Prueba Corta 1

April 28, 2021

1 Repositorio

El archivo *.ipynb* de este Jupyter Notebook se encuentra en https://github.com/plasmallan/anlisis_procesos_ii en la carpeta Prueba Corta 1 dentro de la carpeta Pruebas Cortas.

[1]:
```
import pandas as pd
import numpy as np
from copy import deepcopy
import matplotlib.pyplot as plt
```

2 Problema 1

Se tiene la ecuación química balanceada:

$$C_6H_8 + C_8H_{12} = C_{14}H_{20}$$

Para estimar posibles concentraciones en el equilibrio, se puede tomar que las concentraciones iniciales de $A = C_6H_8$ y $B = C_8H_{12}$ son iguales a x y que el equilibrio puede ser alcanzado una vez se toman partes iguales de cada uno de los compuestos, por ejemplo tomando la mitad de A y B para formar C se tiene que:

$$\frac{\frac{x}{2} + \frac{x}{2}}{\left(x - \frac{x}{2}\right)^2} = 153.818 \Rightarrow x = 0.026 \text{ [frac.mol]}$$

De esta forma, se entiende que las concentraciones iniciales de A y B corresponden a 0.026, mientras que una vez alcanzado el equilibrio, las concentraciones de estos son de 0.013 y la de C de 0.026

[16]:
```
# Se importan los datos y se visualizan las primeras 5 líneas
data=pd.read_csv('Concentrations.csv',low_memory=False)
data.head()
```

[16]:

	Concentration A	Concentration B	Concentration C
0	0.182	0.165	0.653
1	0.039	0.097	0.864
2	0.089	0.132	0.779
3	0.078	0.198	0.724
4	0.118	0.111	0.771

[17]:
```
Q=pd.DataFrame(data['Concentration C']/data['Concentration A']/
 ↪data['Concentration B'])
```

[18]:
```
# Se almacena Q como arreglo de numpy para simplificar la obtención de los␣
 ↪valores que se desean a continuación
Qarray=np.array(Q)
```

[27]:
```
#Se buscan los que están en equilibrio y se almacenan en Qeq, se retorna el la␣
 ↪cantidad que cumplen la condición
Qeq=Qarray[np.where(np.abs(Qarray-153.818)<1)]
len(Qeq)
```

[27]: 36

3 Problema 2

[28]:
```
# Se crean las dos matrices con Numpy
A=np.matrix([[2.8,9.3],
             [4.2,3.1],
             [1.4,9.1]])
B=np.matrix([[8.1,5.3,7.7],
             [6.7,2.0,5.8]])
```

[33]:
```
# Se calcula la traza del producto de ambas matrices
np.trace(np.matmul(A,B))
```

[33]: 177.01000000000002

4 Problema 3

[35]:
```
A=np.matrix([[9.9,9.1,4.8],
           [7.1,7.6,2.5],
           [4.7,9.9,9.9],
           [2.9,7.4,1.5],
           [8.4,6.8,7.6]])
```

[46]:
```
def AijPi(A,ij):
    """
    Función que multiplica la entrada A[i,j] de la matriz A por el númer pi.
    Parámetros de entrada:
    ----------------------
    A: objeto matriz
    ij: tupla que contiene la el número de la fila i y columna j de la matriz␣
 ↪que se quiere tomar.

    Salida:
```

```
    ----------------------
    valor: entrada A[i,j] de la matriz por pi.
    """
    i=ij[0]-1
    j=ij[1]-1
    valor=A[i,j]*np.pi
    return valor
```

[47]:
```
AijPi(A,(4,2))
```

[47]:
```
23.24778563656447
```