# 1 Python/latex practice (2 pts)

## 1.1 Review the materials we discussed in class on Wednesday. Look over scipy.optimize, scipy.minimize, and np.array modules. Post at least one question on piazza about the concepts/examples/tutorials that you find confusing.

Thank you for clarifying my query regarding duplication of an array.

## 1.2 The Fibonacci sequence is defined such that each number is the sum of the two previous numbers in the sequence, starting with [1, 1, 2, 3, 5. . . ]. Generate the first 15 elements in the sequence in Python, with the final result as a list.

In [1]:
```
def Fibonacci(n):
    x = []
    if n <= 2:
        for i in range(n):
            x.append(1)
        return x
    else:
        i = 1
        j = 1
        list = [i,j]
        for x in range(n-2):
            k = i + j
            list = list + [k]
            i = j
            j = k
    return list
```

In [2]:
```
Fibonacci(15)
```

Out [2]:
```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]
```

### 1.2.1 Bonus (1pt) write a fibonacci generator as a recursive function

In [3]:
```
import numpy as np
def Fibonacci2(n):
    x = []
    if n <= 2:
        for i in range(n):
            x.append(1)
```

```
        return x
    else:
        x = Fibonacci2(n-1)
        z = Fibonacci2(n-1)[n-2] + Fibonacci2(n-2)[n-3]
        x.append(z)
        return x
```

In [4]:

```
Fibonacci2(15)
```

Out [4]:

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]
```

### 1.3 Type-set 5 latex equations. Choose the 5 most complicated equations you have seen so far in your other classes (or from your undergrad courses) and typeset them here.

### 1.4 1. Navier-Stokes Equation

#### 1.4.1 In x-direction for incompressible flow:

$$\rho(\frac{\partial u_x}{\partial t} + u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y} + u_z\frac{\partial u_x}{\partial z}) = -\frac{\partial p}{\partial x} + \mu(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial^2 u_z}{\partial z^2}) + \frac{1}{3}\mu\frac{\partial}{\partial x}(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}) + \rho g_x$$

### 1.5 2. Fourier Transform

#### 1.5.1 Forward Fourier Transform and Inverse Fourier Transform respectively are:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

$$f(t) = \frac{1}{2\pi}F(\omega)e^{-i\omega t}d\omega$$

