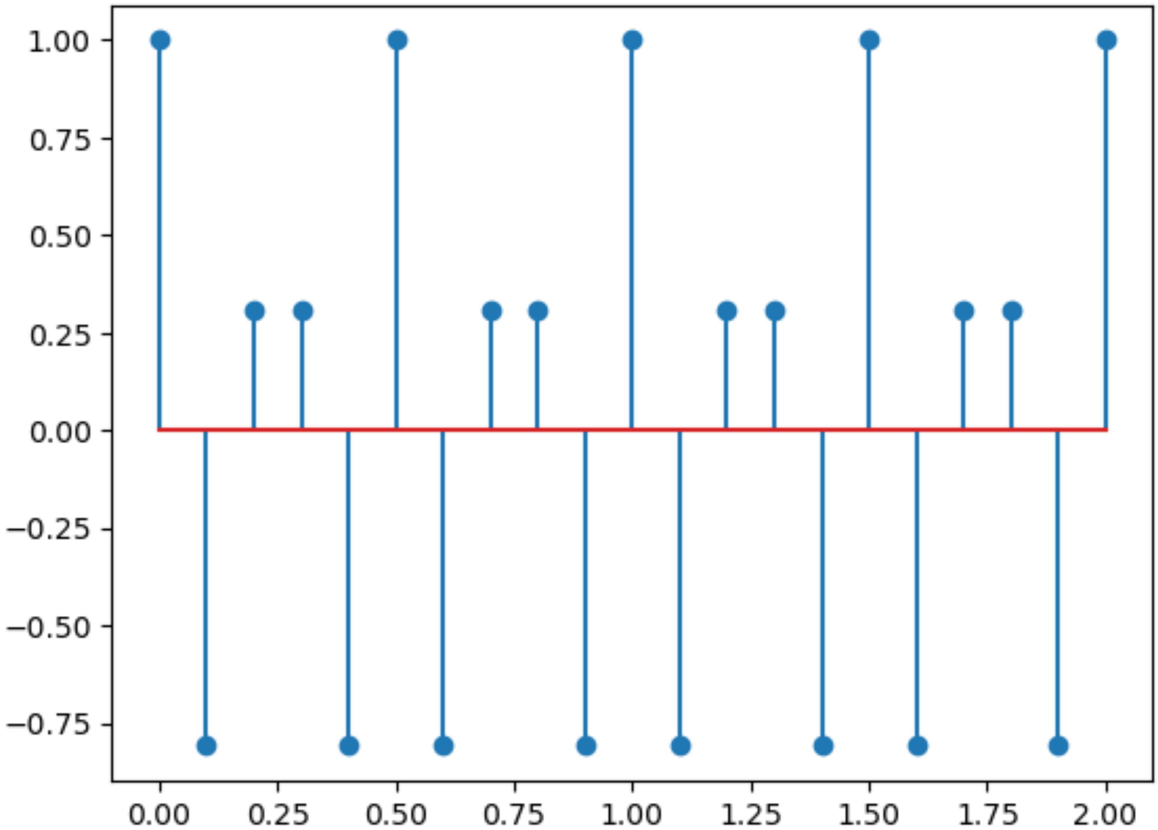
Conversão de taxa de amostragem

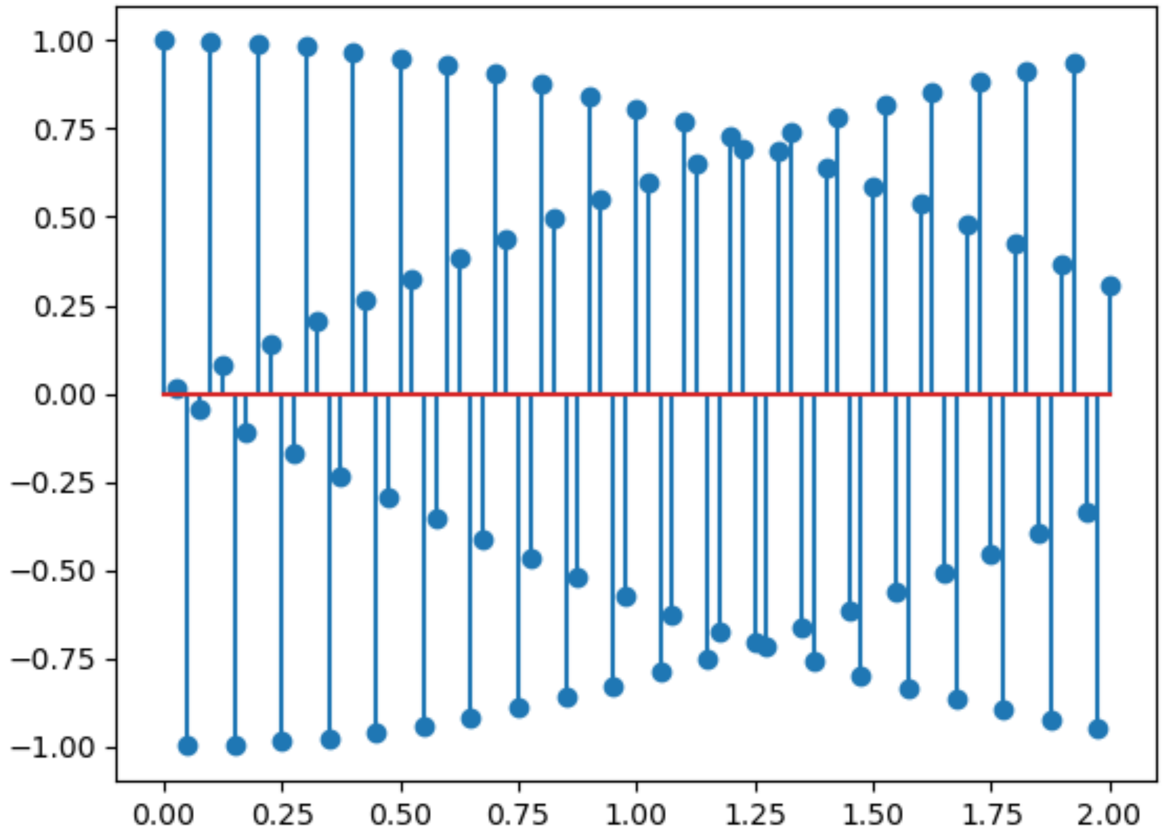
In [1]:
```
using PyPlot, DSP;
```

Decimação

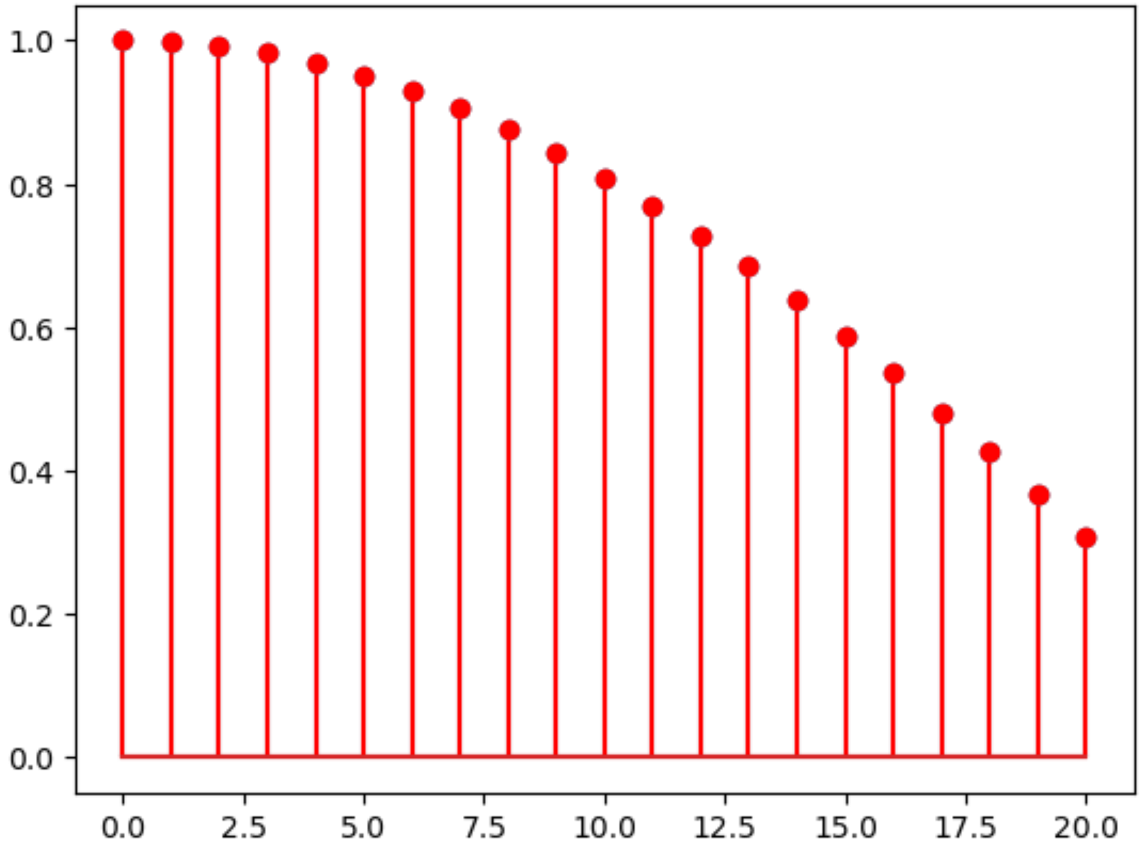
In [2]:
```
n=0:20
fa = 10_000
Ta = 1/fa
f0 = 4_000
xa = cos.(2π*f0*n*Ta);
stem(n*Ta*1000,xa);
```



In [3]:
```
m=0:80
fb = 40_000
Tb = 1/fb
f0 = 9_900
xb = cos.(2π*f0*m*Tb);
stem(m*Tb*1000,xb);
```



In [4]:
```
stem(xb[1:4:end]);
xc = cos.(2π*100*n*Ta)
stem(n,xc, "r", markerfmt="or");
```

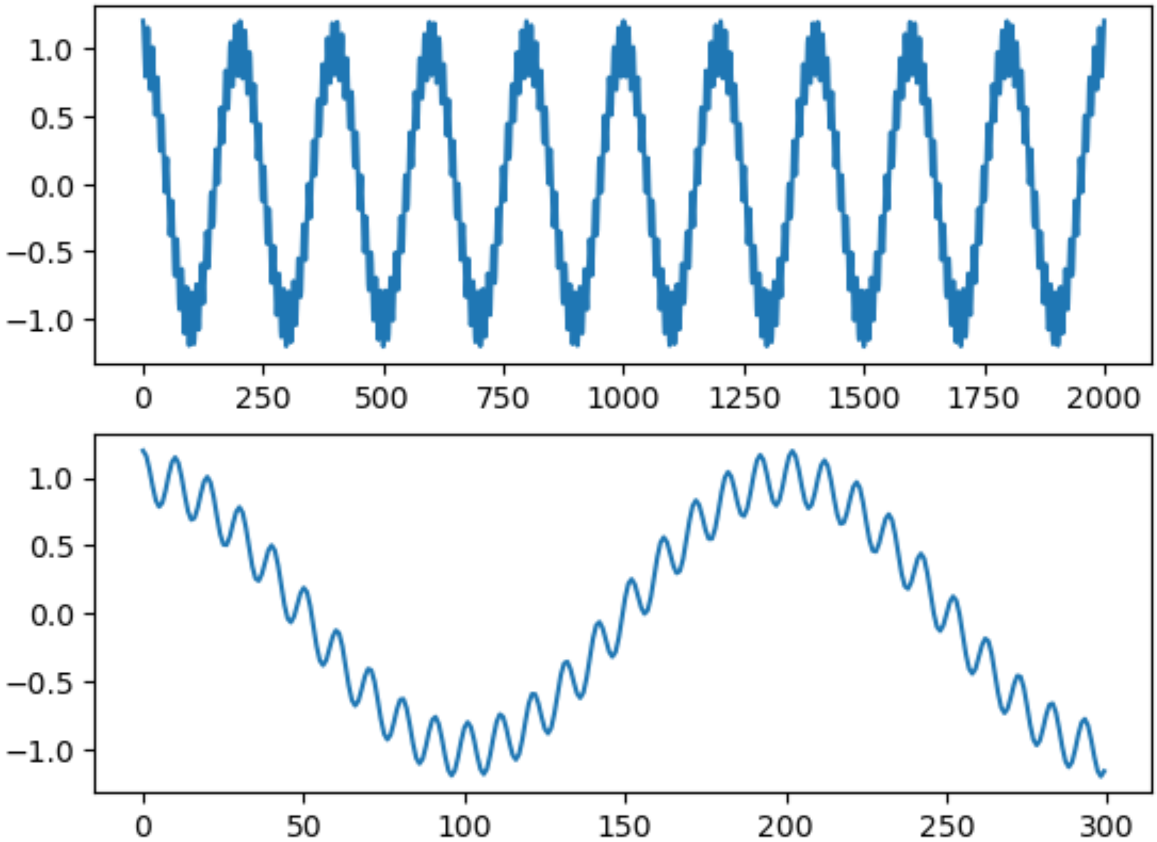


Decimação com rebatimento - filtragem

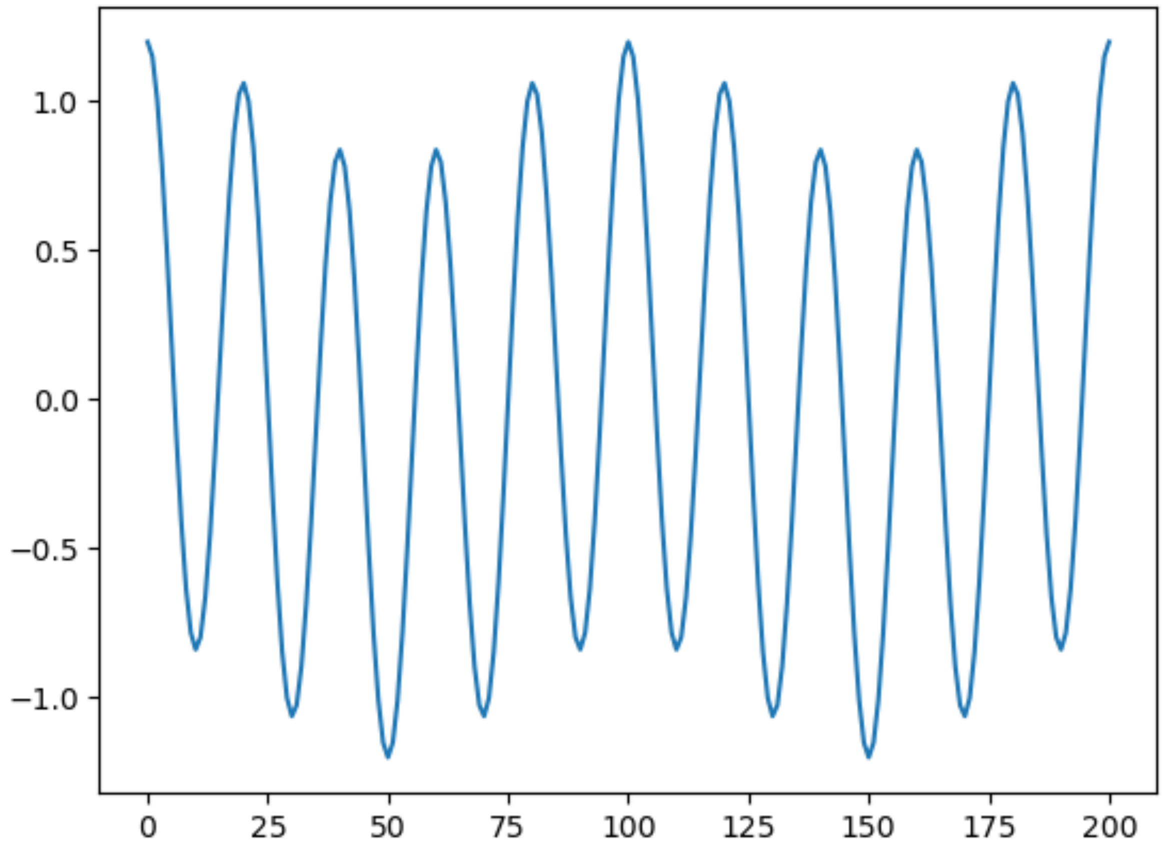
Considere o exemplo a seguir:

In [5]:

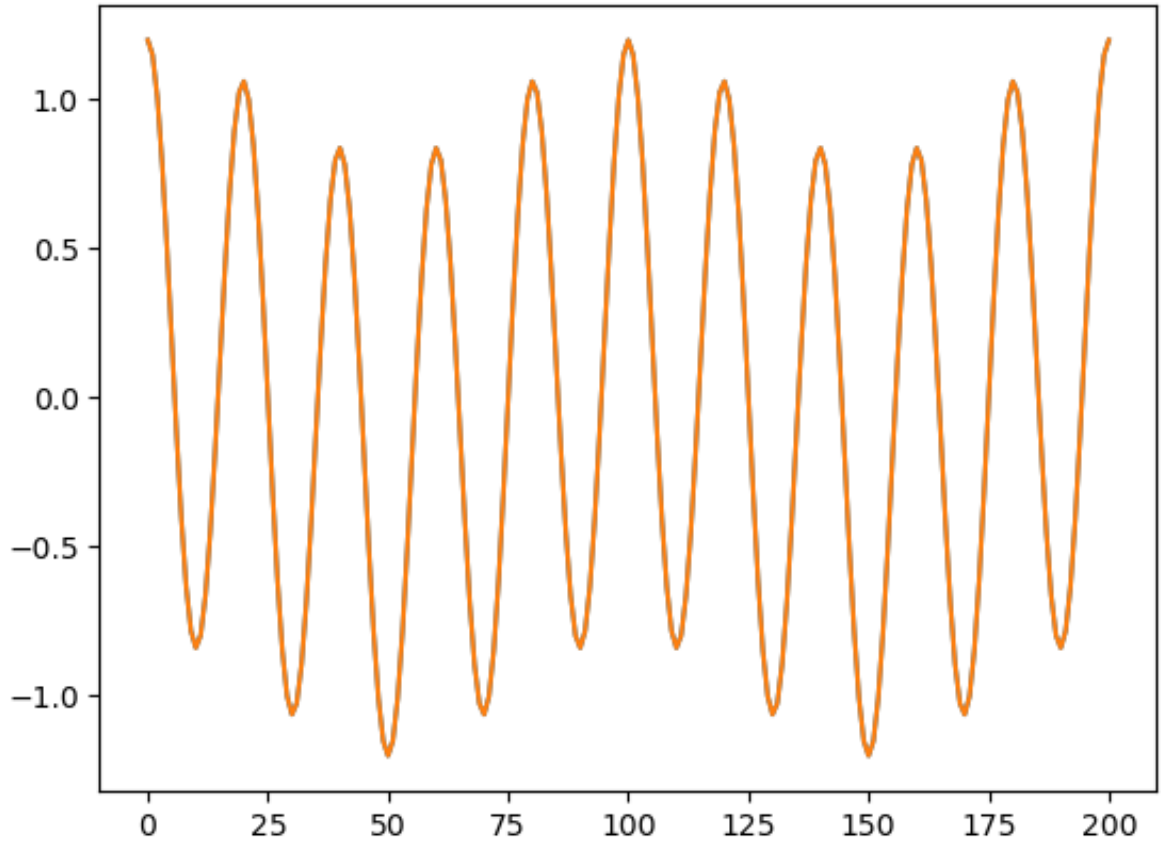
```
f0 = 50
f1 = 990
fa = 10_000
n = 0:10fa÷f0
x0 = cos.(2π*f0*n/fa)
x1 = 0.2cos.(2π*f1*n/fa)
xa = x0 + x1
subplot(211)
plot(n,xa);
subplot(212)
plot(n[1:300],xa[1:300]);
```



