ヒューマンインターフェース
Report2

氏名:津波古正輝
学籍番号:e075739A

平成21年7月11日

音声信号に対する次の 3 つのスペクトルを算出、描画する SCILAB プログラム (lpc2.sci) を提供します。

- Hamming 窓で窓掛けした DFT スペクトル。
- 自己相関 LPC 分析 (Levinson-Durbin 法) で得られたスペクトル。
- LPC 逆フィルタで算出された残差スペクトル。

課題として、次の事を行い、レポートにまとめてください。

(1) プログラムを解読する。(どこで何を行っているかを数式とともに示す。)

(2) 提供プログラムでは Levinson-Durbin 法を SCILAB の関数 lev() で実現しています。関数を使わず SCILAB 言語でプログラミングして実現しましょう。

(3) LSP を算出し、スペクトル、単位円上に PLOT しましょう。

(4) プログラムの最後に伝達特性=0 の根を roots() により算出し、極周波数と帯域幅を極周波数の小さい順に表示するプログラムを追加してください。0 から 5KHz の極だけを抽出し、極周波数で昇順にソーティングしましょう。実幅に根が存在する場合もあるので、0 から 5KHz の極は次数の半分とは限りません。結果があっているか、スペクトルと極配置図により確認しましょう。

(5) 男性音と女性音それぞれで、分析長を 100 から 300、分析次数を 8 から 20 程度に変化させ、スペクトルがどう変化するかを調べ、その結果について考察しましょう。

課題 1

プログラムを解読する。(どこで何を行っているかを数式とともに示す。)

```
clear
FS=10000;                    //LPC の横軸
fft_len=512;                 // FFT のサンプル数
start=5000;                  // 分析位置
len=300;                     // 分析長
order=14;                    // 分析次数
pre_emp=0.0;                 // プリエンファシス

print(%io(2),'start= '); //『start』と表示してスカラ値を標準入力
start = read(%io(1),1,1); //start にスカラ値の代入
print(%io(2),start,len,fft_len,order); //上で設定した値を入力

  x=loadwave('1.wav');   // 1.wav というファイルの読み込み
  //音声データの音声部分が x(1),x(2)…に入力される
```

行っている処理

変数の定義、start 値の決定、wav ファイルの読み込みを行っている。

```
//Hamming 窓掛け&プリエンファシス
//Hamming 窓掛けとは、有限区間以外が 0 になるような処理をする関数である。ここでの有限区間は len
//窓掛けの式；w(x)=0.54-0.46*cos2 π x,
//プリエンファシス:pre_emp=0 なので強調は行っていない。
for i=1:len
  win(i) = 0.54 - 0.46 * cos(2 * %pi * i / len);
  x1(i) = (x(i+start)-pre_emp*x(i-1+start)) * win(i);
end
//区間区域以外を 0 にする
for i=len+1:fft_len   // ゼロ詰め
  x1(i) = 0;
  win(i) = 0;
end
```

行っている処理

実データすべてを数値的にフーリエ変換する場合、無限の長さは扱えないので、ある区間内のデータを繰り返すことにする。そこで、$win(i) = 0.54 - 0.46 * cos(\frac{2\ \pi\ i}{len})$ という関数を用意し、len 区間内のフーリエ変換を行っている。len 以外の区間は 0 にしている。

その後、音声データ x(i+start) をプリエンファシスで調整し x1(i) へコピーする。しかし、pre_emp=0 なので調整は行っていない。

```
//離散フーリエ変換対数スペクトルの算出
//関数 fft：高速フーリエ変換
//高速フーリエ変換とは計算機上で離散フーリエ変換を高速にするアルゴリズムのこと
fft_spc=20*log10(abs(fft(x1,-1)));
```

行っている処理

離散フーリエ変換の式；$f_j = \sum_{k=0}^{N-1}(x1) * e^{-\frac{2\ \pi\ ijk}{N}}$ (j=0⋯N-1)

離散フーリエ変換対数スペクトル；fft_spc=20*log10(|$\sum_{k=0}^{N-1}(x1) * e^{-\frac{2\ \pi\ ijk}{N}}$|)

fft で音声データ x1 を高速フーリエ変換する。その後 abs で絶対値を算出し 20*log10() で db 値に変換。全体として離散フーリエ変換対数スペクトルの算出を行っている。

```
//自己相関関数の算出
//自己相関関数：ある時刻での信号と t 時間後の信号にどのような関係があるのかを調べる。
j=1;
for i=0:order
  r(j)=0;
  for n=1:len-i
    r(j)=r(j)+x1(n)*x1(n+i);
  end
  j=j+1;
end
```

