AP1 - Portos

Hidráulica Marítimica

Aluno: Francisco José Matos Nogueira Filho

Matricula: 384962

In [1]:
```
import numpy as np
from numpy import pi
from IPython.display import Markdown as md
```

Questão 1

Dados da onda em Alto Mar

In [2]:
```
## Dados da onda em Alto Mar

H0 = 13 #m
T = 15 #s
g = 9.81 #m/s^2
```

Dados da Bacia

In [3]:
```
L = 150 #m
cim = 250 #m
cib = 450 #m
pu = 16 #m
```

Item A

Determinação do comprimento de onda L_0, celeridade e σ em águas profundas

$$L_0 = \frac{gT^2}{2\pi}$$

$$C = \frac{L}{T}$$

$$\sigma = \frac{2\pi}{T}$$

In [4]:
```
def comprimentoDeOnda(Periodo):
    return (g * Periodo**2)/(2 * pi)

def Celeridade(Comprimento,T):
    return Comprimento/T

L0 = comprimentoDeOnda(T)
C0 = Celeridade(L0,T)
sigma = 2*pi/T


#Resposta
md("""
$L_0 = %.2f$ m \n
$C_0 = %.2f$ m/s \n
$\sigma = %.4f$ ciclos/seg
"""%(L0,C0,sigma))


k = 0.78
```

Usarei uma função `broyden1` do pacote de calculo númerico scipy (https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.broyden1.html) para minimizar a expressão:

$$\frac{L}{L0} = tanh\left(\frac{2\pi d}{L}\right)$$

In [5]:
```
from scipy.optimize import broyden1
#from scipy.optimize import minimize, newton_krylov, broyden1, curve_fit
d1 = pu #m
def fu(l):
    return (l/L0 - np.tanh((2*pi*d1)/l))**2
x0 = 125 #Palpite inicial
solv = broyden1(fu,x0,iter=50)
L16 = solv

def calcularK(L):
    return (2 * pi)/L

k16 = calcularK(L16)
md("""
$L = %.2f \; m $ \n
$K = %.5f \; $
""" % (L16,k16))
```

Out[5]: $L = 178.93\ m$

$K = 0.03512$

$$n = 0.5\left(1 + \frac{2Kd}{senh(2Kd)}\right)$$

In [6]:
```
def calcularN(K,d):
    return 0.5 * (1 + (2*K*d)/np.sinh(2*K*d))

n16 = calcularN(k16,d1)
md("""
$n = %.3f$
""" % n16)
```

Out[6]: $n = 0.908$

$$h = H_0\sqrt{\frac{b_0}{b}}\sqrt{\frac{1}{2}\frac{1}{n}\frac{C_0}{C}}$$

Neste caso, deve-se considerar $\frac{b_0}{b} \neq 1$ uma vez que se estamos calculando para um cenário em Mar Fechado

In [7]:
```
def calcularH(H0,n,C0,C,b0_b=1):
    return H0 * b0_b**0.5*(1 * 0.5* 1/n * C0/C)**0.5
def VelocidadeOrbitalMax(a,K,sigma):
    return (a * g * K)/sigma
C16 = Celeridade(L16,T)
b0_b = L0/L
Hbacia = calcularH(H0,n16,C0,C16, b0_b)
abacia = Hbacia/2
ubacia = VelocidadeOrbitalMax(abacia,k16,sigma)
md("""
$H = %.3f \; m $ \n
$u = %.3f \; m/s$
""" % (Hbacia, ubacia))
```

Out[7]: $H = 20.681\ m$

$u = 8.504\ m/s$

Item B

Pode-se considerar $\frac{b_0}{b} = 1$ uma vez que se estamos calculando para um cenário em Mar Aberto

In [8]:
```
C16 = Celeridade(L16,T)
b0_b = 1
H16 = calcularH(H0,n16,C0,C16,b0_b)
a16 = H16/2
u16 = VelocidadeOrbitalMax(a16,k16,sigma)
md("""
$H = %.3f \; m $ \n
$u = %.3f \; m/s$
""" % (H16, u16))
```

Out[8]: $H = 13.514\ m$

$u = 5.557\ m/s$

Item C

$$T = \frac{2}{\sqrt{gd}}\left[\left(\frac{1}{L_A}\right)^2 + \left(\frac{1}{L_B}\right)^2\right]^{-\frac{1}{2}}$$

In [9]:
```
def PedriodoFundamentalDeOscilacao(d,La,Lb):
    return 2/(g*d)**0.5 * ((1/La)**2 + (1/Lb)**2)**-0.5

Tbacia = PedriodoFundamentalDeOscilacao(d1,cim,cib)

md("""
$T_{bacia} = %.3f \; seg $
""" % Tbacia)
```

Out[9]: $T_{bacia} = 34.887\ seg$

In [10]:
```
if Tbacia > T:
    print("Não haverá possibilidade de agitação por ressonância dentro da bacia portuária !")

else:
    print("Haverá possibilidade de agitação por ressonância dentro da bacia portuária !")
```

```
Não haverá possibilidade de agitação por ressonância dentro da bacia portuária !
```