In [6]:
```
M = 10
xb = xa[1:M:end]
m=0:length(xb)-1
plot(m, xb);
```



In [7]:
```
plot(m,xb, m, cos.(2π*f0*m*M/fa)+0.2cos.(2π*10*m*M/fa));
```



In [8]:
```
fp = 2*f0/fa
```

Out[8]: 0.01

In [9]:
```
fr = 2*f1/fa
```

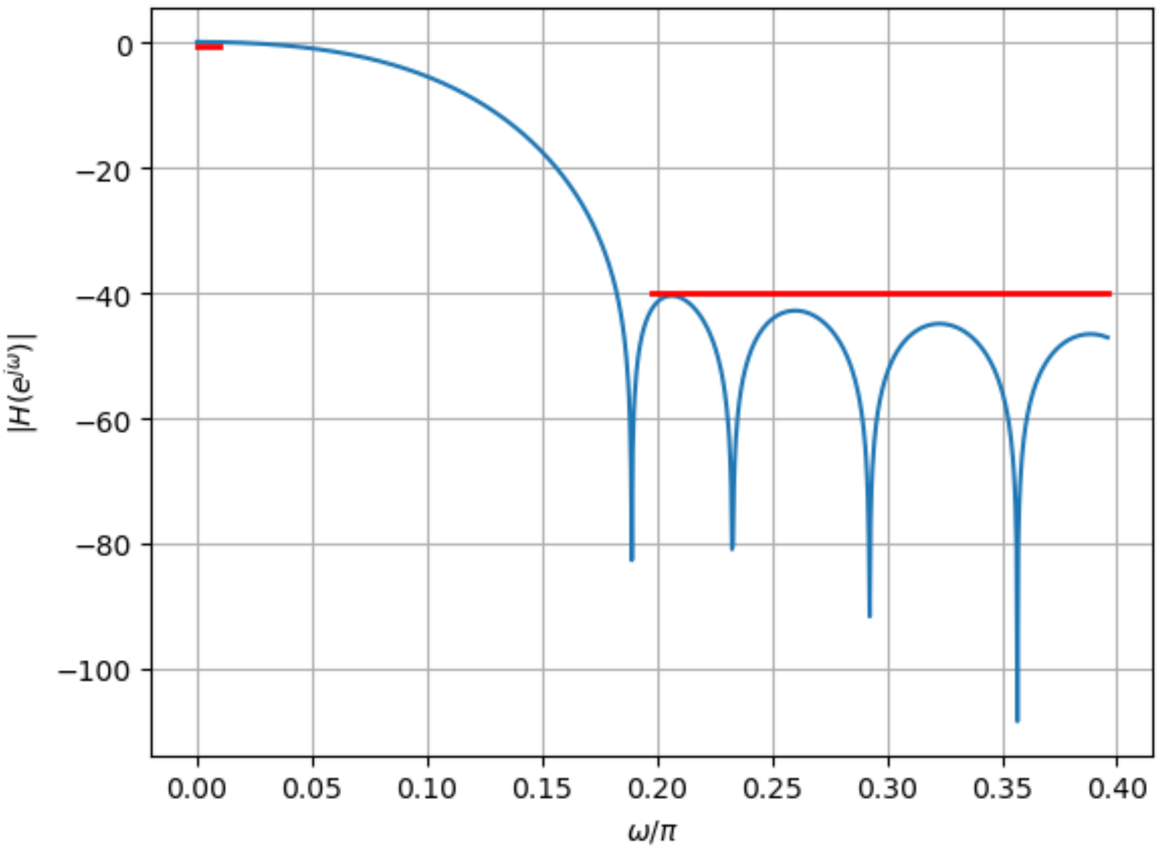
Out[9]: 0.198

In [10]:
```
include("/Users/vitor/docs/cursos/Julia/kaiser.jl")
```

Out[10]: filtrokaiser

In [11]:
```
δp = 0.05
δr = 0.01

