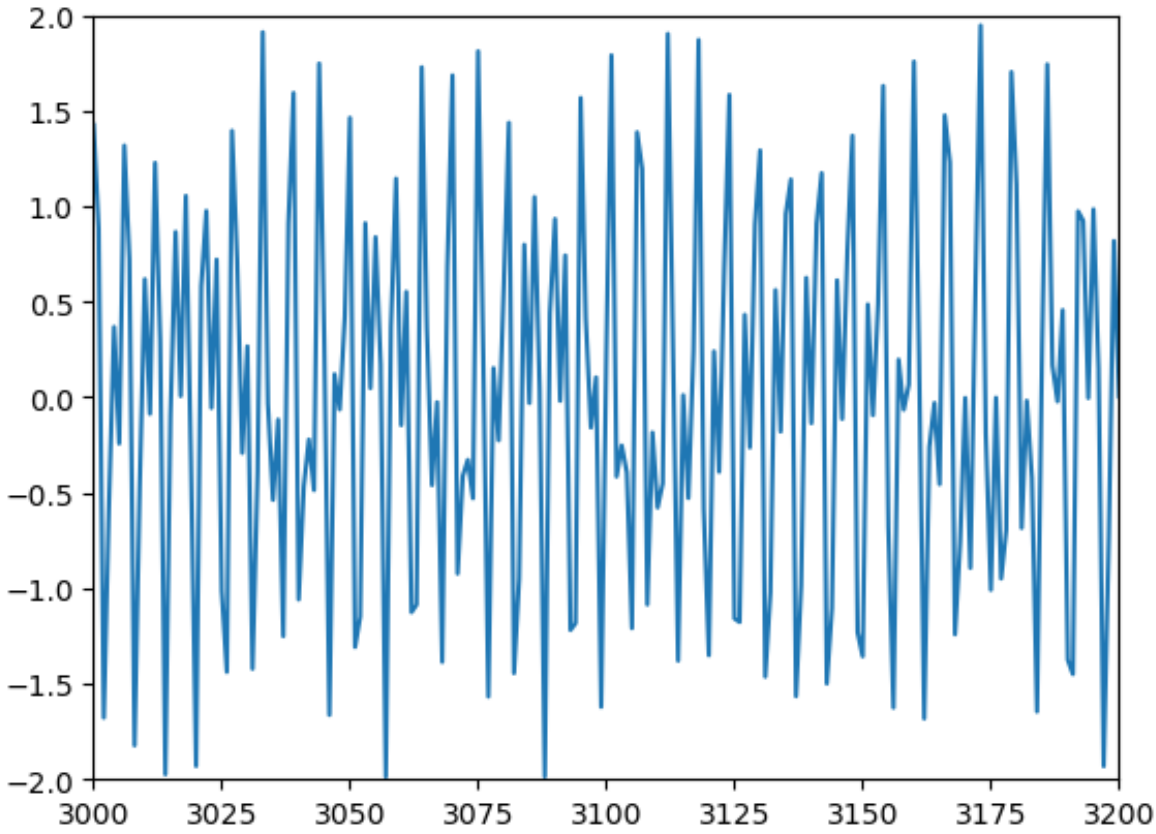
Transformada de Fourier de Tempo Curto (STFT) - Espectrogramas

In [161]:
```
using Pkg
Pkg.activate("/Users/vitor/arquivos/docs/cursos/Julia")
using PyPlot, DSP, FFTW
include("/Users/vitor/arquivos/julia/pfft.jl")
include("/Users/vitor/arquivos/docs/cursos/Julia/plota_stft.jl")
```

```
 Activating environment at `~/arquivos/docs/cursos/Julia/Project.t
oml`
```

Out[161]:
```
plota_stft
```

In [33]:
```
Nsig = 4096
n = 0:Nsig-1
R = 1   # passo no tempo
ω0 = 0.1π
A1 = 10
A2 = 10
Δω = 0.3π/(2Nsig)
ω3 = 0.02
fa = 2 # Frequência de amostragem (normalizada aqui)
s = sin.((ω0 .+ Δω*n) .*n + A1*cos.(ω3*n)) + sin.(2(ω0 .+ Δω*n).*n
+ A2*cos.(ω3*n))
#s = sin.(ω0 * n) + 0.01*sin.(1.5*ω0 * n)
plot(n, s);
init = 3000
axis([init, init+200, -2, 2]);
```


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
3000
3025
3050
3075
3100
3125
3150
3175
3200

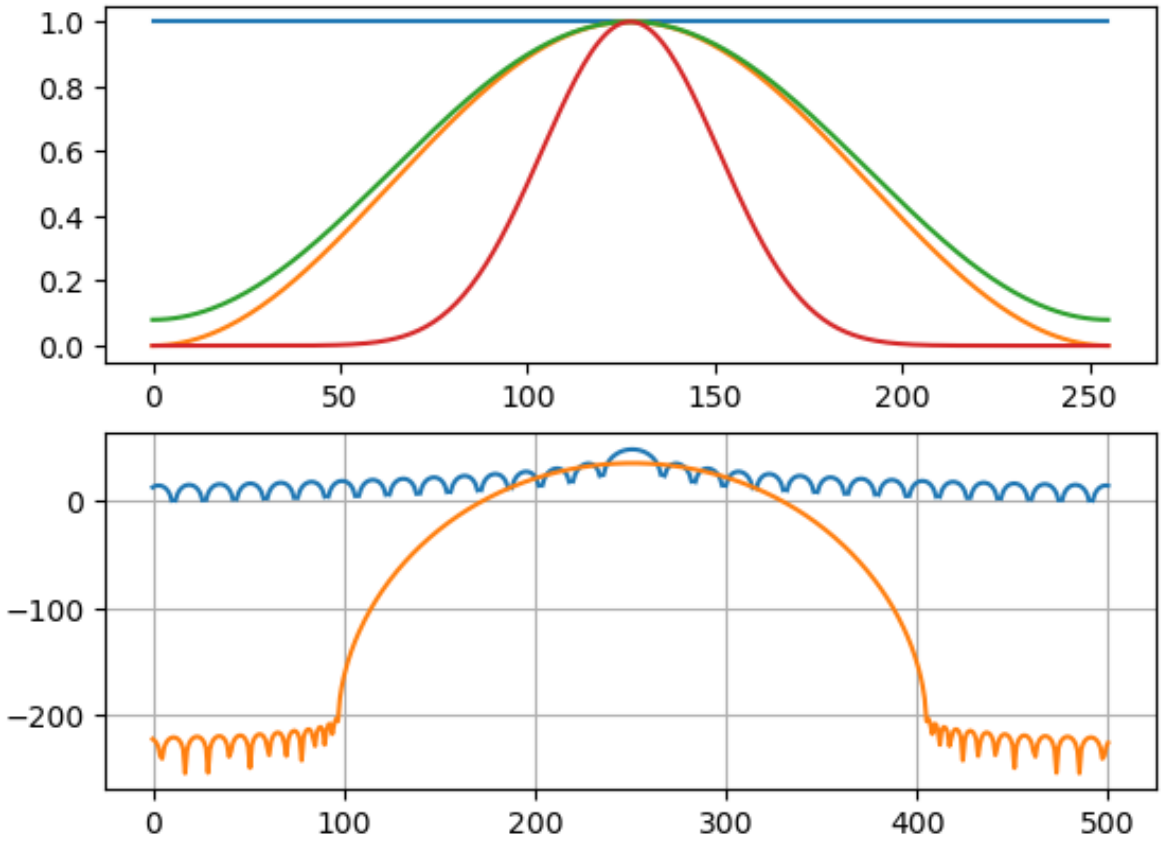
In [34]:
```
M = 256 # comprimento da janela
N = 1024 # número de pontos para a FFT
wr = kaiser(M, 0)
#wr = wr/sum(wr)

w = hanning(M)
#w = w/sum(w)

wh = hamming(M)
#wh = wh/sum(wh)
wk = kaiser(M, 30/π)
#wk = wk/sum(wk)
subplot(211)
plot(wr)
plot(w)
plot(wh)
plot(wk)
subplot(212)
Wr = amp2db.(abs.(pfft(wr, 16M)))
W = amp2db.(abs.(pfft(w, 16M)))
Wh = amp2db.(abs.(pfft(wh, 16M)))
Wk = amp2db.(abs.(pfft(wk, 16M)))

plot([Wr[end-250:end];Wr[1:250]])
#plot([W[end-250:end];W[1:250]])
#plot([Wh[end-250:end];Wh[1:250]])
plot([Wk[end-250:end];Wk[1:250]])

grid();
```



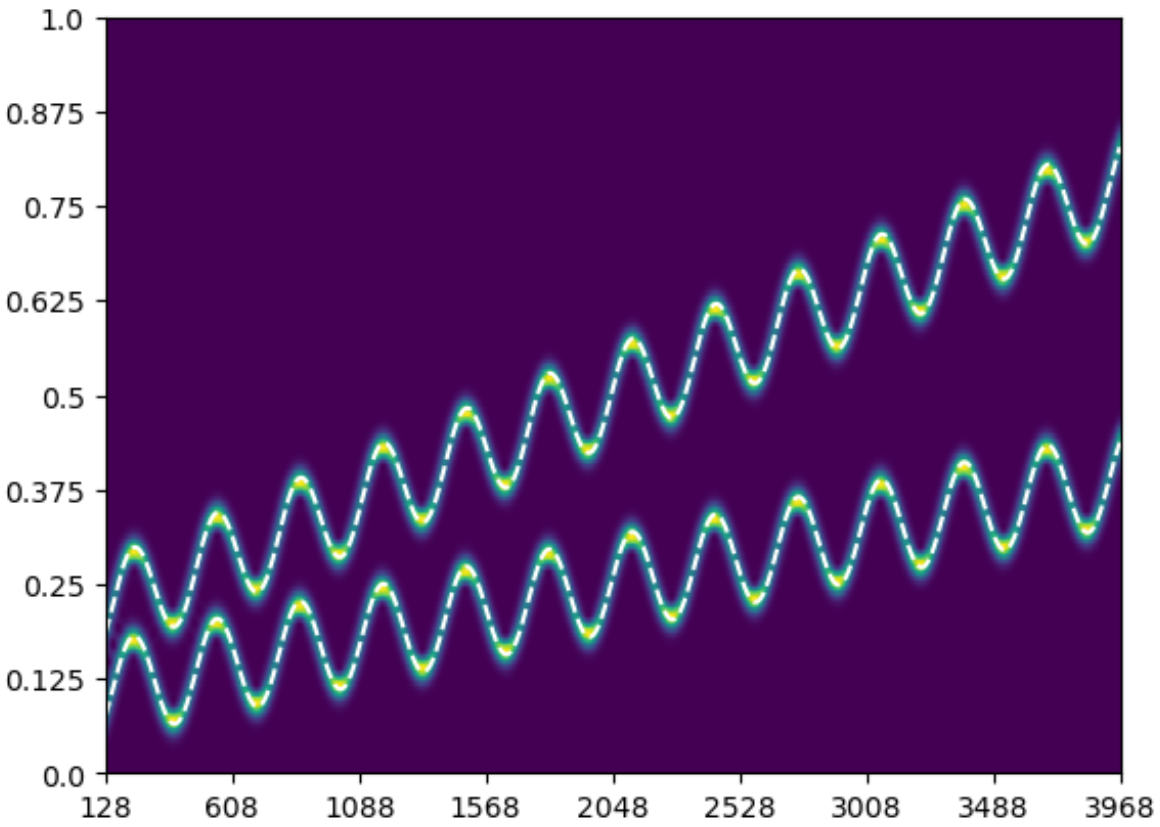
In [116]:
```
methods(plota_stft)
```

