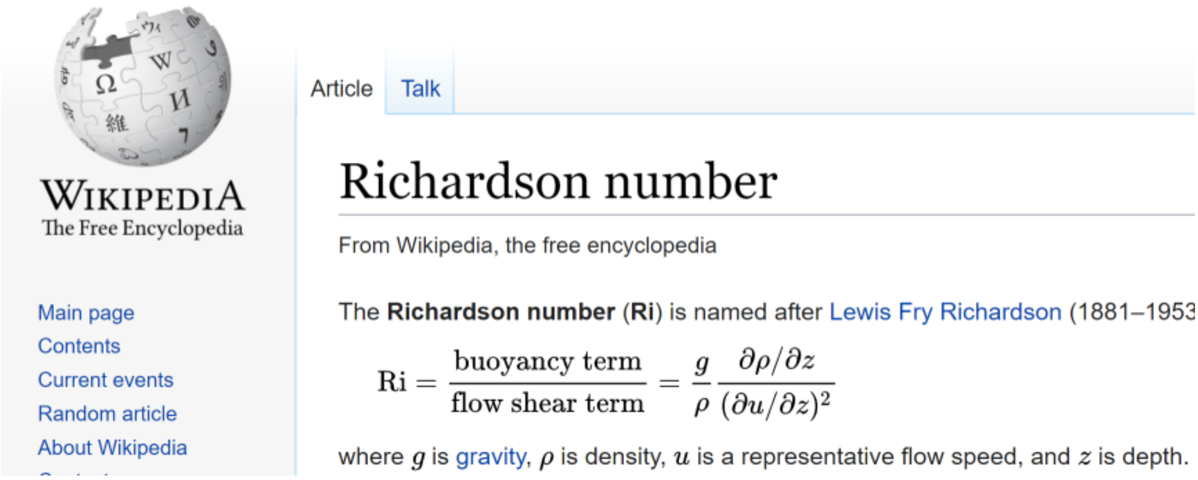
Número de Richardson (Gradiente)

O Número de Richardson é um número adimensional que relaciona estratificação com cisalhamento, e indica se a estratificação da coluna de água é estável.

É muito utilizado em estudos de hidrodinâmica costeira, onde quase sempre há alguma fonte de empuxo, especialmente aporte de água doce.


Wikipedia
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia

Article Talk

Richardson number

From Wikipedia, the free encyclopedia

The **Richardson number** (**Ri**) is named after Lewis Fry Richardson (1881–1953

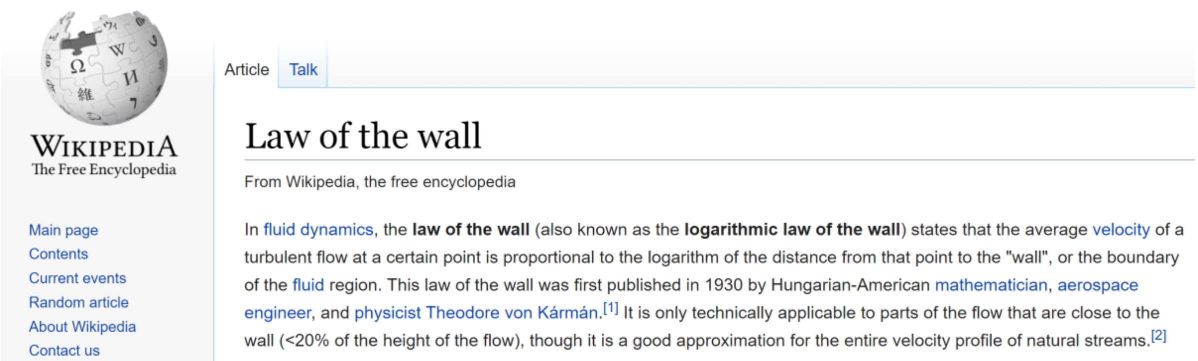
$$\mathrm{Ri} = \frac{\text{buoyancy term}}{\text{flow shear term}} = \frac{g}{\rho}\frac{\partial\rho/\partial z}{(\partial u/\partial z)^2}$$

where g is gravity, ρ is density, u is a representative flow speed, and z is depth.

In [18]:
```
# OBS: as células de código tem um número do lado esquerdo entre [], e nestas célula
# Carregando os pacotes básicos... Numpy para fazer operções numéricas e o Matplotlb

import numpy as np
import matplotlib.pyplot as plt
```

Para calcularmos o Ri precisamos de alguns números... um perfil vertical de corretnes e um perfil vertical de densidade.

Para o perfil vertical de correntes podemos usar a Lei da Parede para fornecer um perfil minimamente realista.


Wikipedia
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us

Article Talk

Law of the wall

From Wikipedia, the free encyclopedia

In fluid dynamics, the **law of the wall** (also known as the **logarithmic law of the wall**) states that the average velocity of a turbulent flow at a certain point is proportional to the logarithm of the distance from that point to the "wall", or the boundary of the fluid region. This law of the wall was first published in 1930 by Hungarian-American mathematician, aerospace engineer, and physicist Theodore von Kármán.[1] It is only technically applicable to parts of the flow that are close to the wall (<20% of the height of the flow), though it is a good approximation for the entire velocity profile of natural streams.[2]

A equação da lei da parede:

$$u = \frac{u_\tau}{\kappa} \ln \frac{y}{y_0}$$

u_τ ou u_* é a velocidade friccional, que pode ser obtida por

$$u_\star = \sqrt{\frac{\tau}{\rho}}$$

Onde τ é a tensão de cisalhamento do fundo e ρ é a densidade da água. A tensão de cisalhamento pode ser calculada por

$\tau = \rho\ c_D\ u^2$

Onde c_D é um coeficiente de arrasto (aproximado ~ 0.002), e u é a velocidade da corrente, normalmente a 1 m acima do fundo!

k é uma constante (von Karmann = 0.4), y é a distância da parede e y_0 é um coeficiente de rugosidade, normalmente o tamanho médio do grão do fundo.

A solução da equação da Lei da Parede fornece a velocidade em função da distância da parede (y na equação). Então, primeiramente devemos criar um domínio para a solução, que será a distância da parede!

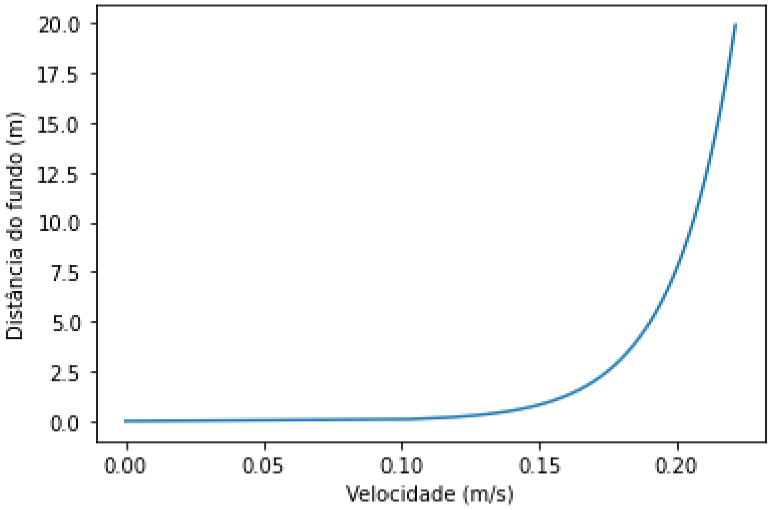
In [19]:

```
# z será um arranjo que começa em 0.001 e vai até 20 com intervalo de 0.1. No caso,
z = np.arange(0.001, 20, .1)
```

In [20]:

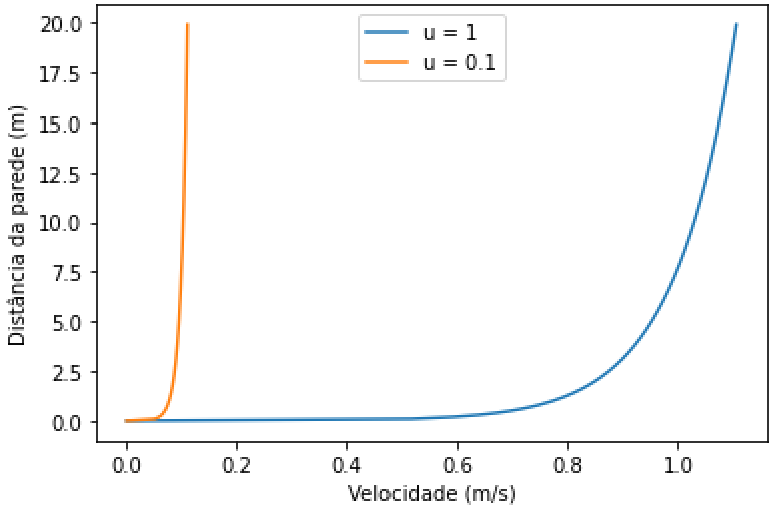
```
# definindo a densidade e cd

rho = 1000   # kg/m3
cd = 0.002 # adimensional

# um valor de velocidade razoável...
u = 0.2     # m/s

# calculando a tensão de cizalhamento
tau = rho * cd * u**2

# calculando a velocidade friccional
u_f = (tau / rho)**.5

# constante de von Karmann
k = 0.4
```

```
# diametro médido, no caso 1 mm (areia)
z0 = 0.001

# e usando a equação!
u_z = u_f/k * np.log(z / z0)

# vendo o que deu...
plt.plot(u_z, z)
plt.xlabel('Velocidade (m/s)')
plt.ylabel('Distância do fundo (m)')
plt.show()
```



In [21]:

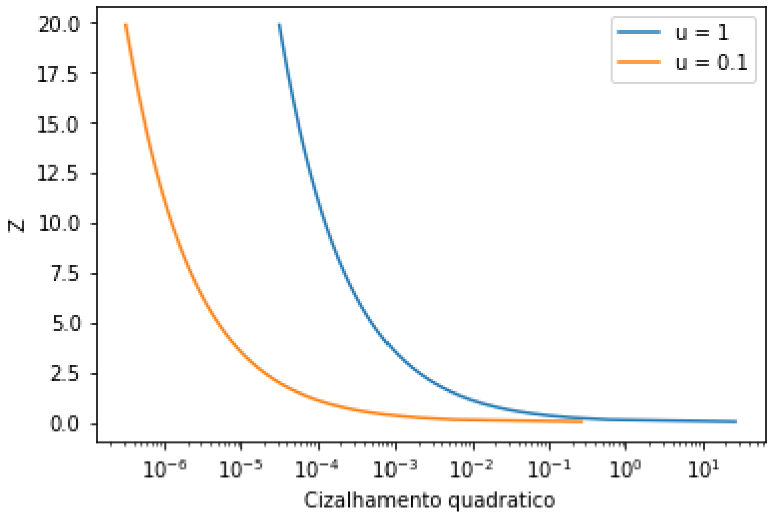
```
# o cálculo acima nós podemos transformar uma 'função', onde entrando com u e z, for
# e, a função pode ser modificada e posso adicionar outros valores de entrada, como

def calc_u_z(u, z):

    rho = 1000
    cd = 0.002
    tau = rho * cd * u**2
    u_f = (tau / rho)**.5
    k = 0.4
    z0 = 0.001
    u_z = u_f/k * np.log(z / z0)

    return u_z

# então, podemos usar a função para calcular vários resultados sem ter que repetir o
u_z1 = calc_u_z(1, z)
u_z01 = calc_u_z(0.1, z)


plt.plot(u_z1, z, label='u = 1')
plt.plot(u_z01, z, label='u = 0.1')

plt.xlabel('Velocidade (m/s)')
plt.ylabel('Distância da parede (m)')
plt.legend()
plt.show()
```



Tendo os perfis de velocidade, agora podemos obter o denominador do Ri. Para obter o cisalhamneto precisamos achar a razão da diferença de velocidade pela distância entre cadas.

In [22]:

```
# função para achar (du/dz)**2
def calc_s2(u, z):
    du = np.diff(u)   # diferencial de u = du
    dz = np.diff(z)   # diferencial de z = dz
    s2 = (du/dz)**2
    z2 = (z[:-1]+z[1:])/2  # ao fazer o diferencial o tamanho do arranho diminui 1,
    return s2, z2

s2_1, z2 = calc_s2(u_z1, z)
s2_01, z2 = calc_s2(u_z01, z)

plt.plot(s2_1, z2, label='u = 1')
plt.plot(s2_01, z2, label='u = 0.1')
plt.xlabel('Cizalhamento quadratico')
plt.ylabel('Z')
plt.xscale('log')  # comente este linha (colocando um # na frente) e veja o gráfico
plt.legend()
plt.show()
```



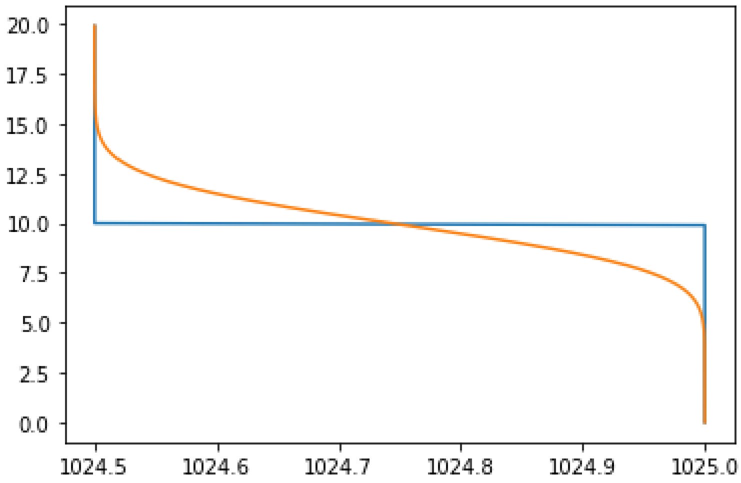
Perfil de Densidade!

Um perfil de densidade tem que ser criado arbitrariamente!

In [38]:

```
# densidade de referência
rho_0 = 1025

# replica o valor da densidade tantas vezes quanto o tamanho de u_z. Como se estivés
rho_z = np.linspace(rho_0, rho_0, len(u_z))

rho_z2 = rho_z.copy()
rho_z2[int(len(u_z)/2):] = rho_z[int(len(u_z)/2):] - .5  # e aqui fazemos com que a

# mas uma estratificação 'quadrada' não parece muito natural... podemos dar uma suav
rho_z3 = rho_z2.copy()
# dá uma suavizada no perfil!
for j in range(500): # para repetir a filtragem... mudando o número muda o grau de s
    for i in range(1, len(rho_z2)-1):
        rho_z3[i] = 1/4*rho_z3[i-1] + 1/2*rho_z3[i] + 1/4*rho_z3[i+1]  # isto é um f

plt.plot(rho_z2, z)
plt.plot(rho_z3, z)
```

Out[38]: [<matplotlib.lines.Line2D at 0x18f85c45700>]

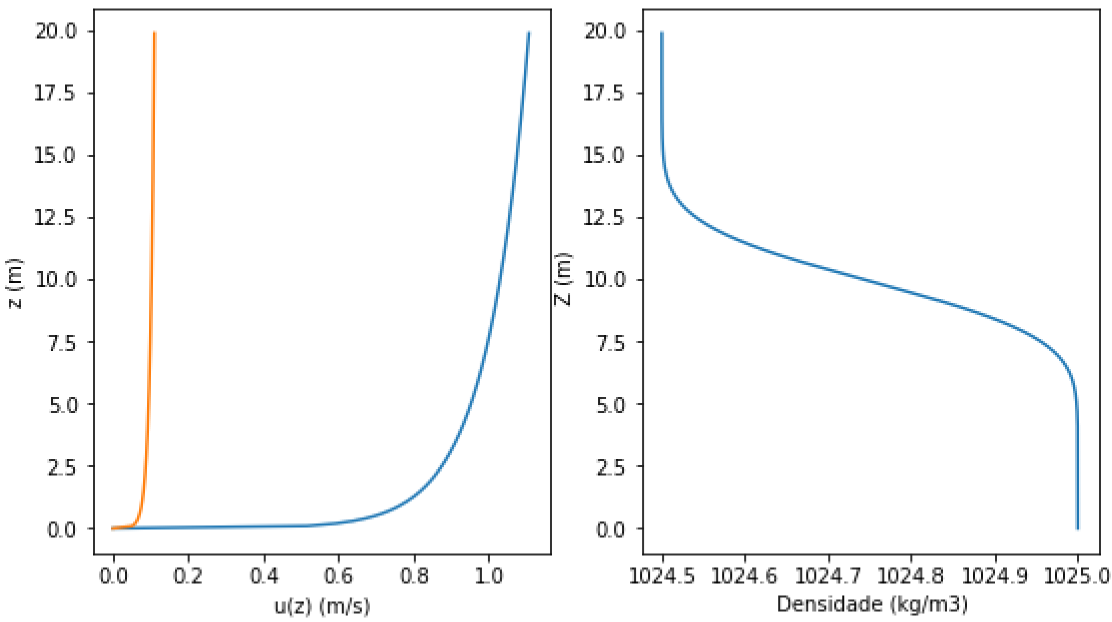


In [39]:

```
# para calcular a frequencia de empuxo / estratificação
def calc_N2(z, rho):
    dz = np.diff(z)
    drho = np.diff(rho) * -1 #o -1 é porque o sistema referencial de z está como o f
    g = 9.8
    rho_0 = 1025

    return g/rho_0 * drho/dz
```

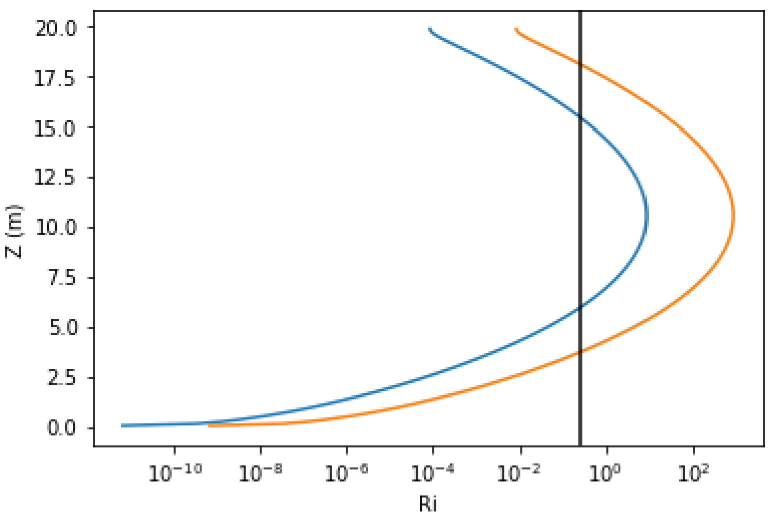
In [40]:

```
# vendo as condições que nós criamos!

fig, axs = plt.subplots(1,2, figsize=(9,5))
axs[0].plot(u_z1, z)
axs[0].plot(u_z01, z)
axs[0].set_xlabel('u(z) (m/s)')
axs[0].set_ylabel('z (m)')

axs[1].plot(rho_z3, z)
axs[1].set_xlabel(' Densidade (kg/m3)')
axs[1].set_ylabel(' Z (m)')
plt.show()
```



In [41]:

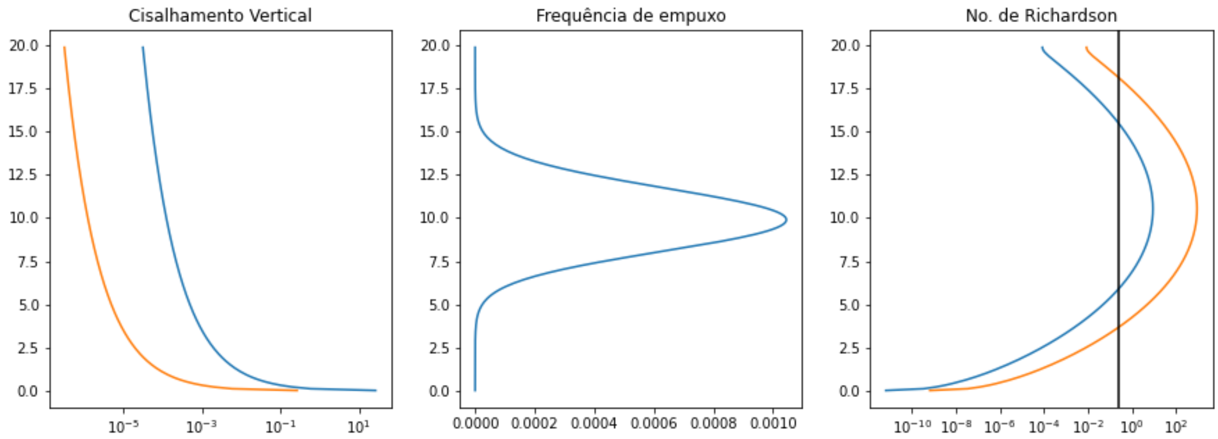
```
# colinha... para lembrar dos nomes das variáveis!
# s2_1, z2 = calc_s2(u_z1, z)
# s2_01, z2 = calc_s2(u_z01, z)

N2 = calc_N2(z, rho_z3)

Ri_a = N2/s2_1
Ri_b = N2/s2_01


Ri_a = N2/s2_1
Ri_b = N2/s2_01
plt.plot(Ri_a, z2)
plt.plot(Ri_b, z2)
plt.xscale('log')
plt.axvline(0.25, color='k') # para colocar o valor limite de 0.25
plt.xlabel('Ri')
plt.ylabel('Z (m)')
plt.show()
```



In [42]:

```
fig, ax = plt.subplots(1,3, figsize=(15,5))

ax[0].plot(s2_1, z2, label='u = 1')
ax[0].plot(s2_01, z2, label='u = 0.1')
```

```
ax[0].set_xscale('log')
ax[0].set_title('Cisalhamento Vertical')

ax[1].plot(N2, z2)
ax[1].set_title('Frequência de empuxo')

ax[2].plot(Ri_a, z2)
ax[2].plot(Ri_b, z2)
ax[2].set_title('No. de Richardson')
ax[2].set_xscale('log')
ax[2].axvline(0.25, color='k')

plt.show()
```



Interpretação:

Para para velocidade de 0,1 m/s, o Ri para a coluna de água onde ocorre a estratificação é bem maior do que o limite 0.25, o que indica que a estratificação é estável. Isto também parece ocorrer para a condição de velocidade de 1 m/s, poré os limites verticais de estabilidade são mais limitados verticalmente.