h = PolynomialRatio(filtrokaiser(π*fp, π*fr, δp, δr), [1])
Nf = length(h.b)
h = PolynomialRatio(filtrokaiser(π*fp, π*fr, δp, δr, Nf+4), [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[11]: 25

In [12]:
```
A = -20*log10(min(δp, δr))
```

Out[12]: 40.0

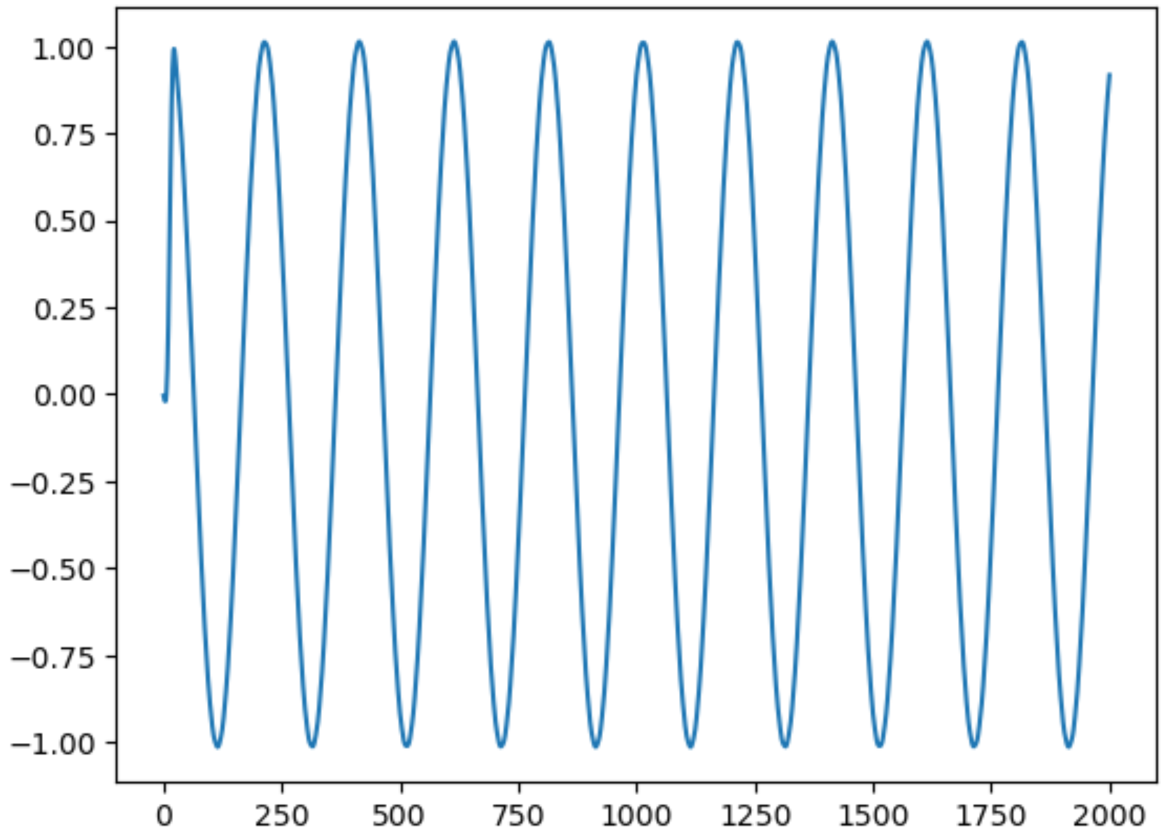
In [13]:
```
Nf = (A-8)/(2.285*(π*fr-π*fp))+1
```

Out[13]: 24.711337487502448

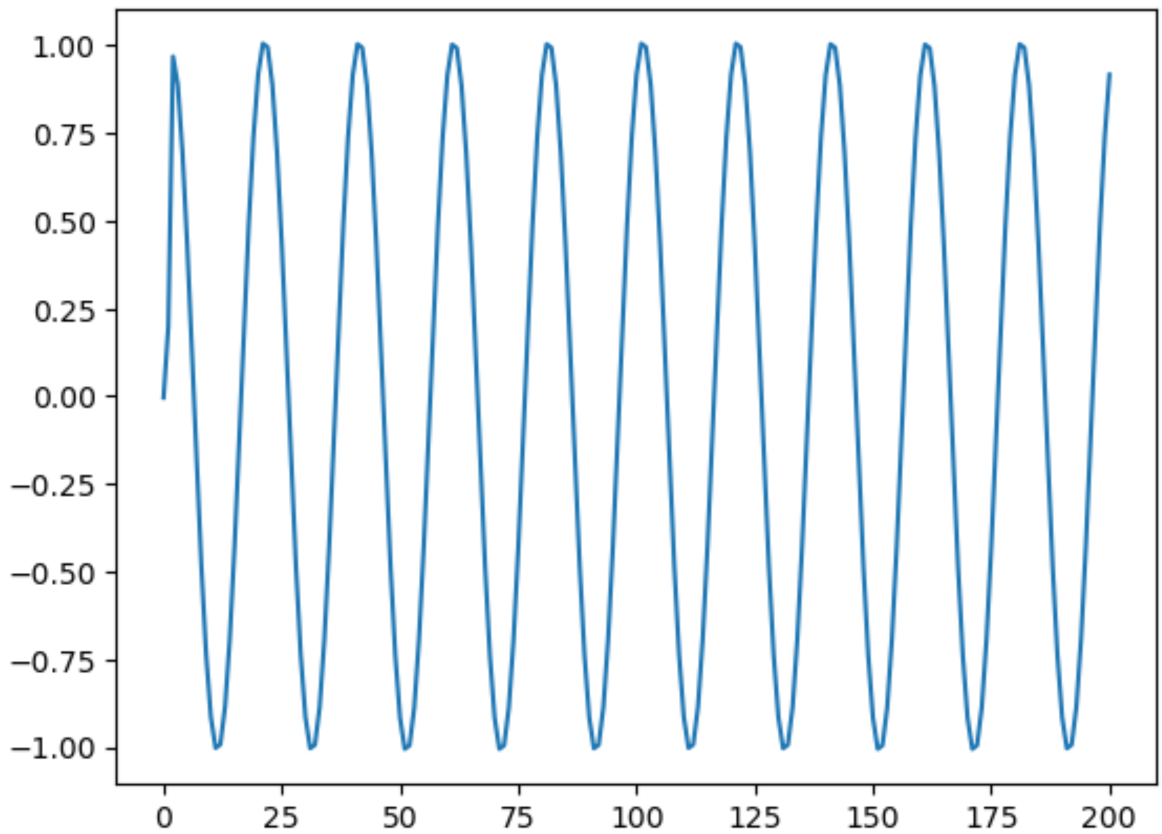
In [14]:
```
β = 0.5842(A-21)^0.4 + 0.07886(A-21)
```

Out[14]: 3.3953210522614574

In [15]:
```
xf = filt(h, xa)
plot(n, xf);
```

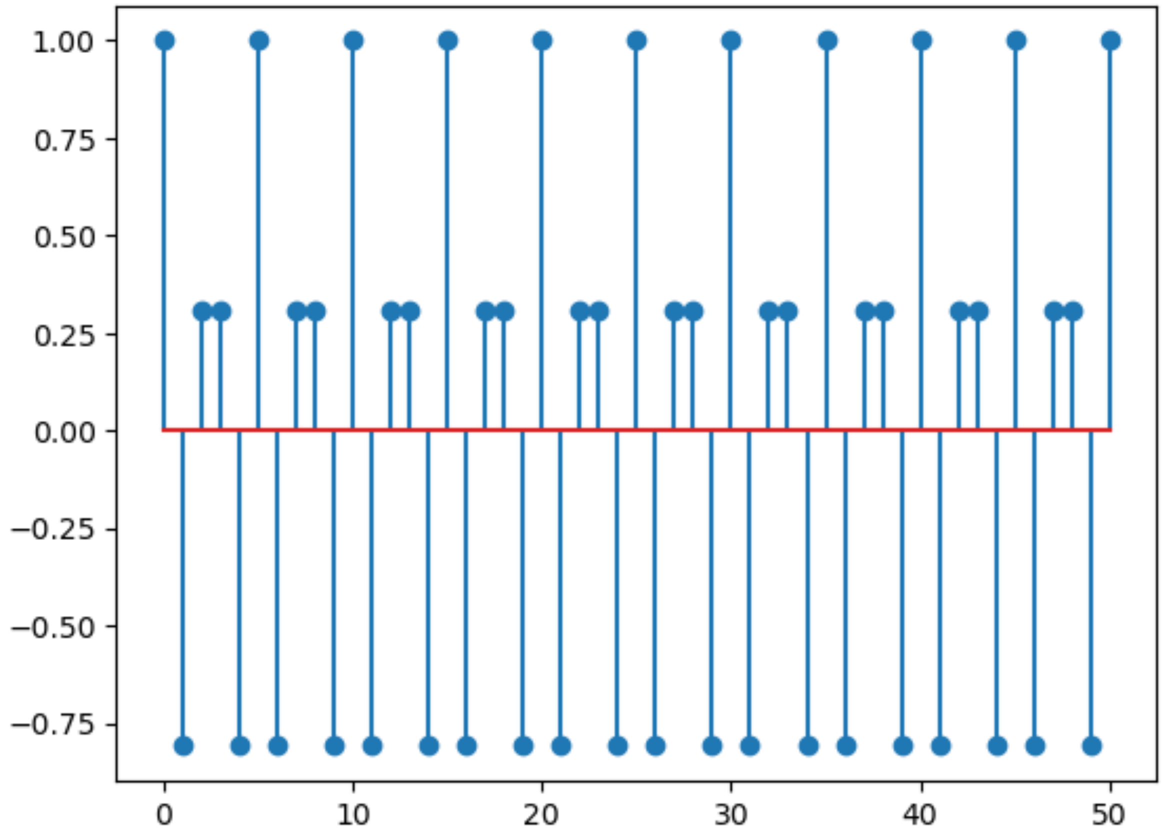


In [16]:
```
xfb = xf[1:M:end]
plot(m, xfb);
```

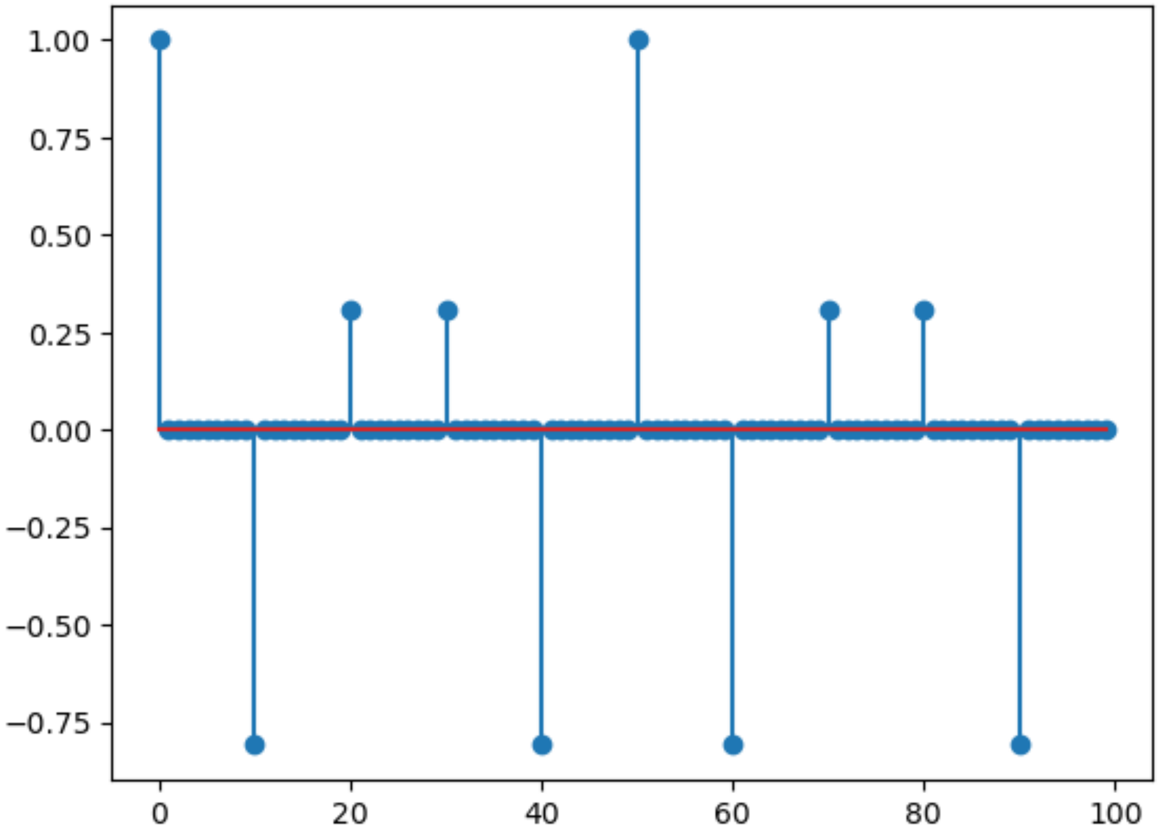


Interpolador

In [17]:
```
f0i = 4_000
xc = cos.(2π*f0i*n/fa)
stem(n[1:51], xc[1:51]);
```



In [18]:
```
L = 10
xci = zeros(L*length(xc))
xci[1:L:end] = xc
mi = 0:length(xci)-1
stem(mi[1:100], xci[1:100]);
```



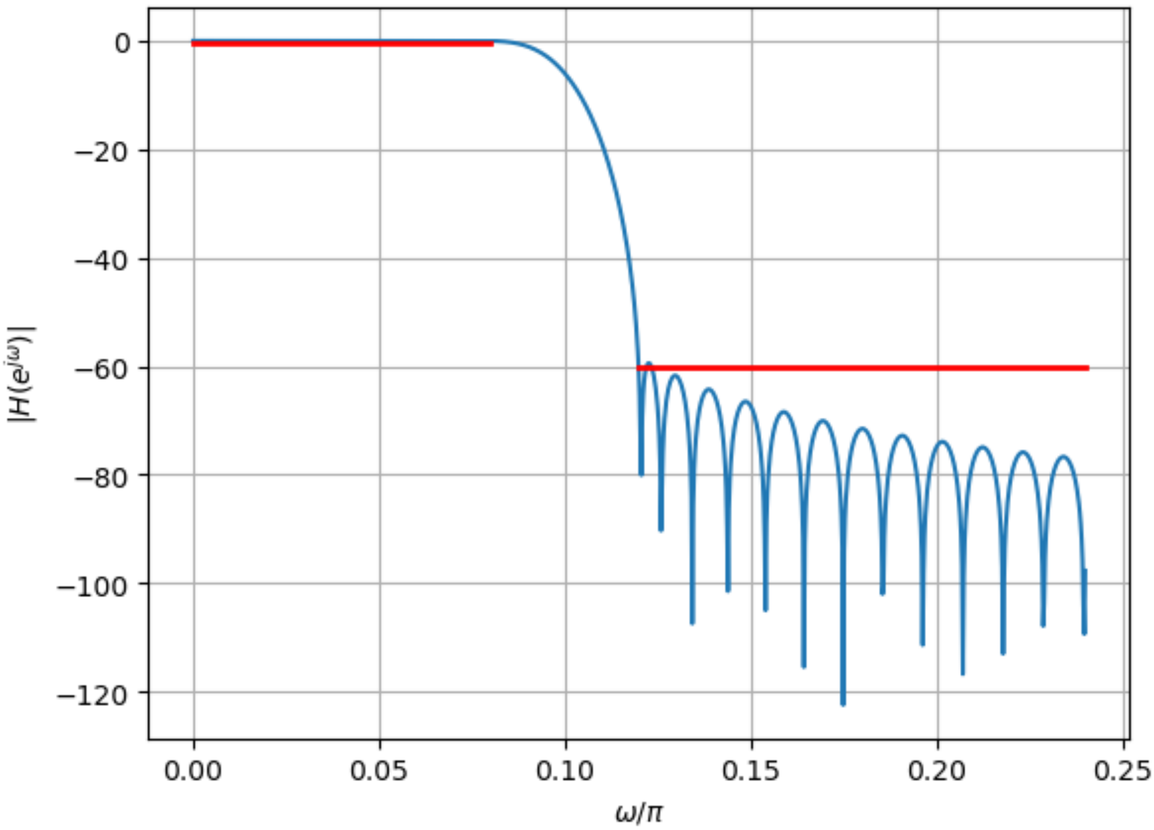
In [19]:
```
fpi = 2*f0i/(L*fa)
```

Out[19]:
```
0.08
```

In [20]:
```
fri = 1/L + (1/L - fpi)
```

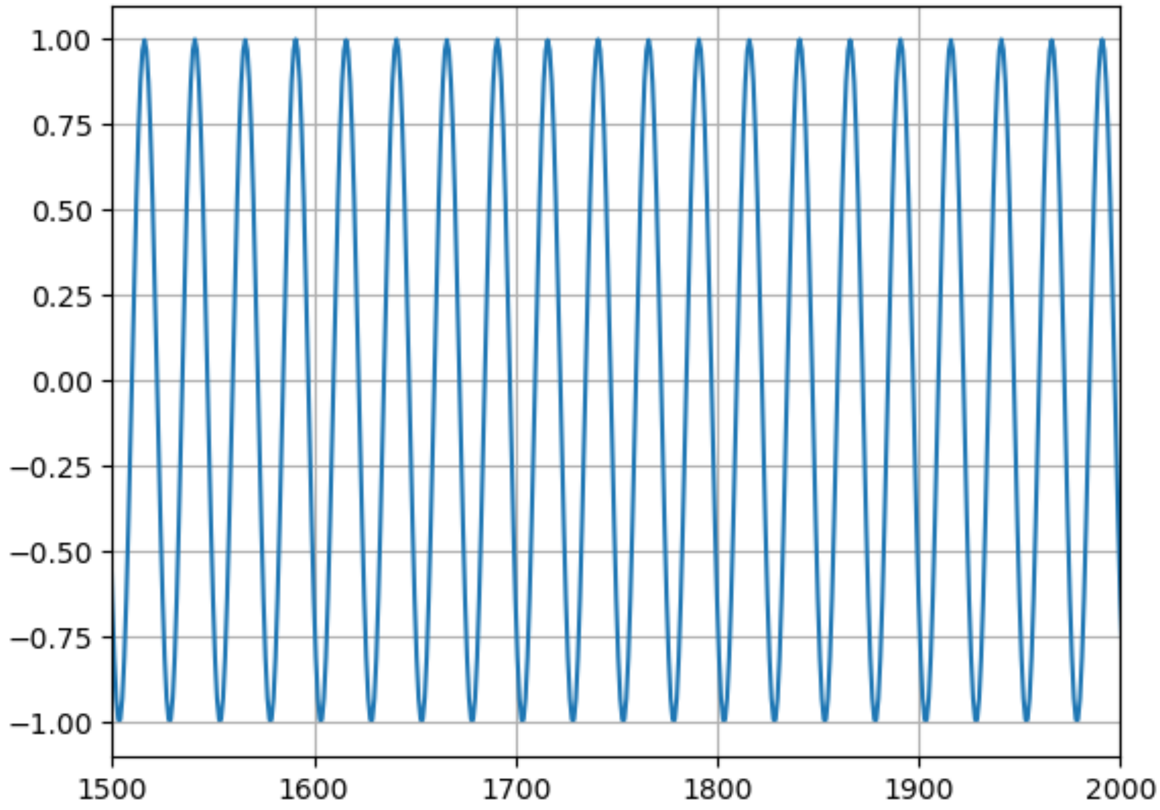
Out[20]:
```
0.12000000000000001
```

In [21]:
```
δpi = 0.05
δri = 0.001 # Tente também 0.001
hi = PolynomialRatio(filtrokaiser(π*fpi, π*fri, δpi, δri), [1])
Nfi = length(hi.b)
#hi = PolynomialRatio(filtrokaiser(π*fp, π*fr, δp, δr, Nfi+120), [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
