Formatação de feixe com sinais reais

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

In [143]:
```
# Direções dos sinais
θ1=40
θ2=-20 #-14.22
# Direção de visada do arranjo
saida = :2
θ0 = eval(Symbol("θ", saida))
```

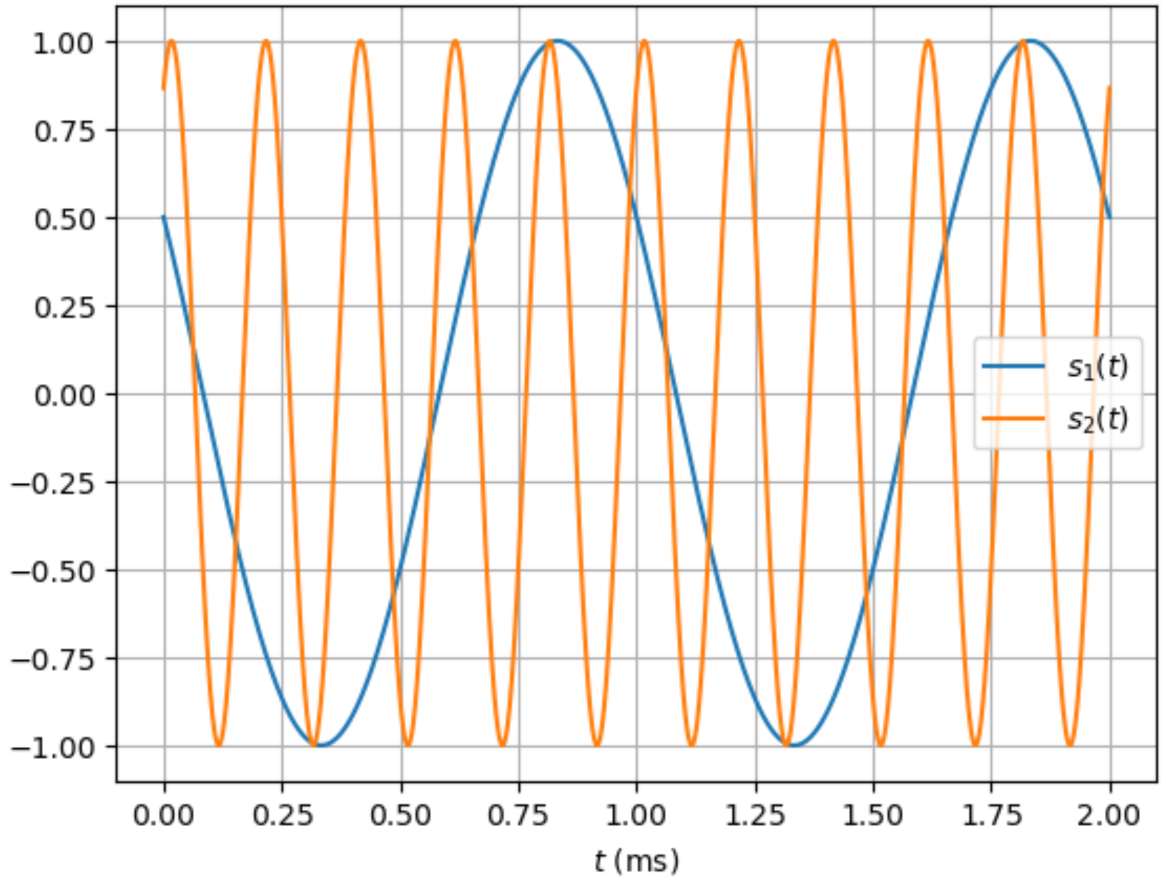
Out[143]:
```
-20
```

In [124]:
```
M=9
Ω0=2π*1e5
Ω1=2π*1000
Ω2=2π*5000
Ta=1e-6
c=3e8
λ=2π*c/Ω0
d=λ/2
m=0:M-1;
d
```

Out[124]:
```
1500.0
```

Os sinais chegando das direções θ_1 e θ_2 são

In [125]:
```
t=0:Ta:0.002
Nt=length(t)
# Sinal na direção θ1
s1 = cos.(Ω1*t .+ π/3)
# Sinal na direção θ2
s2 = cos.(Ω2*t .- π/6)
plot(1000t, s1, label = L"s_1(t)")
plot(1000t, s2, label = L"s_2(t)")
grid();legend();xlabel(L"$t$ (ms)");
```



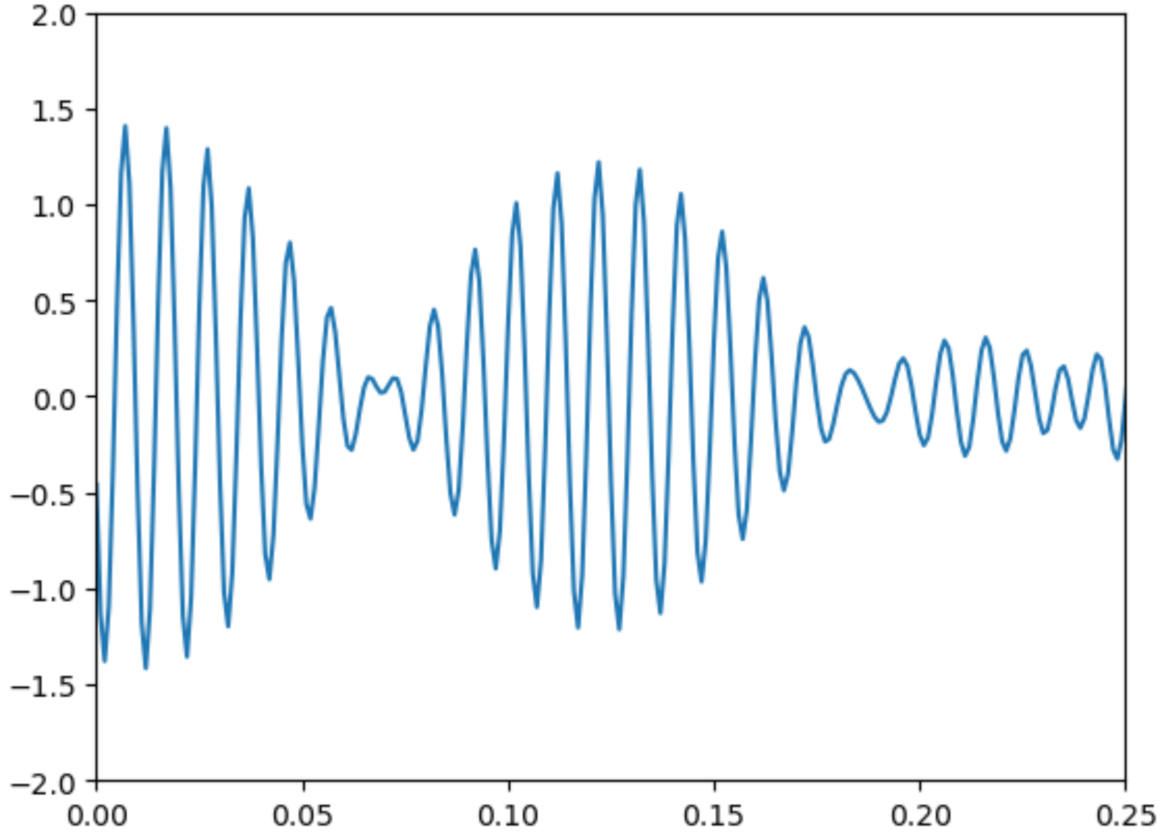
Agora vamos gerar os sinais recebidos em cada antena. Os atrasos de cada sinal em cada antena serão

In [126]:
```
τ1=(d*sind(θ1)/c)*m
τ2=(d*sind(θ2)/c)*m;
collect(1000000τ2)
```

Out[126]:
```
9-element Array{Float64,1}:
   0.0
  -1.7101007166283437
  -3.4202014332566875
  -5.130302149885031
  -6.840402866513375
  -8.550503583141719
 -10.260604299770062
 -11.970705016398405
 -13.68080573302675
```

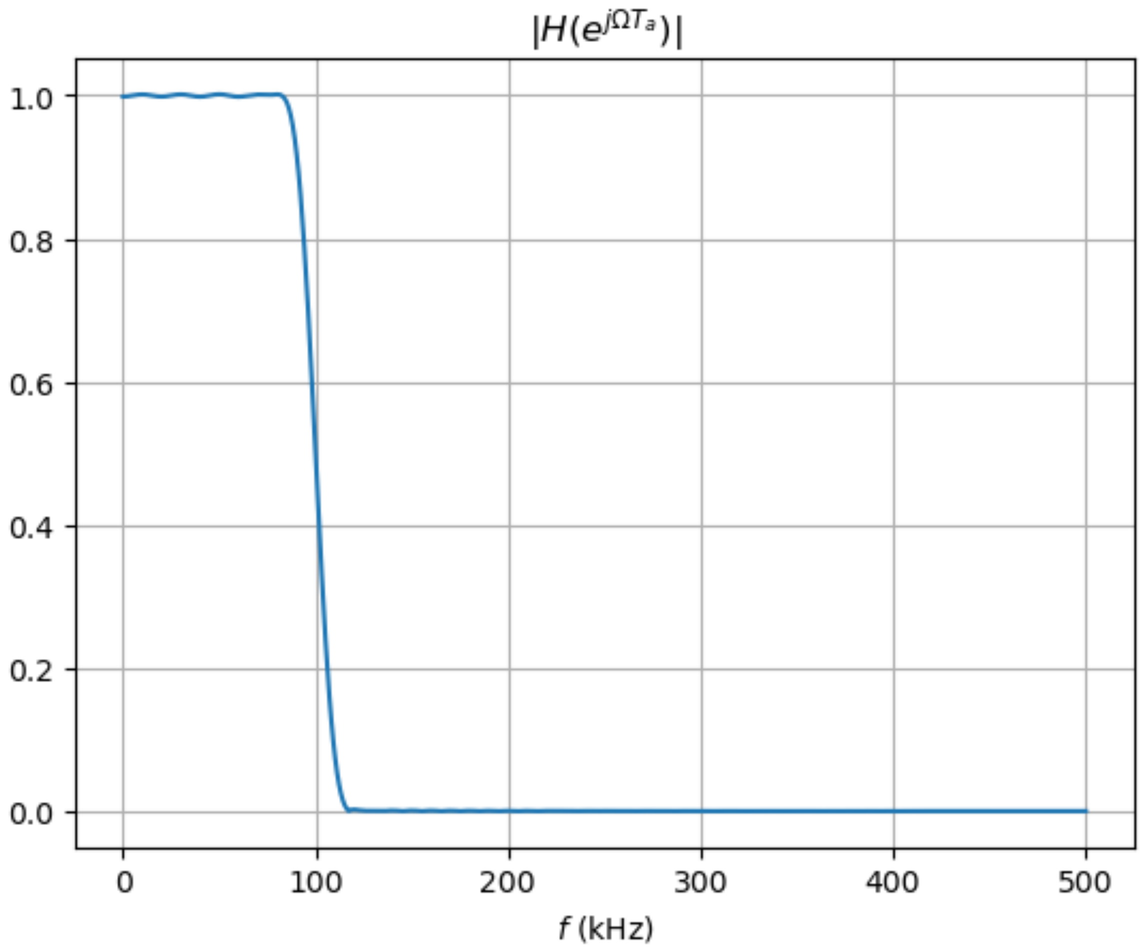
In [127]:
```
# Sinais nas antenas
x=zeros(Nt,M)

for i=1:M
    x[:,i]=s1 .* cos.(Ω0*(t .+ τ1[i])) + s2 .* cos.(Ω0*(t .+ τ2[i]))
)
end
```

In [128]:
```
plot(1000t, x[:,5]);
axis([0, 0.25, -2, 2]);
```



Agora vamos ver como recuperar os sinais: Primeiro, vamos projetar um filtro passa-baixas

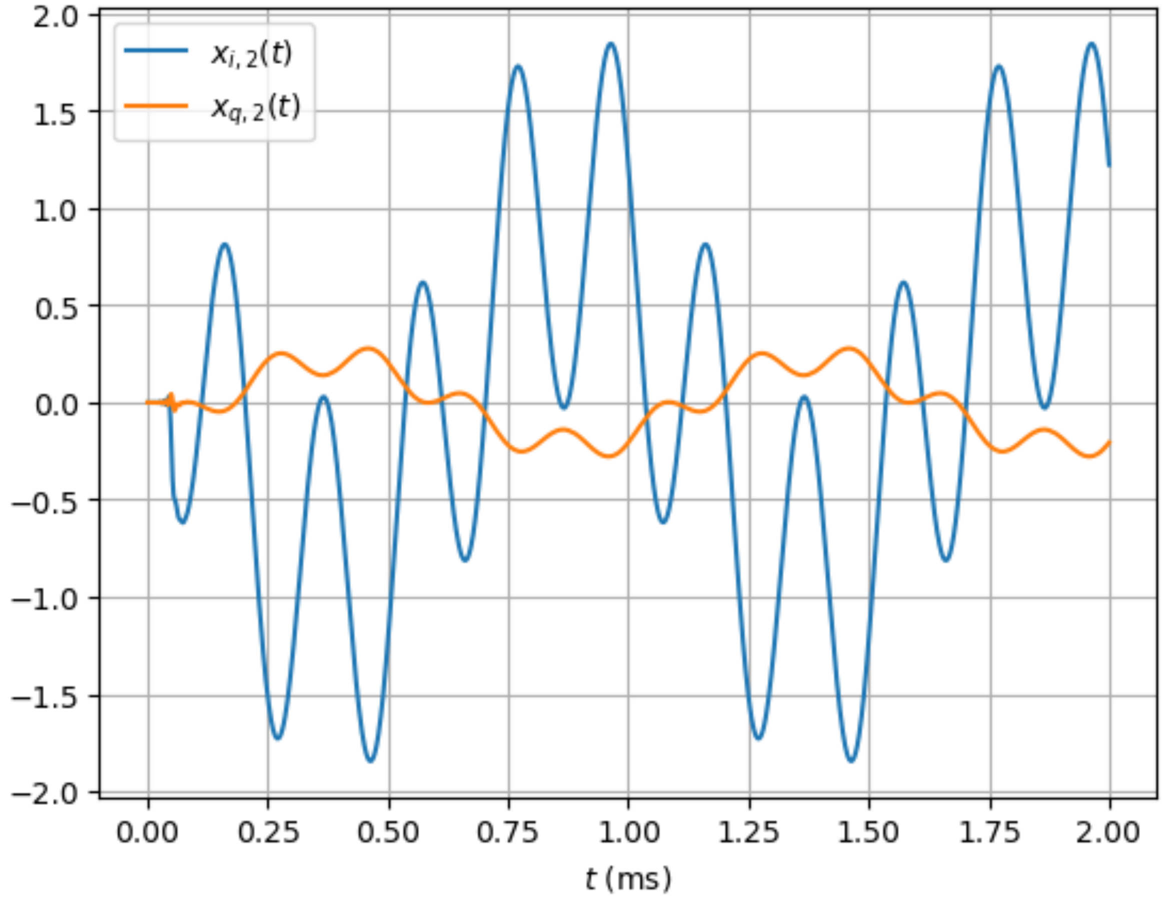
In [129]:
```
fc=Ω0*Ta/π
Nf=101
Lf=(Nf-1)/2
nf=0:Nf-1
h=PolynomialRatio(fc*sinc.(fc*(nf .- Lf)) .* hamming(Nf),[1])
ω=range(0,π,length=500)
H=freqz(h,ω)
figure(3)
clf()
plot(1e-3ω/(2π*Ta), abs.(H))
grid();xlabel(L"$f$ (kHz)")
title(L"$|H(e^{j\Omega T_a})|$");
```



Agora calculamos os sinais em fase, em quadratura, e o "sinal analítico", quer dizer, o sinal complexo equivalente

In [130]:
```
xi=zeros(Nt,M)
xq=zeros(Nt,M)
xa=complex(zeros(Nt,M))
for i=1:M
    xi[:,i] .= 2 .* (x[:,i] .* cos.(Ω0*t))
    xq[:,i] .= -2 .* (x[:,i]) .* sin.(Ω0*t)
    # Sinais em fase e em quadratura demodulados
    xi[:,i] = filt(h,xi[:,i])
    xq[:,i] = filt(h,xq[:,i])
    # Sinal analítico
    xa[:,i] .= xi[:,i] .+ im .* xq[:,i]
end
```

In [131]:
```
plot(1000t, xi[:,4], label = L"x_{i,2}(t)")
plot(1000t, xq[:,4], label = L"x_{q, 2}(t)")
legend();grid();
xlabel(L"$t$ (ms)");
```



Agora vamos projetar o formatador de feixe:

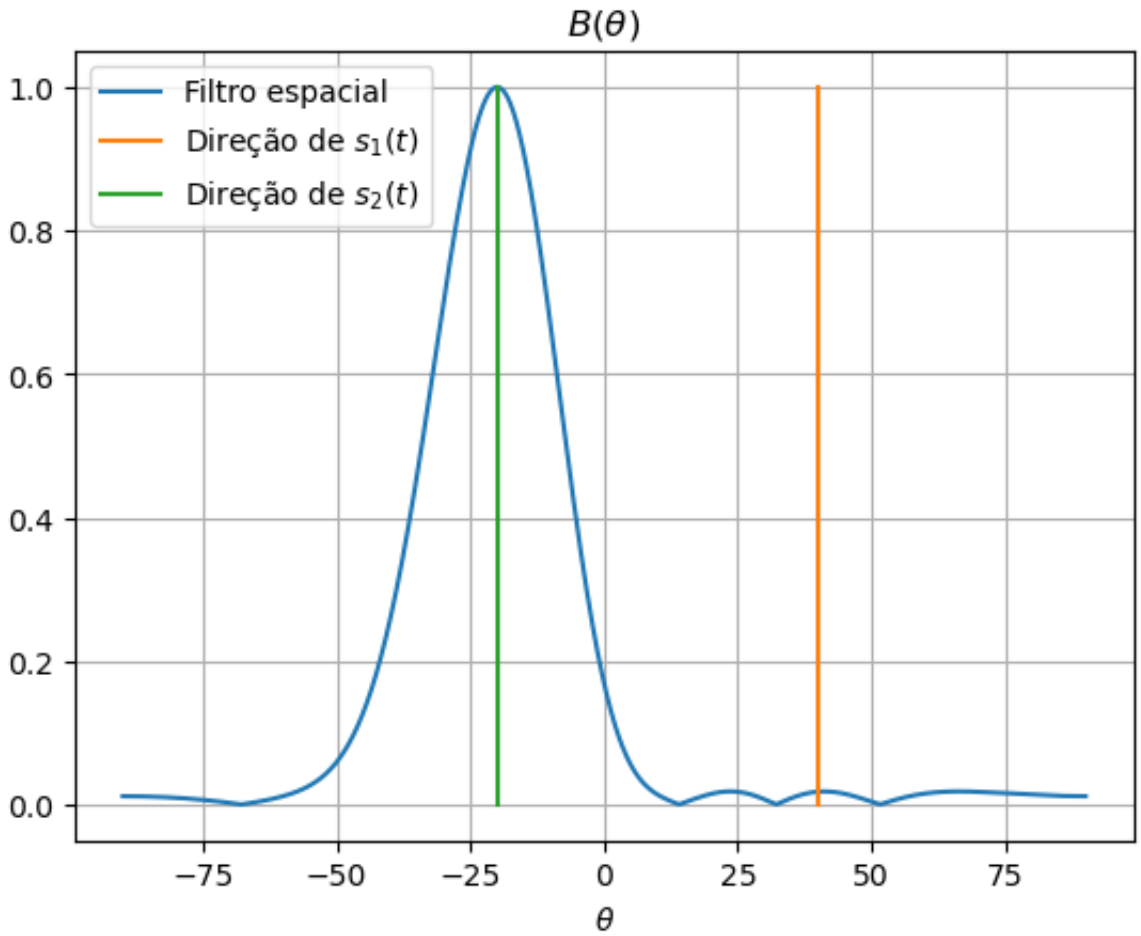
In [144]:
```
v0=exp.((0:M-1)*im*Ω0*d*sind(θ0)/c);
```

In [145]:
```
θ=-90:0.01:90
N=length(θ)
janHamming = true
if janHamming
    w=(v0.*hamming(M))/sum(hamming(M))
else
    w=v0/M
end
function B(w, m, Ω0, d, c, θ)
    N = length(θ)
    Bham=complex(zeros(N))
    for i=1:N
        xm=exp.(m*im*Ω0*d*sind(θ[i])/c)
        Bham[i]=w⋅xm
    end
    return Bham
end
Bham = B(w, m, Ω0, d, c, θ);
```

In [146]:
```
println("|B(θ1, θ0)| = ", abs(B(w, m, Ω0, d, c, [θ1])[1]), "\n|B(θ2, θ0)| = ", abs(B(w, m, Ω0, d, c, [θ2])[1]))
```

```
|B(θ1, θ0)| = 0.017850431615891293
|B(θ2, θ0)| = 1.0
```

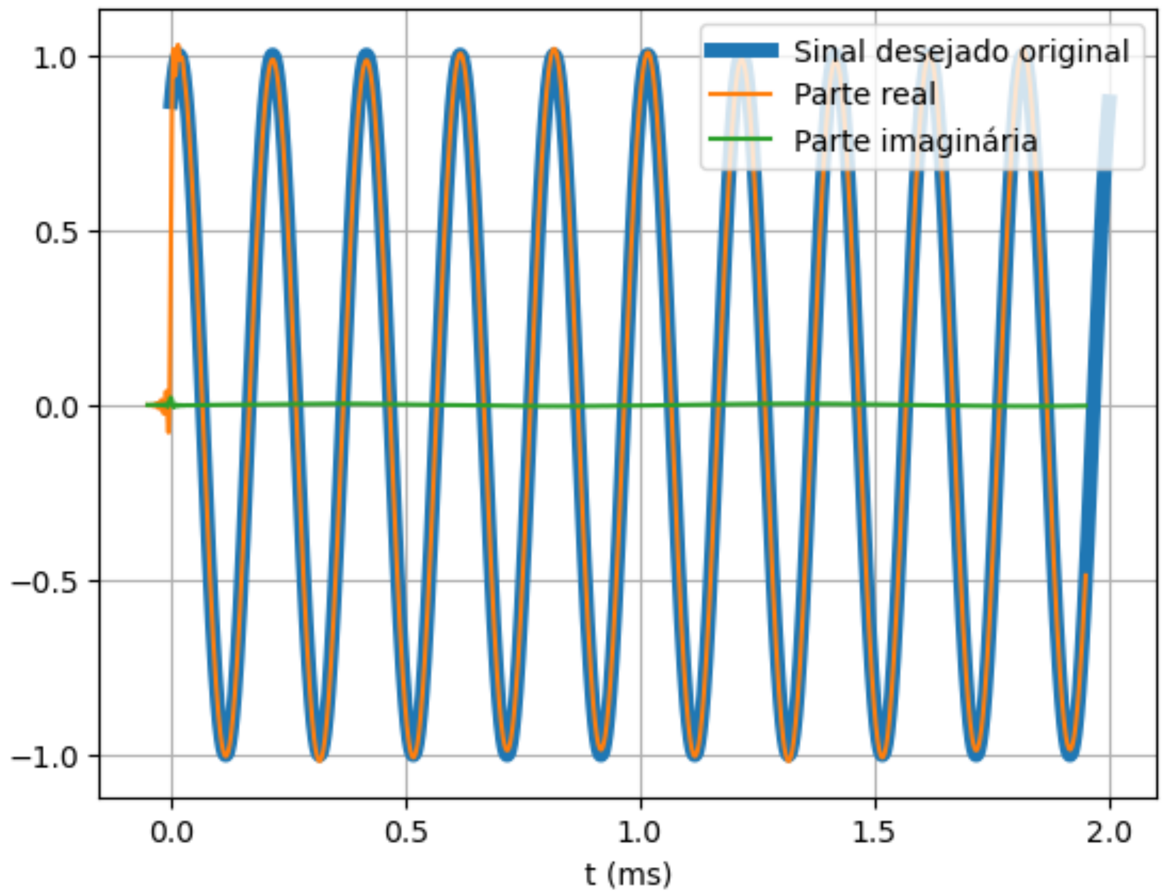
In [147]:
```
plot(θ,abs.(Bham),label="Filtro espacial")
xlabel(L"\theta")
legend();grid()
plot([θ1, θ1],[0,1],label = L"Direção de $s_1(t)$")
plot([θ2, θ2],[0,1],label = L"Direção de $s_2(t)$")
legend()
title(L"$B(\theta)$");
#axis([-15, -14, 0, 0.2]);
```



Agora vamos aplicar o filtro espacial aos sinais recebidos nas antenas:

In [148]:
```
y = (w' * transpose(xa))
y=y[:];
```

In [149]:
```
plot(1000t,eval(Symbol("s",saida)),label="Sinal desejado original",
LineWidth=5)
plot(1000(t .- Lf*Ta),real.(y),label="Parte real")
plot(1000(t .- Lf*Ta) ,imag.(y),label="Parte imaginária")
xlabel("t (ms)")
legend();grid()
```



In [ ]: