FFTによって得られるスペクトルのリストは、周波数とそのエネルギーから成り立っている。この分析結果は実際のスペクトルの近似でしかあり得ないと説明した。これにIFFTを実行しても本当に元波形が再合成できるかどうか疑わしく思える。しかし、この周波数とエネルギーのリストには実のところ元になる波形の振幅と位相とさらに時間要素が複雑に織り込まれている。

30分のシンフォニーに対して、30分の長さのFFTサイズでFFTを行い、1個の長大なスペクトルのリストを得たとしよう。スペクトルは膨大な数の周波数ビンで分割され、そこに30分間に起こったすべての周波数とそのエネルギーが反映される。冒頭の通奏低音の重い響きも、中間部のきらびやかな金管楽器の音も、クライマックスのシンバルの強打もすべてである。いくらスペクトルのリストを眺めてみても、シンバルの強打がいつ起こったのかを知ることはできない。しかし、リストには元になる波形の振幅と位相とさらに時間要素が複雑に織り込まれている。それに対してIFFTを実行すれば、元のシンフォニーが再現されるのだ。

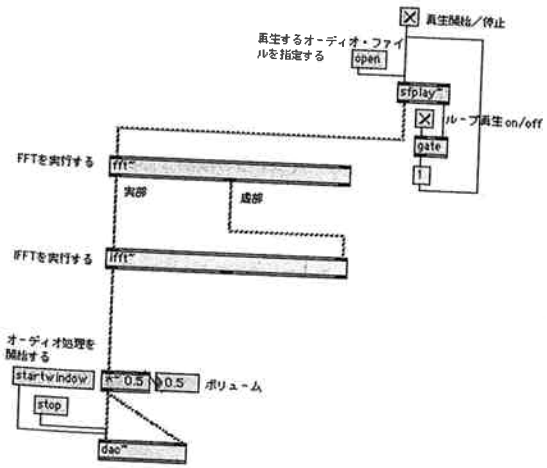
5-18 FFT：FFT関連のMSPオブジェクト

「5-17 FFT：FFTの基礎」でフーリエ変換ならびにFFTについてその基本的な考え方を解説した。ここからは、実際にMSPによるFFTプログラミングの内容を具体的に紹介したい。まず、FFT関連のオブジェクトと、それらの基本的な使用法について触れ、5-17での解説内容がMSPの中でどのように実現しているのかを説明する。

fft~オブジェクトとifft~オブジェクト

MSPではFFTを実行するオブジェクトとしてfft~、その逆のIFFTを実行するオブジェクトとしてifft~ を用意している。fft~オブジェクトが分析対象とするデジタル信号は、MSPではもちろんシグナルである。5-18-1のパッチは、sfplay~オブジェクトから出力されるシグナル（オーディオ・シグナル）をfft~オブジェクトによってスペクトル解析し、その結果をifft~オブジェクトによって再びオーディオ・シグナルに合成して外部オーディオ出力している例だ。

■5-18-1 fft~オブジェクトとifft~オブジェクトの基本的な使用法

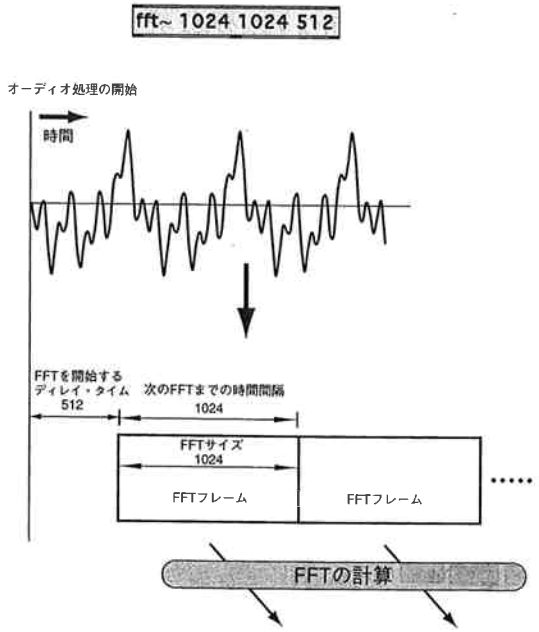


ここで、あえてオーディオ・シグナルという言葉を使ったのは、fft~オブジェクトが出力するのもシグナルだからだ。fft~オブジェクトは、FFTのスペクトル解析結果の実部を第1アウトレットから、虚部を第2アウトレットからシグナルで出力する。これは、各周波数ビンにおけるエネルギーのリストであって、直接dac~オブジェクトにつないでサウンドとして聞くものではない特殊なシグナルになる。

この状態でオーディオ処理を開始し、sfplay~オブジェクトの再生を行うと、その再生サウンドが聞こえるだろう。fft~オブジェクトやifft~オブジェクトは何の処理も行わず、単に入力シグナルを通過させているだけのように見えるかもしれないが、実際にはFFTとIFFTを実行している。

fft~オブジェクトとifft~オブジェクトはともにアーギュメントとして3つの数値をとる。これらの意味は共通していて、それぞれ次の内容を示している。

■5-18-2 fft~オブジェクトのアーギュメント設定とFFT処理の関係



FFTサイズ　次のFFTまでの時間間隔　FFTを開始するディレイ・タイム

これらはすべてサンプル単位の整数で指定し、必ず2のべき乗の数でなければならない。もし、アーギュメントとして何も書き込まなければ、fft~オブジェクトとifft~オブジェクトは初期状態で、FFTサイズ=512、次のFFTまでの時間間隔=512、FFTを開始するディレイ・タイム=0を採用する。例えば、fft~オブジェクトのアーギュメントとして1024 1024　512を書いた場合、どのようなFFT処理が行われるのかを図示すると5-18-2のようになる。

fft~オブジェクトの処理結果

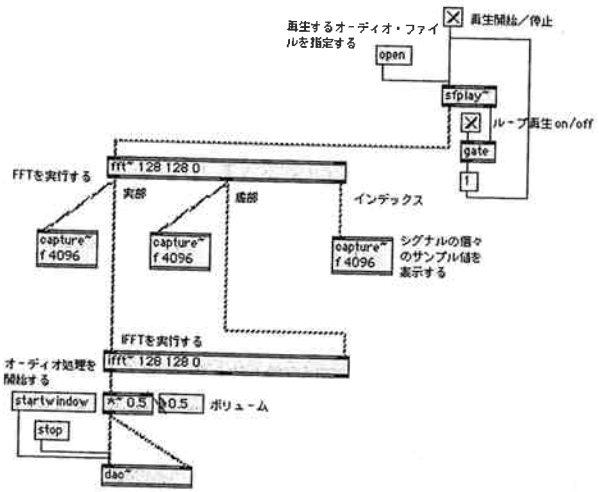
fft~オブジェクトが入力シグナルのスペクトル解析を行っていることを確認してみよう。5-18-3のパッチで、capture~オブジェクトは、受け取ったシグナルの個々のサンプル値を格納し、ダブル・クリックして開くテキスト・ウィンドウでその内容をテキストとして表示してくれる。アーギュメントとして、fを書けば、オーディオ処理開始直後の最初の入力シグナルを格納・表示する。fを書かなければ、capture~オブジェクトは次々と新しく受け取るシグナルで格納内容を更新し、テキスト・ウィンドウを開く時点で格納している内容を表示する。続くアーギュメントではデータの個数を指定する。ここではf 4096としているので、オーディオ処理開始後に受け取る最初の4,096サンプルの値を格納・表示してくれることになる。なお5-18-3では表示を分かりやすくするために、あえてfft~オブジェクトのアーギュメントでfft~オブジェクト・サイズを128という小さな数にした。

再生するオーディオ・ファイルを指定してsfplay~オブジェクトの再生を開始しておいてから、オーディオ処理を開始してみよう。

まず一番左のcapture~オブジェクトをダブル・クリックしてfft~オブジェクトの第1アウトレットの出力シグナルを確認してみよう。これはfft~オブジェクトの処理結果のうち実部の内容にあたり、5-18-4のようになっている。capture~オブジェクトのテキスト・ウィンドウは、DSP Statusウィンドウで設定するシグナル・ベクター・サイズに応じて段が区切られる。ここではシグナル・ベクター・サイズを128にしているので、128個ずつのデータにグルーピングされている(ただし最初のグループだけ1つ分多い)。

最初の128個のサンプル値はすべて0になっている。これはfft~オブジェクトが常に

■5-18-3 capture~オブジェクトによりfft~オブジェクトの出力シグナルを確認する

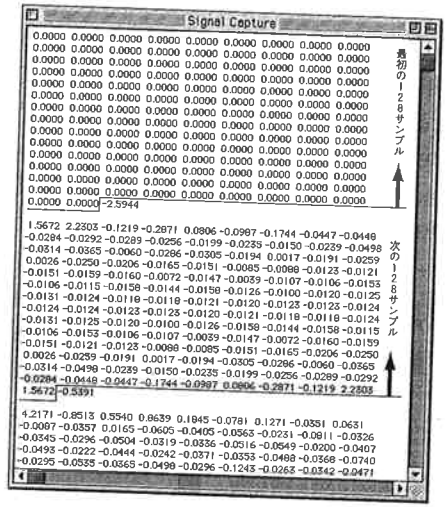


FFTサイズのサンプルを受け取ってからFFTの計算を行うためである。オーディオ処理を開始した直後にはfft~オブジェクトはまだ何もシグナルを受け取っていない。このため最初の128個の数値はすべて0になっている。

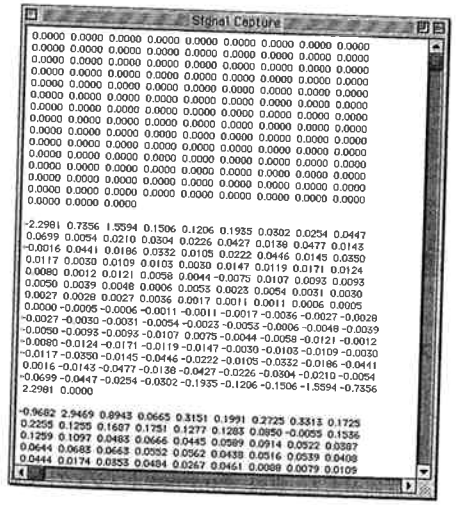
続くデータのグループは、FFTフレームごとのFFT計算結果を示している。つまりサンプリング・レートを128分割した各周波数ビンごとのエネルギーの大きさである。ナイキスト・レートを境にして数値が折り返されていることも確認できる。

続いて、同様にfft~オブジェクトの第2アウトレットの出力シグナルを確認してみよう。これはfft~オブジェクトの処理結果のうち虚部の内容にあたり、5-18-5のようになっている。ここでも最初の128個のサンプル値はすべて0になっている。また、ナイキスト・レートを境にして数値が折り返されていることも確認できる。ただし虚部の場合、この折り返しで符号も逆になっている。

■5-18-4 fft~オブジェクトの第1アウトレット（実部）の出力シグナル

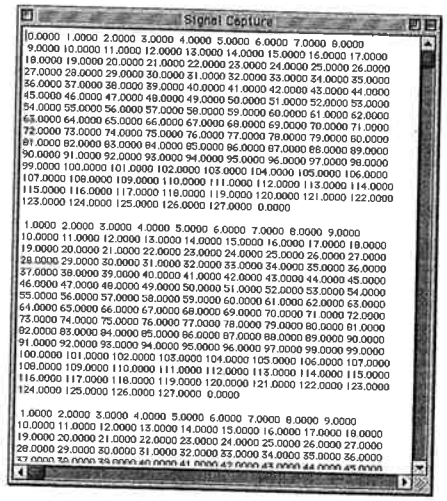


■5-18-5 fft~オブジェクトの第2アウトレット（虚部）の出力シグナル



次に、fft~オブジェクトの第3アウトレット(虚部)の出力シグナルの出力シグナルを確認してみよう。これはfft~オブジェクトが実行するFFTのインデックスにあたり、0～(FFTサイズ－1)の数値を繰り返し出力する。つまりFFTサイズを128にした場合、fft~オブジェクトは128個の入力サンプルをひとまとまり(FFTフレーム)として計算を実行し、その結果を128個のサンプルとして出力する。この1サンプルずつの処理について第3アウトレットから0～127までのインデックスを繰り返し出力していることになる。FFT計算のカウンターと考えてもよい。インデックスは、fft~オブジェクトの出力シグナルをスペクトル・レベルで窓かけする際に、窓関数を同期させる場合など、非常に便利なものだ。使用法については後述する。

■5-18-6 fft~オブジェクトの第3アウトレット(インデックス)の出力シグナル



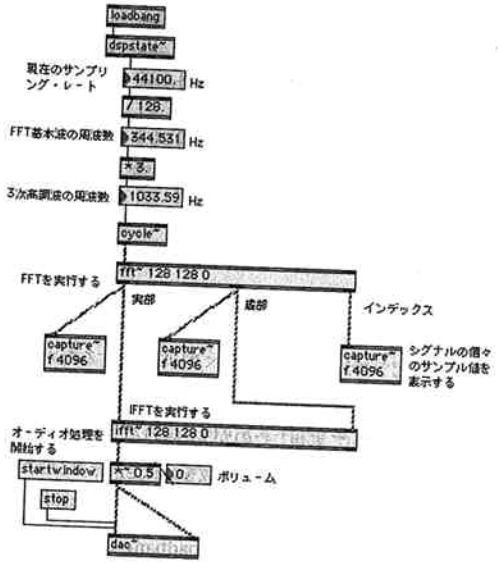
fft~オブジェクトをテストする

5-18-3のパッチではfft~オブジェクトでスペクトル解析されるシグナルは、sfplay~オブジェクトによって再生するオーディオ・ファイルから来るものだった。そのため、それは任意の周波数を含んでいる可能性がある。実際、capture~オブジェクトで確認した

fft~オブジェクトの計算結果は、すべての周波数ビンにエネルギーが分散し、おそらくは元々のシグナルに含まれていない周波数成分を誤って分析したものになっていると想像できる。これを改善するために窓関数とオーバーラッピングというテクニックを使うわけであるが、その前に実際にfft~オブジェクトが正確に動作しているのかを確認しておこう。これを調べるにはFFTの基本波の周波数を求め、その整数倍の周波数を持つサイン波シグナルをfft~オブジェクトで解析してみるのが最も簡単だ。

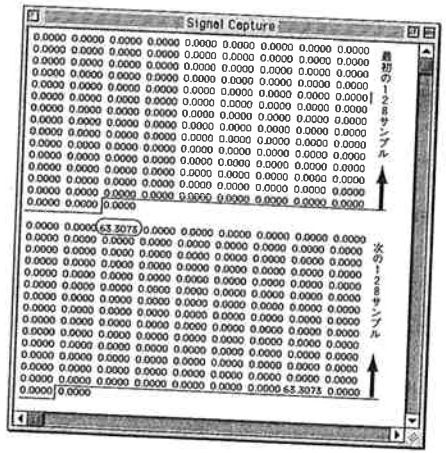
5-18-7のパッチは、テスト用のシグナルとして3次高調波と同じ周波数のサイン波を与え、それを解析している。具体的には、dspstate~オブジェクトはbangメッセージを受け取ると現在DSP Statusウィンドウで設定されているサンプリング・レートを出力する。それをFFTサイズである128で割るとFFT基本波の周波数344.531Hzが得られる。つまりこのFFTでは、0Hzから始まって344.531Hzごとに周波数ビンが設定され、それぞれのエネルギーが解析される。そして、そこに基本波の周波数の3倍、つまり3次高調波の周波数と同じ周波数を持つcycle~オブジェクトの出力シグナルを与えている。

■5-18-7 テスト用のシグナルにFFTを実行する



オーディオ処理を開始して、先ほどと同じ要領でcapture~オブジェクトに格納された内容を確認してみよう。ここではfft~オブジェクトの第1アウトレットにつながれたcapture~オブジェクトをダブル・クリックする。最初の128個は無視し、続く128個のサンプルの値を見てみると、確かに、4番目の数値のみが大きく、それ以外はすべて0になっているのが分かる。fft~オブジェクトの出力は0Hzの周波数ビンから始まるので、4番目のビンは3次高調波のスペクトルに当たる。これでfft~オブジェクトは正確にスペクトル解析を行っていることが確認できた。

■5-18-8　テスト用シグナルのFFT実行結果



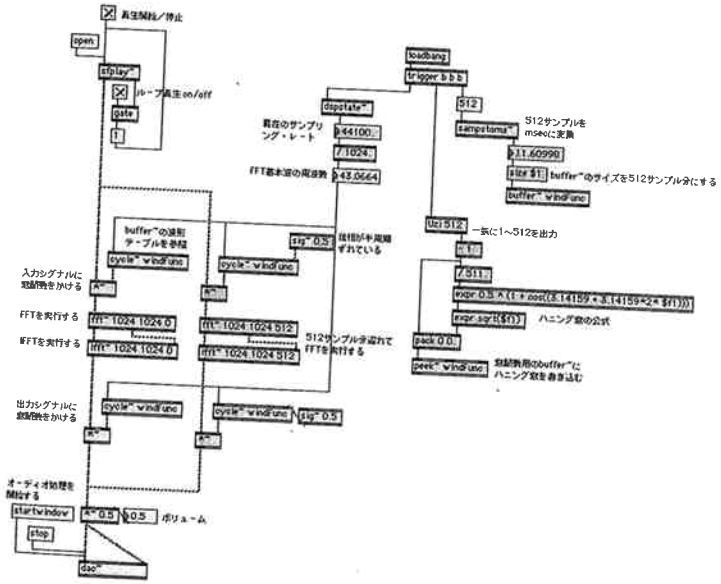
fft~オブジェクトとifft~オブジェクトによる窓関数とオーバーラッピング

任意の周波数成分を含んだシグナルを満足にスペクトル解析するためには、すでに説明したように窓関数とオーバーラッピングというテクニックが用いられる。これをfft~オブジェクトとifft~オブジェクトによって実現したのが5-18-9のパッチだ。

まず、窓関数について説明しよう。ここでは窓関数をbuffer~オブジェクトに格納された波形テーブルとして用意し、それをcycle~オブジェクトにより繰り返し読み出すことで窓かけする方法を採用している。パッチを開くとloadbangオブジェクトが出力するbangメッセージにより、3つの基本的な初期設定が行われる。

まずbuffer~オブジェクトのサイズが512サンプル分に設定される。buffer~オブジェクトにはアーギュメントとしてwindFuncという名前が付けられている。なぜ512サンプルかと言えばcycle~オブジェクトが参照するものが512サンプルの波形テーブルと決められているからだ。buffer~オブジェクトのサイズはmsec単位で与えなければならないので、sampstoms~オブジェクトでサンプル数をmsecに変換して渡している。

■5-18-9　窓関数とオーバーラッピングにより任意の周波数を解析する

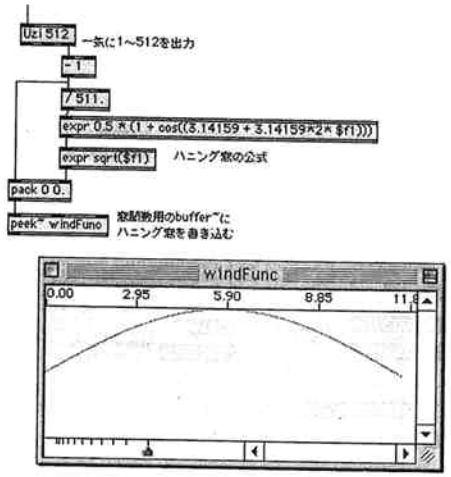


次に、buffer~オブジェクト windFuncにハニング窓を書き込む。数式についてはここでは詳細の解説は控えるが、Uziオブジェクトにより一気に512サンプル分のデータを計算させ、peek~オブジェクトにより1サンプルずつ書き込みを行っている。書き込まれたハニング窓の形状は次の通りだ。cycle~オブジェクトはこれを読み出してシグナルを出力する。

最後に、cycle~オブジェクトの周波数を設定する。これはFFT基本波の周波数と同じでなければならない。なぜならFFTフレームごとに窓かけを行う以上、フレームの周期と窓関数の周期は同じで同期していなければならないからだ。fft~オブジェクトおよびifft~オブジェクトのFFTサイズが1024に設定されているため、サンプリング・レートを1,024で割った周波数をcycle~オブジェクトに与える。

パッチを開いた際に自動的に行われる初期設定は以上である。

■5-18-10 peek~オブジェクトによりbuffer~オブジェクトにハニング窓を書き込む

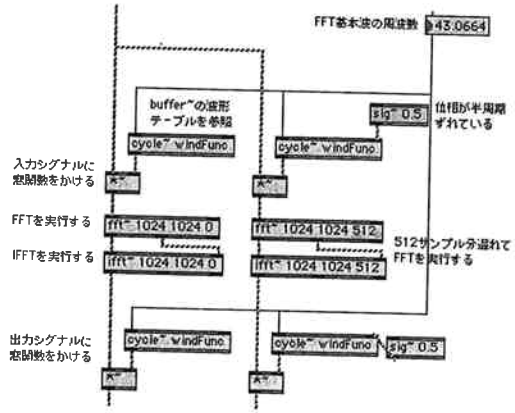


続いて、fft~オブジェクトおよびifft~オブジェクトの部分に移ろう。オーバーラッピングは複数のFFTを時間をずらせて実行することだ。ここでは2つのfft~オブジェクトを用意している。時間的なずれ、すなわちホップサイズは、一方のfft~オブジェクトのアーギュメントでFFTを開始するディレイ・タイムを512にすることで設定している。これにより右側のfft~オブジェクトは左側より512サンプル分遅れてFFTを実行することになる。512サンプルはFFTサイズのちょうど1/2である。

FFTを時間的にずらす以上、入力シグナルの窓かけも同様にずらす必要がある。このため右側のcycle~オブジェクトでは位相を0.5とし、左側のcycle~オブジェクトと半周期分位相をずらせた状態にしている。これで窓関数はぴったりとFFTフレームに張り付くように同期する。

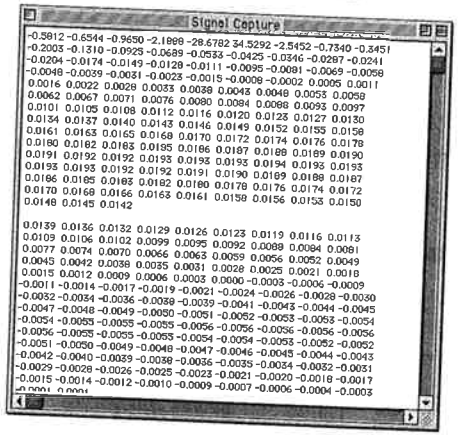
このパッチではifft~オブジェクト後の合成シグナルに対しても同じ窓かけを行っている。入力と出力の両方に同じ窓かけを行う方法は、よりスムーズなスペクトル合成を期待する場合にしばしば用いられる。

■55-18-11 FFTと窓関数の時間的ずれ



5-18-12はこのパッチで200Hzのサイン波シグナルをスペクトル解析した結果（実部）だ。確かに全周波数ビンに渡って本来のシグナルには含まれていない周波数成分が散乱してはいるが、そのエネルギーの大きさはわずかであり、符号を無視すると、5番目（4次高調波）と6番目（5次高調波）の周波数ビンのところが突出して大きくなっている。これにより解析された周波数は、43.06×4＝172.24Hzと、43.06×5＝215.3Hzの間にあると推定できる。

■5-18-12 200Hzのサイン波シグナルを解析した結果



pfft~オブジェクトを利用する

窓かけとオーバーラッピングはスペクトル解析、スペクトル合成にとって欠かせない基本的なテクニックでり、FFT関連のパッチを作ろうとすれば必ずといってよいほど必要になるものだ。しかし、先に紹介した方法ではいちいち窓関数を描かせ、オーバーラップする数だけfft~オブジェクトとifft~オブジェクトを用意しなければならない。もし、プログラミングの結果が不満で、窓関数を別のものに替えたり、オーバーラップ数を増やしたいと思った場合、かなり面倒な作業になるだろう。

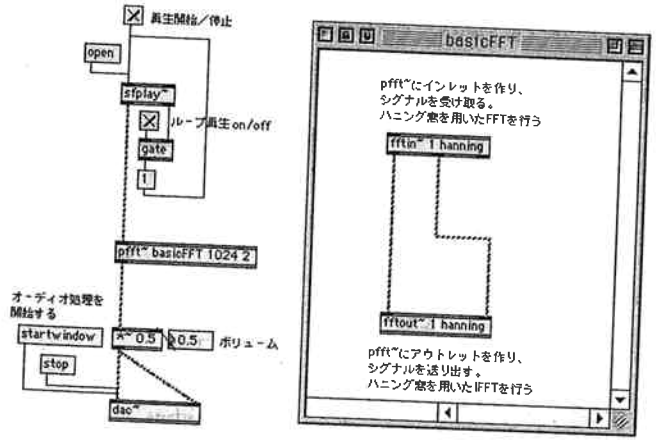
MSP2では、こうした必要性を考慮し、ユーザーが窓かけとオーバーラッピングを気にすることなく、本来試みたいスペクトル合成処理に集中できるようにpfft~というオブジェクトを用意している。

pfft~オブジェクトは、ポリフォニー機能を実現するpoly~オブジェクトに例えることができる。poly~オブジェクトは、サブ・パッチを読み込み、アーギュメントでポリフォニー数を指定することで、それを内部的にコピーし、1つのサブ・パッチによる複数のオーディオ処理を可能にするものだった。同様にpfft~オブジェクトは、FFT処理を行うサブ・パッチを読み込み、アーギュメントでFFTサイズ、オーバーラップ数を指定する。5-18-9のパッチをpfft~オブジェクトを用いて書き換えると5-18-13のような極めてシンプルなものになる。

pfft~オブジェクトのアーギュメントとして、読み込むサブ・パッチ"basicFFT"の名前、FFTサイズ＝1024、オーバーラップ数＝2を書き込んでいる。これによりサブ・パッチbasicFFTは内部的に2個コピーされ、自動的にオーバーラッピングされる。FFTの時間的なずれ、すなわちホップサイズは、FFTサイズをオーバーラップ数で割ったものが自動的に設定される。このようにpfft~オブジェクトを使用すればアーギュメントを書き換えるだけでFFTサイズやオーバーラップ数を簡単に変更することができる。

サブ・パッチbasicFFTは、単に入力シグナルをスペクトル解析し、それを再び時間領域のシグナルに再合成して出力するだけのものだ。pfft~オブジェクトのインレットとアウトレットは読み込むサブ・パッチのfftin~オブジェクト、fftout~オブジェクトによって作られる。fftin~オブジェクトにはアーギュメントとして必ずインレット番号を、fftout~オブジェクトにはアーギュメントとして必ずアウトレット番号を書き込まなければならない。ここではインレット、アウトレットとも1つでよいので、どちらも1番にしている。

■5-18-13 pfft~オブジェクトの基本的使用法

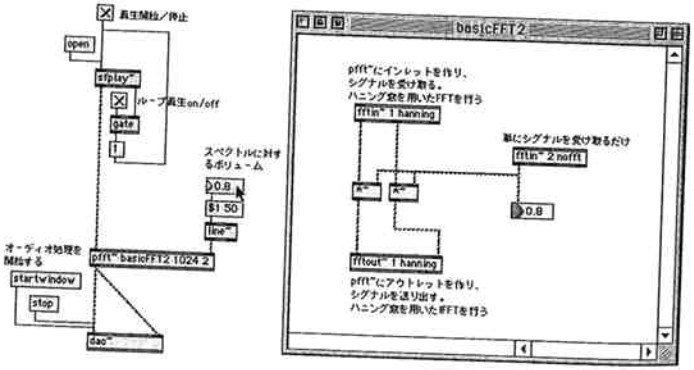


しかし、fftin~オブジェクト、fftout~オブジェクトは単なるシグナル入出力の機能を担うだけではなく、それ自体がサブ・パッチ内でfft~オブジェクトとifft~オブジェクトと同様の機能を果たしている。しかもアーギュメントとして窓関数のタイプを指定すれば、予め内部的に用意されている窓関数を呼び出して使用することができる。用意されている窓関数は次の通りだ。

square	矩形窓（これを指定すると窓かけによる影響を受けない）
triangle	三角窓
hanning	ハニング窓
hamming	ハミング窓
blackman	ブラックマン窓
nofft	FFT処理自体を実行しない

最後のnofftを設定するとfftin~オブジェクトとfftout~オブジェクトはFFT処理あるいはIFFT処理を行わず、単にメイン・パッチとの間でシグナルを受け渡しするだけの入出力オブジェクトになる。以上のようなfftin~オブジェクトとfftout~オブジェクトの機能は、pfft~オブジェクトのサブ・パッチの内部で用いられてはじめて機能する。この点は気を付けてほしい。

■5-18-14　周波数領域でのボリューム・コントロール

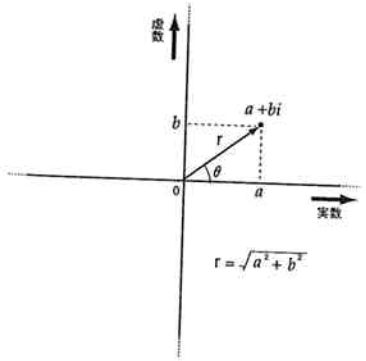


5-8-14のパッチは、fftin~オブジェクトとfftout~オブジェクトの間に *~オブジェクトをはさみ、ボリュームをコントロールしているものだ。時間領域で振幅をコントロールしているのではなく、周波数領域で各スペクトルのエネルギーをコントロールしている点が重要である。fftin~ 2 nofftオブジェクトは、pfft~オブジェクトに第2インレットを作り、そこからシグナルを受け取るが、FFT処理は行わず、受け取ったシグナルをそのまま通過させている。

直行座標を極座標に変換する

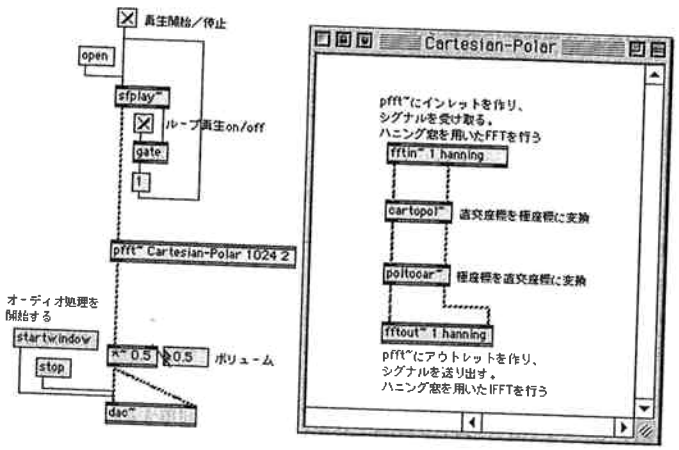
「5-17　FFT：FFTの基礎」(P640)で説明したように、FFTはその計算結果を複素数で出力する。fft~オブジェクトあるいはfftin~オブジェクトの場合も同様で、出力するシグナルは複素数の実部と虚部である。これを図示すると5-18-15のようになる。つまり、fft~オブジェクトあるいはfftin~オブジェクトの第1アウトレットから出力されるのは図のaの値、第2アウトレットから出力されるのは図のbの値になる。これは複素数の直行座標（デカルト座標）の値である。これを極座標に変換すると振幅と位相の値が得られる。図で言うと振幅はrの値、位相はθの値になる。直行座標の実部と虚部を極座標の振幅と位相に変換することはFFTの処理結果を利用する際にさまざまな有効性がある。例えば、極座標は正負の値を取るが、振幅はrの長さであり常に正の値になる。もしスペクトルをグラフィカルに表示しようとすれば、振幅の値を利用するのが適切である。

■5-18-15　直行座標と極座標



MSPには、直行座標を極座標に変換するオブジェクトとしてcartopol~、逆に極座標を直行座標に変換するオブジェクトとしてpoltocar~が用意されている。5-18-16のパッチはfftin~オブジェクトの出力（直行座標）をcartopol~オブジェクトで極座標に変換し、再びそれをpoltocar~オブジェクトで直行座標に変換してfftout~オブジェクトに渡している。

■5-18-16　直行座標と極座標の変換オブジェクト



さて、次の5-18-17のパッチはスペクトルをリアルタイムにグラフィカル表示するパッチ例だ。pfft~オブジェクトに読み込むサブ・パッチ"FFTamp"では、fftin~オブジェクトによりFFTを実行し、cartopol~オブジェクトで直行座標を極座標に変換して振幅だけを出力している。またfftin~オブジェクトの第3アウトレットから出力される現在の周波数ビンのインデックスもメイン・パッチに送っている。

メイン・パッチでは、スペクトル書き込み／読み出し用のbuffer~オブジェクトを用意し、そこにpoke~オブジェクトによって振幅を書き込んでいく。poke~オブジェクトは、record~オブジェクトとは異なって、第2インレットにサンプル・インデックスを受け取り、そのインデックスに第1インレットに受け取るサンプルの値を書き込んでいく。このサン

プル・インデックスとしてfftin~オブジェクトから出力される現在の周波数ビンのインデックスを利用している。こうしてサンプル・インデックスの0～511に刻々とスペクトルが書き込まれていくわけだ。すでに説明した通りスペクトルとして意味があるのは0～511番目までである。

buffer~オブジェクトの読み出しはpeek~オブジェクトを用いて行っている。peek~オブジェクトもサンプル・インデックスによりそのサンプル値を読み出すが、入出力ともシグナルではなく実数である。Uziオブジェクトはbangメッセージを受け取ると一気に1から256までを出力し、これを利用してbuffer~オブジェクトの最初の256個のサンプルを読み出す。これは0Hz～11kHzまでの周波数ビンの振幅に相当する。スペクトル表示にはmulitiSlider《MutiSlider》オブジェクトを用いている。Uziオブジェクトはmetroオブジェクトにより25msecごとに動作しているのでスペクトル表示も25msecごとに更新される。ちなみにテスト用トーンとしてcycle~オブジェクトのサイン波シグナルをpfft~オブジェクトに入力している。スペクトル表示で8kHzのところに大きな高まりがあるのは、このcycle~オブジェクトのシグナルを解析した結果である。

■5-18-17　グラフィカルにスペクトルを表示するパッチ