Simulação de Modulação e Demodulação AM

Criação dos sinais e modulação

Vamos simular a modulação e a demodulação de uma mensagem utilizando AM (Amplitude Modulation). Para tanto, foi escolhida uma onda portadora de 1kHz, para simplificar, consideramos que o sinal de mensagem que queremos transmitir é uma simples senoide de 10 Hz.

Para tanto, criamos dentro da simulação no Matlab os sinais:

Sinal de mensagem: $m(t) = A_m \cdot cos(2\pi \cdot f_M \cdot t)$:

```
m = Am * cos(2*pi*fm*t);
```

Sinal da onda portadora: $c(t) = A_c cos(2\pi f_c t)$:

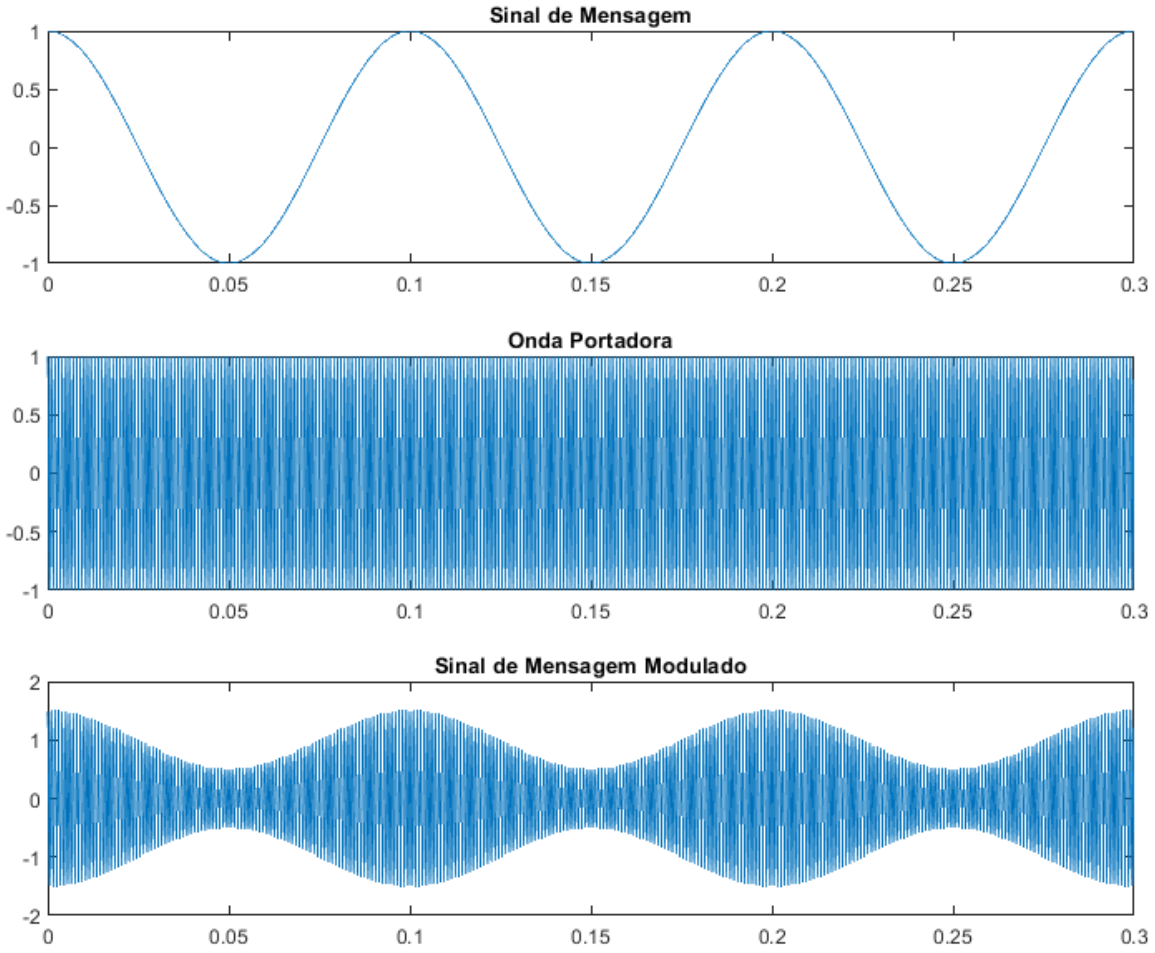
```
c = Ac * cos(2*pi*fc*t);
```

Após a criação desses sinais, criamos o sinal modulado aplicando:
$s(t) = (1 + k \cdot m(t)) \cdot c(t)$:

```
s = (1 + k*m).*c;
```

Pronto! Acabamos de simular uma modulação AM.

Ao plotar esses três vetores, temos o seguinte gráfico:



Visualização do espectros dos sinais

