Conversão de taxa de amostragem

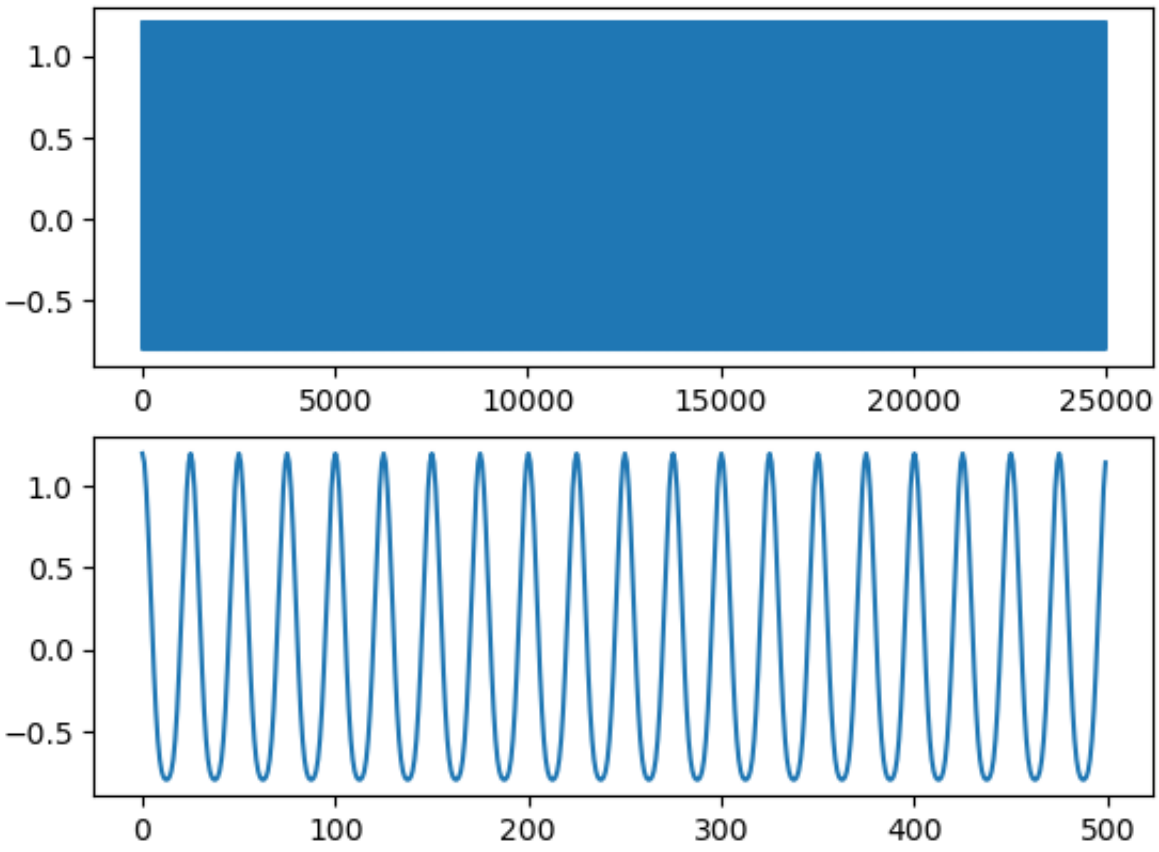
In [1]:
```
using PyPlot, DSP, WAV;
```

Decimação

Decimação com rebatimento - filtragem

Considere o exemplo a seguir:

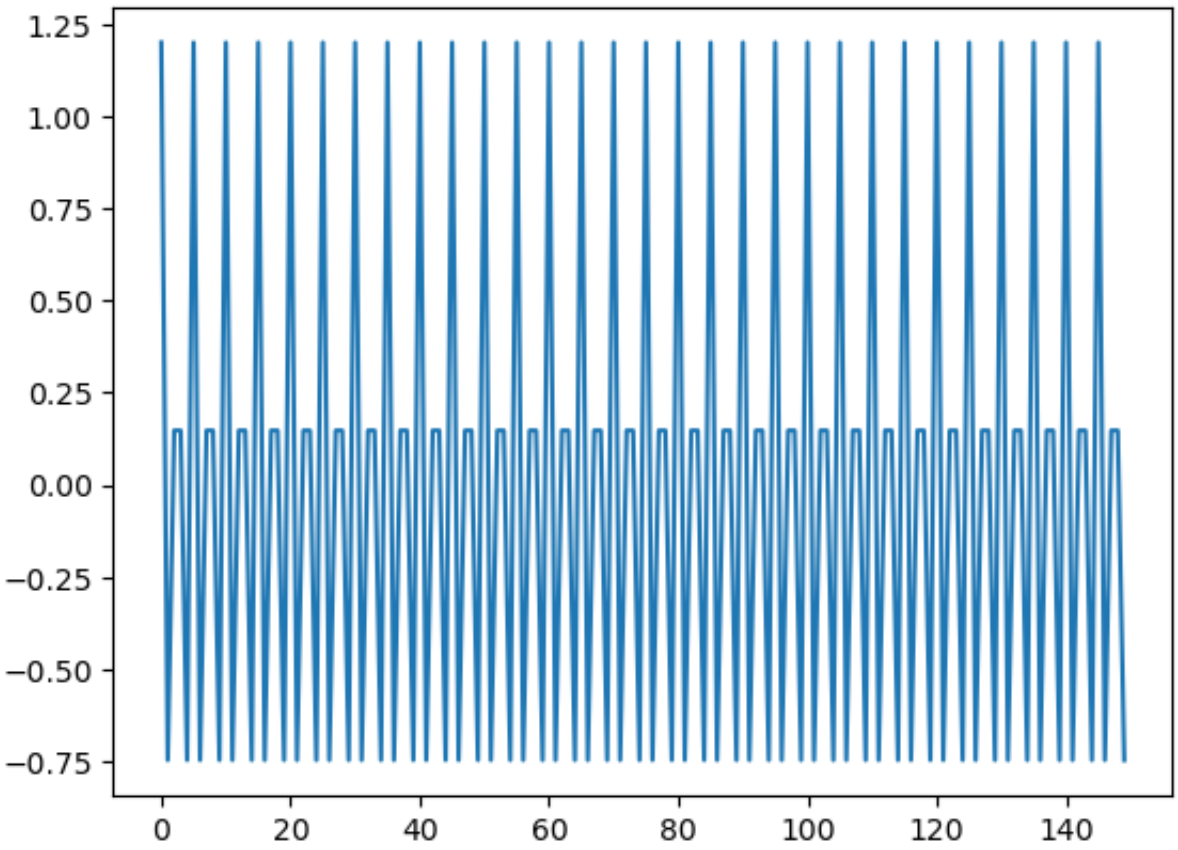
In [23]:
```
f0 = 400
f1 = 800
fa = 10_000
n = 0:1000fa÷f0
x0 = cos.(2π*f0*n/fa)
x1 = 0.2cos.(2π*f1*n/fa)
xa = x0 + x1
subplot(211)
plot(n,xa);
subplot(212)
plot(n[1:500],xa[1:500]);
```



In [27]:
```
wavplay(xa, fa)
```

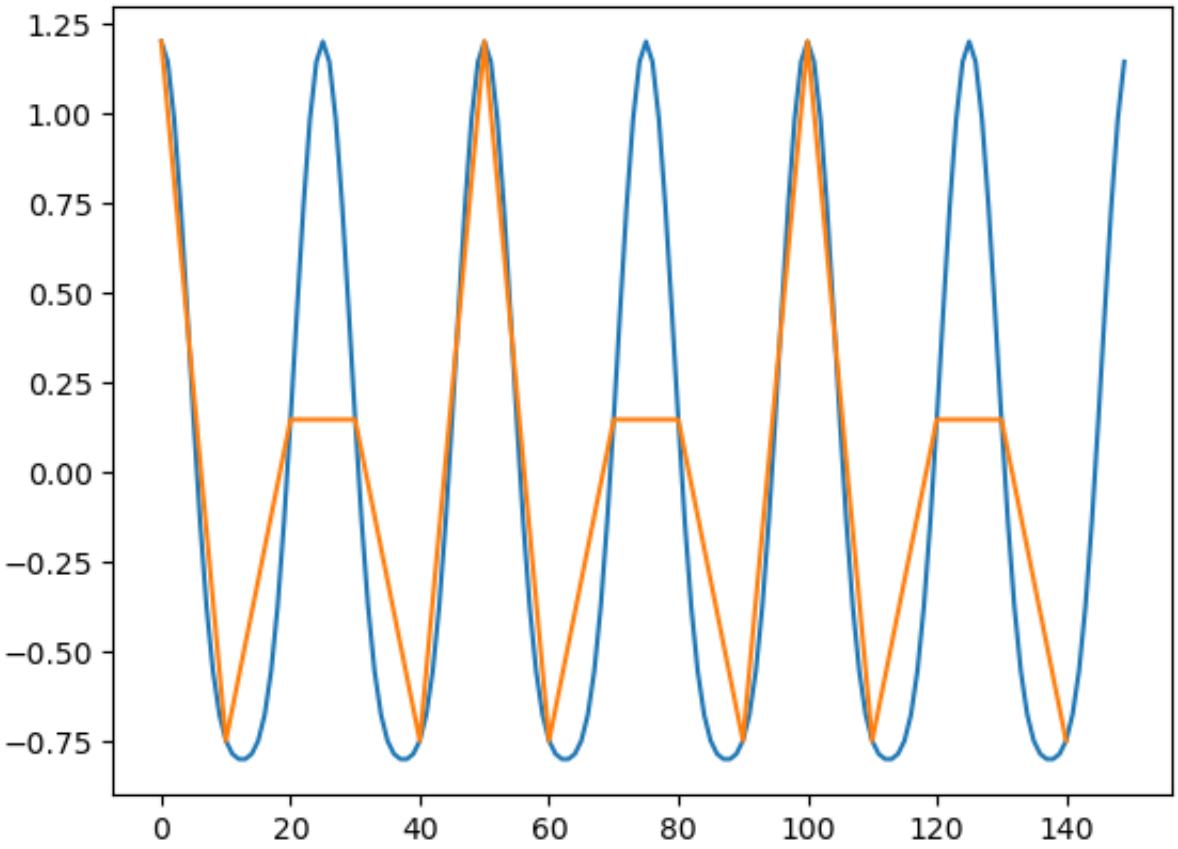
In [28]:

```
M = 10
xb = xa[1:M:end]
m=0:length(xb)-1
plot(m[1:150], xb[1:150]);
```



In [81]:

```
plot(n[1:150],xa[1:150])
plot(n[1:M:150], xb[1:15]);
```

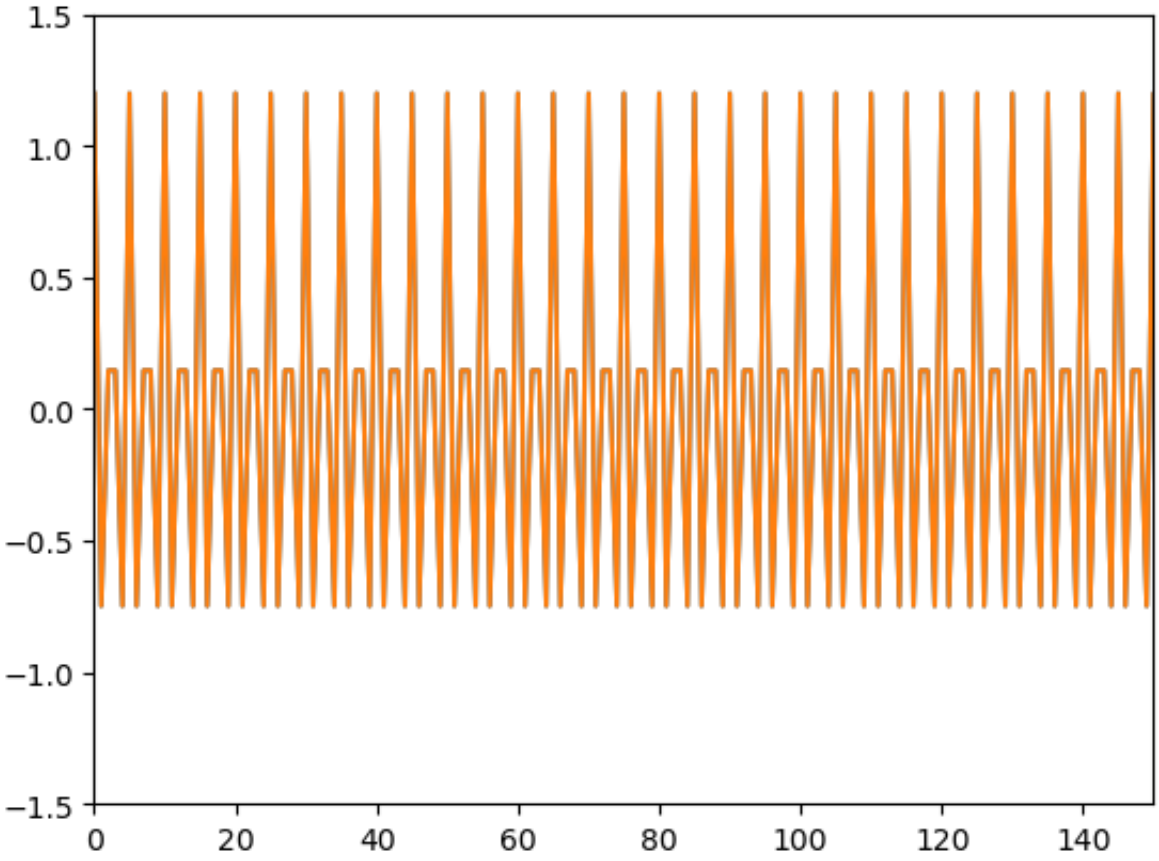


In [29]:
```
wavplay(xb, fa/M)
```

In [33]:
```
fa/M - f1
```

Out[33]:
```
200.0
```

In [34]:
```
plot(m,xb, m, cos.(2π*f0*m*M/fa)+0.2cos.(2π*(fa/M-f1)*m*M/fa))
axis([0, 150, -1.5, 1.5]);
```



