Pressure Drop Calculations & Optimum Line Size Selection

302 ft

Image shows approximate pipe length required, add extra 20% margin for piping turns & elevations

These are `python` libraries needed for unit conversion & mathematical functions

In [1]:
```python
import handcalcs.render
from handcalcs.decorator import handcalc
from math import log, log10, sqrt, pi, exp
```

In [2]:
```python
import forallpeople as si
si.environment('ukhan', top_level=True)
```

Table -1 Pipe Fittings Equivalent Lengths

Fitting	Types	(L/D)eq
90° Elbow Curved, Threaded	Standard Radius (R/D = 1)	30
	Long Radius (R/D = 1.5)	16
90° Elbow Curved, Flanged/Welded	Standard Radius (R/D = 1)	20
	Long Radius (R/D = 2)	17
	Long Radius (R/D = 4)	14
	Lon g Radius (R/D = 6)	12
90° Elbow Mitered	1 weld (90°)	60
	2 welds (45°)	15
	3 welds (30°)	8
45° Elbow Curved. Threaded	Standard Radius (R/D = 1)	16
	Long Radius (R/D = 1.5)	
45° Elbow Mitered	1 weld 45°	15
	2 welds 22.5°	6
180° Bend	threaded, close-return (R/D = 1)	50
	flanged (R/D = 1)	
	all types (R/D = 1.5)	
Tee Through-branch as an Elbow	threaded (r/D = 1)	60
	threaded (r/D = 1.5)	
	flanged (r/D = 1)	20
	stub-in branch	
Tee Run-through	threaded (r/D = 1)	20
	flanged (r/D = 1)	
	stub-i n branch	
Angle valve	45°, full line size, β = 1	55
	90° full line size, β = 1	150
Globe valve	standard, β = 1	340
	branch flow	90
Plug valve	straight through	18
	three-way (flow through)	30
Gate valve	standard, β = 1	8
Ball valve	standard, β = 1	3
Diaphragm	dam type	

Swing check valve	V_{min} = 35 [ρ (lbm/ft^3)]$^{-1/2}$	100
Lift check valve	V_{min} = 40 [ρ (lbm/ft^3)]$^{-1/2}$	600
Hose Coupling	Simple, Full Bore	5

Table-2 Absolute Roughness ξ

Material	Roughness (mm)
Drawn Tubing, Glass, Plastic	0.0015-0.01
Drawn Brass, Copper, Stainless Steel (New)	>0.0015-0.01
Flexible Rubber Tubing - Smooth	0.006-0.07
Flexible Rubber Tubing - Wire Reinforced	0.3-4
Stainless Steel	0.03
Wrought Iron (New)	0.045
Carbon Steel (New)	0.02-0.05
Carb on Steel (Slightly Corroded)	0.05-0.15
Carbon Steel (Moderately Corroded)	0.15-1
Carbon Steel (Badly Corroded)	1-3
Carbon Steel (Cement-lined)	1.5
Asphalted Cast Iron	0.1-1
Cast Iron (new)	0.25
Cast Iron (old, sandblasted)	1
Sheet Metal Ducts (with smooth joints)	0.02-0.1
Galvanized Iron	0.025-0.15
Wood Stave	0.18-0.91
Wood Stave, used	0.25-1
Smooth Cement	0.5
Concrete – Very Smooth	0.025-0.2
Concrete – Fine (Floated, Brushed)	0.2-0.8
Concrete – Rough, Form Marks	0.8-3
Riveted Steel	0.91-9.1
Water Mains with Tuberculations	1.2
Brickwork, Mature Foul Sewers	3

Source: https://neutrium.net (https://neutrium.net/fluid-flow/pressure-loss-from-fittings-in-pipe-summary/)

In [3]:

```
%%render params 1
PipeL = (302*1.2) *ft.to(m)  #20% additional length
PipeID = 50 *mm  #Internal Dia
PP_xi = 0.01 *mm  #PolyPropylene Roughness
```

$\mathrm{PipeL} = 110.5\ \mathrm{m}\ \ (20\%\ \text{additional length}) \qquad \mathrm{PipeID} = 50.0\ \mathrm{mm}\ \ (\text{Internal Dia}) \qquad PP_{\xi} = 10.0\ \mu\mathrm{m}\ \ (\text{PolyPropylene Roughness})$

In [4]:

```
%%render params 1
flow_H2O = 5 *m3_h
rho_H2O = 988 *kg_m3.prefix('unity') # at 50°C
nu_H2O = 0.5465 *cP #Viscosity in centiPoise is equal to mPa.s
```

$\mathrm{flow}_{H2O} = 5.0\ \mathrm{m^3 \cdot h^{-1}} \qquad \rho_{H2O} = 988.0\ \mathrm{kg \cdot m^{-3}}\ \ (\text{at } 50°\mathrm{C}) \qquad \nu_{H2O} = 546.5\ \mu\mathrm{Pa \cdot s}\ \ (\text{Viscosity in centiPoise is equal to mPa.s})$

Water physical properties: https://wiki.anton-paar.com/en/water/ (https://wiki.anton-paar.com/en/water/)

In [5]:

```
%%render params 1
Elbows = 10  #90° Elbow Threaded Standard
Elbow_EqFactor = 30
Valves = 2 #Ball valve
Valve_EqFactor = 3  #Refer Table-1
```

$\mathrm{Elbows} = 10\ \ (90°\ \text{Elbow Threaded Standard}) \qquad \mathrm{Elbow}_{EqFactor} = 30 \qquad \mathrm{Valves} = 2\ \ (\text{Ball valve})$

$\mathrm{Valve}_{EqFactor} = 3\ \ (\text{Refer Table-1})$

In [6]:

```
%%render long
Sigma_PipeL = PipeL + (Elbows*Elbow_EqFactor*PipeID) + (Valves*Valve_EqFactor*PipeID)
```

$$
\begin{aligned}
\Sigma_{PipeL} &= \mathrm{PipeL} + \left(\mathrm{Elbows} \cdot \mathrm{Elbow}_{EqFactor} \cdot \mathrm{PipeID}\right) + \left(\mathrm{Valves} \cdot \mathrm{Valve}_{EqFactor} \cdot \mathrm{PipeID}\right) \\
&= 110.460\ \mathrm{m} + (10 \cdot 30 \cdot 50.000\ \mathrm{mm}) + (2 \cdot 3 \cdot 50.000\ \mathrm{mm}) \\
&= 125.760\ \mathrm{m}
\end{aligned}
$$

In [7]:

```
@handcalc(jupyter_display=True)
def reynolds(D, F, rho, nu):
    A = 0.25 * pi * D**2
    velocity = F / A  #Calculate velocity
    NRe = (D * velocity * rho) / nu #Calculate Reynold's number
    return velocity, NRe
```

In [8]:

```python
velocity, NRe = reynolds(PipeID, flow_H2O, rho_H2O, nu_H2O)
```

$$A = 0.25 \cdot \pi \cdot (D)^2 = 0.25 \cdot 3.142 \cdot (50.000\ \mathrm{mm})^2 = 1963.495\ \mathrm{mm}^2$$

$$\mathrm{velocity} = \frac{F}{A} = \frac{5.000\ \mathrm{m^3} \cdot \mathrm{h}^{-1}}{1963.495\ \mathrm{mm}^2} = 707.355\ \mathrm{mm} \cdot \mathrm{s}^{-1} \quad \text{(Calculate velocity)}$$

$$\mathrm{NRe} = \frac{D \cdot \mathrm{velocity} \cdot \rho}{\nu} = \frac{50.000\ \mathrm{mm} \cdot 707.355\ \mathrm{mm} \cdot \mathrm{s}^{-1} \cdot 988.000\ \mathrm{kg} \cdot \mathrm{m}^{-3}}{546.500\ \mu\mathrm{Pa} \cdot \mathrm{s}} = 63940.260 \quad \text{(Calculate Reynold's number)}$$

In [9]:

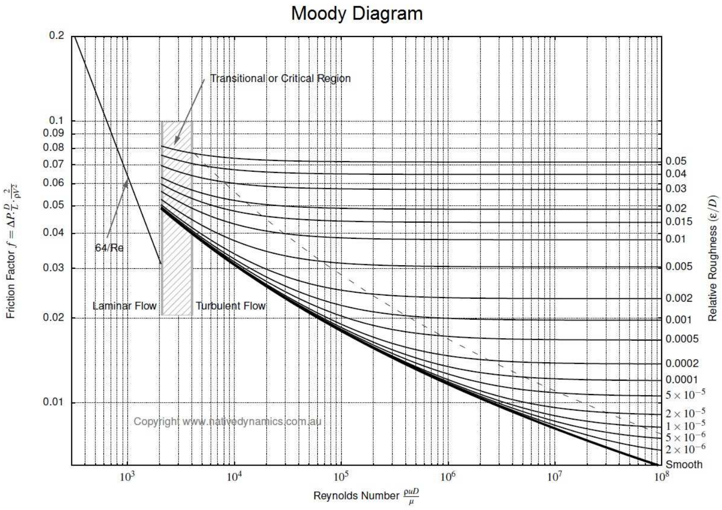
```python
%%render 4
if NRe <= 2100: Flow = 'Laminar'
elif NRe <= 4000: Flow = 'Transient'
elif NRe > 4000: Flow = 'Turbulent'

PipeRR = PP_xi/PipeID #Pipe Relative Roughness
```

Since, $\mathrm{NRe} > 4000 \rightarrow (63940.2597 > 4000)$:

$\mathrm{Flow} = \mathrm{Turbulent}$

$$\mathrm{PipeRR} = \frac{PP_{\xi}}{\mathrm{PipeID}} = \frac{10.0000\ \mu\mathrm{m}}{50.0000\ \mathrm{mm}} = 0.0002 \quad \text{(Pipe Relative Roughness)}$$



Moody Friction Factor Figure

Method-1 Using Graph

Relative Roughness is 0.0002
Reynolds number is approximately $6.4 \cdot 10^4$
Friction factor f is approximately 0.022

In [10]:

```python
@handcalc(jupyter_display=True)
def pressuredrop(f, Sigma_L, D, rho, velocity):
    Delta_p = f * (Sigma_L/D) * (rho*velocity**2)/2
    return Delta_p
```

In [11]:

```python
Delta_p = pressuredrop(0.022,Sigma_PipeL, PipeID, rho_H2O, velocity)
```

$$\Delta_p = f \cdot \left(\frac{\Sigma_L}{D} \right) \cdot \frac{\rho \cdot (\mathrm{velocity})^2}{2}$$
$$= 0.022 \cdot \left(\frac{125.760\ \mathrm{m}}{50.000\ \mathrm{mm}} \right) \cdot \frac{988.000\ \mathrm{kg} \cdot \mathrm{m}^{-3} \cdot \left(707.355\ \mathrm{mm} \cdot \mathrm{s}^{-1}\right)^2}{2}$$
$$= 13.677\ \mathrm{kPa}$$

Method-2 Using Churchill Emperical Equation

In [12]:

```python
@handcalc(jupyter_display=True, precision=4)
def churchill(NRe, D, xi):
    A = ( 2.457*log( 1 /( (7/NRe)**0.9 + 0.27*xi/D )) )**16
    B = (37530/NRe)**16
    f = 8 * ( (8/NRe)**12 + 1/(A+B)**1.5 )**(1/12)
    return f
```

In [13]:
```
f_churchill = churchill(NRe, PipeID, PP_xi)
```

$$A = \left(2.457 \cdot \ln\left(\frac{1}{\left(\frac{7}{\mathrm{NRe}}\right)^{0.9} + 0.27 \cdot \frac{\xi}{D}}\right)\right)^{16}$$
$$= \left(2.457 \cdot \ln\left(\frac{1}{\left(\frac{7}{63940.2597}\right)^{0.9} + 0.27 \cdot \frac{10.0000\ \mu\mathrm{m}}{50.0000\ \mathrm{mm}}}\right)\right)^{16}$$
$$= 524038602297619775488.0000$$

$$B = \left(\frac{37530}{\mathrm{NRe}}\right)^{16} = \left(\frac{37530}{63940.2597}\right)^{16} = 0.0002$$

$$f = 8 \cdot \left(\left(\frac{8}{\mathrm{NRe}}\right)^{12} + \frac{1}{(A+B)^{1.5}}\right)^{\left(\frac{1}{12}\right)}$$
$$= 8 \cdot \left(\left(\frac{8}{63940.2597}\right)^{12} + \frac{1}{(524038602297619775488.0000 + 0.0002)^{1.5}}\right)^{\left(\frac{1}{12}\right)}$$
$$= 0.0206$$

In [14]:
```
Delta_pChurchill = pressuredrop(f_churchill, Sigma_PipeL, PipeID, rho_H2O, velocity)
```

$$\Delta_p = f \cdot \left(\frac{\Sigma_L}{D}\right) \cdot \frac{\rho \cdot (\mathrm{velocity})^2}{2}$$
$$= 0.021 \cdot \left(\frac{125.760\ \mathrm{m}}{50.000\ \mathrm{mm}}\right) \cdot \frac{988.000\ \mathrm{kg \cdot m^{-3}} \cdot \left(707.355\ \mathrm{mm \cdot s^{-1}}\right)^2}{2}$$
$$= 12.786\ \mathrm{kPa}$$

Method-3 Using Serghides Emperical Equation

In [15]:
```
@handcalc(jupyter_display=True, precision=4)
def serghide(NRe, D, xi):
    A = -2*log10( (xi/D)/3.7 + 12/NRe )
    B = -2*log10( (xi/D)/3.7 + 2.51*A/NRe )
    C = -2*log10( (xi/D)/3.7 + 2.51*B/NRe )
    f = ( A - ( (B-A)**2 )/(C - 2*B + A) )**(-2)
    return f
```

In [16]:
```
f_serghide = serghide(NRe, PipeID, PP_xi)
```

$$A = (-2) \cdot \log_{10}\left(\frac{\frac{\xi}{D}}{3.7} + \frac{12}{\mathrm{NRe}}\right) = (-2) \cdot \log_{10}\left(\frac{\frac{10.0000\ \mu\mathrm{m}}{50.0000\ \mathrm{mm}}}{3.7} + \frac{12}{63940.2597}\right) = 7.2333$$

$$B = (-2) \cdot \log_{10}\left(\frac{\frac{\xi}{D}}{3.7} + 2.51 \cdot \frac{A}{\mathrm{NRe}}\right)$$
$$= (-2) \cdot \log_{10}\left(\frac{\frac{10.0000\ \mu\mathrm{m}}{50.0000\ \mathrm{mm}}}{3.7} + 2.51 \cdot \frac{7.2333}{63940.2597}\right)$$
$$= 6.9422$$

$$C = (-2) \cdot \log_{10}\left(\frac{\frac{\xi}{D}}{3.7} + 2.51 \cdot \frac{B}{\mathrm{NRe}}\right)$$
$$= (-2) \cdot \log_{10}\left(\frac{\frac{10.0000\ \mu\mathrm{m}}{50.0000\ \mathrm{mm}}}{3.7} + 2.51 \cdot \frac{6.9422}{63940.2597}\right)$$
$$= 6.9720$$

$$f = \left(A - \frac{(B-A)^2}{C - 2 \cdot B + A}\right)^{(-2)} = \left(7.2333 - \frac{(6.9422 - 7.2333)^2}{6.9720 - 2 \cdot 6.9422 + 7.2333}\right)^{(-2)} = 0.0206$$

In [17]:
```
Delta_pSerghide = pressuredrop(f_serghide, Sigma_PipeL, PipeID, rho_H2O, velocity)
```

$$\Delta_p = f \cdot \left(\frac{\Sigma_L}{D}\right) \cdot \frac{\rho \cdot (\mathrm{velocity})^2}{2}$$
$$= 0.021 \cdot \left(\frac{125.760\ \mathrm{m}}{50.000\ \mathrm{mm}}\right) \cdot \frac{988.000\ \mathrm{kg \cdot m^{-3}} \cdot \left(707.355\ \mathrm{mm \cdot s^{-1}}\right)^2}{2}$$
$$= 12.800\ \mathrm{kPa}$$

Method-4 Goudar- Sonnad

In [18]:
```
@handcalc(jupyter_display=True, precision=4)
def gsonnad(NRe, D, xi):
    a = 2/log(10)
    b = (xi/D)/3.7
    d = log(10)/5.02 *NRe
    s = b*d + log(d)
    q = s**( s/(s+1) )
    g = b*d + log(d/q)
    zeta = q/g
    delta_LA = (g/(g+1))*zeta
    delta_CFA = delta_LA * ( 1 + (zeta/2)/( (g+1)**2 + (zeta/3)*(2*g-1) ) )
    f =  1/( a* ( log(d/q)+ delta_CFA ) )**2
    return f
```

In [19]:
```
f_gsonnad = gsonnad(NRe, PipeID, PP_xi)
```

$$a = \frac{2}{\ln(10)} = 0.8686$$

$$b = \frac{\frac{\xi}{D}}{3.7} = \frac{\frac{10.0000\ \mu\mathrm{m}}{50.0000\ \mathrm{mm}}}{3.7} = 0.0001$$

$$d = \frac{\ln(10)}{5.02} \cdot \mathrm{NRe} = \frac{\ln(10)}{5.02} \cdot 63940.2597 = 29328.2647$$

$$s = b \cdot d + \ln(d) = 0.0001 \cdot 29328.2647 + \ln(29328.2647) = 11.8716$$

$$q = (s)^{\left(\frac{s}{s+1}\right)} = (11.8716)^{\left(\frac{11.8716}{11.8716+1}\right)} = 9.7956$$

$$g = b \cdot d + \ln\left(\frac{d}{q}\right) = 0.0001 \cdot 29328.2647 + \ln\left(\frac{29328.2647}{9.7956}\right) = 9.5897$$

$$\zeta = \frac{q}{g} = \frac{9.7956}{9.5897} = 1.0215$$

$$\delta_{LA} = \left(\frac{g}{g+1}\right) \cdot \zeta = \left(\frac{9.5897}{9.5897+1}\right) \cdot 1.0215 = 0.9250$$

$$\begin{aligned}\delta_{CFA} &= \delta_{LA} \cdot \left(1 + \frac{\frac{\zeta}{2}}{(g+1)^2 + \left(\frac{\zeta}{3}\right) \cdot (2 \cdot g - 1)}\right)\\ &= 0.9250 \cdot \left(1 + \frac{\frac{1.0215}{2}}{(9.5897+1)^2 + \left(\frac{1.0215}{3}\right) \cdot (2 \cdot 9.5897 - 1)}\right)\\ &= 0.9290\end{aligned}$$

$$\begin{aligned}f &= \frac{1}{\left(a \cdot \left(\ln\left(\frac{d}{q}\right) + \delta_{CFA}\right)\right)^2}\\ &= \frac{1}{\left(0.8686 \cdot \left(\ln\left(\frac{29328.2647}{9.7956}\right) + 0.9290\right)\right)^2}\\ &= 0.0166\end{aligned}$$

In [20]:
```
Delta_pGsonnad = pressuredrop(f_gsonnad, Sigma_PipeL, PipeID, rho_H2O, velocity)
```

$$\begin{aligned}\Delta_p &= f \cdot \left(\frac{\Sigma_L}{D}\right) \cdot \frac{\rho \cdot (\mathrm{velocity})^2}{2}\\ &= 0.017 \cdot \left(\frac{125.760\ \mathrm{m}}{50.000\ \mathrm{mm}}\right) \cdot \frac{988.000\ \mathrm{kg} \cdot \mathrm{m}^{-3} \cdot \left(707.355\ \mathrm{mm} \cdot \mathrm{s}^{-1}\right)^2}{2}\\ &= 10.326\ \mathrm{kPa}\end{aligned}$$

Method-5 Using Tkachenko, Mileikovskyi

[Source Moscow University (https://link.springer.com/chapter/10.1007/978-3-030-57340-9_37)](https://link.springer.com/chapter/10.1007/978-3-030-57340-9_37)

In [21]:
```
@handcalc(jupyter_display=True, precision=4)
def tkmile(NRe, D, xi):
    A_0 = -0.79638*log( (xi/D)/8.208 + 7.3357/NRe )
    A_1 = NRe*(xi/D) + 9.3120665*A_0
    f = ( (8.128943 + A_1)/( 8.128943*A_0 - 0.86859209*A_1*log(A_1/(3.7099535*NRe)) ) )**2
    return f
```

In [22]:
```
f_tkmile = tkmile(NRe, PipeID, PP_xi)
```

$$
\begin{aligned}
A_0 &= (-0.79638) \cdot \ln\left(\frac{\frac{\xi}{D}}{8.208} + \frac{7.3357}{\mathrm{NRe}}\right) \\
&= (-0.79638) \cdot \ln\left(\frac{\frac{10.0000\ \mu\mathrm{m}}{50.0000\ \mathrm{mm}}}{8.208} + \frac{7.3357}{63940.2597}\right) \\
&= 7.0721
\end{aligned}
$$

$$
A_1 = \mathrm{NRe} \cdot \left(\frac{\xi}{D}\right) + 9.3120665 \cdot A_0 = 63940.2597 \cdot \left(\frac{10.0000\ \mu\mathrm{m}}{50.0000\ \mathrm{mm}}\right) + 9.3120665 \cdot 7.0721 \qquad = 78.6443
$$

$$
\begin{aligned}
f &= \left(\frac{8.128943 + A_1}{8.128943 \cdot A_0 - 0.86859209 \cdot A_1 \cdot \ln\left(\frac{A_1}{3.7099535 \cdot \mathrm{NRe}}\right)}\right)^2 \\
&= \left(\frac{8.128943 + 78.6443}{8.128943 \cdot 7.0721 - 0.86859209 \cdot 78.6443 \cdot \ln\left(\frac{78.6443}{3.7099535 \cdot 63940.2597}\right)}\right)^2 \\
&= 0.0206
\end{aligned}
$$

In [23]:
```
Delta_pTkmile = pressuredrop(f_tkmile, Sigma_PipeL, PipeID, rho_H2O, velocity)
```

$$
\begin{aligned}
\Delta_p &= f \cdot \left(\frac{\Sigma_L}{D}\right) \cdot \frac{\rho \cdot (\mathrm{velocity})^2}{2} \\
&= 0.021 \cdot \left(\frac{125.760\ \mathrm{m}}{50.000\ \mathrm{mm}}\right) \cdot \frac{988.000\ \mathrm{kg} \cdot \mathrm{m}^{-3} \cdot \left(707.355\ \mathrm{mm} \cdot \mathrm{s}^{-1}\right)^2}{2} \\
&= 12.799\ \mathrm{kPa}
\end{aligned}
$$

In [24]:
```
from IPython.display import HTML, display

def display_table(data):
    html = "<table>"
    for row in data:
        html += "<tr>"
        for field in row:
            html += "<td><h4>%s</h4></td>"%(field)
        html += "</tr>"
    html += "</table>"
    display(HTML(html))

data = [['Emperical Relationship','Friction Factor','Pressure Drop'],
        ['Churchill', round(f_churchill,4), Delta_pChurchill],
        ['Serghide', round(f_serghide,4), Delta_pSerghide],
        ['Goudar-Sonnad', round(f_gsonnad,4), Delta_pGsonnad],
        ['Tkachenko, Mileikovskyi', round(f_tkmile,4), Delta_pTkmile]]
display_table(data)
```

Emperical Relationship	**Friction Factor**	**Pressure Drop**
Churchill	**0.0206**	**12.786 kPa**
Serghide	**0.0206**	**12.800 kPa**
Goudar-Sonnad	**0.0166**	**10.326 kPa**
Tkachenko, Mileikovskyi	**0.0206**	**12.799 kPa**

In [25]:
```
%reload_ext version_information
%version_information handcalcs, forallpeople
```

Out[25]:

Software	**Version**
Python	3.9.18 32bit [Clang 14.0.7 (https://android.googlesource.com/toolchain/llvm-project 4c603efb]
IPython	8.7.0
OS	Linux 4.19.191 25884040 abA137FXXS3CWL2 armv8l with libc
handcalcs	1.6.5
forallpeople	2.6.7
	Mon Jan 22 22:10:49 2024 +03