### 1.6 3. Taylor Series

#### 1.6.1 f(x) near the point 'h' is:

$$f(h) + \frac{f'(h)}{1!}(x-h) + \frac{f''(h)}{2!}(x-h)^2 + \frac{f'''(h)}{3!}(x-h)^3 + ... = \sum_{n=1}^{\infty}\frac{f^{(n)}(h)}{n!}$$

### 1.7 4. Overall Heat Transfer Coefficient

#### 1.7.1 Across an evaporator:

$$\frac{1}{U} = \frac{1}{h_o \times (r_i/r_o)} + R_f\frac{r_o}{r_i} + \frac{r_o ln(r_o/r_i)}{k_{evap}} + \frac{1}{h_i}$$

### 1.8 5. Beattie-Bridgeman Equation of State

$$Z = 1 + (B_0 - \frac{A_0}{RT} - \frac{c}{RT^3})\rho + (B_0 b - \frac{A_0 a}{RT^3})\rho^2 + (\frac{B_0 bc}{T^3})\rho^3$$

## 2 Problem 1: Reaction extent (5 pt)

### 2.1 Problem 1A

Ammonia has been proposed as a chemical to store hydrogen for a hydrogen fuel cell due its favorable volumetric hydrogen density. Write down the stoichiometric vector for the thermal decomposition of ammonia to nitrogen and hydrogen.

$2\,NH_3 \longrightarrow N_2 + 3\,H_2$

**The stoichiometric vector for the above reaction is:**

**[$NH_3, N_2, H_2$] = [-2, 1, 3]**

In [5]:
```
import numpy as np
# storing the stoichiometric vector as 'Alpha'
Alpha = np.array([-2, 1, 3])
Alpha
```

Out [5]:
```
array([-2,  1,  3])
```

### 2.2 Problem 1B

Suppose that we know the initial gas concentration of ammonia in a tank is 100 mol/L, and the fractional reaction extent proceeds as $\Xi = (1 - e^{-3t^2})$ where $t$ is the time in seconds. Plot the concentration of ammonia, nitrogen, and hydrogen as a function of time over the first 2 seconds. Label the plot with a legend to indicate the species.

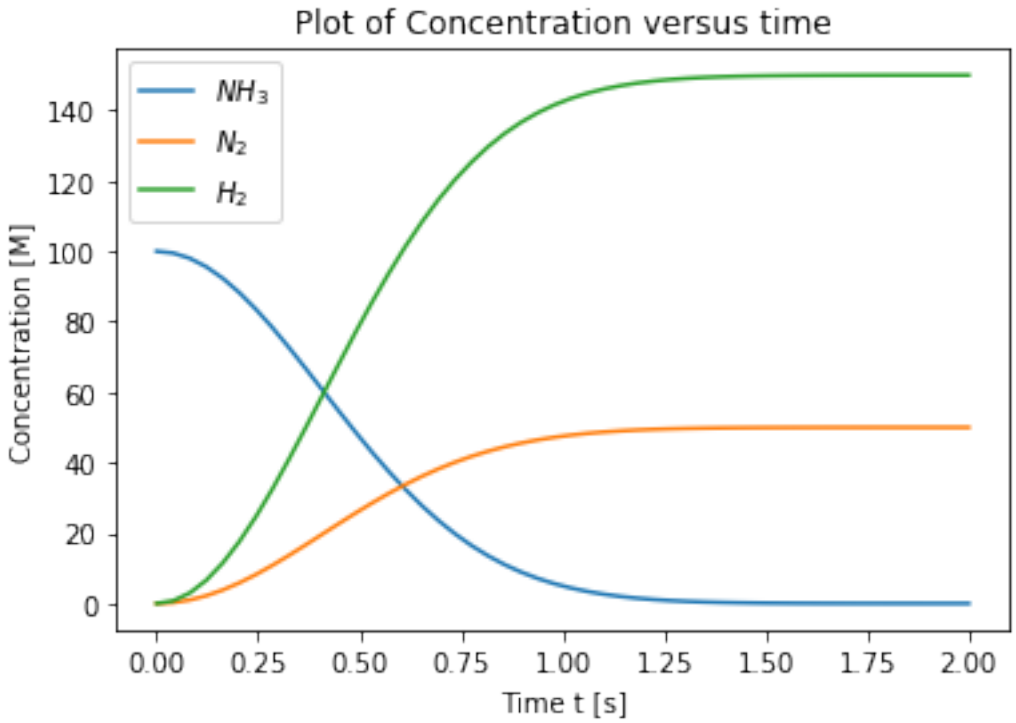
In [6]:
```
import matplotlib.pyplot as plt
C_j0 = np.array([[100, 0, 0],] * 50)  # in mol/L
t = np.linspace(0,2)   # in s for required period of 2.0s
Xi = 1 - np.exp(-3 * t**2)
Xi_t = np.transpose([Xi])  # to form a matrix multipliable by Alpha
# Since Xi*t is intensive, we divide concentration by reactant's coefficient
# Thus, by the equation stated, we use coefficient of ammonia; i.e.,2
rxn = 50 * (Alpha * Xi_t) #  Here we multiply by: 100/2 = 50 mol/L
C_j = C_j0 + rxn  # Concentration (in mol/L)  variation as a function of time
NH3 = C_j[:,0]
N2 = C_j[:,1]
H2 = C_j[:,2]

print(C_j)
```

```
[[1.00000000e+02 0.00000000e+00 0.00000000e+00]
 [9.95014551e+01 2.49272436e-01 7.47817309e-01]
```

```
[9.80206838e+01 9.89658087e-01 2.96897426e+00]
[9.56015399e+01 2.19923003e+00 6.59769008e+00]
[9.23147105e+01 3.84264477e+00 1.15279343e+01]
[8.82542848e+01 5.87285760e+00 1.76185728e+01]
[8.35332833e+01 8.23335835e+00 2.47000751e+01]
[7.82784410e+01 1.08607795e+01 3.25823385e+01]
[7.26245823e+01 1.36877088e+01 4.10631265e+01]
[6.67089326e+01 1.66455337e+01 4.99366011e+01]
[6.06656981e+01 1.96671510e+01 5.90014529e+01]
[5.46212043e+01 2.26893979e+01 6.80681936e+01]
[4.86898243e+01 2.56550879e+01 7.69652636e+01]
[4.29708562e+01 2.85145719e+01 8.55437157e+01]
[3.75464318e+01 3.12267841e+01 9.36803523e+01]
[3.24804620e+01 3.37597690e+01 1.01279307e+02]
[2.78185565e+01 3.60907218e+01 1.08272165e+02]
[2.35887999e+01 3.82056001e+01 1.14616800e+02]
[1.98032270e+01 4.00983865e+01 1.20295159e+02]
[1.64598150e+01 4.17700925e+01 1.25310277e+02]
[1.35448063e+01 4.32275969e+01 1.29682791e+02]
[1.10351822e+01 4.44824089e+01 1.33447227e+02]
[8.90112879e+00 4.55494356e+01 1.36648307e+02]
[7.10836181e+00 4.64458191e+01 1.39337457e+02]
[5.62021369e+00 4.71898932e+01 1.41569679e+02]
[4.39941566e+00 4.78002922e+01 1.43400877e+02]
[3.40954187e+00 4.82952291e+01 1.44885687e+02]
[2.61610953e+00 4.86919452e+01 1.46075836e+02]
[1.98735148e+00 4.90063243e+01 1.47018973e+02]
[1.49469405e+00 4.92526530e+01 1.47757959e+02]
[1.11298368e+00 4.94435082e+01 1.48330524e+02]
[8.20510506e-01 4.95897447e+01 1.48769234e+02]
[5.98878003e-01 4.97005610e+01 1.49101683e+02]
[4.32764314e-01 4.97836178e+01 1.49350854e+02]
[3.09615992e-01 4.98451920e+01 1.49535576e+02]
[2.19307858e-01 4.98903461e+01 1.49671038e+02]
[1.53795583e-01 4.99231022e+01 1.49769307e+02]
[1.06780611e-01 4.99466097e+01 1.49839829e+02]
[7.34006320e-02 4.99632997e+01 1.49889899e+02]
[4.99535161e-02 4.99750232e+01 1.49925070e+02]
[3.36582235e-02 4.99831709e+01 1.49949513e+02]
[2.24530416e-02 4.99887735e+01 1.49966320e+02]
[1.48292107e-02 4.99925854e+01 1.49977756e+02]
[9.69660616e-03 4.99951517e+01 1.49985455e+02]
[6.27740776e-03 4.99968613e+01 1.49990584e+02]
[4.02346092e-03 4.99979883e+01 1.49993965e+02]
[2.55316039e-03 4.99987234e+01 1.49996170e+02]
[1.60404030e-03 4.99991980e+01 1.49997594e+02]
[9.97726055e-04 4.99995011e+01 1.49998503e+02]
[6.14421235e-04 4.99996928e+01 1.49999078e+02]]
```

In [7]:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(t, NH3, t, N2, t, H2)
plt.xlabel('Time t [s]')
plt.ylabel('Concentration [M]')
plt.title('Plot of Concentration versus time')
plt.legend(["$NH_3$","$N_2$", "$H_2$"])
plt.show()
```




## 2.3 Problem 1C

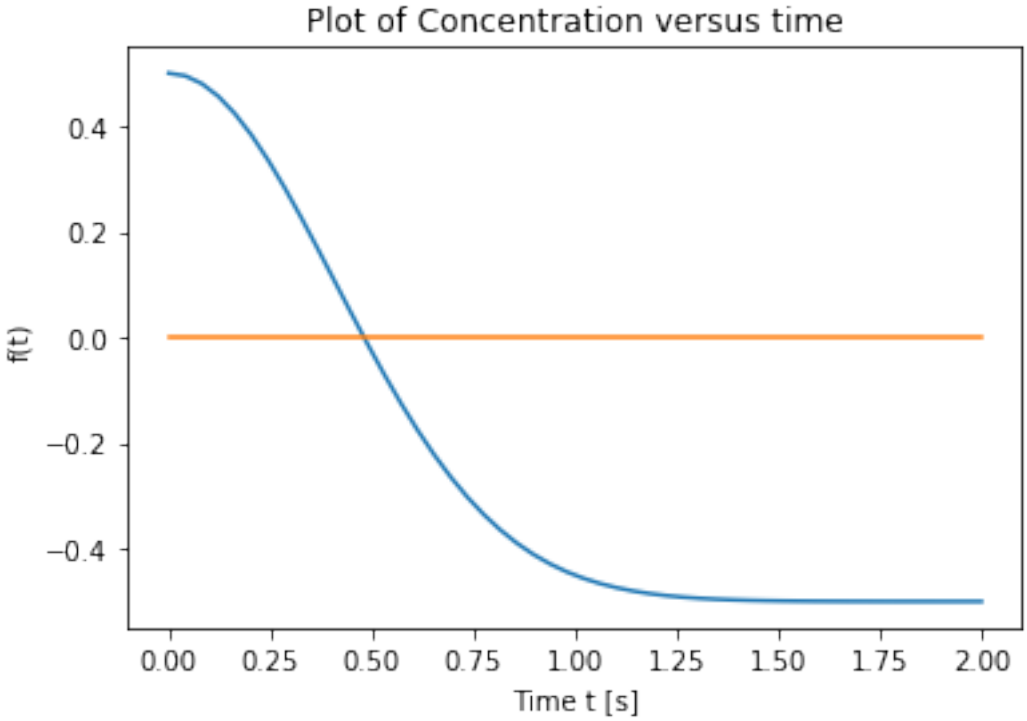
The operator is worried that if the reaction continues to full extent the increase in the number of moles will drive the pressure too high. Calculate the time it will take to achieve a 50% higher pressure in the reactor (assuming ideal gas) due to the increase in the number of moles, using fsolve.

