Laboratório de Sinais e Sistemas

Trabalhando com Funções

Considere a definição da senoide exponencialmente amortecida

$$f(t) = e^{-t}\cos(2\pi t).$$

```
f=@(t) exp(-t).*cos(2*pi*t);
```

Uma vez definida, f(t) pode ser calculada passando, simplesmente, os valores de entrada de interesse. Por exemplo,

```
t=0;
f(t)
```

```
ans = 1
```

determina f(t) para t = 0, confirmando o resultado unitário esperado. O mesmo resultado é obtido passando t = 0 diretamente,

```
f(0)
```

```
ans = 1
```

Entradas na forma de vetores permitem o cálculo de múltiplos valores simultaneamente. Considere a tarefa de traçar f(t) no intervalo ($-2 \leq t \leq 2$). O rascunho da função é fácil de ser visualizado: f(t) deve oscilar quatro vezes com um envelope de amortecimento. Como um gráfico detalhado é mais trabalhoso, gráficos gerados pelo MATLAB são uma alternativa atraente. Como os seguintes exemplos mostram, deve-se ter cuidado para garantir resultados confiáveis.

Suponha que o vetor t seja escolhido para incluir apenas os inteiros contidos em ($-2 \leq t \leq 2$), ou seja, [–2, −1, 0, 1, 2].

```
t=-2:2;
```

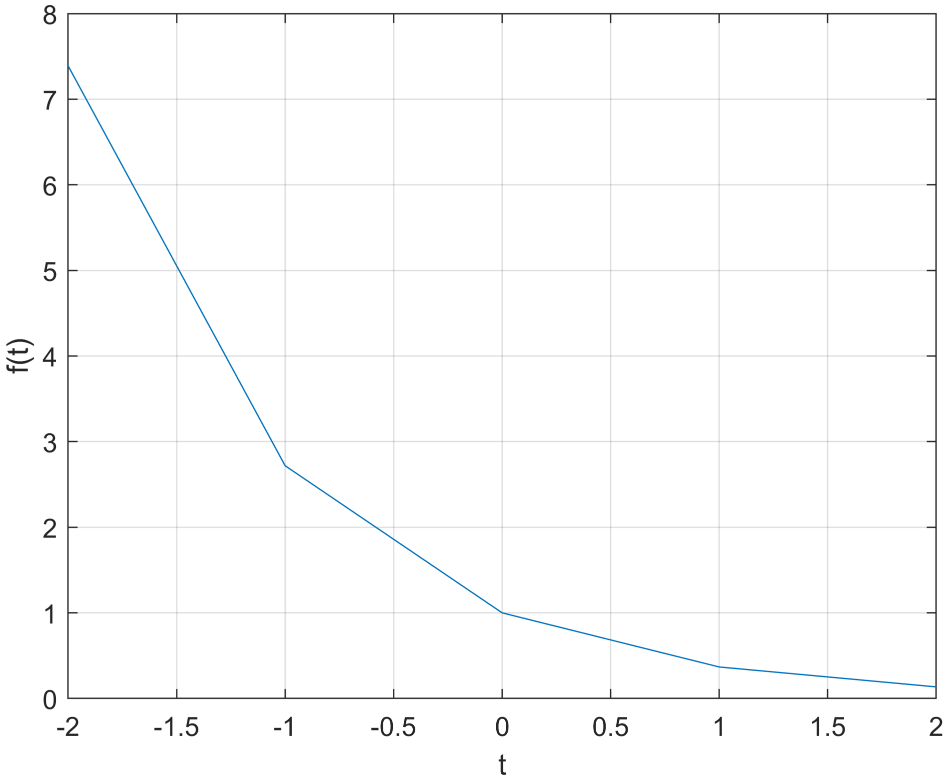
Este vetor de entrada é utilizado para determinarmos o vetor de saída.

```
f(t)
```

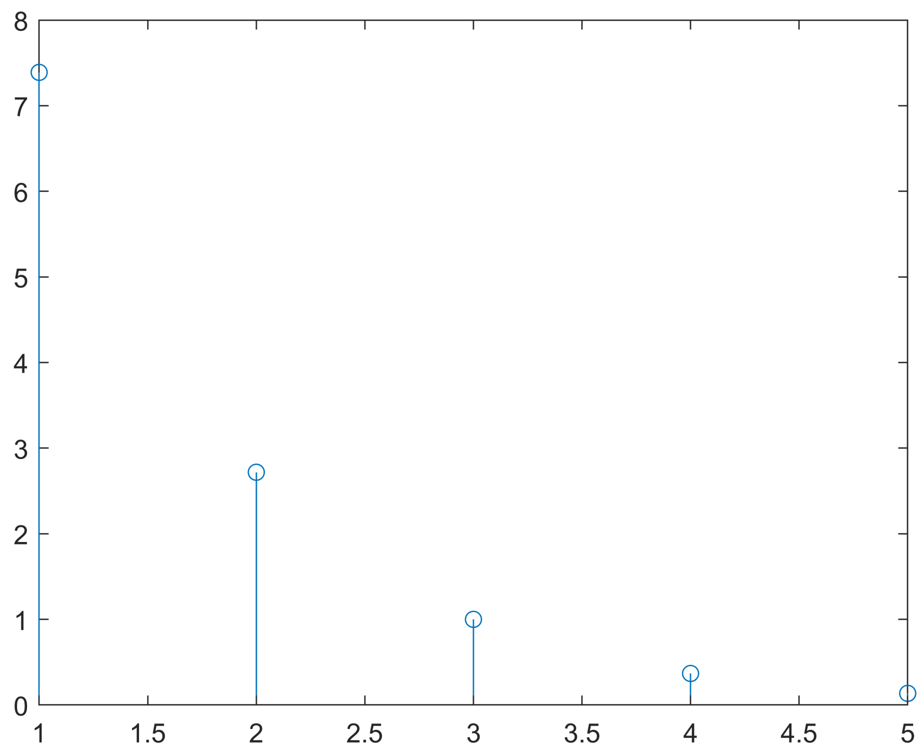
```
ans = 1×5
    7.3891    2.7183    1.0000    0.3679    0.1353
```

O comando plot traça o gráfico do resultado

```
plot(t,f(t))
xlabel('t');
ylabel('f(t)');grid;
```




```
stem(f(t))
```

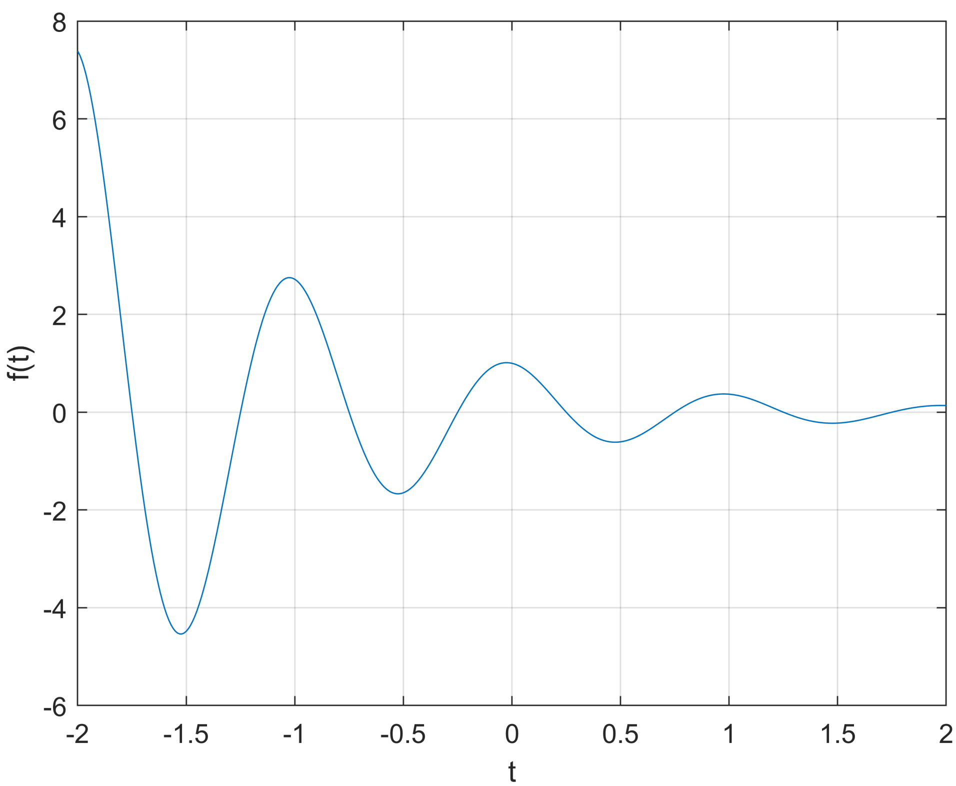


As linhas quadriculadas, inseridas com o comando grid, facilitam a visualização. Infelizmente, o gráfico não ilustra o comportamento oscilatório esperado. Mais pontos são necessários para representar adequadamente f(t). A questão é, então, quantos pontos são suficientes? Se poucos pontos forem escolhidos perde-se informação. Se muitos pontos forem escolhidos, perderemos memória e tempo. É necessário atingir um equilíbrio. Para funções oscilatórias, a utilização de 20 a 200 pontos por oscilação geralmente é adequada. Para o caso em estudo, t é escolhido para fornecer 100 pontos por oscilação.

```
t=-2:0.01:2;
```

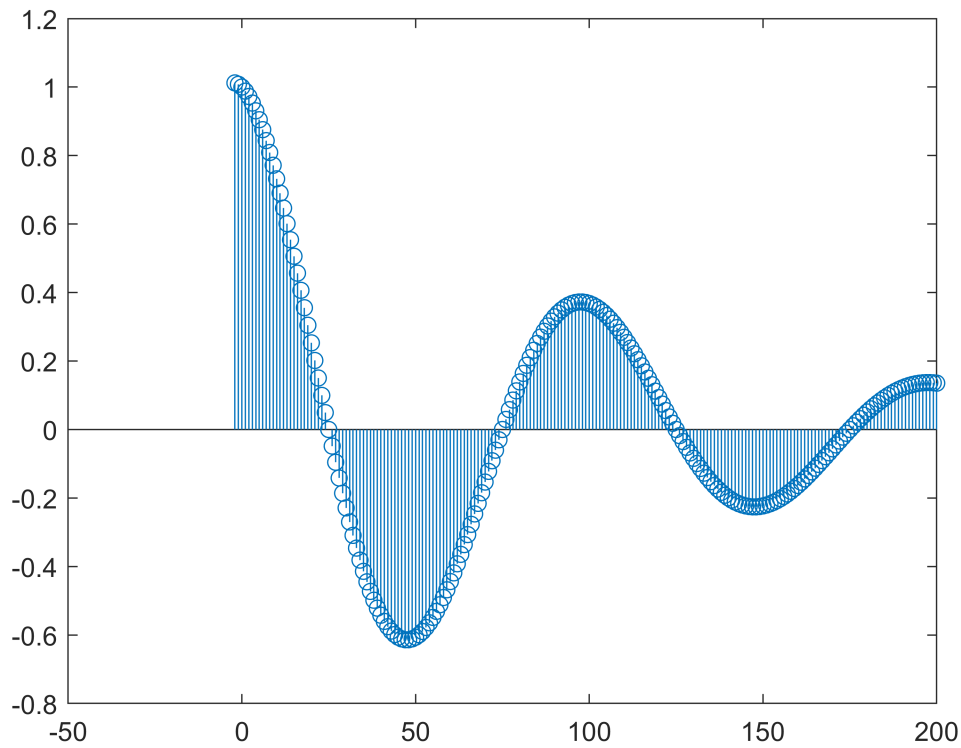
Novamente, a função é determinada e traçada.

```
plot(t,f(t));
xlabel('t');
ylabel('f(t)');
grid;
```



