Victor Moraes - 2016027600

Código	Disciplina	Professor
ELE042	Processamento de Sinais	Hilton de Oliveira Mota

4ª lista de exercícios
Filtros LIT de tempo contínuo

1) Os filtros de Butterworth são especificados a partir de uma resposta de magnitude ao quadrado dada por

$$|H(j\omega)|^2 = \frac{1}{1+(\omega/\omega_c)^{2N}},$$

em que N é a ordem e ω_c é denominada frequência de corte, de ½ potência ou de $-3dB$. Considerando um filtro de Butterworth de ordem $N = 7$ e frequência de corte $f_c = 350Hz$: :

a) Determine, analiticamente, os valores dos polos, zeros e a função de transferência $H(s)$.
b) Faça um programa ou script do Matlab® que:
   - forneça os valores dos polos e zeros e a função de transferência $H(s)$;
   - plote o diagrama de polos e zeros;
   - plote as curvas de superfície de $|H(s)|$ e $\theta(s)$;
   - plote as respostas de magnitude $|H(j\omega)|$ e fase $\theta(\omega)$.

   Verifique se os valores condizem com os calculados anteriormente. Corrija eventuais discrepâncias e apresente, como resposta, o código do programa e os gráficos gerados.

$$|H(j\omega)|^2 = H(j\omega).H^*(j\omega)$$
$$|H(j\omega)|^2 = H(j\omega).H(-j\omega)$$
$$js = s$$
$$|H(s)|^2 = H(s).H(-s)$$
$$|H(s)|^2 = \frac{1}{1+(s/\omega_c)^N}.\frac{1}{1-(s/\omega_c)^N}$$

a) Função de transferência:

$$|H(s)|^2 = \frac{1}{1+(s/\omega_c)^{2N}}$$
$$|H(s)|^2 = \frac{1}{1+(s/700.\pi)^{14}}$$

a) Polos:

$$1+(s/\omega_c)^{2N} = 0$$

$$(s/\omega_c)^N = (-1)$$
$$s = \omega_c(-1)^{1/2N}$$
$$s = 700\pi(-1)^{1/14}$$

$$p_k = -700\pi\angle\frac{2\pi k}{14}, k \in [0, 13]$$

$$H_n(s) = \frac{1}{\prod(s - p_k)}$$

Zeros: não possui zeros do polinômio do numerador.

b) Bibliotecas:

In [2]:
```
import numpy as np
from math import pi, log10
import cmath
from scipy import signal, misc
from scipy.fft import fft, rfft, irfft
import matplotlib.pyplot as plt
from zplane import zplane
from matplotlib import patches
from matplotlib.pyplot import axvline, axhline
from collections import defaultdict
from scipy.signal import (freqz, butter, bessel, cheby1, cheb
y2, ellip,
                          tf2zpk, zpk2tf, lfilter, buttap,
bilinear, cheb2ord, cheb2ap
                          )
from numpy import asarray, tan, array, pi, arange, cos, log1
0, unwrap, angle
from matplotlib.pyplot import (stem, title, grid, show, plot,
xlabel,
                               ylabel, subplot, xscale, figu
re, xlim,
                               margins)
from laplace_utils import *
```

Determinação do polinômio de:

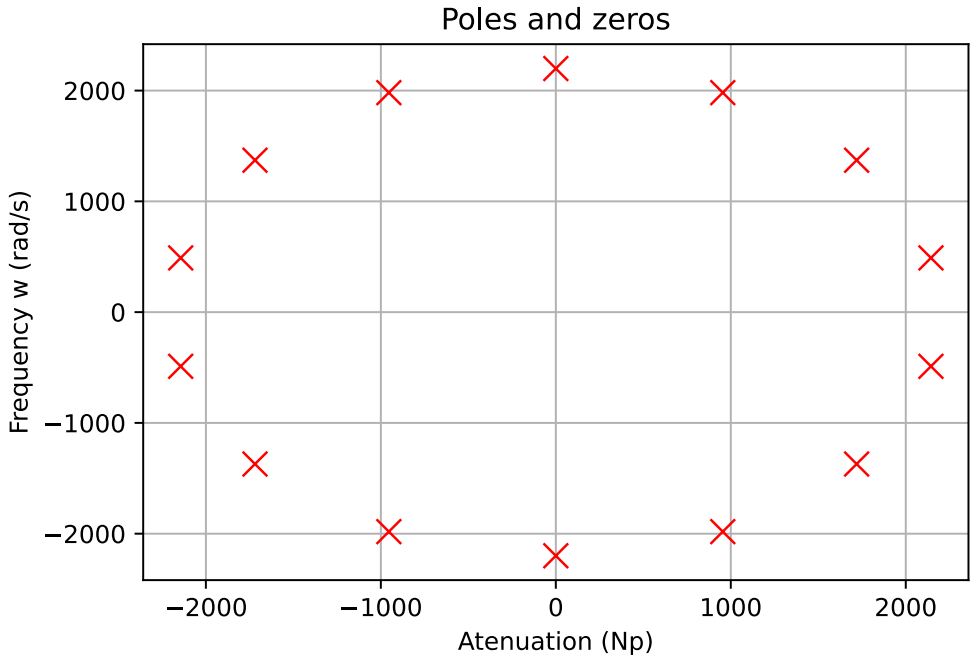
$$H(s)$$

Polos de

$$|H(s)|^2 = \frac{700\pi^{14}}{700\pi^{14} + (s)^{14}}$$

In [2]:
```
Wc = (700*pi)
num = np.poly1d([Wc**14])
# Normalizado para evitar erro numerico:
den= np.poly1d([1,0,0,0,0,0,0,0,0,0,0,0,0,0,Wc**14])

sys = signal.TransferFunction(num,den)
z,p,k = signal.tf2zpk(num,den)
#p=np.singlecomplex(wc*p)
[print(f"Polo {i} = {round(p[i])}") for i, pole in enumerate
(p)]
plot_zpk(z,p,k)
```

```
Polo 0 = (-2144+489j)
Polo 1 = (-2144-489j)
Polo 2 = (-1719+1371j)
Polo 3 = (-1719-1371j)
Polo 4 = (-954+1981j)
Polo 5 = (-954-1981j)
Polo 6 = 2199j
Polo 7 = -2199j
Polo 8 = (954+1981j)
Polo 9 = (954-1981j)
Polo 10 = (2144+489j)
Polo 11 = (2144-489j)
Polo 12 = (1719+1371j)
Polo 13 = (1719-1371j)
```



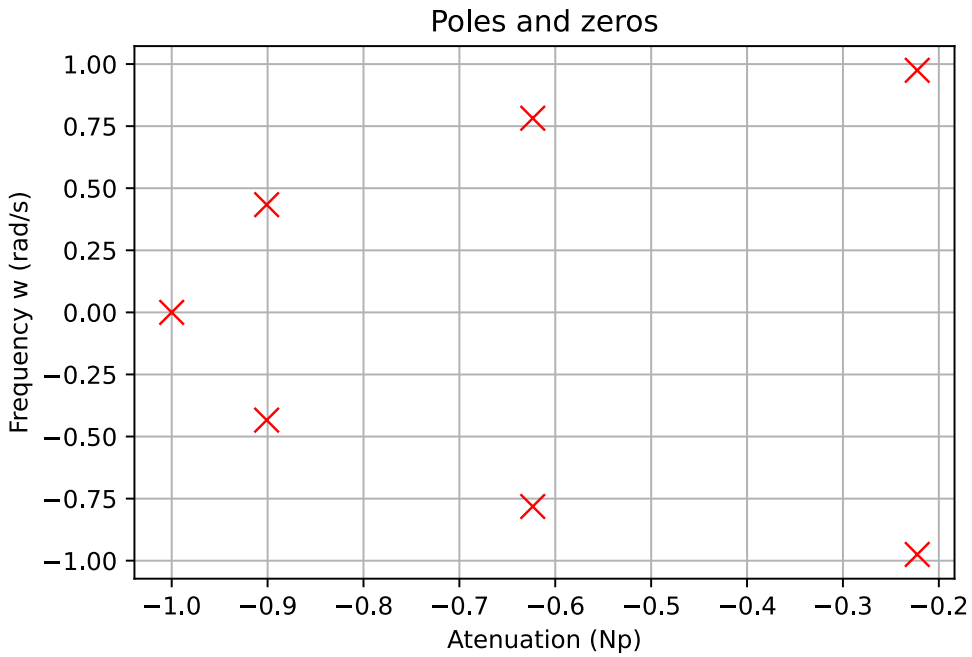
Determinação de

$$H_n(s)$$

Apenas polos do plano negativo utilizando buttap(N):

In [11]:
```
z,p,k = signal.buttap(7)
print(f"Normalizado:\nZeros=\n{zb},\nPolos=\n{pb},\nGanho=\n
{kb}")
plot_zpk(z,p,k)
```

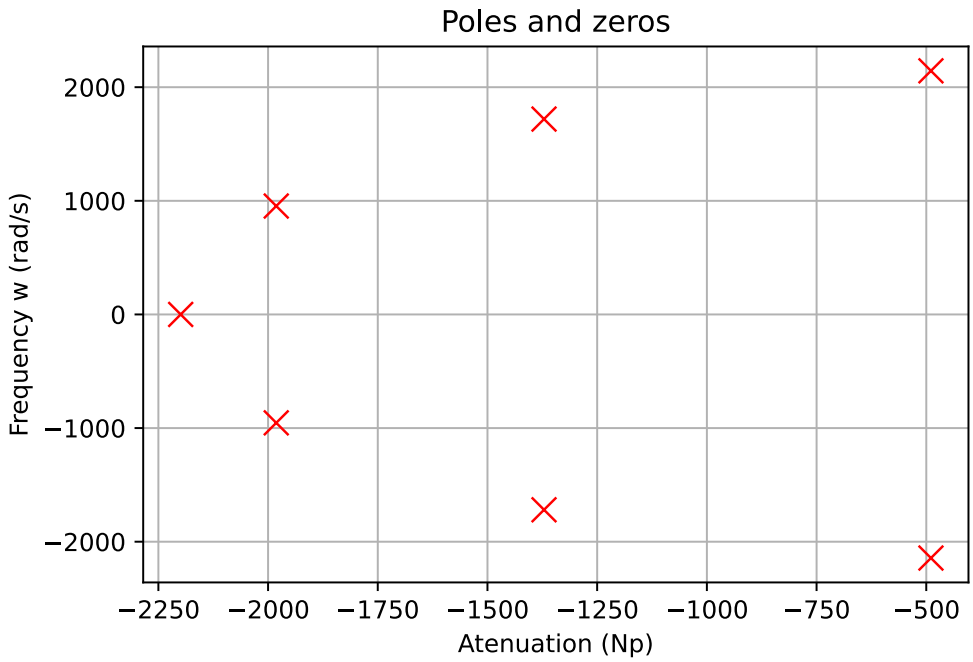
```
Normalizado:
Zeros=
[],
Polos=
[ -489.3491+2143.9785j  -1371.1257+1719.3373j  -1981.334  +95
4.16016j
 -2199.1147   -0.j      -1981.334  -954.16016j -1371.1257-171
9.3373j
  -489.3491-2143.9785j ],
Ganho=
2.4873413456829807e+23
```



In [4]:

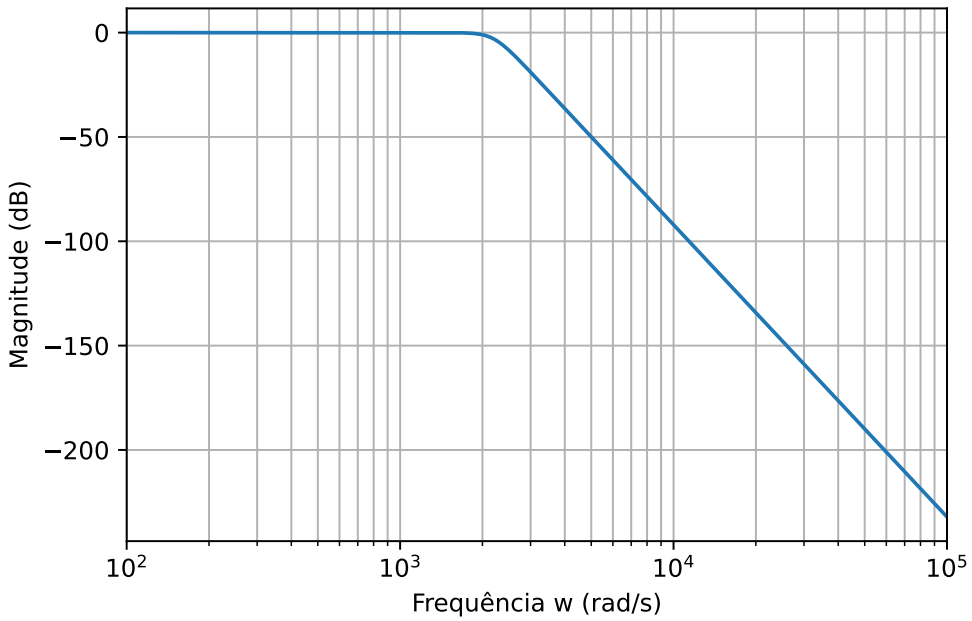
```
zb,pb,kb = signal.buttap(7)
kb = float(Wc**7)
#kb=np.format_float_scientific(kb,precision=2)
pb=np.singlecomplex(Wc*pb)
#pb = pb.round(2)

print(f"Escalonamento de freqência:\nZeros=\n{zb},\nPolos=\n
{pb},\nGanho=\n{kb}")
plot_zpk(zb,pb,kb )
```

```
Escalonamento de freqência:
Zeros=
[],
Polos=
[ -489.3491+2143.9785j  -1371.1257+1719.3373j  -1981.334  +95
4.16016j
 -2199.1147   -0.j      -1981.334  -954.16016j -1371.1257-171
9.3373j
  -489.3491-2143.9785j ],
Ganho=
2.4873413456829807e+23
```



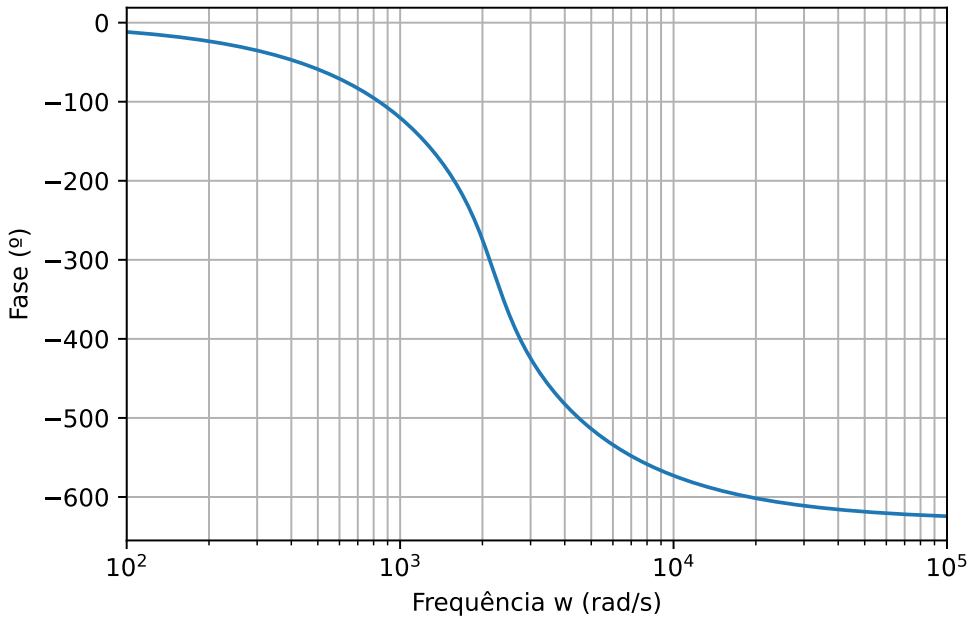
In [5]:
```
f = np.logspace(2,5,1000)
#tf_mag = lambda x: float(20*log(1/(1 + (x/(700*pi))**14), 1
0))
#mag = np.vectorize(tf_mag)(f)
f, mag, phase = signal.bode((zb,pb,kb ),f)

plt.xlim([1e2,1e5])
plt.xlabel("Frequência w (rad/s)")
#plt.ylim([-260,10])
plt.ylabel("Magnitude (dB)")
plt.grid(True, color = '0.7', linestyle='-', which='both', ax
is='both')
plt.semilogx(f, mag)    # Bode magnitude plot
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x7fdfcf8d83d0>]
```



In [6]:
```
phase2 = phase
plt.xlim([1e2,1e5])
plt.xlabel("Frequência w (rad/s)")
#plt.ylim([-260,10])
plt.ylabel("Fase (°)")
plt.grid(True, color = '0.7', linestyle='-', which='both', ax
is='both')
plt.semilogx(f, phase2)    # Bode magnitude plot
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x7fdfcfa61c70>]
```



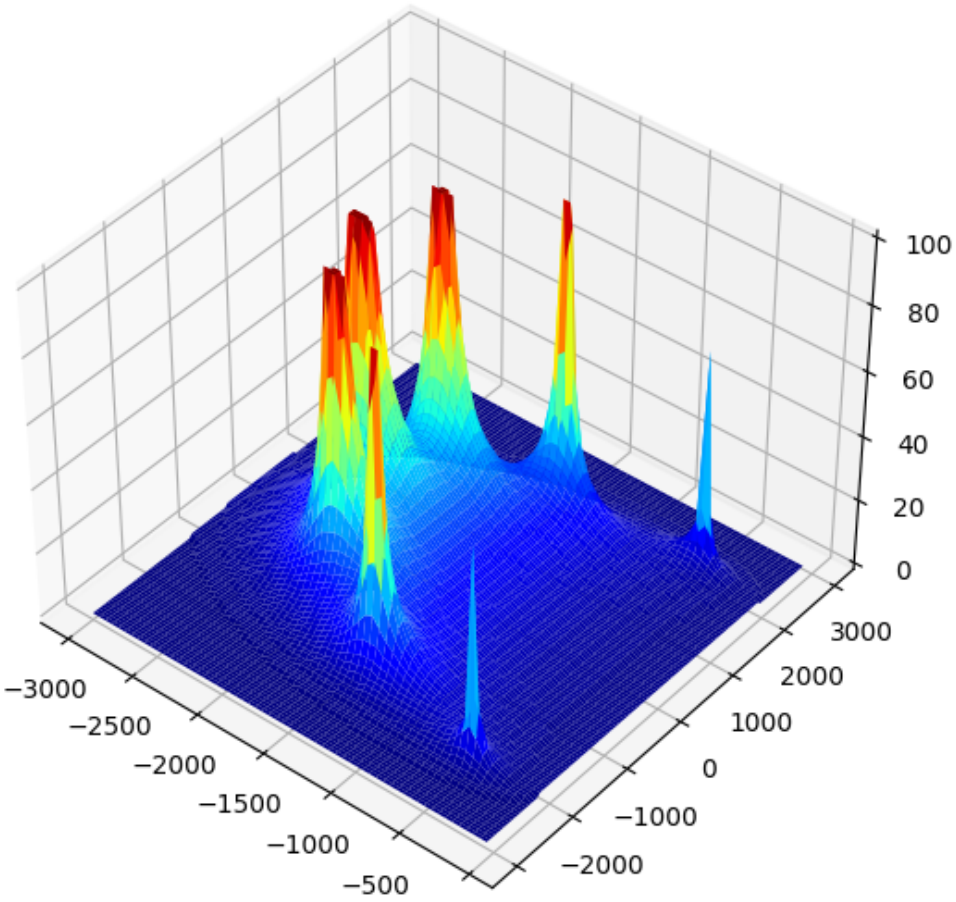
In [7]:
```
def H(s):
    return np.prod([Wc/abs(s-pk) for pk in pb])

H(complex(0,0))
```

Out[7]:
```
0.9999999768106139
```

In [8]:
```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import pylab
import numpy as np
import mpmath
mpmath.dps = 5

def plot_surf(f):
    fig = pylab.figure()
    ax = Axes3D(fig)
    X = np.arange(-3000, 0, 50)
    Y = np.arange(-3000, 3000, 50)
    X, Y = np.meshgrid(X, Y)
    xn, yn = X.shape
    W = X*0
    for xk in range(xn):
        for yk in range(yn):
            try:
                z = complex(X[xk,yk],Y[xk,yk])
                w = float(f(z))
                if w != w:
                    raise ValueError
                W[xk,yk] = w
            except (ValueError, TypeError, ZeroDivisionErro
r):
                # can handle special values here
                pass
        #print (xk, xn)

    # can comment out one of these
    #ax.set_zlim3d(0, 100)
    #ax.plot_surface(X, Y, W, rstride=1, cstride=1, cmap=cm.j
et)
    Z = np.clip(W,0,100)
    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.je
t)
    #ax.contourf(X, Y, W, zdir='z', offset=-2, cmap=plt.cm.ho
t)
    ax.set_zlim(0, 100)
    #ax.plot_wireframe(X, Y, W, rstride=5, cstride=5)

    pylab.show()
```

In [48]:
```
%matplotlib widget
```

In [33]:
```
plot_surf(H)
```



2) Utilize o filtro projetado no exercício 1 para fazer a síntese de um filtro passa-faixa com os mesmos parâmetros mas com frequência central f_0=5kHz . Novamente, reporte os valores dos polos, zeros, função de transferência, o script e as figuras do diagrama de polos e zeros e da resposta em frequência.

Escalonamento de frequência:

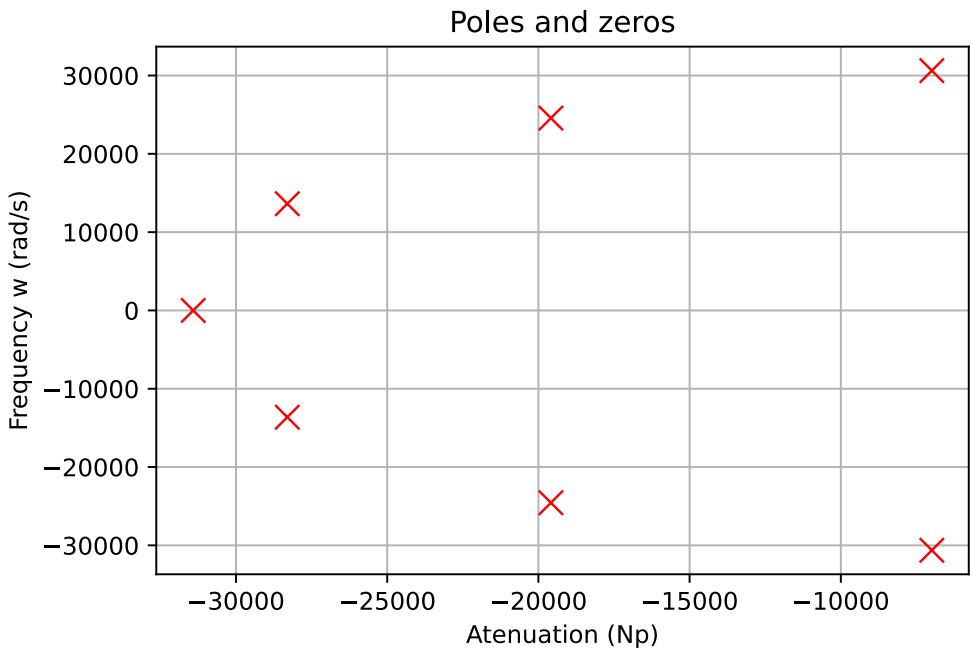
$$k_f = f_1/f_0 = 5000/350$$

In [34]:
```
Wc= float(2 * pi * 350)
kf = 5000/350
kf
```

Out[34]: 14.285714285714286

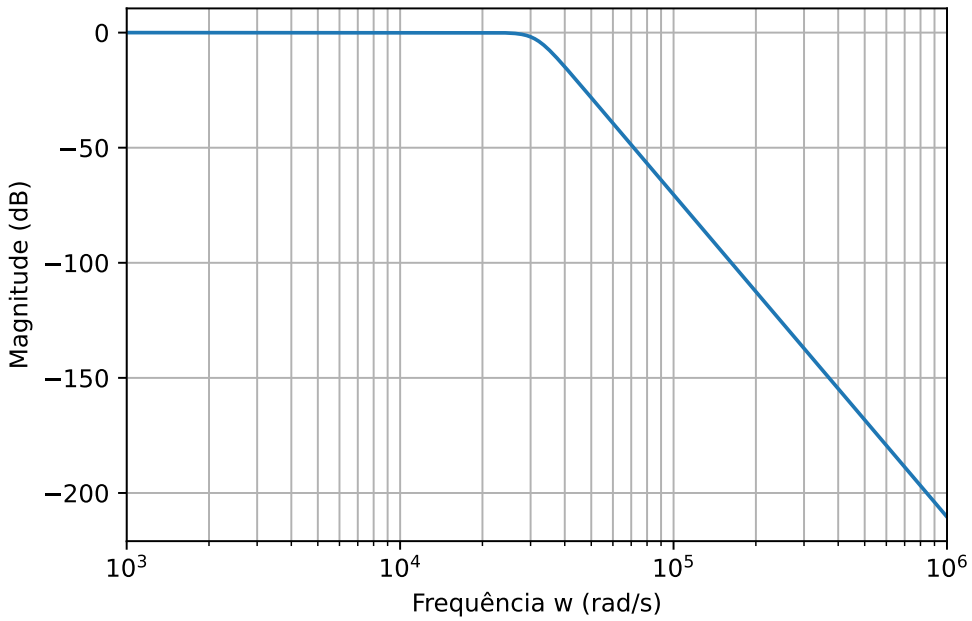
In [37]:
```
zb,pb,kb = signal.buttap(7)
kf = 5000/350
Wc= float(2 * pi * 350 * kf)
kb = float(Wc**7)
#kb=np.format_float_scientific(kb,precision=2)
pb=np.singlecomplex(Wc*pb)
#pb = pb.round(2)

print(f"Escalonamento de freqência:\nZeros=\n{zb},\nPolos=\n
{pb},\nGanho=\n{kb}\nWc={Wc}")
plot_zpk(zb,pb,kb )
```

```
Escalonamento de freqência:
Zeros=
[],
Polos=
[ -6990.701+30628.264j -19587.51 +24561.96j  -28304.771+13630.
859j
 -31415.926    -0.j    -28304.771-13630.859j -19587.51 -24561.
96j
  -6990.701-30628.264j],
Ganho=
3.0202932277767917e+31
Wc=31415.926535897932
```



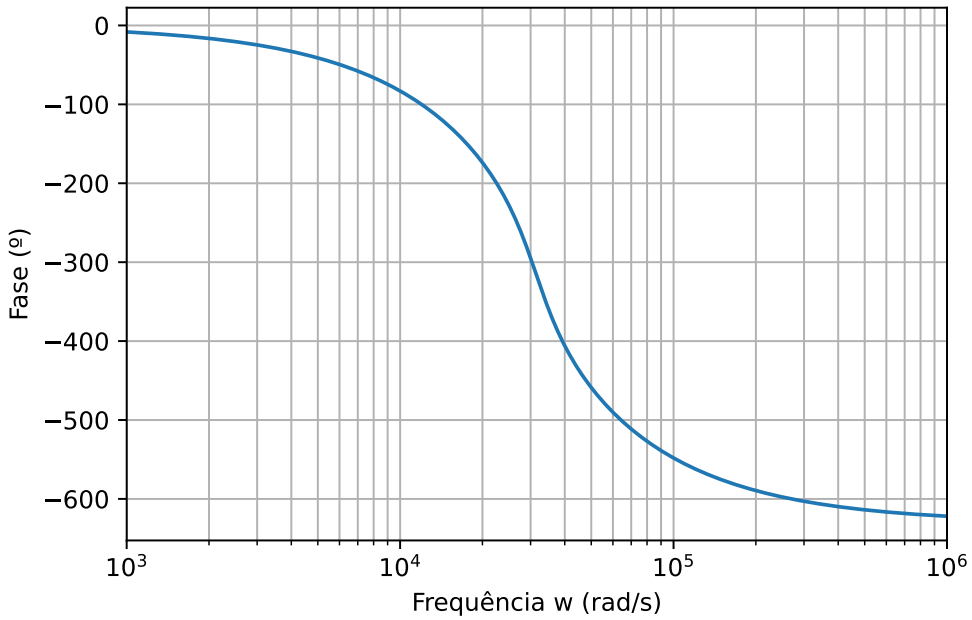
In [43]:
```
f = np.logspace(3,6,1000)
#tf_mag = lambda x: float(20*log(1/(1 + (x/(700*pi))**14), 1
0))
#mag = np.vectorize(tf_mag)(f)
f, mag, phase = signal.bode((zb,pb,kb ),f)

plt.xlim([1e3,1e6])
plt.xlabel("Frequência w (rad/s)")
#plt.ylim([-260,10])
plt.ylabel("Magnitude (dB)")
plt.grid(True, color = '0.7', linestyle='-', which='both', ax
is='both')
plt.semilogx(f, mag)    # Bode magnitude plot
```

Out[43]:
```
[<matplotlib.lines.Line2D at 0x7fdfc75405b0>]
```



In [44]:
```
phase2 = phase
plt.xlim([1e3,1e6])
plt.xlabel("Frequência w (rad/s)")
#plt.ylim([-260,10])
plt.ylabel("Fase (°)")
plt.grid(True, color = '0.7', linestyle='-', which='both', ax
is='both')
plt.semilogx(f, phase2)    # Bode magnitude plot
```

Out[44]:
```
[<matplotlib.lines.Line2D at 0x7fdfc72c5b20>]
```

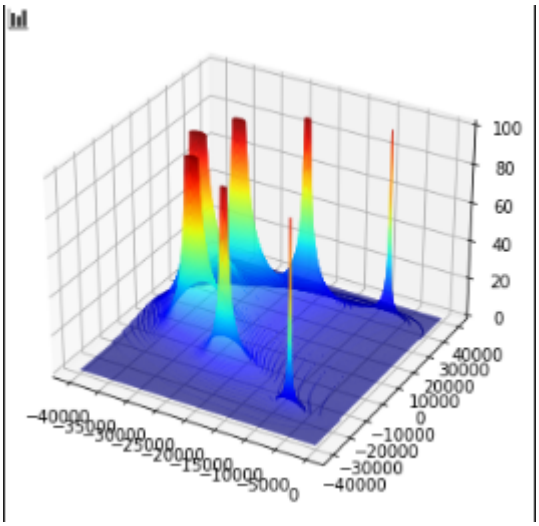


In [45]:
```
def H(s):
    return np.prod([Wc/abs(s-pk) for pk in pb])

H(complex(0,0))
```

Out[45]:
```
1.0000000259473174
```

In [46]:
```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import pylab
import numpy as np
import mpmath
mpmath.dps = 5

def plot_surf(f):
    fig = pylab.figure()
    ax = Axes3D(fig)
    X = np.arange(-40000, 0, 100)
    Y = np.arange(-40000, 40000, 100)
    X, Y = np.meshgrid(X, Y)
    xn, yn = X.shape
    W = X*0
    for xk in range(xn):
        for yk in range(yn):
            try:
                z = complex(X[xk,yk],Y[xk,yk])
                w = float(f(z))
                if w != w:
                    raise ValueError
                W[xk,yk] = w
            except (ValueError, TypeError, ZeroDivisionErro
r):
                # can handle special values here
                pass
        #print (xk, xn)

    # can comment out one of these
    #ax.set_zlim3d(0, 100)
    #ax.plot_surface(X, Y, W, rstride=1, cstride=1, cmap=cm.j
et)
    Z = np.clip(W,0,100)
    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.je
t)
    #ax.contourf(X, Y, W, zdir='z', offset=-2, cmap=plt.cm.ho
t)
    ax.set_zlim(0, 100)
    #ax.plot_wireframe(X, Y, W, rstride=5, cstride=5)

    pylab.show()
```

In [ ]:
```
plot_surf(H)
```


100
80
60
40
20
0
40000
30000
20000
10000
0
−10000
−20000
−30000
−40000
−40000
−5000
0

3) A resposta de magnitude ao quadrado de um filtro Chebyshev tipo I passa-baixas é definida por

$$|H_I(j\omega)|^2 = \frac{1}{1+\epsilon^2 C_{I,N}^2(\omega/\omega_p)},$$

em que ϵ determina o nível de *ripple* na faixa de passagem, ω_p é a frequência de borda da faixa de passagem e $C_{I,N}(\omega/\omega_p)$ é a função de Chebyshev tipo I de ordem *N*. Na forma normalizada, ou seja, para $\omega_p = 1\,rad/s$, $C_{I,N}(\omega)$ pode ser tabelado para cada *N*, tal como mostrado na tabela 1.

A resposta de magnitude ao quadrado de um filtro Chebyshev tipo II pode ser obtida a partir da tipo I fazendo-se $|H_{II}(j\omega)|^2 = 1 - |H_I(j\omega)|^2$ e substituindo-se ω/ω_p por ω_s/ω, O resultado é

$$|H_{II}(j\omega)|^2 = \frac{\epsilon^2 C_{II,N}^2(\omega_s/\omega)}{1+\epsilon^2 C_{II,N}^2(\omega_s/\omega)}$$

em que ϵ representa a atenuação na faixa de rejeição, ω_s é a frequência de borda da faixa de rejeição e $C_{II,N}(\omega_s/\omega)$ representa a *função de Chebyshev inversa* ou do *tipo II*. Na forma normalizada, $C_{II,N}(\omega_s/\omega)$ pode ser obtida a partir da tabela 1 substituindo-se ω por $1/\omega$.

Tabela 1: funções de Chebyshev em forma polinomial para $\omega_p = 1\,rad/s$ e várias ordens *N*.

n	$C_n(\omega)$
0	1
1	ω
2	$2\omega^2 - 1$
3	$4\omega^3 - 3\omega$
4	$8\omega^4 - 8\omega^2 + 1$
5	$16\omega^5 - 20\omega^3 + 5\omega$
6	$32\omega^6 - 48\omega^4 + 18\omega^2 - 1$
7	$64\omega^7 - 112\omega^5 + 56\omega^3 - 7\omega$
8	$128\omega^8 - 256\omega^6 + 160\omega^4 - 32\omega^2 + 1$
9	$256\omega^9 - 576\omega^7 + 432\omega^5 - 120\omega^3 + 9\omega$
10	$512\omega^{10} - 1280\omega^8 + 1120\omega^6 - 400\omega^4 + 50\omega^2 - 1$

Considere um filtro Chebyshev tipo II <u>*normalizado*</u> em $\omega_s = 1\,rad/s$ de ordem *N* = 4 e $\alpha_s = 60\,dB$.

a) Determine a expressão para a resposta de magnitude ao quadrado, $|H_{II}(j\omega)|^2$.
b) Determine, analiticamente, os valores dos polos e zeros.
c) Determine a função de transferência $H(s)$.
d) Faça um programa ou script do Matlab® para projetar o filtro especificado e que:
   - forneça os valores dos polos e zeros e a função de transferência *H*(s);
   - plote o diagrama de polos e zeros;
   - plote as respostas de magnitude $|H(j\omega)|$ e fase $\theta(\omega)$.

   Verifique se os valores obtidos condizem com os calculados anteriormente. Apresente, como resposta, o código do programa e os gráficos gerados.

In [4]:
```
poles = np.around(1e6 * np.array([8, -8, 1]),5)
poles
```

Out[4]:
```
array([ 8000000., -8000000.,  1000000.])
```

$$n = 4,\ \omega_p = 1\ rad/s, \alpha_p = 60\ dB$$

$$C_n = 8\omega^5 - 8\omega^2 + 1$$

$$\epsilon = \sqrt{10^{\alpha/10} - 1} = 0.99762834511$$

$$|H(s)|^2 = \frac{1}{1 + \epsilon^2 * (8\omega^4 - 8\omega^2 + 1)}$$

$$\epsilon^2 = 10^{\alpha/10} - 1 = 0.99526231496$$

$$|H(s)|^2 = \frac{1}{1 + 0.99526231496 * (8\omega^4 - 8\omega^2 + 1)}$$

$$|H(s)|^2 = \frac{1}{1.99526231496 + 7.96209851975(\omega^4 - \omega^2)}$$

In [28]:
```
p = np.roots([7.96209851975, 0, -7.96209851975, 0, 1.99526231
496])
k=1/7.96209851975
p
```

Out[28]:
```
array([-0.707317+0.01724349j, -0.707317-0.01724349j,
        0.707317+0.01724349j,  0.707317-0.01724349j])
```

$$|H(s)|^2 = \frac{1}{1.99526231496 + 7.96209851975(\omega^4 - \omega^2)}$$

$$\omega_p = [-0.707316999672518 - 0.0172434922719683 * I,$$

$$-0.707316999672518 + 0.0172434922719683 * I,$$
$$0.707316999672518 - 0.0172434922719683 * I,$$
$$0.707316999672518 + 0.0172434922719683 * I]$$

Para H(s):

$$\omega_p = [-0.707316999672518 - 0.0172434922719683 * I,$$
$$-0.707316999672518 + 0.0172434922719683 * I]$$

$$H(s) = \frac{1}{\prod(s - p_k)}$$

In [32]:
```
z=[]
tf = signal.zpk2tf(z,p[0:2],k)
tf
```

Out[32]:
```
(array([0.000125]), array([1.        , 1.414634  , 0.5005946
8]))
```

$$H(s) = \frac{0.000125}{\omega^2 + 1.414634\omega^2 + 0.50059468}$$

In [24]:
```
z, p, k  = signal.cheb1ap(4, 60)
tf = signal.zpk2tf(z,p,k)
num, den = signal.zpk2tf(z, p, k)
sys = signal.TransferFunction(num, den)
sys
```

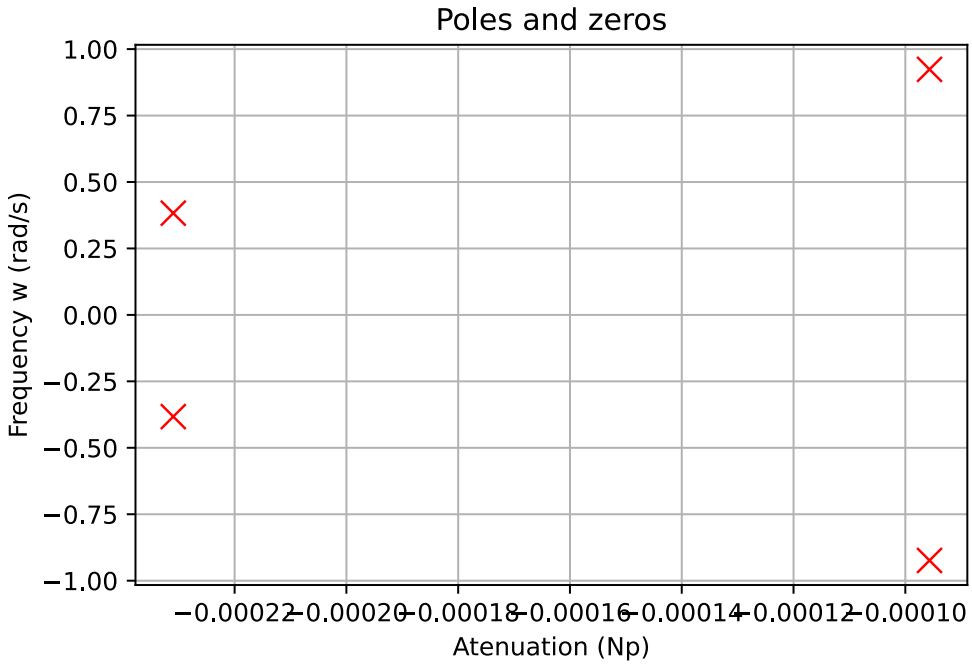
Out[24]:
```
TransferFunctionContinuous(
array([0.000125]),
array([1.00000000e+00, 6.53281707e-04, 1.00000021e+00, 4.22311
785e-04,
       1.25000063e-01]),
dt: None
)
```

Função de transferencia:

$$T_f = \frac{0.000125}{1.0x^4 + 0.000653x^3 + 1.0x^2 + 0.000422x + 0.125}$$

Os coeficientes do denominador diferem em $0.000653x^3 + 0.000422x$que são termos que podem ser desprezados. Pode-se concluir que a implementação de cheblap é uma aproximação do polinomio real.

In [6]:
```
plot_zpk(z,p,k)
```

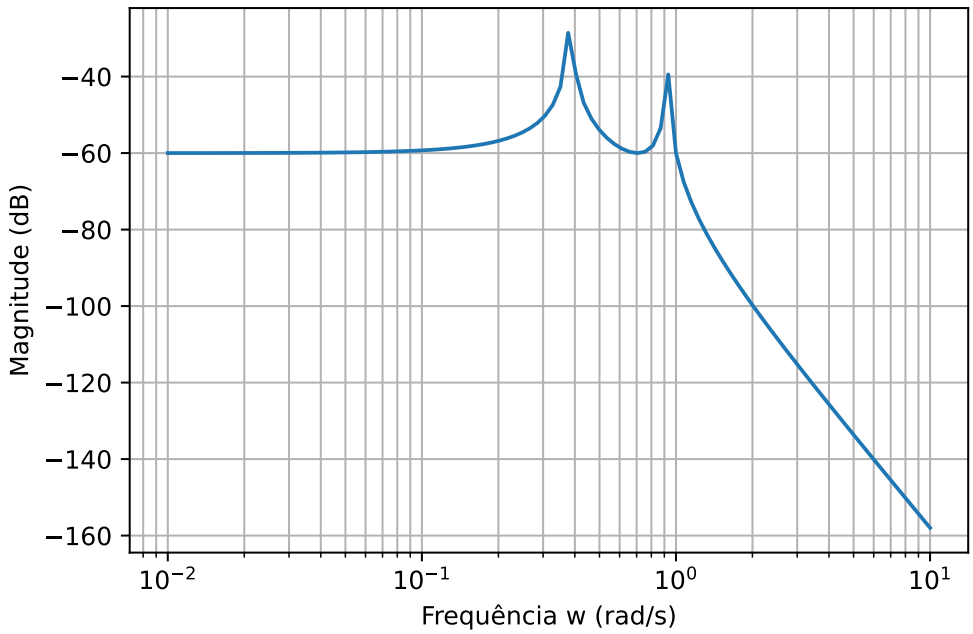


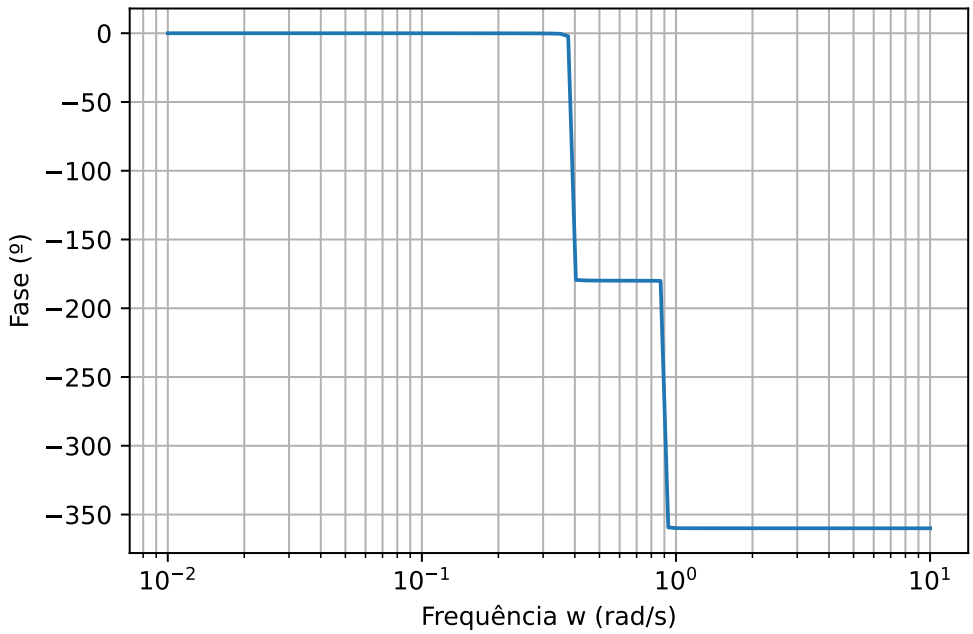
In [9]:
```
from  sympy import *
den_r = den.round(6)
num_r = num.round(6)
x = symbols('x')
tf = Poly(num_r, x)/Poly(den_r, x)
```

In [21]:

```
f = np.logspace(3,6,1000)
#tf_mag = lambda x: float(20*log(1/(1 + (x/(700*pi))**14), 1
0))
#mag = np.vectorize(tf_mag)(f)
f, mag, phase = signal.bode((z,p,k ))
plt.figure()
#plt.xlim([1e3,1e6])
plt.xlabel("Frequência w (rad/s)")
#plt.ylim([-260,10])
plt.ylabel("Magnitude (dB)")
plt.grid(True, color = '0.7', linestyle='-', which='both', ax
is='both')
plt.semilogx(f, mag)    # Bode magnitude plot


plt.figure()
phase2 = phase
#plt.xlim([1e3,1e6])
plt.xlabel("Frequência w (rad/s)")
#plt.ylim([-260,10])
plt.ylabel("Fase (°)")
plt.grid(True, color = '0.7', linestyle='-', which='both', ax
is='both')
plt.semilogx(f, phase2)    # Bode magnitude plot
```

Out[21]: [<matplotlib.lines.Line2D at 0x7f6c1b8b7370>]





4) Filtros Chebyshev tipo II são naturalmente especificados em função da frequência de borda da faixa de rejeição, ω_s . Para se fazer um projeto em termos da frequência de borda da faixa de passagem, ω_p, é necessário determinar qual será a ω_s correspondente, o que pode ser feito por meio da expressão

$$\omega_s = \omega_p \cosh\left\{(1/N)\, acosh\left[1/(\epsilon\sqrt{10^{\alpha_p/10}-1})\right]\right\}.$$

em que α_p representa o ripple na faixa de passagem, em dB.
Considerando essas informações, faça um programa ou script do Matlab® para projetar um filtro Chebyshev passa-baixas tipo II com $\omega_p = 2\pi 500\, rad/s$, $\alpha_p = 0,5\, dB$ e $\alpha_s = 60\, dB$.
O programa deverá solicitar ao usuário a ordem desejada e retornar as seguintes informações:

a) Frequência de borda da faixa de rejeição ω_s corrigida.
b) Polos, zeros e diagrama correspondente.
c) Função de transferência $H(s)$.
d) As respostas de magnitude $|H(j\omega)|$ e fase $\theta(\omega)$.

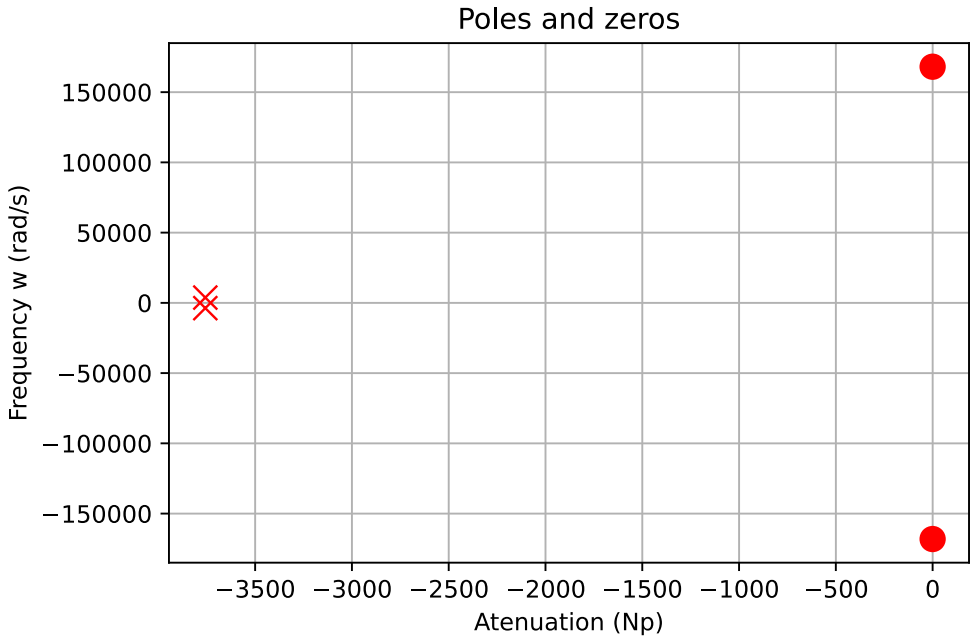
Apresente, como resposta, o código do programa e os gráficos gerados.

In [54]:

```
print('Insira a ordem do filtro passa baixas:\n\n')
n=2
#n = input();

wp = 1000*pi;
Rp = 0.5;
Rs = 60;

eps = 1/(sqrt(10**(Rs/10)-1));

ws = wp*cosh((1/n)*acosh(1/(eps*(sqrt(10**(Rp/10)-1)))))
print(ws)
num, den = cheby2(n, Rs, ws, analog=True)
print(num)
print(den)

z = np.roots(num)
p = np.roots(den)
k=1
print(z)
print(p)
print(k)
plot_zpk(z,p,k)
H = signal.TransferFunction(num, den)

figure (1)

figure (2)
signal.bode(z,p,k)
```

```
Insira a ordem do filtro passa baixas:


37840.2809837298*pi
[1.00000000e-03 0.00000000e+00 2.82643138e+07]
[1.00000000e+00 7.51479201e+03 2.82643138e+07]
[-0.+168119.93875773j  0.-168119.93875773j]
[-3757.39600615+3761.15528274j -3757.39600615-3761.15528274j]
1
```

Out[54]:
```
(array([-3757.39600615+3761.15528274j, -3757.39600615-3761.155
28274j]),
 array([0., 0.]),
 array([180., 180.]))
```




```
<Figure size 432x288 with 0 Axes>
```

5) Considere o desenvolvimento de um filtro elíptico passa-altas com ganho unitário e frequência de borda da faixa de passagem $f_p = 2\,kHz$. Para garantir um nível mínimo de exatidão na aplicação, a distorção máxima aceitável na faixa de passagem é de 1,5%. O filtro deve fornecer uma atenuação mínima na faixa de rejeição de 99,8%, em uma frequência máxima de $1{,}35\,kHz$. Faça um programa ou script do Matlab® para projetar o filtro especificado e que:
   - determine a ordem mínima N que atenda as especificações;
   - forneça os valores dos polos, zeros e a função de transferência $H(s)$;
   - plote o diagrama de polos e zeros;
   - plote as respostas de magnitude $|H(j\omega)|$ e fase $\theta(\omega)$.

   Apresente, como resposta, o código do programa e os gráficos gerados.

In [ ]:

6) Considere o desenvolvimento de um filtro Chebyshev tipo I com ordem N = 3 utilizando um circuito passivo em escala *LC*.
   a) Considerando o filtro normalizado em $\omega_p = 1 rad/s$, consulte os dados da tabela 1, acima, e determine a expressão para a resposta de magnitude ao quadrado na forma polinomial,

$$|H(j\omega)|^2 = \frac{K^2}{1+\epsilon^2\left[\sum_{k=1}^{N} a_k \omega^k\right]^2}$$

   b) Determine a expressão para a impedância de entrada $Z_{11}(s)$ (consulte o exemplo apresentado nos slides da aula 05_B).
   c) Considerando $\alpha_p = 0{,}25\,dB$, $R_S = 0{,}5\,\Omega$ e $R_L = 1\,\Omega$, utilize o método do fracionamento contínuo para obter os dois possíveis circuitos em escada *LC*. Compare os resultados com os apresentados na tabela 10.3, disponível nos slides da aula 06_A. Corrija eventuais discrepâncias.
   d) Faça os escalonamentos para que o filtro apresente uma resposta passa-altas com frequência de borda da faixa de passagem $f_p = 300\,Hz$ alimentando uma impedância de carga $R_L = 50\,\Omega$.
   e) Monte o circuito em um simulador SPICE[1] e apresente o diagrama do circuito e os gráficos de resposta de magnitude $|H(j\omega)|$ e fase $\theta(\omega)$. Corrija eventuais discrepâncias.

$$|H(s)|^2 = \frac{1}{1 + \epsilon^2 * (4w^3 - 3w)}$$

7) Considerando o que foi discutido a respeito das duas formas de implementação de filtros LIT de tempo contínuo, responda:
   a) Indique, pelo menos, três vantagens que a implementação com filtros ativos apresenta em relação à implementação passiva.
   b) Indique, pelo menos, duas áreas de aplicação em que a implementação passiva é desejável ou obrigatória.

a) Filtros Ativos permitem ganho maior que 1; possuem caracteristicas de amplificadores ideiais: alta impedância de entrada, baixa impedância de saída, o que permite concatenar vários estágios, sem perda de ganho.

b) Passivos são necessários onde uma a faixa de resposta é muito larga, como em aplicações RF e nanoeletrônica/ processadores, sendo que em ambos a faixa de operação é em GHz. Isso se deve ao fato de amplificadores possuirem uma banda de passagem mais restrita que componentes discretos.

8) Faça o projeto de um filtro passa-faixa Chebyshev tipo II com as seguintes especificações: $f_0 = 10\,kHz$, $Bp = 1\,kHz$, $Bs = 9\,kHz$, $\alpha_p = 2\,dB$, $\alpha_s = 40\,dB$. Apresente o seu projeto da seguinte forma:
   a) Utilize o Matlab® para determinar e reportar a mínima ordem N que atenda as especificações.

   b) Utilize o Matlab® para obter a função de transferência na forma original e em expansão em termos de 1ª e 2ª ordem.
   c) Implemente o circuito em um simulador SPICE. Utilize um fator de escala de impedância $k_Z = 1000$.
   Apresente, como resposta, o código do programa, o diagrama do circuito e os gráficos de resposta de magnitude $|H(j\omega)|$ e fase $\theta(\omega)$. Corrija eventuais discrepâncias.

In [58]:
```
wp = 1000*pi
Rp = 2
Rs = 40

eps = 1/(sqrt(10**(Rs/10)-1))
for n in range(1,20):
    Bs = wp*cosh((1/n)*acosh(1/(eps*(sqrt(10**(Rp/10)-1)))))
/(2*pi)
    print( f"n={n}, Bs= {Bs}")
    if Bs < 9e3:
        break
print( f"n={n}")
```

```
n=1, Bs= 65374.7445965484
n=2, Bs= 4058.16290322815
n=2
```