Out[116]: # 2 methods for generic function **plota_stft**:

- plota_stft(X::**DSP.Periodograms.Spectrogram{Float64,AbstractFFTs.Frequencies{F**
  N) in Main at /Users/vitor/arquivos/docs/cursos/Julia/plota_stft.jl:8 (file:///Users/vitor/arquivos/docs/cursos/Julia/plota_stft.jl)
- plota_stft(X::**DSP.Periodograms.Spectrogram{Float64,AbstractFFTs.Frequencies{F**
  N, fa) in Main at /Users/vitor/arquivos/docs/cursos/Julia/plota_stft.jl:30 (file:///Users/vitor/arquivos/docs/cursos/Julia/plota_stft.jl)

In [164]:
```
#w = hamming(M)

p = spectrogram(s, M, M-R; fs = fa, window = wk, nfft = N);
ax=plota_stft(p, N);
nt = 2time(p)
ω0inst = (ω0 .+ 2Δω * nt) - ω3*A1*sin.(ω3*nt)
ω1inst = (2ω0 .+ 4Δω*nt) - ω3*A2*sin.(ω3*nt);
especplot(nt, ω0inst/π, nt[1], R, length(time(p)), N÷2+1; col = "w-
-")
especplot(nt, ω1inst/π, nt[1], R, length(time(p)), N÷2+1; col = "w-
-")
```



Out[164]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x1498fcf50>
```

As saídas de p são:

time(p) -> vetor com os instantes correspondentes aos centros de cada janela (considerando a taxa de amostragem informada, que neste caso foi 2). Então, time(p)[1] neste exemplo é 128*0.5

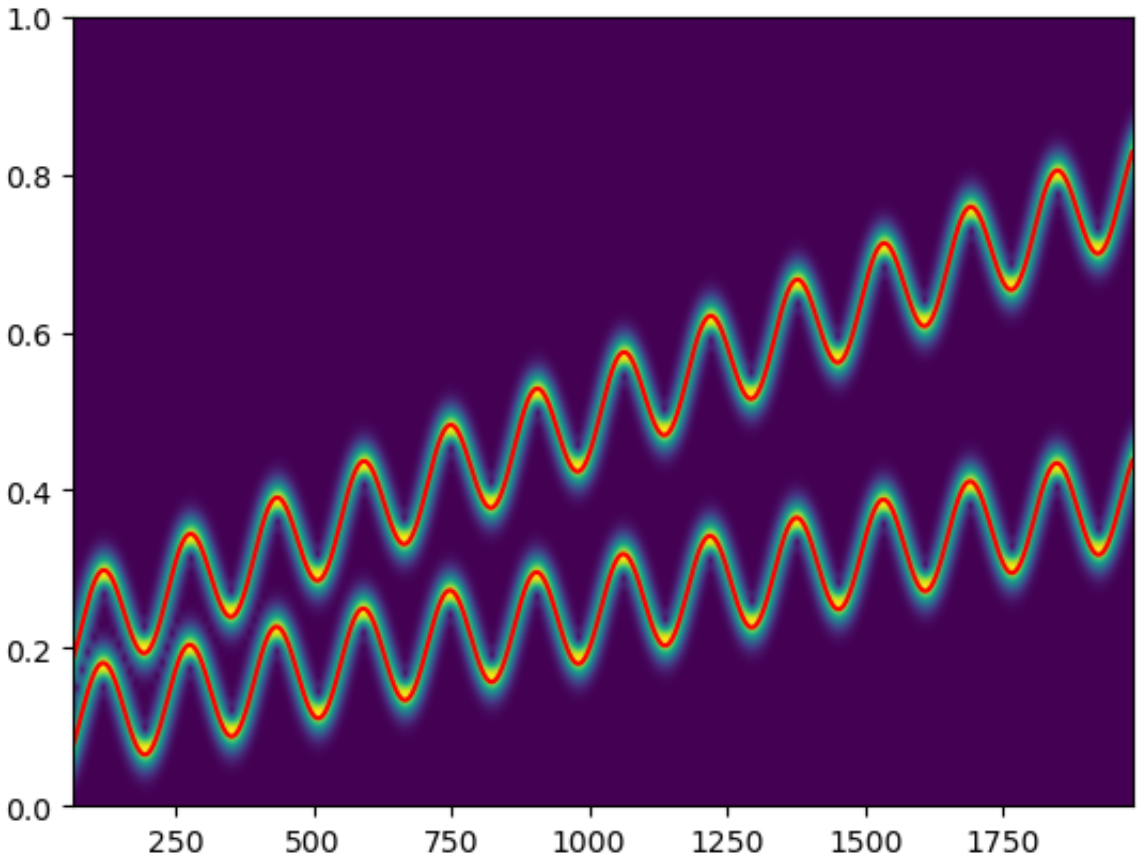
power(p) -> matriz com os valores do espectograma. Cada coluna (power(p)[:,n]) corresponde aos valores absolutos das transformadas nas frequências, que vão até nfft/2.

Para desenhar o espectrograma é possível usar a função `specgram` do PyPlot (que calcula o espectrograma e já faz o desenho) ou a função `pcolormesh` - mas nesse caso é importante planejar o número e posição das marcações horizontais e verticais.

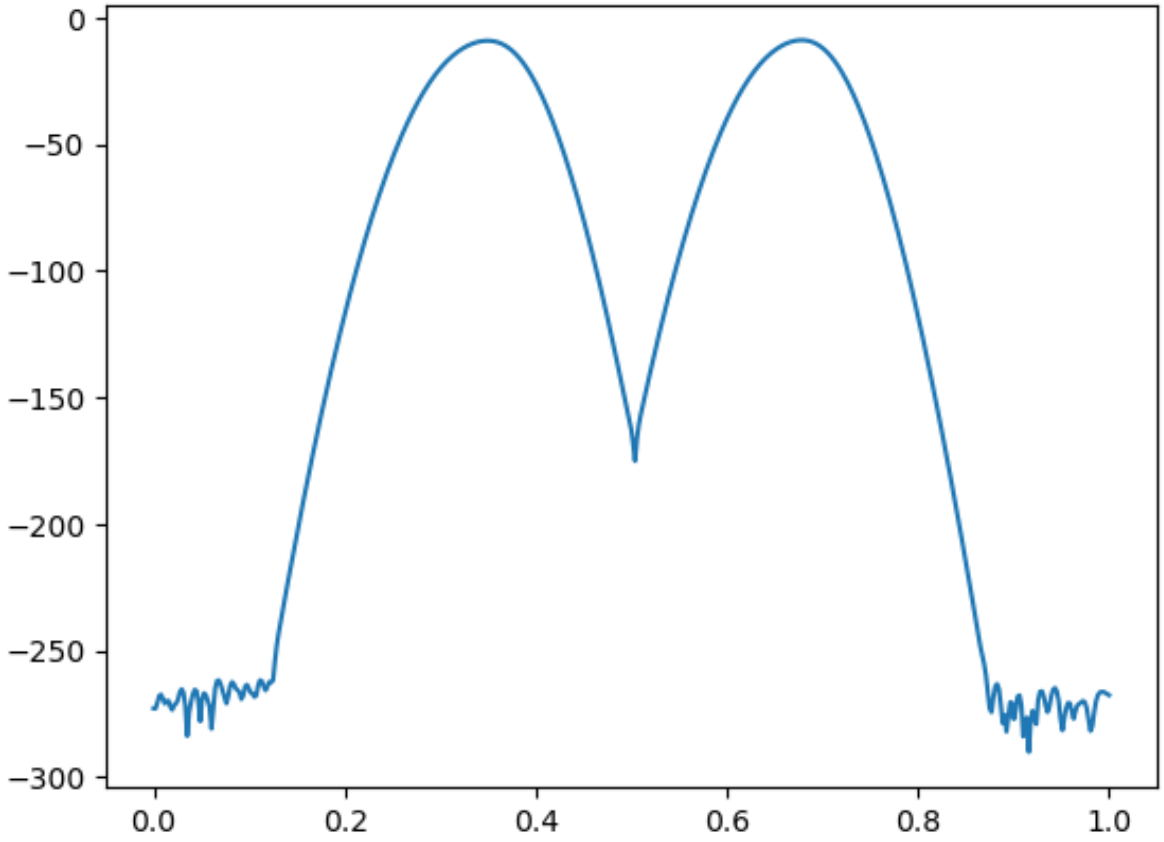
Na vertical, as frequências são dados pelo vetor freq(p).

Na horizontal, o número de pontos depende do comprimento do sinal, de M e de R. A função `time(p)` retorna o valor do instante de tempo correspondente ao centro de cada janela usada no cálculo da STFT. Ou seja, se a janela tiver comprimento M, `time(p)[1]` será igual a $M/2$ --- isto é uma aproximação, um valor melhor seria $(M-1)/2$. Atenção que o valor de fs informado é levado em conta para calcular os instantes de tempo.

In [238]:
```
#subplot(211);
sp,freq,t=specgram(s, Fs = 2, window = wk, scale = "linear", NFFT =
length(wk), noverlap = M-Q, pad_to = N, mode = "magnitude"); # scal
e = "linear" use NFFT = M para usar a janela default
#subplot(212)
nt = 2t # Corrige para a taxa de amostragem (t considera Fs = 2)
ω0inst = ((ω0 .+ 2Δω * (nt)) - ω3*A1*sin.(ω3*(nt)))
ω1inst = ((2ω0 .+ 4Δω*(nt)) - ω3*A2*sin.(ω3*(nt)));
plot(t , ω0inst/π, "r");
plot(t , ω1inst/π, "r");
#axis([0, 2000, 0, 1]);
```



In [240]:
```
k = (0:N÷2)
plot(k*2/N,amp2db.(sp[:,3000]));
```



In [241]:
```
using FileIO: load
```

In [242]:
```
using SampledSignals
```

In [243]:
```
svoz=load("/Users/vitor/arquivos/docs/cursos/psi3531/LPC/antarctica
.wav")
```

Out[243]:
```
([0.0; 0.0; … ; -0.007812738425855281; -0.007812738425855281], 800
0.0f0, 0x0010, WAV.WAVChunk[WAV.WAVChunk(Symbol("fmt "), UInt8[0x1
0, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0x40, 0x1f, 0x00, 0x0
0, 0x80, 0x3e, 0x00, 0x00, 0x02, 0x00, 0x10, 0x00])])
```

In [244]:
```
svozb = SampleBuf(svoz[1],svoz[2])
```

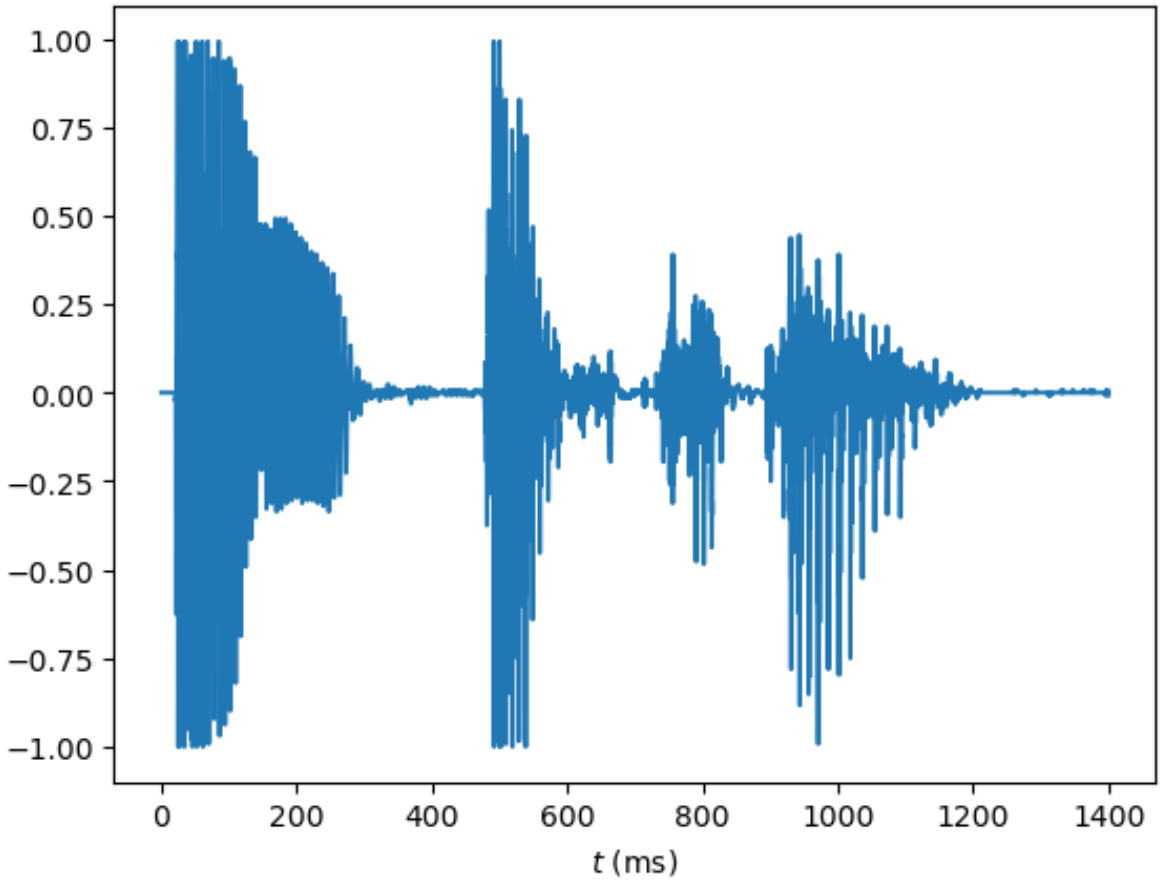
Out[244]:

-0:01

In [245]:
```
tvoz = 1000*(0:length(svoz[1])-1)/svoz[2]
```

Out[245]:
```
0.0f0:0.125f0:1400.0f0
```

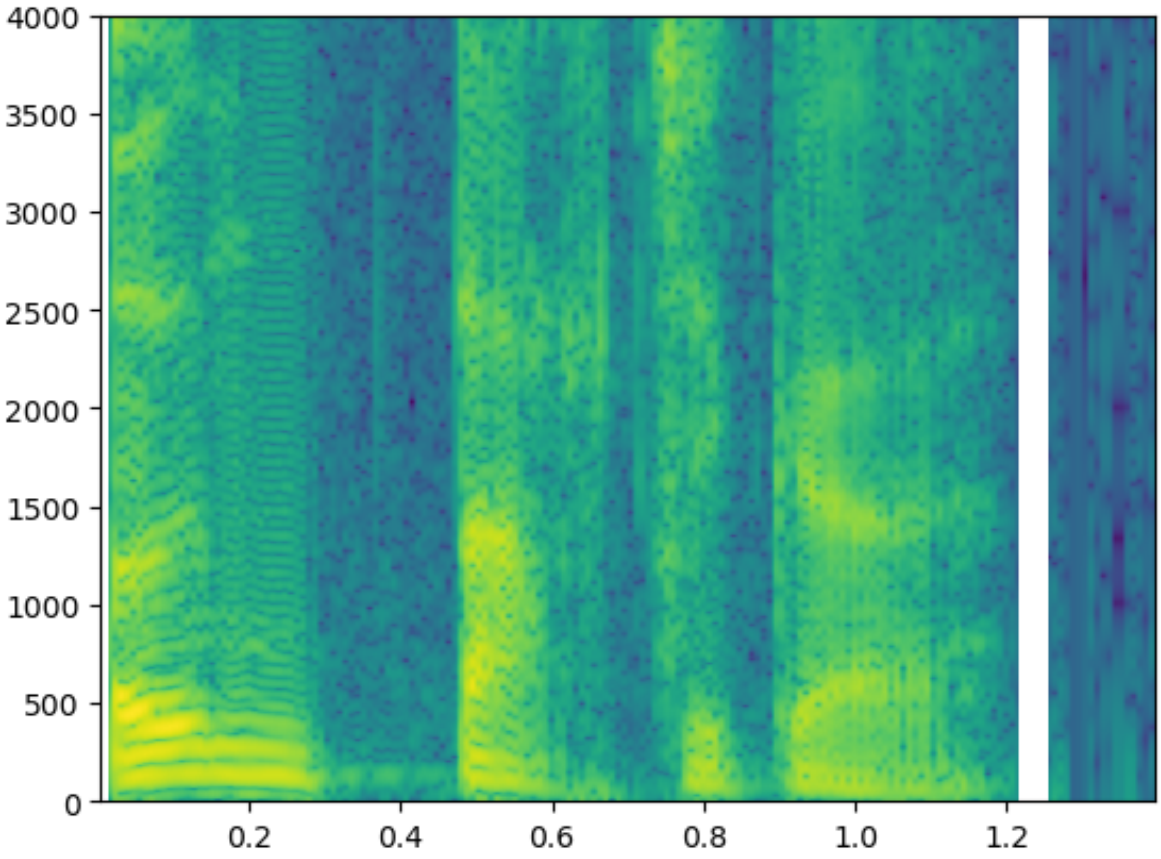
In [246]:
```
plot(tvoz,svoz[1])
xlabel(L"$t$ (ms)");
```



In [247]:
```
Mv = 128

wv = kaiser(Mv, 2/π)

specgram(svoz[1][:,1], Fs = svoz[2], window = wv, NFFT = Mv, noverl
ap = Mv÷2, pad_to = N, mode = "magnitude"); # use NFFT = M para usa
r a janela default
```



In [ ]:

In [ ]: