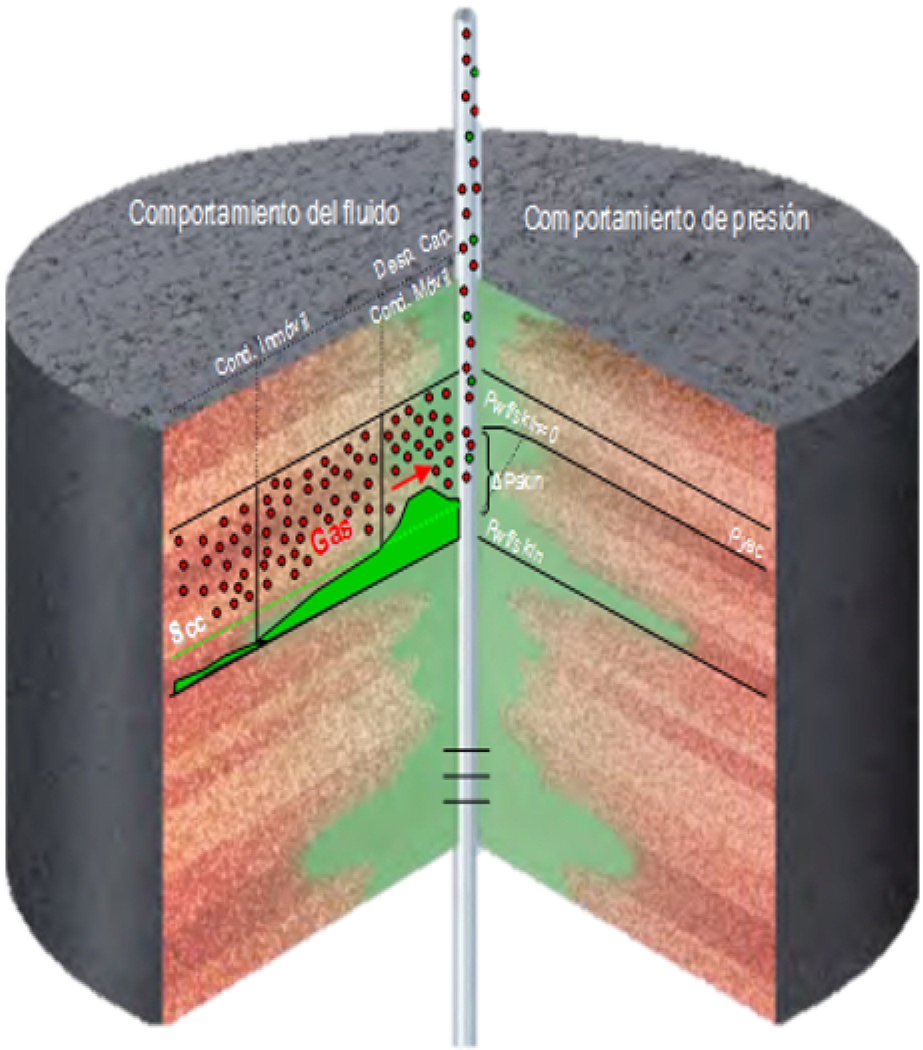
Reservoir Inflow Behaviour

November 1, 2021

1 Production Engineering

2 *Reservoir Inflow Behaviour*



3 Import Python Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
%config Completer.use_jedi = False
```

4 Important Functions

$$q_o = \frac{k_{oh(P_r - Pwf)}}{141.2\beta_o\mu_o\;(\ln\frac{r_e}{r_w} - 0.75 + S)}$$

4.1 Productivity Index Taking into account Petrophysical and Fluid Properties

$$J = \frac{K_o\,h}{141.2\,B_o\,u_o(\ln\frac{r_e}{r_w} - 0.75 + s)}$$

Where:
J: Productivity Index (bpd/psi)
K_o: Effective permeablity (md)
h: Thickness (ft)
B_o: Oil Formation Volume Factor (rb/stb)
u_o: Oil Viscosity (cp)
r_e: Drainage ratio (ft)
r_w: Well ratio (ft)
s: Skin

[3]:
```
# Productivity Index (darcy law)
def J_darcy(ko, h, bo, uo, re, rw, s, flow_regime = 'seudocontinuo'):
    if flow_regime == 'seudocontinuo':
        J = ko * h / (141.2 * bo * uo * (np.log(re / rw) - 0.75 + s))
    elif flow_regime == 'continuo':
        J = ko * h / (141.2 * bo * uo * (np.log(re / rw) + s))
    return J
```

4.2 Productivity Index with productivity test data

$$J = \frac{Q_o}{P_r - Pwf}$$

Where:
J: Productivity Index (bpd/psi)

Q_o: Oil Flow Rate (bpd)
P_r: Reservoir Pressure (psia)
Pwf: Pressure Well Flowing (psia)

[4]:
```
# Productivity Index
def J(q_test, pwf_test, pr, pb):
    if pwf_test > pb:
        J = q_test / (pr - pwf_test)
    else:
        J = q_test / ((pr - pb) + (pb / 1.8) * (1 - 0.2 * (pwf_test / pb) - 0.8
 ↪* (pwf_test / pb)**2))
    return J
```

4.3 Oil Flow Rate at Bubble Point

$$Q_b = J(P_r - P_b)$$

Where:
Q_b: Oil Flow Rate at Bubble Point (bpd)
J: Productivity Index (bpd/psi)
P_r: Reservoir Pressure (psia)
P_b: Bubble Point Pressue

[5]:
```
# Q(bpd) @ Pb
def Qb(q_test, pwf_test, pr, pb):
    qb = J(q_test, pwf_test, pr, pb) * (pr - pb)
    return qb
```

4.4 AOF at different conditions

If P_r **>** P_b -> The oil reservoir is **UNDERSATURATED:**

At this case, there are 2 conditions:

- If Pwf *>=* P_b*:*

$$AOF = JP_r$$

Otherwise,

$$AOF = Q_b + \frac{JP_b}{1.8}$$

On the other hand, if P_r <= P_b -> The oil reservoir is **SATURATED:**

At this situation:

$$AOF = \frac{Qo_{test}}{1 - 0.2(\frac{Pwf_{test}}{P_r}) - 0.8(\frac{Pwf_{test}}{P_r})^2}$$

Where:
AOF: Absolute Open Flow (bpd)

Q_b: Oil Flow Rate at Bubble Point (bpd) J: Productivity Index (bpd/psi)
P_r: Reservoir Pressure (psia)
P_b: Bubble Point Pressue
Pwf: Pressure Well Flowing (psia)
Pwf_{test}: Pressure Well Flowing of productity test (psia)
Qo_{test}: Oil Flow Rate of productivity test (bpd)

[6]:
```
# AOF(bpd)
def aof(q_test, pwf_test, pr, pb, pwf):
    if pr > pb: # Yac. subsaturado
        if pwf >= pb:
            aof = J(q_test, pwf_test, pr, pb) * pr
        elif pwf <  pb:
            aof = Qb(q_test, pwf_test, pr, pb) + ((J(q_test, pwf_test, pr, pb) *␣
↪pb) / (1.8))
    else: # Yac. Saturado
        aof = q_test / (1 - 0.2 * (pwf_test / pr) - 0.8 * (pwf_test / pr)**2)
    return aof
```

4.5 Q_o at Different Conditions

Here, it is assumed that FE (Flow Efficiency) = 100% by using Vogel's statements.

If P_r **>** P_b -> The oil reservoir is **UNDERSATURATED:**

At this case, there are 2 conditions:

- If $Pwf >= P_b$:

$$Q_o = J(P_r - Pwf)$$

Otherwise,

$$Q_o = Q_b + \frac{JP_b}{1.8}(1 - 0.2(\frac{Pwf}{P_b}) - 0.8(\frac{Pwf}{P_b})^2)$$

On the other hand, if $P_r <= P_b$ -> The oil reservoir is **SATURATED:**

At this situation:

$$Q_o = AOF(1 - 0.2(\frac{Pwf}{P_b}) - 0.8(\frac{Pwf}{P_b})^2)$$

Where:
Q_o: Oil Flow Rate of productivity test (bpd)
AOF: Absolute Open Flow (bpd)
Q_b: Oil Flow Rate at Bubble Point (bpd) J: Productivity Index (bpd/psi)
P_r: Reservoir Pressure (psia)
P_b: Bubble Point Pressue
Pwf: Pressure Well Flowing (psia)

[7]:
```
#Qo(bpd) @ vogel conditions
def qo_vogel(q_test, pwf_test, pr, pwf, pb):
    if pr > pb: # Yac. subsaturado
```

```
        if pwf >= pb:
            qo = J(q_test, pwf_test, pr, pb) * (pr - pwf)
        elif pwf < pb:
            qo =  Qb(q_test, pwf_test, pr, pb) + ((J(q_test, pwf_test, pr, pb) *␣
↪pb) / (1.8)) * \
            (1 - 0.2 * (pwf / pb) - 0.8 * ( pwf / pb)**2)
    elif pr <= pb: # Yac. Saturado
        qo = aof(q_test, pwf_test, pr, pb, pwf) * (1 - 0.2 * (pwf / pr) - 0.8 *␣
↪( pwf / pr)**2)
    return qo
```

4.6 *Ejercicio 1*

[8]:
```
# Data
ko = 8.2 #md
h = 53 #ft
bo = 1.2 #rb/stb
uo = 1.2 #cp
re = 2978.4 # ft
rw = 0.328 # ft
s = 0
pr = 5651 #psia
```

4.6.1 a) J

[9]:
```
J_darcy = J_darcy(ko, h, bo, uo, re, rw, s)
print(f"J -> {J_darcy} bpd/psia")
```

```
J -> 0.25555511281513077 bpd/psia
```

4.6.2 AOF

[10]:
```
AOF = J_darcy * pr
print(f"AOF -> {AOF} bpd")
```

```
AOF -> 1444.141942518304 bpd
```

4.7 *Ejercicio 2*

[11]:
```
# Data
pr = 2400 #psia
pb = 2500 #psia
pwf = 1000 #psia
q_test = 100 #stb/d
pwf_test = 1800 #psia
```

4.7.1 a) AOF

[12]:
```
AOF = aof(q_test, pwf_test, pr, pb, pwf)
print(f"AOF -> {AOF} bpd")
```

```
AOF -> 250.00000000000003 bpd
```

4.7.2 b) IPR Curve

[13]:
```
# Creating Dataframe
df = pd.DataFrame()

df['Pwf(psia)'] = np.array([2400, 2000, 1500, 1000, 500, 0])
df['Qo(bpd)'] = df['Pwf(psia)'].apply(lambda x: qo_vogel(q_test, pwf_test, pr,␣
 ↪x, pb))
```

[14]:
```
df
```

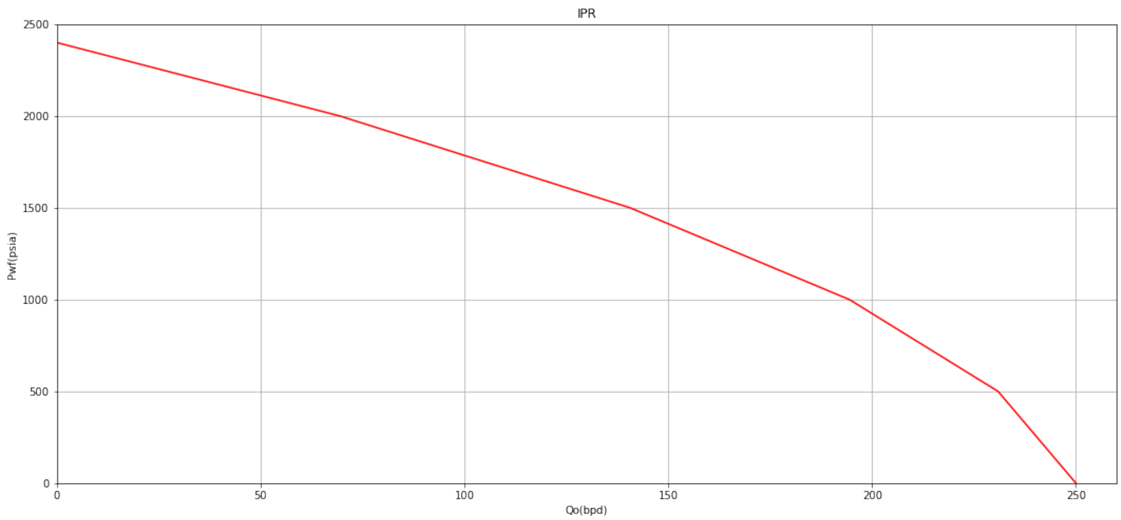
[14]:

	Pwf(psia)	Qo(bpd)
0	2400	0.000000
1	2000	69.444444
2	1500	140.625000
3	1000	194.444444
4	500	230.902778
5	0	250.000000

[15]:
```
# Plot

fig, ax = plt.subplots(figsize=(18, 8))

ax.plot(df['Qo(bpd)'], df['Pwf(psia)'], c='red')
ax.set_xlabel('Qo(bpd)')
ax.set_ylabel('Pwf(psia)')
ax.set_title('IPR')
ax.set(xlim=(0, df['Qo(bpd)'].max() + 10), ylim=(0, df['Pwf(psia)'][0] + 100))
ax.grid()
plt.show()
```



4.8 *Ejercicio 3*

[16]:
```
# Data
pr = 120 #bar
pb = 65 #bar
q_test = 400 #m3/d
pwf_test = 100 #bar
pwf = 40 #bar
```

4.8.1 Qo @ Pwf=40 bar

4.9 *Ejercicio 4*

[18]:
```
# Data
pr = 4000 #psi
pb = 3000 #psi
qo_test = 600 #bpd
pwf_test = 2000 #bpd
```

4.9.1 a) Qmax

4.9.2 b) Qo @ Pwf = 3500 psi

4.9.3 c) Qo @ Pwf = 1000 psi

4.9.4 d) pwf @ Q = 1110 bpd

4.9.5 e) IPR Curve