \quad \cdots\cdots(8.38)$$

のように要素ごとのべき乗の積とする．C^D は，D 次元複素ベクトル集合を意味している．式(8.37)の関係は単純に

$$Z\{x[\mathbf{n}]\}=X(\mathbf{z}) \quad \cdots\cdots(8.39)$$

とも表現される．**表8.1**に多次元Z変換の主な性質をまとめる．

例題8.15　単位インパルス信号のZ変換

$\mathbf{k}$だけシフトしたインパルス信号$\delta[\mathbf{n}-\mathbf{k}]$のZ変換を求めてみよう．

解

Z変換の定義と単位インパルス信号の性質より，

$$Z\{\delta[\mathbf{n}-\mathbf{k}]\}=\sum_{\mathbf{n}\in Z^D}\delta[\mathbf{n}-\mathbf{k}]\mathbf{z}^{-\mathbf{n}}=\mathbf{z}^{-\mathbf{k}}$$

例題8.16　信号のZ変換

例題8.14の配列をZ変換してみよう．

解

例題8.15の結果と**表8.1**の線形性より，

$$X(z)=\cdots+3\mathbf{z}^{-\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]}+4\mathbf{z}^{-\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]}+5\mathbf{z}^{-\left[\begin{smallmatrix}2\\0\end{smallmatrix}\right]}+\cdots+6\mathbf{z}^{-\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]}+7\mathbf{z}^{-\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]}+8\mathbf{z}^{-\left[\begin{smallmatrix}2\\1\end{smallmatrix}\right]}+\cdots$$
$$=\cdots+3+4z_0^{-1}+\ 5z_0^{-2}+\cdots+6z_1^{-1}+7z_0^{-1}z_1^{-1}+8z_0^{-2}z_1^{-1}+\cdots$$

のように表現される．

(1) 逆Z変換

逆Z変換はコーシーの積分定理を用いて

$$x[\begin{smallmatrix}n_0\\n_1\end{smallmatrix}]=\frac{1}{(j2\pi)^2}\oint_{C_0}\oint_{C_1}X(\begin{smallmatrix}z_0\\z_1\end{smallmatrix})z_0^{n_0-1}z_1^{n_1-1}dz_0dz_1 \quad \cdots\cdots(8.40)$$

と導出される．ただし，簡単化のため，2次元の場合について表示している．ここでC_0，C_1は，原点を囲み，収束領域内で回る積分閉路である．

1次元の逆Z変換と異なり，多次元では部分分数展開を利用できない場合が多く，解析的な計算はほとんど行われない．

(2) Z変換とDSFTの関係

変数$\mathbf{z}=e^{j\omega T}$と置き換えることは，関数を

$$X(e^{j\omega T})=X(\mathbf{z})\big|_{\mathbf{z}=e^{j\omega T}} \quad \cdots\cdots(8.41)$$

のように評価していることを意味する．結果として，離散空間フーリエ変換(DSFT)が与えられる．従って，**表8.1**の$\mathbf{z}$を$e^{j\omega T}$と置き換えることで，DSFTの性質を表すこととなる．

表8.1
多次元Z変換の性質(a，bは定数)

性　質	時空間領域	Z領域
線形性	$ax[\mathbf{n}]+by[\mathbf{n}]$	$aX(\mathbf{z})+bY(\mathbf{z})$
時空間シフト	$x[\mathbf{n}-\mathbf{k}]$	$\mathbf{z}^{-\mathbf{k}}X(\mathbf{z})$
周波数シフト	$e^{j\omega_0^T\mathbf{n}}x[\mathbf{n}]$	$X(\mathbf{z}e^{-j\omega_0^T})$
畳み込み	$x[\mathbf{n}]*y[\mathbf{n}]$	$X(\mathbf{z})Y(\mathbf{z})$

● システムの伝達関数

線形シフト不変システム$T\{\cdot\}$の性質は，そのインパルス応答$h[\mathbf{n}] = T\{\delta[\mathbf{n}]\}$によって決定される．このインパルス応答の$Z$変換，$H(\mathbf{z}) = Z\{h[\mathbf{n}]\}$もまた，システムの解析や実現の上で有用である．この$H(\mathbf{z})$は，1次元の場合と同じようにシステムの伝達関数と呼ばれる．**表8.1**の畳み込みの性質から，信号$X(\mathbf{z})$に対するシステム$H(\mathbf{z})$の応答信号$Y(\mathbf{z})$は，$H(\mathbf{z})X(\mathbf{z})$のように多変量多項式の積で与えられる．

多次元の伝達関数における$\mathbf{z}^{-\mathbf{n}}$は遅延素子に相当するが，1次元に比べてその実現コストが非常に大きくなる点に注意を要する．例えば，水平走査された画像を処理する場合，水平方向の遅延と垂直方向の遅延では，同じ一つの遅延でもコストがまったく異なる．垂直の遅延一つは水平遅延器，すなわち記憶素子を画像の幅の数だけ必要とする．さらに時間遅延は，1画面分の記憶素子（フレーム・バッファ）を必要とする．

例題8.17　**フレーム差分処理**

時間的に隣り合うフレーム間の差分を出力するシステムの伝達関数を求めてみよう．

解

映像信号$x[\mathbf{n}_k]$に対するシステムの応答が，

$$y[\mathbf{n}_k] = T\{x[\mathbf{n}_k]\} = x[\mathbf{n}_k] - x[\mathbf{n}_{k-1}]$$

となればよい．$\mathbf{z} = (z_0, z_1, z_T)^T$とし，**表8.1**の時空間シフトの性質を用いると，システムの入出力関係が

$$Y(\mathbf{z}) = X(\mathbf{z}) - \mathbf{z}^{-\begin{bmatrix}0\\0\\1\end{bmatrix}}X(\mathbf{z}) = (1 - z_T^{-1})X(\mathbf{z})$$

のように求まる．よって，

$$H(z) = \frac{Y(z)}{X(z)} = 1 - \mathbf{z}_T^{-1}$$

章末問題

問題8.1　**標本化行列の非一意性**

標本化行列

$$V = \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix}$$

によって与えられる標本化格子LAT($\mathbf{V}$)を描いてみよう．任意のユニモジュラ整数行列$\mathbf{E}$を右から掛けた

$$\hat{\mathbf{V}} = \mathbf{VE}$$

によって与えられる標本化格子LAT($\hat{\mathbf{V}}$)も描き，前者と比較してみよう．ユニモジュラ整数行列$\mathbf{E}$に

ついては，任意の整数$\alpha \in Z$を用いて

$$\mathbf{E} = \begin{bmatrix} 1 & \alpha \\ 0 & \pm 1 \end{bmatrix}$$

と与えることができる．

問題8.2　**2次元信号の周波数**

2次元信号$x(\mathbf{p}) = \cos^2(4\pi p_0 + 2\pi p_1)$を2次元複素正弦波$e^{j\boldsymbol{\Omega}^T\mathbf{p}}$，$\boldsymbol{\Omega} \in R^2$の線形和で表現してみよう．ただし，$\mathbf{p} = (p_0, p_1)^T$とする．

問題8.3　**基本周期内の要素数**

任意の周期行列$\mathbf{N}$において，基本周期内のベクトル集合$\mathbb{N}(\mathbf{N})$の要素数が，$\lfloor |\det\mathbf{N}| \rfloor$で与えられることを確かめてみよう．ただし，$\lfloor \cdot \rfloor$は，小数点以下切り捨てによる整数化を意味する．

問題8.4　**静止画像の周波数特性**（MATLAB演習）

任意の画像$x[\mathbf{n}]$に対してDFTを施し，周波数特性$X[\mathbf{k}]$を求めよう．そして，振幅特性$|X[\mathbf{k}]|$をそのままに，位相特性をすべてゼロ（$\angle X[\mathbf{k}] = 0$）として逆DFTを施し，画像を復元してみよう．また，位相特性$\angle X[\mathbf{k}]$をそのままに，振幅特性をすべて一定（$|X[\mathbf{k}]| =$一定）として逆DFTを施し，画像を復元してみよう．振幅特性と位相特性ではどちらが原画像の面影を残すか，比較してみよう．

問題8.5　**先鋭化フィルタの周波数応答**

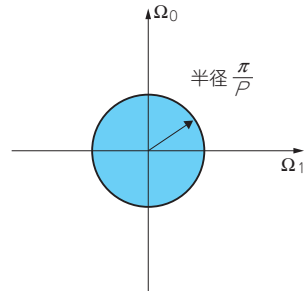
図5.7（**a**）に示す4近傍ラプラシアン・フィルタと，**図5.10**（**b**）に示す4近傍高域強調フィルタの周波数応答を解析的に導いてみよう．

問題8.6　**2次元標本化**

標本化行列

$$V = \begin{pmatrix} P & P \\ \frac{P}{\sqrt{3}} & \frac{P}{\sqrt{3}} \end{pmatrix}$$

により標本化された信号が，フーリエ変換（FT）領域でもつ周期を周期行列$\mathbf{U}$で表現してみよう．また，原画像が下図のように原点を中心とする半径π/Pの円で与えられる周波数サポート領域をもつと仮定して，標本化後の周波数特性を描いてみよう．



問題8.7　大域的定速移動モデル（MATLAB演習）

例題8.13の大域的定速移動モデルの映像に対して，2フレーム間の移動平均フィルタを時間方向に施してみよう．また，その周波数スペクトラムを確認しよう．

問題8.8　多次元Z変換の性質

表8.1の多次元Z変換の各性質を証明してみよう．

問題8.9　システムの伝達関数

伝達関数

$$H(\mathbf{z}) = \frac{1}{2}\left(1+\mathbf{z}^{-\begin{bmatrix}0\\0\\1\end{bmatrix}}\right)+\frac{1}{4}\left(\mathbf{z}^{\begin{bmatrix}0\\1\\0\end{bmatrix}}+\mathbf{z}^{-\begin{bmatrix}0\\1\\0\end{bmatrix}}\right)-\frac{1}{4}\left(\mathbf{z}^{\begin{bmatrix}1\\0\\0\end{bmatrix}}+\mathbf{z}^{-\begin{bmatrix}1\\0\\0\end{bmatrix}}\right)$$

をもつ3次元線形シフト不変システムについて，$\omega^T = (0, 0, 0), (\pi, 0, 0), (0, \pi, 0), (0, 0, \pi)$，および$(\pi, \pi, \pi)$における周波数応答$H(e^{j\boldsymbol{\omega}^T})$を求めてみよう．

参考文献

(1) 吹抜敬彦；TV画像の多次元信号処理，日刊工業新聞社，1988年．
(2) D. E. Dudgeon and R. M. Mersereau；*Multidimensional Digital Signal Processing*，Prentice Hall，1983.
(3) Yao Wan, J orn Ostermann Ya-Qin Zhang；*Video Processing and Communications*，Prentice Hall，2001.
(4) John, W. Woods；*Multidimensional Signal, Image and Video Processing and Coding*，Academic Press，2006.

第9章

多次元線形システム設計

本章では，多次元線形フィルタ，特にFIRフィルタの設計法をいくつか紹介する．また，1次元マルチレート信号処理の議論を多次元に拡張し，さまざまなサンプリング格子を扱う多次元マルチレート信号処理の構成要素について解説しよう．多次元マルチレート信号処理においては，多次元フィルタに加えて，多次元ダウン・サンプラと多次元アップ・サンプラが基本的な構成要素となる．

9.1　多次元フィルタ設計

以下では，多次元線形フィルタの具体的な設計法について，プログラム例を交えながら紹介しよう．まず，準備としてFIRフィルタとIIRフィルタについて復習しておこう．

● FIRフィルタとIIRフィルタ

FIR(有限インパルス応答)フィルタは，インパルス応答のサポート領域が有限の範囲に収まるフィルタであり，IIR(無限インパルス応答)フィルタは，インパルス応答のサポート領域が無限に広がるフィルタである．IIRフィルタに対するFIRフィルタの特徴を以下にまとめる[注9-1]．

- 常に安定である．IIRフィルタは係数によっては不安定となる．
- 量子化誤差の影響が少ない．IIRフィルタはフィードバックによる累積誤差が問題となる．
- 所望の振幅特性を得るためのフィルタ次数がIIRフィルタより高くなり，フィルタリングのための計算量が多い．

(1) システムの安定性

線形シフト不変システムの安定性の条件の一つに「BIBO安定」がある．BIBO安定のシステムでは，有界な入力$|x[\mathbf{n}]| < \infty$に対し，有界な出力$|y[\mathbf{n}]| < \infty$を得る．このBIBO安定の必要十分条件は，システムのインパルス応答$h[\mathbf{n}]$によって

注9-1：川又政征，樋口龍雄 著，「多次元ディジタル信号処理」(1995年，朝倉書店 刊)を参照．

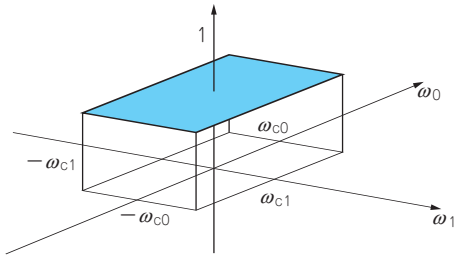


図 9.1　方形理想低域通過フィルタの振幅応答

$$\sum_{\mathbf{n}\in Z^D} |h[\mathbf{n}]| < \infty \quad \cdots\cdots (9.1)$$

のように表現できる．有限な値の係数をもつFIRシステムは，常に安定している．

(2) Shanksの安定定理

分母のみが多項式となっている特殊な2次元システム

$$H(\mathbf{z}) = \frac{1}{D(\mathbf{z})} = \sum_{n\in Z^2} h[\mathbf{n}]\mathbf{z}^{-\mathbf{n}} \quad \cdots\cdots (9.2)$$

が安定であるための必要十分条件は，$|z_0| > 1$かつ$|z_1| > 1$の任意の$\mathbf{z} = (z_0, z_1)^T$に対して，

$$D(\mathbf{z}) \neq 0 \quad \cdots\cdots (9.3)$$

となることである．分母分離型($D(\mathbf{z})$が可分離)の場合，2次元の安定判別は1次元の問題に帰着する．一方で，分子多項式$N(\mathbf{z})$のある一般的なIIRフィルタ

$$H(\mathbf{z}) = \frac{N(\mathbf{z})}{D(\mathbf{z})} \quad \cdots\cdots (9.4)$$

の場合，分子多項式$N(\mathbf{z})$が安定性に影響するため，その安定判別は容易ではない．

● フィルタ仕様

具体的なフィルタ設計法について紹介する前に，設計手順および理想フィルタについてまとめておこう．

(1) 設計手順

フィルタ設計においては，まず，設計仕様を与える．設計仕様には，所望の特性や実現規模などがある．次に，所望の特性を近似することで実現可能な伝達関数を求める．最後に，フィルタ構造を決定し，ソフトウェアあるいはハードウェアとして実現する．この手続きのうち，近似の作業が最も重要である．

次に，設計仕様を与える際の所望特性として，しばしば利用される方形理想低域通過フィルタと円対称理想低域通過フィルタを紹介しよう．

(2) 方形理想低域通過フィルタ

図9.1に2次元方形理想低域通過フィルタの振幅応答を示す．ゼロ位相を仮定すると，多次元の方形理想低域通過フィルタの周波数応答は，任意の正の実数ω_{cd}に対して，

$$H_{\mathbf{I}}(e^{j\omega^T}) = \begin{cases} 1 & |\omega_d| \le \omega_{cd} \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (9.5)$$

と定義される．また，インパルス応答は，周波数応答の逆離散空間フーリエ変換（IDSFT）として

$$h_{\mathbf{I}}[\mathbf{n}] = \frac{1}{(2\pi)^D} \int_{\omega \in [-\omega_{cd},\, \omega_{cd})^D} e^{j\omega^T \mathbf{n}} d\omega = \frac{\sin \omega_{c0} n_0}{\pi n_0} \cdot \frac{\sin \omega_{c1} n_1}{\pi n_1} \mathrm{L}\, \frac{\sin \omega_{cD-1} n_{D-1}}{\pi n_{D-1}} = \prod_{d=0}^{D-1} \frac{\omega_{cd}}{\pi} \mathrm{sinc}\left(\frac{\omega_{cd}}{\pi} n_d \right) \quad \cdots\cdots (9.6)$$

のように与えられる．ここで，$\omega = (\omega_0, \omega_1, \mathrm{L}, \omega_{D-1})^T$である．

例題9.1　**方形理想低域通過フィルタ**

$\omega_{c0} = \omega_{c1} = \pi/3$として，2次元の方形理想低域通過フィルタのインパルス応答を描いてみよう．

解

以下に，MATLAB上でのコマンド例を示そう．

```
% カットオフ周波数の設定
fc0 = 1/3; % 垂直(×π rad)
fc1 = 1/3; % 水平(×π rad)
% 表示範囲の設定
region = 20;
% インパルス応答の計算
[iHorizontal,iVertical] = ...
  meshgrid(-region:region,-region:region);
r0 = (fc0)*sinc((fc0)*iVertical);
r1 = (fc1)*sinc((fc1)*iHorizontal);
idealFilter = r0 .* r1;
% インパルス応答の表示
mesh(iHorizontal,iVertical,idealFilter);
xlabel('n_1');
ylabel('n_0');
```

図9.2にMATLABによる計算結果を示す．ただし，インパルス応答が無限に広がるため，原点を中心とした周辺のみの応答を示している．

実習9.1　**方形理想低域通過フィルタ**

M-file：practice09_1.m

カットオフ周波数ω_{c0}，ω_{c1}を変えたときのインパルス応答を確認してみよう．

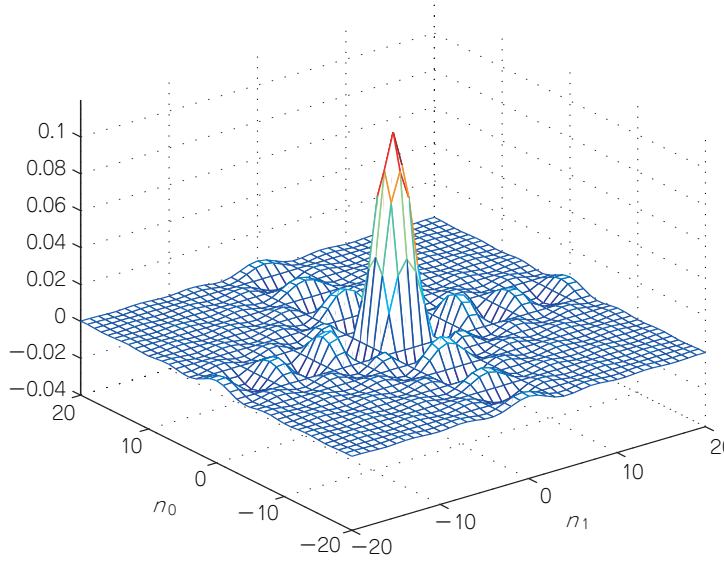


図9.2　方形理想低域通過フィルタのインパルス応答の例

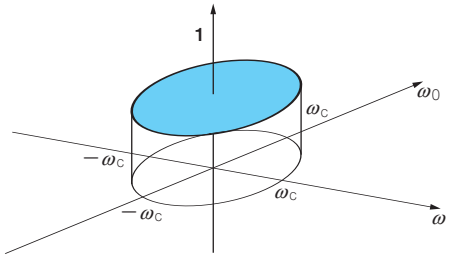


図9.3　円対称理想低域通過フィルタの振幅応答

(3) 円対称理想低域通過フィルタ

図9.3に円対称理想低域通過フィルタの振幅応答を示す．ゼロ位相を仮定すると，円対称理想低域通過フィルタの周波数応答は

$$H_{\mathbf{I}}(e^{j\omega^T}) = \begin{cases} 1 & \omega_0^2+\omega_1^2 \leqq \omega_c^2 \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (9.7)$$

のように定義される．ここでは，議論を2次元の場合に限定する．インパルス応答は，周波数応答の逆離散空間フーリエ変換(IDSFT)として

$$h_{\mathbf{I}}[\mathbf{n}] = \frac{1}{(2\pi)^2}\int_{\omega_0^2+\omega_1^2\leq\omega_c^2} e^{j\omega^T\mathbf{n}}d\omega = \frac{1}{(2\pi)^2}\int_0^{\omega_c}\int_0^{2\pi}\omega e^{j\omega\sqrt{n_0^2+n_1^2}\cos(\theta-\phi)}d\phi d\omega = \frac{1}{2\pi}\int_0^{\omega_c}\omega J_0\left(\omega\sqrt{n_0^2+n_1^2}\right)d\omega$$

$$= \frac{\omega_c}{2\pi}\cdot\frac{J_1\left(\omega_c\sqrt{n_0^2+n_1^2}\right)}{\sqrt{n_0^2+n_1^2}} \qquad \mathbf{n}\neq\mathbf{0} \quad \cdots\cdots (9.8)$$

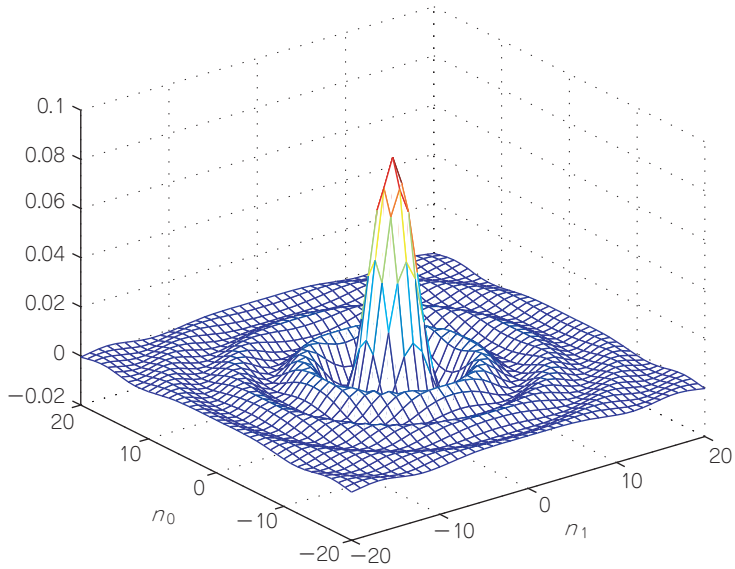


図9.4　円対称理想低域通過フィルタのインパルス応答の例

$$h_{\mathrm{I}}[\mathbf{0}] = \frac{\omega_{\mathrm{c}}}{2\pi} \cdot \frac{\omega_{\mathrm{c}}}{2} \quad \cdots\cdots (9.9)$$

のように与えられる．ここで，$J_n(x)$は，n次の第1種Bessel関数である．

例題9.2　**円対称理想低域通過フィルタ**

$\omega_c = \pi/3$として，円対称理想低域通過フィルタのインパルス応答を描いてみよう．

解

以下に，MATLAB上でのコマンド例を示そう．

```
% カットオフ周波数の設定
fc = 1/3; % (×π rad)
% 表示範囲の設定
region = 20;
% インパルス応答の計算
[iHorizontal,iVertical] = ...
  meshgrid(-region:region,-region:region);
r = sqrt(iHorizontal.^2+iVertical.^2);
iOrg = find(r==0);
r(iOrg) = 1;
idealFilter = (fc/2)*besselj(1,pi*fc*r)./r;
```

```
    idealFilter(iOrg) = pi*fc^2/4;
    % インパルス応答の表示
    mesh(iHorizontal,iVertical,idealFilter);
    xlabel('n_1');
    ylabel('n_0');
```

図9.4に結果を示す．ただし，インパルス応答が無限に広がるため，原点を中心とした周辺のみの応答を示している．

実習9.2　**円対称理想低域通過フィルタ**

M-file：practice09_2.m

カットオフ周波数ω_cを変えたときのインパルス応答を確認してみよう．

● 近似仕様

所望の特性の近似を行うためには**近似仕様**を定めなければならない．近似仕様を与える対象の例を以下にまとめる．

- インパルス応答：$h[\mathbf{n}]$
- 周波数応答：$H(e^{j\boldsymbol{\omega}^T})$
  - 振幅応答：$|H(e^{j\boldsymbol{\omega}^T})|$
  - 位相応答：$\angle H(e^{j\boldsymbol{\omega}^T})$

もしくは，群遅延特性：$-\dfrac{\partial \angle H(e^{j\omega^T})}{\partial \omega_d}$

(1) 振幅応答

2次元の場合について，振幅特性の通過域を設定する際の典型的な近似仕様を**図9.5**に示す．また，低域通過型フィルタの近似仕様を与える際の領域設定の例を**図9.6**に示す．**図9.6**において，R_pは**通過域**を意味しており，この領域内では許容誤差δ_pを用いて次のように振幅応答が設定される．

$$1-\delta_\mathrm{p} \le |H(e^{j\omega^T})| \le 1+\delta_\mathrm{p},\ \omega \in R_\mathrm{p} \quad \cdots\cdots (9.10)$$

