哈尔滨理工大学

实　验　报　告

课程名称：语音信号处理

实验名称：基于线性预测系数的语音合成

预习报告		过程及记录		结果分析		实验总结		总评	
教师签名：									

班　级　电信 21-1

学　号　2105040115

姓　名　刘云翔

指导教师　梁欣涛

2024 年 5 月 21 日

教务处 印制

实验名称	基于线性预测系数的语音合成	时间	2024.5.21
		地点	E1205
同实验者	刘倚云，努尔森巴特·达吾提	班组	电信 21-115

一、 实验预习（准备）报告

1、实验目的

1.掌握基于线性预测系数与基音参数语音合成的原理

2、实验相关原理及内容

1、预测系数和基音参数语音合成模型

相较于前一实验线性预测模型也可设计成一种源滤波器模型，即由白噪声序列和周期性激励脉冲序列构成的激励源信号，经过选通、放大并通过时变数字滤波器(由语音参数控制的声道模型)，获得合成语音信号。语音合成器的示意图如图 5-1 所示。

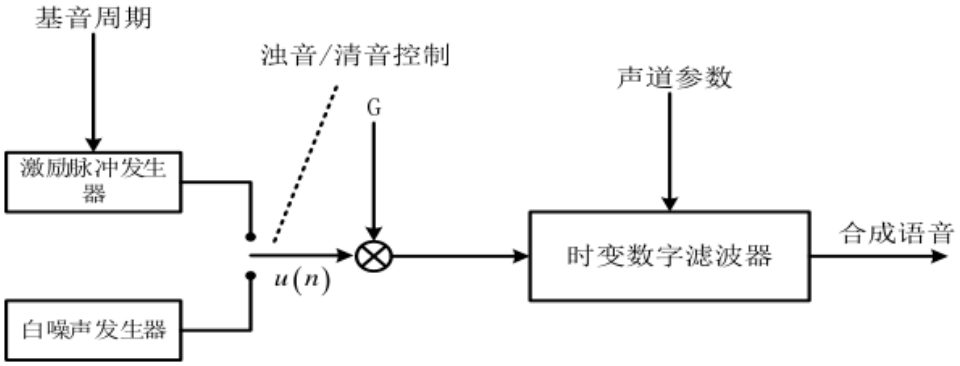


图 5-1 线性预测合成语音模型示意图

图 5-1 所示的线性预测合成语音模型可直接用预测器系数 ai 构成的递归合成滤波器，其结构如图 5-2 所示。用这种方法定时地改变激励参数 $u\quad n$ 和预测系数 a_i ，就能合成出语音。这种结构简单而直观，为了合成一个语音样本，需要进行 p 次乘法和 p 次加法。它合成的语音信号序列为：

$$\tilde{x}(n)=\sum_{i=1}^{p}a_i\tilde{x}(n-i)+Gu(n) \qquad (5\text{-}1)$$

式中，a_i 为预测系数；G 为模型增益；$u(n)$ 为激励源信号（白噪声或周期性激励脉冲序列）；p 为预测阶数。

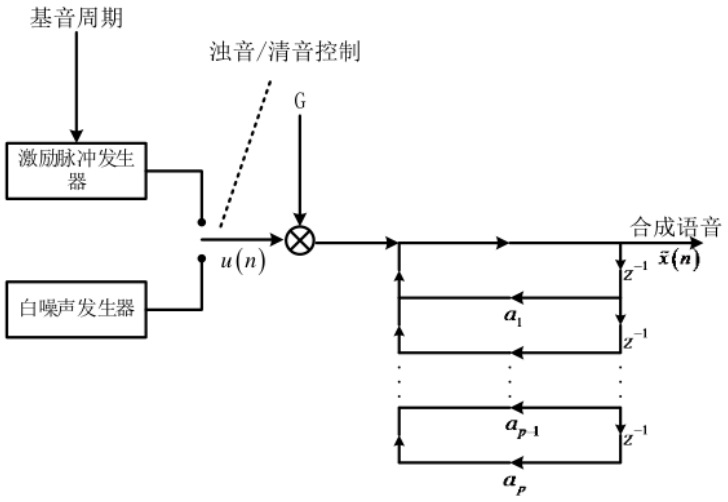


图 5-2　线性预测递归型合成滤波器的语音合成示意图

2、激励脉冲的产生

激励脉冲的产生主要难点在于一帧一帧的连续产生激励脉冲，即要求本帧和上一帧之间激励脉冲序列要连续，同时这一帧和上一帧之间的帧移是 inc 。这就要求本帧的第一个脉冲与上一帧的帧移区内最后一个脉冲之间的间隔要等于本帧的基音周期。

3、实验方法及步骤设计

1. 基于线性预测系数和预测误差的语音合成原理，结合以前所学的内容，基于测试语音，编程实现语音合成。

2. 基于线性预测系数和基音参数的语音合成原理，结合以前所学的内容，调用基音检测函数 pitch_Ceps、滤波处理函数 pitfilterml 等，编程实现语音合成。

4、实验设备仪器选择及材料

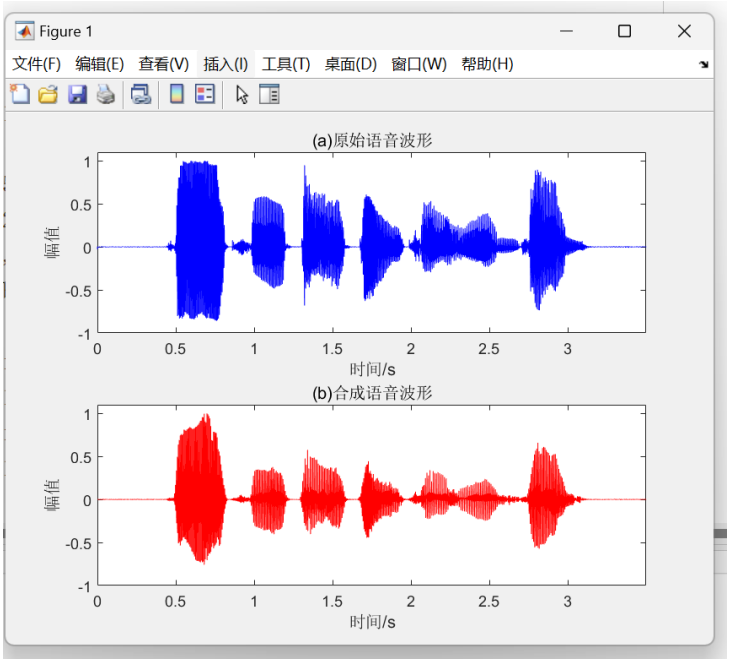
Pc 一台 Matlab2021a

5、实验思考题解答

调用不同的基于检测函数，并比较语音合成的效果：

答：

将实验代码调整，有如下效果：



从对比图看，效果波动不大，但是低频处理可以大部分还原，高频有缺失

二、 实验过程及记录

过程中需要的函数文件：

Filpframe_OverlapA.m

```
%基于重叠相加法的信号还原函数
function frameout=Filpframe_OverlapA(x,win,inc)

[nf,len]=size(x);
nx=(nf-1) *inc+len;                    %原信号长度
frameout=zeros(nx,1);
nwin=length(win);                      % 取窗长
if (nwin ~= 1)                         % 判断窗长是否为 1，若为 1，即表示没有设窗函数
    winx=repmat(win',nf,1);
    x=x./winx;                         % 除去加窗的影响
    x(find(isinf(x)))=0;               %去除除 0 得到的 Inf
end

for i=1:nf
    start=(i-1)*inc+1;
    xn=x(i,:)';
    frameout(start:start+len-1)=frameout(start:start+len-1)+xn;
end
```

Filpframe_OverlapS.m

```
%基于重叠存储法的信号还原函数
function frameout=Filpframe_OverlapS(x,win,inc)

[nf,len]=size(x);
nx=(nf-1) *inc+len;                          %原信号长度
frameout=zeros(nx,1);
nwin=length(win);                             % 取窗长
if (nwin ~= 1)                                % 判断窗长是否为 1，若为 1，即表示没有设窗函数
    winx=repmat(win',nf,1);
    x=x./winx;                                % 除去加窗的影响
    x(find(isinf(x)))=0;                      %去除除 0 得到的 Inf
end
frameout(1:len)=x(1,:);
for i=1:nf
    frameout(len+(i-1)*inc+1:len+i*inc)=x(i,len-inc+1:len);
end
```

FrameTimeC.m

```
function frameTime=FrameTimeC(frameNum,framelen,inc,fs)
% 分帧后计算每帧对应的时间
frameTime=(((1:frameNum)-1)*inc+framelen/2)/fs;
```

Enframe.m

```
function frameout=enframe(x,win,inc)
nx=length(x(:));                  % 取数据长度
nwin=length(win);                 % 取窗长
if (nwin == 1)                    % 判断窗长是否为 1，若为 1，即表示没有设窗函数
    len = win;                    % 是，帧长=win
else
    len = nwin;                   % 否，帧长=窗长
end
if (nargin < 3)                   % 如果只有两个参数，设帧 inc=帧长
    inc = len;
end
nf = fix((nx-len+inc)/inc); % 计算帧数
frameout=zeros(nf,len);           % 初始化
indf= inc*(0:(nf-1)).';       % 设置每帧在 x 中的位移量位置
inds = (1:len);                   % 每帧数据对应 1:len
frameout(:) = x(indf(:,ones(1,len))+inds(ones(nf,1),:));    % 对数据分帧
if (nwin > 1)                     % 若参数中包括窗函数，把每帧乘以窗函数
    w = win(:)';                  % 把 win 转成行数据
    frameout = frameout .* w(ones(nf,1),:);   % 乘窗函数
end
```

findSegment.m

```
function soundSegment=findSegment(express)
if express(1)==0
    voicedIndex=find(express);                          % 寻找 express 中为 1 的位置
else
    voicedIndex=express;
end

soundSegment = [];
k = 1;
soundSegment(k).begin = voicedIndex(1);                 % 设置第一组有话段的起始位置
for i=1:length(voicedIndex)-1,
    if voicedIndex(i+1)-voicedIndex(i)>1,               % 本组有话段结束
        soundSegment(k).end = voicedIndex(i);           % 设置本组有话段的结束位置
        soundSegment(k+1).begin = voicedIndex(i+1);% 设置下一组有话段的起始位置
        k = k+1;
    end
end
soundSegment(k).end = voicedIndex(end);                 % 最后一组有话段的结束位置
% 计算每组有话段的长度
for i=1 :k
    soundSegment(i).duration=soundSegment(i).end-soundSegment(i).begin+1;
end
```

linsmoothm.m

```
function [y] = linsmoothm(x,n)
if nargin< 2
    n=3;
end
win=hanning(n);                                         % 用 hanning 窗
win=win/sum(win);                                       % 归一化
x=x(:)';                                                % 把 x 转换为行序列

len=length(x);
y= zeros(len,1);                                        % 初始化 y
% 对 x 序列前后补 n 个数,以保证输出序列与 x 相同长
if mod(n, 2) ==0
    l=n/2;
    x = [ones(1,1)* x(1) x ones(1,l)* x(len)]';
else
    l=(n-1)/2;
    x = [ones(1,1)* x(1) x ones(1,l+1)* x(len)]';
```

```
end
% 线性平滑处理
for k=1:len
    y(k) = win'* x(k:k+ n- 1);
end
```

pitch_Ceps.m

```
%倒谱法基音周期检测函数
function [voiceseg,vsl,SF,Ef,period]=pitch_Ceps(x,wnd,inc,T1,fs,miniL)
if nargin<6, miniL=10; end
if length(wnd)==1
    wlen=wnd;                    % 求出帧长
else
    wlen=length(wnd);
end
y    = enframe(x,wnd,inc)';                    % 分帧
[voiceseg,vsl,SF,Ef]=pitch_vad(x,wnd,inc,T1,miniL);    % 基音的端点检测
fn=length(SF);
lmin=fix(fs/500);                              % 基音周期的最小值
lmax=fix(fs/60);                               % 基音周期的最大值
period=zeros(1,fn);                            % 基音周期初始化
for k=1:fn
    if SF(k)==1                                % 是否在有话帧中
        y1=y(:,k).*hamming(wlen);              % 取来一帧数据加窗函数
        xx=fft(y1);                            % FFT
        a=2*log(abs(xx)+eps);                  % 取模值和对数
        b=ifft(a);                             % 求取倒谱
        [R(k),Lc(k)]=max(b(lmin:lmax));        % 在 lmin 和 lmax 区间中寻找最大值
        period(k)=Lc(k)+lmin-1;                % 给出基音周期
    end
end
```

pitch_vad.m

```
% 用于基音周期估计的端点检测函数
function [voiceseg,vosl,SF,Ef]=pitch_vad(x,wnd,inc,T1,miniL)
if nargin<5, miniL=10; end
y=enframe(x,wnd,inc)';                 % 分帧
fn=size(y,2);                          % 求帧数
if length(wnd)==1
    wlen=wnd;                    % 求出帧长
else
    wlen=length(wnd);
end
```

```
for i=1:fn
    Sp = abs(fft(y(:,i)));                  % FFT 取幅值
    Sp = Sp(1:wlen/2+1);                    % 只取正频率部分
    Esum(i) = sum(Sp.*Sp);                  % 计算能量值
    prob = Sp/(sum(Sp));                    % 计算概率
    H(i) = -sum(prob.*log(prob+eps));       % 求谱熵值
end
hindex=find(H<0.1);
H(hindex)=max(H);
Ef=sqrt(1 + abs(Esum./H));                  % 计算能熵比
Ef=Ef/max(Ef);                              % 归一化

zindex=find(Ef>=T1);                        % 寻找 Ef 中大于 T1 的部分
zseg=findSegment(zindex);                   % 给出端点检测各段的信息
zsl=length(zseg);                           % 给出段数
j=0;
SF=zeros(1,fn);
for k=1 : zsl                               % 在大于 T1 中剔除小于 miniL 的部分
    if zseg(k).duration>=miniL
        j=j+1;
        in1=zseg(k).begin;
        in2=zseg(k).end;
        voiceseg(j).begin=in1;
        voiceseg(j).end=in2;
        voiceseg(j).duration=zseg(k).duration;
        SF(in1:in2)=1;                      % 设置 SF
    end
end
vosl=length(voiceseg);                      % 有话段的段数
```

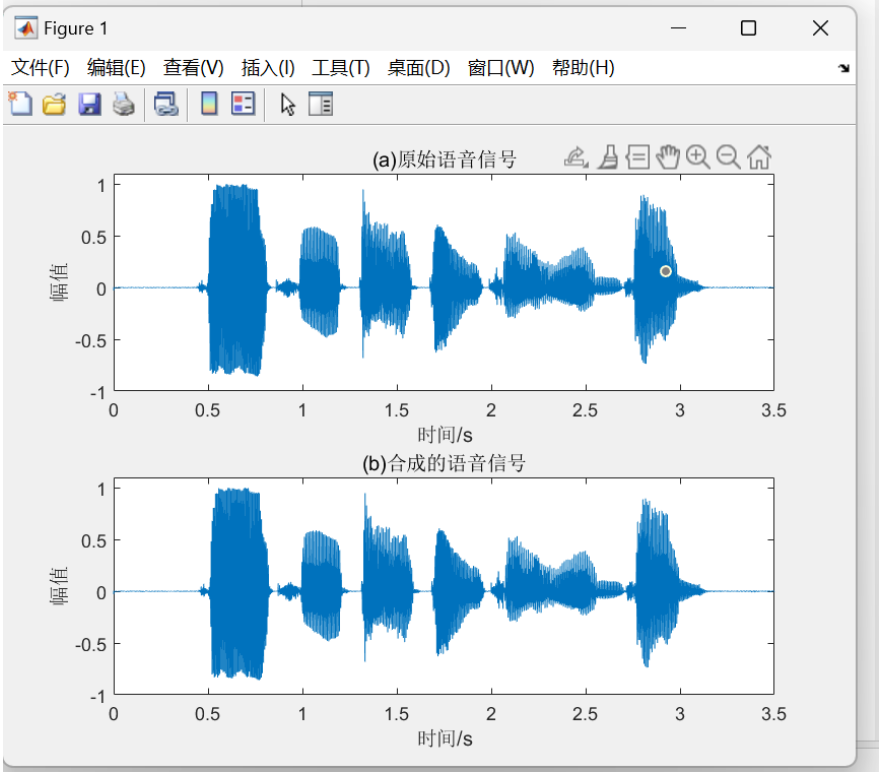
pitfilterm1.m

```
function y=pitfilterm1(x,vseg,vsl)
y=zeros(size(x));                   % 初始化
for i=1 : vsl                       % 有段数据
    ixb=vseg(i).begin;              % 该段的开始位置
    ixe=vseg(i).end;                % 该段的结束位置
    u0=x(ixb:ixe);                  % 取来一段数据
    y0=medfilt1(u0,5);              % 5 点的中值滤波
    v0=linsmoothm(y0,5);            % 线性平滑
    y(ixb:ixe)=v0;                  % 赋值给 y
end
```

1.基于线性预测系数和预测误差的语音合成原理主程序代码如下：

```
clear all;
clc;
close all;
info = audioinfo('C7_2_y.wav');
x = audioread('C7_2_y.wav');
fs = info.SampleRate;
bits = info.BitsPerSample;                  % 读入数据文件                  % 读入数据文件
x=x-mean(x);                                % 消除直流分量
x=x/max(abs(x));                            % 幅值归一
xl=length(x);                               % 数据长度
time=(0:xl-1)/fs;                           % 计算出时间刻度
p=12;                                       % LPC 的阶数为 12
wlen=200; inc=80;                           % 帧长和帧移
msoverlap = wlen - inc;                     % 每帧重叠部分的长度
y=enframe(x,wlen,inc)';                     % 分帧
fn=size(y,2);                               % 取帧数
% 语音分析:求每一帧的 LPC 系数和预测误差
for i=1 : fn
    u=y(:,i);                               % 取来一帧
    A=lpc(u,p);                             % LPC 求得系数
    aCoeff(:,i)=A;                          % 存放在 aCoeff 数组中
    errSig = filter(A,1,u);                 % 计算预测误差序列
    resid(:,i) = errSig;                    % 存放在 resid 数组中
end
% 语音合成:求每一帧的合成语音叠接成连续语音信号
for i=1:fn
    A = aCoeff(:,i);                        % 取得该帧的预测系数
    residFrame = resid(:,i);                % 取得该帧的预测误差
    synFrame(i,:) = filter(1, A', residFrame); % 预测误差激励,合成语音
end;
outspeech=Filpframe_OverlapS(synFrame,wlen,inc);
ol=length(outspeech);
if ol<xl                                    % 把 outspeech 补零,使与 x 等长
    outspeech=[outspeech zeros(1,xl-ol)];
else
    outspeech=outspeech(1:xl);
end

% 作图
subplot 211; plot(time,x);
xlabel(['时间/s']); ylabel('幅值'); ylim([-1 1.1]);
title('(a)原始语音信号')
subplot 212; plot(time,outspeech);
xlabel(['时间/s' ]); ylabel('幅值'); ylim([-1 1.1]);
```

title('(b)合成的语音信号')
结果如下：



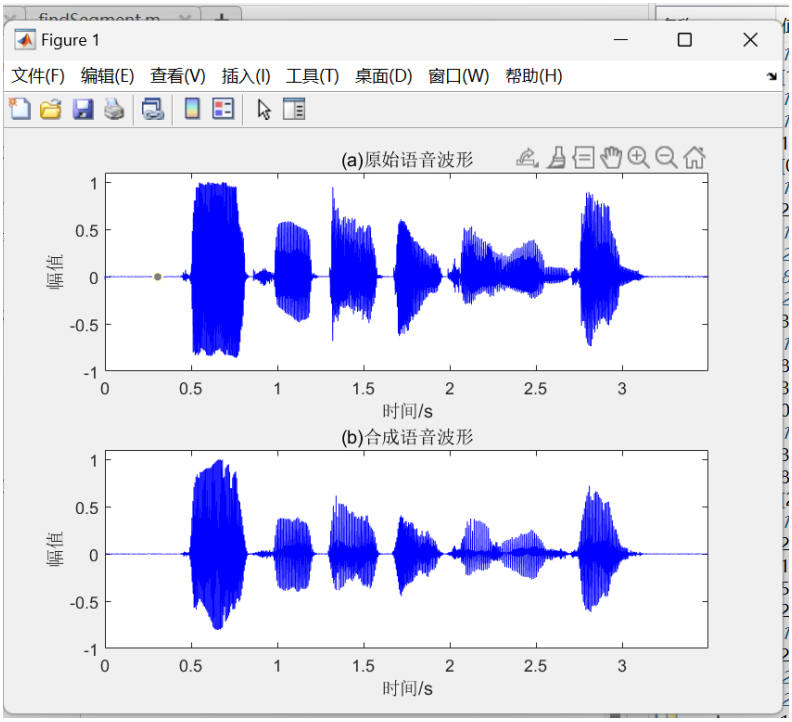
2. 基于线性预测系数和基音参数的语音合成原理主程序代码如下：

```
clear all; clc; close all;
info = audioinfo('C7_2_y.wav');
xx = audioread('C7_2_y.wav');
fs = info.SampleRate;
bits = info.BitsPerSample;                % 读入数据文件
xx=xx-mean(xx);                           % 去除直流分量
x=xx/max(abs(xx));                        % 归一化
N=length(x);                              % 数据长度
time=(0:N-1)/fs;                          % 时间刻度
wlen=240;                                 % 帧长
inc=80;                                   % 帧移
overlap=wlen-inc;                         % 重叠长度
tempr1=(0:overlap-1)'/overlap;            % 斜三角窗函数 w1
tempr2=(overlap-1:-1:0)'/overlap;         % 斜三角窗函数 w2
n2=1:wlen/2+1;                            % 正频率的下标值
wind=hanning(wlen);                       % 窗函数
X=enframe(x,wind,inc)';                   % 分帧
fn=size(X,2);                             % 帧数
T1=0.1; r2=0.5;                           % 端点检测参数
miniL=10;                                 % 有话段最短帧数
mnlong=5;                                 % 元音主体最短帧数
ThrC=[10 15];                             % 阈值
```

```
p=12;                                                    % LPC 阶次
frameTime=FrameTimeC(fn,wlen,inc,fs);          % 计算每帧的时间刻度
for i=1 : fn                                             % 计算每帧的线性预测系数和增益
    u=X(:,i);
    [ar,g]=lpc(u,p);
    AR_coeff(:,i)=ar;
    Gain(i)=g;
end
% 基音检测
[voiceseg,vosl,SF,Ef,period]=pitch_Ceps(x,wlen,inc,T1,fs); %基于倒谱法的基音周期检测
Dpitch=pitfilterm1(period,voiceseg,vosl);            % 对 T0 进行平滑处理求出基音周期 T0

tal=0;                                                   % 初始化前导零点
zint=zeros(p,1);
for i=1:fn;
    ai=AR_coeff(:,i);                                % 获取第 i 帧的预测系数
    sigma_square=Gain(i);                          % 获取第 i 帧的增益系数
    sigma=sqrt(sigma_square);

    if SF(i)==0                                        % 无话帧
        excitation=randn(wlen,1);                % 产生白噪声
        [synt_frame,zint]=filter(sigma,ai,excitation,zint); % 用白噪声合成语音
    else                                                 % 有话帧
        PT=round(Dpitch(i));                       % 取周期值
        exc_syn1 =zeros(wlen+tal,1);           % 初始化脉冲发生区
        exc_syn1(mod(1:tal+wlen,PT)==0) = 1;   % 在基音周期的位置产生脉冲，幅值为 1
        exc_syn2=exc_syn1(tal+1:tal+inc); % 计算帧移 inc 区间内脉冲个数
        index=find(exc_syn2==1);
        excitation=exc_syn1(tal+1:tal+wlen);% 这一帧的激励脉冲源

        if isempty(index)                          % 帧移 inc 区间内没有脉冲
            tal=tal+inc;                           % 计算下一帧的前导零点
        else                                         % 帧移 inc 区间内有脉冲
            eal=length(index);                 % 计算有几个脉冲
            tal=inc-index(eal);                % 计算下一帧的前导零点
        end
        gain=sigma/sqrt(1/PT);                 % 增益
        [synt_frame,zint]=filter(gain, ai,excitation,zint); % 用脉冲合成语音
    end
        if i==1                                      % 若为第 1 帧
            output=synt_frame;                 % 不需要重叠相加,保留合成数据
        else
            M=length(output);                  % 按线性比例重叠相加处理合成数据
            output=[output(1:M-overlap);
```

```
output(M-overlap+1:M).*tempr2+synt_frame(1:overlap).*tempr1; synt_frame(overlap+1:wlen)];
        end
end
ol=length(output);                              % 把输出 output 延长至与输入信号 xx 等长
if ol<N
    output1=[output; zeros(N-ol,1)];
else
    output1=output(1:N);
end
bn=[0.964775    -3.858862    5.788174    -3.858862    0.964775]; % 滤波器系数
an=[1.000000    -3.928040    5.786934    -3.789685    0.930791];
output=filter(bn,an,output1);                   % 高通滤波
output=output/max(abs(output));                 % 幅值归一

subplot 211; plot(time,x,'b'); title('(a)原始语音波形');
axis([0 max(time) -1 1.1]); xlabel('时间/s'); ylabel('幅值')
subplot 212; plot(time,output,'b');   title('(b)合成语音波形');
axis([0 max(time) -1 1.1]); xlabel('时间/s'); ylabel('幅值')
```

实验结果如下：



三、 实验结果分析

原始语音波形（a）
时间范围：波形图覆盖的时间范围是 1.5 秒到 2.5 秒。
振幅变化：在这段时间内，原始语音波形的振幅有显著的变化，这表明说话者在这段时间内可能在发出不同响度的声音。
频率变化：波形的频率变化显示了声音的音调变化。波形的某些部分可能代表高音调，而其他

部分可能代表低音调。
自然性：原始语音波形通常更加不规则和复杂，因为它包含了说话者的自然语音特征，如语调、情感和发音方式。
合成语音波形（b）
时间范围：与原始语音波形相同，合成语音波形也覆盖了 1.5 秒到 2.5 秒的时间范围。
振幅变化：合成语音波形的振幅变化可能更加规则和一致，这表明合成语音在响度上可能更加稳定。
频率变化：合成语音波形的频率变化可能旨在模拟原始语音的音调变化，但可能不如原始语音波形那样自然和丰富。
模拟度：合成语音波形的目的是为了模拟人类语音，因此它的设计旨在尽可能接近自然语音的特征，但可能在某些方面（如自然性和情感表达）与原始语音有所差异。
总结：
对比分析：通过比较原始语音波形和合成语音波形，我们可以观察到两者在振幅和频率上的差异。原始语音波形通常更加自然和动态，而合成语音波形则更加规则和一致。

四、实验总结（200-300 字）

在本次实验中，我们成功掌握了语音合成的原理，并实现了从理论到实践的转变。通过编程，我们利用线性预测系数和基音参数合成了语音信号，深入理解了滤波器模型的激励和声道模型的滤波过程。
实验中，我们首先分析了原始语音信号，提取了线性系数和基音周期，然后通过设计的算法将这些参数合成了连续的语音信号。实验结果表明，合成语音在振幅和频率上与原始语音相比有一定的一致性，但在自然性与原始语音存在差异。
通过本次实验，我们不仅加深了对语音合成技术的理解，而且提高了编程和分析能力