In [8]:

```
import numpy as np
import matplotlib.pyplot as plt  # Plotting to estimate initial guess
from scipy.optimize import fsolve
dNg = (Alpha[0] + Alpha[1] + Alpha[2]) / (-Alpha[0])

def f(t):
    extent = 1 -np.exp(-3 * t*t)  # extent is the Xi function
    p = 1.5 - 1 - (extent) * dNg
    return p

plt.plot(t, f(t))
```

```
plt.plot(t, np.zeros(len(t)))
plt.xlabel('Time t [s]')
plt.ylabel('f(t)')
plt.title('Plot of Concentration versus time')
ans, d, flag, msg = fsolve(f, 0.1, full_output=1)
print(f'The reaction takes {ans[0]:1.3f}s to be at 50% higher pressure.')
```

```
The reaction takes 0.481s to be at 50% higher pressure.
```




## 3 Problem 2: Stoichiometry of reaction systems (3 pt)

Consider the following set of chemical reactions:

$$H_2 + Br_2 \rightleftharpoons 2\,HBrBr_2 \rightleftharpoons 2\,BrBr + H_2 \rightleftharpoons HBr + HH + Br_2 \rightleftharpoons HBr + BrH + HBr \rightleftharpoons H_2 + Br2\,Br \rightleftharpoons Br_2$$

### 3.1 Determine the stoichiometric matrix $\nu$, the species list $A$ for this reaction system (in text)

Write A here: $[H_2, Br_2, HBr, Br, H]$

In [9]:

```
import numpy as np
nu = np.array([[-1,-1,2,0,0],[0,-1,0,2,0],[-1,0,1,-1,1],[0,-1,1,1,-1],
               [1,0,-1,1,-1],[0,1,0,-2,0]])
nu
```

Out [9]:

```
array([[-1, -1,  2,  0,  0],
       [ 0, -1,  0,  2,  0],
       [-1,  0,  1, -1,  1],
       [ 0, -1,  1,  1, -1],
       [ 1,  0, -1,  1, -1],
       [ 0,  1,  0, -2,  0]])
```

### 3.2 Determine the vector of molecular weights $MW$ and verify that all of the equations you have written obey mass balance

In [10]:

```
MW = np.array([2, 160, 81, 80, 1])
ans_matrix = nu * MW
ans = np.dot(nu, MW)
# Checking both the masses of each element and mass balance as following:
print(ans_matrix, ans)
```

```
[[  -2 -160  162    0    0]
 [   0 -160    0  160    0]
 [  -2    0   81  -80    1]
 [   0 -160   81   80   -1]
 [   2    0  -81   80   -1]
 [   0  160    0 -160    0]] [0 0 0 0 0 0]
```

The null matrix of "ans" signifies mass balance is followed.

### 3.3 Determine the rank of the matrix using numpy or scipy. How many reactions are linearly independent?

In [11]:

```
from numpy.linalg import matrix_rank
ni = matrix_rank(nu)
print('The number of independent reactions "ni" are:', ni)
```

```
The number of independent reactions "ni" are: 3
```

### 3.4 Now that you have found the number of independent reactions $n_i$, which $n_i$ of the original set of six reactions can be chosen as an independent set? Try guessing some set of $n_i$ reactions and determine the rank of the new stoichiometric matrix. Stop when you have determined successfully one or more sets of $n_i$ independent reactions.

Reaction **1** is dependent on Reaction 4 and Reaction 3

Reaction **6** is dependent on Reaction 2

Reaction **5** is dependent on Reaction 3

Thus a matrix of 3 independent reactions can be made using Reactions 2, 3 and 4 as follows:

In [12]:

```
nu_2=nu[1:4]
nu_2
```

Out [12]:

```
array([[ 0, -1,  0,  2,  0],
       [-1,  0,  1, -1,  1],
       [ 0, -1,  1,  1, -1]])
```

## 4 How long did it take you to complete this assignment?

Approximately 6-7 hours (including debugging time)