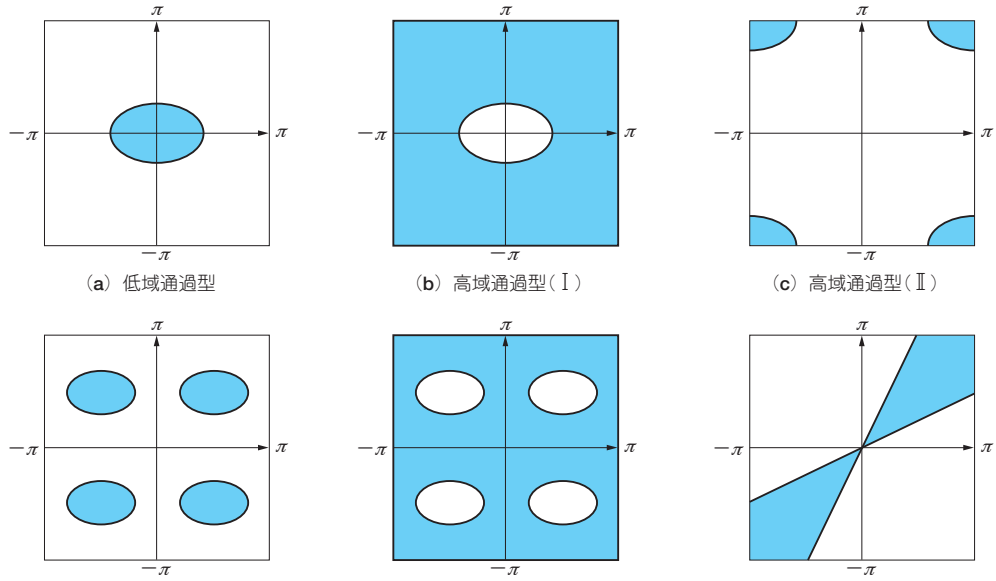
また，R_sは**阻止域**を意味しており，この領域内では許容誤差δ_sを用いて次のように振幅応答が設定される．

$$|H(e^{j\omega^T})| \le \delta_\mathrm{s},\ \omega \in R_\mathrm{s} \quad \cdots\cdots (9.11)$$

最後に，R_tは**遷移域**を意味しており，これは振幅応答の仕様を設定しない領域である．

(2) 位相応答

位相応答に与えられる仕様として，例えば線形位相特性がある．画像処理の応用では，振幅ひずみに比べて位相ひずみの方が知覚しやすく，線形位相特性は音声信号処理に比べて重要な性質である．線形位相フィルタでも特にゼロ位相フィルタが画像処理において頻繁に利用される．インパルス応答$h[\mathbf{n}]$が



(a) 低域通過型　(b) 高域通過型(Ⅰ)　(c) 高域通過型(Ⅱ)

(d) 帯域通過型　(e) 帯域阻止型　(f) ファン通過型

図9.5　典型的な通過域の仕様

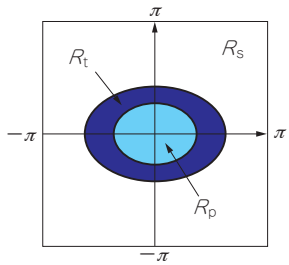


図9.6　典型的な近似仕様

$$h[\mathbf{n}] = h[-\mathbf{n}],\ \mathbf{n} \in Z^D \quad \cdots\cdots (9.12)$$

のように原点を中心として対称性を持てばゼロ位相特性となる．この周波数応答は，

$$H(e^{j\omega^T}) = \sum_{\mathbf{n}\in Z^D} h[\mathbf{n}]e^{-j\omega^T\mathbf{n}} = h[\mathbf{0}] + \sum_{\mathbf{n}\in Z^D,\,\mathbf{n}\neq\mathbf{0},\,n_0\geq 0} h[\mathbf{n}](e^{j\omega^T\mathbf{n}} + e^{-j\omega^T\mathbf{n}}) = h[\mathbf{0}] + \sum_{\mathbf{n}\in Z^D,\,\mathbf{n}\neq\mathbf{0},\,n_0\geq 0} h[\mathbf{n}]2\cos\omega^T\mathbf{n} \quad \cdots\cdots (9.13)$$

のように実数となる．

9.2　FIRフィルタ設計

以下では，常に安定しているFIRフィルタの設計を中心に解説を進めよう．

● 分離設計法

多次元フィルタを可分離として設定すると，そのインパルス応答$h[\mathbf{n}]$，および周波数応答$H(e^{j\omega^T})$は，次式のように各次元に対応する1次元フィルタの積として与えられる．

$$h[\mathbf{n}] = \prod_{d=0}^{D-1} h_d[n_d] \quad \cdots\cdots (9.14)$$

$$H(e^{j\omega^T}) = \prod_{d=0}^{D-1} H_d(e^{j\omega_d}) \quad \cdots\cdots (9.15)$$

従って，可分離の多次元フィルタ設計は各次元の1次元フィルタの設計に帰着できる．それぞれに近似仕様を与えることで，数多く存在する設計手法を直接適用可能である．

各1次元フィルタが線形位相特性をもつ場合，多次元フィルタもまた線形位相特性をもつ．ただし，1次元で与えた振幅応答の仕様が，多次元において直接対応しないことに注意する．例えば2次元フィルタの場合について説明しよう．各次元の通過域誤差をδ_{pd}，阻止域誤差をδ_{sd}とすると，与えられる多次元フィルタの通過域の誤差は，

$$\delta_p = (1+\delta_{p0})(1+\delta_{p1}) - 1 \quad \cdots\cdots (9.16)$$

阻止域の誤差は，

$$\delta_s = \max(\delta_{s0}(1+\delta_{p1}), (1+\delta_{p0})\delta_{s1}) \quad \cdots\cdots (9.17)$$

となる．

例題9.3　可分離フィルタ設計

以下の近似仕様を満たす2次元可分離フィルタを設計しよう．

- 通過域：$|\omega_d| < \pi/3 - \pi/10$，$d = 0, 1$
- 通過域許容誤差：$\delta_p = 0.01$
- 阻止域：$|\omega_d| > \pi/3 + \pi/10$，$d = 0, 1$
- 阻止域許容誤差：$\delta_s = 0.01$

解

各次元のフィルタの通過域許容誤差を$\delta_{pd} = \sqrt{1+\delta_p} - 1 \fallingdotseq 0.00499$，阻止域許容誤差を$\delta_{sd} = \delta_s/(1+\delta_{p(1-d)}) \fallingdotseq 0.00995$と設定すればよい．この仕様から，firpm関数で設計することを前提とすると，firpmord関数より1次元フィルタの近似次数21が求められる．タップ数が奇数となるように次数を22（タップ数23）と設定しよう．以下に，MATLAB上でのコマンド例を示す．

```
% カットオフ周波数
fc0 = 1/3; % 垂直(×π rad)
fc1 = 1/3; % 水平(×π rad)
% 通過域の仕様
passEdge0 = fc0-1/10; % 垂直(×π rad)
passEdge1 = fc1-1/10; % 水平(×π rad)
```

```
% 阻止域の仕様
stopEdge0 = fc0+1/10; % 垂直(×π rad)
stopEdge1 = fc1+1/10; % 水平(×π rad)
% 許容誤差の仕様
deltaPass = 0.01;
deltaStop = 0.01;
% 各次元の許容誤差の仕様
deltaPass0 = sqrt(1+deltaPass)-1;
deltaStop0 = deltaStop/(1+deltaPass0);
deltaPass1 = sqrt(1+deltaPass)-1;
deltaStop1 = deltaStop/(1+deltaPass1);
% フィルタH_0 の近似仕様
specFrq0 = [passEdge0 stopEdge0];
specAmp0 = [1 0];
specErr0 = [deltaPass0 deltaStop0];
spec0 = firpmord(specFrq0,specAmp0,specErr0,2,'cell');
% フィルタのタップ数が奇数となるように修正
if (mod(spec0{1},2)~=0)
  spec0{1}=spec0{1}+1;
end
halfOrder0 = spec0{1}/2;
% フィルタH_1 の近似仕様
specFrq1 = [passEdge1 stopEdge1];
specAmp1 = [1 0];
specErr1 = [deltaPass1 deltaStop1];
spec1 = firpmord(specFrq1,specAmp1,specErr1,2,'cell');
% フィルタのタップ数が奇数となるように修正
if (mod(spec1{1},2)~=0)
  spec1{1}=spec1{1}+1;
end
halfOrder1 = spec1{1}/2;
% 各次元のフィルタ設計
oneDimensionalFilter0 = firpm(spec0{:});
oneDimensionalFilter1 = firpm(spec1{:});
% フィルタ特性の表示
figure(1); impz(oneDimensionalFilter0);
```

```
figure(2); freqz(oneDimensionalFilter0);
figure(3); impz(oneDimensionalFilter1);
figure(4); freqz(oneDimensionalFilter1);
% 可分離2次元フィルタ
twoDimensionalFilter = ...
  kron(oneDimensionalFilter1(:).',...
    oneDimensionalFilter0(:));
figure(5);
mesh(-halfOrder1:halfOrder1,...
    -halfOrder0:halfOrder0,...
    twoDimensionalFilter);
figure(6);
freqz2(twoDimensionalFilter);
xlabel('\omega_1 (\times \pi rad)');
ylabel('\omega_0 (\times \pi rad)');
```

図9.7(**a**)，(**b**)に設計した1次元フィルタのインパルス応答と振幅応答を示し，**図9.7**(**c**)，(**d**)に2次元フィルタのインパルス応答と振幅応答を示す．

Image Processing Toolboxがない場合には `freqz2` 関数を利用できないため，代わりに本書で用意した `freqz2cq` 関数を利用してほしい．

実習9.3　**可分離フィルタの設計**

M-file：practice09_3.m

設計仕様を変えてさまざまな可分離2次元フィルタを設計してみよう．

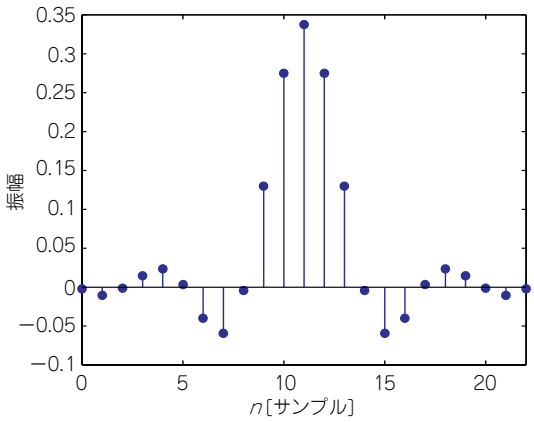
● 窓関数法

先に示した方形理想低域通過フィルタや円対称理想低域通過フィルタは無限に広がるインパルス応答をもつため実現不可能である．そこで，窓関数法では，インパルス応答に窓関数を掛け，有限な範囲で打ち切ることで，理想フィルタを近似したFIRフィルタを設計する．以下では，**分離型**と**対照型**の二つの窓関数法について解説しよう．

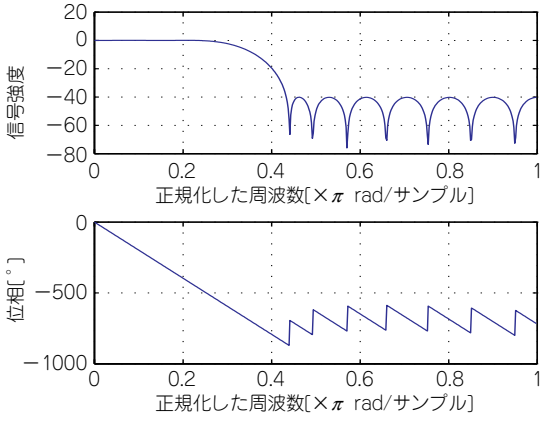
(1) 分離型

分離型は，分離可能な窓関数を用いて，次式のように理想フィルタのインパルス応答$h_{\mathrm{I}}[\mathbf{n}]$を有限で打ち切る．

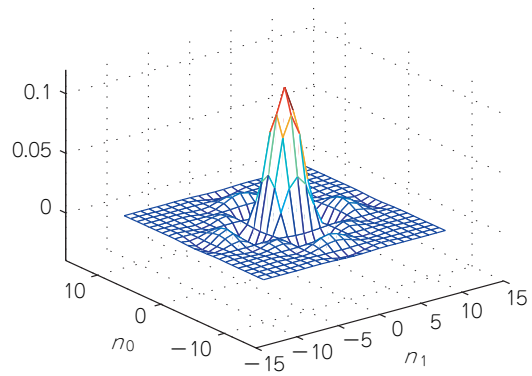
$$h[\mathbf{n}] = \left(\prod_{d=0}^{D-1} W_{N_d}(n_d) \right) \cdot h_{\mathrm{I}}[\mathbf{n}]$$



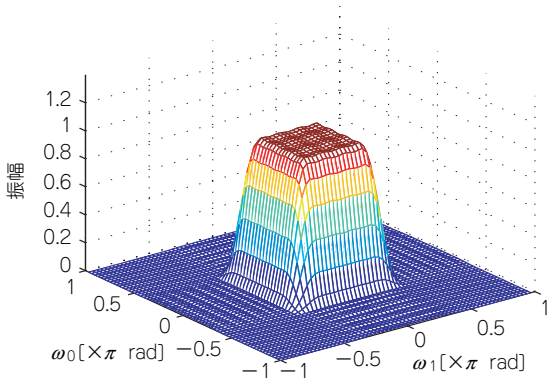
(a) 1次元インパルス応答



(b) 1次元周波数応答



(c) 2次元インパルス応答



(d) 2次元振幅応答

図9.7　可分離フィルタの設計例

ここで，$W_N(n)$は1次元の窓関数であり，Nは応答を打ち切る長さを決めるパラメータである．

例題9.4　カイザー窓による分離型設計

方形理想低域通過フィルタから，例題9.3と同じ近似仕様を満たす2次元フィルタを分離型窓関数法により設計しよう．

解

各次元の通過域許容誤差と阻止域許容誤差は例題9.3と同じように設定すればよい．この仕様から，MATLABのkaiserord関数を用いて窓関数の長さとβパラメータを計算すると，それぞれ27と4.093のように求められる．タップ数が奇数となるように，次数を28(タップ数29)と設定しよう．以下に，MATLAB上でのコマンド例を示す．なお，例題9.3の解答例と重複する部分については省略している．

```
% フィルタH_0 の近似仕様
[n0,Wn0,beta0,typ0] = kaiserord(...
  specFrq0,specAmp0,specErr0,2);
% フィルタのタップ数が奇数となるように修正
if (mod(n0,2)~=0)
  n0=n0+1;
end
halfOrder0 = n0/2;
% フィルタH_1 の近似仕様
[n1,Wn1,beta1,typ1] = kaiserord(...
  specFrq1,specAmp1,specErr1,2);
% フィルタのタップ数が奇数となるように修正
if (mod(n1,2)~=0)
  n1=n1+1;
end
halfOrder1 = n1/2;
% 各次元のフィルタ設計
oneDimensionalFilter0 = fir1(...
  n0,Wn0,typ0,kaiser(n0+1,beta0),'noscale');
oneDimensionalFilter1 = fir1(...
  n1,Wn1,typ1,kaiser(n1+1,beta1),'noscale');
% 可分離2次元フィルタ
twoDimensionalFilter = ...
  kron(oneDimensionalFilter1(:).',...
      oneDimensionalFilter0(:));
```

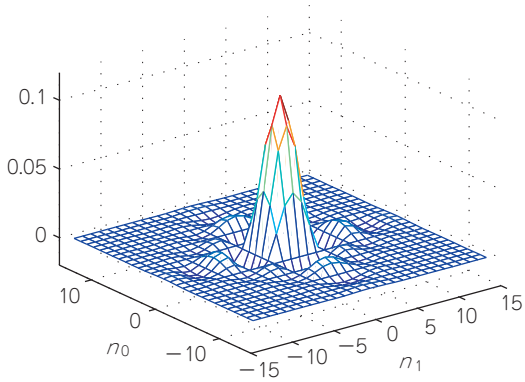
図9.8(**a**)，(**b**)に設計したフィルタのインパルス応答と振幅応答を示す．

Image Processing Toolbox の`fwind1`関数に所望の周波数応答，垂直方向1次元窓関数および水平方向1次元窓関数を与えると，この分離型窓関数法によって2次元フィルタを生成する．

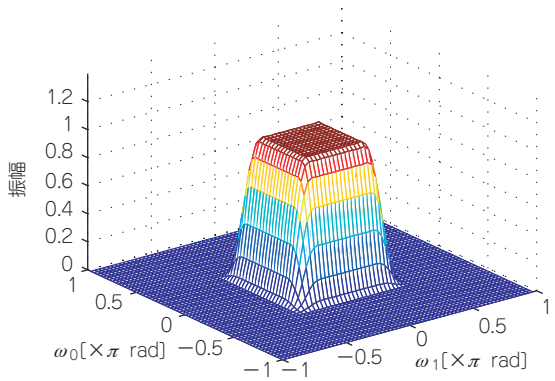
実習9.4　**分離型窓関数法**

M-file：practice09_4.m

近似仕様を変えて，さまざまなフィルタを設計してみよう．



(a) 2次元インパルス応答



(b) 2次元振幅応答

図9.8　分離型窓関数法によるフィルタの設計例

(2) 対称型

対称型では，1次元の窓関数を用いて次式のように理想フィルタのインパルス応答を有限に打ち切る．

$$h[\mathbf{n}] = W_N(\|\mathbf{n}\|)h_{\mathrm{I}}[\mathbf{n}]$$

ただし，$\|\mathbf{n}\|$は二乗ノルムであり，2次元の場合は$\|\mathbf{n}\| = \sqrt{n_0^2 + n_1^2}$となる．$N$は応答を打ち切る長さを決めるパラメータである．

例題9.5　カイザー窓による対称型設計

円対称理想低域通過フィルタから，次の近似仕様を満たす2次元フィルタを対称型窓関数法により設計しよう．

- 通過域：$\sqrt{\omega_0^2 + \omega_1^2} < \pi/3 - \pi/10$
- 通過域許容誤差：$\delta_{\mathrm{p}} = 0.01$
- 阻止域：$\sqrt{\omega_0^2 + \omega_1^2} > \pi/3 + \pi/10$
- 阻止域許容誤差：$\delta_{\mathrm{s}} = 0.01$

解

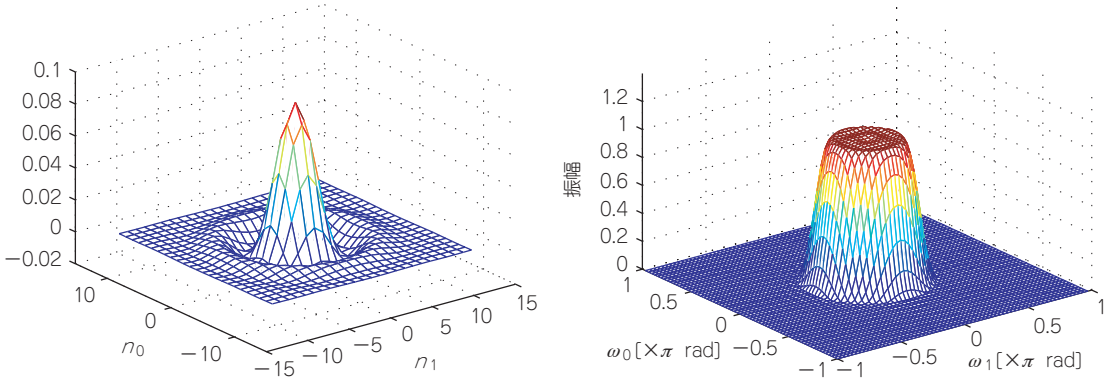
対称型の場合，仕様で与えられた通過域許容誤差と阻止域許容誤差を直接利用して窓関数の近似次数を求められる．この仕様から，MATLABの`kaiserord`関数などを用いて窓関数の長さとβパラメータを計算すると，それぞれ22と3.395のように求められる．タップ数が奇数となるように次数を24(タップ数25)と設定しよう．以下に，MATLAB上でのコマンド例を示す．

```
% カットオフ周波数
fc = 1/3; % (×π rad)
% 通過域の仕様
```

```
passEdge = fc-1/10; % (×π rad)
% 阻止域の仕様
stopEdge = fc+1/10; % (×π rad)
% 許容誤差の仕様
deltaPass = 0.01;
deltaStop = 0.01;
% フィルタH の近似仕様
specFrq = [passEdge stopEdge];
specAmp = [1 0];
specErr = [deltaPass deltaStop];
[n,Wn,beta,typ] = kaiserord(...
  specFrq,specAmp,specErr,2);
% 窓長が奇数となるように次数を偶数に修正
if (mod(n,2)~=0)
  n=n+1;
end
halfOrder = n/2;
% 窓関数と理想フィルタの設定
[iHorizontal,iVertical] = ...
  meshgrid(-halfOrder:halfOrder,...
          -halfOrder:halfOrder);
r = sqrt(iHorizontal.^2+iVertical.^2);
w = contKaiser(r,halfOrder,beta);
iOrg = find(r==0);
r(iOrg) = 1;
idealFilter = (fc/2)*besselj(1,pi*fc*r)./r;
idealFilter(iOrg) = pi*fc^2/4;
% 2次元フィルタの計算
twoDimensionalFilter = w .* idealFilter;
```

図9.9(**a**)，(**b**)に設計したフィルタのインパルス応答と振幅応答を示す．なお，上のコマンド例では次のcontKaiser関数

```
function y = contKaiser(x,halfOrder,beta);
%
if (abs(x) > halfOrder)
```



(**a**) 2次元インパルス応答　　(**b**) 2次元振幅応答

図9.9　対称型窓関数法によるフィルタの設計例

```
    y = 0;
  else
    y = besseli(0,beta*sqrt(1-(x/halfOrder).^2))/...
        besseli(0,beta);
  end
```

を利用している．

Image Processing Toolboxの `fwind1` 関数に所望の周波数応答と一つの1次元窓関数を与えると，この対照型窓関数法によって2次元フィルタを生成する．また，直接2次元の窓関数を与えてフィルタを設計する `fwind2` 関数も用意されている．

実習9.5　**対称型窓関数法**

M-file：practice09_5.m

近似仕様を変えて，さまざまなフィルタを設計してみよう．

● マクレラン変換法

マクレラン変換(McClellan Transform)によって，1次元FIRフィルタから多次元FIRフィルタを得ることができる．マクレラン変換の手続きは以下のとおりである．

1. 1次元ゼロ位相FIRフィルタを設計
2. 変数変換をとおして多次元に変換

特徴は以下のとおりである．

- 1次元フィルタの設計手法が直接役立つ
- 効果的な実現法が存在する

(1) 原理

では，原理について説明しよう．まず，タップ数が$2N+1$の1次元のゼロ位相FIRフィルタ$h[n]$を仮定する．このフィルタの周波数特性$H(e^{j\omega})$は，次式のように余弦波の1次結合として表される．

$$H(e^{j\omega}) = \sum_{n=0}^{N} a[n]\cos\omega n = \sum_{n=0}^{N} a[n]T_n\{\cos\omega\} \quad \cdots\cdots (9.18)$$

ただし，

$$a[n] = \begin{cases} h[0] & n = 0 \\ 2h[n] & n > 0 \end{cases} \quad \cdots\cdots (9.19)$$

式(9.18)における$T_n\{x\}$は**チェビシェフ多項式**と呼ばれ，$\cos\ \omega n$を$\cos\ \omega$のn次の多項式で表現することを可能とする．チェビシェフ多項式は，

$$\begin{aligned} T_0\{x\} &= 1 \\ T_1\{x\} &= x \\ T_2\{x\} &= 2x^2 - 1 \\ &\vdots \\ T_n\{x\} &= 2xT_{n-1}\{x\} - T_{n-2}\{x\} \end{aligned} \quad \cdots\cdots (9.20)$$

と与えられる．

次に，チェビシェフ多項式の変数となっている$\cos\ \omega$を多次元の周波数特性$F(e^{j\omega^T})$で置き換えてしまう．この変換によって与えられる多次元フィルタの周波数特性は，

$$H(e^{j\omega^T}) = \sum_{n=0}^{N} a[n]T_n\{F(e^{j\omega^T})\} \quad \cdots\cdots (9.21)$$

となる．

(2) 2次元フィルタ設計

2次元フィルタを得るための$F(e^{j\omega^T})$の例を以下に示そう．

$$F(\mathbf{z}) = A + \frac{1}{2}B(z_1^1 + z_1^{-1}) + \frac{1}{2}C(z_0^1 + z_0^{-1}) + \frac{1}{2}D(z_1^1 z_0^{-1} + z_1^{-1} z_0^1) + \frac{1}{2}E(z_1^1 z_0^1 + z_1^{-1} z_0^{-1}) \quad \cdots\cdots (9.22)$$

として，

$$F(e^{j\omega^T}) = F(\mathbf{z})\big|_{\mathbf{z}=e^{j\omega^T}} = A + B\cos\omega_1 + C\cos\omega_0 + D\cos(\omega_1 - \omega_0) + E\cos(\omega_1 + \omega_0) \quad \cdots\cdots (9.23)$$

を得る．ただし，$\mathbf{z} = (z_0, z_1)^T$，$\omega = (\omega_0, \omega_1)^T$である．

このフィルタのパラメータA，B，C，D，Eの選択によって疑似円対称低域通過フィルタやファン・フィルタの設計が可能となる．以下に，疑似円対称低域通過フィルタを設計する場合のパラメータの一例を示そう．

$$A = -\frac{1}{2}, B = C = \frac{1}{2}, D = E = \frac{1}{4} \quad \cdots\cdots (9.24)$$

この選択により，1次元ゼロ位相FIRフィルタの周波数応答$H(e^{j\omega})$が**図9.10**に示す等位線に沿って，2次元の周波数応答$H(e^{j\omega^T})$に写像される．