In [35]:
```
fp = 2*f0/fa
```

Out[35]:
```
0.08
```

In [36]:
```
fr = 2*f1/fa
```

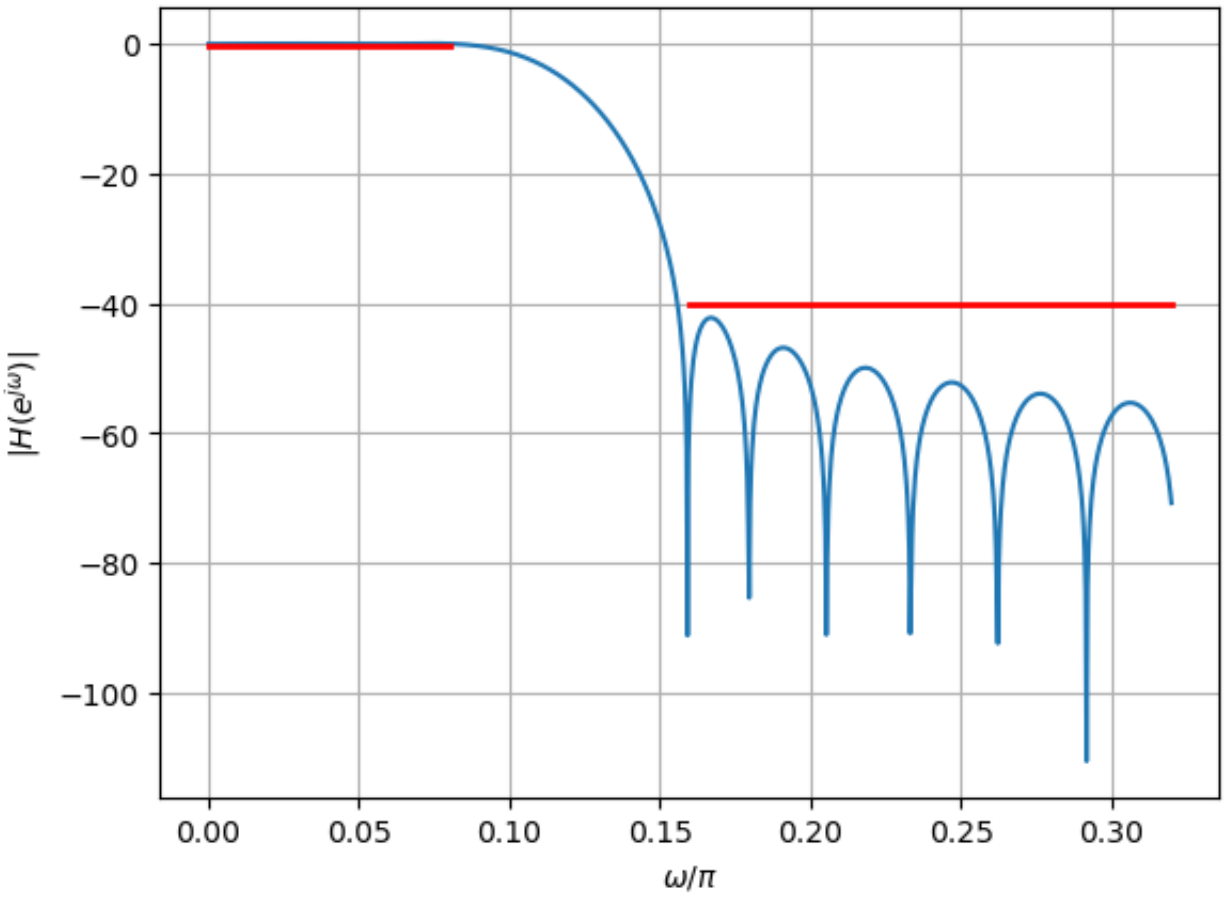
Out[36]:
```
0.16
```

In [37]:
```
include("/Users/vitor/docs/cursos/Julia/kaiser.jl")
```

Out[37]:
```
filtrokaiser
```

In [38]:
```
δp = 0.05
δr = 0.01

h = PolynomialRatio(filtrokaiser(π*fp, π*fr, δp, δr)[1], [1])
Nf = length(h.b)
h = PolynomialRatio(filtrokaiser(π*fp, π*fr, δp, δr, Nf+7)[1], [1])
ω = range(0, 2π*fr, length = 1000)
H = freqz(h, ω)
plot(ω/π, amp2db.(abs.(H)))
xlabel(L"\omega/\pi")
ylabel(L"|H(e^{j\omega})|");
grid()
plot([0, fp], fill(amp2db(1-δp),2), "r", lw=2)
plot([fr, ω[end]/π], fill(amp2db(δr), 2), "r", lw=2);
Nf
```



Out[38]: 57

In [39]:
```
A = -20*log10(min(δp, δr))
```

Out[39]: 40.0

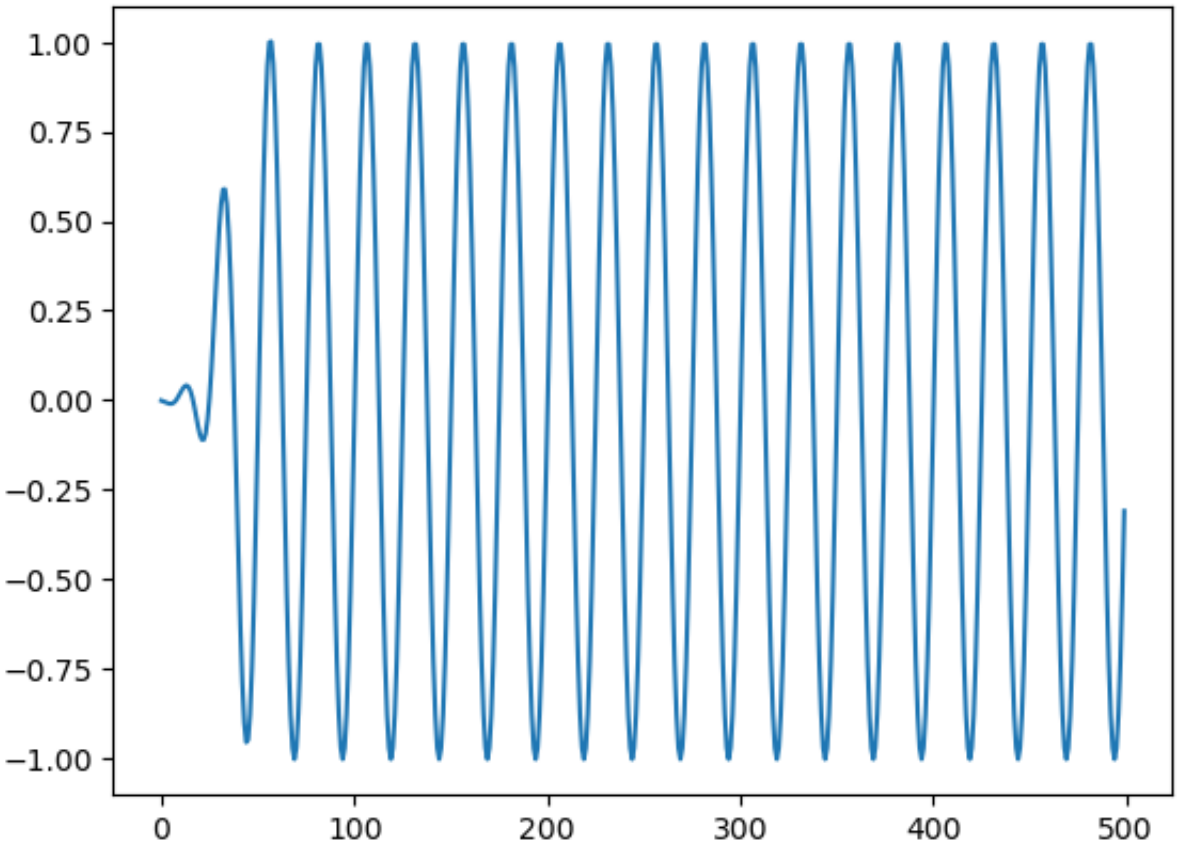
In [40]:
```
Nf = (A-8)/(2.285*(π*fr-π*fp))+1
```

Out[40]: 56.721643095630746

In [41]:
```
β = 0.5842(A-21)^0.4 + 0.07886(A-21)
```

Out[41]: 3.3953210522614574

In [42]:
```
xf = filt(h, xa)
plot(n[1:500], xf[1:500]);
```

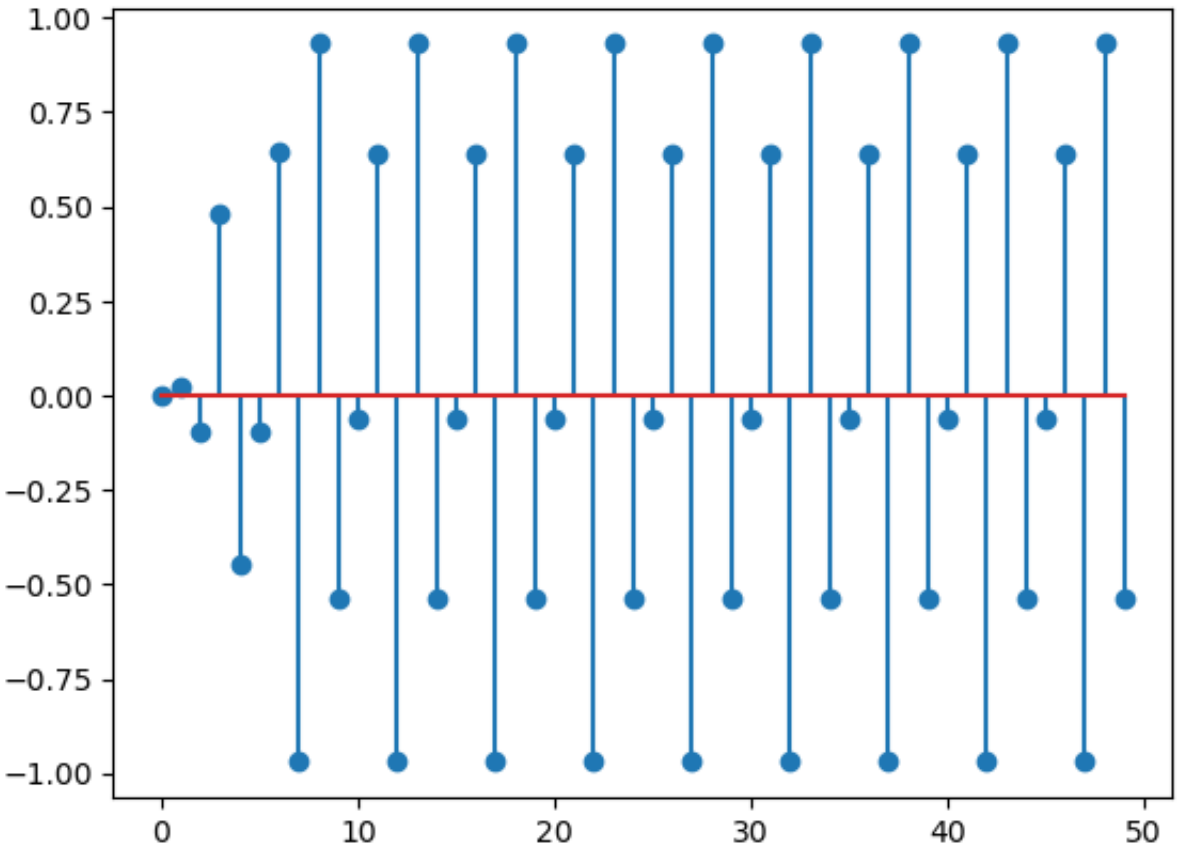


In [53]:

```
wavplay(xf, fa)
```

In [54]:

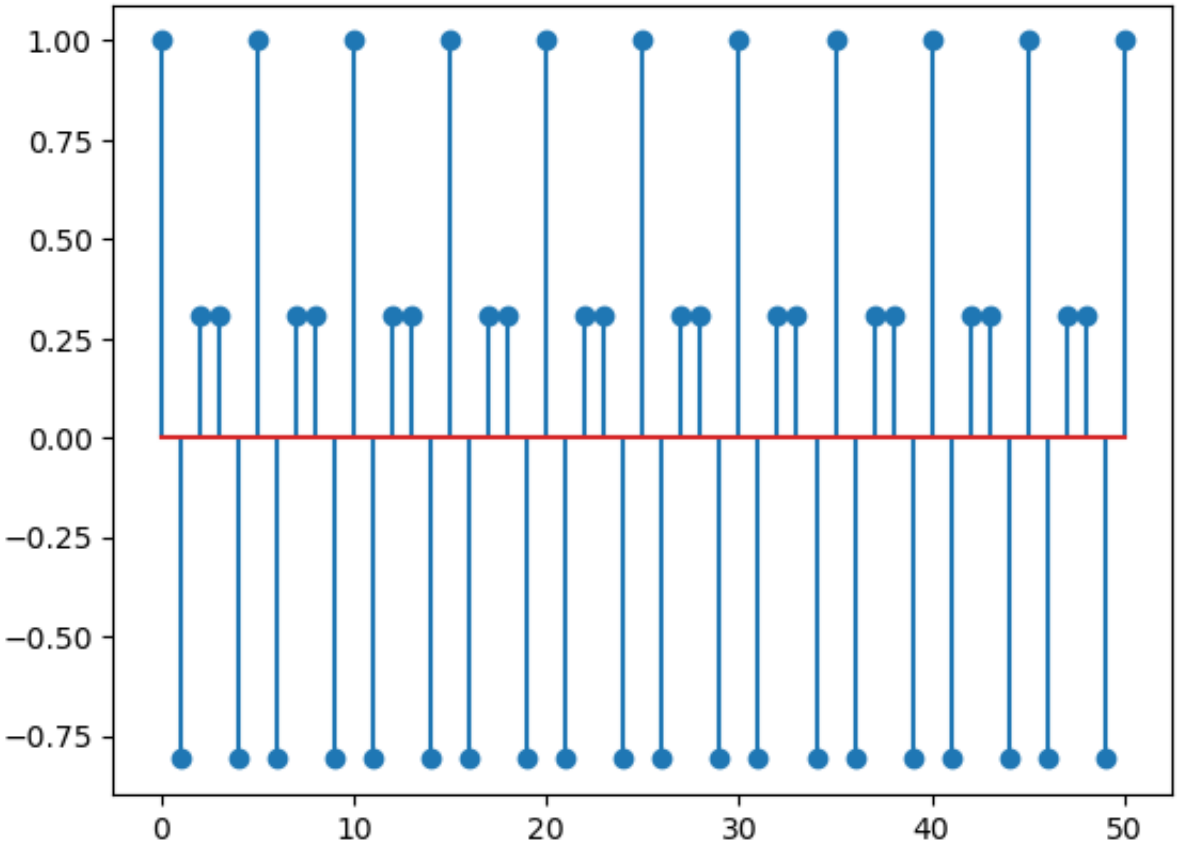
```
xfb = xf[1:M:end]
stem(m[1:50], xfb[1:50]);
```



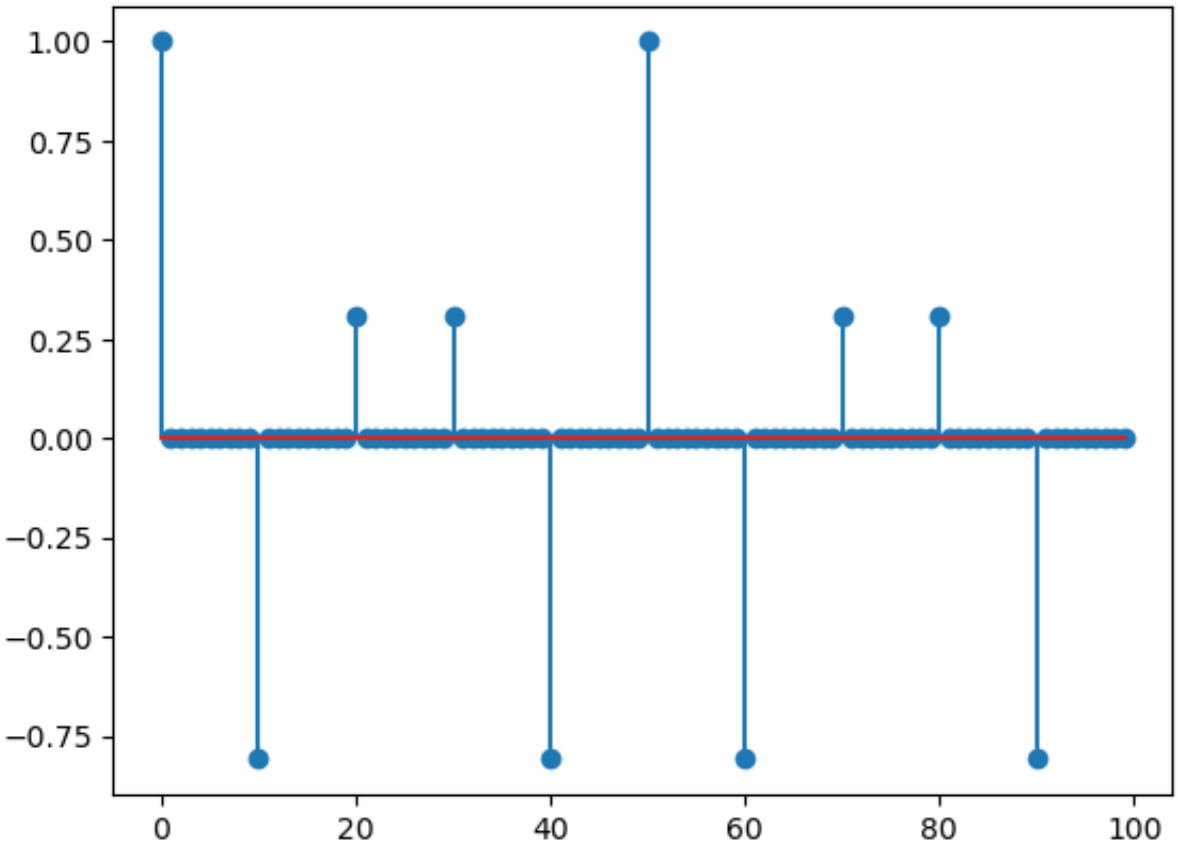
In [52]:
```
wavplay(xfb, fa/M)
```

Interpolador

In [55]:
```
f0i = 4_000
xc = cos.(2π*f0i*n/fa)
stem(n[1:51], xc[1:51]);
```



In [56]:
```
L = 10
xci = zeros(L*length(xc))
xci[1:L:end] = xc
mi = 0:length(xci)-1
stem(mi[1:100], xci[1:100]);
```



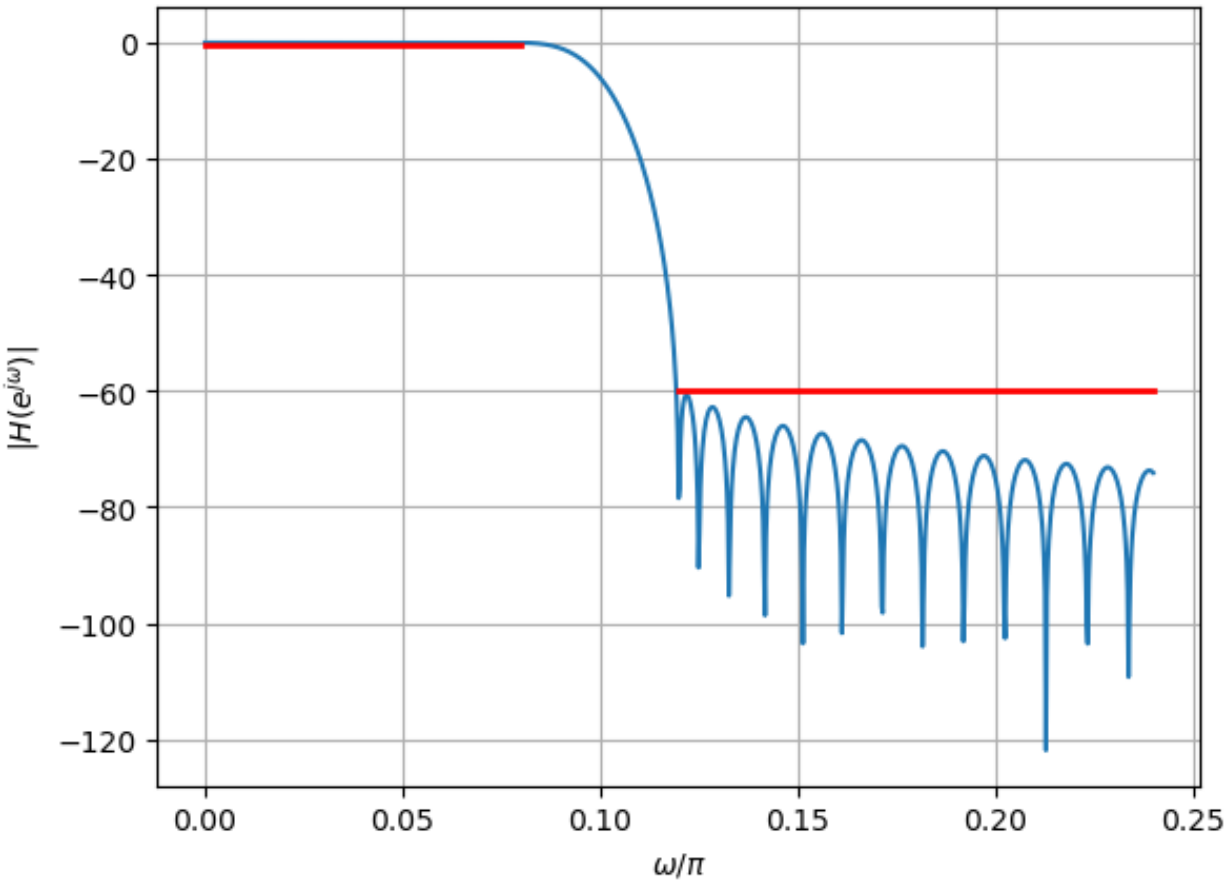
In [64]:
```
fpi = 2*f0i/(L*fa)
```

Out[64]:
```
0.08
```

In [65]:
```
fri = 1/L + (1/L - fpi)
```

Out[65]:
```
0.12000000000000001
```

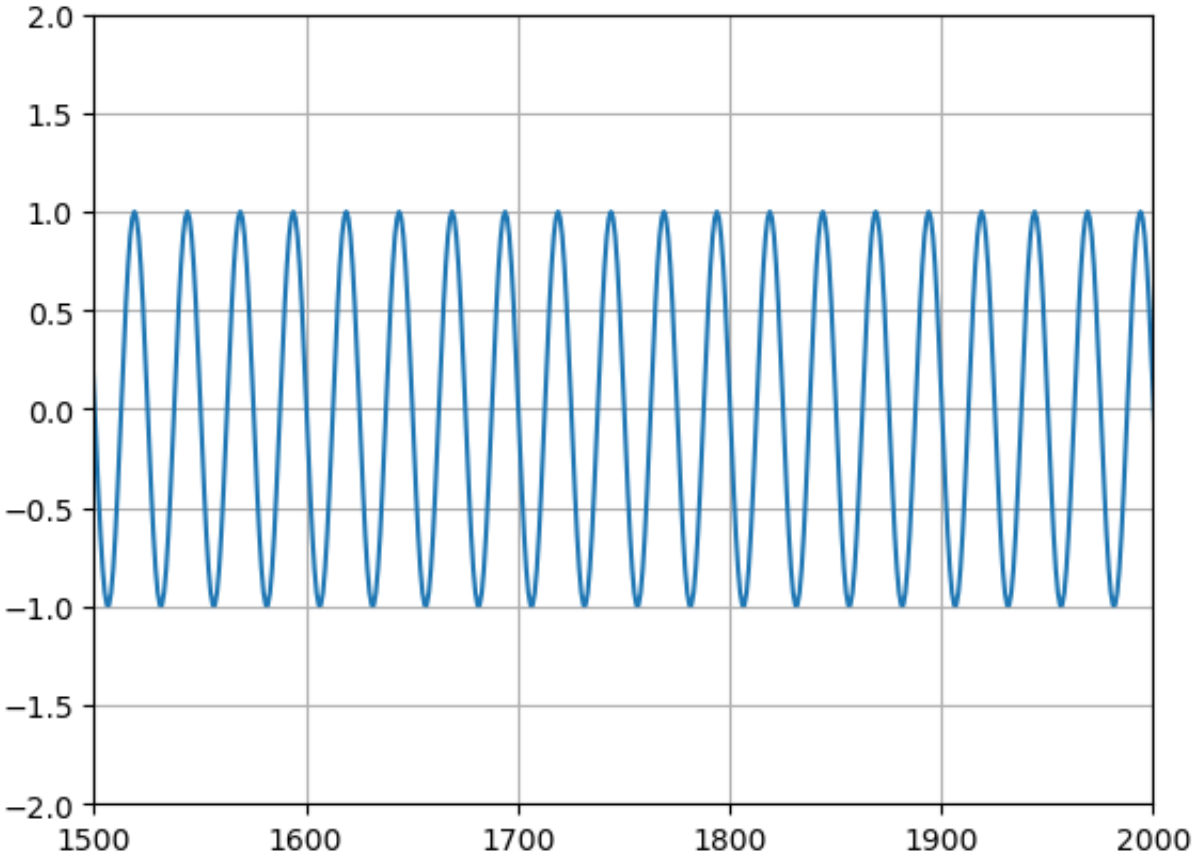
In [66]:
```
δpi = 0.05
δri = 0.001 # Tente também 0.001
hi = PolynomialRatio(filtrokaiser(π*fpi, π*fri, δpi, δri)[1], [1])
Nfi = length(hi.b)
hi = PolynomialRatio(filtrokaiser(π*fpi, π*fri, δpi, δri, Nfi+6)[1], [1])
ωi = range(0, 2π*fri, length = 1000)
Hi = freqz(hi, ωi)
plot(ωi/π, amp2db.(abs.(Hi)))
xlabel(L"\omega/\pi")
ylabel(L"|H(e^{j\omega})|");
grid()
plot([0, fpi], fill(amp2db(1-δpi),2), "r", lw=2)
plot([fri, ωi[end]/π], fill(amp2db(δri), 2), "r", lw=2);
Nfi
```



Out[66]: 183

In [67]:

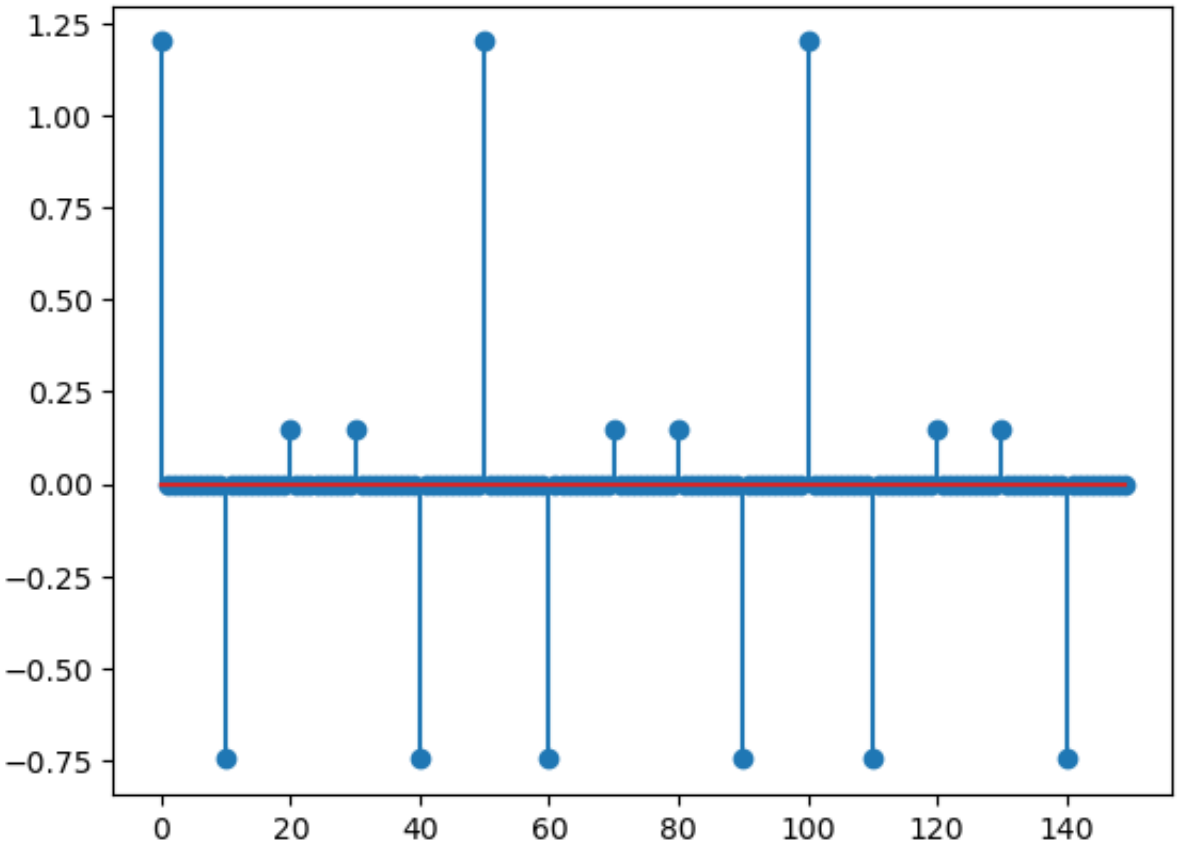
```
xd = L*filt(hi, xci)
plot(mi, xd)
axis([1500, 2000, -2, 2])
grid();
```



Aumentando a taxa de amostragem do sinal decimado do item 1

In [82]:

```
xbi = zeros(L*length(xb))
xbi[1:L:end] = xb
mi2 = 0:length(xbi)-1
stem(mi2[1:150], xbi[1:150]);
```



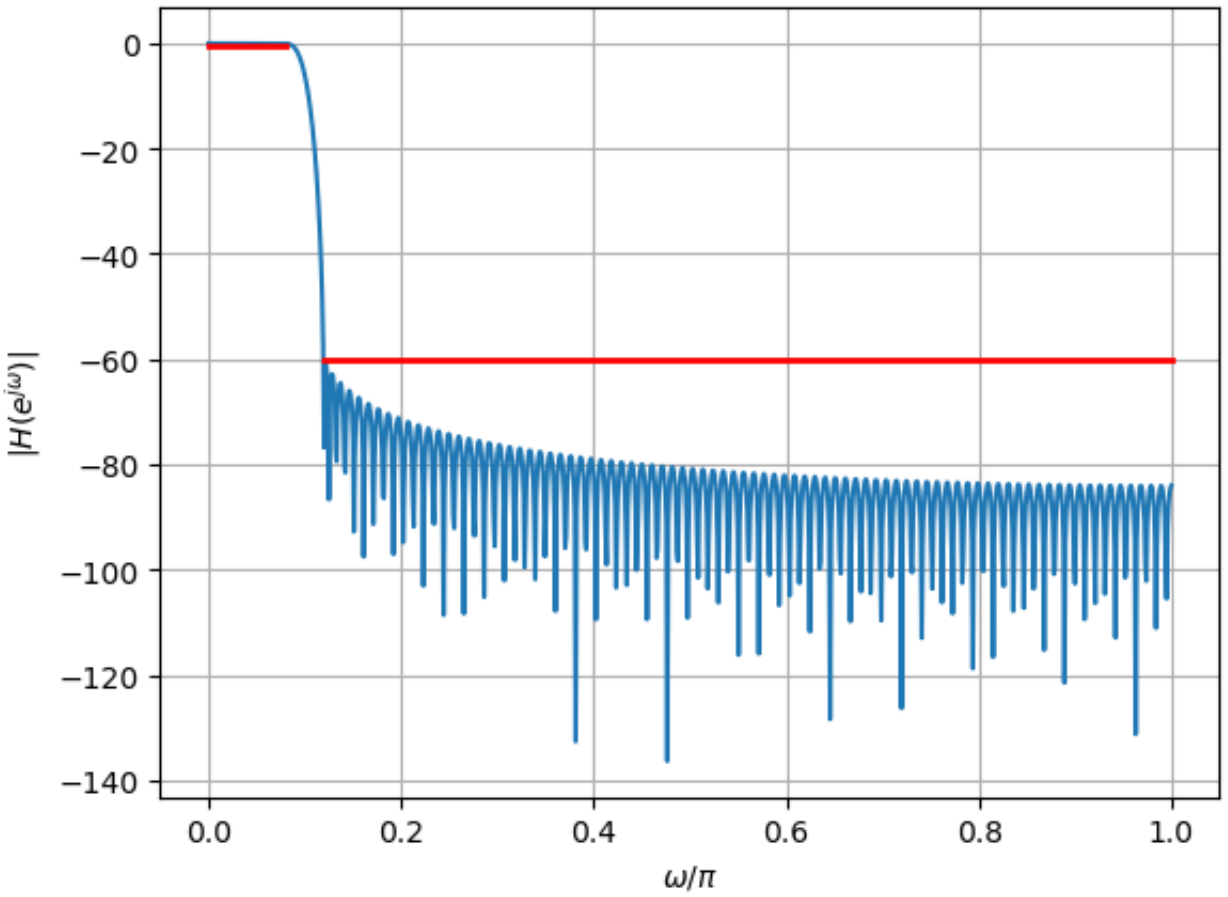
In [70]:
```
fpi2 = 2*f0/(L*fa/M)
```

Out[70]:
```
0.08
```

In [71]:
```
fri2 = 2/L - fpi2
```

Out[71]:
```
0.12000000000000001
```

In [93]:
```
δpi2 = 0.05
δri2 = 0.001 # Tente também 0.001
hi2 = PolynomialRatio(filtrokaiser(π*fpi2, π*fri2,
        δpi2, δri2)[1], [1])
Nfi2 = length(hi2.b) + 6
hi2 = PolynomialRatio(filtrokaiser(π*fpi2, π*fri2,
        δpi2, δri, Nfi2)[1], [1])
ωi2 = range(0, π, length = 1000)
Hi2 = freqz(hi2, ωi2)
plot(ωi2/π, amp2db.(abs.(Hi2)))
xlabel(L"\omega/\pi")
ylabel(L"|H(e^{j\omega})|");
grid()
plot([0, fpi2], fill(amp2db(1-δpi2),2), "r", lw=2)
plot([fri2, ωi2[end]/π], fill(amp2db(δri2), 2), "r", lw=2);
```

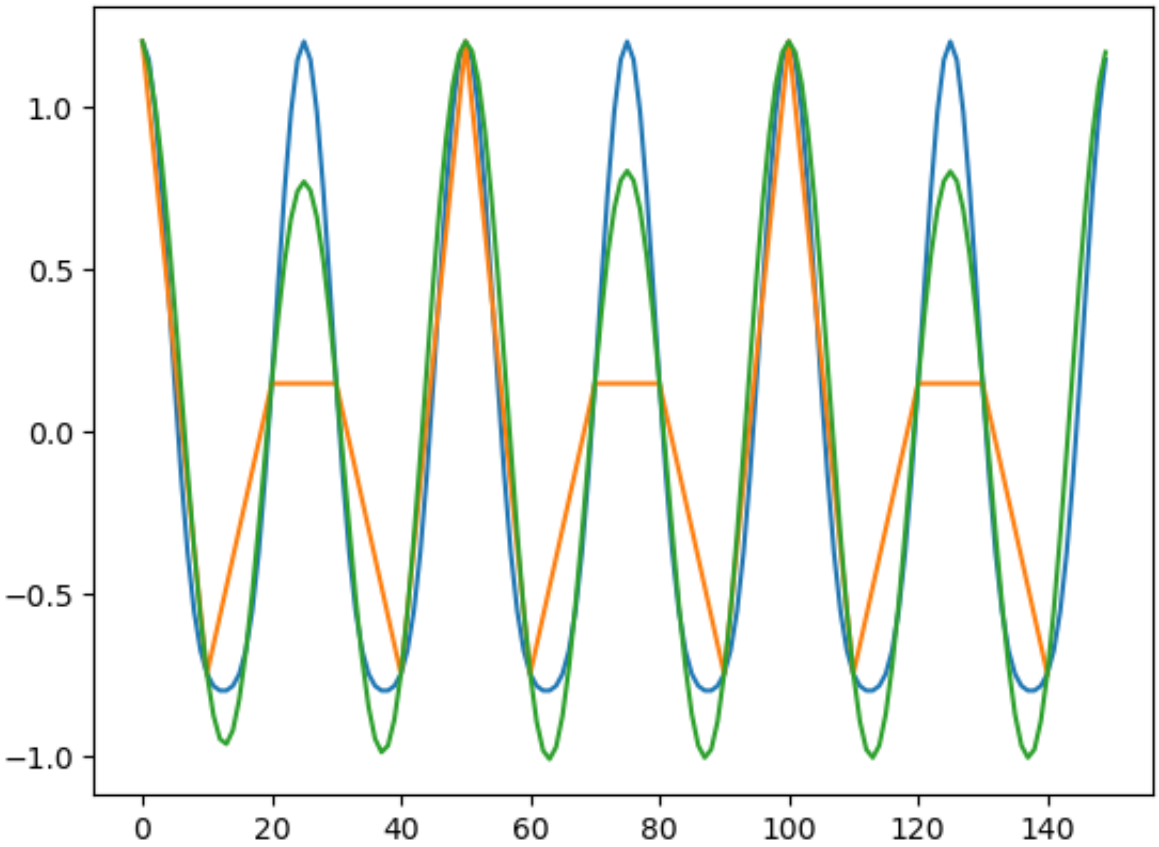


Atraso do filtro:

In [94]:
```
Lfi2 = (Nfi2-1) ÷ 2
```

Out[94]:
```
94
```

In [95]:
```
xbif = L*filt(hi2, xbi)
plot(n[1:150],xa[1:150])
plot(n[1:M:150], xa[1:M:150])
plot(mi2[(1:150)], xbif[(1:150) .+ Lfi2]);
```



Veja como o sinal reamostrado (verde) é diferente do sinal original, por causa do rebatimento. Experimente refazer este exemplo, usando duas frequências para o sinal original, mas de maneira que não haja rebatimento, e você verá que o sinal reamostrado será uma boa aproximação do sinal original nessa situação.

In [ ]: