impedancia_complexa_F540

March 11, 2020

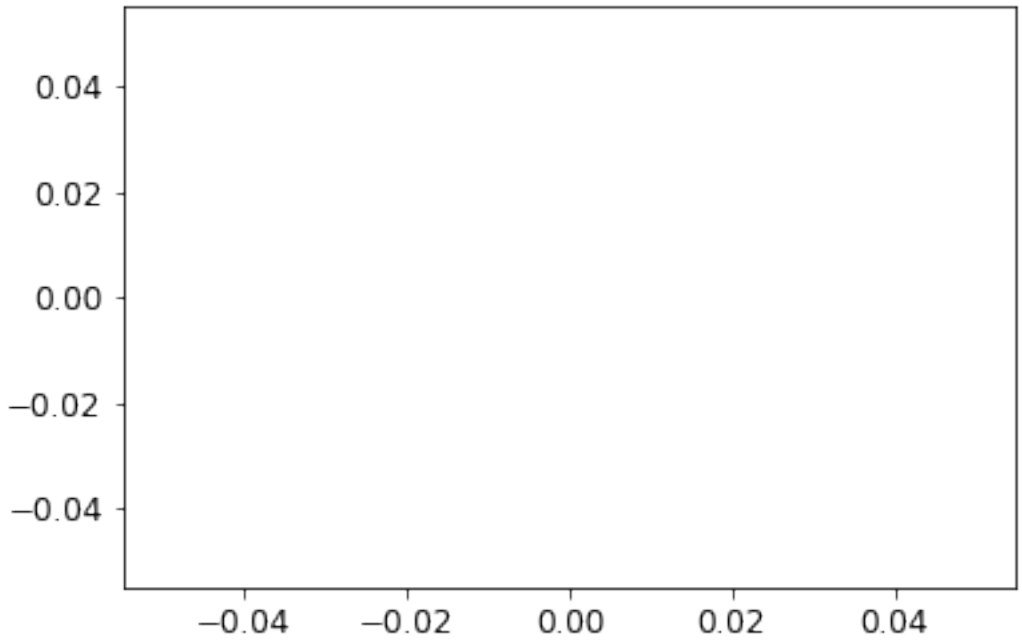
1 0. Definições

In [1]:
```
import numpy as np
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#****************************
#GRAFICOS
#****************************
#Ajsutando fontes padrão dos gráficos
font = { 'weight' : 'normal',
         'size'   : 12}

plt.rc('font', **font)
#Ajsutando espessura das linhas padrão dos gráficos
plt.rcParams['lines.linewidth'] = 2;
plt.locator_params(nticks=4);
# plt.locator_params(nticks=4);
#****************************
#FUNÇÃO PARA GRAFICAR DIAGRAMA DE ARGAND (PLANO COMPLEXO)
#****************************
def z_plot(zlist,scale_factor=1.1):
    ''' FUNÇÃO PARA GRAFICAR NÚMERO COMPLEXO NO DIAGRAMA DE ARGAND, CARTESIANO E POLAR
    entrada:
    zlist = lista contendo um ou mais números (array ou numpy.array)
    obs: se quiser apenas um número, coloque colchetes, [z]
    scale_factor (opcional) = fator de escala para definir o excesso dos eixos
    saída: figura matplotlib
    '''
    fig, ax = plt.subplots(2, figsize=(10,5));
    #----------------------
    #GRÁFICO CARTESIANO
    ax[0] = plt.subplot(1,2,1) # primeiro grafico de uma matrix 1x2
    for z in zlist:
        ax[0].plot([0,z.real],[0,z.imag],'.--k'); # desenha uma linha tracejada da ori
        ax[0].plot(z.real,z.imag,'o',markersize = 10) #desenha o ponto
```

```
    zmax = scale_factor*np.max(np.abs(zlist))
    ax[0].plot(zmax*np.array([0,0]),zmax*np.array([-1,1]),'-k')#eixos cartesianos
    ax[0].plot(zmax*np.array([-1,1]),zmax*np.array([0,0]),'-k')#eixos cartesianos
    ax[0].set_xlim([-zmax,zmax]);
    ax[0].set_ylim([-zmax,zmax]);
    ax[0].set_xlabel('parte real');
    ax[0].set_ylabel('parte imaginária');
    plt.title('Diagrama de Argand');
    ax[0].grid(True);
    #----------------------
    #GRÁFICO POLAR
    ax[1] = plt.subplot(1,2,2,projection='polar') # primeiro grafico de uma matrix 1x2
    count = 1
    for z in zlist:
        ax[1].plot([0,np.angle(z)],[0,np.abs(z)],'.--k'); # desenha uma linha tracejad
        ax[1].plot(np.angle(z),np.abs(z),'o', markersize = 10,  label = 'z'+str(count)
        count = count+1
    #ax[1].set_rmax(scale_factor*np.array([-np.abs(z),np.abs(z)]));
    plt.title('Diagrama de Argand (Forma polar)');
    ax[1].grid(True);
    ax[1].set_rlabel_position(0)
    ax[1].set_rmax(zmax)
    #ax[1].legend(loc='lower right')
    ax[1].legend(loc=(1.2,0.45))
    return fig # retorna a figura
#-----------------------------------------
#***************************
#FUNÇÃO PARA GRAFICAR DIAGRAMA DE BODE (|H|**2 E ARG(H)
#***************************
def fig_bode(freq,Tdbvec,fasevec,labelvec):
    '''Função para graficar painel 2x1 com diagrama de Bode (amplitude e fase)
    entrada:
    freq (vetor numérico de frequência)
    Tdbvec (vetor numérico)
    '''
    fig, ax = plt.subplots(2, sharex=True, figsize=(5, 10));
    #-------------------------------------
    #TRANSMITANCIA EM DECIBEIS
    ax[0] = plt.subplot(211)
    for m in range(np.shape(Tdbvec)[0]):
        ax[0].plot(freq,np.array(Tdbvec)[m,:],'-*',linewidth=2,label=labelvec[m])
        #ax[0].plot(freq,np.array(Tdbvec)[m,:],,label=labelvec[m])
    ax[0].set_ylim((np.min(Tdbvec),np.max(Tdbvec)+5))
    ax[0].set_xscale('log')
    ax[0].set_ylabel('T (dB)')
    ax[0].legend(loc='lower right')
    ax[0].grid(True)
    #-------------------------------------
```

```
ax[1] = plt.subplot(212)
for m in range(np.shape(fasevec)[0]):
    ax[1].plot(freq,np.array(fasevec)[m,:],'-*',linewidth=2,label=labelvec[m])
ax[1].set_ylim([-200,200])
ax[1].set_xscale('log')
ax[1].set_ylabel('Fase (graus)')
ax[1].grid(True)
#-------------------------------------
#geral
ax[0].set_title('Diagrama de Bode')
plt.xlabel('Frequência (Hz)')
return fig
```



2 1. Impedâncias complexas

Abaixo, definimos as impedâncias complexas para o capacitor, para o capacitor:

$$Z_c = -jX_c = -j\frac{1}{\omega C} = -j\frac{1}{2\pi f C},$$

sendo f a frequência (em [Hz]).

Para o indutor,

$$Z_l = jX_l = j\omega L = j2\pi f L,$$

sendo f a frequência (em [Hz]).

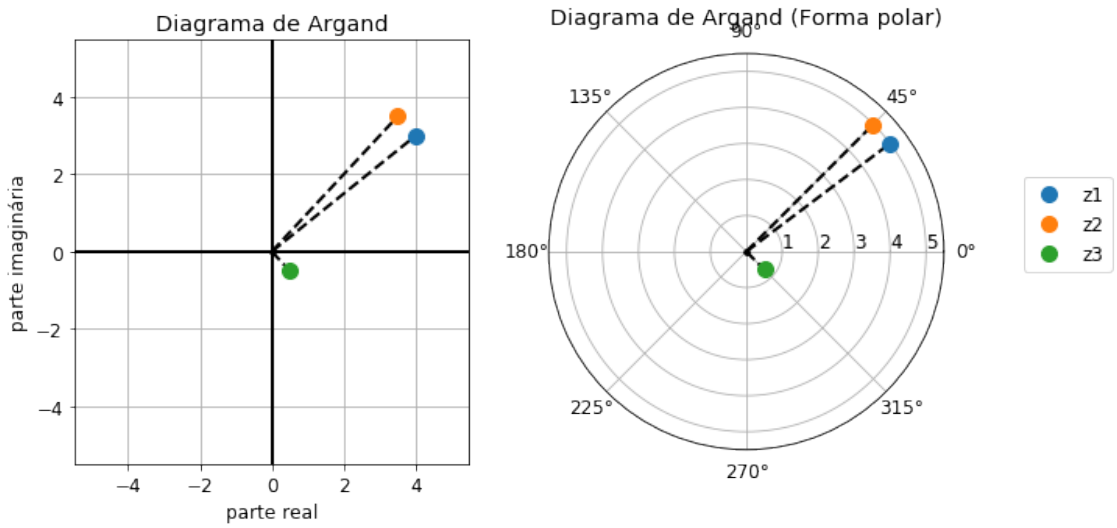
```
In [3]: #-----------------
        #CAPACITOR
        def Zc(freq,C):
            j = complex(0,1)
            #reatancia
            Xc = 1/(2*np.pi*freq*C)
            #impedancia complexa
            Zc = -j*Xc
            return Zc
        #-----------------
        #INDUTOR
        def Zl(freq,L):
            j = complex(0,1)
            #reatancia
            Xl = 2*np.pi*freq*L
            #impedancia complexa
            Zl = j*Xl
            return Zl
```

A impedância de cada um dos componentes é **puramente reativo**, não possui parte real:

Pergunta: Quais os valores possíveis para a fase das impedâncias de componentes **puramente reativos**?

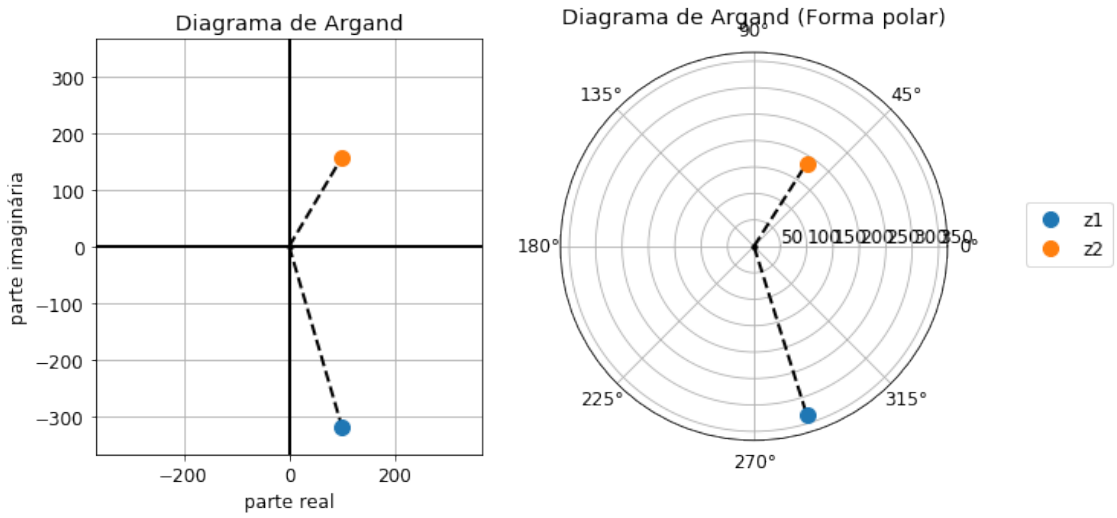
```
In [24]: #definindo valores dos componentes
         freq = 500 # frequencia [Hz]
         C = 90e-6; #capacitancia [F]
         L = 10e-3; # indutancia  [H]
         #calculando
         print('impedância complexa capacitor, Zc= {:1.1g}'.format(Zc(freq,C)))
         print(Zl(freq,L))
         #graficando:
         z0 = 4+3j
         zI = (z0+1j*np.conj(z0))*0.5
         zQ = (z0-1j*np.conj(z0))*0.5
         print('Arg(zQ/zI)=',np.angle(zQ/zI)/np.pi)
         print('Arg(zI)=',np.angle(zI)/np.pi)
         print('Arg(zQ)=',np.angle(zQ)/np.pi)
         test=z_plot([z0,zI,zQ])
```

```
impedância complexa capacitor, Zc= 0-4j
31.41592653589793j
Arg(zQ/zI)= -0.5
Arg(zI)= 0.25
Arg(zQ)= -0.25
```



Já para o circuito RC ou RL série, a **impedância total possui parte real e imaginária**. Note que o RC possui parte imaginária negativa ($\mathcal{I}\Updownarrow(Z) < 0$, *capacitivo*) e o RL possui parte imaginária positivo ($\mathcal{I}\Updownarrow(Z) > 0$, *indutivo*),

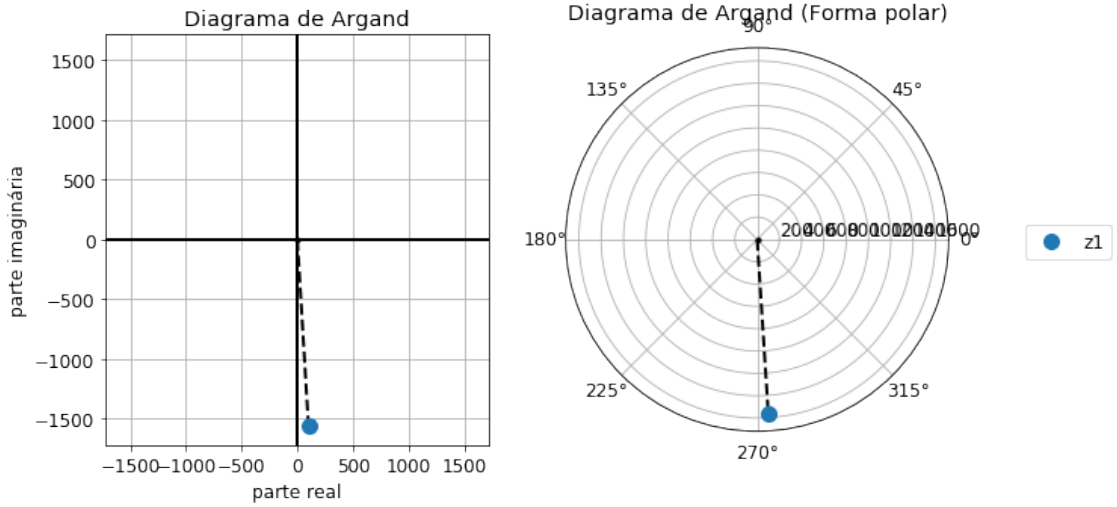
```
In [7]: #definindo valores dos componentes
        freq = 500 # frequencia [Hz]
        C = 1e-6; #capacitancia [F]
        L = 50e-3; # indutancia  [H]
        R = 100; #resistencia [Ohms]
        #impedancia total de um circuito RC ou RL
        print(R + Zc(freq,C))
        print(R + Zl(freq,L))
        #graficando:
        z_plot([R+Zc(freq,C),R+Zl(freq,L)]);

(100-318.3098861837907j)
(100+157.07963267948966j)
```



Já para o circuito *RLC* série, dependendo da frequência, ele pode ser mais indutivo $\mathcal{I}\Updownarrow(Z) > 0$ ou capacitivo $\mathcal{I}\Updownarrow(Z) < 0$. Altere o valor da frequência (`freq`) e verifique este comportamento.

In [8]:
```
#definindo valores dos componentes
freq = 100 # frequencia [Hz]
C = 1e-6; #capacitancia [F]
L = 50e-3; # indutancia  [H]
R = 100; #resistencia [Ohms]
#impedancia total de um circuito RC ou RL
print(R + Zl(freq,L) + Zc(freq,C))
z_plot([R + Zl(freq,L) + Zc(freq,C)]);
```

```
(100-1560.1335043830557j)
```



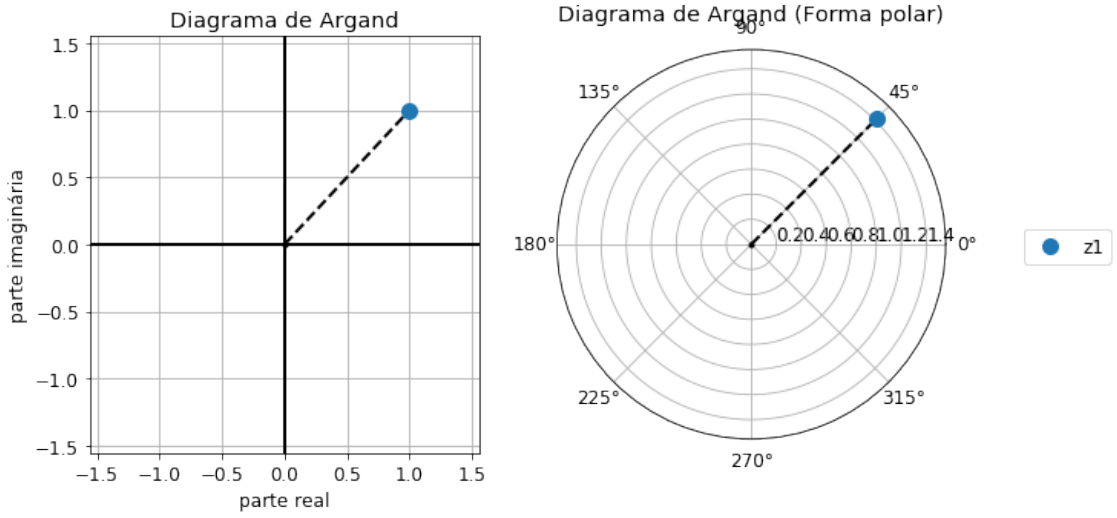
2.1 Forma polar

2.1.1 Note a importância de usar a função `np.angle` (argumento), ao invés de usar o `np.arctan` (arco-tangente)

No primeiro quadrante...

```
In [9]: z = 1 + complex(0,1)
        print(z)
        print(z.real)
        print(z.imag)
        #calculando a fase (em graus):
        fase_atan = (180/np.pi)*np.arctan(z.imag/z.real)
        print(fase_atan)
        fase_angle = (180/np.pi)*np.angle(z)
        print(fase_angle)
        #graficando
        z_plot([z]);
```

```
(1+1j)
1.0
1.0
45.0
45.0
```

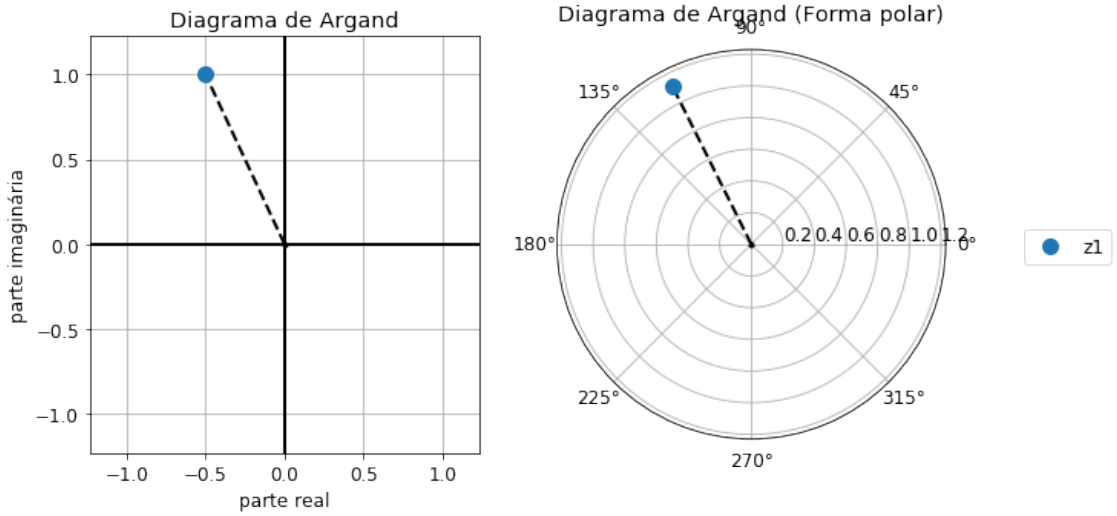


No segundo quadrante...

```
In [10]: z = -0.5 + complex(0,1)
         print(z)
         print(z.real)
         print(z.imag)
```

```
        #calculando a fase (em graus):
        fase_atan = (180/np.pi)*np.arctan(z.imag/z.real)
        print(fase_atan)
        fase_angle = (180/np.pi)*np.angle(z)
        print(fase_angle)
        #graficando
        z_plot([z]);
```

```
(-0.5+1j)
-0.5
1.0
-63.43494882292202
116.56505117707799
```



2.1.2 Convertendo entre as duas formas:

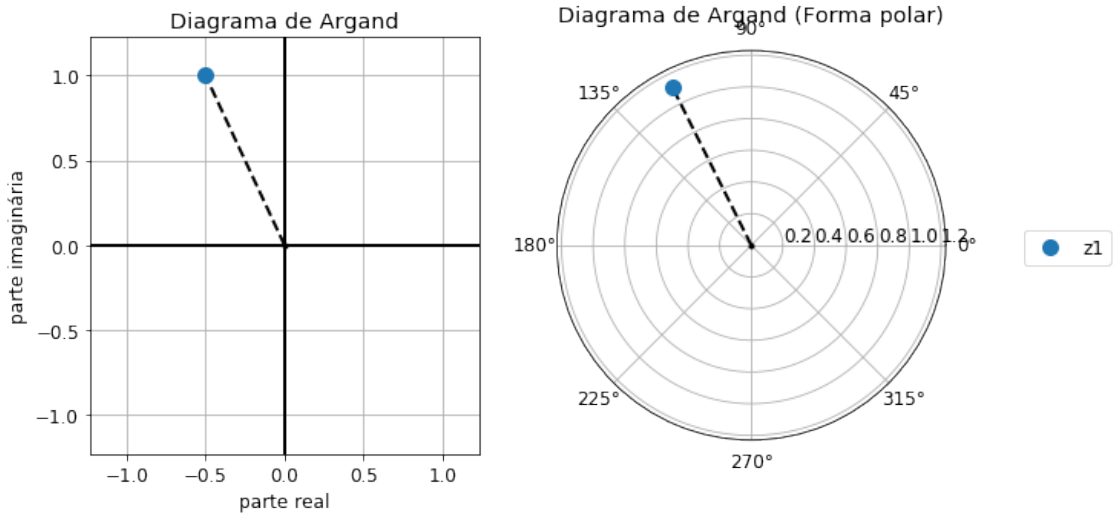
$$z = a + jb \rightarrow z = |z| \exp(j\theta),$$

sendo que $\theta = arg(z)$

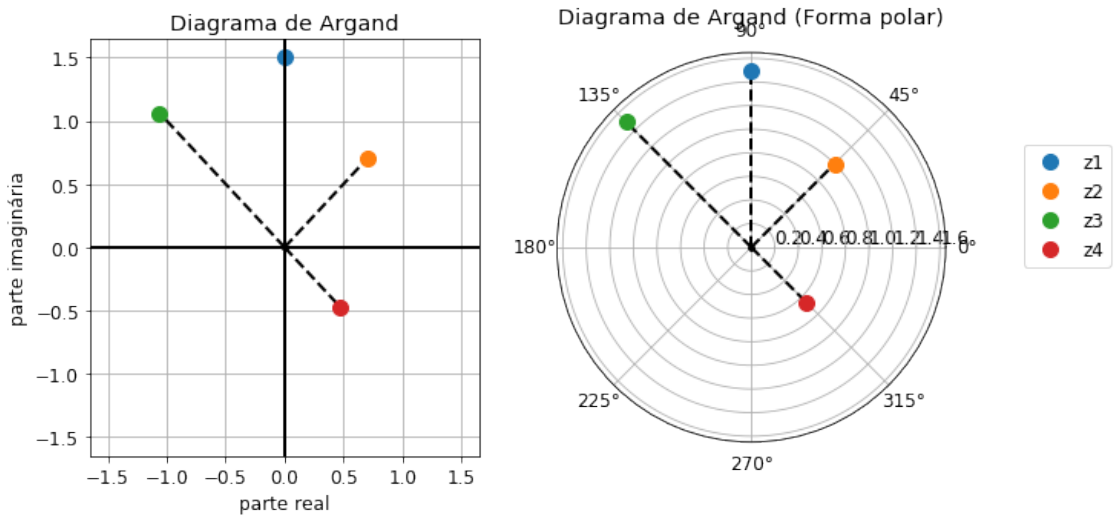
```
In [11]: z = -0.5 + complex(0,1)
         print('forma cartesiana:',z)
         print('parte real:',z.real)
         print('parte imaginaria:',z.imag)
         #fase
         theta = np.angle(z)
         print('fase (rad)',theta)
         #modulo
         zabs = np.abs(z)
         print('módulo',zabs)
```

```
        #FORMA POLAR
        j=complex(0,1)
        zpolar = zabs* np.exp(j*theta)
        print('forma polar:',zpolar)
        #--------------------------------------
        #Grafico representando o número complexo
        #graficando
        z_plot([z]);
```

```
forma cartesiana: (-0.5+1j)
parte real: -0.5
parte imaginaria: 1.0
fase (rad) 2.0344439357957027
módulo 1.118033988749895
forma polar: (-0.5+1j)
```




```
In [12]: j=complex(0,1)
         r1, theta1 = 1.5, np.pi/2
         r2, theta2 = 1.0, np.pi/4
         z1 = r1*np.exp(j*theta1)
         z2 = r2*np.exp(j*theta2)
         z3  = z1*z2
         z4 = z2/z1
         z_plot([z1,z2,z3,z4]);
```

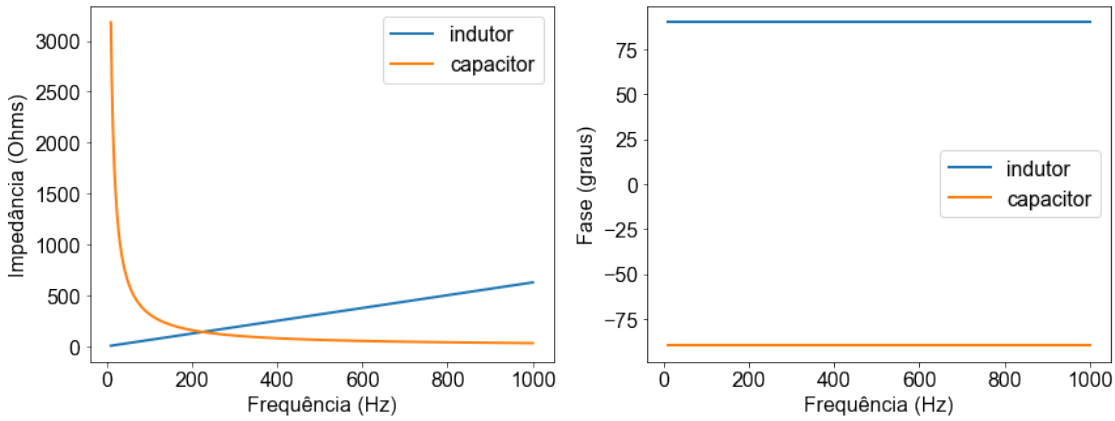


2.2 Graficando as impedâncias dos componentes reativos

```
In [13]: freq=np.linspace(10,1000,500) # vetor que vai de 10**0 até 10**6
         C=5e-6;
         L=100e-3;
         #-----
         #GRAFICOS
         #Ajsutando fontes padrão dos gráficos
         font = {'family' : 'arial',
                 'weight' : 'normal',
                 'size'   : 16} #este tipo de variável é um dicionário
         plt.rc('font', **font)
         plt.rcParams['lines.linewidth'] = 2

         #Ajsutando espessura das linhas padrão dos gráficos
         #IMPEDÂNCIA,|Z|
         fig, ax = plt.subplots(2,figsize=(14, 5))
         ax[0] = plt.subplot(121)
         ax[0].plot(freq,np.abs(Zl(freq,L)),label='indutor')
         ax[0].plot(freq,np.abs(Zc(freq,C)),label='capacitor')
         #nome dos eixos e legendas
         ax[0].set_xlabel('Frequência (Hz)')
         ax[0].set_ylabel('Impedância (Ohms)')
         ax[0].legend(loc='upper right')
         #-----
         #FASE,arg(Z)
         ax[1] = plt.subplot(122)
         ax[1].plot(freq,np.angle(Zl(freq,L),deg=True),label='indutor')
         ax[1].plot(freq,np.angle(Zc(freq,C),deg=True),label='capacitor')
```

```
        #nome dos eixos e legendas
        ax[1].set_xlabel('Frequência (Hz)')
        ax[1].set_ylabel('Fase (graus)')
        ax[1].legend(loc='center right')
```

```
Out[13]: <matplotlib.legend.Legend at 0x118892828>
```



3 2. Circuitos contendo 1 componentes reativos, RC e RL

3.0.1 Abaixo definimos uma função para fazer diagramas de Bode

3.0.2 Wien bridge

```
In [40]: freq=np.logspace(1,6,50) # vetor que vai de 10**0 até 10**6
         #lomega=np.log10(omega)
         C=1e-6;
         L=1e-3/(2*np.pi)**2;
         R=100;
         #----------------
         Zc0 = Zc(freq,C)
         #Z2 = Zc(freq,C)*R/(Zc(freq,C)+R) # impedancia total
         H = 1/(1+(R+Zc0)**2/(R*Zc0)) # H com resistor na said
         H = R*Zc0/(R*Zc0+R*Zc0*Zc0+R**2+Zc0**2) # H com resistor na said
         #-----
         #----------------
         T=20*np.abs(H)
         Tdb=20*np.log10(T)
         fase = np.angle(H,deg=True)
         ##-
         #GRAFICANDO BODE, RC e CR
         fig=fig_bode(freq,[T,],[fase,],['C',]);
```

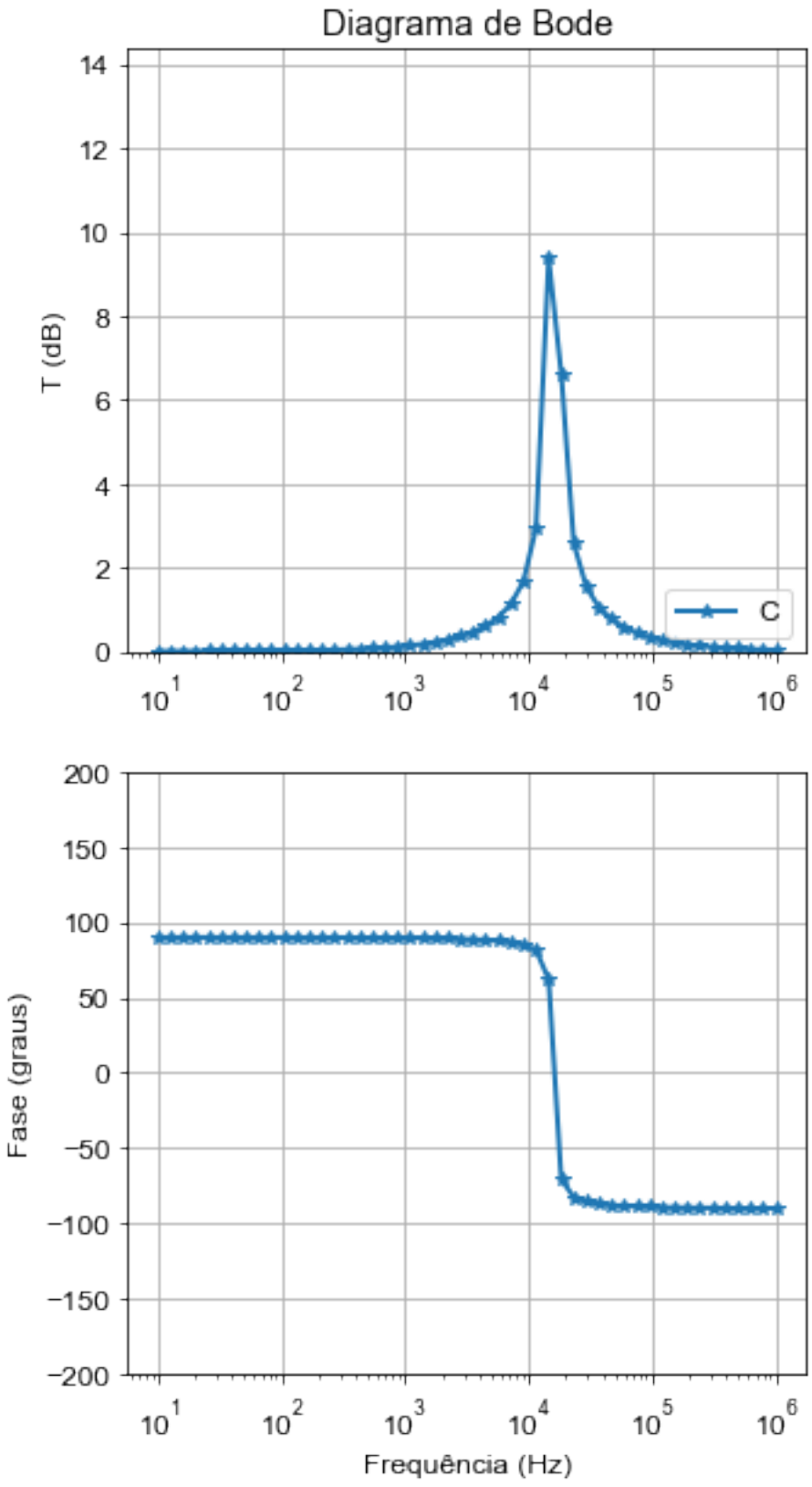


Diagrama de Bode
T (dB)
14
12
10
8
6
4
2
0
C
10^1
10^2
10^3
10^4
10^5
10^6
Fase (graus)
200
150
100
50
0
−50
−100
−150
−200
Frequência (Hz)

In [36]:
```
freq=np.logspace(1,6,50) # vetor que vai de 10**0 até 10**6
#lomega=np.log10(omega)
C=1e-6;
L=1e-3/(2*np.pi)**2;
R=100;
#----------------
Z1 = Zc(freq,C)+R
Z2 = Zc(freq,C)*R/(Zc(freq,C)+R) # impedancia total
H = Z2/(Z1+Z2) # H com resistor na said
#-----
#----------------
T=20*np.abs(H)
Tdb=20*np.log10(T)
fase = np.angle(H,deg=True)
##-
#GRAFICANDO BODE, RC e CR
fig=fig_bode(freq,[T,],[fase,],['C',]);
```

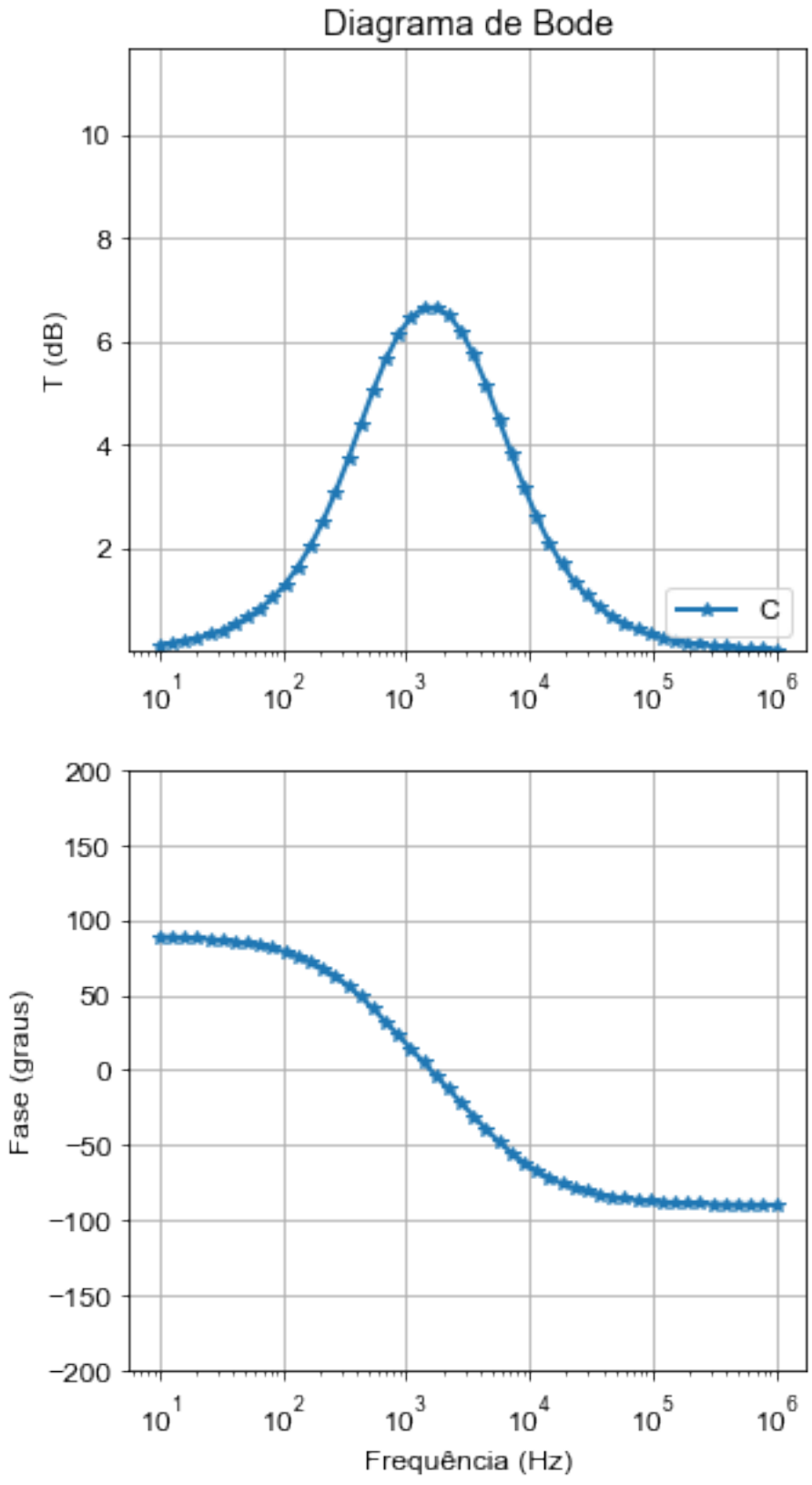


Diagrama de Bode
T (dB)
10
8
6
4
2
C
10^1
10^2
10^3
10^4
10^5
10^6
Fase (graus)
200
150
100
50
0
−50
−100
−150
−200
Frequência (Hz)

3.0.3 passa-banda ativo

```
In [25]: freq=np.logspace(1,6,50) # vetor que vai de 10**0 até 10**6
         #lomega=np.log10(omega)
         C=1e-3;
         L=1e-3/(2*np.pi)**2;
         R1=0.1;
         R2=5;
         #----------------
         Z1 = Zc(freq,C)+Zl(freq,L)+R1
         Z2 = R1 # impedancia total
         H = -R2/(Z1+Z2) # H com resistor na saida
         #-----
         #----------------
         T=np.abs(H)
         Tdb=20*np.log10(T)
         fase = np.angle(H,deg=True)
         ##-
         #GRAFICANDO BODE, RC e CR
         fig=fig_bode(freq,[T,],[fase,],['C',]);
```

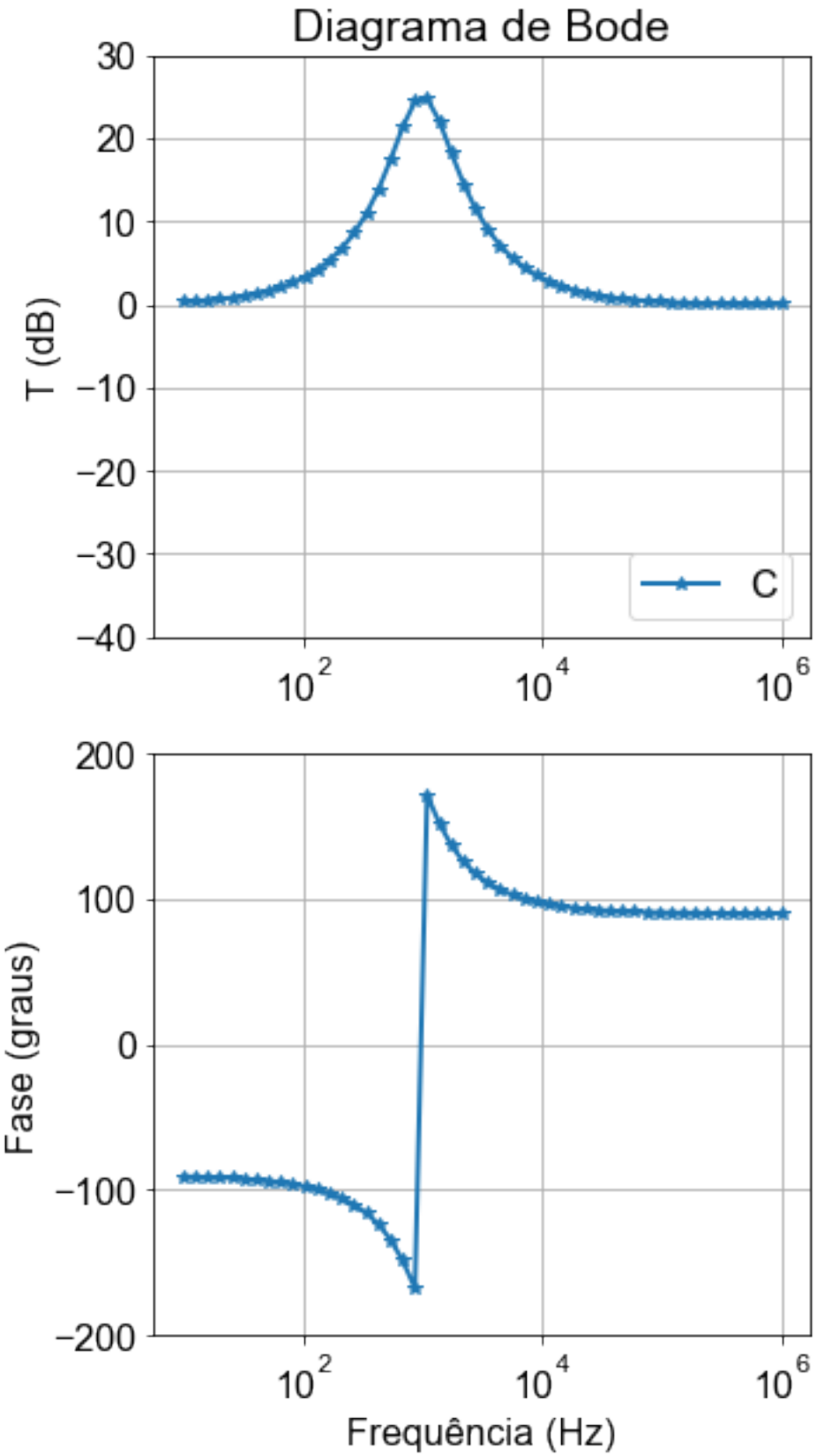


Diagrama de Bode
T (dB)
30
20
10
0
−10
−20
−30
−40
C
10^2
10^4
10^6
Fase (graus)
200
100
0
−100
−200
10^2
10^4
10^6
Frequência (Hz)

3.0.4 circuito RC

```
In [ ]: freq=np.logspace(1,6,20) # vetor que vai de 10**0 até 10**6
        #lomega=np.log10(omega)
        C=1e-3;
        L=1e-3/(2*np.pi)**2;
        R=0.2;
        #----------------
        Zt = Zc(freq,C)+R # impedancia total
        Hrc =  Zc(freq,C)/Zt # H com capacitor na saida
        Hcr = R/Zt # H com resistor na saida

In [15]: #----------------
         Trc=np.abs(Hrc)
         Trcdb=20*np.log10(Trc)
         fase_rc = np.angle(Hrc,deg=True)
         #capacitor
         Tcr=np.abs(Hcr)
         Tcrdb=20*np.log10(Tcr)
         fase_cr = np.angle(Hcr,deg=True)

In [16]: #GRAFICANDO BODE, RC e CR
         fig=fig_bode(freq,[Trcdb,Tcrdb],[fase_rc,fase_cr],['C','R']);
```

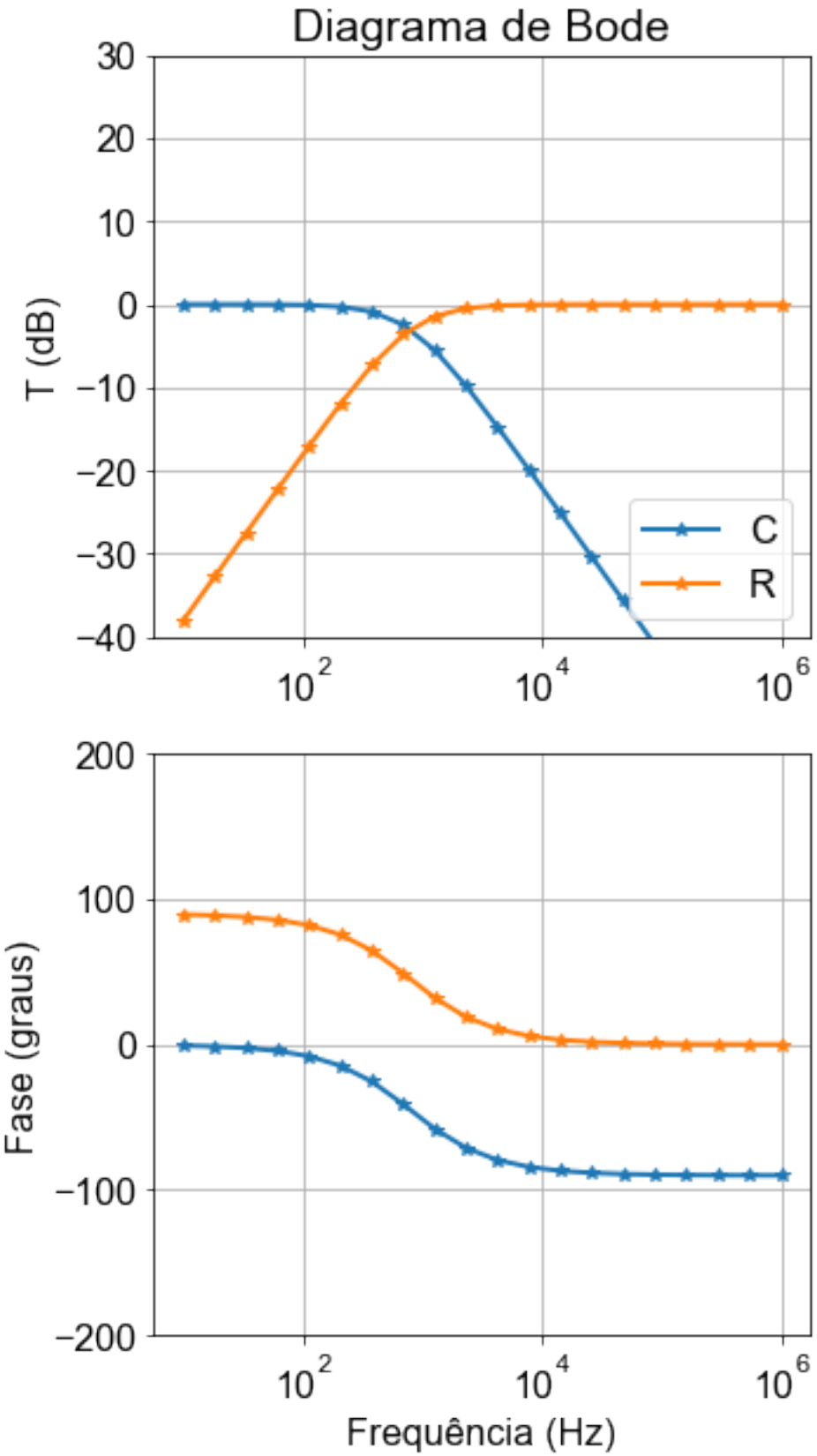


Diagrama de Bode
T (dB)
30
20
10
0
−10
−20
−30
−40
C
R
10^2
10^4
10^6
Fase (graus)
200
100
0
−100
−200
10^2
10^4
10^6
Frequência (Hz)

```
In [17]: #SALVANDO FIGURA
         #-----------------------------
         print('pasta atual:',os.getcwd())
         name='bode_rc'
         folder_path=os.getcwd()
         ext='pdf'
         path=os.path.join(folder_path,name + '.' + ext)
         fig.savefig(path,format='pdf')
         print('arquivo salvo:',path)
```

```
pasta atual: /Users/gsw/Documents/GitHub/F540/Material suplementar
arquivo salvo: /Users/gsw/Documents/GitHub/F540/Material suplementar/bode_rc.pdf
```

3.0.5 circuito RL

```
In [ ]: freq=np.logspace(0,6,200) # vetor que vai de 10**0 até 10**6
        #lomega=np.log10(omega)
        C=1e-3;
        L=1e-3/(2*np.pi)**2;
        R=0.2;
        #----------------
        Zt = Zl(freq,L)+R
        Hrl,Hlr = Zl(freq,L)/Zt, R/Zt
        #----------------
        Trl=np.abs(Hrl)
        Trldb=20*np.log10(Trl)
        fase_rl = np.angle(Hrl,deg=True)
        #capacitor
        Tlr=np.abs(Hlr)
        Tlrdb=20*np.log10(Tlr)
        fase_lr = np.angle(Hlr,deg=True)
```

```
In [ ]: #GRAFICANDO BODE, LR e RL
        fig=fig_bode(freq,[Trldb,Tlrdb],[fase_rc,fase_cr],['L','R']);
```

```
In [ ]: #SALVANDO FIGURA
        #-----------------------------
        print('pasta atual:',os.getcwd())
        name='bode_rl'
        folder_path=os.getcwd()
        ext='pdf'
        path=os.path.join(folder_path,name + '.' + ext)
        fig.savefig(path,format='pdf')
        print('arquivo salvo:',path)
```

3.1 Circuitos contendo 2 componentes reativos, RLC

In [ ]:
```
freq=np.logspace(0,6,500) # vetor que vai de 10**0 até 10**6
#lomega=np.log10(omega)
C=1e-3;
L=1e-3/(2*np.pi)**2;
R=0.1;
#----------------
Zt = Zc(freq,C)+Zl(freq,L)+R # impedancia total
Hr = R/Zt # resistor na saida
Hc = Zc(freq,C)/Zt # capacitor na saida
Hl = Zl(freq,L)/Zt # indutor na saida
Hlc = (Zc(freq,C)+Zl(freq,L))/Zt # indutor e capacitor na saida
#----------------
Tr=np.abs(Hr)
Trdb=20*np.log10(Tr)
fase_r = np.angle(Hr,deg=True)
#capacitor
Tc=np.abs(Hc)
Tcdb=20*np.log10(Tc)
fase_c = np.angle(Hc,deg=True)
#indutor
Tl=np.abs(Hl)
Tldb=20*np.log10(Tl)
fase_l = np.angle(Hl,deg=True)
#indutor+capacitor
Tlc=np.abs(Hlc)
Tlcdb=20*np.log10(Tlc)
fase_lc = np.angle(Hlc,deg=True)
```

In [ ]:
```
fig=fig_bode(freq,[Trdb,Tcdb,Tldb,Tlcdb],[fase_r,fase_c,fase_l,fase_lc],['R','L','C','
```

In [ ]:
```
#-----------------------------
print('pasta atual:',os.getcwd())
name='bode_rlc'
folder_path=os.getcwd()
ext='pdf'
path=os.path.join(folder_path,name + '.' + ext)
fig.savefig(path,format='pdf')
print('arquivo salvo:',path)
```