2.2 Linear Blending Problem

In [ ]:

```
%%capture
!pip install -q pyomo
!apt-get install -y -qq coinor-cbc
```

Problem Statement (Jenchura, 2017)

A brewery receives an order for 100 gallons of 4% ABV (alchohol by volume) beer. The brewery has on hand beer A that is 4.5% ABV that cost USD 0.32 per gallon to make, and beer B that is 3.7% ABV and cost USD 0.25 per gallon. Water could also be used as a blending agent at a cost of USD 0.05 per gallon. Find the minimum cost blend that meets the customer requirements.

Representing Problem Data as a Python Dictionary

We will use this problem as an opportunity to write a Python function that accepts data on raw materials and customer specifications to produce the lowest cost blend.

The first step is to represent the problem data in a generic manner that could, if needed, be extended to include additional blending components. Here we use a dictionary of materials, each key denoting a blending agent. For each key there is a sub-dictionary containing attributes of each blending component.

In [ ]:

```
data = {
    'A': {'abv': 0.045, 'cost': 0.32},
    'B': {'abv': 0.037, 'cost': 0.25},
    'W': {'abv': 0.000, 'cost': 0.05},
}
```

Model Formulation

Objective Function

If we let subscript c denote a blending component from the set of blending components C, and denote the volume of c used in the blend as x_c, the cost of the blend is

$$\text{cost} = \sum_{c\in C} x_c P_c$$

where P_c is the price per unit volume of c. Using the Python data dictionary defined above, the price P_c is given by `data[c]['cost']`.

Volume Constraint

The customer requirement is produce a total volume V. Assuming ideal solutions, the constraint is given by

$$V = \sum_{c\in C} x_c$$

where x_c denotes the volume of component c used in the blend.

Product Composition Constraint

The product composition is specified as 4% alchohol by volume. Denoting this as $\bar{A}$, the constraint may be written as

$$\bar{A} = \frac{\sum_{c\in C} x_c A_c}{\sum_{c\in C} x_c}$$

where A_c is the alcohol by volume for component c. As written, this is a nonlinear constraint. Multiplying both sides of the equation by the denominator yields a linear constraint

$$\bar{A}\sum_{c\in C} x_c = \sum_{c\in C} x_c A_c$$

A final form for this constraint can be given in either of two versions. In the first version we subtract the left-hand side from the right to give

$$0 = \sum_{c\in C} x_c \left(A_c - \bar{A}\right) \qquad \text{Version 1 of the linear blending constraint}$$

Alternatively, the summation on the left-hand side corresponds to total volume. Since that is known as part of the problem specification, the blending constraint could also be written as

$$\bar{A}V = \sum_{c\in C} x_c A_c \qquad \text{Version 2 of the linear blending constraint}$$

Which should you use? Either will generally work well. The advantage of version 1 is that it is fully specified by a product requirement $\bar{A}$, which is sometimes helpful in writing elegant Python code.

Implementation in Pyomo

A Pyomo implementation of this blending model is shown in the next cell. The model is contained within a Python function so that it can be more easily reused for additional calculations, or eventually for use by the process operator.

Note that the pyomo library has been imported with the prefix `pyomo`. This is good programming practive to avoid namespace collisions with problem data.

In [4]:

```
import pyomo.environ as pyomo

vol = 100
abv = 0.040

def beer_blend(vol, abv, data):
    C = data.keys()
    model = pyomo.ConcreteModel()
    model.x = pyomo.Var(C, domain=pyomo.NonNegativeReals)
    model.cost = pyomo.Objective(expr = sum(model.x[c]*data[c]['cost'] for c in C))
    model.vol = pyomo.Constraint(expr = vol == sum(model.x[c] for c in C))
    model.abv = pyomo.Constraint(expr = 0 == sum(model.x[c]*(data[c]['abv'] - abv) for c
in C))

    solver = pyomo.SolverFactory('cbc')
    solver.solve(model)

    print('Optimal Blend')
    for c in data.keys():
        print('  ', c, ':', model.x[c](), 'gallons')
    print()
    print('Volume = ', model.vol(), 'gallons')
    print('Cost = $', model.cost())

beer_blend(vol, abv, data)
```

```
Optimal Blend
   A : 37.5 gallons
   B : 62.5 gallons
   W : 0.0 gallons

Volume =  100.0 gallons
Cost = $ 27.625
```

In [ ]: