In [75]:

```
from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
import math
from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
import math
%matplotlib inline
from ipywidgets import interactive
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from scipy import stats
from IPython.display import Image, display
import ipywidgets as widgets
import ipywidgets as widgets
from IPython.display import Image, display
from ipywidgets import Button, Layout
from chemlib import electrolysis
from chemlib import Galvanic_Cell


```

show plt fuction for Interactive Graphical Method by file or by trials

In [76]:

```
def show_mplt(c,c1,c2,c3,T) :
        r=[]
        tt=[]
        A=[]
        K=[]

        plt.figure(figsize=(15, 25))

        plt.subplot(4, 1, 1)


        plt.xlabel('second')
        plt.ylabel( 'Mol')
        plt.title('Zero order reaction')
        plt.plot(T,c)
        ###############
        plt.subplot(4, 1, 2)

        plt.xlabel("second")
        plt.ylabel("Ln(a-x)")

        plt.plot(T,c1)
        plt.title('First order reaction')

        ###############
        plt.subplot(4, 1, 3)

        plt.plot(T,c2 )
        plt.title('Second order reaction')
        plt.xlabel("second")
        plt.ylabel("1/(a-x)")
        ###############
        plt.subplot(4, 1, 4)

        plt.plot(T, c3 )
        plt.title('Third order reaction')
        plt.xlabel("second")
        plt.ylabel("1/((a-x)**2)")
        kk=plt.show()
        ####################################
        slope, intercept, r_value0, p_value, std_err = stats.linregress(c,T)
        r0=abs(r_value0)
        r.append(r0)
        a0= abs(intercept)
        A.append(a0)
        consentrate0= slope
        K.append(consentrate0)
        t0=(a0)/(2*consentrate0)
        tt.append(t0)
        #####
```

```
        slope, intercept, r_value1, p_value, std_err = stats.linregress(c1,T)
        r1= abs(r_value1)
        r.append(r1)

        a1= math.exp(intercept)
        A.append(a1)

        consentrate1= -slope

        K.append(consentrate1)

        t1=(0.693)/consentrate1
        tt.append(t1.tolist())
        ##########
        slope, intercept, r_value2, p_value, std_err = stats.linregress(c2,T)
        r2= abs(r_value2)
        r.append(r2)

        consentrate2= float(slope)

        K.append(consentrate2)
        a2= (1/intercept)
        A.append(a2)
        t2=float((1)/(float(consentrate2))*(float(a2)))
        tt.append(t2)
        ###########
        slope, intercept, r_value3, p_value, std_err = stats.linregress(c3,T)
        r3= abs(r_value3)
        consentrate3= float(slope/2)

        K.append(consentrate3)
        a3=1/intercept
        a3=math.sqrt((a3))
        A.append(a3)



        max_value = max(r)


        max_index = r.index(max_value)

        n=max_index

        t=tt[n]
        a=A[n]
        k=K[n]
        z= ("The reactant order is", n,'n/ The reaction half-life is ',t ,"  time
        z = "".join(myTuple)

        print(z)
        return
#https://stackoverflow.com/questions/51664659/how-to-calculate-distance-between-tw
```

```
##############################
# this fuction we will use it in  ## #Half-Life Method # #Vant Hoff Differential M

def order_finder(R1,A1,R2,A2) :
        R=math.log(float(R1/R2))
        A=math.log(float(A1/A2))
        n= R/A
        n= round(n)


        return n
#https://stackoverflow.com/questions/51664659/how-to-calculate-distance-between-tw

#############
# this fuction we will use to plot and find order, half time, initail concentratio
def show_plt(c,t,title,xaxil,yaxil,unit_k ) :
        n= plt.figure()
        plt.xlabel(xaxil)
        plt.ylabel(yaxil)

        plt.plot(t, c )
        plt.title(title)
        plt.savefig('kkk.jpeg', dpi=300, bbox_inches='tight')
        plt.show()



        return n
#https://stackoverflow.com/questions/51664659/how-to-calculate-distance-between-tw


```

#Initial Rate Method

In [11]:

```
#Initial Rate Method
#################################################################################
print('" The method of initial rates allows the values of these reaction orders to
#The previos information is cited from :https://chem.libretexts.org/Ancillary_Mater
listOfImageNames = ["rat1.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))

print('\nR1 and R2 are the rate constant')
print('\nA1 and A2 are initial concentration of reactant')

########################Initial Rate Method#########################################
def t( R1,A1,R2,A2):
    try:
        R1 =(float(R1))
        A1=(float(A1))
        R2=(float(R2))
        A2=(float(A2))
        n= order_finder(R1,A1,R2,A2)
        n=('The reactant order is ', round(n))
        return n
    except:
        print('Fill all requirement boxes , if this massage still appears meaning t

interact(t,  R1='Type only number', A1="Type only number",R2='Type only number', A1
################################################################################
btn = widgets.Button(description='Click here to find antoher reactant order by Init
display(btn)
def my_event_handler(btn_object):
    w=widgets.Text(value='Initial Rate Method', disabled=True)
    display(w)

    interact(t, R1='1', A1="1",R2='5', A2='5');

btn.on_click(my_event_handler)
#Reference:
#https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textboo
#https://chem.libretexts.org/Ancillary_Materials/Laboratory_Experiments/Wet_Lab_Exp
```

```
" The method of initial rates allows the values of these reaction orders
to be found by running the reaction multiple times under controlled cond
itions and measuring the rate of the reaction in each case. All variable
s are held constant from one run to the next, except for the concentrati
on of one reactant. The order of that reactant concentration in the rate
law can be determined by observing how the reaction rate varies as the c
oncentration of that one reactant is varied. This method is repeated for
each reactant until all the orders are determined.Citition Source: chem.
libretexts.org"
```

