3　音の成分の分析 (1)

3.1　フーリエ変換

いろいろな周波数の正弦波を足し合わせるといろいろな音が作れる。逆に、全ての音はいろいろな周波数の正弦波に分解できる。それを可能にする技術がフーリエ変換である。我々が今扱おうとしている離散表現の音に対しては、離散フーリエ変換という技術がある。

matlab で離散フーリエ変換を計算する関数は `fft` である。

さっそく fft を試してみよう。

```
>> t = 0:1/100:6-1/100;                    % 0 から 1/100 ずつ 6 (の一つ手前) まで
                                           % ベクタを作る
>> y = sin(2*pi*15*t) + sin(2*pi*40*t);    % 15Hz と 40Hz の成分を持つ信号
>> f=fft(y);                               % y の長さ = 600 点で fft を y にかける
>> f(10) % f(10) の値を見てみる
ans =
  7.7936e-013 +1.5210e-013i                % 複素数になっている
>> s = abs(f);                             % フーリエ変換した結果の「大きさ」は
                                           % 振幅スペクトル
>> plot(s);                                % 4 本の線がプロットされる
```

プロットには、真ん中の 300 点のところを中心に線対象の関係で 2 本ずつの線が観測される。

どこの場所に線があるかを調べるには次のようにすると便利である。プロットした結果を見ると、線の場所以外は、ほとんど 0 の値となっている。

```
>> find(s>1)
ans =
    91   241   361   511
```

matlab で

```
>> s>1
```

と実行すると、s の全ての要素について 1 より大きいかどうかの論理演算を行ない、結果を 0 か 1 の値を持つ配列とする。そのままでは、全ての要素 (今の場合は 600 個) を表示してしまう。ここで用いる `find` という関数は、ある配列の非零要素のインデックスの一覧を返却する。つまり、今の場合は、線が観測されているのはどの値のときかがわかることになる。

この場合、`91,241,361,511` のところに線があるということである。

この線のある場所は周波数に対応している。フーリエ変換をおこなった 600 点でサンプリング周波数 100Hz に対応している。したがって、1 点では、1/6Hz をあらわす。線があるのは、matlab の添字が 1 から始まることを考慮すると、前半の部分は 90 点、240 点のところである。したがって、それぞれ、15Hz, 40Hz をあらわしていることがわかる。つまり、fft をおこなってえられるスペクトルを見れば元の信号が含んでいる成分がわかるのである。

fft は、いろいろな長さでかけることができる。短かい範囲でかけてみる。

```
>> plot(abs(fft(y,599)))
>> plot(abs(fft(y,590)))
```

線の下の方が太くなることがわかる。

実は、離散フーリエ変換では、範囲をきめて変換しているが、範囲の外は、範囲の中と同じ変動が繰り返されていると仮定している。つまり、`fft(y)` の場合は、y(600) の次に y(1) が来ると仮定しているのである。

`fft(y,599)` の場合は、y(599) の次に y(1) がくると仮定している。つまり、その部分で、実際の周期的変動とはずれた変化が起きてしまうのである。

信号 (の成分) の周期がわかっていれば、その周期にあわせた幅で `fft` をかければよいが、どのような成分が含まれるかを調べるときには、周期がわかっていない。

そこで、周辺の部分で幅が周期にはずれて悪影響を与えることを防ぐため、窓関数を利用する。

窓関数の例としてハニング窓を見てみる。

```
>> h = hanning(600);
>> plot(h);
```

このように裾の部分を 0 にすることにより、境界の部分 (裾の部分) でなめらかにつながるようにしている。

ハニング窓をかけてみると、fft の幅 (以降、この幅を窓幅と呼ぶことにする) が周期とずれていても、その影響が小さくなることがわかる。

```
>> hold on
>> plot(abs(fft(y(1:599), 599)));
>> plot(abs(fft(y(1:599).*hanning(599)', 599)), 'r');
```

ハニング窓をかけた結果は赤い色でプロットされている。この図を見ればわかるように、窓をかけた結果の方が線が細く、不要な成分が検出されにくいことがわかる。

`fft` を使って、実際の音を分析してみよう。

`a-.wav` というファイルは、サンプリング周波数 8000Hz で「あー」としゃべった音声を含んでいる。

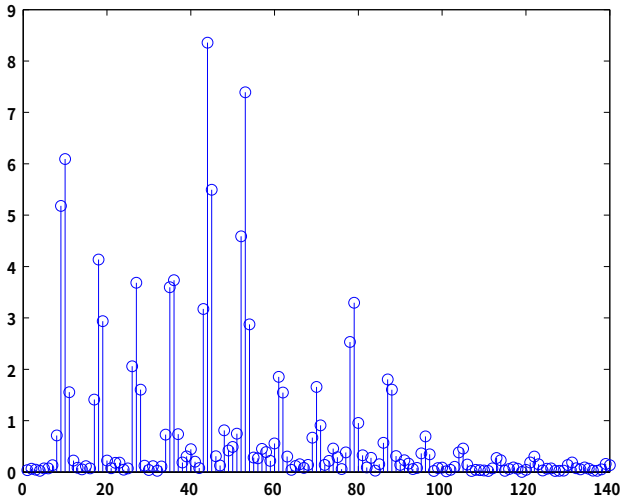
男性の声は 100Hz よりは高いことが多い。今、512 点で `fft` をかけるとすると 1 点で $8000/512 = 15.625$ なので 15.625Hz をあらわすことになる。とりあえず、最低限の分析性能はあるだろうから 512 点の fft で分析する。また、512 点は $1/8000 * 512 = 0.064$ なので 64 ミリ秒間である。

適当に中程の 64 ミリ秒分のデータに対し 512 点の `fft` をかける。

```
>> [y, fs, n] = wavread('a-.wav');
>> h = hanning(512);
>> yy = abs(fft(y(2001:2512).*h)); % wavread で読み込んだ場合は y は
                                   % 列ベクタとなるので h を転置する
                                   % 必要がない
>> plot(yy);
>> plot(yy(1:70)); % 15.625 * 70 = 1094 (Hz) まで表示
>> stem(yy(1:70));
>> find(yy(1:70)>1) % ピークを探す
```

例えば、ピークが 9, 18, 27, 36 であれば、それぞれ 15.625 をかけた 141, 281, 422, 563 (Hz) の成分が含まれていることがわかる。また、この音声の基本周波数 (音の高さ) は 141 Hz であることがわかる。

下図は、約 2000Hz までのスペクトルを表示したものである。



このスペクトルのピークの成分を足し合わせて音を作ってみる。

```
>> t = 0:1/8000:1;
>> for k = 1:12,
x = x + yy(9*k) * sin(2 * pi * k * 9 * 15.625 * t);
end
>> soundsc(x, 8000);
>> plot(x(1:512));
```

なんとなく、それらしい音が聞こえる??

課題 1 自分で適当な音を録音して `fft` を用いてスペクトルを観察せよ。窓関数や `fft` の点数をいろいろ試してみること。定常的な音でない場合は、いろいろな部分を分析すること。サンプリング周波数に注意して、`fft` の点数などを設定すること。

課題 2 `fft` を用いて分析したスペクトルに基づいて正弦波を組み合わせて、音を作ってみよ。