Como observado, o sinal de tempo discreto mostrado na figura após a função stem, não representada adequadamente o sinal. Para isso, devemos fazer algumas alterações, como segue

```
fd=@(n,Ts) exp(-n*Ts).*cos(2*pi*n*Ts);
n=-2:200;
stem(n,fd(n,0.01))
```

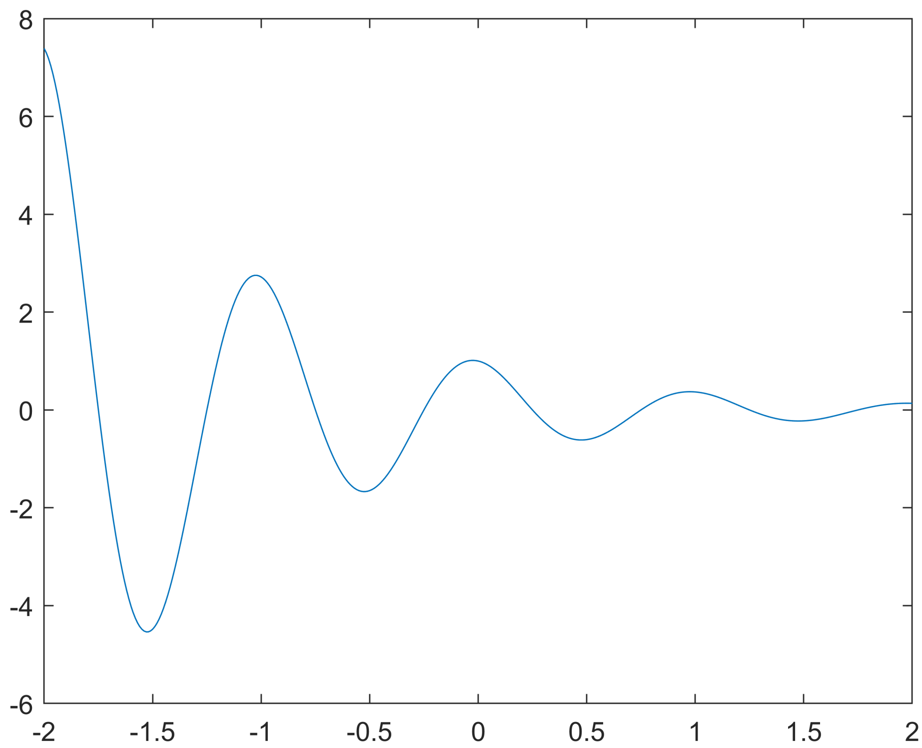


Podemos também não usar a função @, mas para isso somos obrigados a criar o vetor de tempo antes da linha de comando da função que desejamos plotar. O procedimento é descrito a seguir.

```
t = -2:0.01:2;
f = exp(-t).*cos(2*pi*t)
```

```
f = 1×401
    7.3891    7.3011    7.1856    7.0437    6.8763    6.6847    6.4701    6.2338 ···
```

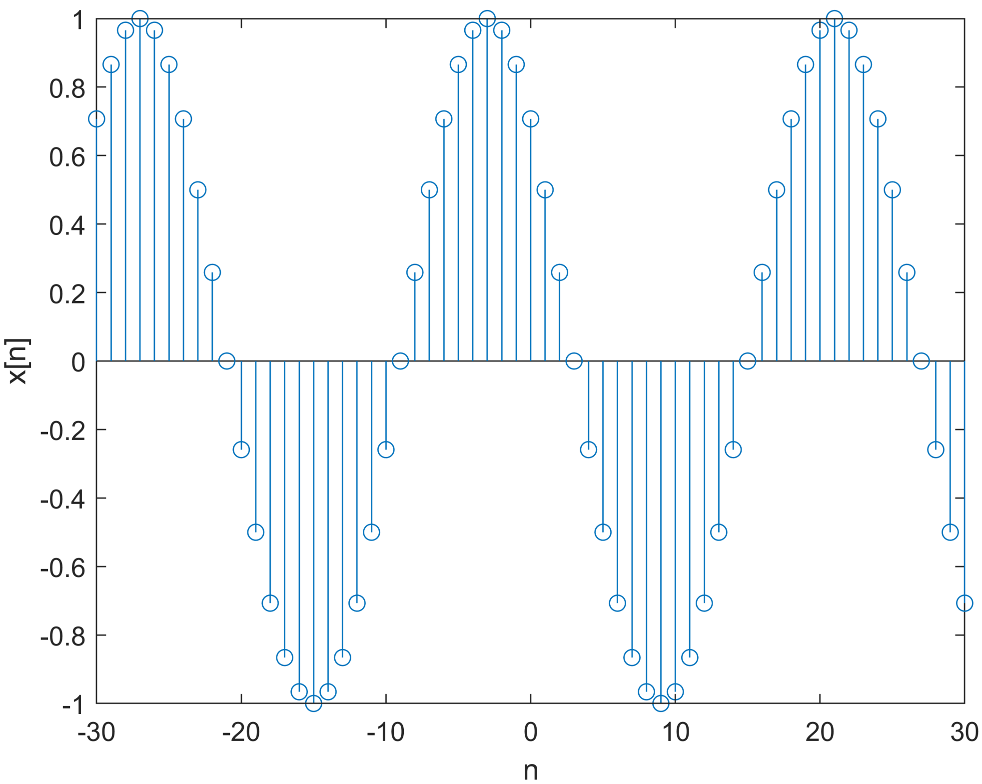
```
plot(t,f);
```



Agora vamos traçar a seguinte senoide discreta no tempo

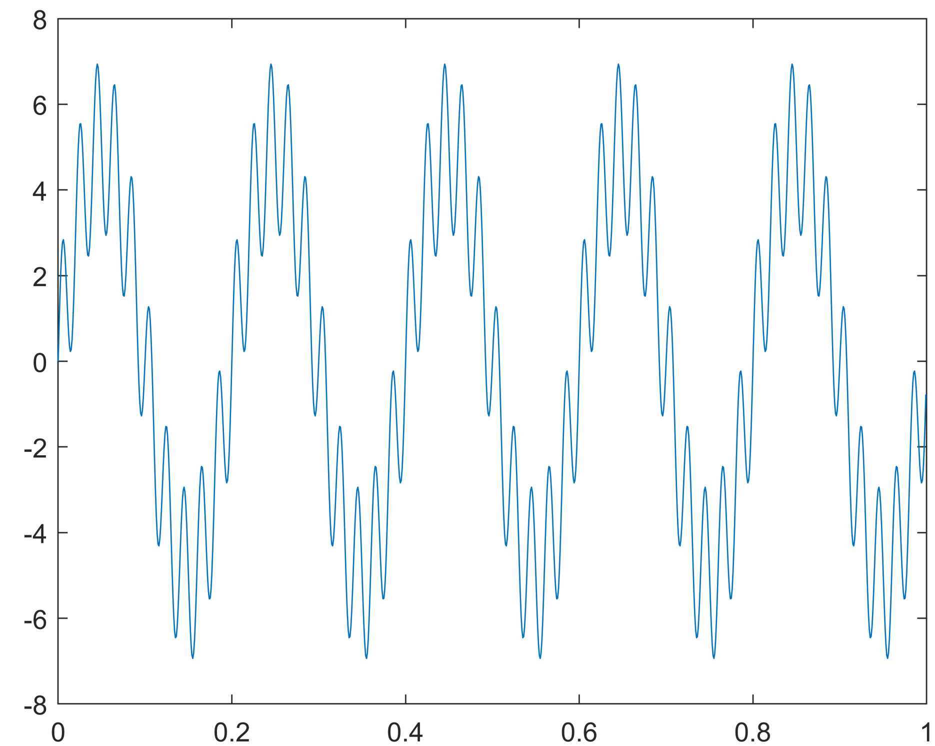
$$x[n] = \cos\left(\frac{\pi}{12}n + \frac{\pi}{4}\right)$$

```
clear all
n=-30:30;
x = cos(n*pi/12+pi/4);
figure(2)
stem(n,x);
xlabel('n');
ylabel('x[n]');
```

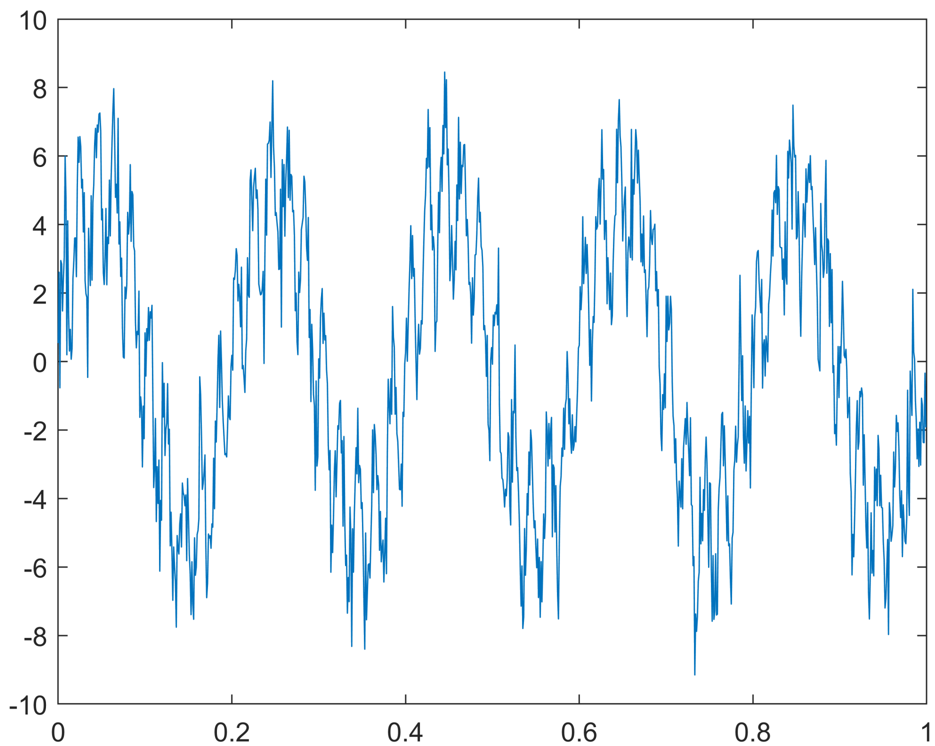


Agora vamos considerar um sinal senoidal no tempo continuo sinal $x(t) = 5\text{sen}(2\pi 5t) + 2\text{sen}(2\pi 50t)$, amostrado a $F_s = 1000\text{Hz}$ é corrompido por uma pequena quantidade de ruíido. Esse sinal é gerado pelos seguintes comandos:

```
amplitude_1 = 5;
freq_1 = 5;
amplitude_2 = 2;
freq_2 = 50;
Fs = 1000;
time = 0:1/Fs:(1-1/Fs);
sine_1 = amplitude_1*sin(2*pi*freq_1.*time);
sine_2 = amplitude_2*sin(2*pi*freq_2.*time);
noise = randn(1,length(time));
x_clean = sine_1 + sine_2;
x_noisy = x_clean + noise;
figure(14);
plot(time,x_clean);
```




```
figure(15);
plot(time,x_noisy);
```



Tarefas

Trace os seguintes sinais:

a) $(-0.5)^n$

b) $(2)^{-n}$

c) $(-2)^n$

d) e^{-2t}

e) $2\cos(2\pi 50t)$