Agora que temos os sinais de mensagem, a onda portadora e o sinal modulado, vamos analisar como eles se comportam no domínio da frequência. Para tanto, vamos calcular a transformada de Fourier desses sinais utilizando a função do arquivo `fourier.m`. É necessário ter o arquivo `fourier.m` na mesma pasta onde está o arquivo em que estamos fazendo a simulação para que isso funcione.
Esta função facilita o nosso cálculo, já que tudo o que precisamos fazer é chamar a função da seguinte forma:

```
[f, X] = fourier(x, fs);
```

Onde utilizamos as variáveis:

`x`	Vetor que contém o sinal no tempo que desejamos transformar
`fs`	Frequência de amostragem do sinal que desejamos transformar
`X`	Vetor onde armazenamos o sinal x transformado
`f`	Vetor onde armazenaremos o eixo de frequência do sinal transformado

Por exemplo, para fazer um gráfico em frequência de $m(t)$ precisamos utilizar:

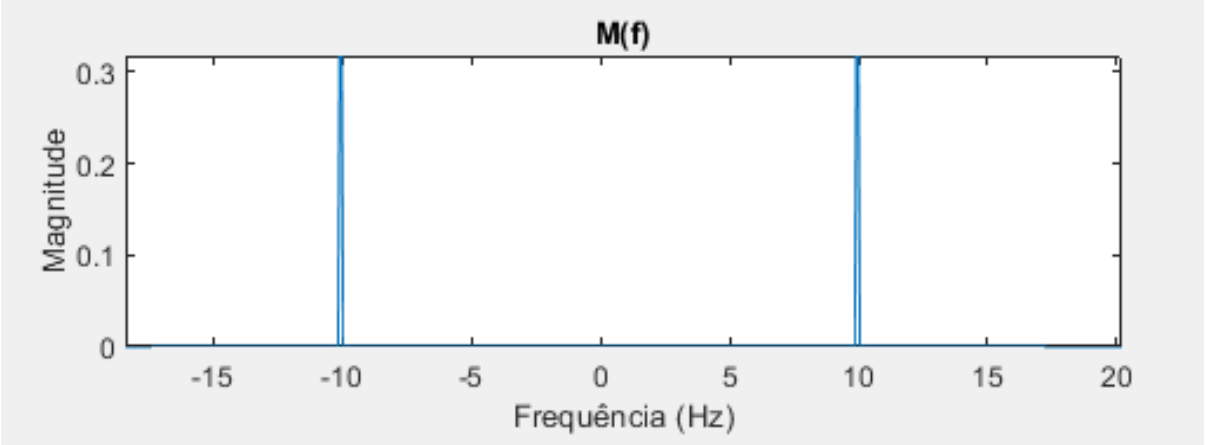
```
[f, M] = fourier(m, fs);

plot(f, abs(M))
xlabel('Frequência (Hz)')
ylabel('Magnitude')
title('M(f)')
```

Observe a utilização da função `abs`, que calcula o módulo de um número complexo (a transformada de Fourier pega sinais no tempo e nos devolve sinais complexos).

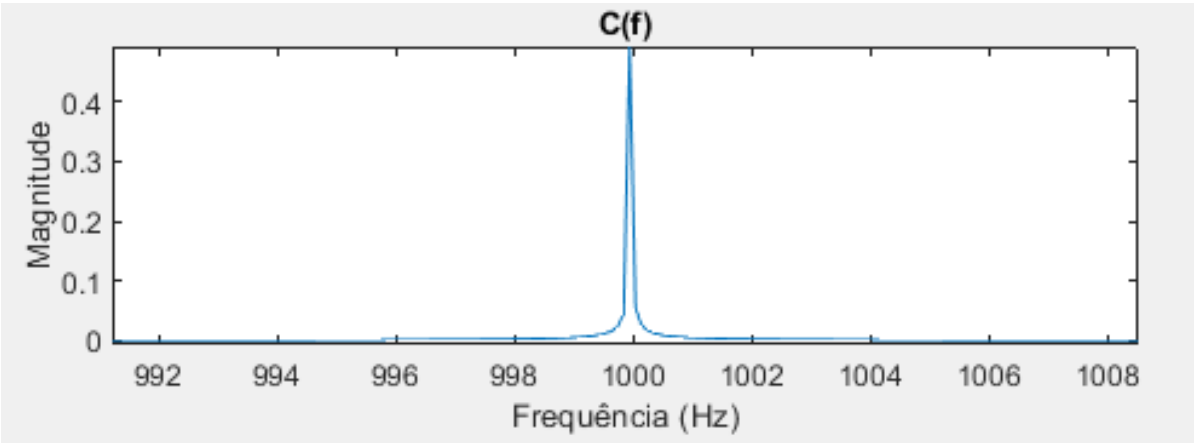
Ao calcular a transformada de Fourier, temos os seguintes gráficos:

Nesse caso, não conseguimos ver bem o que está acontecendo porque a frequência da onda portadora (1000Hz) é muito maior do que a frequência do sinal de mensagem (10Hz), o que inclusive é o usual na prática. Vamos então dar um zoom nos sinais para ver o que está acontecendo. Preste atenção nas escalas no eixo da frequência!

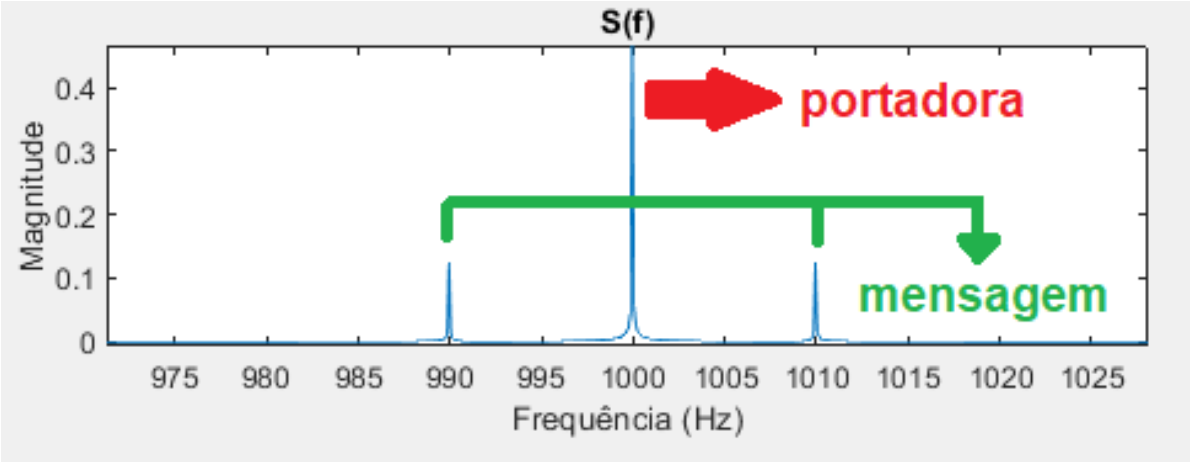
Dando zoom na transformada do sinal de mensagem, vemos dois impulsos, um em 10Hz e um em -10Hz. Isso está dentro do esperado porque a transformada de Fouriers de uma senoide é realmente dois deltas de Dirac (consulte sua tabela de transformadas de Fourier):



Dando zoom no espectro do sinal da onda portadora, temos também dois impulsos, um em 1000Hz e outro em -1000Hz. Isso acontece porque a onda portadora também é uma senoide e não tem nenhuma diferença do sinal de mensagem a não ser sua frequência. Na figura temos o impulso direito da portadora (o impulso esquerdo está em -1000Hz).

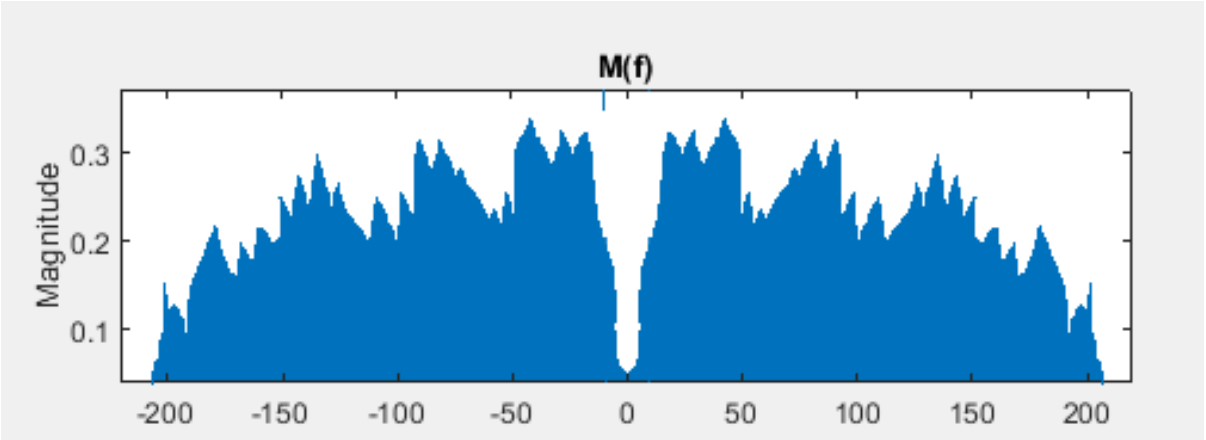


Por fim, vamos ver agora como está o sinal modulado. Dando zoom nos impulsos da parte direita, vemos isso:



Observe como esta figura nada mais é do que uma “sobreposição” das duas últimas. Vamos a portadora em 1000Hz e o sinal de mensagem com os dois impulsos, um em -10Hz e outro em 10Hz, mas agora deslocados pela onda portadora em 1000Hz, ou seja, aparecendo em 1000Hz - 10Hz = 990Hz e em 1000Hz + 10Hz = 1010Hz.

***Observe que como estamos fazendo um exemplo didático, estamos usando como sinal de mensagem uma senoide, que no espectro aparece como um impulso. Mas o sinal de mensagem pode ser o que quer que desejamos, por exemplo, uma música. Nesse caso, não temos uma tabela de transformadas de Fourier para músicas, mas ao calculá-las computacionalmente, obtemos algo parecido com isso:

