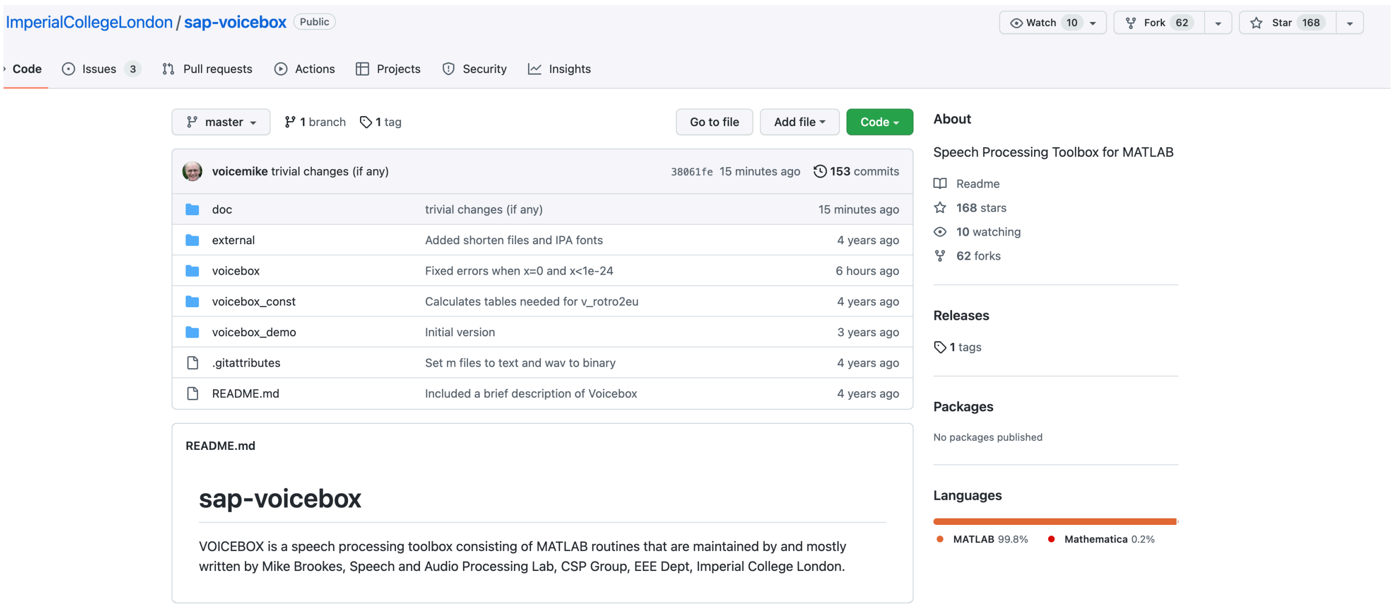
《Matlab信号处理及仿真》第二次作业报告

撰写人：邱楚寒　学号：2020209023026　班级：录音工程

Matlab官方库中是没有单纯对音频进行分帧的函数的

因此需要自行下载voicebox库: https://github.com/ImperialCollegeLondon/sap-voicebox



ImperialCollegeLondon / sap-voicebox Public

Watch 10 · Fork 62 · Star 168

Code · Issues 3 · Pull requests · Actions · Projects · Security · Insights

master · 1 branch · 1 tag · Go to file · Add file · Code

voicemike trivial changes (if any) · 38061fe 15 minutes ago · 153 commits

doc	trivial changes (if any)	15 minutes ago
external	Added shorten files and IPA fonts	4 years ago
voicebox	Fixed errors when x=0 and x<1e-24	6 hours ago
voicebox_const	Calculates tables needed for v_rotro2eu	4 years ago
voicebox_demo	Initial version	3 years ago
.gitattributes	Set m files to text and wav to binary	4 years ago
README.md	Included a brief description of Voicebox	4 years ago

README.md

sap-voicebox

VOICEBOX is a speech processing toolbox consisting of MATLAB routines that are maintained by and mostly written by Mike Brookes, Speech and Audio Processing Lab, CSP Group, EEE Dept, Imperial College London.

About

Speech Processing Toolbox for MATLAB

Readme · 168 stars · 10 watching · 62 forks

Releases

1 tags

Packages

No packages published

Languages

MATLAB 99.8% · Mathematica 0.2%

复制到matlab/toolbox路径下，在Matlab命令行使用addpath对路径进行添加，才能使用voicebox库中的enframe函数进行分帧。

```
>> which voicebox.m
/Applications/MATLAB_R2021b.app/toolbox/voicebox/voicebox.m
```

在对音频信号进行分帧之前需要进行预处理。预加重是为了补偿口唇辐射阻抗造成的高频损失，为了对上课内容进行更完整的仿真，本次使用语音信号进行实验。在预加重前也进行了去直流和归一化处理，让对比分析结果更加直观。

接下来，在正式进行分帧前，先仔细阅读enframe函数每个传入参数代表的含义：

```
function varargout=enframe(varargin)
%
% For calling details please see v_enframe.m
%
% This dummy routine is included for backward compatibility only
% and will be removed in a future release of voicebox. Please use
% v_enframe.m in future and/or update with v_voicebox_update.m
%
%      Copyright (C) Mike Brookes 2018
%      Version: $Id: enframe.m 10863 2018-09-21 15:39:23Z dmb $
%
if nargout
    varargout=cell(1,nargout);
    [varargout{:}]=v_enframe(varargin{:});
else
    v_enframe(varargin{:});
end
```

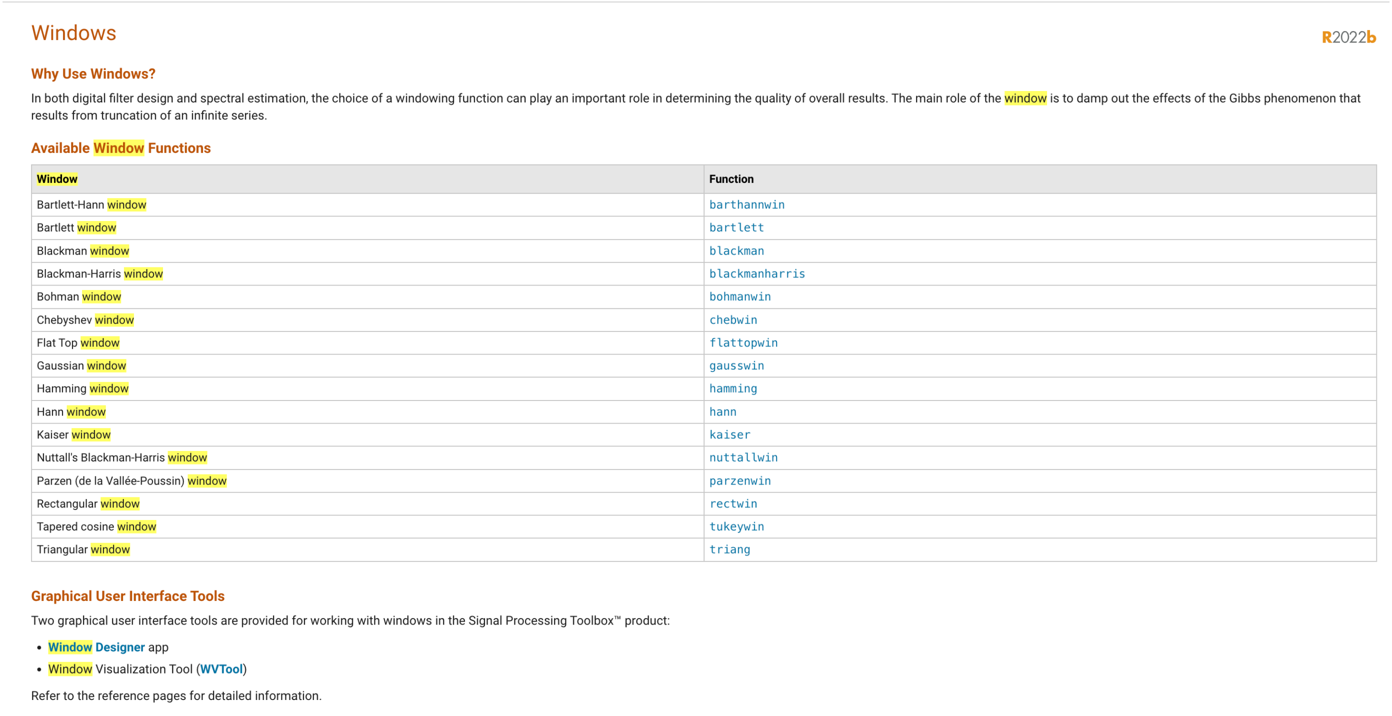
```matlab
function [f,t,w]=v_enframe(x,win,hop,m,fs)
%V_ENFRAME split signal up into (overlapping) frames: one per row. [F,T]=(X,WIN,HOP)
%
% Usage:  (1) f=v_enframe(x,n)                          % split into frames of length n
%         (2) f=v_enframe(x,hamming(n,'periodic'),n/4)  % use a 75% overlapped Hamming window of length n
%         (3) calculate spectrogram in units of power per Hz
%
%               W=hamming(NW);                      % analysis window (NW = fft length)
%               P=v_enframe(S,W,HOP,'sdp',FS);        % computer first half of PSD (HOP = frame increment in samples)
%
%         (3) frequency domain frame-based processing:
%
%               S=...;                              % input signal
%               OV=2;                               % overlap factor of 2 (4 is also often used)
%               NW=160;                             % DFT window length
%               W=sqrt(hamming(NW,'periodic'));     % omit sqrt if OV=4
%               [F,T,WS]=v_enframe(S,W,1/OV,'fa');    % do STFT: one row per time frame, +ve frequencies only
%               ... process frames ...
%               X=v_overlapadd(v_irfft(F,NW,2),WS,HOP); % reconstitute the time waveform with scaled window (omit "X=" to plot waveform)
%
%  Inputs:   x    input signal
%          win    window or window length in samples
%          hop    frame increment or hop in samples or fraction of window [window length]
%            m    mode input:
%                  'z'  zero pad to fill up final frame
%                  'r'  reflect last few samples for final frame
%                  'A'  calculate the t output as the centre of mass
%                  'E'  calculate the t output as the centre of energy
%                  'f'  perform a 1-sided dft on each frame (like v_rfft)
%                  'F'  perform a 2-sided dft on each frame using fft
%                  'p'  calculate the 1-sided power/energy spectrum of each frame
%                  'P'  calculate the 2-sided power/energy spectrum of each frame
%                  'a'  scale window to give unity gain with overlap-add
%                  's'  scale window so that power is preserved: sum(mean(v_enframe(x,win,hop,'sp'),1))=mean(x.^2)
%                  'S'  scale window so that total energy is preserved: sum(sum(v_enframe(x,win,hop,'Sp')))=sum(x.^2)
%                  'd'  make options 's' and 'S' give power/energy per Hz: sum(mean(v_enframe(x,win,hop,'sp'),1))*fs/length(win)=mean(x.^2)
%           fs    sample frequency (only needed for 'd' option) [1]
%
% Outputs:   f    enframed data - one frame per row
%            t    fractional time in samples at the centre of each frame
%                 with the first sample being 1.
%            w    window function used
```

> 可以看到一个有趣的事情是，分帧过程并没有写在enframe函数内部，真正的实现过程在v_enframe函数中，enframe只是作为v_enframe的调用接口（实际测试直接调用v_enframe和enframe并无差别）。并且在翻阅voicebox的源码后，发现其中基本所有函数结构都是这样的，通过作者的注释来看这样设计是为了方便版本更新向后兼容。

回归正题，查看v_enframe的输入参数注释：x代表输入数据；win代表窗口长度或窗类型，**因此其实并不用单独再多写一个循环去进行加窗，v_enframe(或者说enframe)函数本身在分帧时就可以进行加窗了**，这样可以直接减少一次对数据的遍历；hop代表帧移，m则代表输入模式的设置。

> 关于mode input参数，'z'和'r'是老师上课时提到过对尾帧的处理方法，通过注释可以发现，如果不传入'z'或'r'的话，**v_enframe函数会默认向下取整，忽略最后长度不足一帧的部分采样点**；
>
> 使用'f'或'F'可以进行单边/双边FFT的运算，此时v_enframe的返回就相当于变成了stft，若使用'p'或'P'还可以返回能量/功率谱；
>
> Usage的最后一个例子中，利用v_overlapadd和v_irfft（单边快速傅立叶逆变换）函数可以由语谱图(stft的结果，储存形式为矩阵)重构出音频数据的原始波形，我认为这就是传入的'a'参数的意义（不知道具体是如何实现的）；
>
> 's'或'S'需要和'p'或'P'参数一起使用，作用是返回音频数据的功率和能量大小，如果同时还传入了'd'参数，则还需传入频谱分辨率参数(fs)，此时返回量变成每Hz的功率/能量大小(Hz的步长由频谱分辨率决定)。

阅读完enflame(v_enframe)函数的源码后，本次作业还有一个比较重要的部分是窗函数的调用。查找Matlab官方文档上对窗函数的介绍：



Windows

R2022b

Why Use Windows?