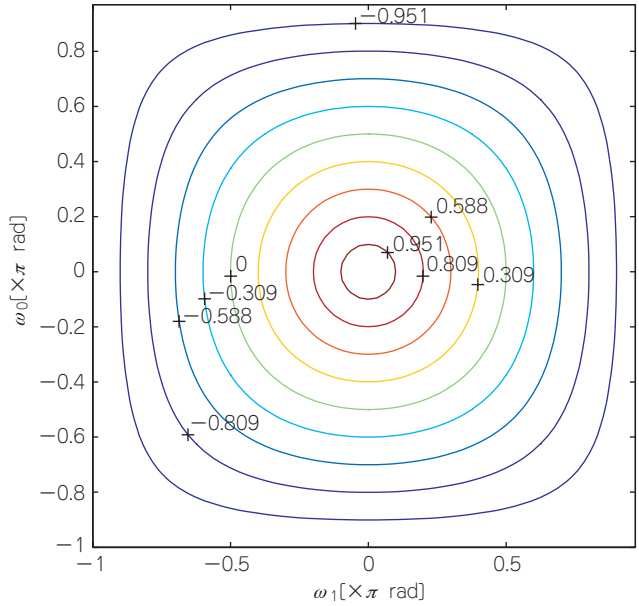


図9.10
マクレラン変換の等位線
（数値は$\cos\omega$の値）

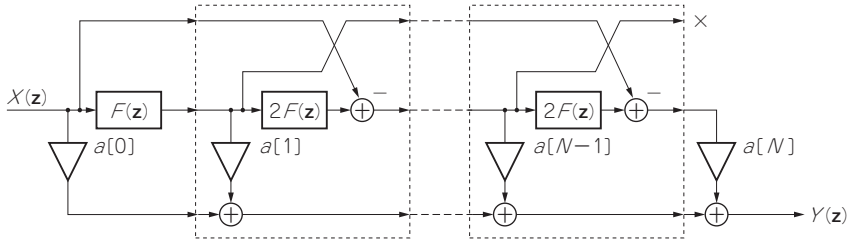


図9.11　マクレラン変換による非分離フィルタの効果的実現法

(3) 効果的実現法

マクレラン変換によって設計されたフィルタは非分離型のフィルタとなるが，**図9.11**に示すような効果的な構成によって，一般的な非分離フィルタよりも少ない演算量で実現可能である．以下に，マクレラン変換によって設計される非分離フィルタの効果的実現法をMATLAB関数として掲載しよう．

```
function y = ...
  mcClellanTransConv2(h, x, trans, shape);
% マクレラン変換フィルタの効果的実現
%
% y = mcClellanTransConv2(h,x,trans,shape)
%
```

```
% 入力
%     h :1次元線形位相原型フィルタ
%     x :入力2次元配列
%     trans : マクレラン変換係数
%     shape : conv2関数に直接渡される出力形状
%               (デフォルト'full')
%
% 出力
%     y :出力2次元配列
%
% 参考
%     conv2
%
% 1次元フィルタの折り返し係数計算
h = h(:);
halfOrder = (length(h)-1)/2;
a = [h(halfOrder+1); 2*h(halfOrder+2:end)];
% 初段の計算
delay = [0 0 0; 0 1 0; 0 0 0]; % 位相調整用
x0 = conv2(delay, x, shape);
x1 = conv2(trans, x, shape);      % F(z)
x2 = a(1)*conv2(delay, x, shape); % a[0]
% 2段目以降の計算（プライベート関数の呼び出し）
for i=2:halfOrder
  [x0, x1, x2] = ...
    mcClellanTransModule(...
      x0, x1, x2, trans, a(i), shape);
end
% 最終段の計算
y = a(halfOrder+1)*x1 + x2;
% プライベート関数：マクレラン変換モジュール
function [y0,y1,y2] = ...
  mcClellanTransModule(...
    x0, x1, x2, trans, a, shape);
%
delay = [0 0 0; 0 1 0; 0 0 0]; % 位相調整用
```

```
y0 = conv2(delay, x1, shape);
y1 = -conv2(delay, x0, shape)...
    + 2.0*conv2(trans,x1, shape); % 2F(z)
y2 = conv2(delay, (a*x1)+x2, shape);
```

例題9.6　マクレラン変換法

以下の近似仕様を満たす1次元ゼロ位相FIRフィルタからマクレラン変換によって2次元ゼロ位相FIRフィルタを設計しよう．

- 通過域：$|\omega| < \pi/3 - \pi/10$
- 通過域許容誤差：$\delta_p = 0.01$
- 阻止域：$|\omega| > \pi/3 + \pi/10$
- 阻止域許容誤差：$\delta_s = 0.01$

解

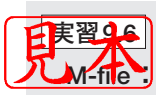
与えられた仕様から，firpmord関数を用いて1次元ゼロ位相FIRフィルタの近似次数を計算すると，19と求められる．タップ数が奇数となるように次数を20（タップ数21）と設定しよう．以下に，MATLAB上でのコマンド例を示す．

```
% 1次元ゼロ位相FIR フィルタの準備
%% カットオフ周波数
fc = 1/3; % (×π rad)
%% 通過域の仕様
passEdge = fc-1/10; % (×π rad)
%% 阻止域の仕様
stopEdge = fc+1/10; % (×π rad)
%% 許容誤差の仕様
deltaPass = 0.01;
deltaStop = 0.01;
%% 1次元フィルタの近似仕様
specFrq = [passEdge stopEdge];
specAmp = [1 0];
specErr = [deltaPass deltaStop];
spec = firpmord(...
  specFrq,specAmp,specErr,2,'cell');
% フィルタのタップ数を奇数に修正
if(mod(spec{1},2)~=0)
  spec{1}=spec{1}+1;
```

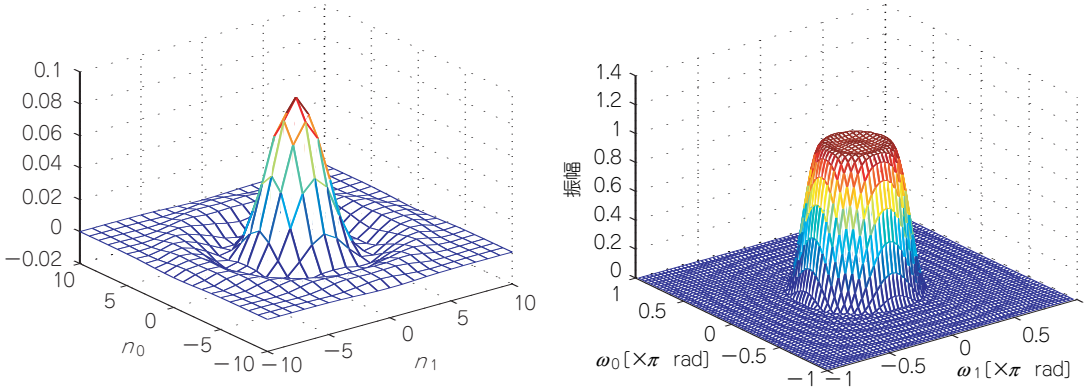
```
end
halfOrder = spec{1}/2;
h = firpm(spec{:});
% 疑似円対称低域通過フィルタ用パラメータ
params.A = -1/2;
params.B = 1/2;
params.C = 1/2;
params.D = 1/4;
params.E = 1/4;
% 等位線の計算
nPoints = 64;
[f1 f0] = freqspace(nPoints);
[w1 w0] = meshgrid(pi*f1,pi*f0);
g = mcClellanTrans(w1, w0, params);
% 等位線の表示
figure(1);
c = contour(...
  f1, f0, g, [cos(0.1*pi:0.1*pi:0.9*pi)]);
clabel(c);
xlabel('\omega_1 (\times \pi rad)');
ylabel('\omega_0 (\times \pi rad)');
pbaspect([1 1 1]);
% 2次元フィルタ設計
T = [ params.D params.B params.E;
      params.C 2*params.A params.C;
      params.E params.B params.D]/2;
delta = 1; % インパルス入力
twoDimensionalFilter = ...
  mcClellanTransConv2(h, delta, T, 'full');
```

図9.12に設計した2次元フィルタのインパルス応答と振幅応答を示す．

Image Processing Toolboxでは，マクレラン変換法を`ftrans2`関数によって提供している．



実習9.6　マクレラン変換設計
M-file：practice09_6.m



(a) 2次元インパルス応答　　(b) 2次元振幅応答

図9.12　マクレラン変換による疑似円対称低域通過フィルタの設計例

各種パラメータを変更し，マクレラン変換法を利用してさまざまな疑似円対称低域通過フィルタを設計してみよう．

例えば，1次元高域通過フィルタをもとに2次元フィルタを設計してみよう．

● 各種変換法

ここでは，**図9.5**(a)に示す低域通過フィルタから**図9.5**(b)～(d)に示すようなフィルタを設計する手法についてまとめておこう．ただし，便宜上$H(\mathbf{z})$は，通過域の利得が1で，ゼロ位相特性をもつ低域通過フィルタの伝達関数と仮定する．

(1) 高域通過型(Ⅰ)

高域通過型(Ⅰ)の伝達関数$F(\mathbf{z})$は，低域通過フィルタ$H(\mathbf{z})$の伝達関数より

$$F(\mathbf{z}) = 1 - H(\mathbf{z}) \quad \cdots\cdots (9.25)$$

と与えることができる．それぞれのインパルス応答は，

$$f[\mathbf{n}] = \delta[\mathbf{n}] - h[\mathbf{n}] \quad \cdots\cdots (9.26)$$

のように関係する．この変換により，通過域と阻止域が反転する．

(2) 高域通過型(Ⅱ)

高域通過型(Ⅱ)の伝達関数$F(\mathbf{z})$は，低域通過フィルタ$H(\mathbf{z})$の伝達関数より

$$F(\mathbf{z}) = H(-\mathbf{z}) \quad \cdots\cdots (9.27)$$

と与えることができる．それぞれのインパルス応答は，

$$f[\mathbf{n}] = (-\mathbf{1})^{\mathbf{n}} h[\mathbf{n}] \quad \cdots\cdots (9.28)$$

のように関係する．$(-\mathbf{1})$は，要素がすべて−1のベクトルである．2次元の場合，$(-\mathbf{1})^{\mathbf{n}} = (-1)^{n_0}(-1)^{n_1}$となる．この変換は，各次元独立に$\pi$変調を施すことにほかならない．

(3) 帯域通過型

高域通過型の伝達関数$F(\mathbf{z})$は，低域通過型フィルタ$H(\mathbf{z})$の伝達関数より

$$F(\mathbf{z}) = H(-\mathbf{z}^{2\mathbf{I}}) \quad \cdots\cdots (9.29)$$

と与えることができる．それぞれのインパルス応答は，

$$f[\mathbf{n}] = [(-1)^{\mathbf{n}} h[\mathbf{n}]]_{\uparrow 2\mathbf{I}} \quad \cdots\cdots (9.30)$$

のように関係する．ただし，$[x[\mathbf{n}]]_{\uparrow 2\mathbf{I}}$は，信号$x[\mathbf{n}]$に対して各次元独立に補間率2のアップ・サンプリング（ゼロ値挿入）を施すことを意味する．すなわち，この変換は各次元独立にπ変調を施し，その後，アップ・サンプリングすることにほかならない．

(4) 帯域阻止型

高域阻止型の伝達関数$F(\mathbf{z})$は，低域通過フィルタ$H(\mathbf{z})$の伝達関数より

$$F(\mathbf{z}) = 1 - H(-\mathbf{z}^{2\mathbf{I}}) \quad \cdots\cdots (9.31)$$

と与えることができる．それぞれのインパルス応答は，

$$f[\mathbf{n}] = \delta[\mathbf{n}] - [(-1)^{\mathbf{n}} h[\mathbf{n}]]_{\uparrow 2\mathbf{I}} \quad \cdots\cdots (9.32)$$

のように関係する．これは，帯域通過フィルタへの変換の後に，通過域と阻止域を反転することを意味する．

なお，本書で紹介した設計法のほかにも，周波数標本化法（Image Processing Toolboxの`fsamp2`関数）や最適化設計法など，1次元フィルタ設計と同じように行える多次元FIRフィルタ設計法が存在する．

9.3 多次元標本化格子変換

第6章では，画像の拡大・縮小処理について解説した．そこでは分離処理のみを扱っていたが，ここではより一般的な多次元標本化格子変換処理について解説しよう．まず，多次元標本化格子変換に欠かせない二つの基本要素である**多次元ダウン・サンプラ**と**多次元アップ・サンプラ**について紹介する．

● 多次元ダウン・サンプラ

以下では，多次元ダウン・サンプラについて，可分離処理と非分離処理に分けて解説する．

(1) 可分離処理

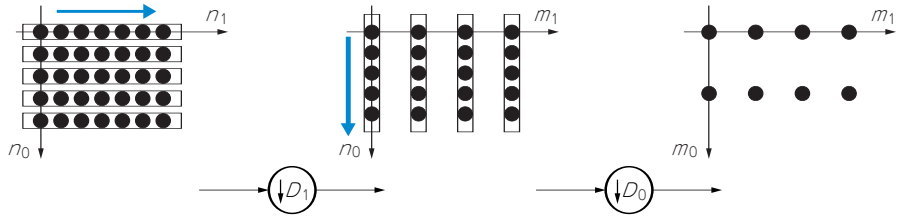
分離可能な多次元ダウン・サンプリングは，各次元独立に1次元のダウン・サンプリングを施すことで実現できる．**図9.13(a)**に，垂直方向，水平方向の間引き率がそれぞれ3と2の格子変換の様子を示す．D_0，D_1の値を変えることで，縮小率を変えることができる．また，3次元，4次元への拡張も容易である．

ここで，**図9.13(b)**のように多次元処理として再表現しよう．間引き率を行列$\mathbf{D}$として与えることで，多次元のダウン・サンプリングは，

$$y[\mathbf{m}] = x[\mathbf{Dm}] \quad \cdots\cdots (9.33)$$

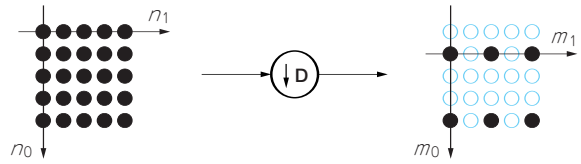
のようにも表現できる．行列$\mathbf{D}$を間引き行列と呼ぶ．

多次元ダウン・サンプリングは，**図9.13**に示されるような可分離処理で十分な場合が多い．しかしプログレッシブ映像とインターレース映像の間の変換や，ハニカム・センサから得られた六角



（a）1次元処理の縦続表示

$$x\begin{bmatrix} n_0 \\ n_1 \end{bmatrix} \qquad \mathbf{D}=\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \qquad y\begin{bmatrix} m_0 \\ m_1 \end{bmatrix}=x\begin{bmatrix} 3m_0 \\ 2m_1 \end{bmatrix}$$



（b）多次元処理表示

図9.13　2次元の可分離ダウン・サンプラ（ただし，$D_0 = 3$，$D_1 = 2$）

格子画像を方形格子に変換する場合など，非分離ダウン・サンプリングが必要となる重要な応用も少なくない．

例題9.7　**可分離ダウン・サンプリング**

3次元信号$x[\mathbf{n}]$に対して，各次元の間引き率を$D_0 = 2$，$D_1 = 3$，$D_2 = 4$として3次元ダウン・サンプリングを施したい．間引き行列$\mathbf{D}$を求めよ．

解

間引き行列$\mathbf{D}$，各次元の間引き率を対角要素としてもつ対角行列となる．よって，

$$\mathbf{D}=\begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

と与えられる．

(2) 非分離処理

式(9.33)において，間引き行列$\mathbf{D}$を対角行列以外に選ぶと，処理後の標本化格子はどのようになるのであろうか？ **図9.14**にその一例を示そう．

このダウン・サンプリングは，各次元n_0，n_1ごとに独立に施すことはできない．すなわち非分離のダウン・サンプリングとなる．$x[\mathbf{n}]$がアナログ信号$x(\mathbf{p})$を標本化行列$\mathbf{V}$で標本化したものとすると，

$$y[\mathbf{n}] = x[\mathbf{Dm}] = x(\mathbf{VDm}) \qquad (9.34)$$

より，$y[\mathbf{m}]$は$x(\mathbf{p})$を標本化行列$\mathbf{V}' = \mathbf{VD}$で標本化したものとみなせる．$\mathbf{V}$が対角行列で，標本化格子

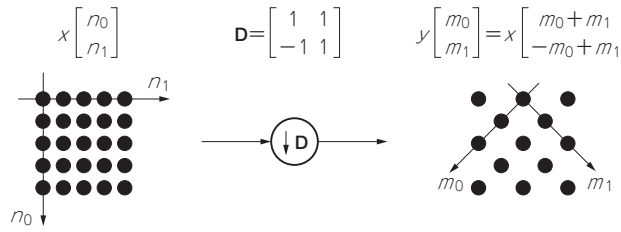


図9.14　2次元の非分離ダウン・サンプラ

が方形であったとしても，非分離ダウン・サンプリング後は，非方形の標本化格子となる．

より一般的には，間引き行列**D**として任意の非特異整数行列を選択することができる．また，間引き率Dは

$$D = |\det \mathbf{D}| \quad \cdots\cdots (9.35)$$

と与えられる．ここで，$|\det \mathbf{D}|$は**D**の行列式の絶対値を意味する．

例題9.8　**非分離ダウン・サンプリング**

図9.15の左の格子から右の格子を得るための間引き行列**D**を求めよ．なお，処理前と処理後の標本化行列**V**，**V**′はそれぞれ，

$$\mathbf{V} = \begin{pmatrix} P_0 & 0 & 0 \\ 0 & P_1 & 0 \\ 0 & 0 & \dfrac{T}{2} \end{pmatrix}, \mathbf{V}' = \begin{pmatrix} 2P_0 & 0 & P_0 \\ 0 & P_1 & 0 \\ 0 & 0 & \dfrac{T}{2} \end{pmatrix}$$

と与えられる．

解

$\mathbf{V}' = \mathbf{DV}$より，

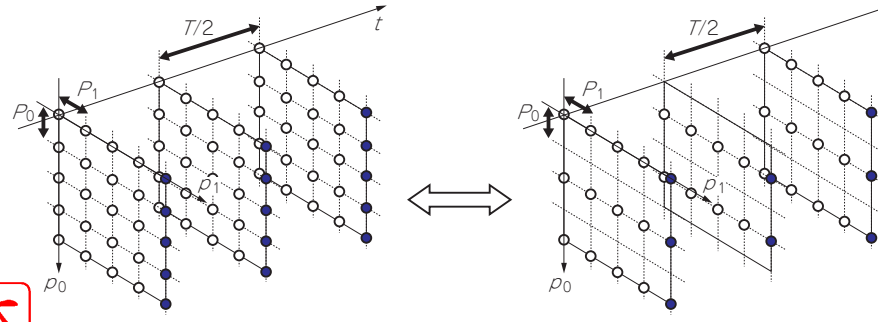


図9.15　3次元の非分離格子変換

$$\mathbf{D} = \mathbf{V}^{-1}\mathbf{V}' = \begin{pmatrix} \frac{1}{P_0} & 0 & 0 \\ 0 & \frac{1}{P_1} & 0 \\ 0 & 0 & \frac{2}{T} \end{pmatrix} \begin{pmatrix} 2P_0 & 0 & P_0 \\ 0 & P_1 & 0 \\ 0 & 0 & \frac{T}{2} \end{pmatrix} = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

なお，間引き率は$D = |\det \mathbf{D}| = 2$となる．このダウン・サンプリングはプログレッシブ映像からインターレース映像を得る変換に対応する．

● ダウン・サンプラと周波数スペクトラム

1次元の場合と同じように，多次元のダウン・サンプリングにおいても周波数スペクトラムの折り返し（エリアジング）によるひずみが問題となる．例題6.2で示した平均処理は，このエリアジングを回避する効果をもつ．多次元のダウン・サンプリングと周波数スペクトラムの関係を調べるために，まず，Z変換領域におけるダウン・サンプラの入出力関係について解説しよう．

例題9.9　ダウン・サンプラの入出力関係

図9.14のダウン・サンプラの入出力関係をZ変換領域で表現してみよう．

解

入力配列$x[\mathbf{n}]$のうち，$n_0 + n_1$が奇数となる，棄却される位置のサンプルをゼロ値に置換しよう．この配列$u[\mathbf{n}]$は，

$$u[\mathbf{n}] = \frac{1}{2}\{x[\mathbf{n}] + (-1)^{(n_0+n_1)}x[\mathbf{n}]\}$$

のように$x[\mathbf{n}]$とそれを変調した配列$e^{j\pi(n_0+n_1)}x[\mathbf{n}] = (-1)^{(n_0+n_1)}x[\mathbf{n}]$との平均操作で与えられる．$u[\mathbf{n}]$の$Z$変換$U(\mathbf{z})$は，

$$U(\mathbf{z}) = \frac{1}{2}\{X(\mathbf{z}) + X(-\mathbf{z})\}$$

のように表現される．結局，$u(\mathbf{n})$が$\mathbf{n} = \mathbf{Dm}$以外の位置でゼロ値をとるため，$X(\mathbf{z})$と$Y(\mathbf{z})$の関係を

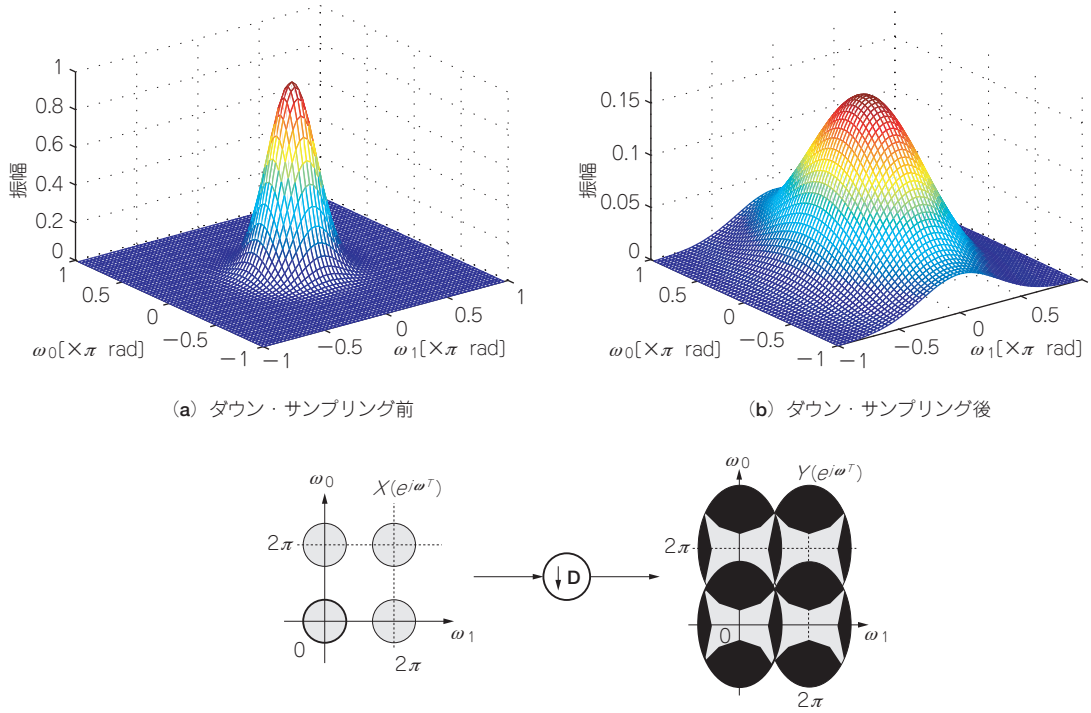
$$Y(\mathbf{z}) = \sum_{\mathbf{m}\in\mathbb{Z}^2} y[\mathbf{m}]\mathbf{z}^{-\mathbf{m}} = \sum_{\mathbf{m}\in\mathbb{Z}^2} u[\mathbf{Dm}]\mathbf{z}^{-\mathbf{m}} = \sum_{\mathbf{n}\in\mathbb{Z}^2} u[\mathbf{n}]\mathbf{z}^{-\mathbf{D}^{-1}\mathbf{n}}$$
$$= U(\mathbf{z}^{\mathbf{D}^{-1}}) = \frac{1}{2}\{X(\mathbf{z}^{\mathbf{D}^{-1}}) + X(-\mathbf{z}^{\mathbf{D}^{-1}})\}$$

のように導くことができる．

より一般的な間引き行列$\mathbf{D}$に対しては，$2\pi\mathbf{D}^{-T}\mathbf{k}$変調成分間の平均

$$U(\mathbf{z}) = \frac{1}{|\det \mathbf{D}|}\sum_{\mathbf{k}\in\mathbb{N}^{(\mathbf{D}^T)}} X(e^{-j2\pi\mathbf{k}^T\mathbf{D}^{-1}}\mathbf{z}) \quad \cdots\cdots (9.36)$$

によって棄却されるサンプルをゼロ値に置き換えることができる．この関係より



（a）ダウン・サンプリング前

（b）ダウン・サンプリング後

（c）スペクトラムの変化

図9.16　可分離ダウン・サンプリングの周波数スペクトル例

$$Y(\mathbf{z}) = \frac{1}{|\det \mathbf{D}|} \sum_{\mathbf{k} \in \mathcal{N}(\mathbf{D}^T)} X(e^{-j2\pi \mathbf{k}^T \mathbf{D}^{-1}} \mathbf{z}^{\mathbf{D}^{-1}}) \quad \cdots\cdots (9.37)$$

という関係が導かれる．ここで，$\mathcal{N}(\mathbf{D}^T)$は，$\mathbf{D}^T$の各列ベクトルで生成される平行超平面体内の整数ベクトル集合である（第8.2節を参照）．例題9.9においては，

$|\det \mathbf{D}| = 2$

$\mathcal{N}(\mathbf{D}^T) = \{[\begin{smallmatrix}0\\0\end{smallmatrix}], [\begin{smallmatrix}0\\1\end{smallmatrix}]\}$

$2\pi \mathbf{k}^T \mathbf{D}^{-1} = \pi \mathbf{k}^T [\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}]$

となる．

(1) 周波数スペクトラム

式(9.37)に$\mathbf{z} = e^{j\ T}$を代入すると離散空間フーリエ変換（DSFT）領域での入出力関係

$$Y(e^{j\omega^T}) = \frac{1}{|\det \mathbf{D}|} \sum_{\mathbf{k} \in \mathcal{N}(\mathbf{D}^T)} X(e^{j(\omega - 2\pi \mathbf{k})^T \mathbf{D}^{-1}}) \quad \cdots\cdots (9.38)$$

が導かれる．$\mathbf{k} = \mathbf{0}$以外の変調成分は，ダウン・サンプリングによって生じるエリアジング成分である．

例題9.10　可分離ダウン・サンプリング

図9.16(a)の周波数振幅特性をもつ2次元ガウス信号$X(e^{j\omega T})$に対して，垂直方向と水平方向の間

引き率がそれぞれ3と2のダウン・サンプリングを施し，その結果の周波数振幅特性を描いてみよう．

解

以下に，MATLAB上でのコマンド例を示そう．

```
% 垂直の間引き率
verticalDecFactor = 3;
% 水平の間引き率
horizontalDecFactor = 2;
% 配列の標準偏差
sigma = 2;
% 配列のサイズ
sizeX = 31;
% 2次元ガウス関数による配列の生成
arrayX = gaussian2cq(sizeX,sigma.^2);
% ダウン・サンプリング
arrayY = ...
  downsample(...
    downsample(arrayX,...
      verticalDecFactor).',...
        horizontalDecFactor).';
```

gaussian2cqは，2次元ガウス分布を生成する関数として本書で用意した．**図9.16(b)**にダウン・サンプル後の周波数振幅特性を示す．いま，間引き行列が$\mathbf{D}=\mathrm{diag}([3\ \ 2]^T)$と与えられるので，ダウン・サンプル前後の信号間の周波数スペクトラムは，

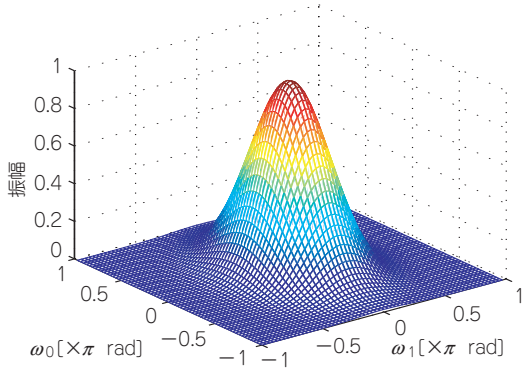
$$Y(e^{j\omega^T})=\frac{1}{6}\left\{X\left(e^{j\left[\frac{\omega_0}{3}\ \frac{\omega_1}{2}\right]^T}\right)+X\left(e^{j\left[\frac{\omega_0}{3}\ \frac{\omega_1}{2}-\pi\right]^T}\right)+X\left(e^{j\left[\frac{\omega_0}{3}-\frac{2\pi}{3}\ \frac{\omega_1}{2}\right]^T}\right)+X\left(e^{j\left[\frac{\omega_0}{3}-\frac{2\pi}{3}\ \frac{\omega_1}{2}-\pi\right]^T}\right)+X\left(e^{j\left[\frac{\omega_0}{3}-\frac{4\pi}{3}\ \frac{\omega_1}{2}\right]^T}\right)+X\left(e^{j\left[\frac{\omega_0}{3}-\frac{4\pi}{3}\ \frac{\omega_1}{2}-\pi\right]^T}\right)\right\}$$

という関係をもつ．**図9.16(c)**に示すように，元のスペクトラムが垂直方向に3倍，水平方向に2倍に膨張し，振幅は1/6となる．本例題の信号は適切な帯域制限が施されていないので，ダウン・サンプル後はエリアジング(折り返しひずみ)を受けてしまう．

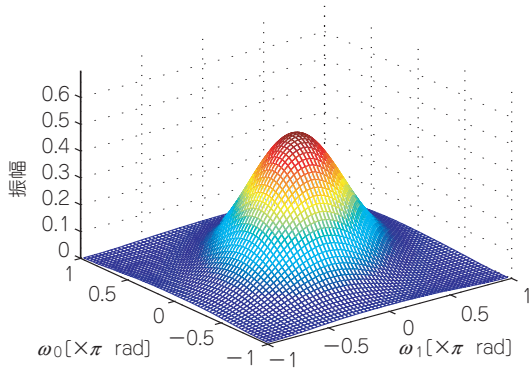
実習9.7　**ダウン・サンプルと周波数スペクトラム**

M-file：practice09_7.m

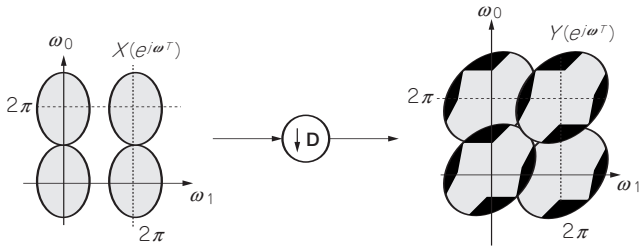
水平方向，垂直方向の間引き率，配列の標準偏差を変えてダウン・サンプリングを実行し，その周波数スペクトラムの変化を見てみよう．



（a）ダウン・サンプリング前



（b）ダウン・サンプリング後



（c）スペクトラムの変化

図9.17　非分離ダウン・サンプリングの周波数スペクトル例

例題9.11　**非分離ダウン・サンプリング**

図9.17(a)の周波数振幅特性をもつ2次元ガウス信号$X(e^{j\boldsymbol{\omega}T})$に対して，以下の間引き行列**D**によるダウン・サンプリングを施し，その結果の周波数振幅特性を描いてみよう．

$$\mathbf{D}=\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix}$$

解

以下に，MATLAB上でのコマンド例を示そう．

```
% 間引き行列（2x2）
downMtx = [ 1 1 ; -1 1 ];
% 配列の共分散行列
covMtx = [2 0 ; 0 1];
% 配列のサイズ
sizeX = 31;
% 2次元ガウス関数による配列の生成
arrayX = gaussian2cq(sizeX,covMtx);
```

```
% 2次元ダウン・サンプリング
arrayY = downsample2(arrayX,downMtx);
```

ただし，downsample2を2次元非分離ダウン・サンプルを行う関数として以下のように定義する．

```
function outputArray = ...
  downsample2(inputArray,downMtx)
% 入力配列のサイズ
nRowsInputArray = size(inputArray,1);
nColsInputArray = size(inputArray,2);
nCompInputArray = size(inputArray,3);
% ダウン・サンプリング後の配列範囲計算
vertexPoints(:,1) = [ 0 0 ].';
vertexPoints(:,2) = ...
  downMtx \ [ 0 nColsInputArray-1 ].';
vertexPoints(:,3) = ...
  downMtx \ [ nRowsInputArray-1 0 ].';
vertexPoints(:,4) = ...
  downMtx \ [ nRowsInputArray-1 ...
              nColsInputArray-1 ].';
minPoint = floor(min(vertexPoints,[],2));
maxPoint = ceil(max(vertexPoints,[],2));
% ダウン・サンプリング
clear arrayY;
iRow = 1;
for m0 = minPoint(1):maxPoint(1)
  iCol = 1;
  for m1 = minPoint(2):maxPoint(2)
    originalPoint = downMtx * [ m0 m1 ].';
    n0 = originalPoint(1);
    n1 = originalPoint(2);
    if n0 >= 0 && ...
      n0 < nRowsInputArray && ...
      n1 >= 0 && ...
        n1< nColsInputArray
```

```
        outputArray(iRow, iCol, :) = ...
          inputArray(n0+1, n1+1, :);
      else
        outputArray(iRow, iCol, :) = ...
          zeros(1,nCompInputArray,...
            class(inputArray));
      end
      iCol = iCol + 1;
    end
    iRow = iRow + 1;
  end
```

図9.17(**b**)にダウン・サンプル後の周波数振幅特性を示す．いま，ダウン・サンプル前後の信号間の周波数スペクトラムは，

$$Y(e^{j\omega T}) = \frac{1}{2}\left\{ X\left(e^{j\left[\frac{\frac{\omega_0+\omega_1}{2}}{\frac{-\omega_0+\omega_1}{2}}\right]^T} \right) + X\left(e^{j\left[\frac{\frac{\omega_0+\omega_1}{2}-\pi}{\frac{-\omega_0+\omega_1}{2}-\pi}\right]^T} \right) \right\}$$

という関係をもつ．**図9.17**(**c**)に示すように，元のスペクトラムが垂直・水平方向にそれぞれ$\sqrt{2}$倍に膨張し，45度回転した形となる．振幅は1/2となる．

実習9.8　**ダウン・サンプルと周波数スペクトラム**

M-file：practice09_8.m

ほかの間引き行列**D**についてもダウン・サンプリングを実行し，その周波数スペクトラムの変化を見てみよう．

● 多次元アップ・サンプラ

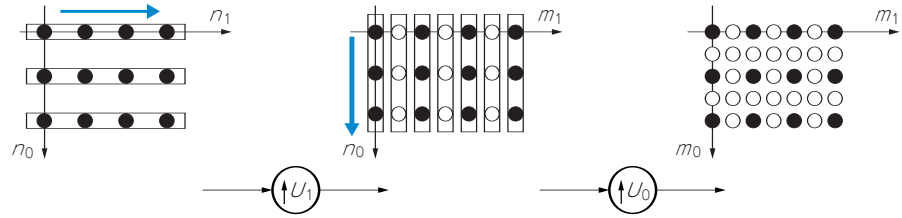
多次元ダウン・サンプラと同じように，可分離処理と非分離処理を分けて解説しよう．

(1) 可分離処理

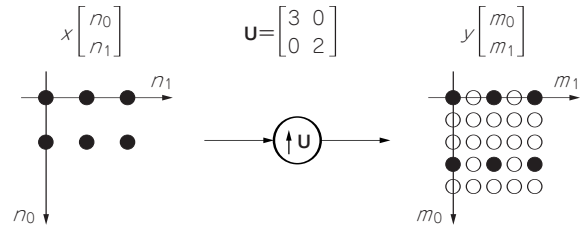
多次元アップ・サンプラは，各次元独立に1次元のアップ・サンプリングを施すことで実現できる．例題6.3はその一例である．**図9.18**(**a**)に，垂直方向，水平方向の補間率がそれぞれ3と2の場合の格子変換の様子と処理構成を示す．

U_0とU_1の値を変えることで拡大率を変えることができる．また，3次元，4次元への拡張も容易である．

ここで**補間行列U**を用いると，**図9.18**(**a**)は(**b**)のように表現できる．この行列**U**によって多次元アップ・サンプリングの入出力関係は，



（**a**）1次元処理の縦続表示



（**b**）多次元処理表示

図9.18　2次元の可分離アップ・サンプラ（ただし，$U_0 = 3$，$U_1 = 2$）

$$y[\mathbf{m}] = \begin{cases} x[\mathbf{U}^{-1}\mathbf{m}] & \mathbf{m} \in \mathrm{LAT}(\mathbf{U}) \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (9.39)$$

と表現される．ここで，LAT(**U**)は，**U**によって定義される格子を意味している（8.1節を参照）．

(2) 非分離処理

可分離アップ・サンプリングの場合，式(9.39)における補間行列**U**は対角行列となる．では，対角行列以外の**U**を選択した場合，処理後の標本化格子はどのように変わるであろうか？ **図9.19**にその一例を示そう．

このアップ・サンプリングは，各次元n_0，n_1ごとに独立に施すことはできない．すなわち，非分離のアップ・サンプリングとなる．$x[\mathbf{n}]$がアナログ信号$x(\mathbf{p})$を標本化行列**V**で標本化したものとすると，

$$y[\mathbf{m}] = \begin{cases} x(\mathbf{V}\mathbf{U}^{-1}\mathbf{m}) & \mathbf{m} \in \mathrm{LAT}(\mathbf{U}) \\ 0 & \text{その他} \end{cases} \quad \cdots\cdots (9.40)$$

より，$y[\mathbf{m}]$は，$x(\mathbf{p})$を標本化行列$\mathbf{V}' = \mathbf{V}\mathbf{U}^{-1}$で標本化した後，$y[\mathbf{m}] = 0$，$\mathbf{m} \notin \mathrm{LAT}(\mathbf{U})$と置き換えた

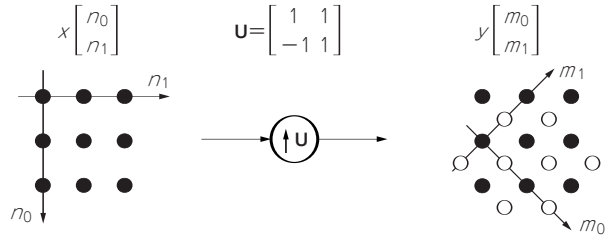


図9.19　2次元の非分離アップ・サンプラ

ものと見なせる．

なお，一般的には補間行列$\mathbf{U}$には，任意の非特異整数行列を選択することができる．また，補間率Uは，

$$U = |\det \mathbf{U}| \quad \cdots\cdots (9.41)$$

と与えられる．

例題9.12　非分離アップ・サンプリング

図9.15の右の格子から左の格子を得るための補間行列$\mathbf{U}$を求めよ．なお，処理前と処理後の標本化行列$\mathbf{V}$，$\mathbf{V}'$はそれぞれ，

$$\mathbf{V} = \begin{pmatrix} 2P_0 & 0 & P_0 \\ 0 & P_1 & 0 \\ 0 & 0 & \dfrac{T}{2} \end{pmatrix},\ \mathbf{V}' = \begin{pmatrix} P_0 & 0 & 0 \\ 0 & P_1 & 0 \\ 0 & 0 & \dfrac{T}{2} \end{pmatrix}$$

と与えられる．

解

$\mathbf{V}' = \mathbf{V}\mathbf{U}^{-1}$より，

$$\mathbf{U} = \mathbf{V}'^{-1}\mathbf{V} = \begin{pmatrix} \dfrac{1}{P_0} & 0 & 0 \\ 0 & \dfrac{1}{P_1} & 0 \\ 0 & 0 & \dfrac{2}{T} \end{pmatrix} \begin{pmatrix} 2P_0 & 0 & P_0 \\ 0 & P_1 & 0 \\ 0 & 0 & \dfrac{T}{2} \end{pmatrix} = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

なお，補間率は$U = |\det \mathbf{U}| = 2$となる．このアップ・サンプリングは，インターレース映像からプログレッシブ映像を得る変換に対応する．

● アップ・サンプラと周波数スペクトラム

1次元の場合と同じように，多次元のアップ・サンプリングも周波数領域においてイメージングを生じる．例題6.3で示したゼロ次ホールド処理は，このイメージングを抑える効果をもつ．

アップ・サンプラの周波数領域での応答について解説する前に，Z変換領域でのアップ・サンプラの入出力関係を示そう．まず，Z変換の定義と式(9.39)より，

$$Y(\mathbf{z}) = \sum_{\mathbf{m}\in\mathbb{Z}^D} y[\mathbf{m}]\mathbf{z}^{-\mathbf{m}} = \sum_{\mathbf{m}\in\mathrm{LAT}(\mathbf{U})} x[\mathbf{U}^{-1}\mathbf{m}]\mathbf{z}^{-\mathbf{m}} = \sum_{\mathbf{n}\in\mathbb{Z}^D} x[\mathbf{n}]\mathbf{z}^{-\mathbf{U}\mathbf{n}} \quad \cdots\cdots (9.42)$$

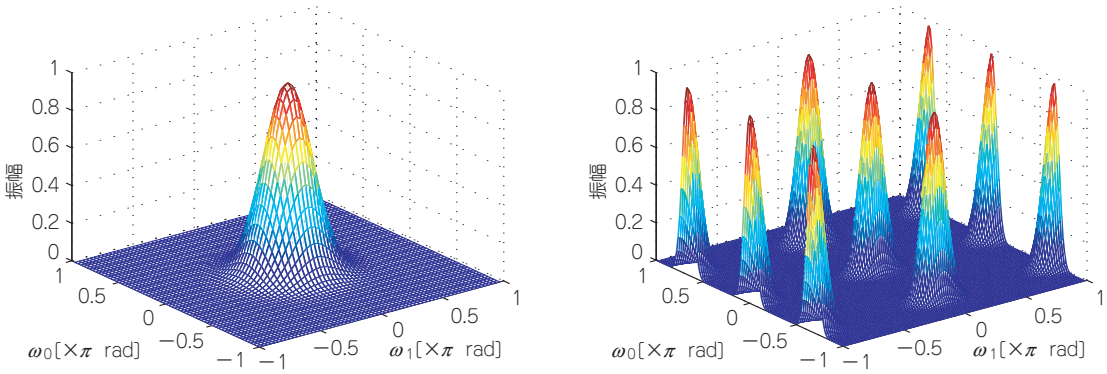
となる．よって，補間行列$\mathbf{U}$をもつアップ・サンプラの入出力関係は

$$Y(\mathbf{z}) = X(\mathbf{z}^{\mathbf{U}}) \quad \cdots\cdots (9.43)$$

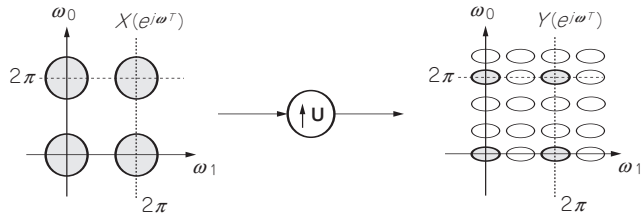
のように表現される．

(1) 周波数スペクトラム

式(9.43)に$\mathbf{z} = e^{j\omega T}$を代入すると離散空間フーリエ変換(DSFT)領域での入出力関係が導かれる．次



(a) アップ・サンプリング前　　(b) アップ・サンプリング後



(c) スペクトラムの変化

図9.20　可分離アップ・サンプリングの周波数スペクトラム例

式がその結果である．

$$Y(e^{j\omega^T}) = X(e^{j\omega^T \mathbf{U}}) \quad \cdots\cdots (9.44)$$

例題9.13　**可分離アップ・サンプリング**

図9.20(a)の周波数振幅特性をもつ2次元ガウス信号$X(e^{j\boldsymbol{\omega}^T})$に対して，垂直方向と水平方向の補間率がそれぞれ3と2のアップ・サンプリングを施し，その結果の周波数振幅特性を描いてみよう．

解

以下に，MATLAB上でのコマンド例を示そう．

```
% 垂直の補間率
verticalInpFactor = 3;
% 水平の補間率
horizontalInpFactor = 2;
% 配列の標準偏差
sigma = 2;
% 配列のサイズ
sizeX = 31;
```

```
% 2次元ガウス関数による配列の生成
arrayX = gaussian2cq(sizeX,sigma.^2);
% アップ・サンプリング
arrayY = ...
  upsample(...
    upsample(arrayX,...
      verticalInpFactor).',...
        horizontalInpFactor).';
```

図9.20(**b**)にアップ・サンプル後の周波数振幅特性を示す．いま，補間行列が$\mathbf{U} = \mathrm{diag}([3\ 2]^T)$と与えられるので，アップ・サンプル前後の信号間の周波数スペクトラムは，

$$Y(e^{j\omega^T}) = X\left(e^{j\left[\begin{smallmatrix}3\omega_0\\2\omega_1\end{smallmatrix}\right]^T}\right)$$

という関係をもつ．**図9.21**(**c**)に示すように，元のスペクトラムを水平方向，垂直方向それぞれ1/2，1/3に縮小したイメージングが現れる．

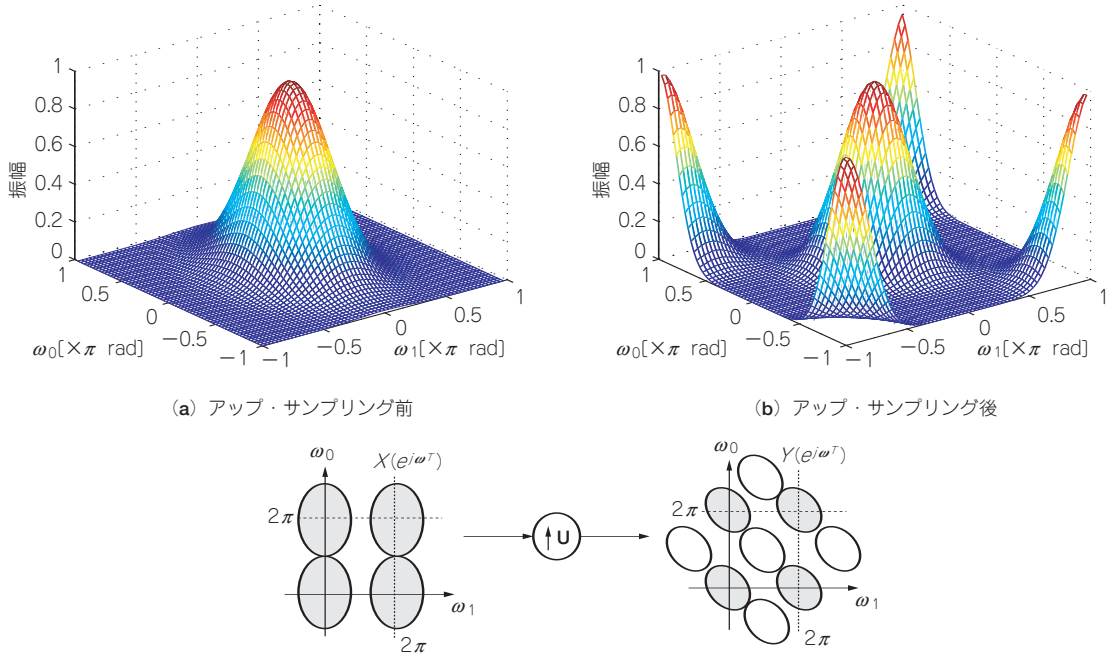


図9.21　非分離アップ・サンプリングの周波数スペクトラム例

実習9.9　**アップ・サンプルと周波数スペクトラム**

M-file：practice09_9.m

水平方向，垂直方向の補間率，配列の標準偏差を変えてアップ・サンプリングを実行し，その周波数スペクトラムの変化を見てみよう．

例題9.14　**非分離アップ・サンプリング**

図9.21(**a**)の周波数振幅特性をもつ2次元ガウス信号$X(e^{j\omega_r})$に対して，以下の補間行列$\mathbf{U}$によるアップ・サンプリングを施し，その結果の周波数振幅特性を描いてみよう．

$$\mathbf{U}=\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix}$$

解

以下に，MATLAB上でのコマンド例を示そう．

```
% 補間行列（2x2）
upMtx = [ 1 1 ; -1 1 ];
% 配列の共分散行列
covMtx = [2 0 ; 0 1];
% 配列のサイズ
sizeX = 31;
% 2次元ガウス関数による配列の生成
arrayX = gaussian2cq(sizeX,covMtx);
% 2次元アップ・サンプリング
arrayY = upsample2(arrayX,upMtx);
```

ただし，downsample2を2次元非分離ダウン・サンプルを行う関数として以下のように定義する．

```
function outputArray = ...
  upsample2(inputArray,upMtx)

% 入力配列のサイズ
nRowsInputArray = size(inputArray,1);
nColsInputArray = size(inputArray,2);
nCompInputArray = size(inputArray,3);
% アップ・サンプリング後の配列範囲計算
vertexPoints(:,1) = [ 0 0 ].';
```

```
vertexPoints(:,2) = ...
  upMtx * [ 0 nColsInputArray ].';
vertexPoints(:,3) = ...
  upMtx * [ nRowsInputArray 0 ].';
vertexPoints(:,4) = ...
  upMtx * [ nRowsInputArray nColsInputArray ].';
minPoint = floor(min(vertexPoints,[],2));
maxPoint = ceil(max(vertexPoints,[],2))-1;
% アップ・サンプリング
clear arrayY;
iRow = 1;
for m0 = minPoint(1):maxPoint(1)
  iCol = 1;
  for m1 = minPoint(2):maxPoint(2)
    outputArray(iRow, iCol, :) = ...
      zeros(1,nCompInputArray,...
        class(inputArray));
    originalPoint = upMtx \ [ m0 m1 ].';
    if [m0 m1].' == upMtx * fix(originalPoint)
      n0 = originalPoint(1);
      n1 = originalPoint(2);
      if n0 >= 0 && ...
          n0 < nRowsInputArray && ...
            n1 >= 0 && ...
              n1< nColsInputArray
        outputArray(iRow, iCol, :) = ...
          inputArray(n0+1, n1+1, :);
      end
    end
    iCol = iCol + 1;
  end
  iRow = iRow + 1;
end
```


図9-21(b)にアップ・サンプル後の周波数振幅特性を示す．いま，アップ・サンプル前後の信号の間の周波数スペクトラムは，

$$Y(e^{j\omega^T}) = X\left(e^{J\left[\frac{\omega_0-\omega_1}{\omega_0+\omega_1}\right]^T}\right)$$

という関係をもつ．**図9.21**(c)に示すように，元のスペクトラムが垂直方向，水平方向にそれぞれ$1/\sqrt{2}$に縮小し，45度回転した形となる．

実習9.10　アップ・サンプルと周波数スペクトラム

M-file：practice09_10.m

ほかの補間行列$\mathbf{U}$についてもアップ・サンプリングを実行し，その周波数スペクトラムの変化を見てみよう．

9.4　標本化格子変換フィルタ設計

本節では，多次元ダウン・サンプラや多次元アップ・サンプラと組み合わせて利用されるエリアジング回避フィルタ，イメージング除去フィルタの理想特性について概説しよう．

● 可分離デシメータ

(1) 基本構成

多次元ダウン・サンプラは周波数領域でエリアジングを生じさせてしまう．そこで，1次元の場合と同じように，ダウン・サンプラの前にエリアジング回避のためのフィルタを置くことが望ましい．**図9.22**にその構成を示す．第6章で紹介した1次元デシメータを多次元に拡張したものである．

(2) 理想フィルタ

ダウン・サンプリングによって生じるエリアジングは，$2\pi\mathbf{D}^{-T}\mathbf{k}$変調成分の折り返しにより生じる．まず，話をわかりやすくするため，可分離の場合について考えよう．

いま，d番目の次元の間引き率をD_dとすると，ダウン・サンプラへの入力信号$x[\mathbf{n}]$が各次元において$\pm\pi/D_d$で帯域制限されていれば，スペクトラムの重なりが生じない．従って，デシメーション・フィルタ$h[\mathbf{n}]$の理想周波数応答$H_{\mathbf{I}}(e^{j\omega^T})$は

$$H_{\mathbf{I}}(e^{j\omega^T}) = \begin{cases} 1 & |\omega_d| < \dfrac{\pi}{D_d} \\ 0 & \dfrac{\pi}{D_d} \le |\omega_d| < \pi \end{cases} \qquad d = 0, 1, \cdots, M-1 \quad \cdots\cdots (9.45)$$

のように与えられる(**図9.23**)．ただし，Mは次元数，ω_dはωのd番目要素である．

なお，上記フィルタはエリアジングを回避するという点においては理想的であるといえる．が，多次元信号の場合，後述する非分離標本化格子変換のところで触れるように，原信号のスペクトラム形状によっては必ずしも適切ではないことに注意されたい．現実の問題では，可分離ダウン・サ

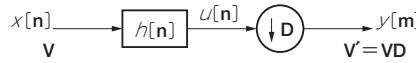


図9.22　多次元デシメータの基本構成

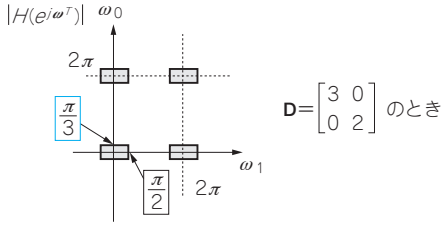


図9.23　2次元デシメーション・フィルタの理想周波数応答の例

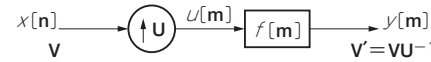


図9.24　多次元インターポレータの基本構成

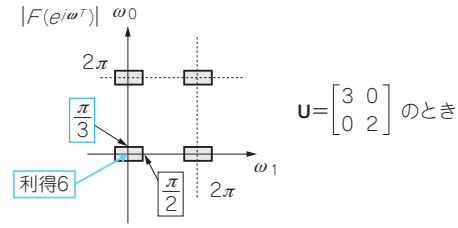


図9.25　2次元インターポレーション・フィルタの理想周波数応答の例

ンプラと非分離フィルタ，場合によっては適応的に変化するフィルタが要求される．

● 可分離インターポレータ

(1) 基本構成

多次元アップ・サンプラはイメージングを生じる．そこで，1次元の場合と同じように，アップ・サンプラの後にイメージング抑圧のためのフィルタを置く．**図9.24**にその構成を示す．これは第6章で紹介した1次元インターポレータを多次元に拡張したものである．

(2) 理想フィルタ

アップ・サンプリングによって生じるイメージングは，$2\pi\mathbf{U}^{-T}$周期で現れる．可分離の場合，d番目の次元の補間率をU_dとすると，アップ・サンプラからの出力信号$u[\mathbf{m}]$を各次元において$\pm\pi/U_d$で帯域制限すればイメージングを抑圧できる．従って，インターポレーション・フィルタ$f[\mathbf{m}]$の理想周波数応答$F_{\mathbf{I}}(e^{j\omega T})$は，

$$F_{\mathbf{I}}(e^{j\omega T}) = \begin{cases} |\det \mathbf{U}| & |\omega_d| < \dfrac{\pi}{U_d} \\ 0 & \dfrac{\pi}{U_d} \le |\omega_d| < \pi \end{cases} \qquad d = 0,1,\text{L}\ ,M-1 \quad \cdots\cdots (9.46)$$

のように与えられる(**図9.25**)．

ただし，この場合も原信号のスペクトラム形状によっては必ずしも適切ではないことに注意されたい．

● 有理数比標本化格子変換

1次元の場合と同じように，間引き行列$\mathbf{D}$のデシメータと補間行列$\mathbf{U}$のインターポレータを縦続接続すると，有理数比の標本化格子変換を実現できる．$\mathbf{V}$，$\mathbf{V}'$を，それぞれ変換前と変換後の標本化格子行列とすると，$\mathbf{V}' = \mathbf{V}\mathbf{U}^{-1}\mathbf{D}$という関係が与えられる．

図9.26に有理数比標本化格子変換の基本構成を示そう．図中のフィルタ$g[\mathbf{l}]$の理想周波数応答$G_{\mathbf{I}}(e^{j\omega T})$は，

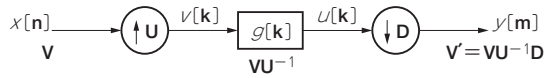


図9.26　有理数比標本化格子変換の基本構成

$$G_{\mathbf{I}}(e^{j\omega^T}) = \begin{cases} |\det \mathbf{U}| & |\omega_d| < \min\left(\dfrac{\pi}{D_d}, \dfrac{\pi}{U_d}\right) \\ 0 & \min\left(\dfrac{\pi}{D_d}, \dfrac{\pi}{U_d}\right) \le |\omega_d| < \pi \end{cases} \qquad d = 0,1,\mathrm{L}\ ,M-1 \quad \cdots\cdots (9.47)$$

のように与えられる．

例題9.15　**有理数比画像解像度変換**

入力画像を垂直2/3，水平2/3の解像度に変換する格子変換器を，フィルタ・サイズを11×11として設計しよう．また，この変換器で**図6.1**に示す画像の解像度を変換してみよう．

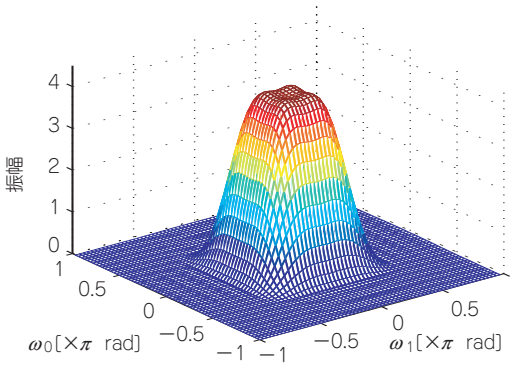
解

題意より，補間行列，間引き行列をそれぞれ

$$\mathbf{U} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$

のように設定すればよい．ここでは，フィルタを分離型で設計しよう．まず，1次元11タップの直線位相フィルタを固有フィルタ設計法で設計し，これを用いて2次元フィルタを構成する．以下に，MATLAB上でのコマンド例を示そう．なお，pictureOriginalを原画像のワークスペース上での参照とする．

```
% 垂直補間率
up0 = 2;
% 垂直間引き率
down0 = 3;
% 水平補間率
up1 = 2;
% 水平間引き率
down1 = 3;
% 垂直用フィルタ設計
factor0 = max(up0,down0);
order0 = 2 * (up0 * down0 - 1);
edgePassBand0 = 1/factor0 * 0.5;
edgeStopBand0 = 1/factor0 * 1.5;
weight0 = 0.5; % pass:stop = 1:1
g0 = eigLpFir(order0,...
```



(**a**) フィルタ

(**b**) 処理画像

図9.27　画像の解像度変換例

```
  edgePassBand0,edgeStopBand0,weight0);
% 水平用フィルタ設計
factor1 = max(up1,down1);
order1 = 2 * (up1 * down1 - 1);
edgePassBand1 = 1/factor1 * 0.5;
edgeStopBand1 = 1/factor1 * 1.5;
weight1 = 0.5; % pass:stop = 1:1
g1 = eigLpFir(order1,...
  edgePassBand1,edgeStopBand1,weight1);
% 2次元フィルタ
G = up0 * up1 * kron(g1.',g0);
% 2次元アップ・サンプリング
upMtx = diag([up0 up1]);
pictureU = upsample2(pictureOriginal,upMtx);
% 2次元畳み込み
clear pictureG;
pictureG = filter2(G,double(pictureU),'same');
% 2次元ダウン・サンプリング
downMtx = diag([down0 down1]);
pictureOutput = ...
  uint8(downsample2(pictureG,downMtx));
```


2次元フィルタの設計仕様として，$\min(\pi/D_d,\ \pi/U_d) = \pi/3$ より，通過域端を $\omega_p = \pi/3 \times 0.5$，阻止域端を $\omega_s = \pi/3 \times 1.5$ と設定した．通過域と阻止域の近似の重みを1：1として設計し，**図9.27**

(a)に示す振幅応答をもつフィルタを得た．ここで利得$|\det \mathbf{U}| = 4$を与えたことに注意されたい．このフィルタを使って標本化格子変換を施すと，**図9.27**(b)に示す画像が与えられる．

図9.26の基本構成では，ダウン・サンプラで多くの画素を棄却するにもかかわらず，いったん画像を大きくしてしまう．演算量，記憶容量の両面で負担が大きくなる．この問題を回避するためには，第6.4節で紹介したポリフェーズ実現が望ましい．上記の例題は，アップ・サンプラ，ダウン・サンプラ，フィルタがすべて可分離なので，水平，垂直それぞれ独立に有理数比レート変換を施すことで実現できる．

ただし，動画像の場合，第8.4節でも触れたように，動きに応じてスペクトラムの形状が変化する．従って，動き推定の情報など，時間方向も含めた対策が必要となることに注意されたい．

実習9.11　**有理数比画像解像度変換**

M-file：practice09_11.m

変換比やフィルタの仕様を変えて，有理数比画像解像度変換器を設計してみよう．

● 非分離標本化格子変換

ここでは非分離標本化格子変換のためのフィルタ設計手法について解説しよう．ただし，多次元標本化格子変換フィルタの理想特性は単純な問題ではない．以下では，典型的な理想特性の設定について説明する．

(1) フィルタの理想特性

格子変換の一般的な理想特性を与える前に，非特異実行列$\mathbf{V}$で与えられる対称平行超平面体領域(Symmetrical Parallelpiped)SPD($\mathbf{V}$)について定義を示す．SPD($\mathbf{V}$)は，

$$\mathrm{SPD}(\mathbf{V}) = \{\mathbf{V}\mathbf{x} \mid \mathbf{x} \in [-1, 1)^M\} \quad \cdots\cdots (9.48)$$

と定義される．ここで，$\mathbf{x} \in [a, b)^M$は，M次元ベクトル$\mathbf{x}$のi番目要素$\mathbf{x}_i$が$a < \mathbf{x}_i < b$の範囲にあることを意味する．

例題9.16　**SPD**

次の行列$\mathbf{V}$によって与えられるSPD($\mathbf{V}$)を求めよ．

$$\mathbf{V} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

解

まず，**図9.28**に示すようなLAT($\mathbf{V}$)を考える．すると，原点を囲む四つの領域がSPD($\mathbf{V}$)となる．

SPD($\pi\mathbf{D}^{-T}$)内に帯域制限された信号は，間引き行列$\mathbf{D}$の間引き処理を行ってもエリアジングを生じない．より一般的には，任意のユニモジュラ行列$\mathbf{E}$を用いてSPD($\pi(\mathbf{DE})^{-T}$)と表現できる．したがって，間引き行列$\mathbf{D}$のデシメーション・フィルタの理想特性は，

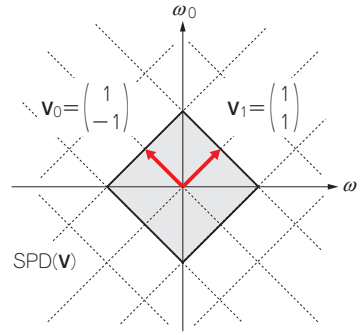


図9.28　SPD(V)の例

$$H_{\mathbf{I}}(e^{j\omega^T})=\begin{cases}1 & \omega\in \mathrm{SPD}(\pi\mathbf{D}'^{-T})\\ 0 & \omega\in[-\pi,\pi)^M-\mathrm{SPD}(\pi\mathbf{D}'^{-T})\end{cases} \quad \cdots\cdots(9.49)$$

のように設定できる．ただし，$\mathbf{D}'=\mathbf{DE}$とする．元のスペクトラムの連続性や形状を保持するためには，理想特性の設定に注意を払わなければならない．

図9.29(**a**)，(**b**)に次の間引き行列

$$\mathbf{D}=\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix} \quad \cdots\cdots(9.50)$$

に対する理想フィルタの通過域を示す．**図9.29**(**a**)では$\mathbf{E}=\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$，**図9.29**(**b**)では$\mathbf{E}=\left(\begin{smallmatrix}1&0\\1&-1\end{smallmatrix}\right)$と選択した．参考までに入力信号のスペクトラム$X(e^{j\boldsymbol{\omega}^T})$のサポート領域の例を示す．**図9.29**(**a**)のフィルタでは，信号スペクトラムが一部阻止域に重なるが，**図9.29**(**b**)のフィルタではすべて通過することが分かる．すなわち，理想特性の設定が入力信号に大きく依存することが分かる．なお，**図9.29**(**b**)の特性は，周波数応答の周期性から，通過域が$|\omega_0|<\pi/2$の垂直方向の1次元低域通過フィルタとしてよい．

補間行列**U**のインターポレーション・フィルタの理想特性は，

$$F_{\mathbf{I}}(e^{j\omega^T})=\begin{cases}|\det\mathbf{U}| & \omega\in \mathrm{SPD}(\pi\mathbf{U}'^{-T})\\ 0 & \omega\in[-\pi,\pi)^M-\mathrm{SPD}(\pi\mathbf{U}'^{-T})\end{cases} \quad \cdots\cdots(9.51)$$

のように設定できる．ただし，$\mathbf{U}'=\mathbf{UE}$とする．ここでも，イメージング成分を適切に除去できるか

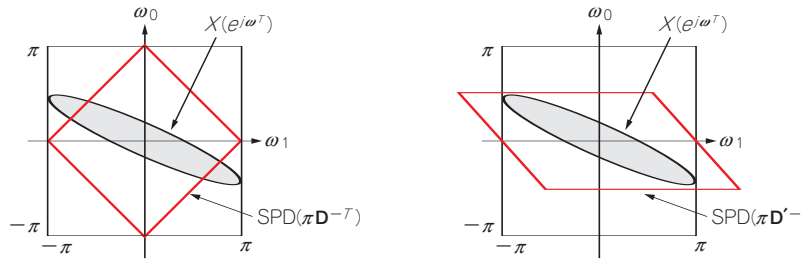
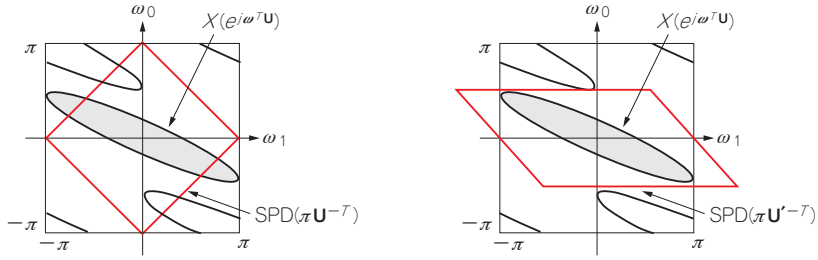


図9.29　デシメーション・フィルタの理想特性



(**a**) 特性(1)　　(**b**) 特性(2)

図9.30　インターポレーション・フィルタの理想特性

否かは，元のスペクトラムの形状に依存する．スペクトラムの連続性や形状を保持するためには，理想特性の設定に注意を払わなければならない．

図9.30(**a**)，(**b**)に補間行列

$$\mathbf{U} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \quad \cdots\cdots (9.52)$$

に対する理想フィルタの通過域を示す．**図9.30**(**a**)では$\mathbf{E} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$，**図9.30**(**b**)では$\mathbf{E} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$と選択した．参考までにアップ・サンプリング後の信号のスペクトラム$X(e^{j\omega^T\mathbf{U}})$のサポート領域の例も重ねて示す．**図9.30**(**a**)のフィルタでは，イメージングのスペクトラムが一部通過域に重なり，オリジナルのスペクトラムが一部阻止域に重なる．一方，**図9.30**(**b**)のフィルタでは，イメージングを完全に除去でき，オリジナルを完全に通過させることが分かる．ここでも，理想特性の設定が入力信号に依存することが分かる．

(2) 間引き設計法

理想特性が適切に設定されたとして，以下では，一般的な非対角の間引き行列$\mathbf{D}$あるいは補間行列$\mathbf{U}$に対応できる間引き設計法を紹介しよう．

間引き設計法の特徴は以下のとおりである．

- 1次元フィルタから設計できる．
- 任意の次元M，任意の非特異行列$\mathbf{D}$に対し，式(9.49)の特性を近似できる．
- 一般の多次元非分離フィルタに比べて，設計と実装が効率的である．
- 多次元ナイキスト・フィルタを設計できる．

以下にその手続きを示す．

1. **図9.31**に示す理想特性を近似する1次元低域通過フィルタ$P(e^{j\omega})$を設計する．
2. 次式のように多次元可分離フィルタ$h^{(s)}[\mathbf{n}]$を定義する．

   $h^{(s)}[\mathbf{n}] = p[n_0] \cdot p[n_1] \cdots p[n_{M-1}]$

   ただし，$p[n_d]$は，$P(e^{j\omega})$のd番目次元のインパルス応答
3. $h^{(s)}[\mathbf{n}]$を次式のように行列$\hat{\mathbf{D}}$で間引き，スケーリングすることで，フィルタ$H(e^{j\omega^T})$のインパルス応答$h[\mathbf{n}]$を得る．

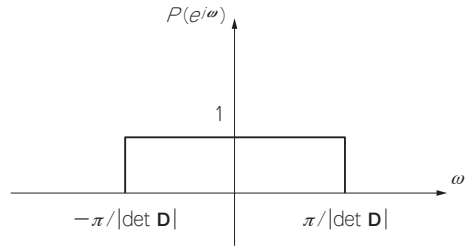


図9.31　プロトタイプ1次元低域通過フィルタの理想特性

$h[\mathbf{n}] = |\det \mathbf{D}|^{M-1} h^{(s)}[\hat{\mathbf{D}}\mathbf{n}]$

ただし，$\hat{\mathbf{D}} = |\det \mathbf{D}|\mathbf{D}^{-1}$である．

例題9.17　間引き設計法

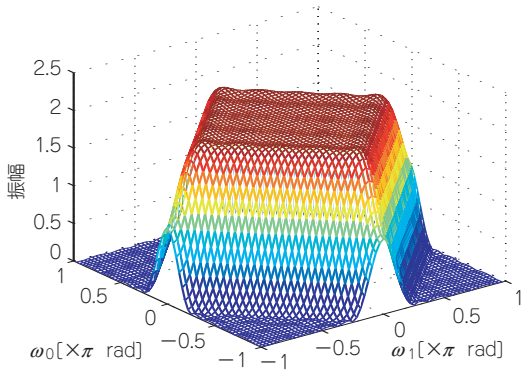
式(9.50)の間引き行列$\mathbf{D}$に対応するデシメーション・フィルタを設計しよう．

解

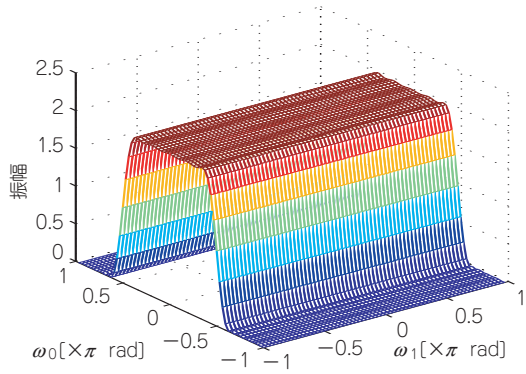
以下に，MATLAB上でのコマンド例を示そう．

```
% 間引き行列（2x2）
downMtx = [ 1 1 ; -1 1 ];
absDetD = abs(det(downMtx));
% プロトタイプ1次元フィルタ設計
nOrder = 20; % プロトタイプフィルタの次数
passBandEdge = 1/absDetD * 0.8;
stopBandEdge = 1/absDetD * 1.2;
freqWeight = 0.5;
prtFilter = eigLpFir(nOrder,...
  passBandEdge,stopBandEdge,freqWeight);
% 間引きフィルタ設計1
filter4Dec1 = ...
  downsampleFilterDesign2(prtFilter, downMtx);
% 間引きフィルタ設計2
unimodMtx = [ 1 0 ; 1 -1 ];
filter4Dec2 = downsampleFilterDesign2(...
  prtFilter, downMtx*unimodMtx);
```


$|\det \mathbf{D}| = 2$より，通過域$|\omega| < \pi/2$を理想特性とする近似フィルタを1次元プロトタイプ・フィル



(a) フィルタ(1)



(b) フィルタ(2)

図9.32　間引き設計法による設計例

タ $P(e^{j\omega})$として固有フィルタ設計法により設計した．次数を20，通過域端を $\omega_p = \pi/4$，阻止域端を $\omega_s = 3\pi/4$，通過域阻止域の周波数重みを1：1とし，$\mathbf{E} = \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$，$\mathbf{E} = \left(\begin{smallmatrix}1 & 0\\ 1 & -1\end{smallmatrix}\right)$の2種類のユニモジュラ行列を用いて間引き設計している．

downsampleFilterDesign2は，間引き設計法を行う関数として次のように定義した．

```
function h = downsampleFilterDesign2(...
  prtFilter, slaMtx);

% 間引き行列の行列式の絶対値
absDetMtx = abs(det(slaMtx));
% 2次元可分離フィルタの作成
spFilter = kron(prtFilter(:).',prtFilter(:));
% ダウン・サンプリング
adjMtx = absDetMtx * inv(slaMtx);
h = absDetMtx * ...
  downsample2(spFilter,adjMtx);
```

設計したフィルタの周波数振幅応答を，それぞれ**図9.32**(**a**)，(**b**)に示す．

実習9.12　**間引き設計法**

M-file：practice09_12.m

さまざまな間引き行列**D**に対応するデシメーション・フィルタを設計しよう．また，設計したフィルタを使ってデシメーションを行おう．

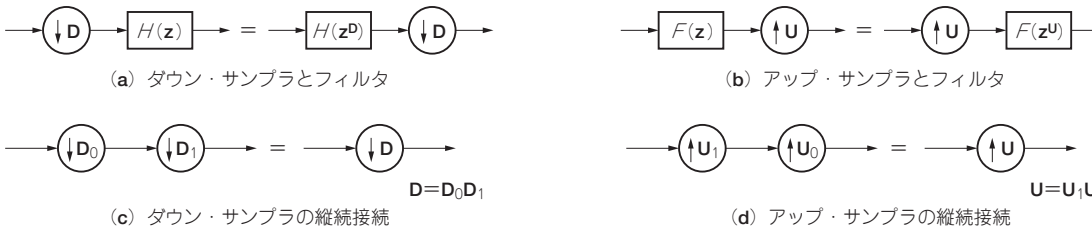


図9.33　格子変換器の等価関係

(3) ナイキスト(*M*)フィルタ設計

$M = |\det \mathbf{M}|$としてプロトタイプの1次元低域通過フィルタをナイキスト(*M*)フィルタ(6.5節を参照)で与えると，間引き手法は多次元のナイキスト(**M**)フィルタを与える．

実習9.13　**多次元ナイキスト・フィルタ設計**

M-file：practice09_13.m

式(9.52)の補間行列**U**に対応するインターポレーション・フィルタをナイキスト(**U**)フィルタとして設計しよう．

● 多次元マルチレート・システムの諸性質

非分離システムにおいても，デシメータとインターポレータを縦続接続することで$\mathbf{UD}^{-1}$倍の標本化格子変換が可能となる．また，**図9.33**に示す諸性質を利用することで，ポリフェーズ実現も同様に可能である．

● 適応処理の必要性

分離型，非分離型共に多次元信号のフィルタリング問題は単純ではない．信号のスペクトラム形状にフィルタの理想特性が依存することが主な理由である．従って，実際には信号に依存した適応的な処理が必要とされる．

例えば，単板式のカラー画像センサでは**図9.34**に示すようなカラー・フィルタを通して撮像を行う．各色成分の欠けた部分については補間を行うことになるが，空間的にこう配の小さな方向(エッ

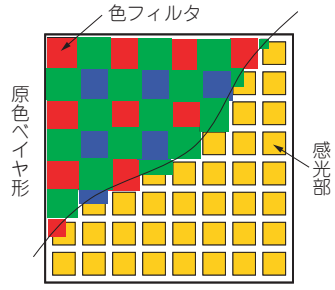


図9.34
ベイヤ形カラー・フィルタ

ジ)を検出し，その方向に対して平滑化による補間を施す方法がある．それは局所的なスペクトラム形状に合わせてフィルタの周波数応答を適応的に変えることにほかならない．

プログレッシブ映像からインターレース映像への変換や，インターレース映像からプログレッシブへの変換もまた，多次元標本化格子変換そのものである．空間解像度変換もしばしば要求される．映像のスペクトラムが動きによって変化することから，使用するフィルタの特性も合わせて変化することが望ましい．このため，動き適応/補償フィルタの技術が必要となる．次章にて簡単なシミュレーション例を紹介しよう．

章末問題

問題9.1 **分離処理**(MATLAB演習)

任意の2次元配列に対して，例題9.3で設計したフィルタによる2次元畳み込みを実行してみよう．また，この設計に用いた1次元フィルタによる1次元畳み込みを各次元独立に施し，2次元畳み込みの結果と一致することを確かめ，演算量について考察してみよう．

問題9.2 **ファン・フィルタ**(MATLAB演習)

以下のパラメータを設定すると，マクレラン変換によって2次元ファン・フィルタを設計できる．

$$A = B = C = 0,\ D = \frac{1}{2},\ E = -\frac{1}{2}$$

例題9.6を参考に設計してみよう．

問題9.3 **高域通過型(Ⅰ)フィルタ**(MATLAB演習)

例題9.5で設計した低域通過フィルタから，式(9.26)の変換を利用して線形位相をもつ(インパルス応答が対称な)高域通過型(Ⅰ)のFIRフィルタを設計してみよう．ただし，式(9.26)がゼロ位相フィルタに対する変換であることから，インパルス応答の中心の位置に注意しよう．

問題9.4 **高域通過型(Ⅱ)フィルタ**(MATLAB演習)

例題9.5で設計した低域通過フィルタから，式(9.28)の変換を利用して線形位相をもつ高域通過型(Ⅱ)FIRフィルタを設計してみよう．

問題9.5 **帯域通過型FIRフィルタ**(MATLAB演習)

例題9.5で設計した低域通過フィルタから，式(9.30)の変換を利用して線形位相をもつ帯域通過型FIRフィルタを設計してみよう．

問題9.6 **非分離ダウン・サンプリング**

対角の標本化行列

$$\mathbf{V}=\begin{pmatrix} P_0 & 0 & 0 \\ 0 & P_1 & 0 \\ 0 & 0 & \dfrac{T}{2} \end{pmatrix}$$

によって標本化したプログレッシブ映像に対して，間引き行列

$$\mathbf{D}=\begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

によるダウン・サンプリングを施した後の標本化行列$\mathbf{V}'$を求めよう．また，処理後の標本化格子（FCO：Face-Centered Orthorhombic格子となる）も描いてみよう．

問題9.7　可分離アップ・サンプリング

任意の3次元信号に対して，各次元の補間率を$U_0=2$，$U_1=3$，$U_2=4$として3次元アップ・サンプリングを施したい．補間行列$\mathbf{U}$を求めよ．

問題9.8　非分離有理数比標本化格子変換

1080iフォーマットの映像（画素数1080×1920，フレーム・レート29.97fps，インターレース方式）を720pフォーマットの映像（画素数720×1280，フレーム・レート59.94fps，プログレッシブ方式）に変換するための補間行列$\mathbf{U}$と間引き行列$\mathbf{D}$を求めてみよう．

問題9.9　ポリフェーズ実現

垂直時間（VT：Vertical-temporal）フィルタ

$$F(\mathbf{z})=1+\frac{1}{4}(z_0+z_0^{-1})+\frac{1}{2}z_T^{-1}$$

をイメージング除去フィルタ（**図9.24**を参照）として利用したインターレース-プログレッシブ（IP）変換（例題9.12を参照）のポリフェーズ実現構成を考えてみよう．

参考文献

(1) 川又政征，樋口龍雄；多次元ディジタル信号処理，朝倉出版，1995年．
(2) D. E. Dudgeon and R. M. Mersereau；*Multidimensional Digital Signal Processing*，Prentice Hall，1983.
(3) John, W. Woods；*Multidimensional Signal, Image and Video Processing and Coding*，Academic Press，2006.
(4) P.P. Vaidyanathan；*Multirate Systems and Filter Banks*，Prentice Hall，1993.
(5) Bahadir K. Gunturk, John Glotzbach, Yucel Altunbasak, Ronald W. Schafer, and Russel M. Mersereau；*Demosaicking: Color FIlter Array Interpolation*，IEEE Signal Proc. Magazine，Vol.44，Jan. 2005.

第10章

動き適応/補償フィルタリング

カメラでは，通常，明確な帯域制限が施されることなく，光学系を通してセンサ上に結ばれた像が画像・映像信号として取り込まれる．従って，画像・映像信号はエリアジングを含む可能性がある．また，スペクトラム形状に応じた処理が必要であることにも注意しなければならない．特に映像では，動きに応じてスペクトラムが傾くので，フィルタリングの際には動き情報の利用が欠かせない．本章では，動きの情報の有効性を示す応用技術としてインターレース-プログレッシブ(IP)変換について解説しよう．

10.1　固定係数IP変換

まず，例題9.12で紹介したIP変換を簡単な固定係数フィルタを使って実装し，動き情報を使わない場合の問題点を確認しよう．

なお，以降のシミュレーションでは，次に示すmkInterlacedMov関数で生成したインターレース映像を利用する．この関数は，**図10.1**に示すようにランダムなパターンを背景とし，二つのオブ

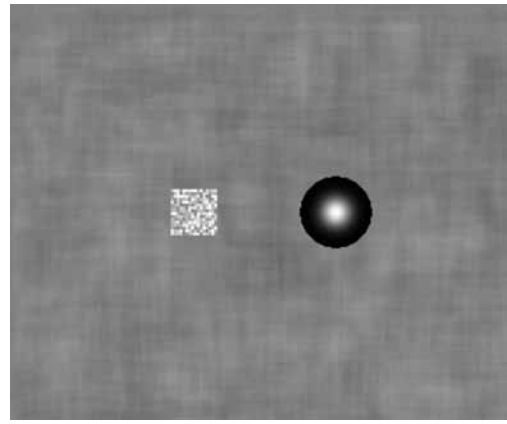


図10.1
シミュレーション用画像

ジェクトが一定の速度で移動する映像を与える．

```
function frameSeq = mkInterlacedMov(velocity)
% 静止画像の読み込み
pictureObj = ...
  imread('./data/squareandgauss.tif');
pictureBg = ...
  imread('./data/background.tif');
% インターレース映像の生成
nFrames = 16; % フレーム数
pictureI = ...
  zeros(size(pictureObj),'uint8');
for iFrame = 1:nFrames
  pictureObj = ...
    circshift(pictureObj, velocity);
  picture = pictureObj + ...
    uint8(not(pictureObj)) .* pictureBg;
  pictureI(2:2:end,:) = picture(2:2:end,:);
  pictureObj = ...
    circshift(pictureObj, velocity);
  picture = pictureObj + ...
    uint8(not(pictureObj)) .* pictureBg;
  pictureI(1:2:end,:) = picture(1:2:end,:);
  frameSeq(iFrame) = ...
    im2frame(pictureI,gray(256));
end
```

IP変換の概要を**図10.2**に示す．水平方向の補間は必要ないため，時間-垂直平面のみを表示している．白丸がインターレース映像の標本点であり，黒丸はIP変換によって補間される標本点を表している．

固定係数フィルタを使った補間は，映像の性質とは独立に各黒丸の画素値を周辺の白丸の画素値から与える．9.4節でも示したとおり，このフィルタの理想特性は処理される映像の性質に依存する．

● 時間方向補間

時間方向補間の様子を**図10.3**に示す．同図の手法はフィールド挿入，あるいはWeaveとも呼ばれ，一つの補間を与える時間方向補間フィルタは，

$$G(z) = 1 + z_T^{-1} \quad \cdots\cdots (10.1)$$

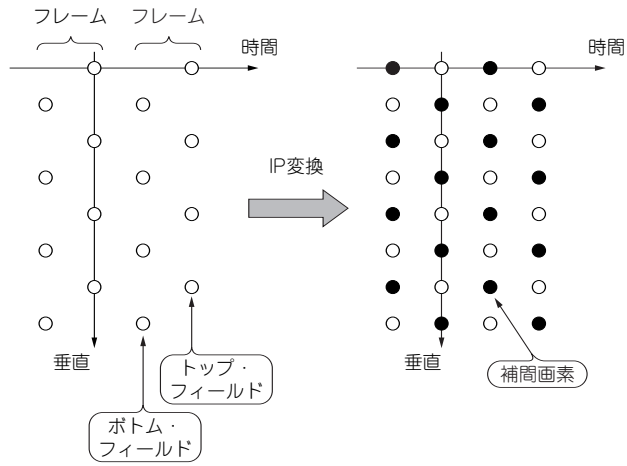


図10.2　IP変換（デインターレース）の概要

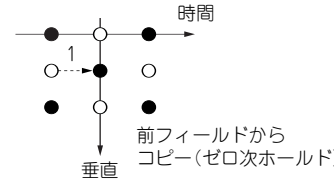
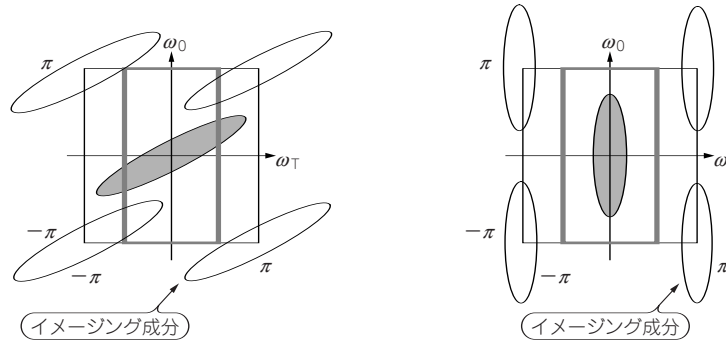


図10.3
時間方向補間
（点線矢印上の数値は重み係数）



（a）速い動きのある映像　　（b）動きのない映像

図10.4　アップ・サンプリング後の映像の周波数特性と時間方向フィルタの周波数応答

と表現される．速い動きのある映像では，傾いたスペクトラムの端部が除去され，空間解像度が失われてしまう．また，イメージング成分が除去されずに時間方向の高周波成分が空間の高周波成分「くし状効果」として残ってしまう（**図10.4**（**a**））．垂直方向のみならず水平方向の動きでも同じ問題が生じる．一方，このフィルタは時間方向のみに帯域制限を施すため，動きがなく時間方向の高周波成分が少ない映像では，空間解像度を高く保つ特性をもつ（**図10.4**（**b**））．

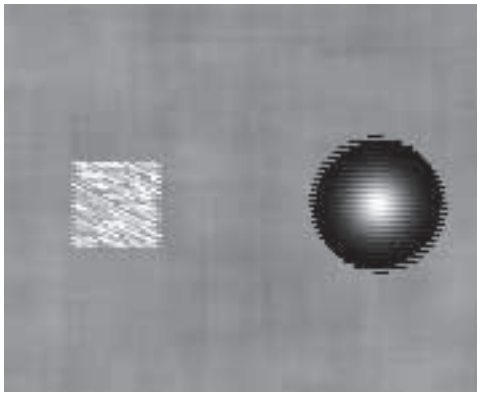
例題10.1　時間方向補間

一定の速度$(v_0, v_1)^T = (2, 2)^T$［画素/フィールド］でオブジェクトが並進移動するインターレース映像に対して時間方向補間IP変換を施し，プログレッシブ映像を生成してみよう．また，動きのない$(v_0, v_1)^T = (0, 0)^T$の場合と比較してみよう．

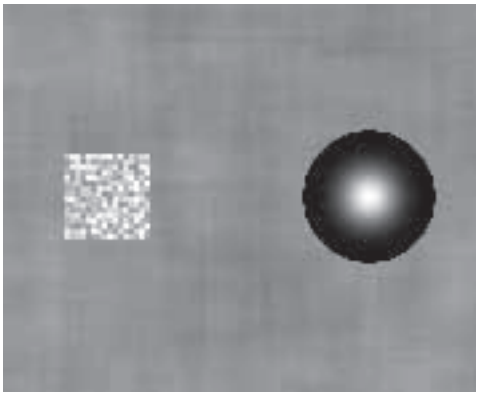
解

以下にMATLAB上でのコマンド例を示す．

```
% 速度ベクトル（垂直，水平）
velocity = [2 2];
% インターレース映像モデルの生成
frameSeq = mkInterlacedMov(velocity);
nFramesIn = length(frameSeq);
% avifile オブジェクトの生成
outputFileName = 'tmp/tempip.avi';
frameSize = size(frame2im(frameSeq(1)));
frameRate = 30;
nFrameOut = 2 * nFramesIn;
clear mex;
%
aviObj = avifile(outputFileName,...
  'FPS',frameRate,'COMPRESSION','None');
% 時間方向補間IP変換
deintPicture0 = 128*ones(frameSize,'uint8');
deintPicture1 = 128*ones(frameSize,'uint8');
previousField = ...
  128*ones(frameSize./[2 1],'uint8');
for iFrame = 1:nFramesIn
  pictureI = frame2im(frameSeq(iFrame));
  %--- Process for picture0
  deintPicture0(1:2:end,:) = previousField;
  deintPicture0(2:2:end,:) = pictureI(2:2:end,:);
  %--- Process for picture1
  deintPicture1 = pictureI;
  previousField = pictureI(1:2:end,:);
  %
```



(**a**) $(v_0,\ v_1)^T=(2,\ 2)^T$



(**b**) $(v_0,\ v_1)^T=(0,\ 0)^T$

図10.5　時間方向補間の結果

```
    aviObj = addframe(aviObj, ...
      im2frame(deintPicture0,gray(256)));
    aviObj = addframe(aviObj, ...
      im2frame(deintPicture1,gray(256)));
  end
  aviObj = close(aviObj);
```

動きのない場合については，velocity＝[0　0]と設定すればよい．**図10.5**に出力のオブジェクト部を拡大して示す．

図10.5より，時間方向補間は動きのある領域でくし状効果を生じ，動きのない映像に対して鮮明なフレーム画像を与えることが確認できる．

実習10.1　**時間方向補間**

M-file：practice10_1.m

オブジェクトの移動速度(velocity)を変えて試してみよう．例題4.9で紹介したCIF映像mobile.cifはインターレース映像である．この映像に対して時間方向補間を施してプログレッシブ化してみよう．

● 垂直方向補間

垂直方向補間の様子を**図10.6**に示す．しばしば，この手法はBobと呼ばれる．同図の補間を与える垂直方向補間フィルタは，

$$G(z)=1+\frac{1}{2}(z_0^{1}+z_0^{-1}) \quad \cdots\cdots(10.2)$$

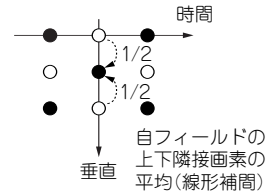
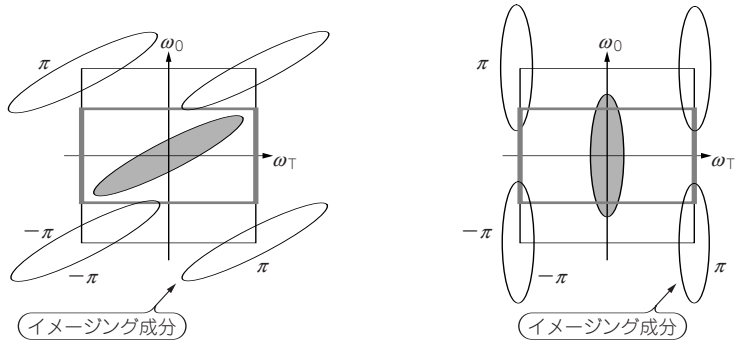


図10.6
垂直方向補間
(点線矢印上の数値は重み係数)



(**a**) 速い動きのある映像　　(**b**) 動きのない映像

図10.7　アップ・サンプリング後の映像の周波数特性と垂直方向フィルタの周波数応答

と表現され，垂直方向のみに帯域制限を施す．処理がフィールド内で閉じているため，動きのある映像に対しても時間方向フィルタのようなくし状効果は現れない(**図10.7**(**a**))．しかし，動きのない映像では，イメージングが除去されずに空間の高周波成分が時間方向の高周波成分「ちらつき」として残ってしまう(**図10.7**(**b**))．

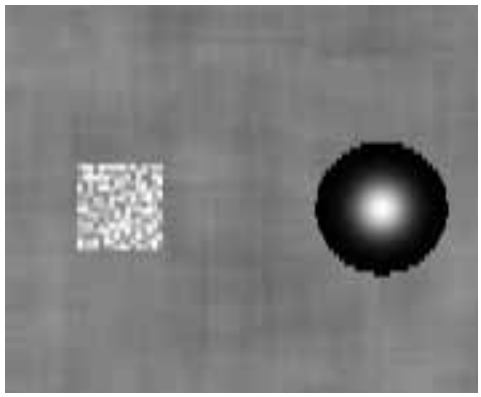
例題10.2　**垂直方向補間**

一定の速度$(v_0, v_1)^T = (2, 2)^T$[画素/フィールド]でオブジェクトが並進移動するインターレース映像に対して垂直方向補間IP変換を施し，プログレッシブ映像を生成してみよう．また，動きのない$(v_0, v_1)^T = (0, 0)^T$の場合と比較してみよう．

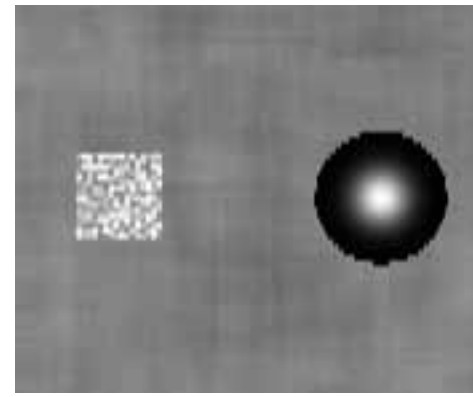
解

以下にMATLAB上でのコマンド例を示そう．ただし，forループの外側の処理は例題10.1の解とほぼ同じであるため，ここでは割愛する．

```
for iFrame = 1:nFramesIn
  pictureI = frame2im(frameSeq(iFrame));
  %--- Process for picture0
  deintPicture0(1:2:end,:) = ...
      1/2 * pictureI(2:2:end,:) ...
    + 1/2 * [ pictureI(2,:) ; ...
```



(**a**) $(v_0,\ v_1)^T=(2,\ 2)^T$



(**b**) $(v_0,\ v_1)^T=(0,\ 0)^T$

図10.8　垂直方向補間の結果

```
            pictureI(2:2:end-2,:)];
    deintPicture0(2:2:end,:) = pictureI(2:2:end,:);
    %--- Process for picture1
    deintPicture1(1:2:end,:) = pictureI(1:2:end,:);
    deintPicture1(2:2:end,:) = ...
      1/2 * pictureI(1:2:end,:) ...
      + 1/2 * [ pictureI(3:2:end,:) ; ...
            pictureI(end-1,:)] ;
    %
    aviObj = addframe(aviObj, ...
      im2frame(deintPicture0,gray(256)));
    aviObj = addframe(aviObj, ...
      im2frame(deintPicture1,gray(256)));
  end
```

なお，上記のコマンド例は画面境界部にて対称拡張法を適用している．**図10.8**に処理後のオブジェクト部を拡大して示す．

図10.8より，垂直方向補間は動きの有無によらず同等のフレーム画像を与えることが分かる．画面のちらつきについては処理後の映像で確認されたい．

実習10.2　垂直方向補間

M-file：practice10_2.m

オブジェクトの移動速度(velocity)を変えて試してみよう．また，例題4.9で紹介したインターレース映像mobile.cifに対して垂直方向補間を施し，プログレッシブ化してみよう．

● 時間垂直補間

時間垂直補間の様子を**図10.9**に示す．同図の補間を与える時間垂直補間フィルタは，

$$G(\mathbf{z}) = 1 + \frac{1}{2}z_T^{-1} + \frac{1}{4}(z_0^1 + z_0^{-1}) \quad \cdots\cdots (10.3)$$

と表現され，ひし形の通過帯域をもつ．時間方向補間と垂直方向補間の中間的な結果が得られ，動きのある領域についてはくし状部の抑圧効果をもち(**図10.10**(**a**))，動きのない領域については空間高周波を保つ効果をもつ(**図10.10**(**b**))．

例題10.3　**時間垂直補間**

一定の速度$(v_0, v_1)^T = (2, 2)^T$[画素/フィールド]でオブジェクトが並進移動するインターレース映像に対して時間垂直補間IP変換を施し，プログレッシブ映像を生成してみよう．また，動きのない$(v_0, v_1)^T = (0, 0)^T$の場合と比較してみよう．

解

以下にMATLAB上でのコマンド例を示そう．ただし，forループの外側の処理は例題10.1の解とほぼ同じであるため．ここでは割愛する．

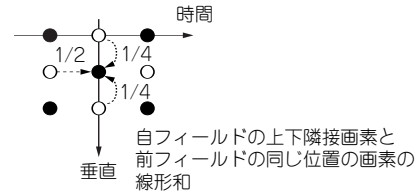
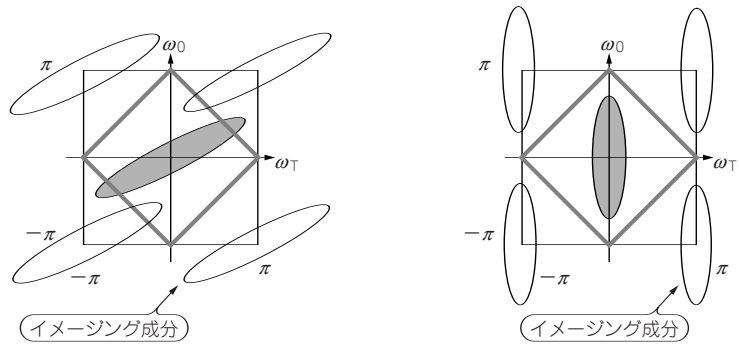


図10.9
時間垂直間
(点線矢印上の数値は重み係数)



(**a**) 速い動きのある映像　　(**b**) 動きのない映像

図10.10　アップ・サンプリング後の映像の周波数特性と時間垂直フィルタの周波数応答

```
for iFrame = 1:nFramesIn
  pictureI = frame2im(frameSeq(iFrame));
  %--- Process for picture0
  deintPicture0(1:2:end,:) = ...
    1/4 * pictureI(2:2:end,:)...
    + 1/2 * previousField...
    + 1/4 * [ pictureI(2,:) ; ...
            pictureI(2:2:end-2,:)];
  deintPicture0(2:2:end,:) = pictureI(2:2:end,:);
  previousField = pictureI(2:2:end,:);
  %--- Process for picture1
  deintPicture1(1:2:end,:) = pictureI(1:2:end,:);
  deintPicture1(2:2:end,:) = ...
    1/4 * pictureI(1:2:end,:)...
    + 1/2 * previousField...
    + 1/4 * [ pictureI(3:2:end,:) ; ...
            pictureI(end-1,:)];
  previousField = pictureI(1:2:end,:);
  %
  aviObj = addframe(aviObj, ...
    im2frame(deintPicture0,gray(256)));
  aviObj = addframe(aviObj, ...
    im2frame(deintPicture1,gray(256)));
end
```

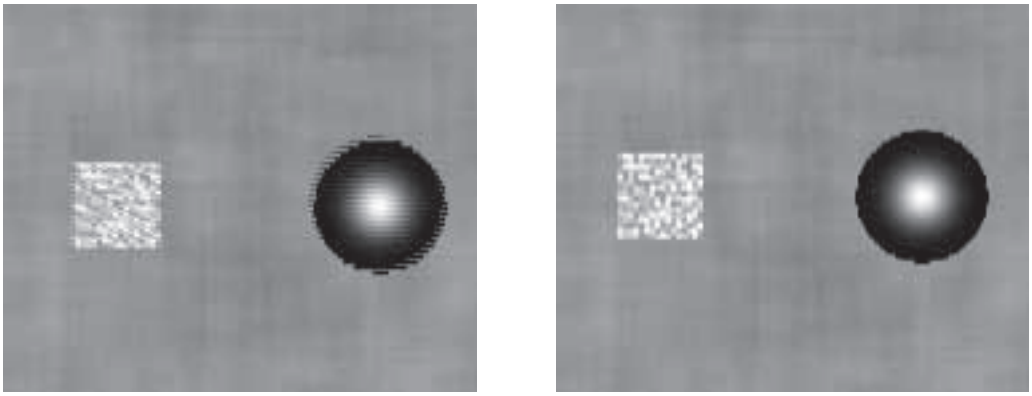
図10.11 に処理後のオブジェクト部を拡大して示す.

時間垂直補間では，最良の結果は得られないものの，時間方向補間，垂直方向補間で生じる最悪の結果を回避することができる.

実習10.3　**時間垂直補間**

M-file：practice10_3.m

オブジェクトの移動速度(velocity)を変えて試してみよう．また，例題4.9で紹介したインターレース映像mobile.cifに対して時間垂直補間を施し，プログレッシブ化してみよう.



(**a**) $(v_0,\ v_1)^T=(2,\ 2)^T$　　(**b**) $(v_0,\ v_1)^T=(0,\ 0)^T$

図10.11　時間垂直補間の結果

10.2　動き適応IP変換

固定係数IP変換では，その補間フィルタの周波数応答によって得手不得手があることを理解できたと思う．では次に，固定係数IP変換からの改善の第一歩として，動きの有無に応じて処理を切り替える**動き適応IP変換**について解説しよう．

● 動き検出型適応補間

動き適応IP変換器の典型的な構成を**図10.12**に示す．「動き検出器」は，映像の状態を解析し，パラメータαを抽出してこれを補間フィルタ$G_\alpha(\mathbf{z})$に与える．補間フィルタ$G_\alpha(\mathbf{z})$がパラメータαに応じて特性を変えるという仕掛けである．動きがなければ時間方向補間，動きがあれば垂直方向補間，あるいは時間垂直補間となるように設計すればよい．

(1) 可変係数補間

以下に$G_\alpha(\mathbf{z})$の一例を示そう．

$$G_\alpha(\mathbf{z})=1+(1-\alpha)z_T^{-1}+\frac{\alpha}{2}(z_0^{1}+z_0^{-1})\qquad 0\le\alpha\le 1 \quad \cdots\cdots(10.4)$$

図10.13はこのフィルタによる補間の様子を示している．$\alpha=0$のときは時間方向補間，$\alpha=1$のときは垂直方向補間，$\alpha=1/2$のときは時間垂直補間となることを確認できる．

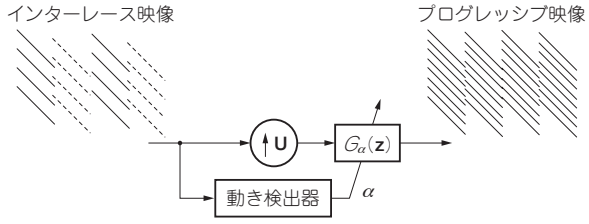


図10.12　動き検出型適応補間IP変換の基本構成

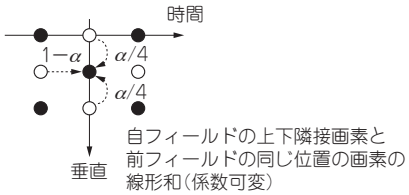


図10.13　可変係数補間の例

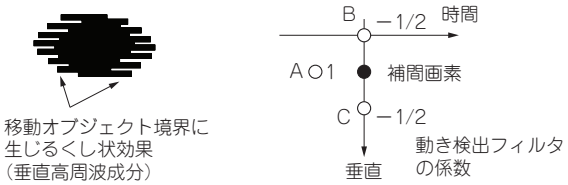


図10.14　くし状効果と動き（くし状部）検出フィルタ係数

(2) 動き検出

動きの定義により検出器の仕様は変わる．時間方向補間では動きによってくし状の効果が生じるが，この垂直高周波成分の強い領域を動き領域として定義する動き検出フィルタを**図10.14**に示す．補間画素周辺の画素A，B，Cの重み付け差分の絶対値と非線形変換$T(x)$によって動き検出器を構成できる．

$$\alpha = T\left(\frac{1}{Z}\left|A - \frac{B+C}{2}\right|\right) \quad \cdots\cdots(10.5)$$

ここで，Zは最大値を1とするための正規化定数である．例えば，符号なし8ビット整数の場合は，$Z = 255$となる．また，$0 < x < 1$に対して$0 < T(x) < 1$とする．変換$T(\cdot)$を2値化変換とすれば，スイッチング動作になる．なお，パラメータαは補間画素ごとに検出できるので，補間処理も画素単位で制御できる．

例題10.4　**動き検出型適応補間**

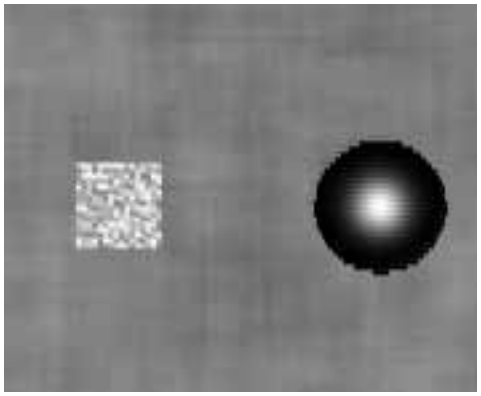
一定の速度$(v_0, v_1)^T = (2, 2)^T$［画素/フィールド］でオブジェクトが並進移動するインターレース映像に対して動き検出型適応補間IP変換を施し，プログレッシブ映像を生成してみよう．また，動きのない$(v_0, v_1)^T = (0, 0)^T$の場合と比較してみよう．

解

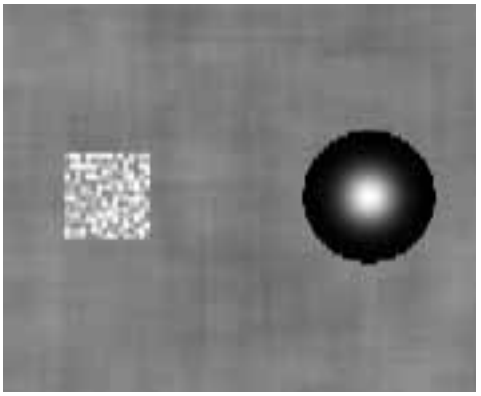
以下にMATLAB上でのコマンド例を示す．ただし，forループの外側の処理は例題10.1の解とほぼ同じになるため，ここでは割愛する．なお，パラメータαの変換を$T(x) = x^{0.25}$と与え，垂直方向補間を優先させている．

```
for iFrame = 1:nFramesIn
  pictureI = frame2im(frameSeq(iFrame));
  %--- Process for picture0
  deintPicture0(2:2:end,:) = ...
    pictureI(2:2:end,:);
  deintPicture0(1:2:end,:) =...% 垂直平均
      1/2 * pictureI(2:2:end,:) ...
    + 1/2 * [pictureI(2,:) ; ...
```

```
            pictureI(2:2:end-2,:)];
    alphaMap0 = abs( (... % 動き検出
      double(previousField) ...
      - double(deintPicture0(1:2:end,:))...
      )/255 );
    alphaMap0 = alphaMap0.^.25; % T(.)
    deintPicture0(1:2:end,:) = uint8(...
      (1-alphaMap0) .* ... % 適応補間
        double(previousField) ...
      + alphaMap0 .* ...
        double(deintPicture0(1:2:end,:)));
    previousField = pictureI(2:2:end,:);
    %--- Process for picture1
    deintPicture1(1:2:end,:) = ...
      pictureI(1:2:end,:);
    deintPicture1(2:2:end,:) =...% 垂直平均
      1/2 * pictureI(1:2:end,:) ...
      + 1/2 * [pictureI(3:2:end,:) ; ...
            pictureI(end-1,:)];
    alphaMap1 = abs( (...   % 動き検出
      double(previousField) ...
      - double(deintPicture1(2:2:end,:))...
      )/255 );
    alphaMap1 = alphaMap1.^.25; % T(.)
    deintPicture1(2:2:end,:) = uint8(...
      (1-alphaMap1) .* ... % 適応補間
        double(previousField) ...
      + alphaMap1 .* ...
        double(deintPicture1(2:2:end,:)));
    previousField = pictureI(1:2:end,:);
    %
    aviObj = addframe(aviObj, ...
      im2frame(deintPicture0,gray(256)));
    aviObj = addframe(aviObj, ...
      im2frame(deintPicture1,gray(256)));
    end
```



(**a**) $(v_0,\ v_1)^T=(2,\ 2)^T$



(**b**) $(v_0,\ v_1)^T=(0,\ 0)^T$

図10.15　動き検出型適応補間の結果

図10.15に処理後のオブジェクト部を拡大して示す．動領域においては，くし状効果の抑圧を確認できる．また，映像からは静止領域のちらつきの抑圧を確認できる．

なお，ここで紹介した動き検出法には問題がある．動きが原因ではない垂直高周波成分も動きとして検出してしまう．結果として，静止領域の垂直高周波成分を過剰に平滑化することになる．前後フィールド間の差分を利用して問題を解消するなど，改善の余地がある．

実習10.4　**動き検出型適応補間**

M-file：practice10_4.m

オブジェクトの移動速度(`velocity`)を変えて試してみよう．また，例題4.9で紹介したインターレース映像`mobile.cif`に対して動き検出型適応補間を施し，プログレッシブ化してみよう．

● 時間垂直メディアン補間

時間垂直メディアン補間では，**図10.16**に示すように，時間方向，垂直方向の周辺画素の中央値(メディアン値)を補間画素値とする．補間画素値が領域の状態に依存して選択される．画素B，Cと大きく異なる値を持てば，前フレームの画素Aが選択されることはない．従って，動きのある場合は時間方向補間が行われないため，動き検出型適応補間と類似の効果を期待できる．

例題10.5　**時間垂直メディアン補間**

一定の速度$(v_0,\ v_1)^T=(2,\ 2)^T$[画素/フィールド]でオブジェクトが並進移動するインターレース映像に対して時間垂直メディアン補間IP変換を施し，プログレッシブ映像を生成してみよう．また，動きのない$(v_0,\ v_1)^T=(0,\ 0)^T$の場合と比較してみよう．

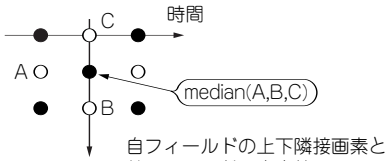
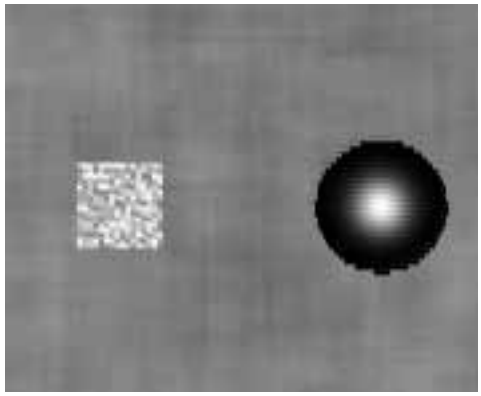


図10.16
時間垂直メディアン補間の例

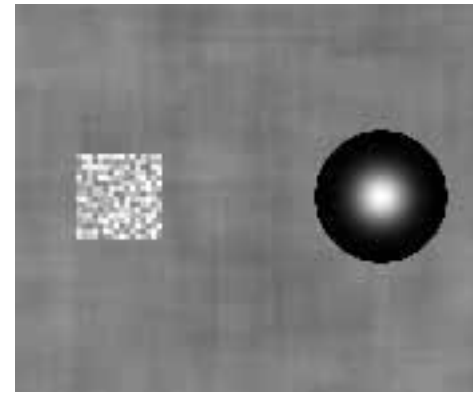
解

以下にMATLAB上でのコマンド例を示す．ただし，for ループの外側の処理は例題10.1の解とほぼ同じになるため，ここでは割愛する．

```
for iFrame = 1:nFramesIn
  pictureI = frame2im(frameSeq(iFrame));
  %--- Process for picture0
  deintPicture0(2:2:end,:) = ...
    pictureI(2:2:end,:);
  deintPicture0(1:2:end,:) = ...
    threeNeighborMedian(...
      pictureI(2:2:end,:),...
      previousField,...
      [ pictureI(2,:) ; ...
        pictureI(2:2:end-2,:)] );
  previousField = pictureI(2:2:end,:);
  %--- Process for picture1
  deintPicture1(1:2:end,:) = ...
    pictureI(1:2:end,:);
  deintPicture1(2:2:end,:) = ...
    threeNeighborMedian(...
      pictureI(1:2:end,:),...
      previousField,...
      [ pictureI(3:2:end,:,:) ; ...
        pictureI(end-1,:,:)] );
  previousField = pictureI(1:2:end,:);
  %
  aviObj = addframe(aviObj, ...
    im2frame(deintPicture0,gray(256)));
  aviObj = addframe(aviObj, ...
```



(**a**) $(v_0,\ v_1)^T=(2,\ 2)^T$



(**b**) $(v_0,\ v_1)^T=(0,\ 0)^T$

図10.17　時間垂直メディアン補間の結果

```
    im2frame(deintPicture1,gray(256)));
end
```

なお，threeNeighborMedian関数を

```
function outputField = ...
  threeNeighborMedian(field1, field2, field3)
%
fieldSize = size(field1);
outputField = median( [field1(:) field2(:) field3(:) ], 2);
outputField = ...
  reshape(outputField,fieldSize);
```

と定義する．**図10.17**に処理後のオブジェクト部を拡大して示す．動領域，静止領域共に良好な結果が得られている．

時間垂直メディアン補間は簡便でよい性能が与えられるが，斜め線や曲線がギザギザに補間されるなど，ジャギの問題がある．

実習10.5　**時間垂直メディアン補間**

M-file：practice10_5.m

オブジェクトの移動速度(velocity)を変えて試してみよう．また，例題4.9で紹介したインターレース映像mobile.cifに対して時間垂直メディアン補間を施し，プログレッシブ化してみよう．

10.3　動き推定

映像の処理に**動き**の解析は欠かせない．動き適応IP変換では動きを検出してこれを利用したが，フレーム間の**動きの量**を利用する動き補償IP変換を利用すると，より良い画質が得られる可能性をもつ．動きの量を利用するためには**動き推定**(ME：Motion Estimation)が必要になるが，これには，

- ブロック・マッチング法
- 階層的ブロック・マッチング法
- 画素再帰法
- オプティカル・フロー法

など，さまざまな手法が存在する．本節では，映像符号化の世界で広く利用されている**ブロック・マッチング法**，特にアルゴリズムが簡単な**全探索ブロック・マッチング法**について解説しよう．

● 全探索ブロック・マッチング法

ブロック・マッチング法は，あるマッチング評価式に従ってフレーム間でブロックごとの類似性を評価し，最も類似したブロックの位置を動きベクトルとする．**図10.18**に概要を示そう．

演算コストや推定精度を改善するために，ブロック・マッチング法にもさまざまなバリエーションが存在する．

- **探索法による違い**：全探索，3ステップ探索，ダイヤモンド探索など
- **マッチング評価による違い**：差分絶対値和，自乗誤差和など

全探索ブロック・マッチング法は演算コストは高いが，最も単純なブロック・マッチング法である．探索窓内の候補ブロックのすべてと現ブロックの間で評価を逐一行う．

マッチング評価関数を$f(\cdot)$，探索窓内の候補ベクトルを$\mathbf{d} = (d_0, d_1)^T$とすると，動きベクトル$\hat{\mathbf{d}} = (\hat{d}_0, \hat{d}_1)^T$は，

$$\hat{\mathbf{d}} = \arg\min_{\mathbf{d}\in S} f(\mathbf{d}) \quad \cdots\cdots (10.6)$$

と表現される．ここでSは探索窓内のすべての候補ベクトルの集合である．式(10.6)は$f(\mathbf{d})$を最小

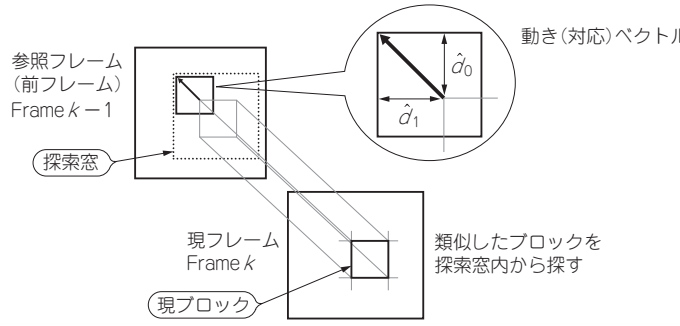


図10.18　ブロック・マッチング法の概要

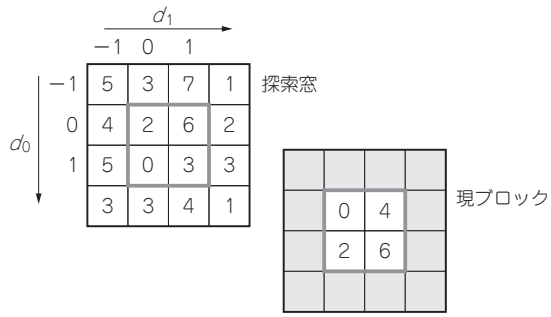


図10.19　ブロック・マッチング法の例

(min)にする引き数(arg)$\mathbf{d} \in S$を動きベクトル$\hat{\mathbf{d}}$とすること意味する．

図10.19にブロック・サイズ2×2，探索範囲$-1 < d_0 < 1$，$-1 < d_1 < 1$の例を示す．このときSは，

$$S = \{\binom{-1}{-1}, \binom{-1}{0}, \binom{-1}{1}, \binom{0}{-1}, \binom{0}{0}, \binom{0}{1}, \binom{1}{-1}, \binom{1}{0}, \binom{1}{1}\}$$

となる．

● マッチング評価

マッチング評価$f(\mathbf{d})$としてよく利用される二つの例を示そう．

(1) 自乗誤差和(SSE)

自乗誤差和(SSE：Sum of Squared Error)は，差分エネルギを評価する式

$$f_{\mathrm{SSE}}(\mathbf{d}) = \sum_{\mathbf{n} \in B} \{x(\mathbf{n}_k) - x(\mathbf{n+d}_{k-\ell})\}^2 \quad \cdots\cdots (10.7)$$

によって定義される．なお，Bは現ブロック内の画素位置の集合，1は現フレームと参照フレームの時間差である．

自乗誤差和は，次に紹介する差分絶対値和に比べて良い品質を与える．一方，2乗の計算を要するため，演算コストが高い．

(2) 差分絶対値和(SAD)

差分絶対値和(SAD：Sum of Absolute Difference)は，

$$f_{\mathrm{SAD}}(\mathbf{d}) = \sum_{\mathbf{n} \in B} |\, x(\mathbf{n}_k) - x(\mathbf{n+d}_{k-\ell}) \,| \quad \cdots\cdots (10.8)$$

と定義される．自乗誤差和に比べて演算コストが低く，LSI実装(集積回路による実現)にも適している．このため，映像符号化を中心に利用されている．

例題10.6　全探索ブロック・マッチング法

図10.19に示す2×2の現ブロックに最も近い2×2のブロックを探索窓内から探し，動きベクトルを推定しよう．ただし，差分絶対値和(SAD)を評価式とした全探索ブロック・マッチング法を利用しよう．

解

左上から順に差分絶対値和$f_{\mathrm{SAD}}(\mathbf{d})$を計算すると，以下のようになる．

$$f_{\mathrm{SAD}}\left(\begin{smallmatrix}-1\\-1\end{smallmatrix}\right)=|0-5|+|4-3|+|2-4|+|6-2|=12$$
$$f_{\mathrm{SAD}}\left(\begin{smallmatrix}0\\-1\end{smallmatrix}\right)=|0-3|+|4-7|+|2-2|+|6-6|=6$$
$$f_{\mathrm{SAD}}\left(\begin{smallmatrix}1\\-1\end{smallmatrix}\right)=|0-7|+|4-1|+|2-6|+|6-2|=17$$
$$f_{\mathrm{SAD}}\left(\begin{smallmatrix}-1\\0\end{smallmatrix}\right)=|0-4|+|4-2|+|2-5|+|6-0|=15$$
$$f_{\mathrm{SAD}}\left(\begin{smallmatrix}0\\0\end{smallmatrix}\right)=|0-2|+|4-6|+|2-0|+|6-3|=9$$
$$f_{\mathrm{SAD}}\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)=|0-6|+|4-2|+|2-3|+|6-3|=12$$
$$f_{\mathrm{SAD}}\left(\begin{smallmatrix}-1\\1\end{smallmatrix}\right)=|0-5|+|4-0|+|2-3|+|6-3|=13$$
$$f_{\mathrm{SAD}}\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)=|0-0|+|4-3|+|2-3|+|6-4|=4$$
$$f_{\mathrm{SAD}}\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)=|0-3|+|4-3|+|2-4|+|6-1|=11$$

最小の差分絶対値和を与える$\mathbf{d}$が動きベクトルの推定値となるので，$\hat{\mathbf{d}}=(0,1)^T$となる．

この動きベクトル$\hat{\mathbf{d}}$は実際の変位と逆を向いている．さらに，探索窓内の最もマッチしたブロックの位置を示しているだけで，実際の動きを反映しているとは限らない．$\hat{\mathbf{d}}$は**対応ベクトル**と呼ぶのがふさわしいが，本書では誤解のないことを前提に動きベクトルという呼び方を用いる．

例題10.7　全探索ブロック・マッチング動き推定

一定の速度$(v_0, v_1)^T = (2, 2)^T$［画素/フィールド］でオブジェクトが並進移動するプログレッシブ映像の隣接する2フレーム間で動き(対応)ベクトル場を推定しよう．

解

以下にMATLAB上でのコマンド例を示す．

```
% 速度ベクトル（垂直，水平）
velocity = [2 2];
% 探索パラメータ（構造体に保持）
me.mbSize = [16 16]; % ブロックサイズ
me.searchRegion = ...
  [ -7 7 ... % 探索範囲（垂直）
    -7 7 ]; % 探索範囲（水平）
% フレーム画像の準備
pictureObj = ...
  imread('./data/squareandgauss.tif');
pictureBg = ...
  imread('./data/background.tif');
% 参照フレームの設定
```

```
  referenceFrame = pictureObj + ...
    uint8(not(pictureObj)) .* pictureBg;
  % 現フレームの設定
  pictureObj = ...
    circshift(pictureObj, velocity);
  currentFrame = pictureObj + ...
    uint8(not(pictureObj)) .* pictureBg;
  me.frameSize = ... % フレーム・サイズ
    size(currentFrame);
  % 動きベクトル場の推定
  mvField = fullSearchBlockMatchingMe(...
    referenceFrame, currentFrame, me);
  % 現フレームの表示（動きベクトル場が見やすいよう加工）
  imshow(... % もしくは，imshowcq(...
    (double(currentFrame)/510+.5).^.25)
  hold on
  % 動きベクトル場の表示
  [n0,n1] = ndgrid(...
    me.mbSize(1)/2:me.mbSize(1):...
      me.frameSize(1),...
    me.mbSize(2)/2:me.mbSize(2):...
      me.frameSize(2));
  d0 = mvField(:,:,1);
  d1 = mvField(:,:,2);
  quiver(n1,n0, ...
    -d1,-d0,... % 対応ベクトルを反転表示
    'LineWidth',2)
  hold off
```

図10.20に結果を示そう．移動オブジェクトの周辺に動き（対応）ベクトルが検出されている様子が分かる．

上のコマンド例では動きベクトル場の推定に，`fullSearchBlockMatchingMe`関数を利用している．以下にプログラムを示そう．


```
function mvField = ...
```

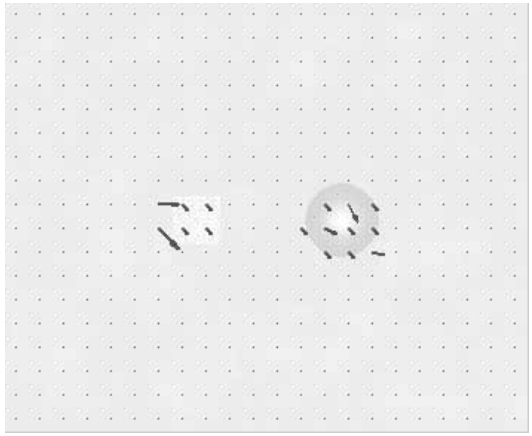


図10.20　全探索ブロック・マッチング動き推定の結果

```
  fullSearchBlockMatchingMe(...
    referenceFrame, currentFrame, me)
% 動きベクトル場の探索開始
iMbRow = 1;    % ブロックの行インデックス
for iRow = 1:me.mbSize(1):me.frameSize(1)
  iRowTop = iRow;
  iRowBottom = iRowTop + me.mbSize(1) - 1;
  iMbCol = 1; % ブロックの列インデックス
  for iCol = 1:me.mbSize(2):me.frameSize(2)
    iColLeft = iCol;
    iColRight = iColLeft + me.mbSize(2) - 1;
    % 現ブロックの切り出し
    currentBlock = currentFrame(...
      iRowTop:iRowBottom, ...
      iColLeft:iColRight);
    % 動きベクトル場の推定
    mvField(iMbRow,iMbCol,1:2) = ...
      subSearch(...
        referenceFrame, ...
        currentBlock,...
        [iRowTop iColLeft], me);
    iMbCol = iMbCol + 1;
  end
```

```
  iMbRow = iMbRow + 1;
end
%% ブロックごとの処理（サブ関数）
function mv = subSearch(...
  referenceFrame, currentBlock, ...
  iTopLeft, me)
%
iRowTop = iTopLeft(1);
iColLeft = iTopLeft(2);
% 動きベクトルの探索開始
minimumSad = Inf; % ∞に初期化
for d0 = me.searchRegion(1):me.searchRegion(2)
  iRowTopInRefFrame = iRowTop + d0;
  iRowBottomInRefFrame = ...
    iRowTopInRefFrame + me.mbSize(1) - 1;
  for d1 = me.searchRegion(3):me.searchRegion(4)
    iColLeftInRefFrame = iColLeft + d1;
    iColRightInRefFrame = ...
      iColLeftInRefFrame + me.mbSize(2) - 1;
    % 探索範囲をフレーム内に限定
    if ( iRowTopInRefFrame > 0 ...
      && iColLeftInRefFrame > 0 ...
      && iRowBottomInRefFrame ...
        <= me.frameSize(1) ...
      && iColRightInRefFrame ...
        <= me.frameSize(2) )
      % 候補ブロックの切り出し
      candidateBlock = ...
        referenceFrame(...
          iRowTopInRefFrame:...
          iRowBottomInRefFrame,...
            iColLeftInRefFrame:...
            iColRightInRefFrame);
        currentSad = ...% 差分絶対値和
          sum(abs( currentBlock(:) - candidateBlock(:) ));
        if (currentSad < minimumSad)
```

```
            mv = [d0 d1]; % 更新
            minimumSad = currentSad;
          elseif ( d0 == 0 && d1 == 0 &&...
                currentSad == minimumSad )
            % 評価が一致した場合[0 0] を優先
            mv = [0 0];
          end
        end
      end
    end
```

この関数は，

- 与えられた現フレーム（currentFrame）を重複のない16 × 16のブロックに分割
- それぞれのブロック（currentBlock）に対して，参照フレーム（referenceFrame）内に $-7 \leqq d_0 \leqq 7$, $-7 \leqq d_1 \leqq 7$ の探索範囲（searchRegion）を設定
- 動きベクトル場の推定

を行う．

実習10.6　**全探索ブロック・マッチング法**

M-file：practice10_6.m

オブジェクトの移動速度（velocity）やブロック・サイズ（me.mbSize），探索範囲を（me.searchRegion）を変えて，動きベクトル場の推定を試してみよう．なお，ブロック・サイズはフレームを整数分割できるように設定しよう．

(3) 推定方向

これまでに解説した動き推定では，過去のフレームに探索窓を設定した．フレーム・バッファを用いると，現在のフレームよりも後の時刻のフレーム，すなわち未来のフレームに探索窓を設定することもできる．**図10.21** に示すように，動きベクトルは，過去のフレームを参照した場合は**前方ベクトル**（Forward Vector），未来のフレームを参照した場合は**後方ベクトル**（Backward Vector）と呼ばれる．後方ベクトルを導入すると動き補償の性能が向上する．そのため，後方ベクトルは多くの映像符号化において採用されている．動き補償については次節で解説する．

(4) 推定精度

動きベクトルを整数位置に制限する必要はない．より細かな動きを推定して利用することで，映像符号化における性能の向上が確認されている．例えば，半画素精度（Half-pel Precision）の動き推定は次のように実現できる．

1. 整数精度にて動きベクトルを推定

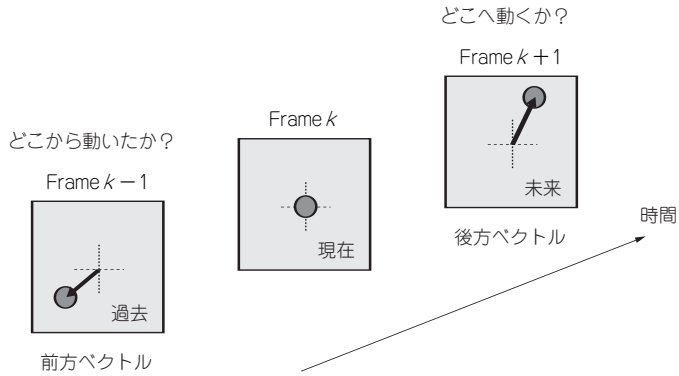


図10.21　推定方向の例

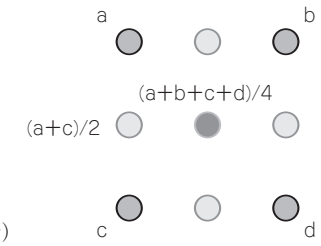


図10.22
参照画素の補間処理
（a，b，c，dは整数位置の画素）

2. 1.で得られた最適なブロックとその境界部を**図10.22**のように線形補間
3. 2.で得られた補間画素からなる周辺8ブロックについてマッチング評価を行い，半画素精度の動きベクトルを推定

本書では，以降，整数精度の動き推定のみを取り扱う．詳細については，ほかの文献を参照されたい．

● 動き補償予測

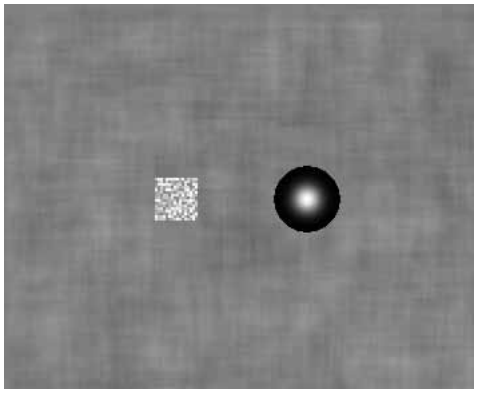
動き補償予測（Motion-compensated Prediction）では，動き推定によって得られた動きベクトル場と参照フレームから現フレームの予測画像を生成する．映像符号化の世界では，この予測画像が，時間方向の冗長性を圧縮する際に利用されている．後述するように，動き補償予測はIP変換にも利用できる．

例題10.8　**動き補償予測**

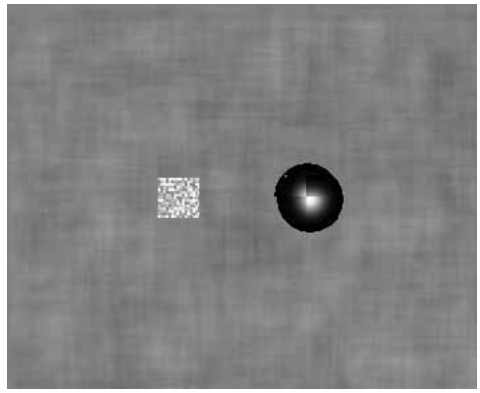
例題10.7の動きベクトル場と参照フレームから動き補償予測フレームを生成し，現フレームと比較してみよう．

解

以下に，MATLAB上でのコマンド例を示す．ただし，動きベクトル場の推定より前の部分は例題10.7の解と同じため，ここでは割愛する．



（a）現フレーム



（b）動き補償予測フレーム

図10.23　動き補償予測のシミュレーション結果

```
% 動きベクトル場の推定
mvField = fullSearchBlockMatchingMe(...
  referenceFrame, currentFrame, me);
% 動き補償予測
predictedFrame = mcPrediction(...
  referenceFrame, mvField, me);
% 現フレームの表示
figure(1)
imshow(currentFrame) % もしくは，imshowcq
title('Current frame')
% 動き補償予測フレームの表示
figure(2)
imshow(predictedFrame) % もしくは，imshowcq
title('Predicted frame')
```

図10.23(**a**)に現フレームを，**図10.23**(**b**)に動き補償予測フレームを示す．過去のフレームの移動量が補償され，現フレームに似た画像が与えられる．

上のコマンド例では，動き補償予測にmcPrediction関数を利用している．この関数は，参照フレーム内から動きベクトルで指定される最適なブロックを抽出し，タイルのように並べた画像を予測フレームとする．以下にプログラムを示そう．


```
function predictedFrame = ...
```

```
  mcPrediction(referenceFrame, mvField, me)
%
frameSize = me.frameSize;
mbSize = me.mbSize;
% 動き補償開始
iMbRow = 1;
for iRow = 1:mbSize(1):frameSize(1)
  iRowTop = iRow;
  iRowBottom = iRowTop + mbSize(1) - 1;
  iMbCol = 1;
  for iCol = 1:mbSize(2):frameSize(2)
    iColLeft = iCol;
    iColRight = iColLeft + mbSize(2) - 1;
    % 動きベクトルの参照
    d = mvField(iMbRow,iMbCol,1:2);
    % 予測ブロックの位置の計算
    iRowTopInRefFrame = ...
      iRowTop + d(1);
    iRowBottomInRefFrame = ...
      iRowTopInRefFrame + mbSize(1) - 1;
    iColLeftInRefFrame = ...
      iColLeft + d(2);
    iColRightInRefFrame = ...
      iColLeftInRefFrame + mbSize(2) - 1;
    % 予測ブロックの切り出し
    predictedBlock = referenceFrame(...
      iRowTopInRefFrame:...
      iRowBottomInRefFrame,...
      iColLeftInRefFrame:...
      iColRightInRefFrame);
    % 予測フレームの生成
    predictedFrame(...
      iRowTop:iRowBottom, ...
      iColLeft:iColRight) ...
      = predictedBlock;
    iMbCol = iMbCol + 1;
```

```
    end
    iMbRow = iMbRow + 1;
  end
```

実習10.7　**動き補償予測**

M-file：practice10_7.m

オブジェクトの移動速度(velocity)やブロック・サイズ(me.mbSize)，探索範囲(me.searchRegion)を変えて，動き補償予測を試してみよう．なお，ブロック・サイズはフレームを整数分割できるように設定しよう．

● 動き推定の問題

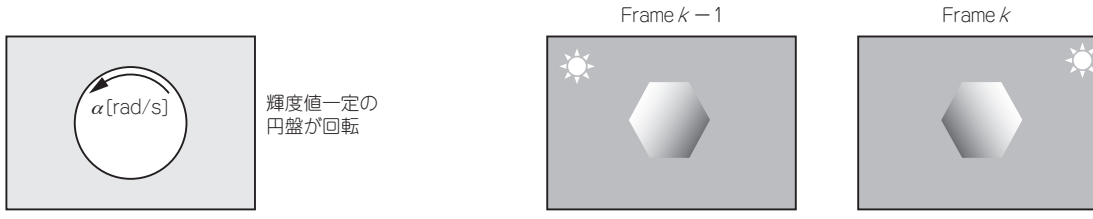
映像の動き推定は本質的に困難である．本来，物体や画面の動きを知りたいのだが，実際は**見かけ上の動き**(Apparent Motion)で代用せざるを得ない．例えば，物体の変位ベクトルは輝度値の変位(対応ベクトル)で代用し，物体の速度ベクトルは輝度値の変化(オプティカル・フロー)で代用する．これら見かけ上の変位や速度は，実際の変位や速度と異なり，

- 空間こう配の欠落(**図10.24**(**a**))
- 光源変化の影響(**図10.24**(**b**))

を前提として利用しなければならない．

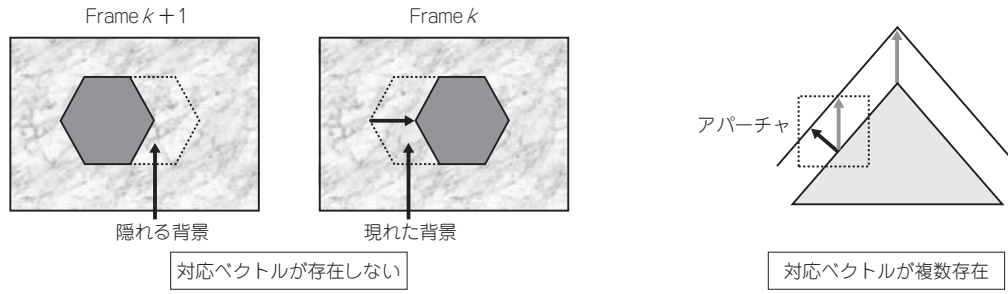
また，3次元空間内での物体の動きを2次元平面上で推定する際，対応ベクトルが必ずしも存在しないという**オクルージョン**(Occlusion)問題がある(**図10.25**(**a**))．さらに，ブロック・サイズの大きさも重要である．物体に対してブロック・サイズが小さいと，**図10.25**(**b**)に示すように対応ベクトルが複数存在し，結果として誤った推定が行われてしまう．これを**アパーチャ**(Aperture)問題という．

映像符号化への応用では，実際の動きと推定した動きが異なっていても，現フレームと予測フレームの誤差が小さければ問題ない．しかし，後述する動き補償IP変換など，動き補償フィルタリングへの応用では，より正確な動き推定が望ましい．本書では取り扱わないが，この問題の改善策として，階層的ブロック・マッチング(HBM：Hirarchical Block Mathiching)の利用などが挙げられる．



(**a**) 空間こう配の欠落　　(**b**) 光源変化の影響

図10.24　実際の動きと見かけ上の動きの違い



（**a**）オクルージョン問題　　（**b**）アパーチャ問題

図10.25　動き推定における問題

10.4　動き補償IP変換

動き補償IP変換では動き補償により前フィールドの予測を与え，これを現フィールドの補間に利用する．

● 固定係数IP変換との関係

動き補償は動作領域の動きを止めることと同じ効果があり，動きの軌跡に沿った補間を与える．例えば，前フィールドを動き補償予測フィールドに置き換えて時間方向補間を行う操作は，次の補間フィルタを利用することに相当する．

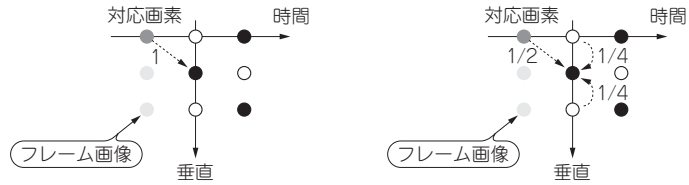
$$G_{\hat{\mathbf{d}}}(\mathbf{z}) = 1 + z_0^{-\hat{d}_0} z_1^{-\hat{d}_1} z_T^{-1} \quad \cdots\cdots (10.9)$$

ここで$\hat{d}_0$，$\hat{d}_1$は，それぞれ推定された動きベクトルの垂直方向成分と水平方向成分である．**図10.26**（**a**）に概念図を示そう．また，時間方向補間の代わりに時間垂直補間を行う操作は，次の補間フィルタを利用することに相当する．

$$G_{\hat{\mathbf{d}}}(\mathbf{z}) = 1 + \frac{1}{2} z_0^{-\hat{d}_0} z_1^{-\hat{d}_1} z_T^{-1} + \frac{1}{4}(z_0^1 + z_0^{-1}) \quad \cdots\cdots (10.10)$$

図10.26（**b**）に概念図を示す．

動き補償予測フィールドが正確に与えられれば，時間方向補間は動作領域においても空間解像度を高く保持できる．一方，動き推定の誤りが目立ち，誤差が後のフレームに波及する問題がある．



（**a**）時間方向補間　　（**b**）時間垂直補間

図10.26　動き補償IP変換の概念（$\hat{d}_0 = -1$，$\hat{d}_1 = 0$の場合）

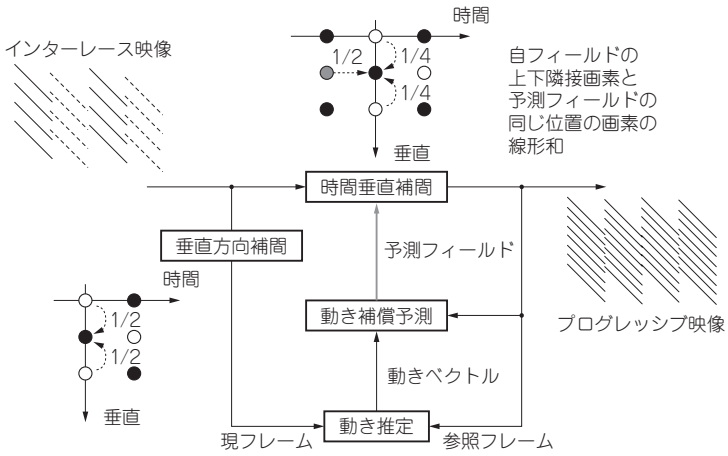


図10.27　動き補償時間垂直補間IP変換の構成例

時間垂直補間は空間解像度を犠牲にするが，動き推定の誤りを保護する効果が得られる．

● 動き補償時間垂直補間

本節では，動き補償時間垂直補間によるIP変換について調べてみよう．**図10.27**に動き補償時間垂直補間IP変換の構成例を示す．一般に，IP変換においては，入力がインターレース映像のためフィールド単位の動き推定に工夫を要する．**図10.27**の構成例では，過去の出力結果が理想的にIP変換(デインターレース処理)されたものと仮定して，これを参照フレームとしている．また，現フレームをフィールドの垂直方向補間で与え，動き推定と動き補償予測を行う．

例題10.9　**動き補償時間垂直補間**

一定の速度$(v_0, v_1)^T = (2, 2)^T$［画素/フィールド］で並進移動するインターレース映像に対して，ブロック・サイズ16×16および32×32の全探索ブロック・マッチング法を用いた動き補償時間垂直補間IP変換を施し，それぞれプログレッシブ映像を生成してみよう．

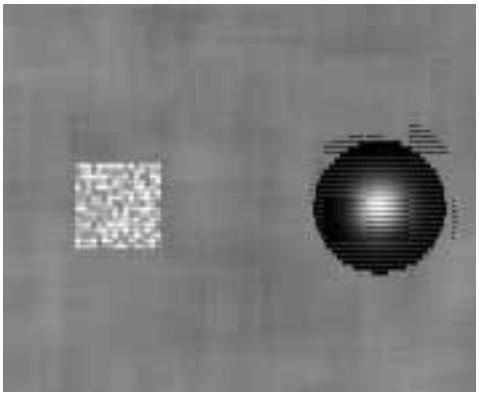
解

以下にMATLAB上でのコマンド例を示す．ただし，forループの外側の処理は例題10.1の解とほぼ同じであるため，割愛する．なお，ブロック・サイズなど，動き推定のパラメータ設定を構造体meに保持しておくことも必要である．例題10.6を参照されたい．

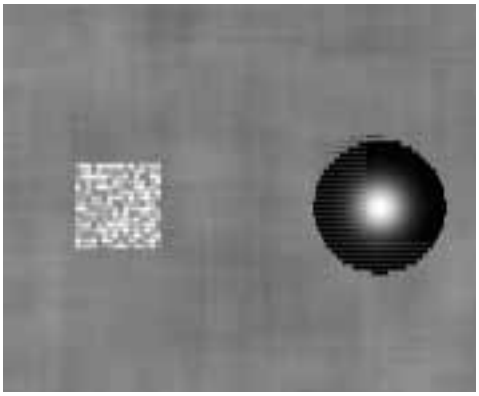
```
for iFrame = 1:nFramesIn
  pictureI = frame2im(frameSeq(iFrame));
  %--- Process for picture0
  deintPicture0(2:2:end,:) = pictureI(2:2:end,:);
  % 垂直方向補間
  deintPicture0(1:2:end,:) = ...
```

```
    1/2 * pictureI(2:2:end,:)...
    + 1/2 * [ pictureI(2,:) ;...
            pictureI(2:2:end-2,:)];
  mvField = ...   % 動き推定
    fullSearchBlockMatchingMe(...
      deintPicture1, deintPicture0, me);
  predictedPicture = ... % 動き補償予測
    mcPrediction(deintPicture1, mvField, me);
  % 時間垂直補間
  deintPicture0(1:2:end,:) = ...
    1/2 * predictedPicture(1:2:end,:) ...
    + 1/2 * deintPicture0(1:2:end,:);
  %--- Process for picture1
  deintPicture1(1:2:end,:) = pictureI(1:2:end,:);
  % 垂直方向補間
  deintPicture1(2:2:end,:) = ...
    1/2 * pictureI(1:2:end,:)...
    + 1/2 * [ pictureI(3:2:end,:) ;...
            pictureI(end-1,:)];
  mvField = ... % 動き推定
    fullSearchBlockMatchingMe(...
      deintPicture0, deintPicture1, me);
  predictedPicture = ... % 動き補償予測
    mcPrediction(deintPicture0, mvField, me);
  % 時間垂直補間
  deintPicture1(2:2:end,:) = ...
    1/2 * predictedPicture(2:2:end,:)...
    + 1/2 * deintPicture1(2:2:end,:);
  %
  aviObj = addframe(aviObj, ...
    im2frame(deintPicture0,gray(256)));
  aviObj = addframe(aviObj, ...
    im2frame(deintPicture1,gray(256)));
end
```


図 10.28 にブロック・サイズ 16 × 16，32 × 32 の場合について，処理後のオブジェクト部を拡大



（a）サイズ16×16



（b）サイズ32×32

図10.28　動き補償時間垂直補間の結果（$(v_0, v_1)^T = (2, 2)^T$）

して示す．**図10.28**(**a**)については，動き推定の誤りが視覚的に強調され，好ましくない結果となっている．一方，**図10.28**(**b**)については，**図10.11**(**a**)の結果と比較すると分かるとおり，動き補償の導入により，くし状効果の抑圧が達成されている．

動き補償技術では，動き推定が良好に働くと優れた性能を達成できる一方で，動き推定が誤ったときの悪影響は大きい．動き推定の正確さの向上に加えて，各種補間技術の切り替えをいかに実現するかが，IP変換をはじめとする動き補償フィルタリング全般の課題といえる．

実習10.8　**動き補償時間垂直補間**

M-file：practice10_8.m

オブジェクトの移動速度(`velocity`)やブロック・サイズ(`me.mbSize`)，探索範囲を(`me.searchRegion`)を変えて，動き補償時間垂直補間を試してみよう．なお，ブロック・サイズはフレームを整数分割できるように設定しよう．また，例題4.9で紹介したインターレース映像`mobile.cif`をプログレッシブ化してみよう．

章末問題

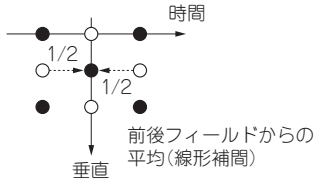
問題10.1　**補間フィルタの周波数振幅応答**(MATLAB演習)

式(10.1)，(10.2)，(10.3)で与えられる各種固定係数補間フィルタ$G(\mathbf{z})$の周波数振幅応答を`freqz2`関数で確認してみよう．また，式(10.4)に示される可変係数補間フィルタ$G_\alpha(\mathbf{z})$の周波数振幅応答についてもαを与えて確かめてみよう．

問題10.2　**時間平均補間**(MATLAB演習)

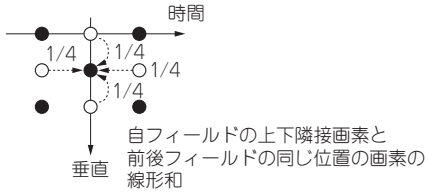
任意のインターレース映像に対して下図に示す時間平均補間IP変換を施し，プログレッシブ映像

を生成してみよう．



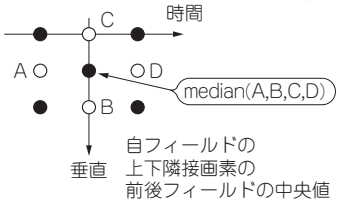
問題10.3　**4近傍時間垂直補間**(MATLAB演習)

任意のインターレース映像に対して下図に示す4近傍時間垂直補間IP変換を施し，プログレッシブ映像を生成してみよう．



問題10.4　**4近傍時間垂直メディアン補間**(MATLAB演習)

任意のインターレース映像に対して下図に示す4近傍時間垂直メディアン補間IP変換を施し，プログレッシブ映像を生成してみよう．なお，4点のうち中央2点の平均値が中央値となる．



問題10.5　**動き検出パラメータ**(MATLAB演習)

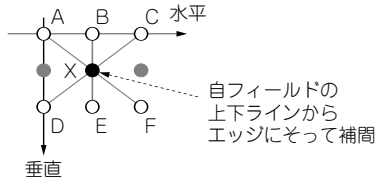
例題10.4のパラメータ α のマップをフィールドごとに映像として表示し，動領域と静領域の検出が行われているかを確認してみよう．

問題10.6　**エッジ適応空間補間**(MATLAB演習)

任意のインターレース映像に対してエッジ適応空間補間IP変換を施し，プログレッシブ映像を生成してみよう．ただし，エッジ適応空間補間は下図に示す X の位置の補間画素を

$$X=\begin{cases}\dfrac{A+F}{2} & (|A-F|<|C-D|)\wedge(|A-F|<|B-E|)\\ \dfrac{C+D}{2} & (|C-D|<|A-F|)\wedge(|C-D|<|B-E|)\\ \dfrac{B+E}{2} & \text{その他}\end{cases}$$

と与える．エッジを保存するために空間こう配の低い方向に平均処理を施す．



問題10.7　**自乗誤差和マッチング評価**（MATLAB演習）

マッチング評価式を差分絶対値和（SAD）から自乗誤差和（SSE）に置き換えて，例題10.6～10.9と同じ内容を試してみよう．

問題10.8　**動き補償補間フィルタの周波数振幅応答**（MATLAB演習）

式(10.9)，(10.10)で与えられる補間フィルタ$G_{\hat{\mathbf{a}}}(\mathbf{z})$の周波数振幅応答を確認してみよう．$\hat{d}_1 = 0$とし，$\hat{d}_0$のみを変えて時間垂直周波数振幅応答を`freqz2`関数で表示すればよい．

問題10.9　**動き補償時間方向補間**（MATLAB演習）

図10.27の時間垂直補間を時間方向補間に置き換えて，例題10.9と同じ内容を確かめてみよう．

問題10.10　**動き補償時間垂直メディアン補間**（MATLAB演習）

図10.27の時間垂直補間を時間垂直メディアン補間に置き換えて，例題10.9と同じ内容を確かめてみよう．

参考文献

(1) 吹抜敬彦；画像・メディア工学，コロナ社，2002年．
(2) Gerard De Haan and Erwin B. Bellers；Deinterlacing - An Overview，Proc. of IEEE，vol.86，no.9，pp.1839-1857，Sep 1998.
(3) E.B. Bellers and G. De Haan ；De-interlacing: A Key Technology for Scan Rate Conversion ，Elsevier Science，2000.
(4) Yao Wan, J orn Ostermann Ya-Qin Zhang；Video Processing and Communications，Prentice Hall，2002.
(5) John W. Woods；Multidimensional Signal, Image, and Video Processing and Coding，Academic Press，2006.

索　引

【か・カ行】

【さ・サ行】

【た・タ行】

【な・ナ行】

【は・ハ行】

【ま・マ行】

【や・ヤ行】

【ら・ラ行】

【わ・ワ行】

著者紹介

村松 正吾（むらまつ・しょうご）
新潟大学工学部 電気電子工学科 准教授

1991年 3月　東京工業高等専門学校 電子工学科 卒業
1993年 5月　東京都立大学工学部 電気工学科 卒業
1995年 3月　同大学大学院工学研究科 電気工学専攻 修士課程 修了
1996年12月　同大学大学院工学研究科 電気工学専攻 博士課程 中退
1997年 1月　同大学工学部 電子・情報工学科 助手
1999年10月　新潟大学工学部 電気電子工学科 助手
2001年 2月　同大学工学部 電気電子工学科 助教授
2003年10月～2004年9月　イタリア フローレンス大学 文部科学省在外研究員
現在に至る

学位：
博士（工学）［1998年11月 東京都立大学］

主な研究：
信号処理，特に映像符号化，映像解析，LSIアーキテクチャの研究に従事．

主な著書：
貴家仁志，村松正吾；マルチメディア技術の基礎DCT入門，CQ出版社，1997年．

MATLABによる画像＆映像信号処理

2007年5月20日　初版発行

著　者　村松 正吾
発行人　山本　潔
発行所　CQ出版株式会社
〒170-8461 東京都豊島区巣鴨1-14-2
☎ 03-5395-2124（販売部）
☎ 03-5395-2148（編集部）
振替　00100-7-10665

乱丁・落丁本はお取り替えいたします．
定価はカバーに表示してあります．
JAN9784789800000

編集担当　中山 俊一
DTP・印刷・製本　美和印刷㈱
Printed in Japan

MATLAB による画像＆映像信号処理

2007 年 5 月 20 日　初版発行
2011 年 7 月 1 日　電子版 初版発行

著　者　村松 正吾
発行所　CQ 出版株式会社

編集担当　　中山 俊一
電子版担当　中山 俊一