行っている処理

$$\text{自己相関関数；} r = \sum_{n=1}^{len-1} (x1(n)) * (x1(n+i))$$

分析次数の数だけ読み込みデータを少しずつずらし、掛け合わせた総和を自己相関係数 r としている。

```
-------------------------------------------------------------------------
//Levinson 算法による LPC 分析
//LPC 分析とは線形予測分析のことでここでは関数 lev() がその役割を果たしている。
//sigma2:予測誤差のエネルギーであり、区間内の二乗の総和である。
//rc; 反射係数 (PARCOR 係数)
//ar:LPC 係数
[ar,sigma2,rc]=lev(r); // sigma2=r(0)+_sum_{i=1}^order(r(i)*a(i))
-------------------------------------------------------------------------
```

行っている処理

$$\text{予測残差エネルギー；} sigma2 = r(0) + \sum_{i=1}^{order} (r(i)) * (a(i))$$

lev 関数で自己相関係数 r を LPC 分析し、LPC 係数、予測誤差、反射係数を算出している。

```
-------------------------------------------------------------------------
// LPC 対数スペクトルの算出
//a(1) には 1。分析次数 (order) まで a(i+1) に格納
//分析次数以外は 0 にする
//ar_spc;LPC スペクトル
a(1)=1;
for i=1:order
  a(i+1)=ar(i);
end
for i=order+1:fft_len-1  // ゼロ詰め
  a(i+1)=0;
end
ar_spc=-20*log10(abs(fft(a,-1)))+10*log10(sigma2);
-------------------------------------------------------------------------
```

行っている処理

$$\text{LPC スペクトル；ar\_spc=20*log10(}|\sum_{k=0}^{N-1} a * e^{-\frac{2\ \pi\ ijk}{N}}|) + 10 * log10(r(0) + \sum_{i=1}^{order}(r(i)) * (a(i)))$$

求めた ar(LPC 係数) をフーリエ変換し絶対値算出、デシベル化を行っている。デシベル化した LPC 係数に予測誤差の対数を 10 倍したものを加算して LPC スペクトルとしている。

```
// 残差のスペクトルの算出
//入力された start 値から分析長 (len) だけ res に格納。LPC 係数と少しずらした音声データを掛け、元のデータに加算している。
//分析長以外は 0 にする
//求めた残差を高速フーリエ変換し、絶対値を求め、デシベルに変換
for n=1:len
  res(n)=x(start+n);
    for i=1:order
      res(n)=res(n)+ar(i)*x(start+n-i);
    end
  end
  for n=len+1:fft_len
    res(n)=0;
  end
res_spc=20*log10(abs(fft(res,-1)));
```

行っている処理

$$\text{残差；res=}\sum_{i=i}^{order} ar(i) * x(start + i)$$

$$\text{残差スペクトル；res\_ spc=20*log10|(x(start)+}\sum_{i=i}^{order} ar(i) * x(start + i)))|$$

現在のデータ (x(start)) を基準とし、そのデータから予測した少し未来のデータを用意する。少し予測した未来のデータから過去 (次数の分だけ) のデータの LPC 係数の乗算の総和を残差としている。フーリエ変換、絶対値、デシベルに変換された現在のデータに残差を加えている。

```
// スペクトルの描画
 xset('window',1);  xbasc(1);
 tics=[2,4,2,4];
 Hz_flag=1;
if Hz_flag == 0
 rect=[1,min(fft_spc),fft_len/2,max(fft_spc)];
 plotframe(rect,tics,[%f,%f],['LPC','Freq.','Amp.[dB]'],[0,0,1.0,0.5]);
 n=1:fft_len/2;
 plot2d(n,fft_spc(n),1,"000");
 plot2d(n, ar_spc(n),2,"000");
 plot2d(n,res_spc(n),3,"000");
else
 rect=[0,min(min(fft_spc),min(ar_spc),min(res_spc)),FS/2,max(max(fft_spc),
 max(ar_spc),max(res_spc))];
 plotframe(rect,tics,[%f,%f],['LPC','Freq.[Hz]','Amp.[dB]'],[0,0,1.0,0.5]);
 Hz=(0:fft_len/2)/(fft_len/2)*(FS/2); // 修正
 fft_spc=fft_spc(1:(fft_len/2+1));
 ar_spc = ar_spc(1:(fft_len/2+1));
 res_spc=res_spc(1:(fft_len/2+1));
 plot2d(Hz,fft_spc,1,"000");
 plot2d(Hz, ar_spc,2,"000");
 plot2d(Hz,res_spc,3,"000");
// set(gca(),'data_bounds', [0,min(min(fft_spc),min(ar_spc),min(res_spc));
FS/2,max(max(fft_spc),max(ar_spc),max(res_spc))]); // 追加
// xgrid();
end
// 極を算出し描画
 HAR=poly(a(1:order+1),'z','coeff');
 pp=roots(HAR);
 for i=1:order
  pp(i)=1/pp(i);
 end
 x=0:0.1:2*%pi;
// xbasc();
 rect=[-1,-1,1,1];
```

```
 tics=[2,5,2,5];
 plotframe(rect,tics,[%f,%f],["Unit Circle","Re.","Im."],[0.25,0.5,0.5,0.5]);
 plot2d(cos(x),sin(x),1,"000");
 ra=real(pp); ia=imag(pp);
 plot2d(ra,ia,-3);
 xgrid();
```

行っている処理

スペクトルの描画を行っている。

課題 2

提供プログラムでは Levinson-Durbin 法を SCILAB の関数 lev() で実現しています。関数を使わず SCILAB 言語でプログラミングして実現しましょう。

関数 lev() の実現:

lpc2.sci は lev(r) で LPC 係数 ar を求めていた。なので、ar と同じ値を求めることができればよいということになる。

```
--------------------------------------------------------------------------
// Levinson 算法による LPC 分析
[ar,sigma2,rc]=lev(r); // sigma2=r(0)+_sum_{i=1}^order(r(i)*a(i))
a2=zeros(order,order);//a2(m,i)
w=zeros(1,order+1);
u=zeros(1,order+1);
u(0+1)=r(0+1);
w(0+1)=r(1+1);
for m=1:order
  k(m)=w(m-1+1)/u(m-1+1);
  u(m+1)=u(m-1+1)*(1-k(m)*k(m));
  for i=1:m-1
  a2(m,i)=a2(m-1,i)-k(m)*a2(m-1,m-i);
  end
  a2(m,m)=-k(m);
  if m==order
    break;
  end
  w(m+1)=r(m+1+1);
  for i=1:m
  w(m+1)=w(m+1)+a2(m,i)*r(m+2-i);
  end
end
ar2=a2(order,:)';
rc2=-k;
sigma22=u(order+1);
--------------------------------------------------------------------------
```

```
-->ar
 ar  =

  - 1.5325606
    0.8189212
  - 0.6505553
    1.0991983
  - 0.1440923
  - 0.5707576
  - 0.2186618
    0.3937994
  - 0.1829691
    0.8143875
  - 1.0981572
    0.6057201
  - 0.3127022
    0.1978371
```

```
--> rc
 rc  =

  - 0.7932610
    0.8038744
    0.1148795
    0.2616105
  - 0.6413447
  - 0.0529957
    0.2742545
    0.1200309
    0.0488004
  - 0.2009120
  - 0.3999150
    0.4466840
  - 0.0098920
    0.1978371
```

```
--> sigma2
 sigma2  =

    0.0022461

-->sigma22
 sigma22  =

    0.0022461
```

```
-->ar2
 ar2  =

  - 1.5325606
    0.8189212
  - 0.6505553
    1.0991983
  - 0.1440923
  - 0.5707576
  - 0.2186618
    0.3937994
  - 0.1829691
    0.8143875
  - 1.0981572
    0.6057201
  - 0.3127022
    0.1978371
```

```
-->rc2
 rc2  =

  - 0.7932610
    0.8038744
    0.1148795
    0.2616105
  - 0.6413447
  - 0.0529957
    0.2742545
    0.1200309
    0.0488004
  - 0.2009120
  - 0.3999150
    0.4466840
  - 0.0098920
    0.1978371
```

図 1: Levinson 算法による LPC 分析

課題 3

LSP(線スペクトル対) を算出し、スペクトル、単位円上に PLOT しましょう。

lev 関数、又は課題 2 で LPC 係数を求めた後に、lsp を求めるプログラムを追加する

```
--------------------------------------------------------------------------
[ar,sigma2,rc]=lev(r); // sigma2=r(0)+_sum_{i=1}^order(r(i)*a(i))
lpc_order=order;
fs=FS;
lpc(1)=1;
for i=1:order
  lpc(i+1)=ar(i);
end
dft_size=4096;
lsp=zeros(1,lpc_order);
omega=zeros(1,lpc_order/2);
theta=zeros(1,lpc_order/2);
p=zeros(1,dft_size);
q=zeros(1,dft_size);
p(1)=lpc(1);
q(1)=lpc(1);
for n=2:lpc_order+1,
  p(n)=lpc(n)+lpc(lpc_order+3-n);
  q(n)=lpc(n)-lpc(lpc_order+3-n);
end
p(lpc_order+2)=lpc(1);
q(lpc_order+2)=-lpc(1);
P=abs(fft(p,-1));
m=1;
for k=2:dft_size/2,
  if P(k-1)>P(k) & P(k)<P(k+1)
    omega(m)=k-1;
    m=m+1;
  end
end
Q=abs(fft(q,-1));
```

```
m=1;
for k=2:dft_size/2,
  if Q(k-1)>Q(k) & Q(k)<Q(k+1)
    theta(m)=k-1;
    m=m+1;
  end
end
for m=1:lpc_order/2,
  lsp(2*m-1)=omega(m)*fs/dft_size;
  lsp(2*m)=theta(m)*fs/dft_size;
end
```

```
//課題３の調整（上の図のプロット部分）
for i=1:length(lsp)
 y(i) = min(fft_spc);
end
plot2d3(lsp,y,5);
```

```
 //課題３の調整（下の図のプロット部分）
 plot2d(cos(lsp*(%pi/5000)),sin(lsp*(%pi/5000)),-7);
```

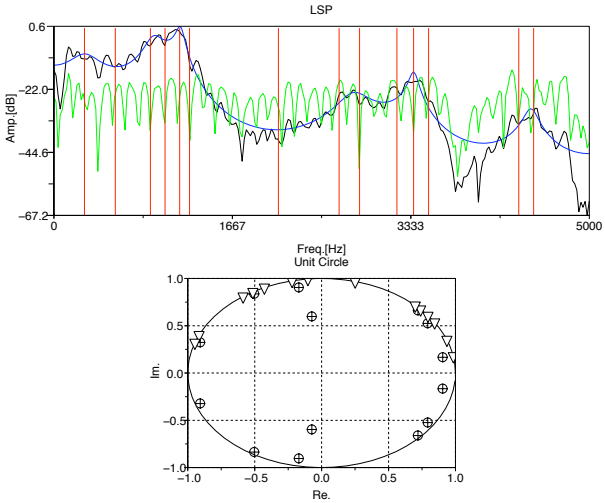

図 2: LSP のプロット

課題 4

プログラムの最後に伝達特性=0 の根を roots() により算出し、極周波数と帯域幅を極周波数の小さい順に表示するプログラムを追加してください。0 から 5KHz の極だけを抽出し、極周波数で昇順にソーティングしましょう。実幅に根が存在する場合もあるので、0 から 5KHz の極は次数の半分とは限りません。結果があっているか、スペクトルと極配置図により確認しましょう。

```
-----------------------------------------------------------------------
F=atan(imag(pp),real(pp))/(2*%pi)*FS;
B=-log(abs(pp))/(%pi)*FS;
FF=F;
BB=B;

//正の数値と負の数値を分ける
n=0;
for i=1:order
  if FF(i)>0
    n=n+1;
    FF(n)=FF(i);
    BB(n)=BB(i);
  end;
end;
//分けられた数値で小さい順で並び替え。負の数値のグループもある。正と負のグループの境目は n で示しており、n が指すまでが正のグループである。
for i=1:n
  for b=i+1:n
    if FF(i)>FF(b)
      m=FF(b);
      FF(b)=FF(i);
      FF(i)=m;
    end;
  end;
  i=i+1;
end;
```

```
//正のグループのみを出力
for i=1:n
  printf("F=%f B=%f \n",FF(i),BB(i));
end;
```

伝達特性の解を求めたものが pp に格納されている。伝達関数の分母多項式が零になる周波数の事を極と言い、pp は極を表している。以下の式で極周波数が求められる。

$$\text{極周波数：F}=\frac{tan^{-1}(\frac{pp\ の\ y\ 軸数値}{pp\ の\ x\ 軸数値})}{2 * \pi * FS}$$

複素平面上の座標からアークタンジェントで角度を求め、その角度を 2 π*FS で割り、極周波数を求めている。
以下の式で帯域幅が求められる

$$\text{帯域幅：B}=-\frac{log(|pp|)}{\pi * FS}$$

出力結果：

```
F=289.061601 B=71.447946
F=929.838857 B=78.653612
F=1183.031928 B=263.449821
F=2693.409238 B=159.777313
F=2795.169382 B=265.500449
F=3361.283275 B=122.415711
F=4455.512984 B=1617.560788
```

課題 5

男性音と女性音それぞれで、分析長を 100 から 300、分析次数を 8 から 20 程度に変化させ、スペクトルがどう変化するかを調べ、その結果について考察しましょう。

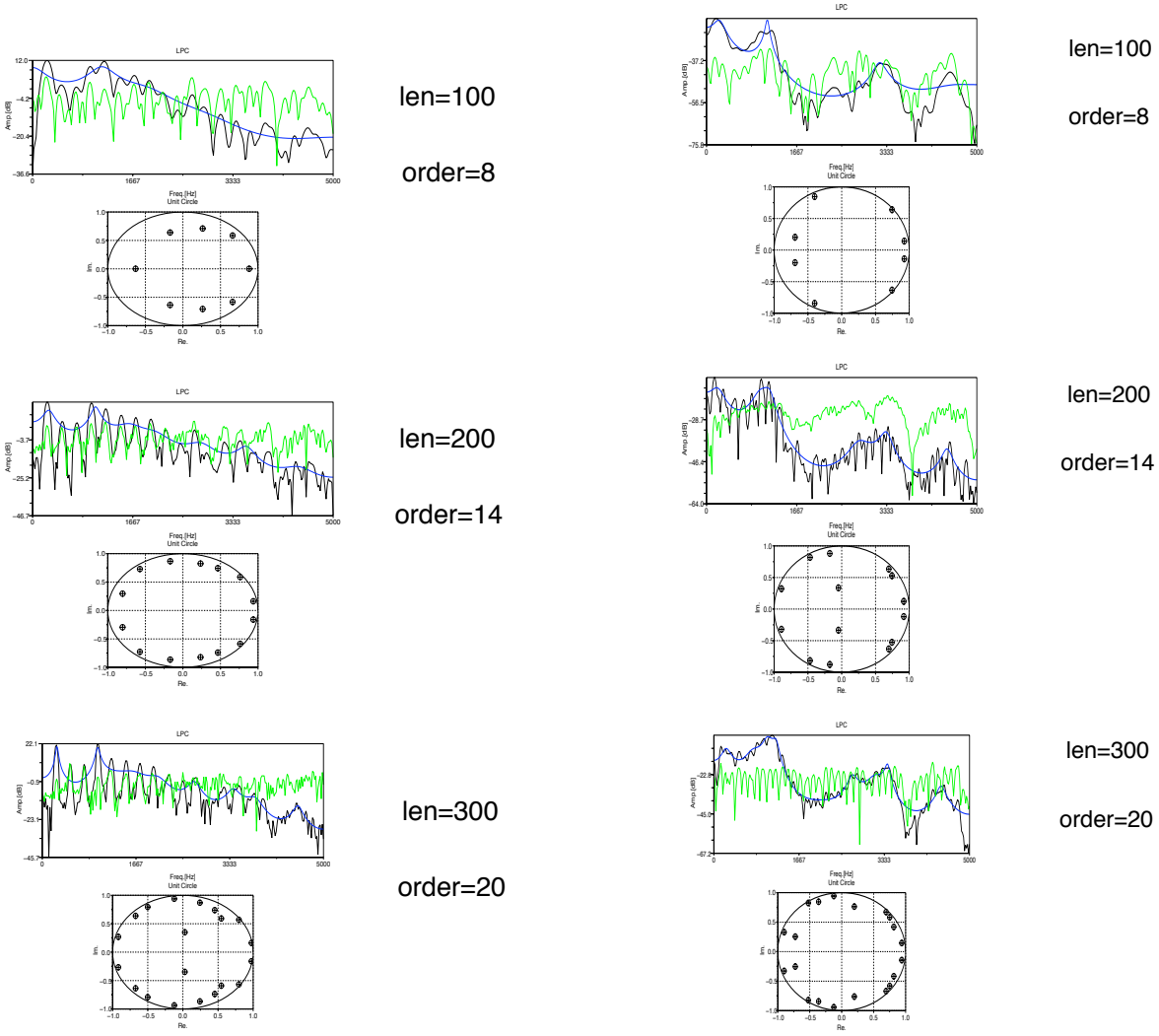

図 3: 女性の声 (左) と男性の声 (右)

- 分析次数を多くすれば予測誤差がなくっている。
- 分析長を多くすればギザギザのグラフになる。分析長が短いと滑らかな線が描かれるが、これはかなり粗い分析であることがわかる。

考察：
len と order を上げていくと男性の方がスペクトルのばらつきが少なくなってきている。これは、男性の声の方が女性より声の分析がしやすいのだろう。