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 = 2 #m
T = 142 #s
g = 9.81 #m/s^2
```

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 [3]:
```
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))
```

Out[3]: $L_0 = 31482.25$ m

$C_0 = 221.71$ m/s

$\sigma = 0.0442$ ciclos/seg

Usarei uma função `broyden1` do pacote de calculo númerico [scipy (https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.broyden1.html)](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 [4]:
```
from scipy.optimize import broyden1
#from scipy.optimize import minimize, newton_krylov, broyden1, curve_fit
d1 = 10 #m
def fu(l):
    return (l/L0 - np.tanh((2*pi*d1)/l))**2
x0 = 125 #Palpite inicial
solv = broyden1(fu,x0,iter=50)
L10 = solv

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

k10 = calcularK(L10)
md("""
$L = %.2f \; m $ \n
$K = %.5f \; $
""" % (L10,k10))
```

Out[4]: $L = 1405.98 \; m$

$K = 0.00447$

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

In [5]:
```
def calcularN(K,d):
    return 0.5 * (1 + (2*K*d)/np.sinh(2*K*d))

n10 = calcularN(k10,d1)
md("""
$n = %.3f$
""" % n10)
```

Out[5]: $n = 0.999$

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

Pode-se considerar $\frac{b_0}{b} = 0$ uma vez que se estamos calculando para um cenário em Mar Aberto

In [6]:
```
def calcularH(H0,n,C0,C):
    return H0 * (1 * 0.5* 1/n * C0/C)**0.5
def VelocidadeOrbitalMax(a,K,sigma):
    return (a * g * K)/sigma
C10 = Celeridade(L10,T)
H10 = calcularH(H0,n10,C0,C10)
a10 = H10/2
u10 = VelocidadeOrbitalMax(a10,k10,sigma)
md("""
$H = %.3f \; m $ \n
$u = %.3f \; m/s$
""" % (H10, u10))
```

Out[6]: $H = 6.694\ m$

$u = 3.316\ m/s$

Item B

Usarei uma função `broyden1` do pacote de calculo númerico [scipy (https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.broyden1.html)](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 [7]:
```
from scipy.optimize import minimize
from scipy.optimize import minimize, newton_krylov, broyden1, curve_fit
d2 = 5 #m
def fu2(l):
    return (l/L0 - np.tanh((2*pi*d2)/l))**2
x0 = 125 #Palpite inicial
solv = broyden1(fu2,x0,iter=50)
L5 = solv

k5 = calcularK(L5)
md("""
$L = %.2f \; m $ \n
$K = %.5f \; $
""" % (L5,k5))
```

Out[7]: $L = 994.34\ m$

$K = 0.00632$

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

In [8]:
```
def calcularN(K,d):
    return 0.5 * (1 + (2*K*d)/np.sinh(2*K*d))

n5 = calcularN(k5,d2)
md("""
$n = %.7f$
""" % n5)
```

Out[8]: $n = 0.9996674$

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

Pode-se considerar $\frac{b_0}{b} = 0$ uma vez que se estamos calculando para um cenário em Mar Aberto

In [9]:
```
C5 = Celeridade(L5,T)
H5 = calcularH(H0,n5,C0,C5)
a5 = H5/2
u5 = VelocidadeOrbitalMax(a5,k5,sigma)
md("""
$H = %.3f \; m $ \n
$u = %.3f \; m/s$
""" % (H5, u5))
```

Out[9]: $H = 7.959\ m$

$u = 5.575\ m/s$

Item C

$$sen(\alpha_2) = \frac{C_2}{C_1} sen(\alpha_1)$$

In [10]:
```
def AnguloDeRefracao(C1, C2, alfa):
    return C2/C1* np.sin(2* pi* alfa/360)
alfa1 = 35
senAlfa2 = AnguloDeRefracao(C10,C5,alfa1)
alfa2 = 360*np.arcsin(senAlfa2)/(2 * pi)
md("""
$ sen(α_2) = %.3f $ \n
$ α_2 = %.2f ° $

""" % (senAlfa2,alfa2))
```

Out[10]: $sen(\alpha_2) = 0.406$

$\alpha_2 = 23.93°$

Item D

$$\xi_M = \frac{\pi^2 H^2}{2gT^2}\left[1 + \frac{1.5}{senh\left(\frac{2\pi d}{L}\right)}\right]$$

In [11]:
```
def sobrelevacaoNvlMedio(H, T, d, L):
    return (pi*H)**2/(2*g*T**2) * (1 + 1.5/np.sinh(2* pi * d/L))

xi5 = sobrelevacaoNvlMedio(H5, T, d2, L5)

md("""
$ξ = %.5f  \; m$

""" % xi5)
```

Out[11]: $\xi = 0.07659 \; m$

Item E

In [12]:
```
cotaIgreja = 3.5 #m

Honda = H5/2 + xi5


if Honda > cotaIgreja:
    print("A Onda atingirá a Igreja")
else:
    print("A Igreja não será atingida pela Onda")
```

```
A Onda atingirá a Igreja
```

Questão 1

Dados da onda em Alto Mar

In [13]:
```
## Dados da onda em Alto Mar

H0 = 6 #m
T = 8 #s
g = 9.81 #m/s^2
```

Item A

In [14]:
```
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))
```

Out[14]: $L_0 = 99.92$ m

$C_0 = 12.49$ m/s

$\sigma = 0.7854$ ciclos/seg

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 [15]:
```
from scipy.optimize import broyden1
#from scipy.optimize import minimize, newton_krylov, broyden1, curve_fit
d1 = 7 #m
def fu(l):
    return (l/L0 - np.tanh((2*pi*d1)/l))**2
x0 = 50 #Palpite inicial
solv = broyden1(fu,x0,iter=50)
L7 = solv

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

k7 = calcularK(L7)
md("""
$L = %.2f \; m $ \n
$K = %.5f \; $
""" % (L7,k7))
```

Out[15]: $L = 61.41\ m$

$K = 0.10232$

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

In [16]:
```
def calcularN(K,d):
    return 0.5 * (1 + (2*K*d)/np.sinh(2*K*d))

n7 = calcularN(k7,d1)
md("""
$n = %.3f$
""" % n7)
```

Out[16]: $n = 0.863$

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

Pode-se considerar $\frac{b_0}{b} = 0$ uma vez que se estamos calculando para um cenário em Mar Aberto

In [17]:
```
def calcularH(H0,n,C0,C):
    return H0 * (1 * 0.5* 1/n * C0/C)**0.5
def VelocidadeOrbitalMax(a,K,sigma):
    return (a * g * K)/sigma
C7 = Celeridade(L7,T)
H7 = calcularH(H0,n7,C0,C7)
a7 = H7/2
u7 = VelocidadeOrbitalMax(a7,k7,sigma)
md("""
$H = %.3f \; m $ \n
$u = %.3f \; m/s$
""" % (H7, u7))
```

Out[17]: $H = 5.827 \; m$

$u = 3.723 \; m/s$

Item B

$$\Delta p = \frac{H}{cosh\left(\frac{2\pi d}{L}\right)}$$

$$\Delta h = \frac{\pi H^2}{L} coth\left(\frac{2\pi d}{L}\right)$$

In [18]:
```
def DeltaH(H, L, d):
    return (pi * H**2)/L * (np.tanh((2 * pi * d)/L))**-1
def DeltaP(H,L,d):
    return H/(np.cosh((2 * pi * d)/L))
dP = DeltaP(H7, L7, d1)
dH = DeltaH(H7, L7, d1)
md("""
$\Delta p = %.2f \; m$ \n
$\Delta h = %.2f \; m$
""" % (dP,dH))
```

Out[18]: $\Delta p = 4.60 \; m$

$\Delta h = 2.83 \; m$

In [19]:
```
md("""
$H_{clapotis} = %.2f \; m$
""" % (H7 * 2))
```

Out[19]: $H_{clapotis} = 11.65 \; m$

In [20]:
```
md("""
$Cota = %.2f \; m$
""" % (H7 + dH))
```

Out[20]: $Cota = 8.65 \; m$

Item C

$$\bar{E} = EL = \frac{\rho g H^2 L}{8}$$

In [21]:
```
rho = 1034
H = 2 * H7
def EnergiaDeOnda(H, L):
    return (rho * g * H**2 * L)/8

E = EnergiaDeOnda(H, L7)
# Para comprimento de crista = 1m
md("""$E = %.2f \; Joules$ \n
    ou \n $E = %.2f \; tf·m $  """%(E, E/(9.81*1000)))
```

Out[21]: $E = 10574940.14 \; Joules$

ou

$E = 1077.98 \; tf \cdot m$

Item D

In [22]:
```
def MomentoDeTombamento():
    return

cota = H7 + dH
A1 = 0.5 * cota**2
A2 = d1*(cota - dP)/2
A3 = dP * d1
At = sum([A1,A2,A3])

x1 = d1 + cota/3
x2 = 2/3 *d1
x3 = d1/2

x = (np.array([A1,A2,A3]) @ np.array([x1,x2,x3]))/At
md("""
$x = %.2f \; m $
""" % x)
```

Out[22]: $x = 6.55 \; m$

In [23]:
```
R = At * rho/1000
Mt = R * x
md("""
$M_t = %.2f \; tf·m/m_{linear}$
""" % Mt)
```

Out[23]: $M_t = 567.62 \; tf \cdot m/m_{linear}$

In [24]:
```
R
```

Out[24]: 86.66562384202894