In both digital filter design and spectral estimation, the choice of a windowing function can play an important role in determining the quality of overall results. The main role of the window is to damp out the effects of the Gibbs phenomenon that results from truncation of an infinite series.

Available Window Functions

Window	Function
Bartlett-Hann window	`barthannwin`
Bartlett window	`bartlett`
Blackman window	`blackman`
Blackman-Harris window	`blackmanharris`
Bohman window	`bohmanwin`
Chebyshev window	`chebwin`
Flat Top window	`flattopwin`
Gaussian window	`gausswin`
Hamming window	`hamming`
Hann window	`hann`
Kaiser window	`kaiser`
Nuttall's Blackman-Harris window	`nuttallwin`
Parzen (de la Vallée-Poussin) window	`parzenwin`
Rectangular window	`rectwin`
Tapered cosine window	`tukeywin`
Triangular window	`triang`

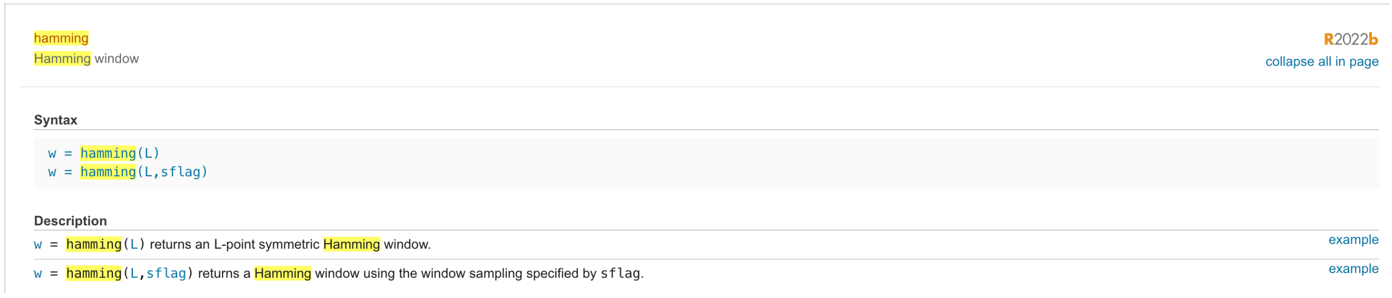
Graphical User Interface Tools

Two graphical user interface tools are provided for working with windows in the Signal Processing Toolbox™ product:

- **Window Designer** app
- Window Visualization Tool (**WVTool**)

Refer to the reference pages for detailed information.

其中部分窗函数除了传入窗长作为参数，还可以传入窗口采样改变窗口样式，比如Hamming窗：



hamming
Hamming window

R2022b
collapse all in page

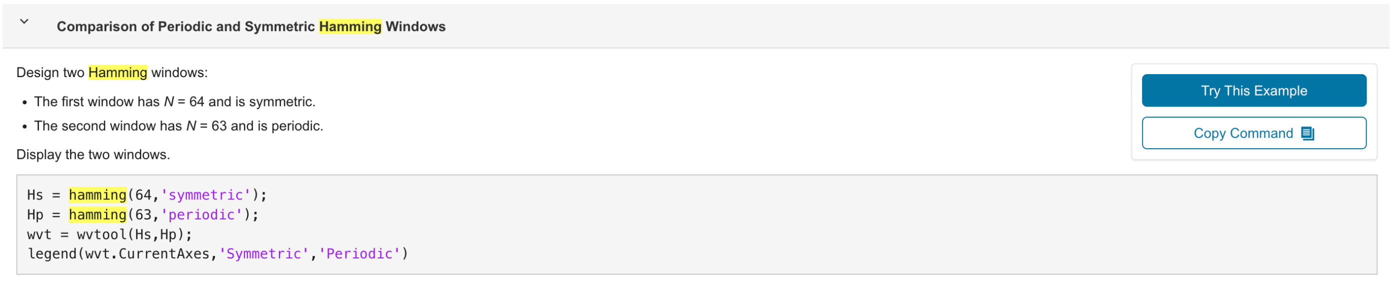
Syntax

```
w = hamming(L)
w = hamming(L,sflag)
```

Description

`w = hamming(L)` returns an L-point symmetric Hamming window. example

`w = hamming(L,sflag)` returns a Hamming window using the window sampling specified by `sflag`. example



Comparison of Periodic and Symmetric Hamming Windows

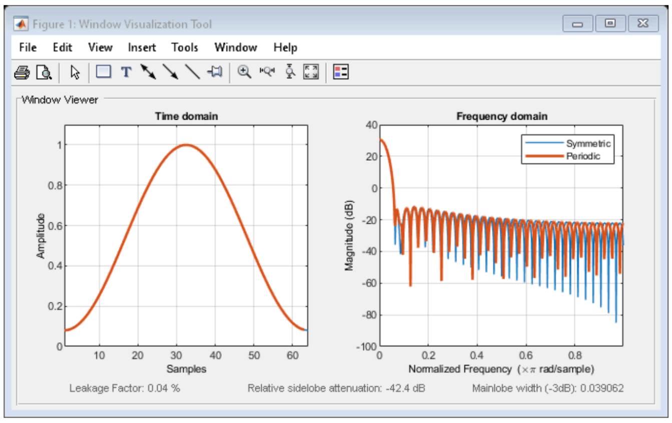
Design two Hamming windows:

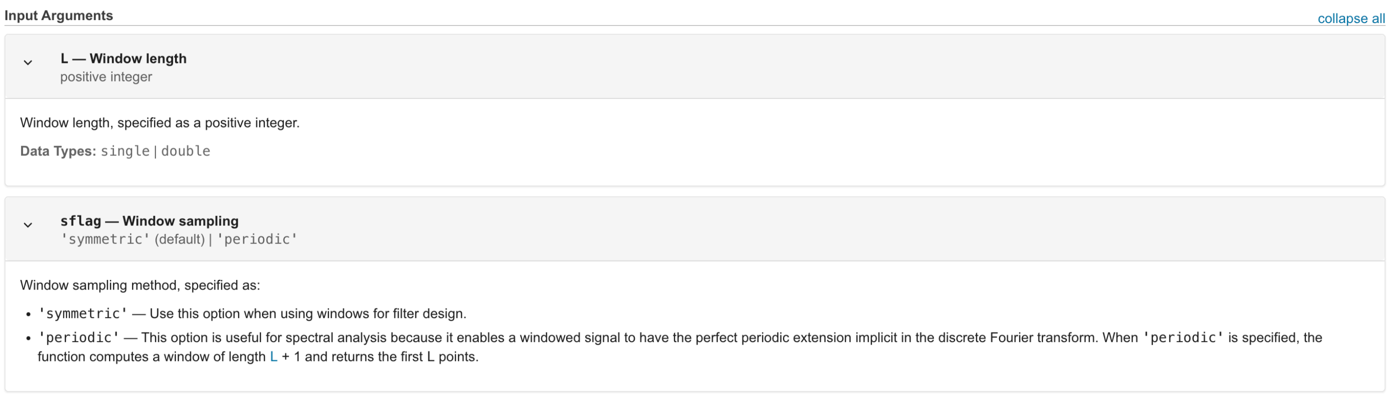
- The first window has *N* = 64 and is symmetric.
- The second window has *N* = 63 and is periodic.

Display the two windows.

Try This Example
Copy Command

```
Hs = hamming(64,'symmetric');
Hp = hamming(63,'periodic');
wvt = wvtool(Hs,Hp);
legend(wvt.CurrentAxes,'Symmetric','Periodic')
```





Input Arguments collapse all

L — Window length
positive integer

Window length, specified as a positive integer.

Data Types: `single` | `double`

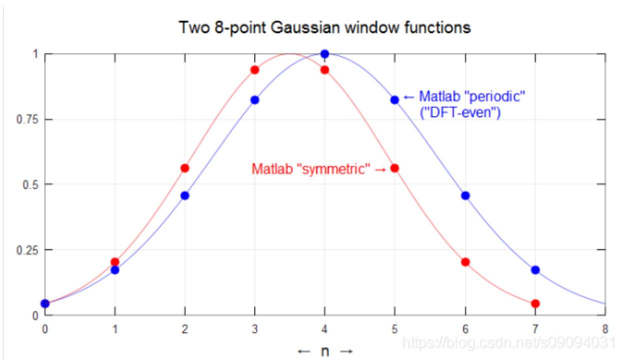
`sflag` — Window sampling
`'symmetric'` (default) | `'periodic'`

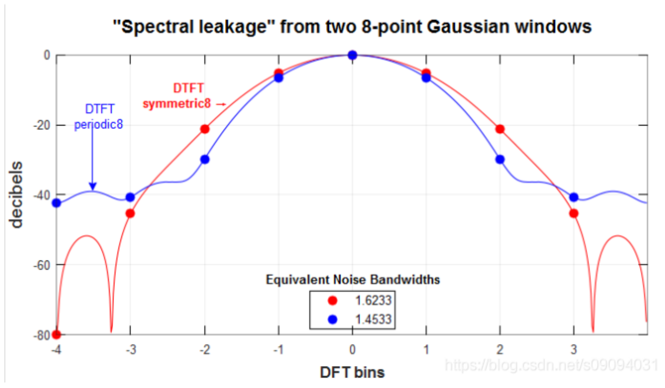
Window sampling method, specified as:

- `'symmetric'` — Use this option when using windows for filter design.
- `'periodic'` — This option is useful for spectral analysis because it enables a windowed signal to have the perfect periodic extension implicit in the discrete Fourier transform. When `'periodic'` is specified, the function computes a window of length L + 1 and returns the first L points.

默认样式为symmetric，即普通的对称窗，periodic为周期窗。

两者在时域上的波形相差不大，但根据periodic简介，样例中periodic样式传入的窗长参数为63，函数会计算出长度为64的窗并只返回其前63个点（我认为是为了周期延拓所以舍弃最后一个点，下一个周期开始的第一个点作为上个周期的末尾点，因而说它“隐含完美的周期延展性“）。两者在频响上的区别则非常巨大，从wiki上找到的两张图如下所示：





> 网上有一些结论说两者主要区别是主瓣和旁瓣的区别，但我认为主瓣和旁瓣是区分不同窗函数的主要特征，symmetric和periodic的差别应该是周期延展性，如官方文档所说，symmetric用于设计滤波器（FIR窗函数设计法），periodic用于进行频谱分析（DFT本身也有周期延展性的，两者相对应）

对于本次作业而言，如果是为了减少分帧过程中产生的频谱泄漏，使用默认的symmetric样式可满足要求。

有了以上的理论基础，便可以开始愉快地写代码了

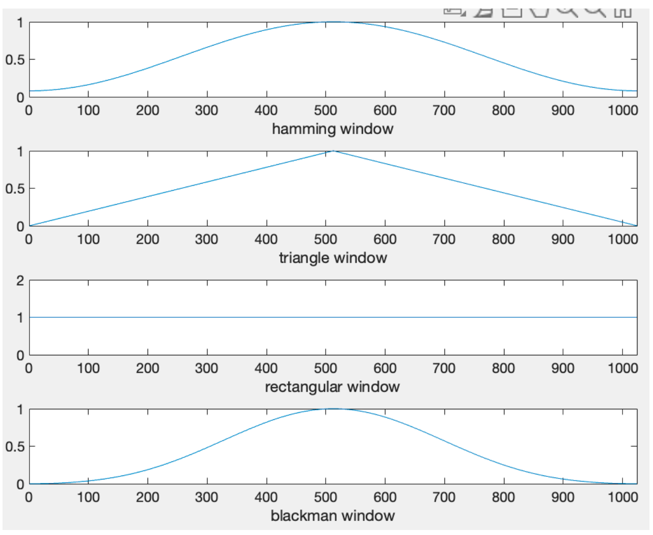
使用audioread函数读取音频数据，向mean()函数传入读取的数据返回平均值，从原数据减去均值完成去均值处理；max(abs())得到音频数据绝对值最大值，作为除数完成归一化处理；利用简单的一阶高通滤波器完成预加重处理（此处使用的信号为语音信号）

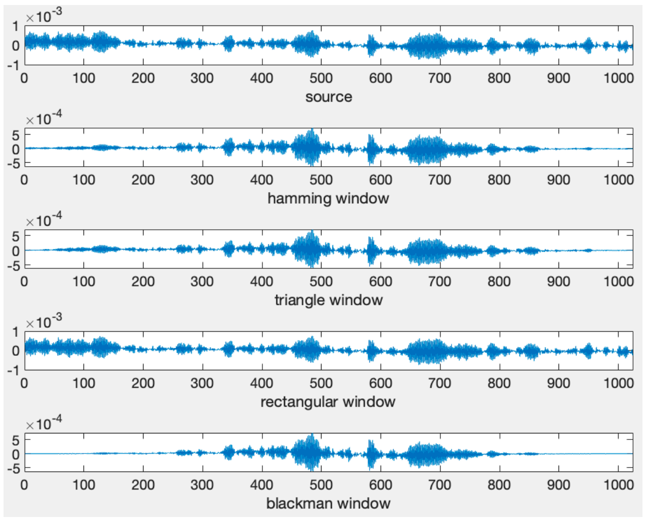
完成上述三步后便完成了数据的预处理操作。接下来利用enframe/v_enframe函数对音频进行分帧。如上所述，加窗也可以在这一步完成，但为了使代码步骤更加清晰，我选择把两步分开进行，先利用enframe单独只进行分帧，再单独定义窗函数，使用循环语句完成加窗。

为增加代码简洁性，我把加窗过程封装为函数，特别注意，窗函数的保存形式为列向量，音频分帧后行向量代表的是每一帧的采样点。因此使用.*的话需要先将窗函数转置为行向量，才能将窗函数与每一帧的信号进行相乘。

> 这是我在写代码时因为对.*不熟悉碰到的一个比较有趣的bug，写在这里记录一下。
>
> ```
> 请求的 297385x297385 (658.9GB)数组超过预设的最大数组大小(32.0GB)。这可能会导致 MATLAB 无响应。
>
> 出错 homework_2 (第 36 行)
> sound(y .* hamming(length(y))',  fs); pause(time);
> ```
>
> 此处我希望能对整段音频进行加窗并播放，导致内存溢出了.....检查了一遍变量类型，发现：
>
> | | |
> |---|---|
> | y | 297385x1 double |
> | hamm_win | 1024x1 double |
>
> (这里hamm_win只是举个例子)原始音频数据和窗函数均为列向量，如果先转置在进行.*，变成行向量和列向量相乘，计算复杂度为O(n^2)；去掉转置符，复杂度变成O(n)，便不会导致内存溢出了。当然，即便不会导致内存溢出先前那样计算得到的结果也是错误的。

接下来画出每个窗函数和加窗后第一帧内的波形，如下图所示（为了作为特例对比，我这里加了矩形窗）：





可以看到，加窗后的波形相比原始波形两端有不同程度的衰减（矩形窗除外），对于不同窗，波形曲线不同，因此衰减的趋势也不一样，比如Blackman窗相比Hamming窗的两端衰减到了0，故其加Hamming窗的两端相比Blackman要大一些；Triangle窗和Blackman窗的两端都衰减至0，但Triangle窗的斜率绝对值整体比Blackman窗更大，因此其静音段（电平非常小的部分）比Blackman窗要少很多。

经过上述分析后，我们可以发现，窗函数在时域的主要作用是对信号的两端进行衰减，从而减少因截断效应而产生的频谱泄漏。因此，如果整段音频都经过窗函数处理，会呈现一个声音信号先小后大又变小的趋势变化。为了验证这个猜想，我将整段音频数据与与之相同长度的窗函数相乘，并用sound函数播放，最终听感与窗函数形状非常相关。

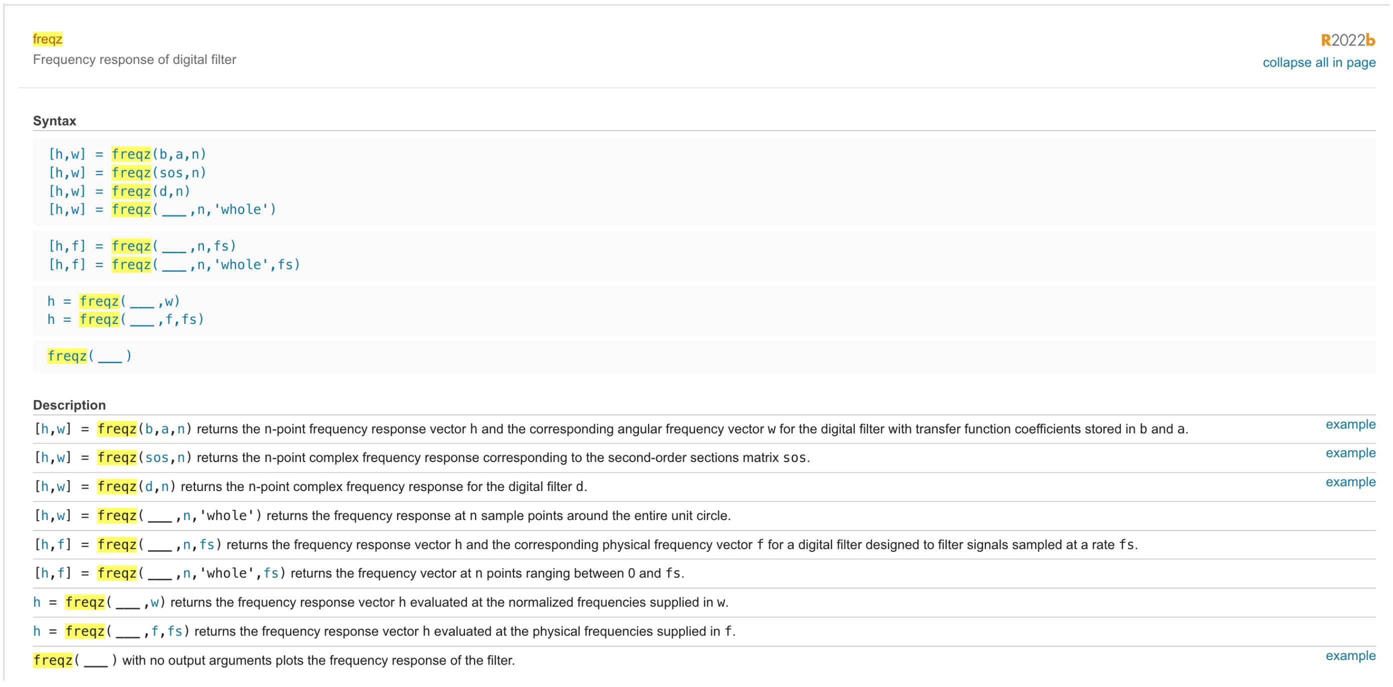
1、Hamming：和Triangle窗相比两者听感极为相似，理论上Hamming窗的起始电平要比Triangle窗大，然后Triangle窗的电平会超过Hamming（斜率更大），在中间时两者响度相等，后半部分则是逆向过程，但实际听感并不明显。

2、Triangle：同上。

3、Rectangular：与原始音频没有任何区别。

4、Blackman：整体响度是所有窗中最小的，连开头的“嗯”都已经听不到了。

除了上述在时域的分析外，上课还提到了freqz函数的使用，Matlab官方文档对freqz函数的介绍如下：



freqz
Frequency response of digital filter

R2022b
collapse all in page

Syntax

```
[h,w] = freqz(b,a,n)
[h,w] = freqz(sos,n)
[h,w] = freqz(d,n)
[h,w] = freqz(___,n,'whole')

[h,f] = freqz(___,n,fs)
[h,f] = freqz(___,n,'whole',fs)

h = freqz(___,w)
h = freqz(___,f,fs)

freqz(___)
```

Description

`[h,w] = freqz(b,a,n)` returns the n-point frequency response vector h and the corresponding angular frequency vector w for the digital filter with transfer function coefficients stored in b and a. example

`[h,w] = freqz(sos,n)` returns the n-point complex frequency response corresponding to the second-order sections matrix `sos`. example

`[h,w] = freqz(d,n)` returns the n-point complex frequency response for the digital filter d. example

`[h,w] = freqz(___,n,'whole')` returns the frequency response at n sample points around the entire unit circle.

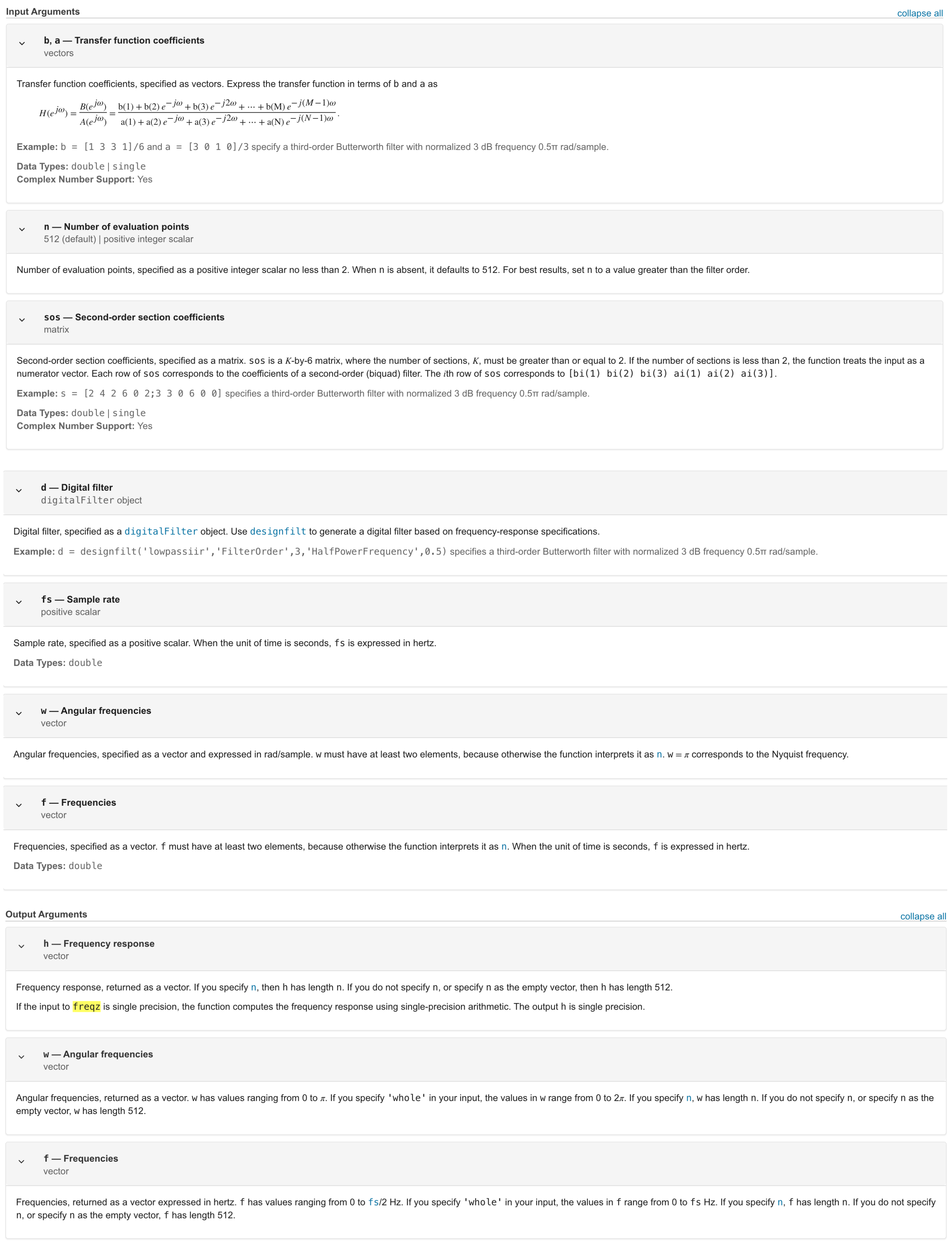
`[h,f] = freqz(___,n,fs)` returns the frequency response vector h and the corresponding physical frequency vector `f` for a digital filter designed to filter signals sampled at a rate `fs`.

`[h,f] = freqz(___,n,'whole',fs)` returns the frequency vector at n points ranging between 0 and `fs`.

`h = freqz(___,w)` returns the frequency response vector h evaluated at the normalized frequencies supplied in `w`.

`h = freqz(___,f,fs)` returns the frequency response vector h evaluated at the physical frequencies supplied in `f`.

`freqz(___)` with no output arguments plots the frequency response of the filter. example



Input Arguments

collapse all

b, a — Transfer function coefficients
vectors

Transfer function coefficients, specified as vectors. Express the transfer function in terms of b and a as

$$H(e^{j\omega}) = \frac{B(e^{j\omega})}{A(e^{j\omega})} = \frac{\mathrm{b}(1) + \mathrm{b}(2)\, e^{-j\omega} + \mathrm{b}(3)\, e^{-j2\omega} + \cdots + \mathrm{b}(\mathrm{M})\, e^{-j(M-1)\omega}}{\mathrm{a}(1) + \mathrm{a}(2)\, e^{-j\omega} + \mathrm{a}(3)\, e^{-j2\omega} + \cdots + \mathrm{a}(\mathrm{N})\, e^{-j(N-1)\omega}}.$$

Example: `b = [1 3 3 1]/6` and `a = [3 0 1 0]/3` specify a third-order Butterworth filter with normalized 3 dB frequency 0.5π rad/sample.

Data Types: `double` | `single`
Complex Number Support: Yes

n — Number of evaluation points
512 (default) | positive integer scalar

Number of evaluation points, specified as a positive integer scalar no less than 2. When n is absent, it defaults to 512. For best results, set n to a value greater than the filter order.

`sos` — Second-order section coefficients
matrix

Second-order section coefficients, specified as a matrix. `sos` is a K-by-6 matrix, where the number of sections, K, must be greater than or equal to 2. If the number of sections is less than 2, the function treats the input as a numerator vector. Each row of `sos` corresponds to the coefficients of a second-order (biquad) filter. The ith row of `sos` corresponds to `[bi(1) bi(2) bi(3) ai(1) ai(2) ai(3)]`.

Example: `s = [2 4 2 6 0 2;3 3 0 6 0 0]` specifies a third-order Butterworth filter with normalized 3 dB frequency 0.5π rad/sample.

Data Types: `double` | `single`
Complex Number Support: Yes

d — Digital filter
`digitalFilter` object

Digital filter, specified as a `digitalFilter` object. Use `designfilt` to generate a digital filter based on frequency-response specifications.

Example: `d = designfilt('lowpassiir','FilterOrder',3,'HalfPowerFrequency',0.5)` specifies a third-order Butterworth filter with normalized 3 dB frequency 0.5π rad/sample.

`fs` — Sample rate
positive scalar

Sample rate, specified as a positive scalar. When the unit of time is seconds, `fs` is expressed in hertz.

Data Types: `double`

`w` — Angular frequencies
vector

Angular frequencies, specified as a vector and expressed in rad/sample. w must have at least two elements, because otherwise the function interprets it as n. w = π corresponds to the Nyquist frequency.

`f` — Frequencies
vector

Frequencies, specified as a vector. `f` must have at least two elements, because otherwise the function interprets it as n. When the unit of time is seconds, `f` is expressed in hertz.

Data Types: `double`

Output Arguments

collapse all

h — Frequency response
vector

Frequency response, returned as a vector. If you specify n, then h has length n. If you do not specify n, or specify n as the empty vector, then h has length 512.

If the input to `freqz` is single precision, the function computes the frequency response using single-precision arithmetic. The output h is single precision.

`w` — Angular frequencies
vector

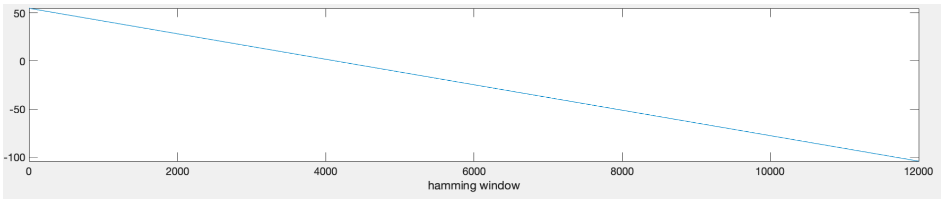
Angular frequencies, returned as a vector. w has values ranging from 0 to π. If you specify `'whole'` in your input, the values in w range from 0 to 2π. If you specify n, w has length n. If you do not specify n, or specify n as the empty vector, w has length 512.

`f` — Frequencies
vector

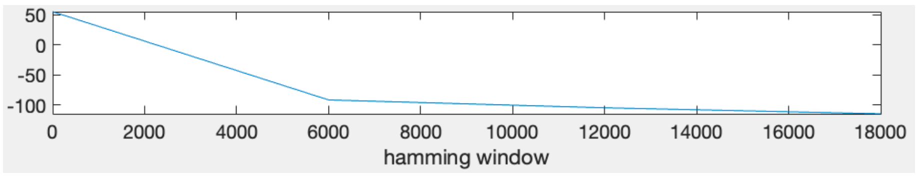
Frequencies, returned as a vector expressed in hertz. `f` has values ranging from 0 to `fs`/2 Hz. If you specify `'whole'` in your input, the values in `f` range from 0 to `fs` Hz. If you specify n, `f` has length n. If you do not specify n, or specify n as the empty vector, `f` has length 512.

其与python的spicy.signal.freqz功能相似，都是根据系数输出数字滤波器的频响特性。其中有几个可传入的参数，b、a分别代表系统函数的分子和分母；关于参数n的测试如下：

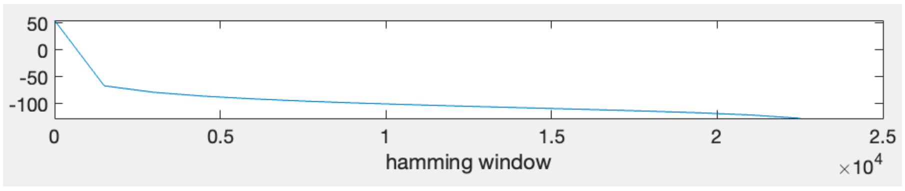
> n=2时输出



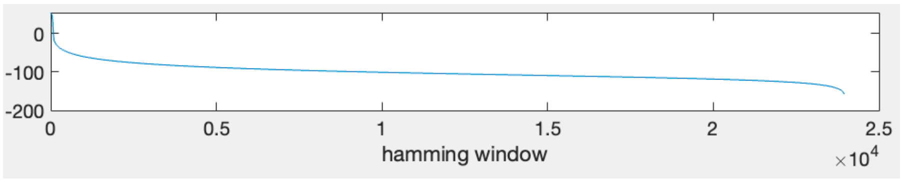
> n=4时输出



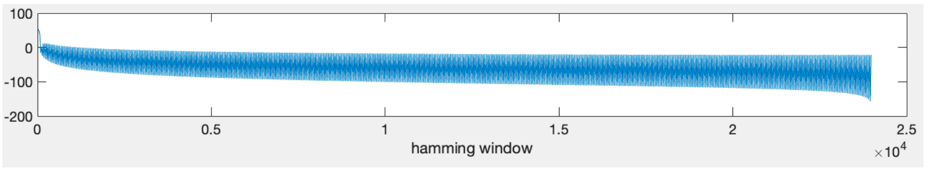
> n=16时输出



> n=512时输出



> n=1024时输出



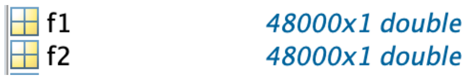
可以看到参数n代表最终输出频响的采样点数量。

sos为二阶截面系数，与[b, a]属于替代关系，如官方注释所说，sos的形式为k*6大小的矩阵，第i行的系数依次代表[b_i(1) b_i(2) b_i(3) a_i(1) a_i(2) a_i(3)]，此外，如果k=1，则传入的一行均视为分子系数。

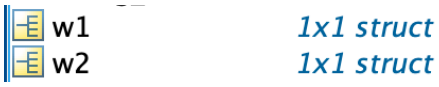
fs代表输入数据的采样率，根据奈奎斯特采样定律，信号的最高频率分量不得大于其采样率的一半，因此freqz函数输出的频响大小会小于fs/2；使用audioread获取的采样率即可；w和f分别代表输出频响横轴的角频率和频率形式，当传入两者之一时输出的频响将按照传入的向量分布。

对于freqz函数的输出就非常有趣了。h毫无疑问代表最终的频响函数。但后两者w和f，分别表示频响函数横坐标的角频率和线性频率形式，但经过测试，w和f并不能同时返回：不传入参数n时，函数默认返回的角频率形式；传入n时，返回的是线性频率形式。如下：

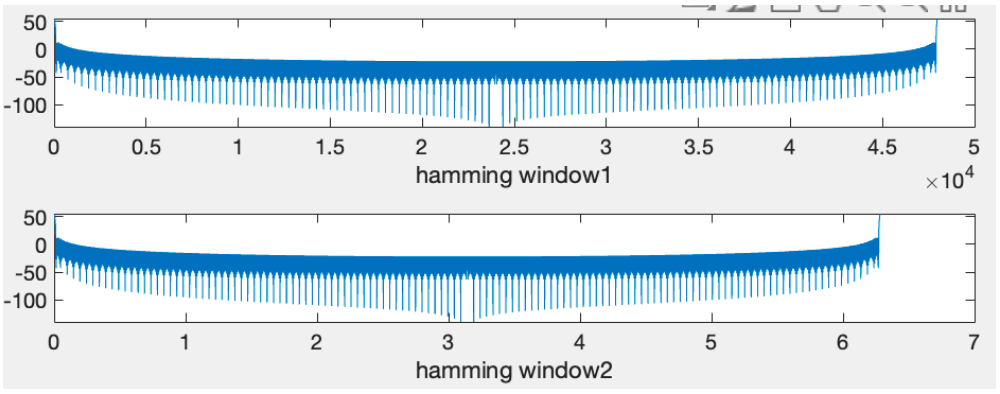
```
N = 48000;
[H_hamm1, f1, w1] = freqz(hamm_win, 1, N, "whole", fs);
[H_hamm2, f2, w2] = freqz(hamm_win, 1, "whole", fs);
```



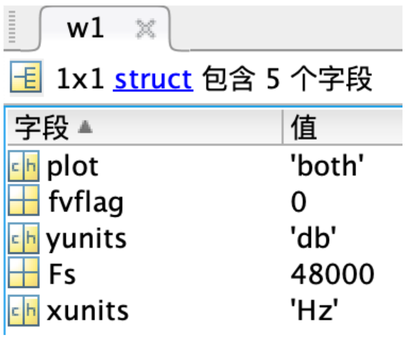
```
f1          48000x1 double
f2          48000x1 double
```



```
w1          1x1 struct
w2          1x1 struct
```



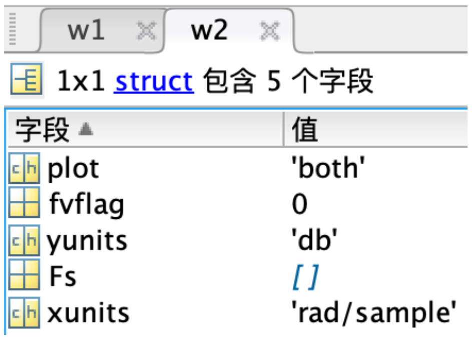
> 其中关于w1和w2实际上是一个结构体：



w1

1x1 struct 包含 5 个字段

字段 ▲	值
plot	'both'
fvflag	0
yunits	'db'
Fs	48000
xunits	'Hz'



w1 w2

1x1 struct 包含 5 个字段

字段 ▲	值
plot	'both'
fvflag	0
yunits	'db'
Fs	[ ]
xunits	'rad/sample'

若想让freqz函数返回线性频率就必须同时传入n和fs参数，返回的第三个变量只代表本次返回变量的一些特征（采样率、线性还是角频率等），这是官方文档中没有提及的。

> 本次实验主要有以下几个问题未能完全解决：

1、voicebox的函数结构为什么为了向后兼容要将接口设计成这样；

2、v_enframe函数mode input中'a'起到的具体作用是什么；

2、symmetric和periodic窗区别的猜想是否正确。