spectrogram

```
s, = spectrogram(x)
```

返回输入信号 x 的短时傅立叶变换（STFT）。s 的每列包含 x 的短期、时间局部化频率内容的估计。s 的幅度平方被称为 x[1] 的频谱图时间-频率表示。

```
s, = spectrogram(x, window)
```

使用 window 将信号划分为段并执行开窗。

```
s, = spectrogram(x, window, noverlap)
```

使用相邻线段之间重叠的 noverlap 样本。

```
s, = spectrogram(x, window, noverlap, nfft)
```

使用 nfft 采样点计算离散傅立叶变换。

```
s, w, t = spectrogram(___)
```

返回归一化频率向量 w 和计算 STFT 的时间时刻向量 t。该语法可以包含前面语法中输入参数的任意组合。

```
s, f, t = spectrogram(___, fs)
```

返回循环频率的矢量 f，用采样率 fs 表示。fs 必须是声谱图的第五个输入。要输入采样率并仍然使用前面可选参数的默认值，请将这些参数指定为空，[]。

```
s, w, t = spectrogram(x, window, noverlap, w)
```

以 w.w 中指定的归一化频率返回 STFT。w 必须至少有两个元素，否则函数会将其解释为 nfft。

```
s, f, t = spectrogram(x, window, noverlap, f, fs)
```

以 f 中指定的循环频率返回 STFT。f 必须至少有两个元素，否则函数会将其解释为 nfft。

```
_, _, _, ps = spectrogram(___)
```

还会返回一个与 x 的频谱图成正比的矩阵 ps。

- 如果指定频谱类型为”psd”，则 ps 的每一列都包含一个窗口分段的功率谱密度（PSD）估计值。
- 如果指定频谱类型为”power”，则 ps 的每一列都包含一个窗口分段的功率谱估计值。

```
_ = spectrogram(___, "reassigned")
```

将每个 PSD 或功率谱估算值重新分配到其能量中心的位置。如果您的信号包含定位良好的时间或频谱成分，那么该选项生成的频谱图会更清晰。

```
_, _, _, ps, fc, tc = spectrogram(___)
```

还会返回两个矩阵 fc 和 tc，其中包含每个 PSD 或功率谱估计值的能量中心频率和时间。

```
___ = spectrogram(___, freqrange)
```

返回 freqrange 指定频率范围内的 PSD 或功率谱估计值。freqrange 的有效选项为”onesided”、”twosided” 和”centered”。

```
___ = spectrogram(___, Name,Value)
```

使用名称值参数指定其他选项。选项包括最小阈值和输出时间维度。

```
___ = spectrogram(___, spectrumtype)
```

如果频谱类型指定为”psd”,则返回短期交叉功率谱密度估计值;如果频谱类型指定为”power”,则返回短期交叉功率谱估计值。

```
spectrogram(___)
```

在没有输出参数的情况下，在当前图形窗口中绘制交叉谱图。

```
spectrogram(___, freqloc)
```

指定绘制频率的轴。将 freqloc 指定为”xaxis” 或”yaxis”。

```
# 频谱图的默认值

using TySignalProcessing
using TyPlot
using TyMath
```

```
N = 1024
n = 0:(N - 1)

w0 = 2 * pi / 5
x = @. sin(w0 * n) + 10 * sin(2 * w0 * n);
s, = spectrogram(x)

figure()
spectrogram(x, "yaxis"; plotfig=true)
```

Listing 1: 频谱图的默认值

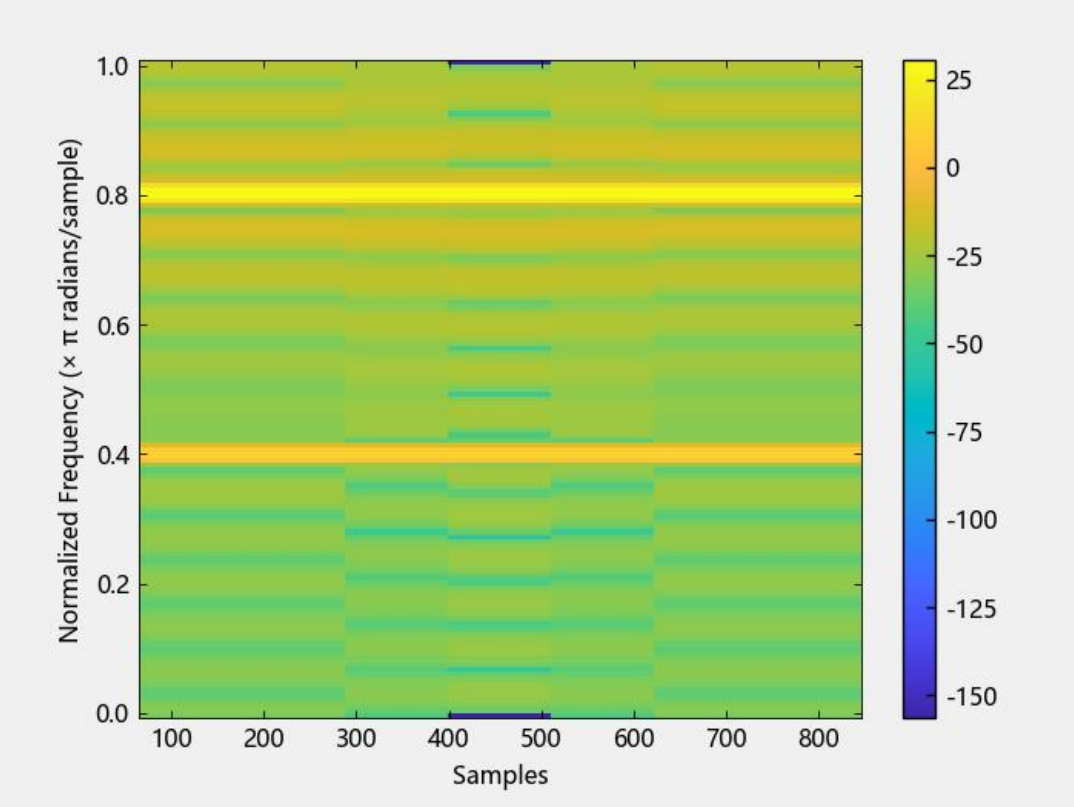

Figure 1: Listing1 结果图

```
# 沿 x轴 的 频 率
```

```
2
3  using TySignalProcessing
4  using TyPlot
5  t = 0:0.001:2
6  x = chirp(t, 100, 1, 200, "quadratic")
7
8  # 将信号划分为长度为 128 的段落，并使用汉明窗口。
9  figure()
10 spectrogram(x, 128, 120, 128, 1e3; plotfig=true)
11
12 # 用 Blackman 窗口取代汉明窗口。将重叠减少到 60 个样本。绘制时间
       轴，使其数值从上到下递增
13 figure()
14 spectrogram(x, blackman(128), 60, 128, 1e3; plotfig=true)
15 gca().set_ylim(reverse(ylim()))
```

Listing 2: 沿 x 轴的频率

```
1  # 线性调频的功率谱密度
2
3  using TySignalProcessing
4  using TyPlot
5  fs = 1000
6  t = 0:(1 / fs):2
7  x = chirp(t, 100, 1, 200, "quadratic")
8
9  # 二次线性调频
10 figure()
11 spectrogram(x, 128, 120, 128, fs, "yaxis"; plotfig=true)
12 title("Quadratic Chirp")
13
14 # 线性线性调频信号
15 x = chirp(t, 0, 1, 150)
16
17 figure();
```

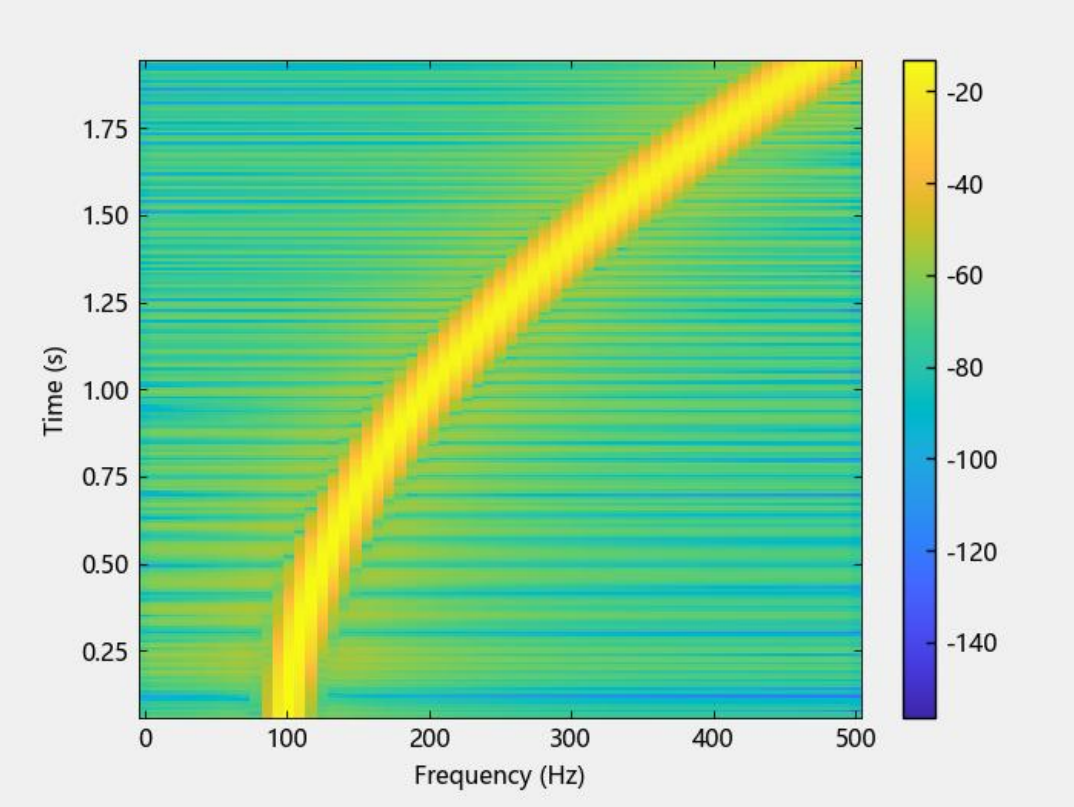

Figure 2: Listing2 结果图 1

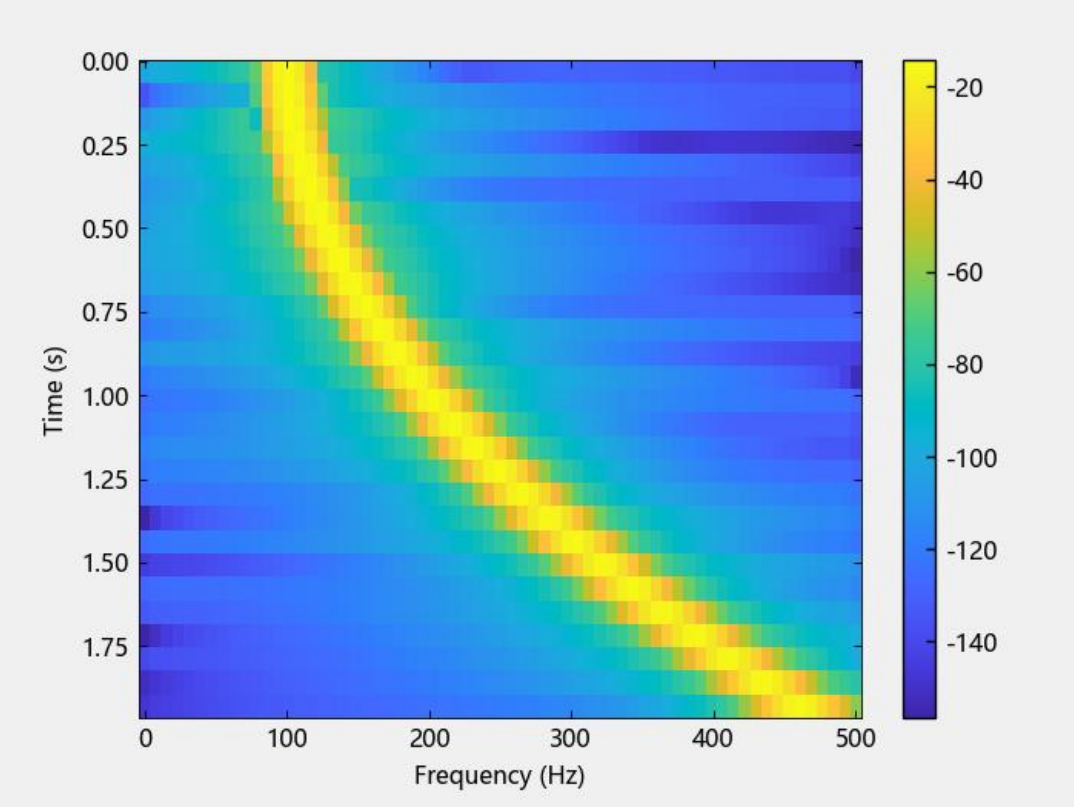

Figure 3: Listing2 结果图 2

```
spectrogram(x, 256, 250, 256, fs, "yaxis"; plotfig=true);
title("Linear Chirp")

# 对数线性调频
x = chirp(t, 20, 1, 60, "logarithmic")

figure();
spectrogram(x, 256, 250, [], fs, "yaxis"; plotfig=true);
title("Logarithmic Chirp")
gca().set_yscale("log")
gca().set_ylim([4, 500])
gca().set_yscale("linear")
```

Listing 3: 线性调频的功率谱密度

```
# 频谱图和瞬时频率
using TySignalProcessing
using TyPlot
fs = 1000
t = 0:(1 / fs):(2 - 1 / fs)
y = chirp(t, 100, 1, 200, "quadratic")

figure()
spectrogram(y, 100, 80, 100, fs, "yaxis"; plotfig=true)
```

Listing 4: 频谱图和瞬时频率

```
# 复信号频谱图

using TySignalProcessing
using TyPlot
N = 512
n = 0:(N - 1)

x = @. exp(1im * pi * sin(8 * n / N) * 32)
```

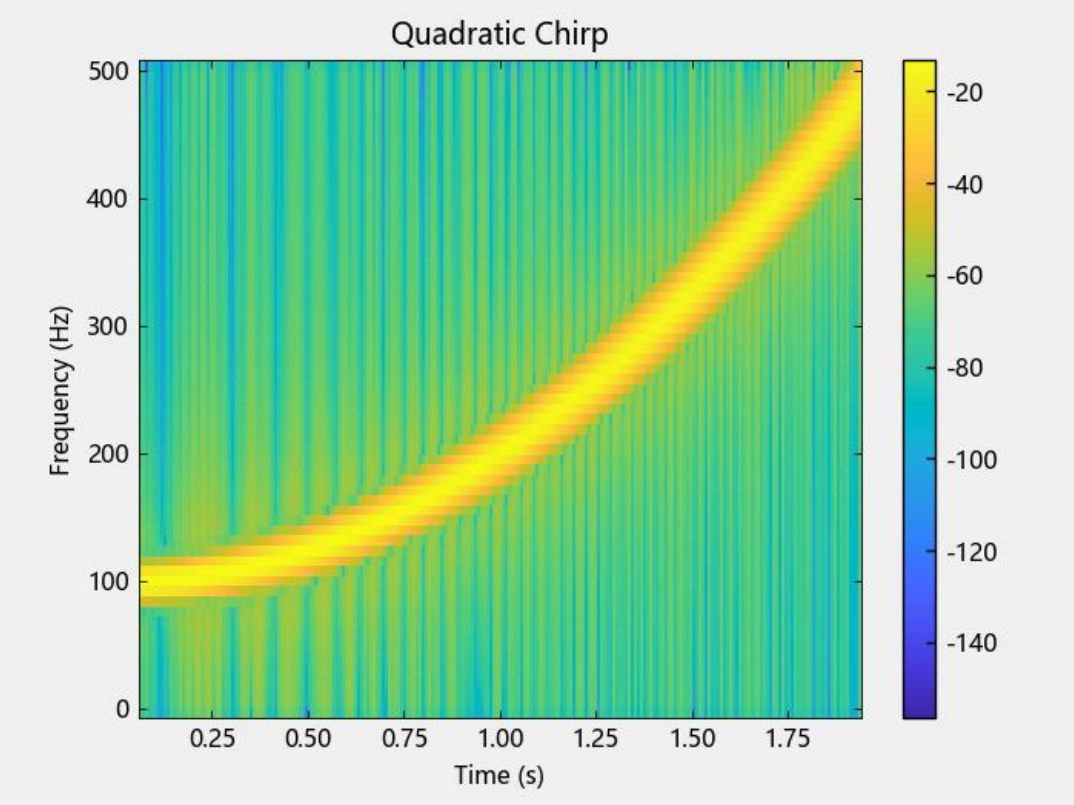

Figure 4: Listing3 结果图 1

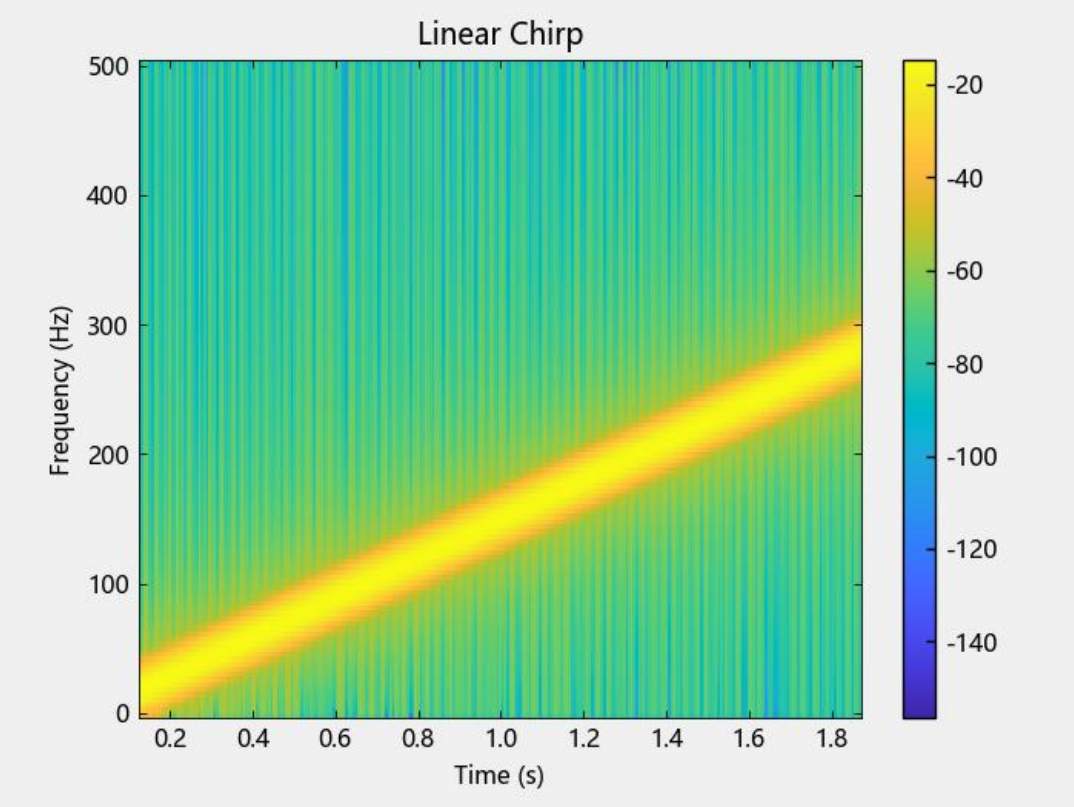

Figure 5: Listing3 结果图 2

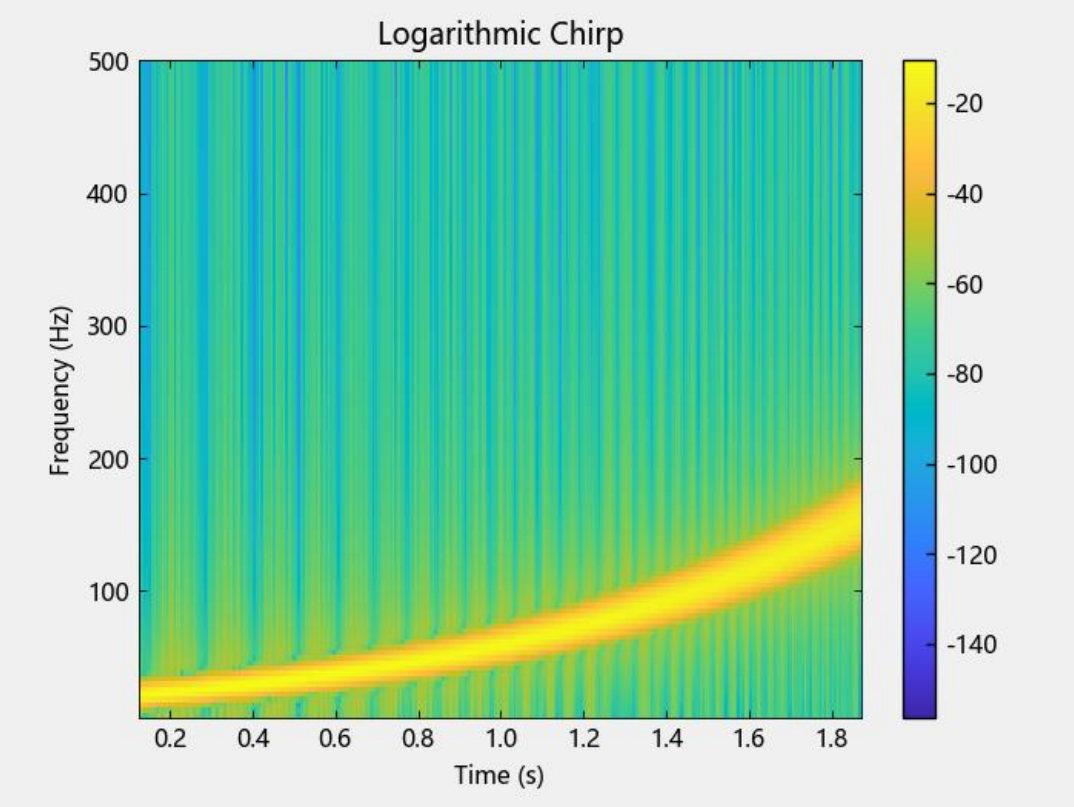

Figure 6: Listing3 结果图 3

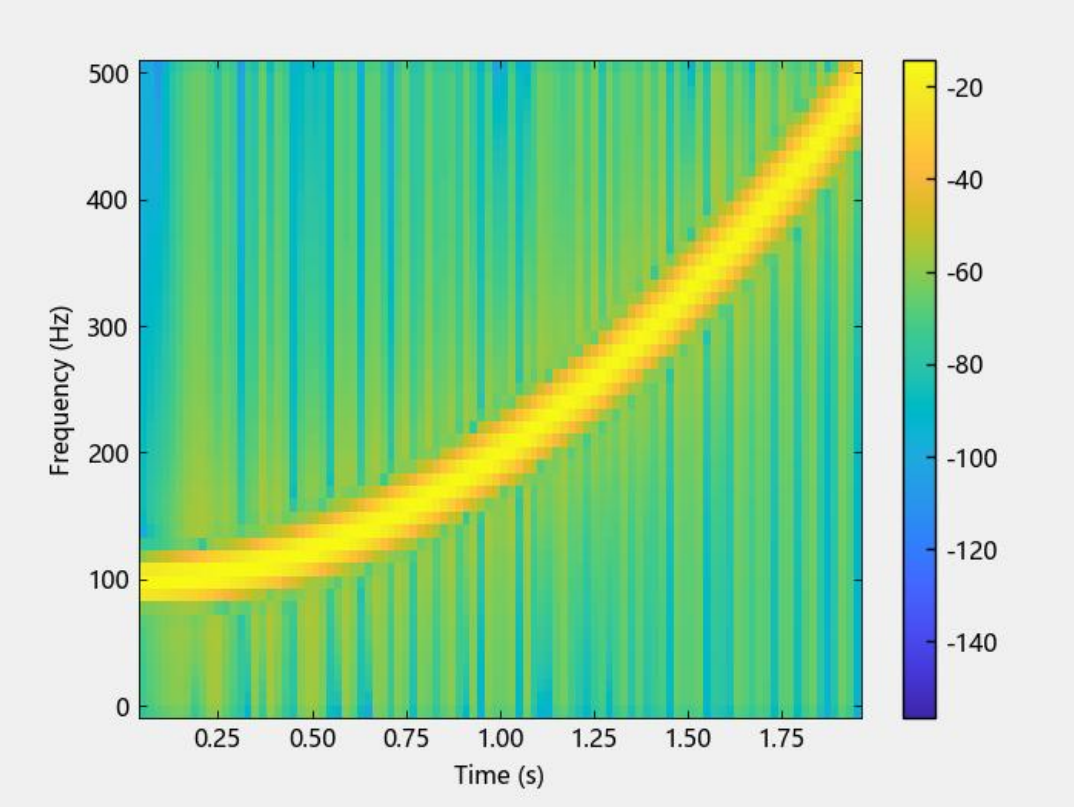

Figure 7: Listing4 结果图

```

scalar, fs, ts = spectrogram(x, 32, 16, 64, "centered")

figure()
spectrogram(x, 32, 16, 64, "centered", "yaxis"; plotfig=true)

# fintv = (-pi + pi / 32):(pi / 32):pi

# vector, fv, tv = spectrogram(x, 32, 16, fintv)

# figure()
# spectrogram(x, 32, 16, fintv, "yaxis"; plotfig=true)
```

Listing 5: 复信号频谱图

```
# 二次线性调频的重新分配频谱图

using TySignalProcessing
using TyPlot
Fs = 1000
t = 0:(1 / Fs):2
y = chirp(t, 100, 1, 200, "quadratic")

figure()
spectrogram(y, kaiser(128, 18), 120, 128, Fs, "reassigned", "yaxis"
    ; plotfig=true)
```

Listing 6: 二次线性调频的重新分配频谱图

```
# 带阈值的频谱图

using TySignalProcessing
using TyPlot
Fs = 1000
t = 0:(1 / Fs):2
```

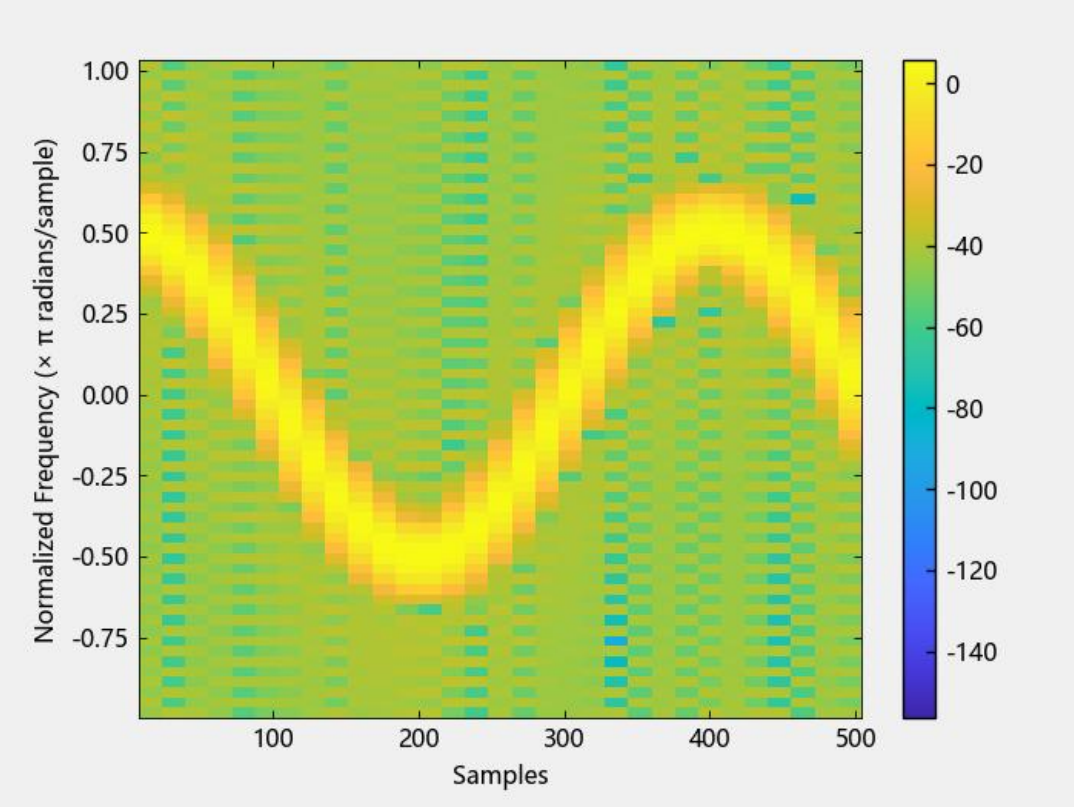

Figure 8: Listing5 结果图

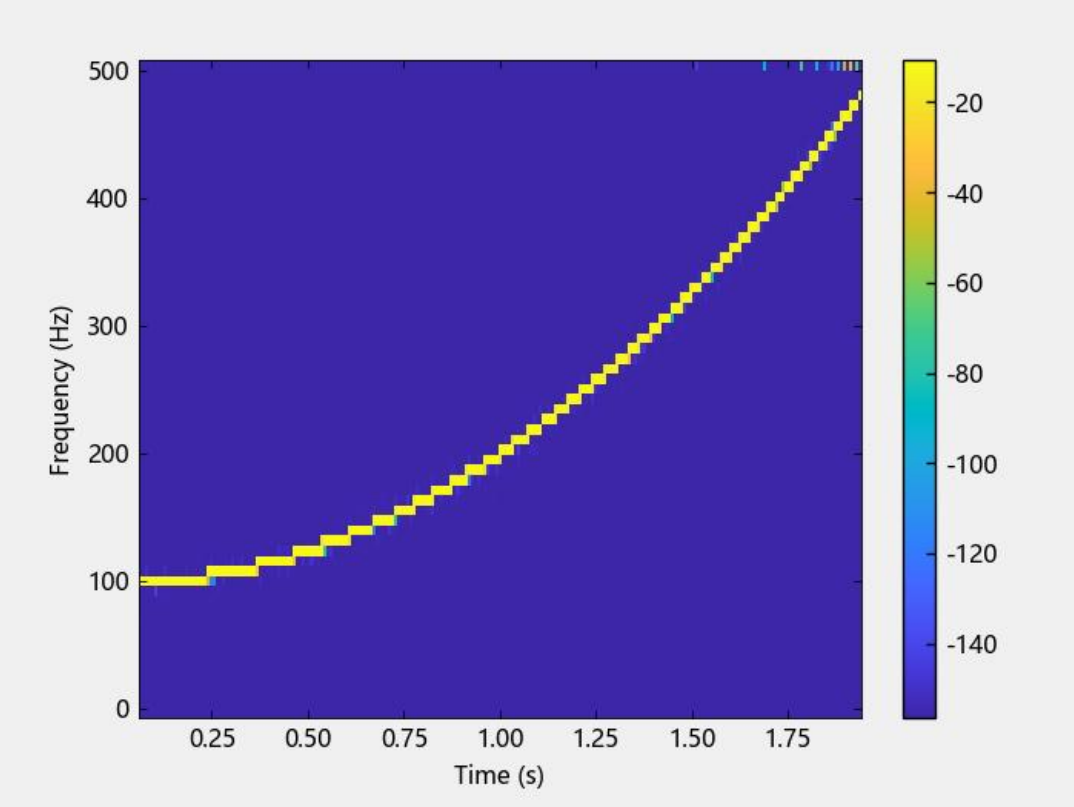

Figure 9: Listing6 结果图

```
y = chirp(t, 100, 1, 200, "quadratic")
_, _, _, pxx, fc, tc = spectrogram(y, kaiser(128, 18), 120, 128, Fs
    , "MinThreshold", -30)
plot(tc[pxx .> 0], fc[pxx .> 0], ".")
```

Listing 7: 带阈值的频谱图

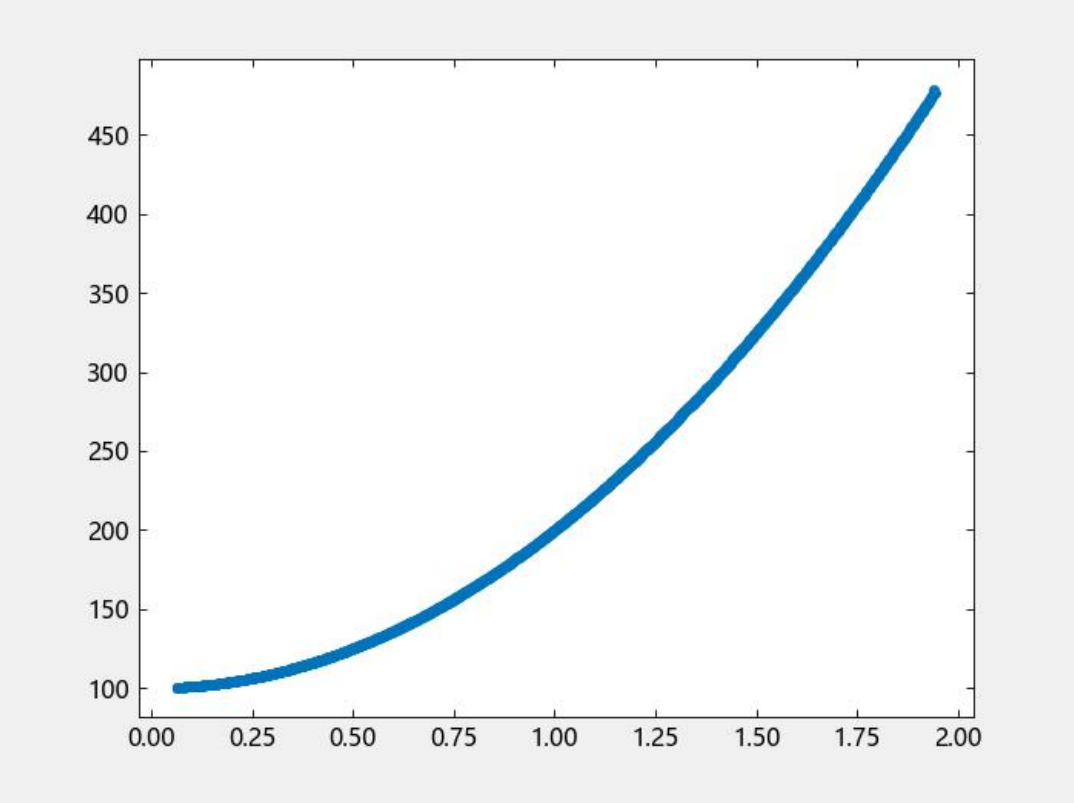

Figure 10: Listing7 结果图

```
# 频谱图重新分配和阈值处理

using TySignalProcessing
using TyPlot
using TyMath
using TyControlSystems
```

```
7
8  nSamp = 2048
9  Fs = 1024
10 t = (0:(nSamp - 1)) / Fs
11
12 t1 = t[1:Int(nSamp / 2)]
13
14 x11 = @. sin(2 * pi * 400 * t1)
15 x12 = chirp(t1 .- t1[Int(nSamp / 4)], 150, nSamp / Fs, 1750, "
       quadratic")
16 x1 = x11 + x12
17
18 t2 = t[Int(nSamp / 2 + 1):nSamp]
19
20 x21 = chirp(t2, 400, nSamp / Fs, 100)
21 x22 = chirp(t2, 550, nSamp / Fs, 250)
22 x2 = x21 + x22
23
24 SNR = 20
25 rng = MT19937ar(1234)
26
27 sig = [x1; x2]
28
29 sig = sig + randn(rng, size(sig)) * std(sig) / db2mag(SNR)
30
31 nwin = 63
32 wind = kaiser(nwin, 17)
33 nlap = nwin - 10
34 nfft = 256
35 # 计算并绘制信号频谱图
36 figure()
37 spectrogram(sig, wind, nlap, nfft, Fs, "yaxis"; plotfig=true)
38
39 # 对频谱图进行阈值处理，将数值小于信噪比的元素置零
```

```
figure()
spectrogram(sig, wind, nlap, nfft, Fs, "MinThreshold", -SNR, "yaxis"; plotfig=true)

# 将每个 PSD 估计值重新分配到其能量中心位置。
figure()
spectrogram(sig, wind, nlap, nfft, Fs, "reassigned", "yaxis"; plotfig=true)

# 对重新分配的频谱图进行阈值处理，将值小于信噪比的元素设置为零。
figure()
spectrogram(
    sig, wind, nlap, nfft, Fs, "reassigned", "MinThreshold", -SNR, "yaxis"; plotfig=true
)
```

Listing 8: 频谱图重新分配和阈值处理

```
# 三维频谱图可视化

using TySignalProcessing
using TyPlot
fs = 10e3
t = 0:(1 / fs):2
x1 = vco(sawtooth(2 * pi * t, 0.5), [0.1 0.4] * fs, fs)

figure()
spectrogram(x1, kaiser(256, 5), 220, 512, fs, "yaxis"; plotfig=true)
```

Listing 9: 三维频谱图可视化

xspectrogram

```
s, = xspectrogram(x, y)
```

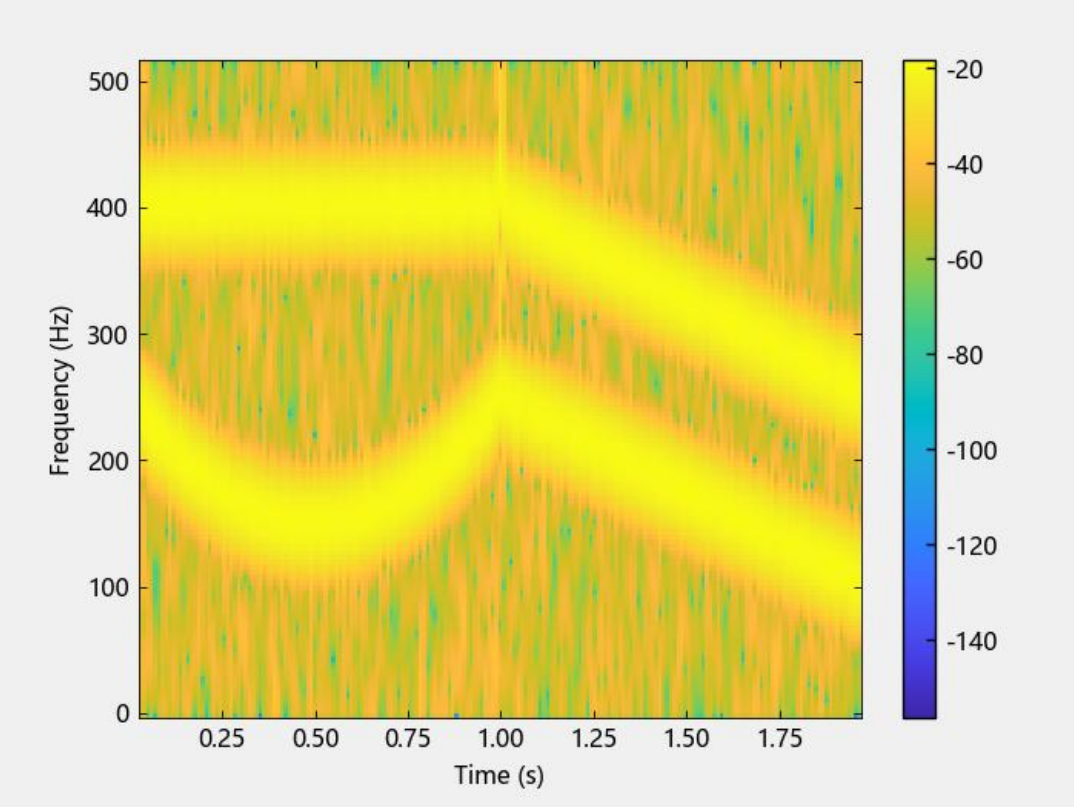

Figure 11: Listing8 结果图 1

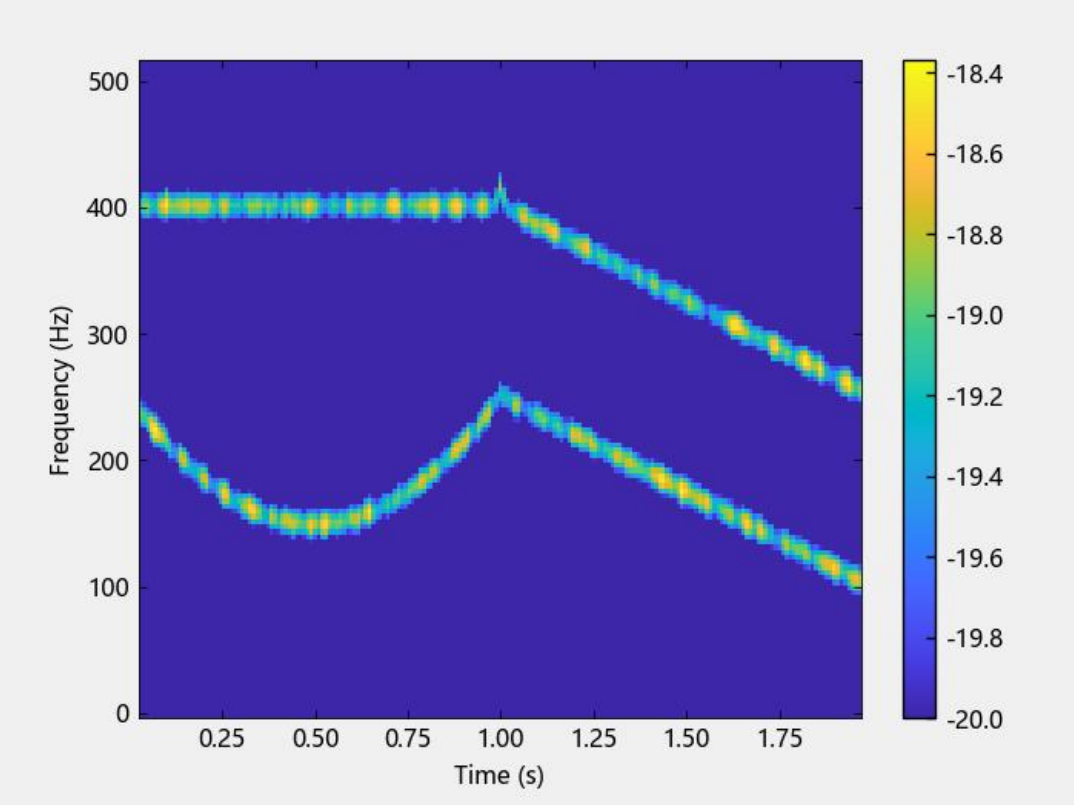

Figure 12: Listing8 结果图 2

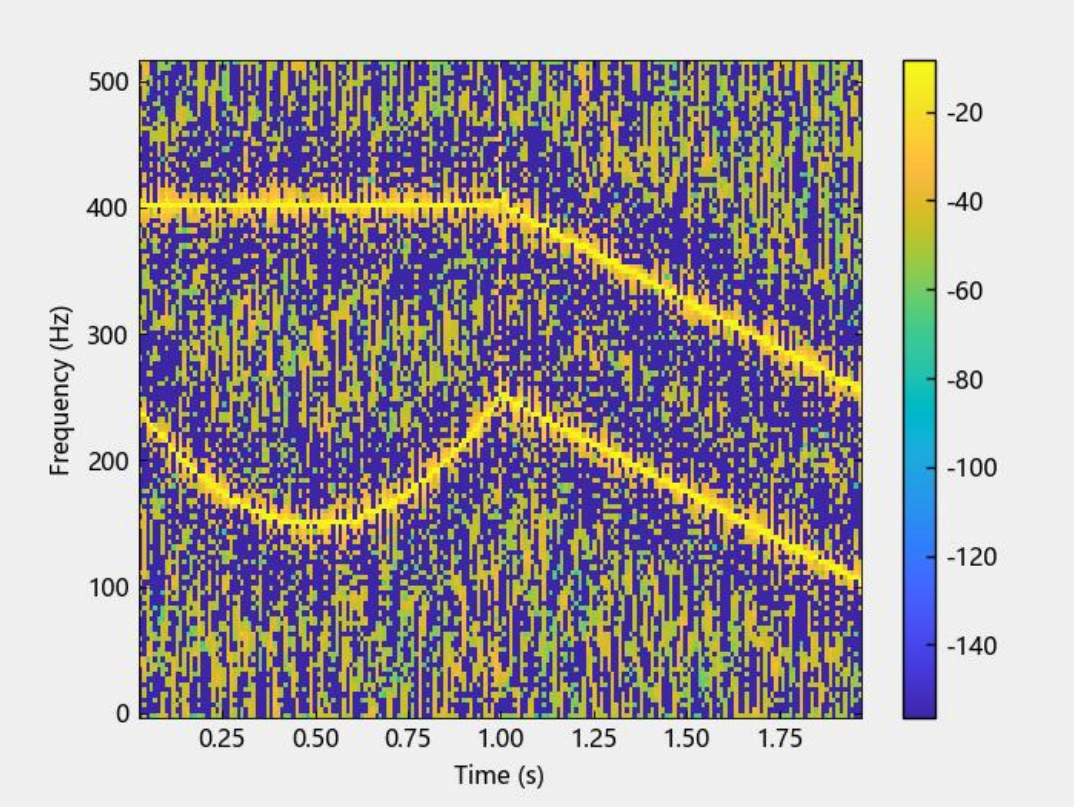

Figure 13: Listing8 结果图 3

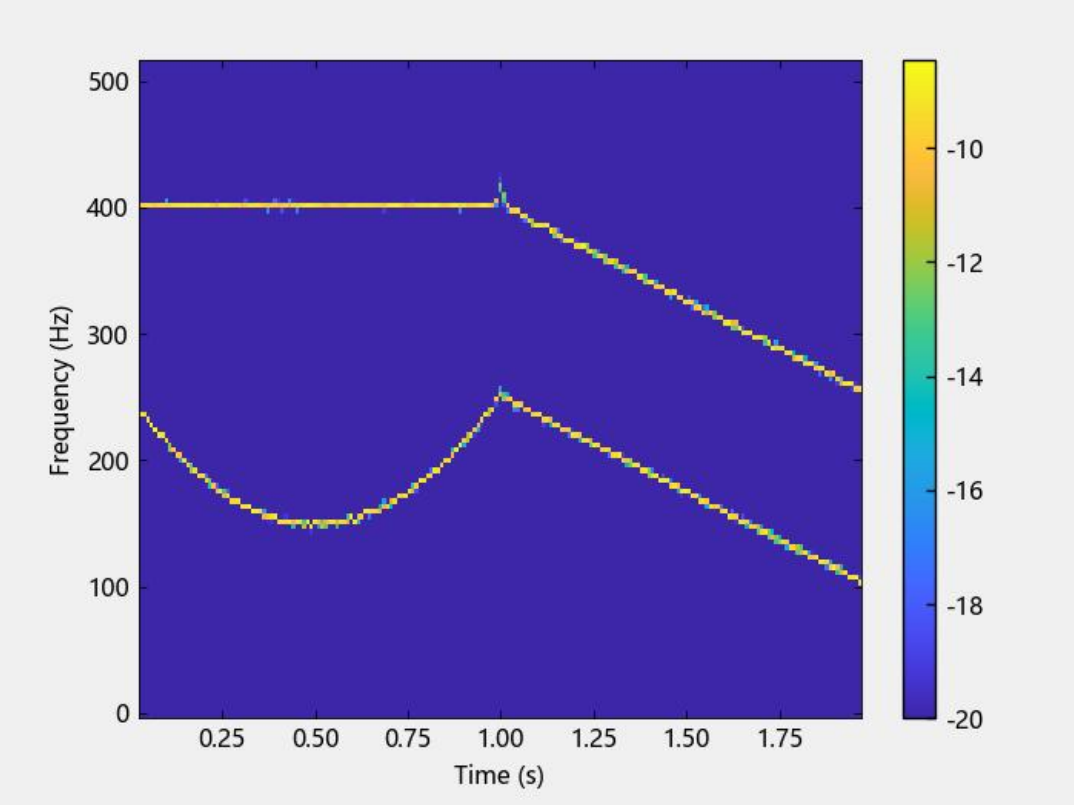

Figure 14: Listing8 结果图 4

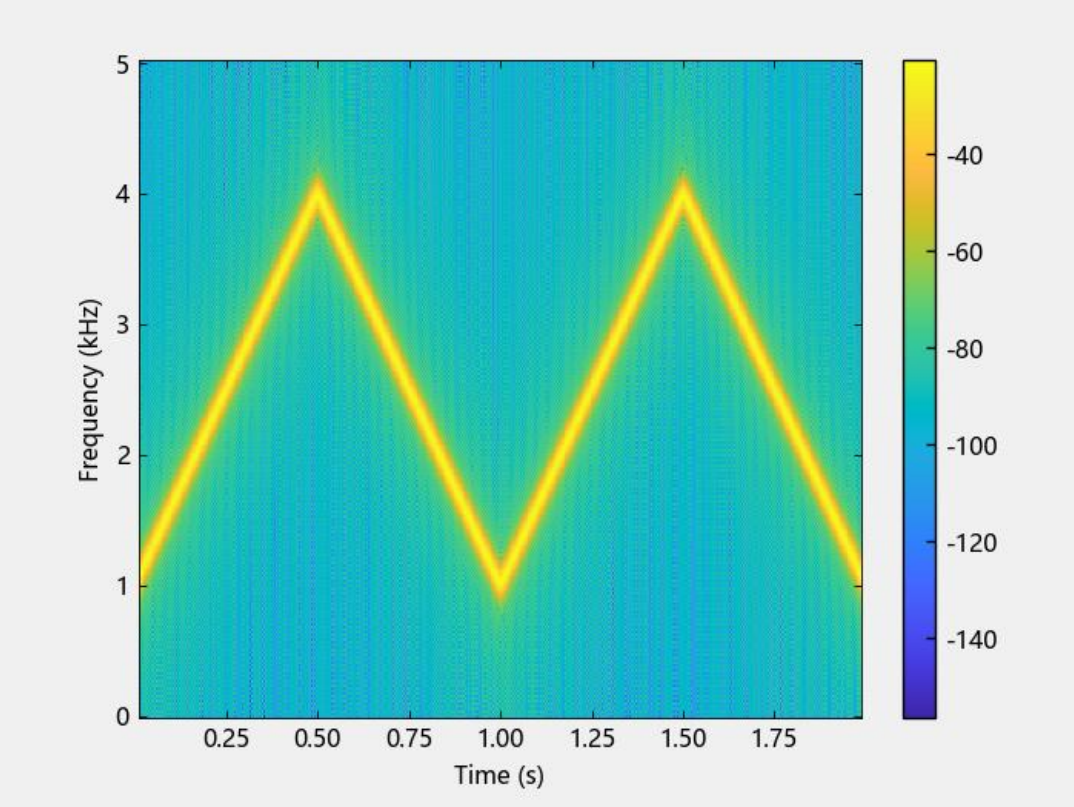

Figure 15: Listing9 结果图

返回 x 和 y 指定的信号的交叉频谱图。输入信号必须是元素数量相同的矢量。s 的每一列都包含 x 和 y 共同的短期、时间局部化频率内容的估计。

```
s, = xspectrogram(x, y, window)
```

使用 window 将 x 和 y 划分为多个分段并执行开窗操作。

```
s, = xspectrogram(x, y, window, noverlap)
```

使用相邻线段之间重叠的 noverlap 样本。

```
s, = xspectrogram(x, y, window, noverlap, nfft)
```

使用 nfft 采样点来计算离散傅立叶变换。

```
s, w, t = xspectrogram(___)
```

返回归一化频率的矢量 w 和计算交叉频谱图的时刻的矢量 t。此语法可以包括以前语法中输入参数的任何组合。

```
s, f, t = xspectrogram(___, fs)
```

返回频率向量 f，用采样率 fs 表示。fs 必须是 xsspectrogram 的第六个输入。要输入采样率并仍然使用前面可选参数的默认值，请将这些参数指定为空。

```
s, w, t = xspectrogram(x, y, window, noverlap, w)
```

返回 w 中指定的归一化频率下的交叉频谱图。

```
s, f, t = xspectrogram(x, y, window, noverlap, f, fs)
```

返回 f 中指定频率的交叉频谱图。

```
___, c = xspectrogram(___)
```

还返回一个矩阵 c，该矩阵包含输入信号的时变复互谱的估计。交叉谱图 s 是 c 的大小。

```
___ = xspectrogram(___, freqrange)
```

返回 freqrange 指定频率范围内的交叉谱图。freqrange 的有效选项为”onesided”、”twosided”和”centered”。

```
___ = xspectrogram(___, Name, Value)
```

使用名称-值对参数指定其他选项。选项包括最小阈值和输出时间维度。

```
___ = xspectrogram(___, spectrumtype)
```

如果频谱类型指定为”psd”,则返回短期交叉功率谱密度估计值;如果频谱类型指定为”power”,则返回短期交叉功率谱估计值。

```
xspectrogram(___)
```

在没有输出参数的情况下，在当前图形窗口中绘制交叉频谱图。

```
xspectrogram(___, freqloc)
```

指定绘制频率的轴。将 freqloc 指定为”xaxis” 或”yaxis”。

```
# 线性线性调频交叉谱图
using TyMath
using TySignalProcessing
using TyPlot
using TyControlSystems
nSamp = 10000
Fs = 1000e3
SNR = 40
t = [0:(nSamp - 1);] ./ Fs

rng = MT19937ar(1234)
x1 = chirp(t, 150e3, t[end], 350e3)
x1 = x1 + randn(rng, size(x1)) * std(x1) / db2mag(SNR)

x2 = chirp(t, 200e3, t[end], 300e3)
x2 = x2 + randn(rng, size(x2)) * std(x2) / db2mag(SNR)

figure()
xspectrogram(x1, x2, hamming(200), 80, 1024, Fs, "yaxis"; plotfig=
    true)

```

```
x2 = chirp(t, 50e3, t[end], 350e3)
x2 = x2 + randn(rng, size(x2)) * std(x2) / db2mag(SNR)

figure()
xspectrogram(x1, x2, kaiser(500, 5), 450, 256, Fs, "yaxis"; plotfig
    =true)
```

Listing 10: 线性线性调频交叉频谱图

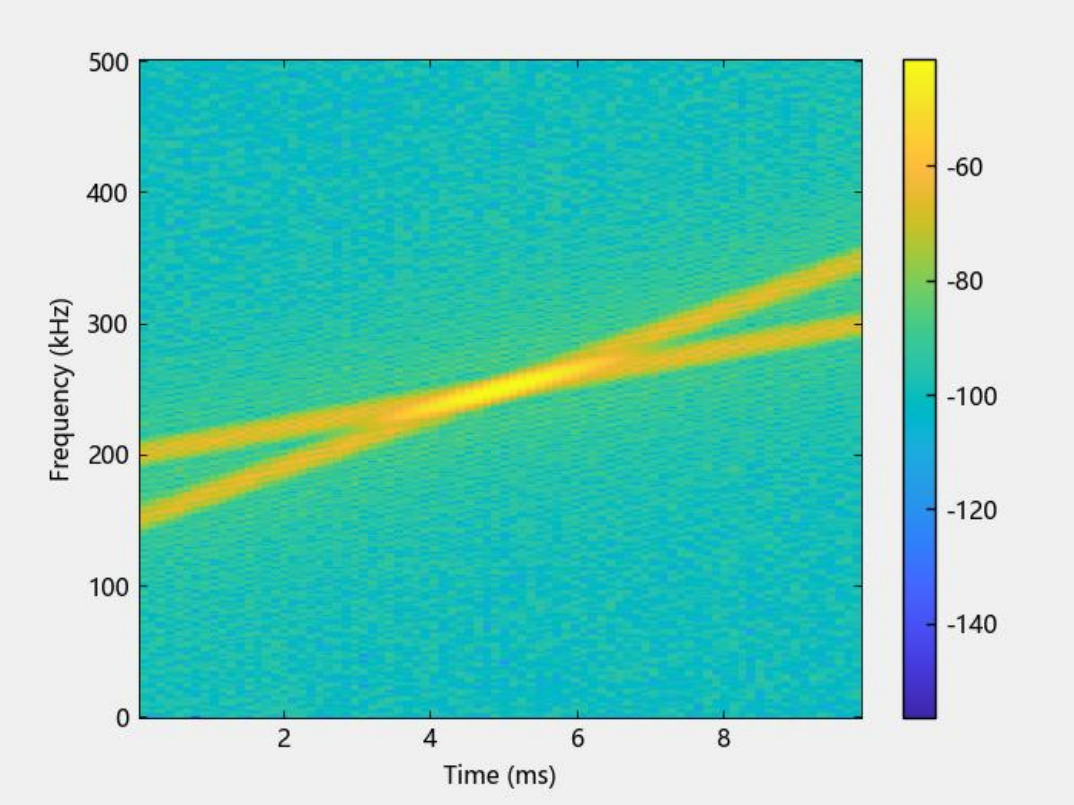

Figure 16: Listing10 结果图 1

```
# 两个二次线性调频之间的相移

using TySignalProcessing
using TyPlot
```

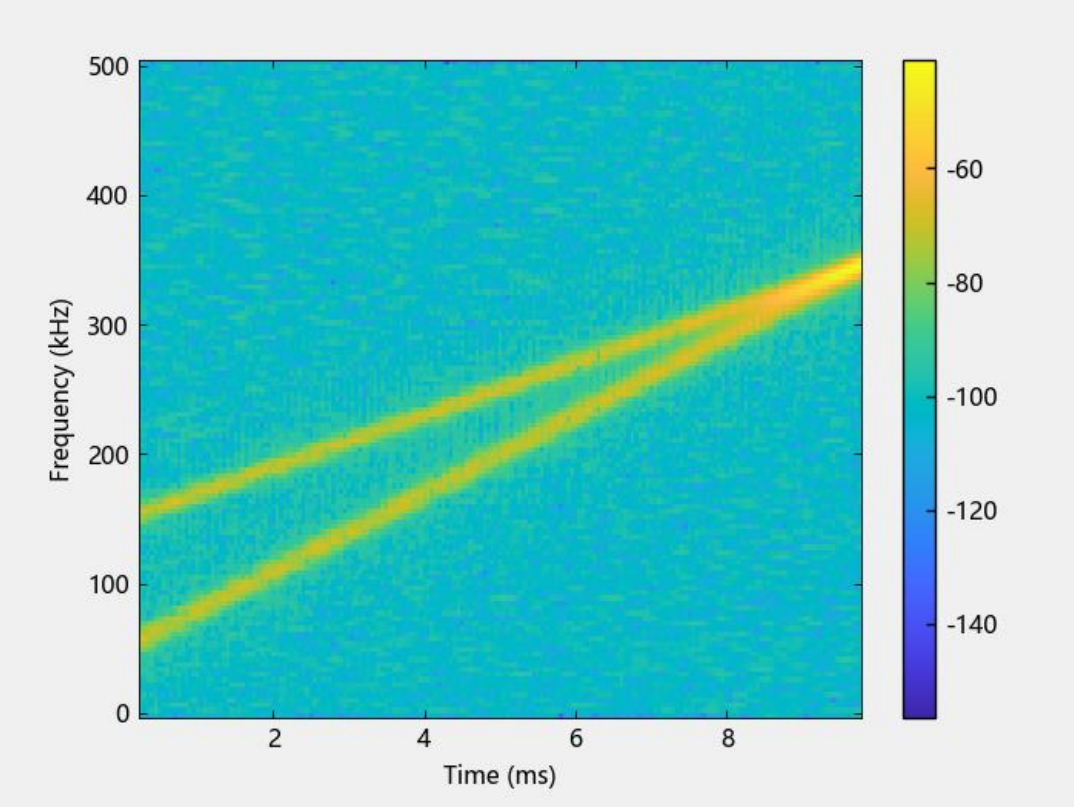

Figure 17: Listing10 结果图 2

```
fs = 1e3
t = 0:(1 / fs):2

y1 = chirp(t, 100, 1, 200, "quadratic", 0)
y2 = chirp(t, 100, 1, 200, "quadratic", 23)

_, f, xt, c = xspectrogram(y1, y2, kaiser(128, 18), 120, 128, fs)

figure()
xspectrogram(y1, y2, kaiser(128, 18), 120, 128, fs, "yaxis";
   plotfig=true)
```

Listing 11: 两个二次线性调频之间的相移

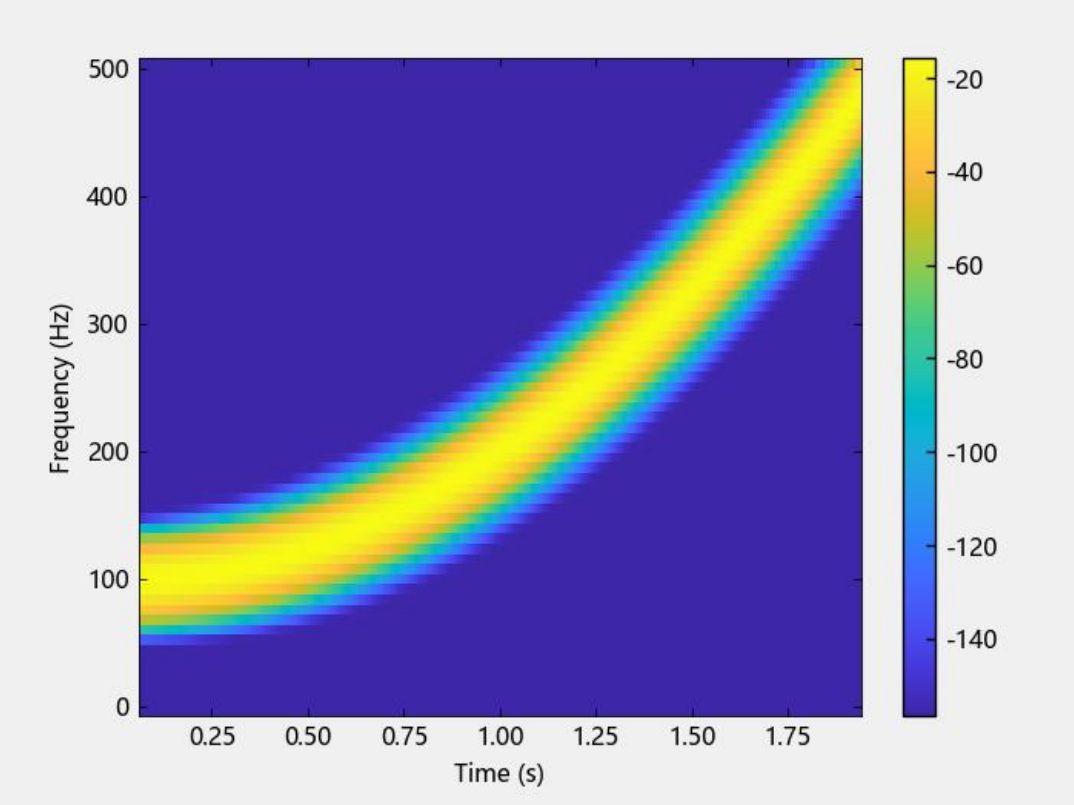

Figure 18: Listing11 结果图

```
# 复信号交叉谱图

using TyMath
using TySignalProcessing
using TyPlot
fs = 3000
t = 0:(1 / fs):(1 - 1 / fs)
rng = MT19937ar(1234)
x1 = chirp(t, 300, t[end], 1300, "quadratic") + randn(rng, size(t))
    / 100

x2 = exp.(2im * pi * 100 * cos.(2 * pi * 2 * t)) + randn(rng, size(
   t)) / 100

nwin = 256

figure()
xspectrogram(x1, x2, kaiser(nwin, 30), nwin - 1, [], fs, "centered"
   , "yaxis"; plotfig=true)

figure()
xspectrogram(
    x1,
    x2,
    kaiser(nwin, 30),
    nwin - 1,
    [],
    fs,
    "power",
    "MinThreshold",
    -40,
    "yaxis";
    plotfig=true,
```

```
)
title("Cross-Spectrogram of Quadratic Chirp and Complex Chirp")
```

Listing 12: 复信号交叉谱图

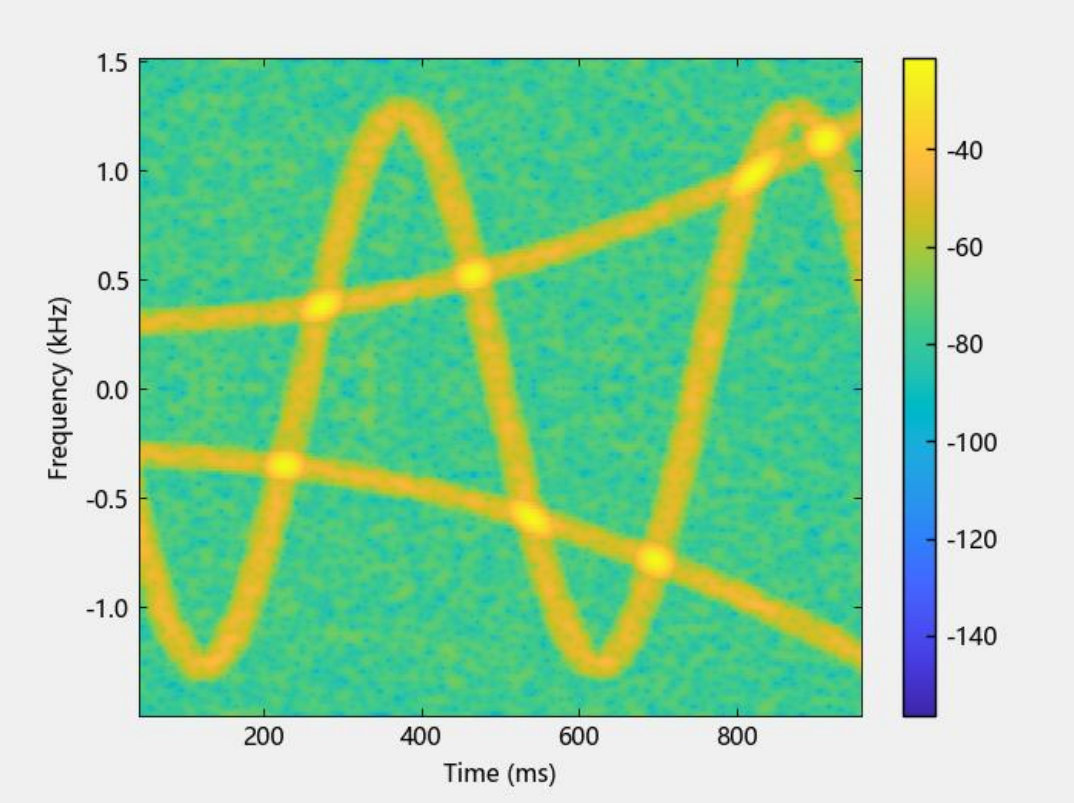

Figure 19: Listing12 结果图 1

```
# 两个序列的交叉谱图

using TyMath
using TySignalProcessing
using TyPlot
N = 4096

rng = MT19937ar(1234)
```

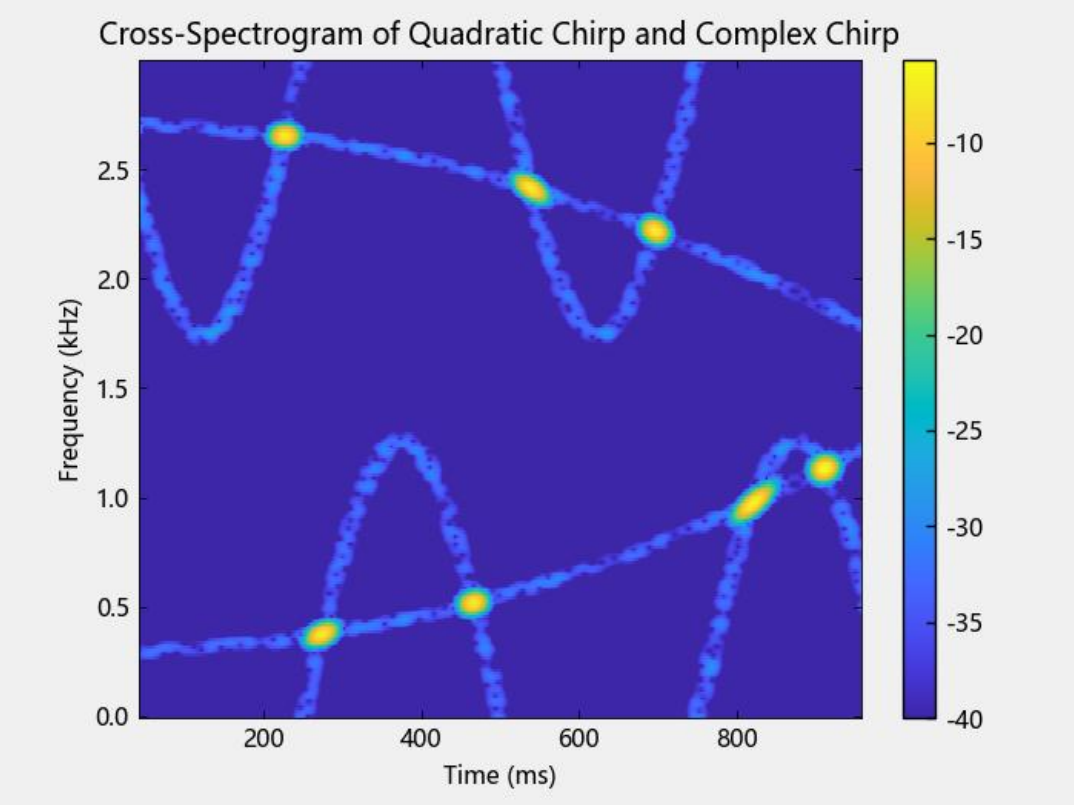

Figure 20: Listing12 结果图 2

```
rx = chirp(0:(N - 1), 0.1 / 2, N, 0.8 / 2, "quadratic", [], "convex
    ") + randn(rng, N) / 100
b1, a1 = cheby2(16, 60, [0.2 0.4], "bandpass", "z")
x, = filter1(real(b1), a1, rx)

ry = chirp(0:(N - 1), 0.9 / 2, N, 0.1 / 2) + randn(rng, N) / 100
b2, a2 = cheby1(16, 1, [0.6 0.8], "bandstop", "z")
y, = filter1(b2, a2, ry)

plot(x)
hold("on")
plot(y .+ 2)

figure()
xspectrogram(x, y, hamming(512), 500, 2048, "yaxis"; plotfig=true)

figure()
xspectrogram(x, y, hamming(512), 500, 2048, "MinThreshold", -50, "
    yaxis"; plotfig=true)
```

Listing 13: 两个序列的交叉谱图

stft

```
s, = stft(x)
```

返回 x 的短时傅立叶变换 (STFT)。

```
s, = stft(x,fs)
```

使用采样率 fs 返回 X 的 STFT。

```
s, = stft(___;Name = Value)
```

使用名称 = 值参数对指定其他选项。选项包括 FFT 窗口长度和重叠样本数。这些参数可以添加到任何先前的输入语法中。

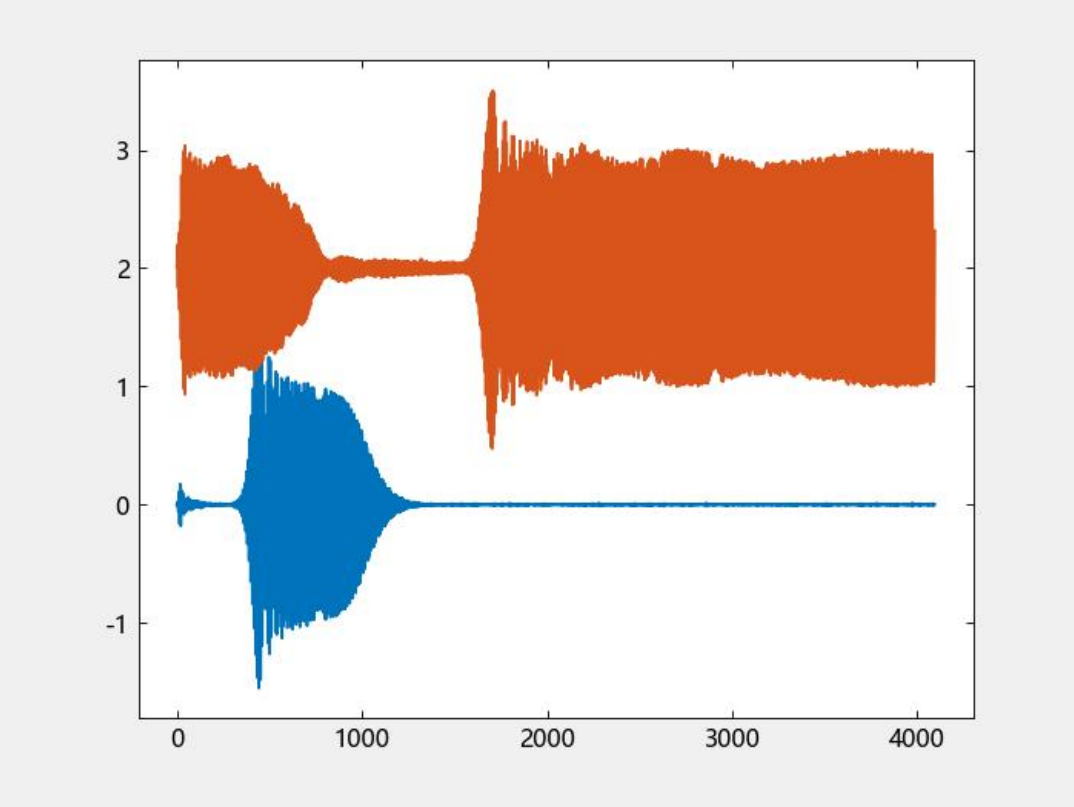

Figure 21: Listing13 结果图 1

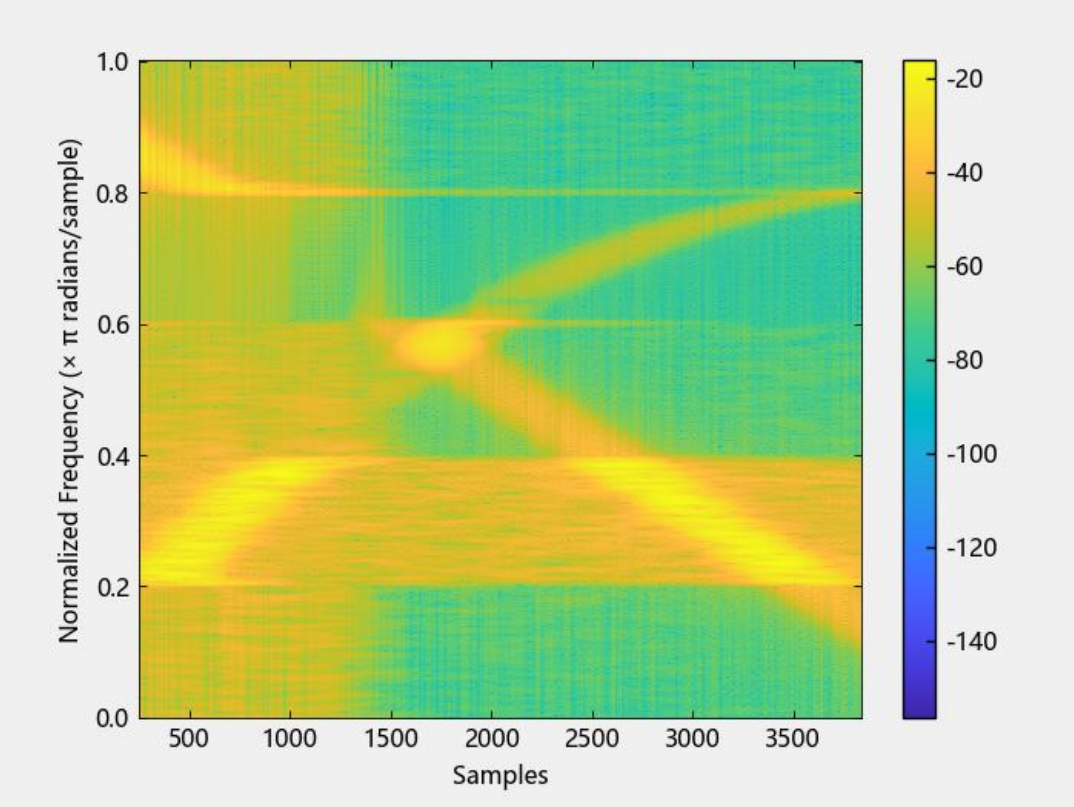

Figure 22: Listing13 结果图 2

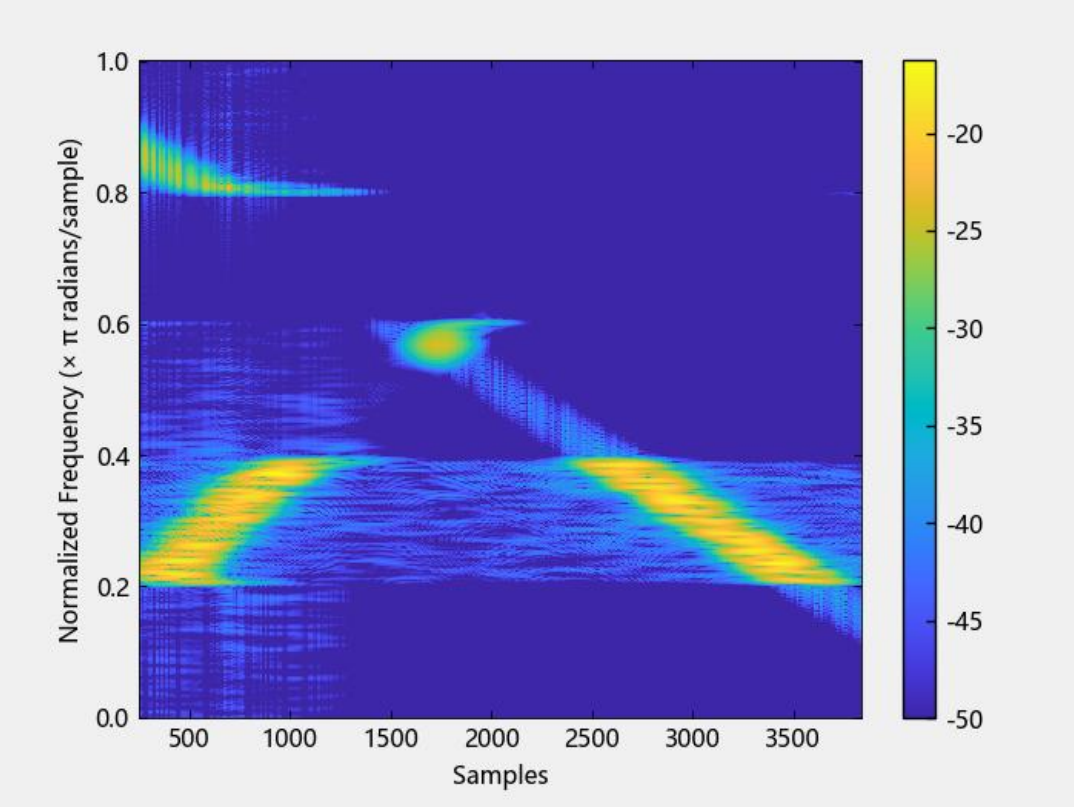

Figure 23: Listing13 结果图 3

```
s,f, = stft(___)
```

返回评估 STFT 的信号频率。

```
s,f,t = stft(___)
```

返回评估 STFT 的信号时间。

```
# 短时傅里叶变换

using TyPlot
using TyBase
using TySignalProcessing

fs = 10e3
t = 0:1/fs:2
x = vco(sin.(2*pi*t),[0.1 0.4]*fs,fs)

x,f,t = stft(x,fs;Window=kaiser(256,5),OverlapLength=220,FFTLength
    =512)
T,F = meshgrid2(t,f)
m = surf(F,T,abs.(x))
colorbar(gca(),m)
plt_view(2)
```

Listing 14: 短时傅里叶变换

```
# 扫频余弦信号 STFT

using TyPlot
using TyBase
using TySignalProcessing

ts = [0:1/1e3:2;]
f0 = 100
f1 = 200
x = chirp(ts,f0,1,f1,"quadratic")
```

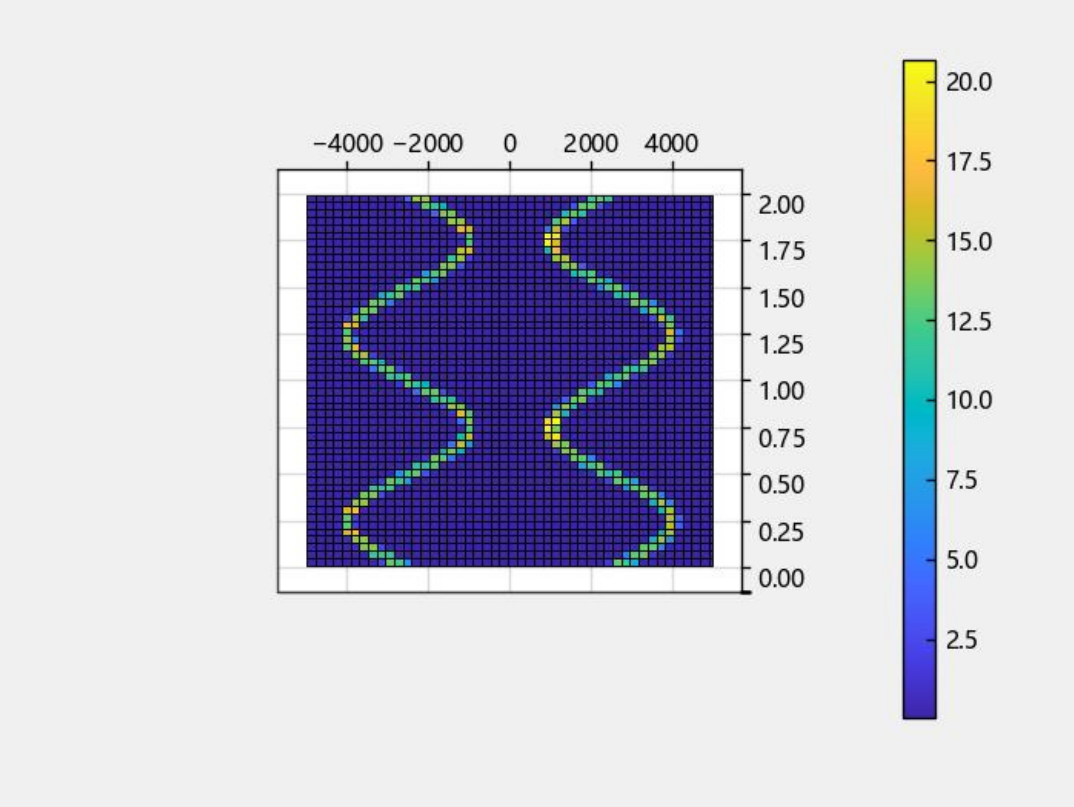

Figure 24: Listing14 结果图

```
11
12 d = 1e-3
13 win = hamming(100,"periodic")
14 x,f,t = stft(x,d;Window=win,OverlapLength=98,FFTLength=128)
15 T,F = meshgrid2(t,f)
16 m = surf(F,T,abs.(x))
17 colorbar(gca(),m)
18 plt_view(2)
```

Listing 15: 扫频余弦信号 STFT

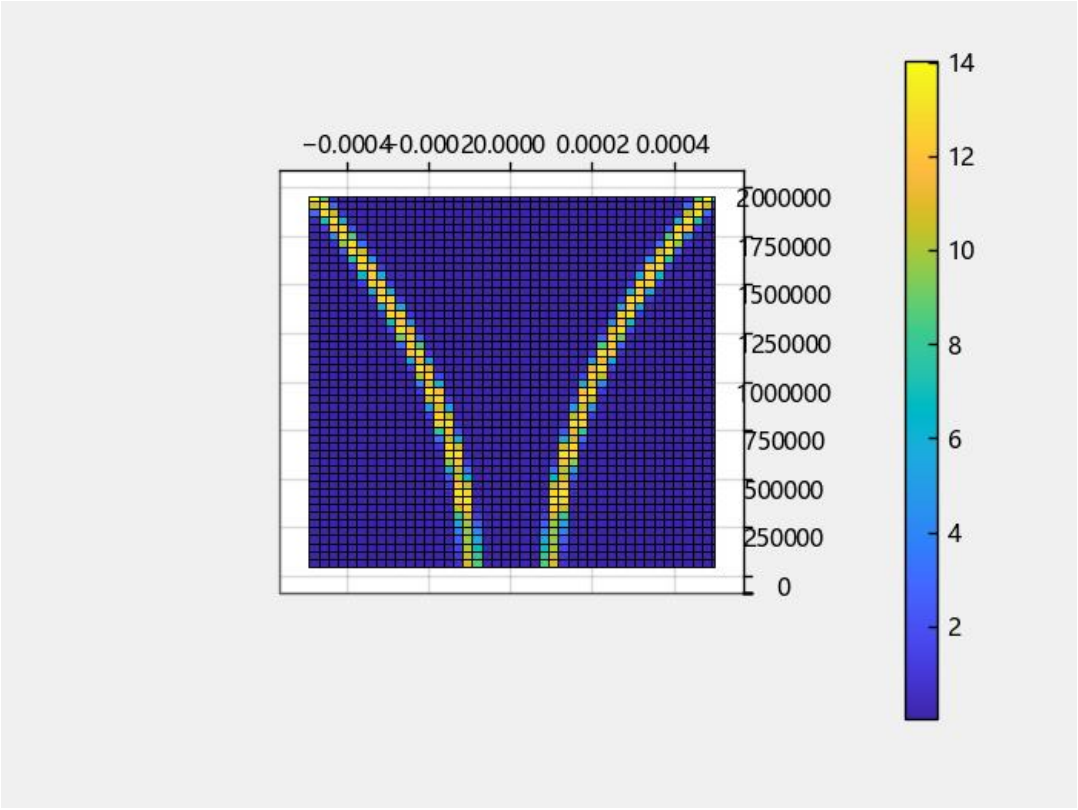

Figure 25: Listing15 结果图

```
1 # STFT 频率范围
2
```

```
using TyPlot
using TyBase
using TySignalProcessing
using TyMath
fs = 5000
t1 = [0:1/fs:4-1/fs;]
x = besselj.(0,600*(sin.(2*pi*(t1.+1).^3/30).^5))
figure()
plot(t1,x)

rngs = ["onesided","twosided","centered"]
figure()
for kj = 1:length(rngs)
y,f,t = stft(x,fs;Window=kaiser(202,10),FrequencyRange=rngs[kj])
T,F = meshgrid2(t,f)
subplot(length(rngs),1,kj)
m = surf(F,T,abs.(y))
colorbar(gca(),m)
title(string(rngs[kj]," [",string(round(f[1]/1000,digits=3)),", ",
    string(round(f[end]/1000,digits=3)),"] ","kHz"))
end

figure()
for kj = 1:length(rngs)
y,f,t = stft(x,fs;Window=kaiser(203,10),FrequencyRange=rngs[kj])
T,F = meshgrid2(t,f)
subplot(length(rngs),1,kj)
m = surf(F,T,abs.(y))
colorbar(gca(),m)
title(string(rngs[kj]," [",string(round(f[1]/1000,digits=3)),", ",
    string(round(f[end]/1000,digits=3)),"] ","kHz"))
end
```

Listing 16: STFT 频率范围

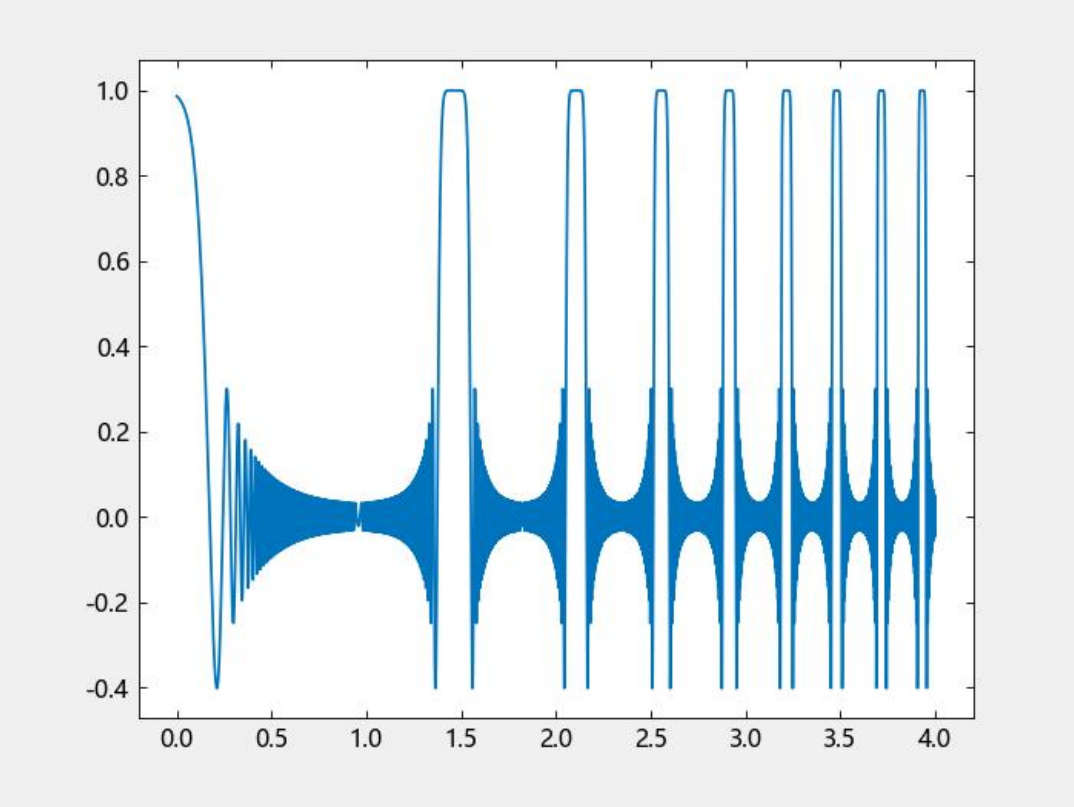

Figure 26: Listing16 结果图 1

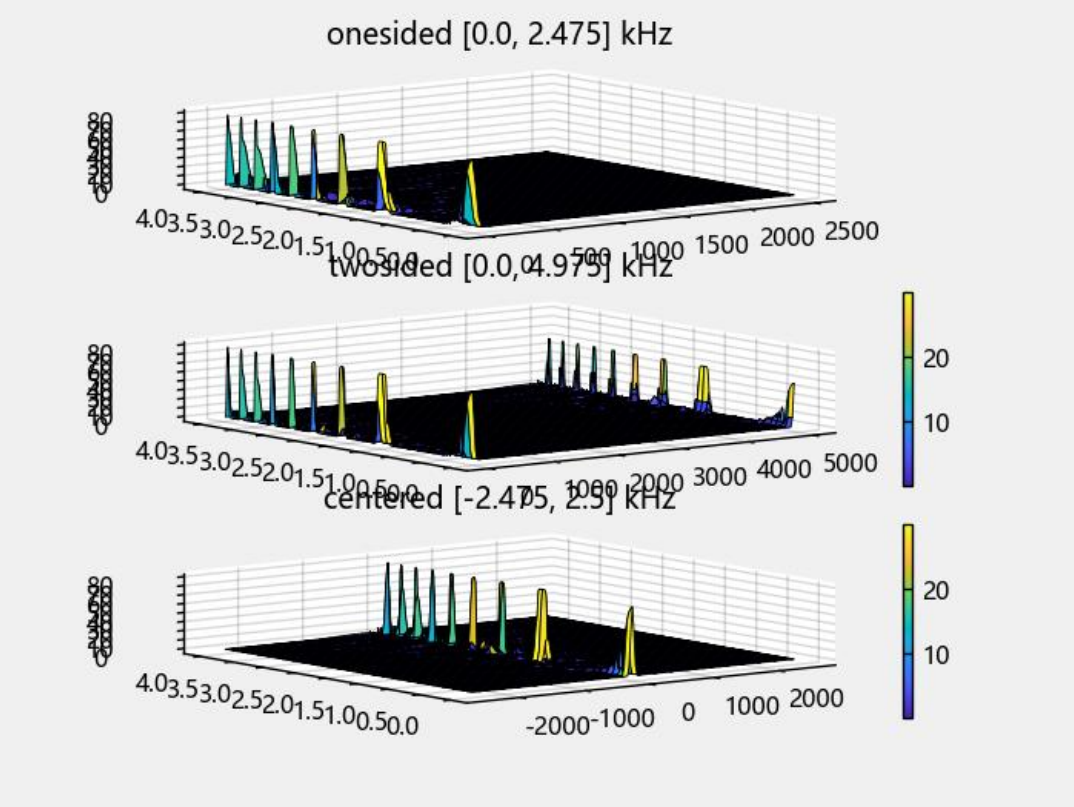

Figure 27: Listing16 结果图 2

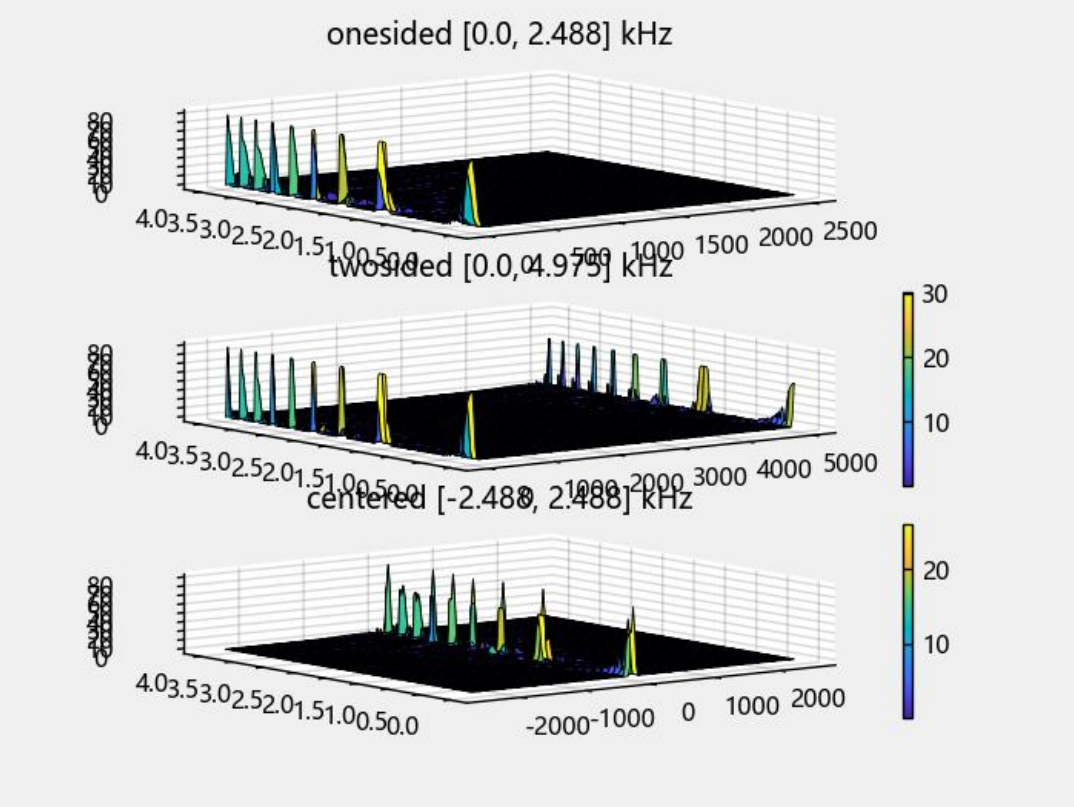

Figure 28: Listing16 结果图 3

iscola

```
b = iscola(window,noverlap)
```

检查指定的窗口和重叠是否满足恒定重叠相加 (COLA) 约束，以确保逆短时傅立叶变换对未修改的光谱产生完美的重建。

```
b = iscola(window,noverlap,method)
```

指定要使用的反演方法。

```
b,m= iscola(___)
```

还返回 COLA 总和的中位数。您可以将这些输出参数与任何先前的输入语法一起使用。

```
b,m,maxDeviation = iscola(___)
```

返回与中位数 m 的最大偏差。

```
using TySignalProcessing

# 检查 Root-Hann 窗口的 COLA 合规性

win = sqrt.(hann(120,"periodic"))
noverlap = 60

b, = iscola(win,noverlap)

# 检查 Hamming 窗的 COLA 合规性

window = hamming(256,"periodic")
method = "ola"
noverlap = 128

a,m,maxDeviation =  iscola(window,noverlap,method)

b,a,m,maxDeviation
```

Listing 17: COLA 合规性