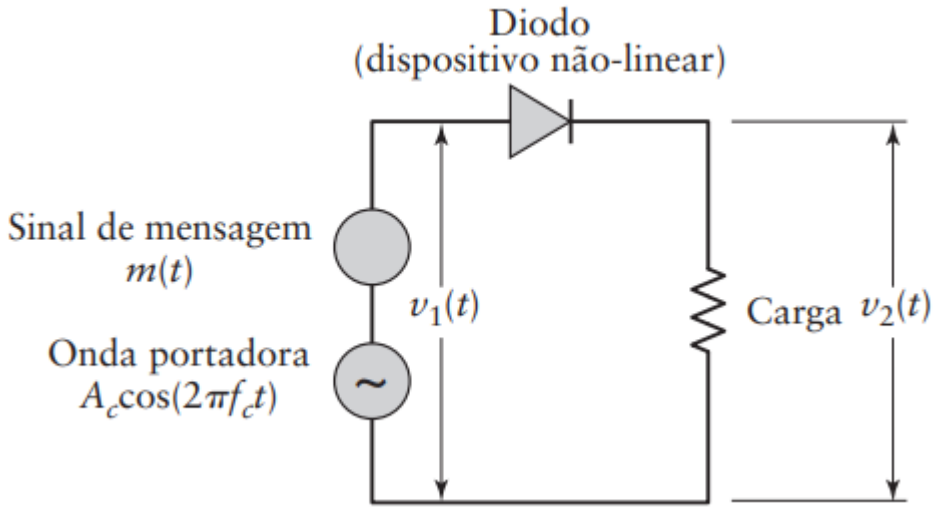


Um exercício interessante pode ser tentar fazer a modulação AM para um sinal de áudio em um arquivo `wav`. Para tanto, seria necessário encontrar uma forma de importar esse arquivo para o Matlab de forma que ele seja representado como um vetor igual ao sinal `m(t)`. Existem diversos métodos para fazer isso, e não é um tópico tão avançado, mas está fora do nosso escopo. Recomenda-se apenas utilizar um arquivo no formato `wav` de poucos segundos porque esse formato de áudio não aplica compressão ao áudio e porque um arquivo de áudio muito longo tomaria muito tempo para ser processado.

Demodulação

A seguir, vamos fazer a demodulação do sinal AM. De acordo com o sugerido por *Haykin - Introdução aos Sistemas de Comunicação*, uma forma de se fazer a demodulação é:

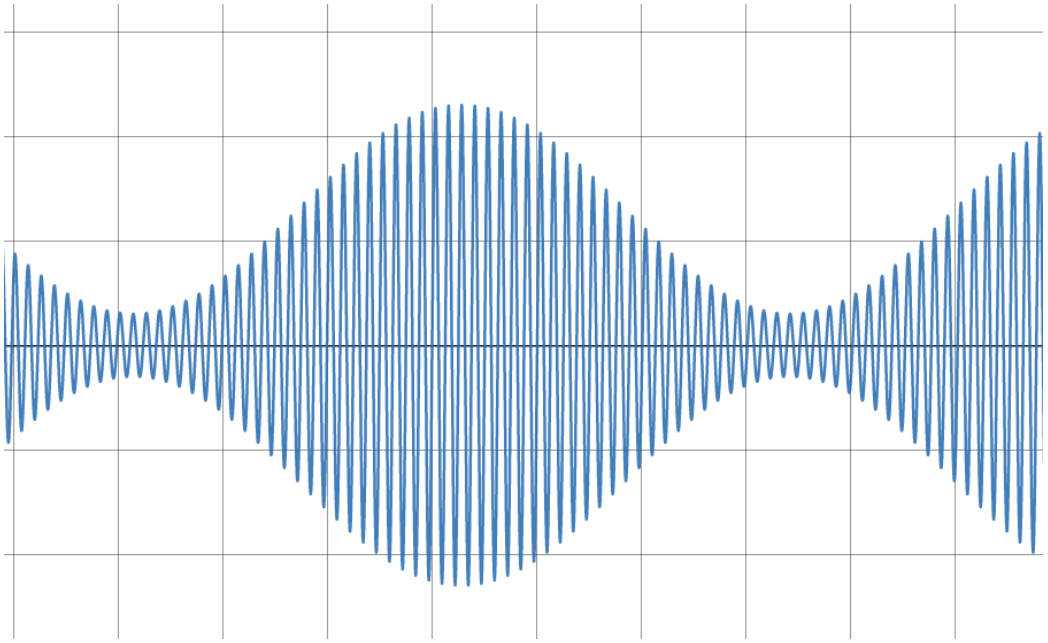
- Cortar a parte negativa do sinal (para isso, podemos passá-lo por um diodo)
- Filtrar a alta frequência do sinal de saída (não mostrado na figura).



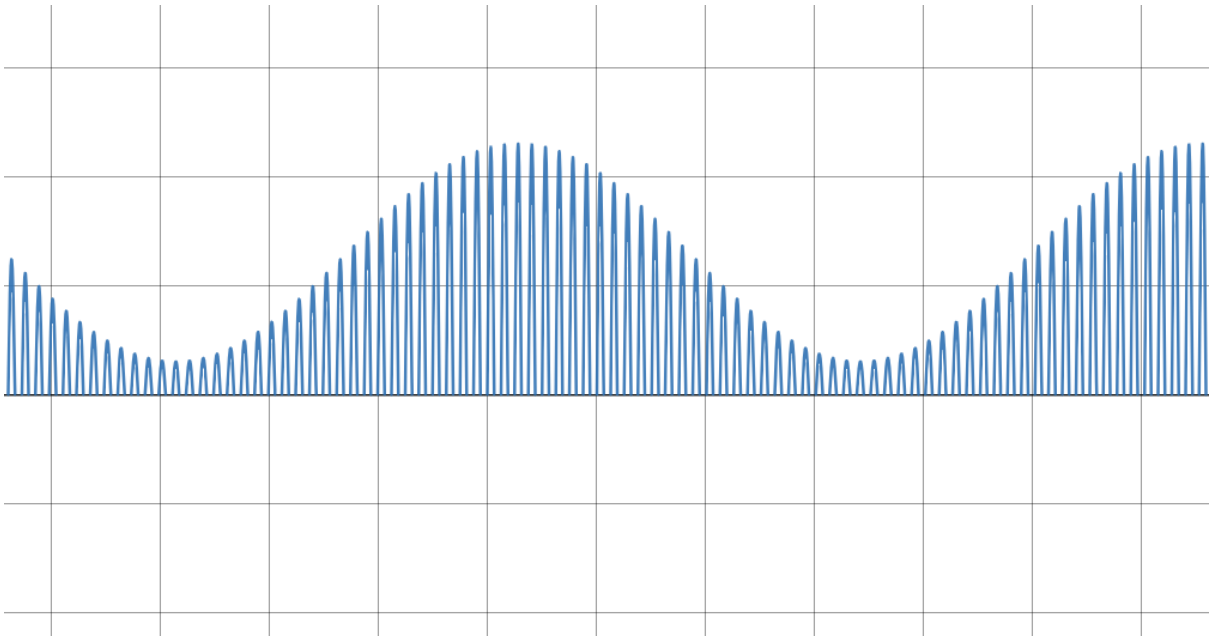
Fonte: *Haykin - Introdução aos Sistemas de Comunicação, Capítulo 3.*

Na simulação, estamos fazendo isso:

Sinal AM modulado, que pode ser recebido por uma antena:



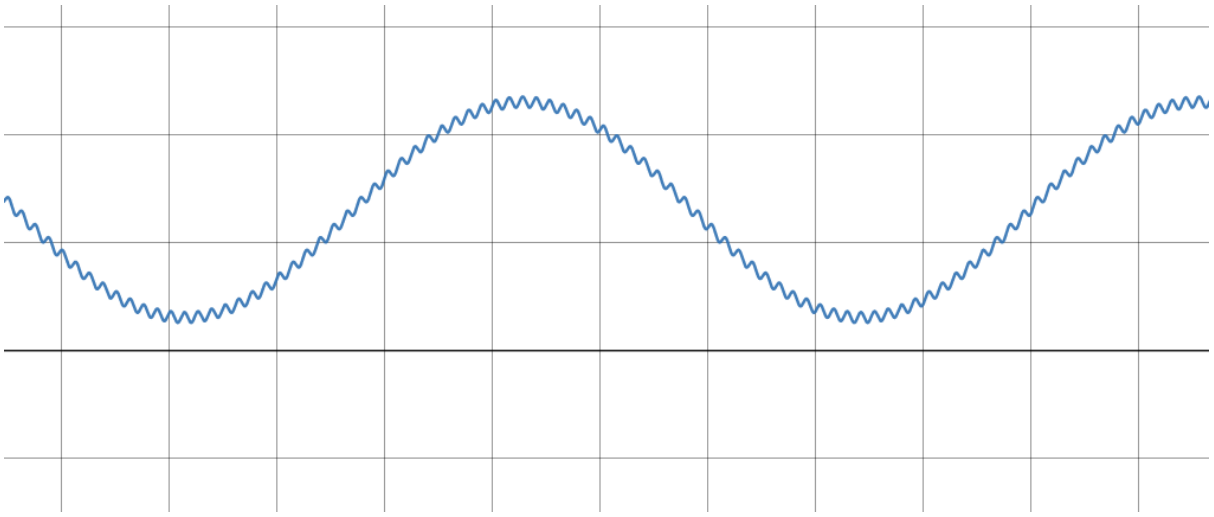
Sinal após a remoção da parte negativa:



Essa remoção dos valores negativos pode ser feita no Matlab criando um novo vetor que contém para cada valor do sinal s o máximo entre esse valor do sinal s e zero:

```
s_d = max(s,0);
```

Sinal AM após a filtragem das altas frequências:



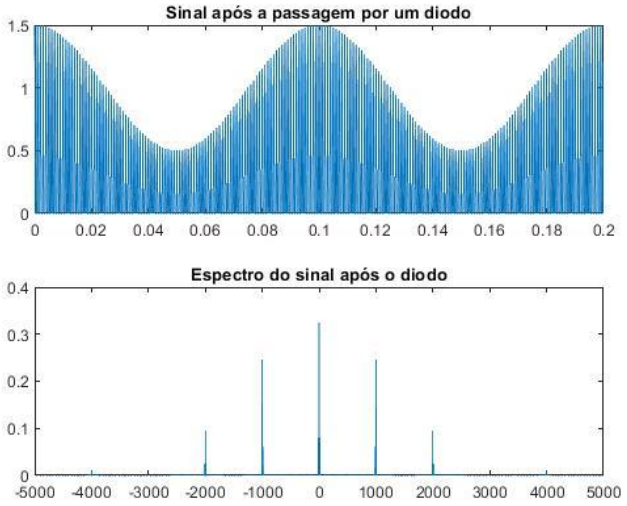
Idealmente, obteríamos uma senoide “sem oscilações menores”, mas isso é muito difícil de se obter na prática, geralmente tem um sinal com algum ruído, como mostrado acima.
Para fazer a filtragem do sinal no Matlab, utilizaremos uma função já criada previamente no arquivo *PassaBaixa.m* que faz a filtragem:

```
function H = PassaBaixa(f, f_c)

    %{
    Retorna a função de transferência de um filtro passa-baixas passivo
    f_c é a frequência de corte do filtro
    f é o vetor de frequências, em Hertz
    %}

    % Constante de tempo do circuito
    RC = 1/(2*pi*f_c);

    % Eixo de frequência em radianos
    w = 2*pi*f;

    % Vetor da função de transferência do filtro (em valores complexos)
    H = w;

    % Aplicar a fórmula da função de transferência para cada valor de H
    for i = 1 : length(H)
        % Obter os valores complexos do filtro
        H(i) = 1/(1 + 1i*w(i)*RC);
    end

end
```

Observe que esta função **somente nos retorna a função de transferência do filtro!**. Cabe a nós então fazer a multiplicação no domínio da frequência para obter o sinal desejado.

Por isso, primeiro calculamos a transformada de Fourier do sinal de entrada do filtro, que é o sinal `s_d`, já sem a parte negativa (utilizando também a função da transformada de Fourier, no arquivo *fourier.m*):

```
[f, S_D] = fourier(s_d, fs);
```



Após a obtenção da função de transferência do filtro:

```
H = PassaBaixa(f, f_c);
```

Fazemos a multiplicação em frequência do sinal sem a parte negativa pela função de transferência do filtro:

```
saida_tf = S_D.*H;
```

E, finalmente, convertemos novamente nosso sinal de saída para o domínio do tempo (utilizando outra função auxiliar):

```
[tSaida, saida] = inv_fourier(saida_tf, fs);
```