```



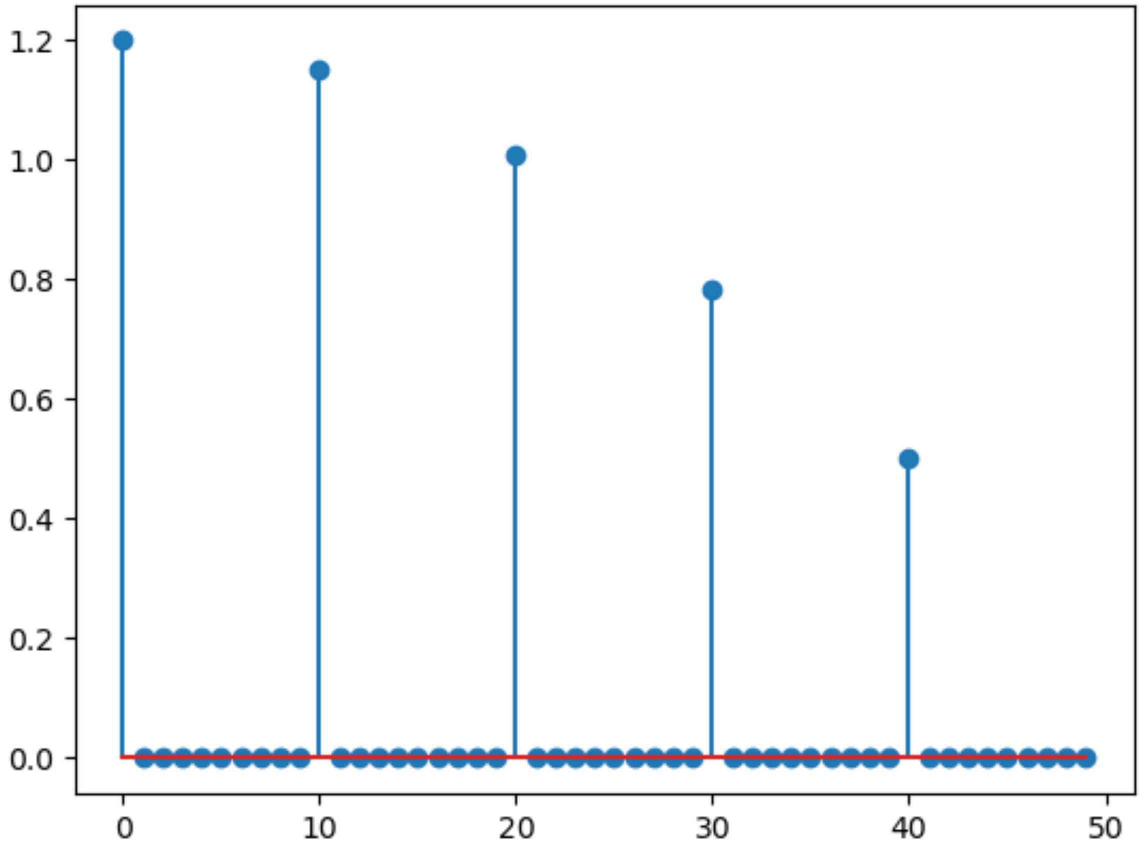
Out[21]: 183

In [22]:
```
xd = L*filt(hi, xci)
plot(mi, xd)
axis([1500, 2000, -1.1, 1.1])
grid();
```

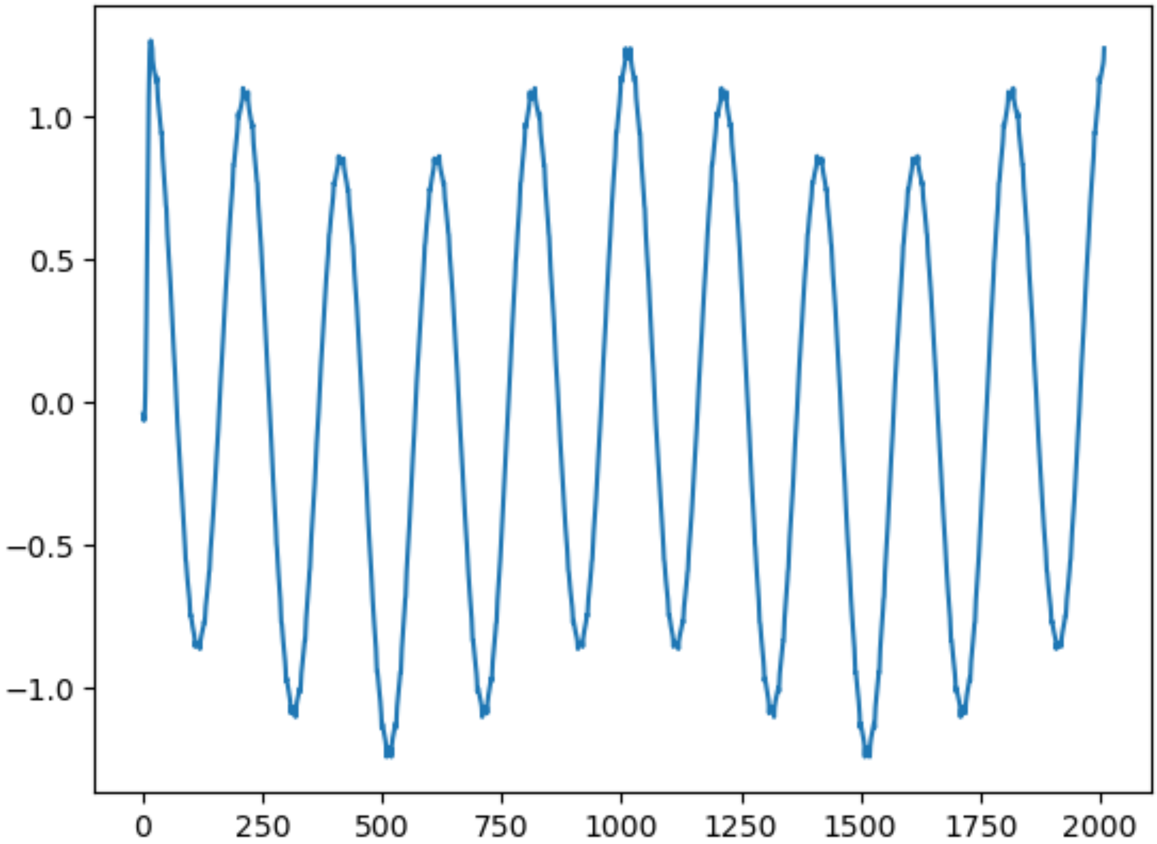


In [23]:

```
xbi = zeros(L*length(xb))
xbi[1:L:end] = xb
mi2 = 0:length(xbi)-1
stem(mi2[1:50], xbi[1:50]);
```



In [24]:
```
xbif = L*filt(h, xbi)
plot(mi2, xbif);
```



In [ ]: