CH ENGR 100 Discussion Week 1 by Yi Ming (Michael) Ren

Credits to Jeffrey Kantor (jeff at nd.edu) for I used some of his course notes

Installing Python and Numpy:

Python: https://www.ics.uci.edu/~pattis/common/handouts/pythoneclipsejava/python.html (https://www.ics.uci.edu/~pattis/common/handouts/pythoneclipsejava/python.html)

Numpy: https://numpy.org/install/ (https://numpy.org/install/)

Juypter Notebook: https://jupyter.readthedocs.io/en/latest/install.htm (https://jupyter.readthedocs.io/en/latest/install.htm)

The Fundamental Units of Measurement

There are seven fundamental units in the Système International d'Unités (SI) system, all other unit can be generated from these.

Dimension	**SI Units**	**CGS**	**English**
Mass	kilogram (kg)	gram (g)	lb-mass, slug
Length	meter (m)	centimeter (cm)	inch (in); foot (ft)
Time	second (s)	second (s)	second (sec)
Temperature	degree Kelvin (K)	degree Celsius (°C)	degree Rankine (°R)
Quantity	mole (gmol)	mole (gmol)	mole (gmol)

Coherent Derived Units

The SI system describes 22 'derived units' that are expressed in terms of the fundamental units.

Calculations with coherent units is straightforward since they involve just ratios or products of base units. The calculations break down into three steps:

1. Recast all of the problem quantities in base or coherent derived units.
2. Perform the numerical calculations.
3. Using algebraic rules for the cancelation of terms, simplify and reduce units to units relevant to the context of the problem.

Example: The kinetic energy of a object with mass $m = 3\ \mathrm{kg}$ traveling at velocity $v = 10\ \mathrm{m/s}$ is given by

$$\begin{aligned} K.E. &= \frac{1}{2}mv^2 \\ &= \frac{1}{2}(3\ \mathrm{kg})\left(10\frac{\mathrm{m}}{\mathrm{s}}\right)^2 \\ &= 150\ \frac{\mathrm{kg}\cdot\mathrm{m}^2}{\mathrm{s}^2} = 150\ \mathrm{N}\cdot\mathrm{m} = \boxed{150\ \mathrm{J}} \end{aligned}$$

In [24]:
```
m  = 3
v = 10
KE = 1/2*m*v**2
KE
```

Out[24]:
```
150.0
```

Example: What is the mass of air in a spherical balloon with radius of 10 cm inflated to 1 atm at 25 degrees Celsius? Assume the molecular weight of air is 28.966 grams/mole, and that atmospheric pressure is 101,325 Pa.

Solution: Start by converting all of the quantities in the problem to base or derived units.

$$MW_{air} = 28.966\ \frac{\mathrm{g}}{\mathrm{mol}} = 0.028966\ \frac{\mathrm{kg}}{\mathrm{mol}}$$
$$P = 101,325\ \mathrm{Pa} = 101,325\ \frac{\mathrm{N}}{\mathrm{m}^2} = 101,325\ \frac{\mathrm{kg}}{\mathrm{m}\cdot\mathrm{s}^2}$$
$$V = \frac{4}{3}\pi r^3 = \frac{4}{3}\pi(0.10\ \mathrm{m})^3 = 0.004189\ \mathrm{m}^3$$
$$R = 8.314\frac{\mathrm{J}}{\mathrm{K}\cdot\mathrm{mol}} = 8.314\frac{\mathrm{kg}\cdot\mathrm{m}^2}{\mathrm{s}^2\cdot\mathrm{K}\cdot\mathrm{mol}}$$
$$T = 25 + 273.15 = 298.15K$$

Using molecular weight and the ideal gas law

$$PV = nRT$$
$$m_{air} = MW_{air}n_{air} = MW_{air}\frac{PV}{RT}$$

Substituting quantities into this expression for m_{air}

$$m_{air} = \left(0.028966\ \frac{\mathrm{kg}}{\mathrm{mol}}\right)\frac{\left(101,325\ \frac{\mathrm{kg}}{\mathrm{m}\cdot\mathrm{s}^2}\right)\left(0.004189\ \mathrm{m}^3\right)}{\left(8.314\frac{\mathrm{kg}\cdot\mathrm{m}^2}{\mathrm{s}^2\cdot\mathrm{K}\cdot\mathrm{mol}}\right)(298.15\ \mathrm{K})}$$
$$= \frac{0.028966\cdot 101,325\cdot 0.004189}{8.314\cdot 298.15}\times\frac{\frac{\mathrm{kg}}{\mathrm{mol}}\cdot\frac{\mathrm{kg}}{\mathrm{m}\cdot\mathrm{s}^2}\cdot\mathrm{m}^3}{\frac{\mathrm{kg}\cdot\mathrm{m}^2}{\mathrm{s}^2\cdot\mathrm{K}\cdot\mathrm{mol}}\cdot\mathrm{K}}$$
$$= \boxed{0.00496\ \mathrm{kg}}$$

- make sure you do unit conversion, I have seen so many people lose points because they do not do unit conversion

In [25]:
```
MW_air = 28.966 # g/mol
P = 101325 # Pa
r = 10 # cm
R = 8.314 # J/(K mol)
T = 25 # C
m_air = (MW_air/1000)*(P)*(4/3*(r/100)**3*3.14)/((T+273)*R)
m_air
```

Out[25]:
```
0.004959606713346778
```

What is Python and Numpy

Python is a general-purpose programming language that is really useful for data analysis.

Currently, it is also used as the dominant programming language for machine learning and deep learning purposes.

Numpy is just a python library that is used for array and matrix calculations.

In [26]:
```
import numpy as np
```

Solving systems of equations in Python

This will make your lives a lot easier and will help you later on.

Why solve such 6 equations when you can write 1 line of code??

Example:

$$8a + 7b + 10c = 20$$

$$-2a + 2b + 30c = 1$$

$$5a - 3b + 0.1c = 30$$

This translates to the problem of $Ax = b$

where A is the coefficient before the variables, x is a vector containing the variables [a,b,c], and b is the answer vector.

$$A = \begin{pmatrix} 8 & 7 & 10 \\ -2 & 2 & 30 \\ 5 & -3 & 0.1 \end{pmatrix}, x = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, b = \begin{pmatrix} 20 \\ 1 \\ 30 \end{pmatrix}$$

In [27]:
```
A = np.array([[8,7,10],[-2,2,30],[5,-3,0.1]])
A
```

Out[27]:
```
array([[ 8. ,  7. , 10. ],
       [-2. ,  2. , 30. ],
       [ 5. , -3. ,  0.1]])
```

```
In [28]: b = np.array([20,1,30])
         b
```

```
Out[28]: array([20,  1, 30])
```

```
In [29]: x = np.linalg.solve(A, b)
         x
```

```
Out[29]: array([ 4.31234853, -2.79584535,  0.50721293])
```

If you hand solve the problem, then you should get the same answer as above where

$$a = 4.31234853$$
$$b = -2.79584535$$
$$c = 0.50721293$$

Why not just do it by hand???

While you may be able to solve the above systems of equation by hand, what if the A matrix has the dimensions of 100x100 (100 variables and 100 equations). Then solving by hand is not really a good strategy to compute the answer efficiently.

Another reason if when we need to calculate a range of different coefficients. For example, what if we need to vary the first item in the coefficient matrix, A? Let us say that a_{11} needs to be calculate from a range of 1-10.

$$A = \begin{pmatrix} a_{11} & 7 & 10 \\ -2 & 2 & 30 \\ 5 & -3 & 0.1 \end{pmatrix}, x = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, b = \begin{pmatrix} 20 \\ 1 \\ 30 \end{pmatrix}$$

where a_{11} ranges from 1-10.

Then we would needs to use a loop in order to calculate the equation and store all the answers accordingly.

```
In [30]: ans = {}
         for ai in range(1,11):
             A = np.array([[ai,7,10],[-2,2,30],[5,-3,0.1]])
             b = np.array([20,1,30])
             x = np.linalg.solve(A, b)
             ans[ai] = x
         ans
```

```
Out[30]: {1: array([6.78404139, 1.31998911, 0.39760349]),
          2: array([6.27059909, 0.46501091, 0.42037255]),
          3: array([ 5.82940718, -0.26965679,  0.4399376 ]),
          4: array([ 5.44621775, -0.9077394 ,  0.45693048]),
          5: array([ 5.11029814, -1.46710886,  0.47182713]),
          6: array([ 4.81340976, -1.96148396,  0.48499292]),
          7: array([ 4.54912345, -2.40157049,  0.49671293]),
          8: array([ 4.31234853, -2.79584535,  0.50721293]),
          9: array([ 4.09900176, -3.15110794,  0.51667398]),
          10: array([ 3.90576983, -3.47287551,  0.52524302])}
```

As you can see, the ans variable is a hashmap (dictionary) in which we can store the equations regarding to when a_{11}. We can type what value of a_{11} we want and retrive it instantly.

For example: If I want the solution of for when $a_{11} == 4$, then

```
In [31]: ans[4]

Out[31]: array([ 5.44621775, -0.9077394 ,  0.45693048])
```

Chemical Engineering Problem

Problem Statement:

Liquid benzene and liquid n-hexane are blended to form a stream flowing at a rate of 700 lbm/h. An on-line densitometer (an instrument used to determine density) indicates that the stream has a density of 0.810 g/mL. Using specific gravities from Table B.1, estimate the mass and volumetric feed rates of the two hydrocarbons to the mixing vessel (in U.S. customary units). State at least two assumptions required to obtain the estimate from the recommended data.

Assumptions:

1. Volumes of benzene and hexane are additive
2. No accumulation of mass (steady-state)
3. Density values are valid for this problem at STP

Balance Equations

$$V_B + V_H = V_f$$

$$m_B + m_H = 1700 lb_m/hr$$

$$V_f = 1700 \frac{lb_m}{hr} * \frac{1}{0.81} * \frac{ft^3}{64.23 lb_m} = 32.675 ft^3/hr$$

$$m_B = V_B * 0.879 * \frac{64.23 lb_m}{ft^3}$$

$$m_H = V_H * 0.659 * \frac{64.23 lb_m}{ft^3}$$

```
In [32]: v_f = 700/(0.81*64.23)
         a = 0.879*62.43
         b = 0.659*62.43
         v_f

Out[32]: 13.454733471340456
```

This translates to the problem to the form of $Ax = b$

$$A = \begin{pmatrix} 1 & 1 \\ 54.88 & 41.14 \end{pmatrix}, x = \begin{pmatrix} V_B \\ V_H \end{pmatrix}, b = \begin{pmatrix} 32.67 \\ 700 \end{pmatrix}$$

In [33]:
```
A = np.array([[1,1],[a,b]])
B = np.array([v_f,700])
x = np.linalg.solve(A, B)
x
```

Out[33]:
```
array([10.66313049,  2.79160298])
```

In [34]:
```
V_B = x[0]
V_H = x[1]
m_B = a*V_B
m_H = b*V_H
print(m_B,m_H,m_B+m_H)
```

```
585.1496287804089 114.85037121959115 700.0
```

Material Balance Equation

$$\text{Accumulation} = \text{Inflow} - \text{Outflow} + \text{Generation} - \text{Consumption}$$

$$\underbrace{\text{Accumulation}_X}_{\frac{d(C_X V)}{dt}} = \underbrace{\text{Inflow}_X}_{=0} - \underbrace{\text{Outflow}_X}_{=0} + \underbrace{\text{Generation}_X}_{=0} - \underbrace{\text{Consumption}_X}_{=0}$$