STFT uses overlap-add to reconstruct the signal

tags: STFT overlap-add Frequency domain FIR Window function COLA

STFT

Short-time Fourier change (STFT) is often used to analyze time-varying signals such as speech. The formula is as follows

$$Y(m,w)=\sum_{n=-\infty}^{+\infty} x(n)W(n-mR)e^{-jwn}$$

Y (m , w) = ∑ n = − ∞ + ∞ x (n) W (n − m R) e − j w n

The simple explanation is to use a window frame to calculate a DFT, and then calculate the DFT by translating the window R point to calculate the DFT...
Such calculations are carried out section by section, which is equivalent to artificially recording the time node, and finally obtained $Y(m,w)$ Y (m , w) With both time and frequency

information, the example window moves as shown below

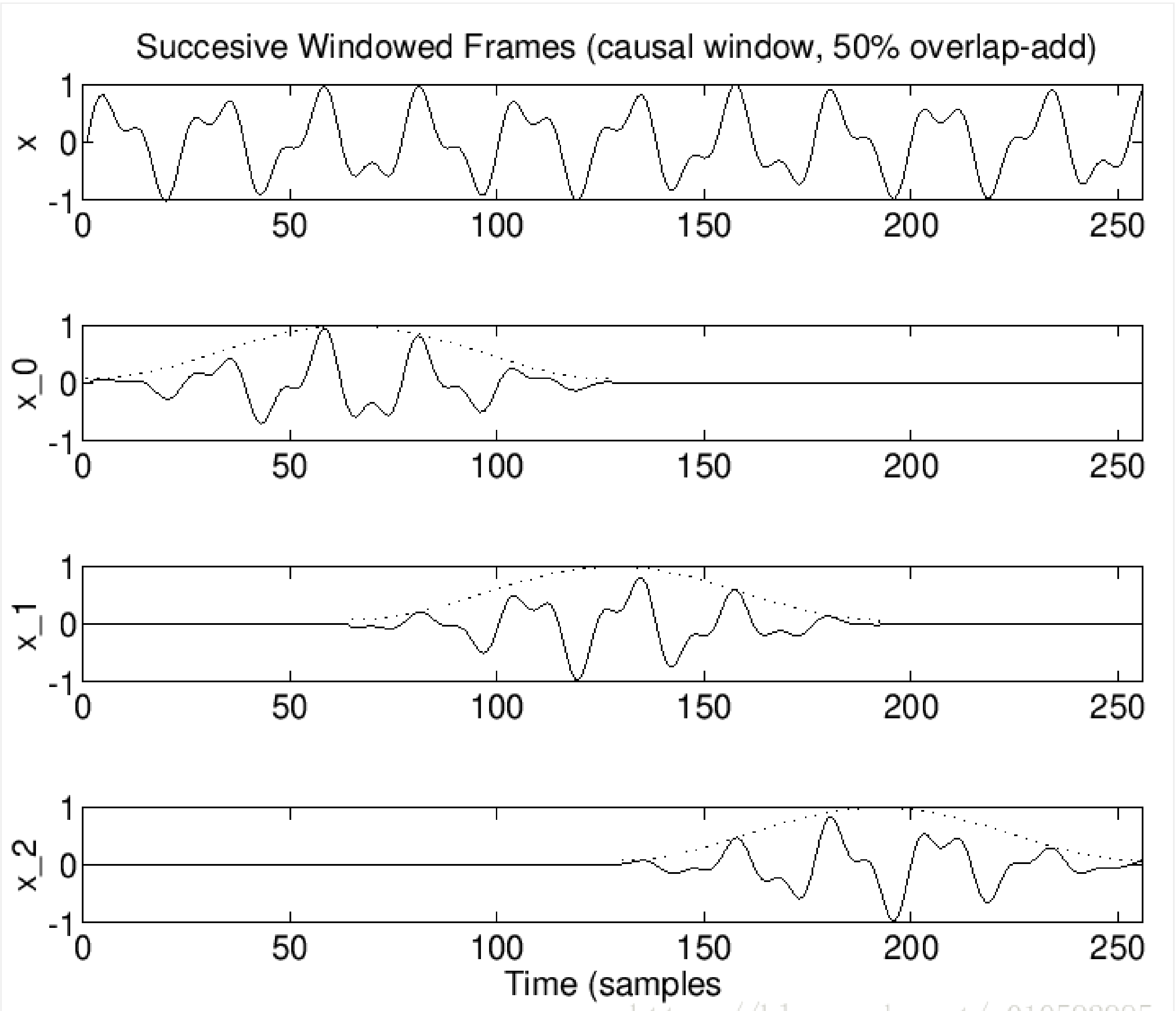


Figure 8.8: Input signal (top) and three successive windowed data frames.

Convolution with long signals

First give a section of frequency domain FIR filter program, using overlap-add

```
L = 31;           % FIR filter length in taps
fc = 600;         % lowpass cutoff frequency in Hz
fs = 4000;        % sampling rate in Hz

Nsig = 150;       % signal length in samples
period = round(L/3); % signal period in samples
%FFT processing parameters:
M = L;                  % nominal window length
Nfft = 2^(ceil(log2(M+L-1))); % FFT Length
M = Nfft-L+1            % efficient window length
```

TOP

```
R = M;                    % hop size for rectangular window
Nframes = 1+floor((Nsig-M)/R);  % no. complete frames
%Generate the impulse-train test signal:
sig = zeros(1,Nsig);
sig(1:period:Nsig) = ones(size(1:period:Nsig));
%Design the lowpass filter using the window method:
epsilon = .0001;     % avoids 0 / 0
nfilt = (-(L-1)/2:(L-1)/2) + epsilon;
hideal = sin(2*pi*fc*nfilt/fs) ./ (pi*nfilt);
w = hamming(L)'; % FIR filter design by window method
h = w' .* hideal; % window the ideal impulse response

hzp = [h zeros(1,Nfft-L)];  % zero-pad h to FFT size
H = fft(hzp);               % filter frequency response
%Carry out the overlap-add FFT processing:
y = zeros(1,Nsig + Nfft); % allocate output+'ringing' vector
for m = 0:(Nframes-1)
    index = m*R+1:min(m*R+M,Nsig); % indices for the mth frame
    xm = sig(index);  % windowed mth frame (rectangular window)
    xmzp = [xm zeros(1,Nfft-length(xm))]; % zero pad the signal
    Xm = fft(xmzp);
    Ym = Xm .* H;               % freq domain multiplication
    ym = real(ifft(Ym))         % inverse transform
    outindex = m*R+1:(m*R+Nfft);
    y(outindex) = y(outindex) + ym; % overlap add
end
```

At first glance, this block-processing method of overlap-add is very similar to STFT. Can the overlap-add method be used to reconstruct the STFT signal? Of course, it is possible. In the above example, this is done when the frequency domain filter is restored. In the example, the signal block does not appear to be windowed. The window is not added. In fact, the rectangular window is added. The analysis is as follows

Filter length L= 31

Window length M=34

FFT points Nfft=64

Linear convolution length L+M-1=64, which is exactly equal to the number of FFT points, so it will not cause time-domain aliasing

The step size R=34, equal to the window length, that is, there is no overlap between the blocks

Does the overlap-add signal in the above program reconstruct the signal? Verify that the impulse response is directly convolved with the original signal

```
y_conv=conv(sig,h);
```

TOP

By comparison, you can see that y_conv is the same as y (the last segment is temporarily ignored by the signal length)
In the above example, a rectangular window is used, and there is no overlap in the block. If you use other windows and there is an overlap, can it be restored?

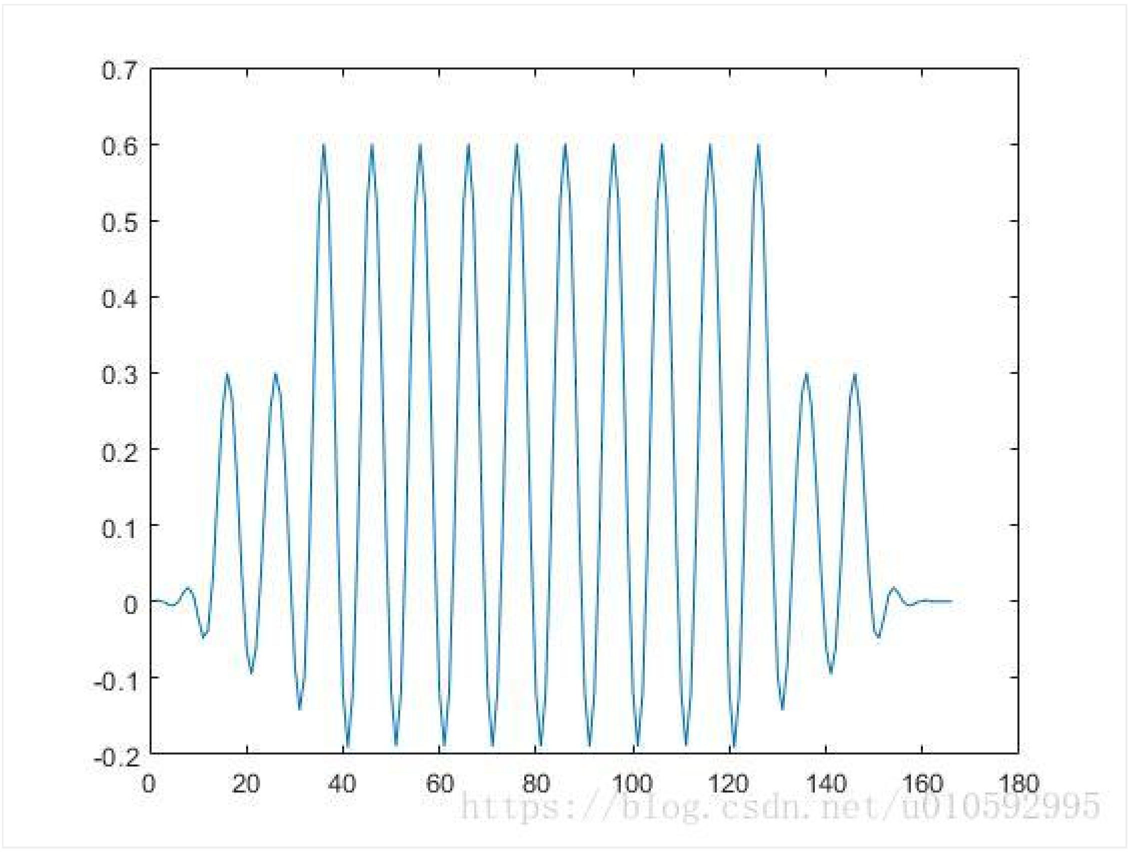
time-invariant filter

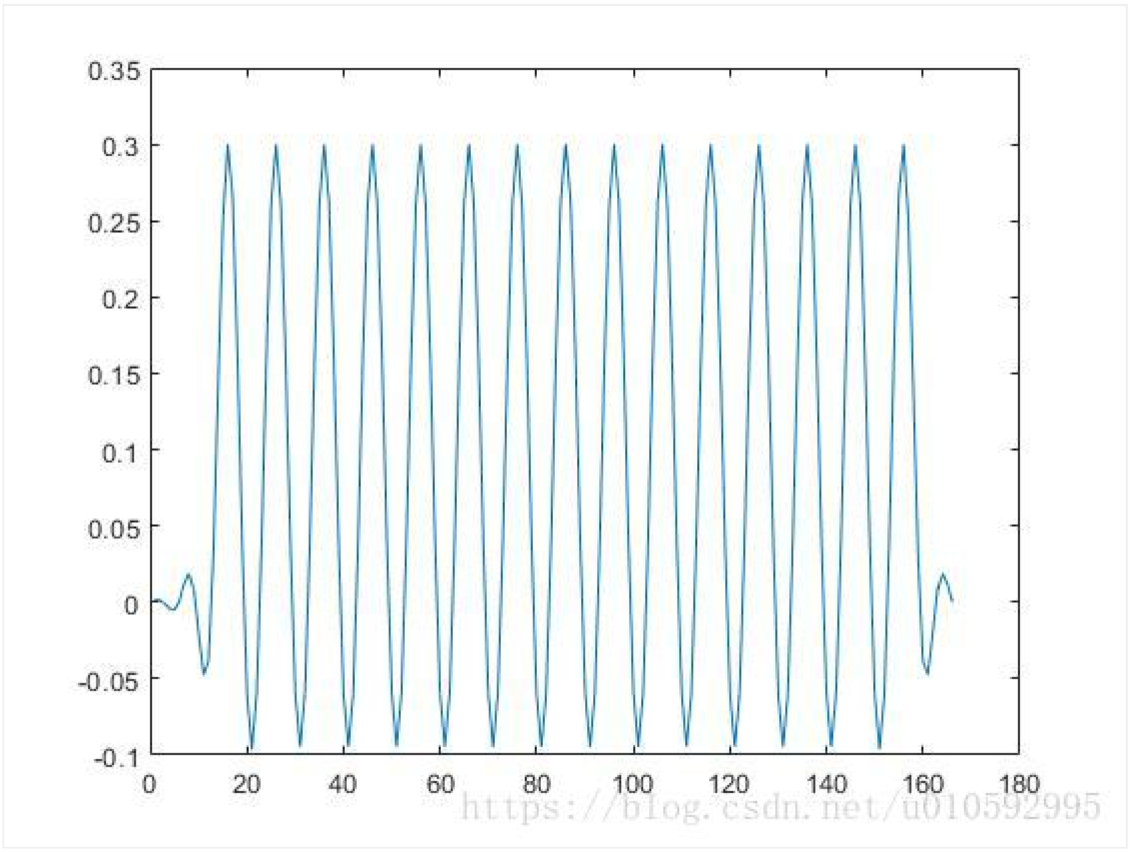
In the above example program, the filter coefficients are time-invariant. In this case, there is no need to use the idea of STFT. The rectangular window is used to block the signal without overlap.

time-variant filter

In many adaptive filtering applications, the filter coefficient will change according to the change of the statistical characteristics of the input signal, which is time-varying. At this time, in order to reduce the coefficient mutation caused by the discontinuity between the signal frame and the frame, generally in the sub-frame Sometimes it overlaps a part, which is also the step size in the STFT formula R R Less than window length M M At the same time, in order to improve the spectral resolution and reduce the spectral leakage, other windows such as hamming and hann windows will be selected. In this case, can the overlap-add recover?
Still the above example, set the step size to $\frac{1}{2}$ 1 2 Window length, compare $y\mathrm{versus}$ y versus What is the difference between y_conv

TOP


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0
20
40
60
80
100
120
140
160
180
https://blog.csdn.net/u010592995


0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
20
40
60
80
100
120
140
160
180
https://blog.csdn.net/u010592995

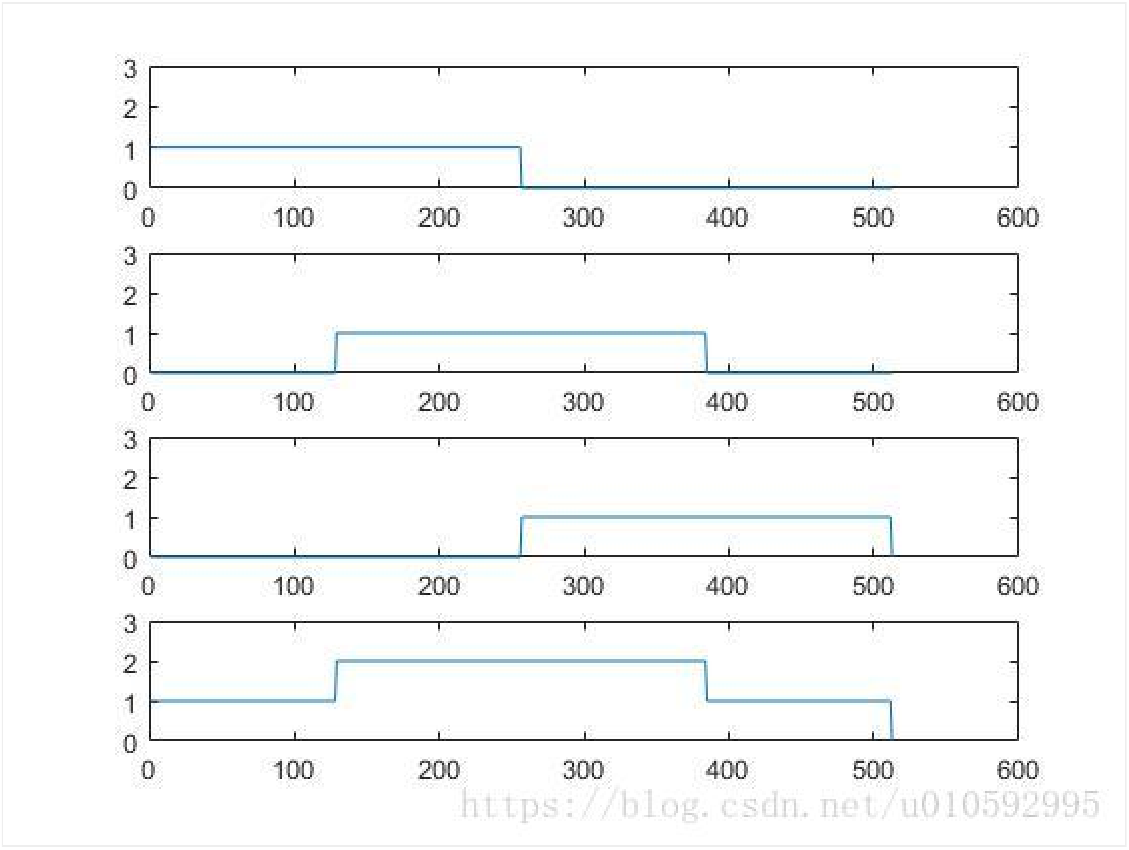
TOP

A simple observation can see that the shape of the data in the two pictures starts and ends differently. The middle is the same but there is a 2x relationship. This is definitely caused by the overlapping of the windows. First draw an overlapping rectangular window to analyze

```
M = 257;          % window length
w = rectwin(M);
R = (M-1)/2;     % maximum hop size
w(M) = 0;        % 'periodic Hamming' (for COLA)
%w(M) = w(M)/2; % another solution,
%w(1) = w(1)/2; %  interesting to compare

w1 = [w;zeros(M-1,1)];
w2 = [zeros(R,1);w;zeros(R,1)];
w3 = [zeros(M-1,1);w];

figure,subplot(4,1,1),plot(w1),ylim([0,3])
subplot(4,1,2),plot(w2),ylim([0,3])
subplot(4,1,3),plot(w3),ylim([0,3])
subplot(4,1,4),plot(w1+w2+w3),ylim([0,3])
```



As can be seen from the above figure, when the rectangular windows overlap by 50%, the shape changes, but the sum of the middle overlapping parts is still equal to a constant, which expla TOP

why the y and y_conv are the same in the above program results, and the middle difference is 2 Times
So, what conditions need to be met during the STFT reconstruction process?

COLA

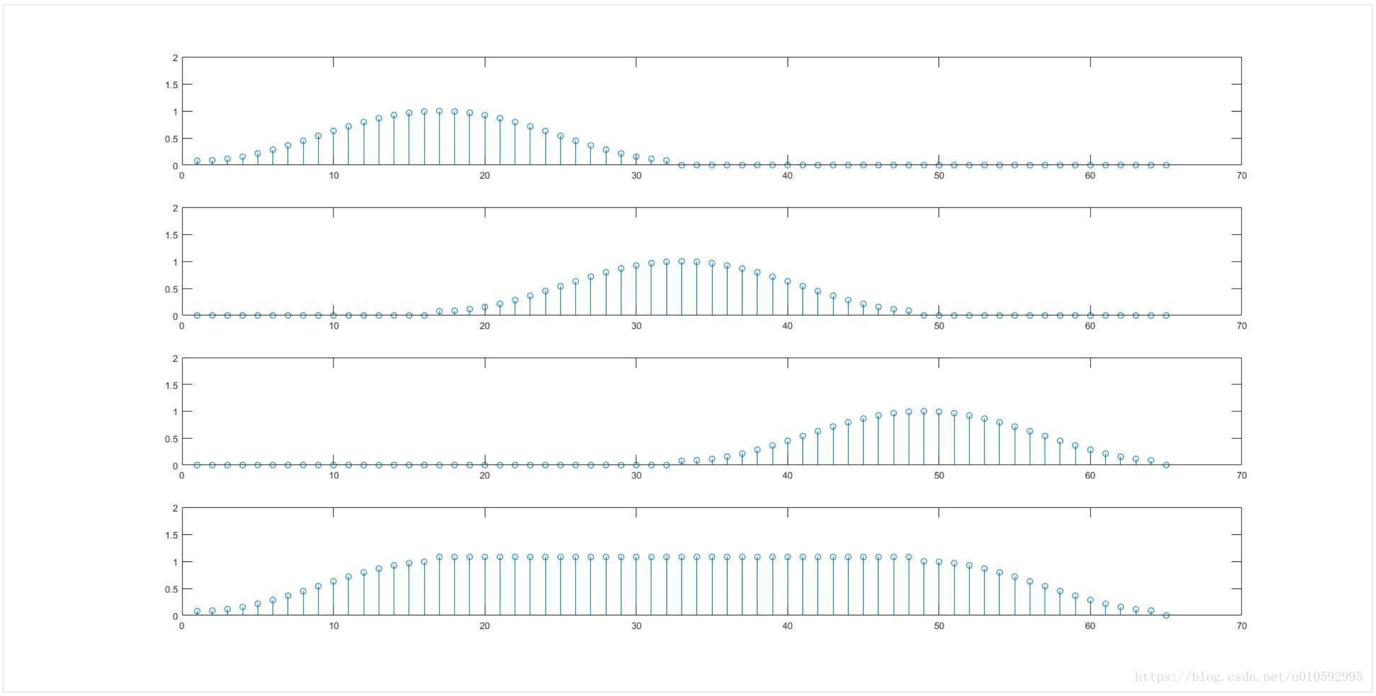
constant-overlap-add, the formula is explained as follows

$$
\begin{aligned}
x(n) &= \sum_{m=-\infty}^{\infty} x_m(n) \\
&= \sum_{m=-\infty}^{\infty} x(n)w(n-mR) \\
&= x(n) \sum_{m=-\infty}^{\infty} w(n-mR)
\end{aligned}
$$

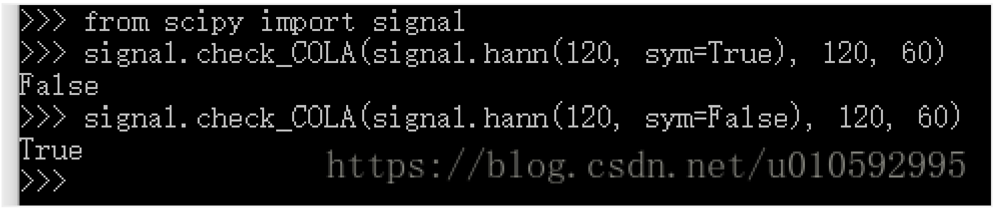
$$\sum_{m=-\infty}^{+\infty} w(n-mR) = 1$$ ∑ m = − ∞ + ∞ w (n − m R) = 1

Therefore, when the framed windowing meets the above conditions, using overlap-add can signal perfect-reconstruction, how to explain this condition, meaning that the window function gives the same weight to all points of the signal.
Then for the non-overlapping rectangular window in the above example, imagine that tiles were laid on a road, and then laid forward one by one, so that the entire road was flat after the paving. , So it is obvious that the non-overlapping rectangular window satisfies the condition of COLA
Many windows can meet this condition, draw a 50% overlapping hamming window

TOP



There are special functions in scipy to check whether the specified window meets the COLA condition



```
>>> from scipy import signal
>>> signal.check_COLA(signal.hann(120, sym=True), 120, 60)
False
>>> signal.check_COLA(signal.hann(120, sym=False), 120, 60)
True
>>>
```

In the introduction page of this function, you can see some examples of windows that satisfy the COLA condition

In order to enable inversion of an STFT via the inverse STFT in istft, the signal windowing must obey the constraint of "Constant OverLap Add" (COLA). This ensures that every point in the input data is equally weighted, thereby avoiding aliasing and allowing full reconstruction.
Some examples of windows that satisfy COLA:

- Rectangular window at overlap of 0, 1/2, 2/3, 3/4, …
- Bartlett window at overlap of 1/2, 3/4, 5/6, …
- Hann window at 1/2, 2/3, 3/4, …
- Any Blackman family window at 2/3 overlap

TOP