$$\Rightarrow \frac{R_{0_1}}{R_{0_2}} = (\frac{[A_0]_1}{[A_0]_2})^n$$

R1 and R2 are the rate constant

A1 and A2 are initial concentration of reactant

R1 1

A1 8

R2 2

A2 16

('The reactant order is ', 1)

Click here to find antoher reactant order by Initial Rate Method

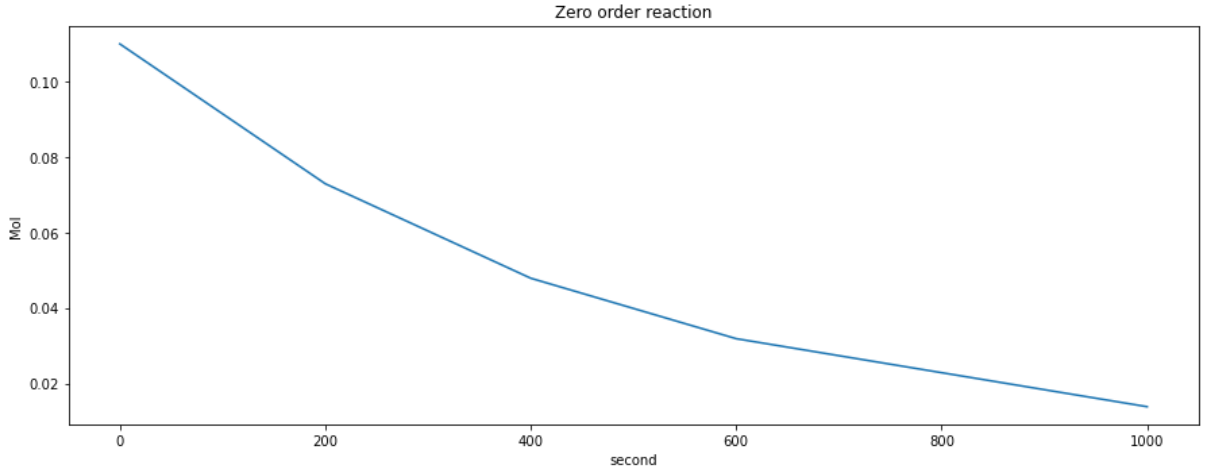
#Interactive Graphical Method

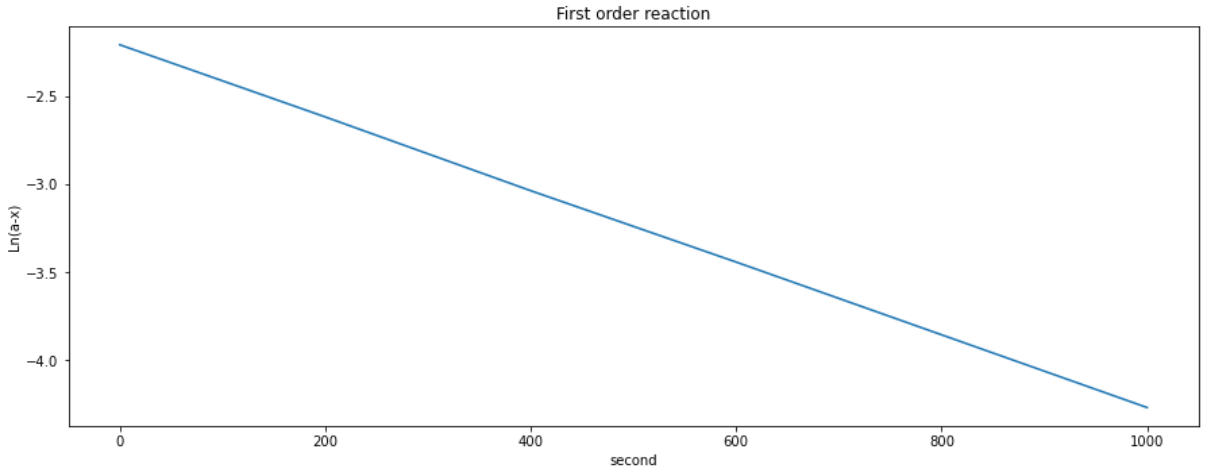
#Interactive Graphical Method by file

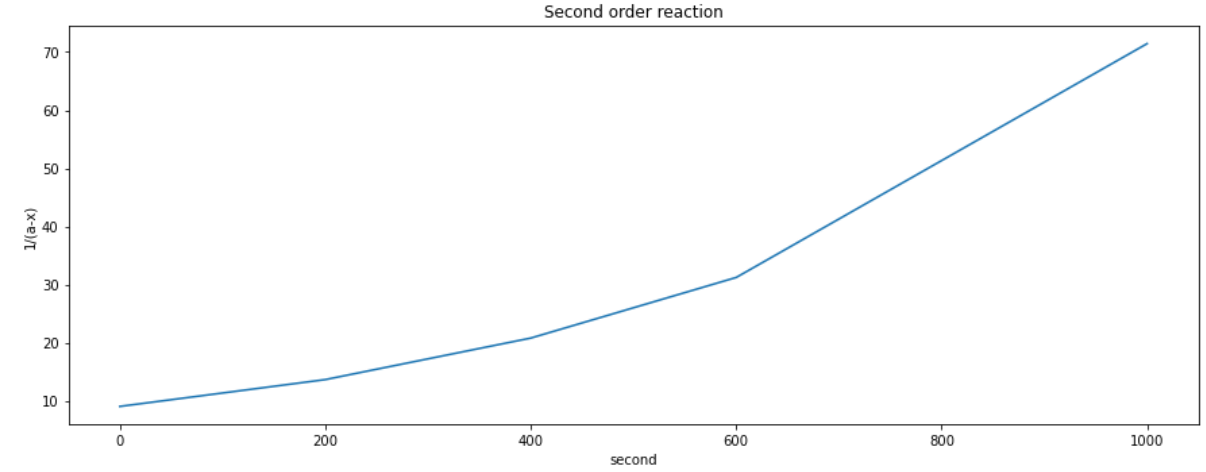
In [13]:

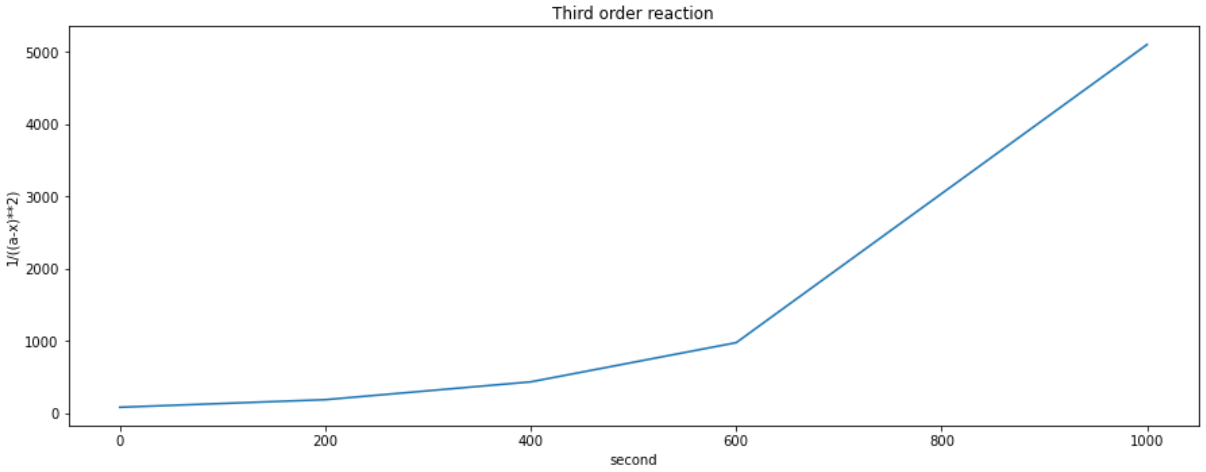
```
import math
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
import os
import glob

import numpy as np
import os
file_name=str(input('Enter your file name (ex:kinetic.csv)'))
abs_file = os.path.abspath(file_name)
print(abs_file)
data = np.genfromtxt(fname=abs_file,delimiter=',',dtype='unicode')
print(type(data))
data_no_header = data[1:]
print(data_no_header)
data_float = data_no_header.astype(np.float)
x= (data_float[:,0]) #x
y= (data_float[:,1])
c=[]
c1=[]
c2=[]
c3=[]
r=[]
A=[]
K=[]
tt=[]
import math
for i in y:
        c.append(i)# for zero order
        c1.append(math.log(i))# for 1 order
        c2.append(1/i)#for second order
        c3.append((1)/(i**2))# for third order

T=x



m= show_mplt(c,c1,c2,c3,T)

```

```
Enter your file name (ex:kinetic.csv)kinetic.csv
C:\Users\aeh-1\Desktop\2 midel\kinetic.csv
<class 'numpy.ndarray'>
[['0' '0.11']
 ['200' '0.073']
 ['400' '0.048']
 ['600' '0.032']
 ['1000' '0.014']]
```


Zero order reaction
Mol
0.10
0.08
0.06
0.04
0.02
0
200
400
600
800
1000
second


First order reaction
Ln(a-x)
−2.5
−3.0
−3.5
−4.0
0
200
400
600
800
1000
second


Second order reaction
1/(a-x)
70
60
50
40
30
20
10
0
200
400
600
800
1000
second


Third order reaction
1/((a-x)**2)
5000
4000
3000
2000
1000
0
200
400
600
800
1000
second

```
('The reactant order is', 1, ' and the reaction half-life is ', 0.001428
6045315754549, '  Second and the Constant rate is   ', 485.0887594733893
6)
```

#Interactive Graphical Method by trial

In [36]:

```
##############################################################################
##############################################################################
#Interactive Graphical Method
##############################################################################
w=widgets.Text(value='Interactive Graphical Method', disabled=True)
display(w)

print('The graphical method makes use of the concentrations of reactants. It is mos
print('\nCx= reactant concentration ')
print('\ntx= time when it is found Cx reactant concentration ')

listOfImageNames = [ "rate.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))




###############The graphical method
def G(C1,t1,C2,t2,C3,t3,C4,t4,C5,t5, Time_Unite='Sec', Concenteration_Unite='Mole'
# we make several  empty lists to store time (T) and con.   for zeroth equation and
# to plot data
    try:

        y=[]
        T=[]

        C1 =float(C1)
        # for zeroth we need just C1.
        y.append(C1)
        t1= float(t1)
        T.append(t1)

        C2=float((C2))
        y.append(C2)
        t2=float((t2))
        T.append(t2)

        C3 =float((C3))
        y.append(C3)
        t3=float((t3))
        T.append(t3)

        C4=float((C4))
        y.append(C4)
        t4=float((t4))
        T.append(t4)

        C5=float((C5))
        y.append(C5)
```

```
        t5=float((t5))
        T.append(t5)
        Time_Unite= str(Time_Unite)
        Concenteration_Unite= str(Concenteration_Unite)
        c=[]
        c1=[]
        c2=[]
        c3=[]
        r=[]
        A=[]
        K=[]
        tt=[]
        import math
        for i in y:
                c.append(i)# for zero order
                c1.append(math.log(i))# for 1 order
                c2.append(1/i)#for second order
                c3.append((1)/(i**2))# for third order
        m= show_mplt(c,c1,c2,c3,T)

        return m
    except:
        print('Fill all requirement boxes , if this massage still appears meaning t

interact(G, C1='Type only number', t1="Type only number",C2='Type only number', t2=
#########################################################
btnnnn = widgets.Button(description='Click here to find antoher reactant order by 1
display(btnnnn)

def my_event3_handler(btnnnn_object):
    w=widgets.Text(value='Interactive Graphical Method', disabled=True)
    display(w)

    interact(G, C1='1', t1="1",C2='5', t2='5', C3='2', t3="1",C4='5', t4='5', C5='5
btnnnn.on_click(my_event3_handler)


#Reference:
#https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textboo
```

C1 0.11

#Half-Life Method

In [38]:

```
#Half-Life Method
################################################################################
w=widgets.Text(value='Half-Life Method', disabled=True)
display(w)

print('\nThe Method of Half-Lives for determining the order of a reaction is to exa
print('\na1 and a2 are the reactant concentration first and second  intial reactant
print('\nt1 and t2 are the half-life's dependence on first and second  concentratio
print('\nInmportant note: all other factor may affect on the reaction should be cor
listOfImageNames = ["t.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))

def H(a1,t1,a2,t2):
    try:
        a1 =float((a1))
        t1=float((t1))
        a2=float((a2))
        t2=float((t2))
        a1 and t1 and a2 and t2>=1
        n= order_finder(a1,t1,a2,t2)


        n=1+(1/n)
        return ('The reactant order is ',round(n))
    except:
        print('Fill all requirement boxes , if this massage still appears meaning t


interact(H, a1='type reactant concentration EX:5 ', t1="type first half_live EX:5"



###############
#Half-Life Method
btnn = widgets.Button(description='Click here to find anothr reactant order by Half
display(btnn)

def my_eventt_handler(btnn_object):

    w=widgets.Text(value='Half-Life Method', disabled=True)
    display(w)


    interact(H, a1='1', t1="1",a2='5', t2='5');

btnn.on_click(my_eventt_handler)

#reference: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemis
```

Half-Life Method

The Method of Half-Lives for determining the order of a reaction is to examine the behavior of the half-life as the reaction progresses. The half-life can be defined as the time it takes for the concentration of a reactant to fall to half of its original value. The method of half-lives involved measuring the half-life's dependence on concentration. The expected behavior can be predicted using the integrated rate laws. Source: Patrick Fleming ,Assistant Professor (Chemistry and Biochemistry) at California State University East Bay

a1 and a2 are the reactant concentration first and second intial reactant concentration, respectively

t1 and t2 are the half-life's dependence on first and second concentration, respectively

Inmportant note: all other factor may affect on the reaction should be constant during this wothed

$$\left(\frac{a1}{a2}\right)^{n-1} = \frac{\left(t_{\frac{1}{2}}\right)1}{\left(t_{\frac{1}{2}}\right)2}$$

a1 1

t1 4

a2 2

t2 8

('The reactant order is ', 2)

Click here to find anothr reactant order by Half-Life Method

#Vant Hoff Differential Method

In [39]:

```
#Vant Hoff Differential Method
w=widgets.Text(value='Vant Hoff Differential Method', disabled=True)
display(w)

print("\n1)The rate of a reaction varies as the nth power of the concentration of t
print('\nThus,for two different initial concentrations C1 and C2, equations can be
print('\nC1 and C2 are initial  Concentration of reactant  ')
print('\nR1 and R2 are two rate reaction thus all other reactant and reaction envi

listOfImageNames = ["Vt.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))


def v(C1,C2,t1,t2,C3,C4,t3,t4 ):
    try:

        C1 =(float(C1))
        C2=(float(C2))
        t1=(float(t1))
        t2=(float(t2))
        C3 =(float(C3))
        C4=(float(C4))
        t3=(float(t3))
        t4=(float(t4))
        R1=(float((C2/C1)/(t2/t1)))
        R1=math.log(R1)

        R2=math.log(float((C4/C3)/(t4/t3)))
        n= order_finder(R1,C1,R2,C3)

        return ("Your reaction order is", n)
    except:
        print('Fill all requirement boxes , if this massage still appears meaning t
interact(v,C1='initial concentrations C1',C2='concentrations C1 after t1_2 time ',t
#Van't Hoff Differential Method
btnv = widgets.Button(description='Click here to find antoher reactant order by Va
display(btnv)

def my_eventv_handler(btnv_object):
    w=widgets.Text(value='Vant Hoff Differential Method', disabled=True)
    display(w)

    interact(v,C1_1='initial concentrations C1',C1_2='concentrations C1 after t1_2
btnv.on_click(my_eventv_handler)
##################################################################


#https://www.askiitians.com/iit-jee-physical-chemistry/chemical-kinetics/methods-fo

```

Vant Hoff Differential Method

1)The rate of a reaction varies as the nth power of the concentration of the reactant where (n) is the order of the reaction.
2)Thus, for two different initial concentrations C1 and C2, equations ca n be written in the form

Thus,for two different initial concentrations C1 and C2, equations can b e written in the form

C1 and C2 are initial Concentration of reactant

R1 and R2 are two rate reaction thus all other reactant and reaction env iroment constant except the reactor which wat to know its order

Van't Hoff Differential Method

- As we know that, the rate of a reaction varies as the nth power of the concentration of the reactant where 'n' is the order of the reaction.
- Thus, for two different initial concentrations C_1 and C_2, equations can be written in the form

$$log(\frac{dC_1}{dt}) = logk + nlogC_1 \quad ....(i)$$

and

$$log(\frac{dC_2}{dt}) = logk + nlogC_2 \quad ....(ii)$$

Taking logarithms,

Subtracting Eq. (ii) from (i),

$$log(\frac{dC_1}{dt}) - log(\frac{dC_2}{dt}) = n(logC_1 - logC_2)$$

C1	0.11
C2	0.073
t1	0.073
t2	200
C3	0.048
C4	0.048
t3	400
t4	500

('Your reaction order is', 4)

Click here to find antoher reactant order by Vant Hoff Differential Method

#Zero Reaction Order

In [74]:

```
#Zero Reaction Order
############################################################
w=widgets.Text(value='Zero  Reaction Order', disabled=True)
display(w)

print('\nThis Zero-order reaction means that has a rate that is independent of the
listOfImageNames = [ "zeroth.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))
print('\nC1, C2, C3, and C4 represent [A] of the reactant which is wanted to find :
print('\nt1, t2, t3, and t4 represent time [t] from begining reaction until the tr:

def Z(unite_time,unite_con,C1,t1,C2,t2,C3,t3,C4,t4):
# we need two empty list to draw zeroth equation T for time and C for concenteratio
    try:
        C=[]
        T=[]

        C1 =float(C1)
        C.append((C1))
        t1= float(t1)
        T.append(t1)

        C2=float((C2))
        C.append((C2))
        t2=float((t2))
        T.append(t2)

        C3 =float((C3))
        C.append((C3))
        t3=float((t3))
        T.append(t3)

        C4=float((C4))
        C.append((C4))
        t4=float((t4))
        T.append(t4)

        xaxil=(unite_time)
        yaxil=(unite_con)
                ###################################
        slope, intercept, r_value0, p_value, std_err = stats.linregress(C,T)
        r_value=(r_value0)
        a= (intercept)
        k= -slope
        t=(a)/(2*k)
        std_err=std_err

        title ='Zero  Reaction Order'
        unite_time= str(unite_time)# for first
```

```
        unit_k= str(unite_con)
        unit_k= str( unite_time)


        unit_k= (unite_con, '/', unite_time)
        unit_k= (('').join(str(unit_k)))
        unit_k= (('').join(unit_k))









        show_plt(C,T,title,xaxil,yaxil,unit_k )
        print("The rate constant is", k,' ',unit_k, '\nThe half-life is ', t," ",


        return
    except:
        print('Fill all requirement boxes , if this massage still appears meaning t

interact(Z,unite_time= "Type time unite",unite_con= 'Type concenteration unite',


    ############################################################ zero ORDER
    ############################################################ zero ORDER

#####################################################################
btnZ = widgets.Button(description='Click for another zeroth Reaction Order', layou
display(btnZ)


def my_eventZ_handler(btnZ_object):

    w=widgets.Text(value='Zero  Reaction Order', disabled=True)
    display(w)
    interact(Z,unite_time= "Type time unite",unite_con= 'Type concenteration unite

btnZ.on_click(my_eventZ_handler)









#####################################################################

```

First Reaction Order

In [62]:

```
#First Reaction Order
#####################################
w=widgets.Text(value='First  Reaction Order', disabled=True)
display(w)


print('\nA first-order reaction means that proceeds at a rate that depends linearly
listOfImageNames = [ "first.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))
print('\nC1, C2, C3, and C4 represent[A] of the reactant which is wanted to find it
print('\nt1, t2, t3, and t4 represent time [t] from begining reaction until the tri

def ff(unite_con, unite_time, C1,t1,C2,t2,C3,t3,C4,t4 ):
    # we need two empty list to draw zeroth equation T for time and C for concenter
    try:
        C=[]
        T=[]
        C1 =float(C1)
    # we need to take log concenteration
        C.append(math.log(C1))
        t1= float(t1)
        T.append(t1)

        C2=float((C2))
        C.append(math.log(C2))
        t2=float((t2))
        T.append(t2)

        C3 =float((C3))
        C.append(math.log(C3))
        t3=float((t3))
        T.append(t3)

        C4=float((C4))
        C.append(math.log(C4))
        t4=float((t4))
        T.append(t4)
        unite_con= str(unite_con)# for first
        unite_time= str(unite_time)# for first
        unit_k=('1/',(unite_time))
        unit_k= (('').join(unit_k)))
        title=('First order reaction')
        xaxil=unite_time
        yaxil= unite_con



        if C1 and t1 and C2 and t2  and C3 and t3 and C4 and t4>=1:
```

```
            slope, intercept, r_value, p_value, std_err = stats.linregress(T,C)
            a=math.exp(intercept)
            k=-slope
            t=(math.log(2))/(k)
            unite= (u'min\u207B\u00B9')# first

            print("The rate constant is", k,' ',unit_k, '\nThe half-life is ', t,"
        show_plt(C,T,title,xaxil,yaxil,unit_k )
        show_plt(C,T,title,xaxil,yaxil,unit_k )


        return
    except:

        print('Fill all requirement boxes , if this massage still appears meaning t

interact(ff,unite_con='mol',unite_time='s', C1='Type only number',t1='Type only nur
##################################################################SECOND ORDER
##################################################################SECOND ORDER

#####################################################################
btnff = widgets.Button(description='Click here for another first Reaction Order', l
display(btnff)
def my_eventff_handler(btnff_object):

    w=widgets.Text(value='First  Reaction Order', disabled=True)
    display(w)

    interact(ff,unite_con='mol',unite_time='s', C1='Type only concentration number


btnff.on_click(my_eventff_handler)
```

Second Reaction Order

In [63]:

```
#Second Reaction Order

w=widgets.Text(value='Second  Reaction Order', disabled=True)
display(w)

print('\nThe simplest kind of second-order reaction is one whose rate is proportion
listOfImageNames = [ "second.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))

print('\nC1, C2, C3, and C4 represent[A] of the reactor which is wanted to find its
print('\nt1, t2, t3, and t4 represent time[t] from begining reaction until the tria
print('\nIf the reaction has two reactors it must be costant one of these reactor c


#################################################################
def Sc(unite_con,unite_time,C1,t1,C2,t2,C3,t3,C4,t4):
    try:
        C=[]
        T=[]

        C1 =float(C1)
        C.append(1/(C1))
        t1= float(t1)
        T.append(t1)

        C2=float((C2))
        C.append(1/(C2))
        t2=float((t2))
        T.append(t2)

        C3 =float((C3))
        C.append(1/(C3))
        t3=float((t3))
        T.append(t3)

        C4=float((C4))
        C.append(1/(C4))
        t4=float((t4))
        T.append(t4)
        unite_con= str(unite_con)# for first
        unite_time= str(unite_time)# for first
        unit_k=('1/',unite_time,'*', unite_con)
        unit_k= (('').join(unit_k)))


        if C1 and t1 and C2 and t2 and C3 and t3 and C4 and t4>=1:
            slope, intercept, r_value, p_value, std_err = stats.linregress(T,C)
            a=1/(intercept)
            k=slope
```

```
            t=(1)/(k*a)

            xaxil=(unite_time)
            yaxil=(unite_con)

            title= ('Second order reaction')
            show_plt(C,T,title,xaxil,yaxil,unit_k )



            print("The rate constant is", k,' ',unit_k, '\nThe half-life is ', t,"


    except:

        print("Fill all requirement boxes , if this massage still appears meaning t

interact(Sc,unite_con='mol',unite_time='s', C1='Type only concentration number',t1=
##################################################
btnSc = widgets.Button(description='Second Reaction Order(all the same reactant af
display(btnSc)

def my_eventSc_handler(btnSc_object):
    w=widgets.Text(value='Second  Reaction Order', disabled=True)
    display(w)


    interact(Sc,unite_con='type concentration unit',unite_time='type unite time', (

btnSc.on_click(my_eventSc_handler)

```

Second Reaction Order

The simplest kind of second-order reaction is one whose rate is proporti
onal to the square of the concentration of one reactant. These generally
have the form 2A → products. A second kind of third-order reaction has a
reaction rate that is proportional to the product of the concentrations
of two reactants.

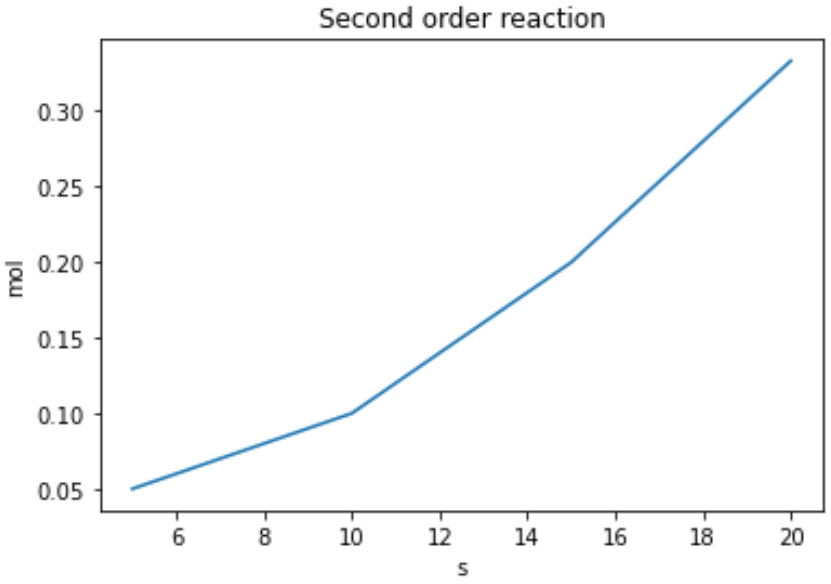
Reaction Order	Differential Rate Law	Integrated Rate Law	Characteristic Kinetic Plot	Slope of Kinetic Plot	Units of Rate Constant
Second	$\frac{-d[A]}{dt} = k[A]^2$	$[A] = \frac{[A]^0}{1 + k\,t\,[A]^0}$	1/[A] vs *t*	*k*	L mole^{-1} sec^{-1}

C1, C2, C3, and C4 represent[A] of the reactor which is wanted to find i ts order and their units must be Mole in this program

t1, t2, t3, and t4 represent time[t] from begining reaction until the tr ial to find rest concenteration and their units must be Minutes in thi s program

If the reaction has two reactors it must be costant one of these reactor or they have same concentration

unite_con	mol
unite_time	s
C1	20
t1	5
C2	10
t2	10
C3	5
t3	15
C4	3
t4	20




```
The rate constant is 0.019   1/s*mol
The half-life is  -3.5087719298245603   s
The initial conccenteration is  -15.000000000000004   mol
The cofficent correlation is  0.9811557810392123
The stander deviation is  0.0026457513110645903
```

Second Reaction Order(all the same reactant affecting on rate constant have sam…

Second Order Reaction with multiple reactants

In [64]:

```
#333333333333333two reactants
w=widgets.Text(value='Second Order Reaction with multiple reactants', disabled=Tru
display(w)
listOfImageNames = [ "secondm.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))

print('k_A and k_B indicate to a and b costant rate for A and B raectant, respecti
print('A_0 and B_0 indicate to [A0] and[B0] initial concenteration of A and B reac
print('C_Ax and C_Bx inidcate to [A] and [B] concentration of A and B in time tx t

def Scc(n,unite_con, unite_time, k_A,k_B,A_0,B_0, C_A1,C_B1,t1,C_A2,C_B2,t2,C_A3,C
    try:
        T=[]
        b=[]
        y=[]

        B=[]#C.B
        B1=[]#1/C
        T=[]
        n=float((n))
        k_A=float((k_A))
        k_B=float((k_A))
        A_0=float(A_0)
        A_10=1/A_0
        B_0=float(B_0)
        B_0=1/B_0
        CC=((k_B*A_0)-(k_B*A_0))
        CC0=math.log(A_0/B_0)

        C_A1 =float(C_A1)
        C_B1 =float(C_B1)
        x1= math.log(float(C_A1/C_B1))
        b.append(x1)
        t1= float(t1)
        T.append((t1))
#####################################
        C_A2=float((C_A2))
        C_B2 =float(C_B2)
        B.append(1/(C_B2))
        x2= math.log(C_A2/C_B2)
        b.append(x2)

        t2=float(((t2)))
        T.append(t2)

#####################################################################

        C_A3 =float((C_A3))
        C_B3 =float(C_B3)
```

```
        x3= math.log(C_A3/C_B3)
        b.append(x3)
        t3=float((t3))
        T.append((n-1)*t3)
     ######################################################################

        C_A4=float((C_A4))
        C_B4 =float(C_B4)
        x4= math.log(C_A4/C_B4)
        b.append(x4)

        t4=float((t4))


########################################
        T.append((n-1)*t4)
        for i in T:
            i=i*CC0
            y.append(i)
        unite_con= str(unite_con)# for first
        unite_time= str(unite_time)# for first

                #slope, intercept, r_value1, p_value, std_err = stats.linregress(F
        slope, intercept, r_value1, p_value, std_err = stats.linregress(b,y)

        r1= (r_value1)

        k= float(slope)


        t3=float(((2-((k_B)*(A_0)))/(k_A*(B_0))))
        t3=math.log(abs(t3))
        t33 = float(k_B)*(float(t3))

        t=float(t3/t2)


        xaxil=(unite_time)
        yaxil=(unite_con)

        title=('Second order reaction')

        unit_k=('1/',unite_time+unite_con)
        unit_k= (''.join(unit_k)))

        print("The rate constant is", k,' ',unit_k, '\nThe half-life is ', t," ",

        show_plt(b,y,title,xaxil,yaxil,unit_k )
    except:

        print("Fill all requirement boxes , if this massage still appears meaning


```

```
104
105
106
107
108 interact(Scc,n='Type chemical order is 2', unite_con='type reactant concenteration
109
110
111 ##########################Type only number####THIRD ORDER
112 btnScc = widgets.Button(description='Click here for another second Reaction Order(
113 display(btnScc)
114
115 def my_eventScc_handler(btnScc_object):
116
117     w=widgets.Text(value='Second Order Reaction with multiple reactants', disabled
118     display(w)
119
120     interact(Scc,n='2', unite_con='type reactant concenteration unite', unite_time
121
122 btnScc.on_click(my_eventScc_handler)
123
124 #https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbo
125
126
127 #https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbo
128
```

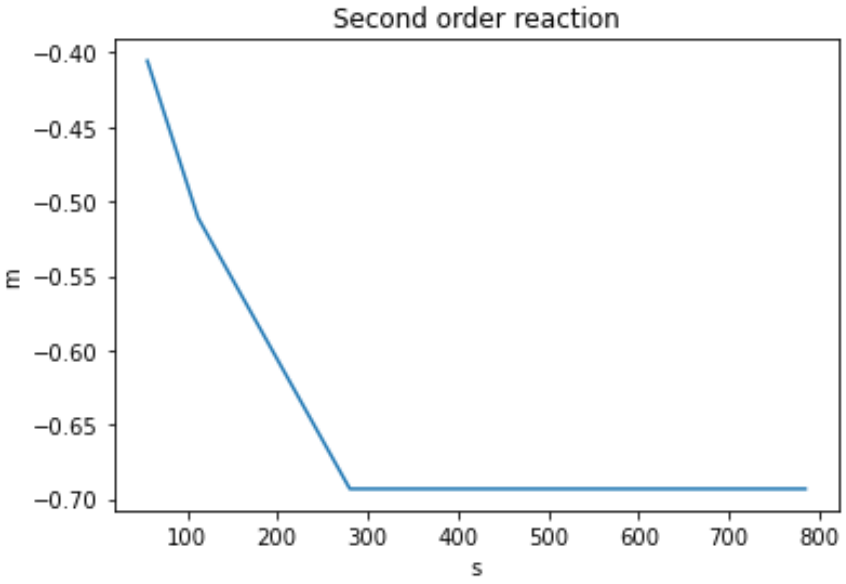
Second Order Reaction with multiple reactants

Reaction order	Differential equation	Integrated equation	Half-life $t_{1/2}$ (A)
2	$\frac{dx}{dt} = k[A][B]$	$\ln\frac{[A]}{[B]} = kt(b[A_0] - a[B_0]) + \ln\frac{[A_0]}{[B_0]}$	$\frac{\ln(2 - b[A_0]/a[B_0])}{k(a[B_0] - b[A_0])}$

```
k_A and k_B indicate to a and b costant rate for A and B raectant, respe
ctively
A_0 and B_0 indicate to [A0] and[B0] initial concenteration of A and B r
eactants, respectively
C_Ax and C_Bx inidcate to [A] and [B] concentration of A and B in time t
x thus x indicates to trial number
```

n	2
unite_con	m
unite_time	s
k_A	100
k_B	222
A_0	222
B_0	333
C_A1	200
C_B1	300
t1	5
C_A2	150
C_B2	250
t2	10
C_A3	100
C_B3	200
t3	25
C_A4	75
C_B4	150
t4	70

```
The rate constant is -1782.5289601170032   1/sm
The half-life is  1.1210729777704276   s
The cofficent correlation is  -0.7650707429241049
The stander deviation is  1060.8881855465586
```



Click here for another second Reaction Order(two reactants affecting on rate cons…

Third Reaction Order

In [78]:

```
listOfImageNames = [ "third.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))

print('\nC1, C2, C3, and C4 indicate to [A]t  concentration of the reactant which :
print('\nt1, t2, t3, and t4 indicate to (t) time from begining reaction until the t


##########################Type only number####THIRD ORDER

def RDd(C1,t1,C2,t2,C3,t3,C4,t4):
    C=[]
    T=[]
    try:
        C1 =float(C1)
        C.append(1/(C1**2))
        t1= float(t1)
        T.append(t1)

        C2=float((C2))
        C.append(1/(C2**2))
        t2=float((t2))
        T.append(t2)

        C3 =float((C3))
        C.append(1/(C3**2))
        t3=float((t3))
        T.append(t3)

        C4=float((C4))
        C.append(1/(C4**2))
        t4=float((t4))
        T.append(t4)
        if  C1 and t1 and C2 and t2 and C3 and t3 and C4 and t4>=1:

            slope, intercept, r_value, p_value, std_err = stats.linregress(T,C)
            a=1/(intercept)
            a=math.sqrt(a)
            k=2*slope
            t=(3)/(k*(a**2))
            unite= (u'Mol\u207B\u00B2'  u'min\u207B\u00B9')# third

            xlabel= ("Minutes")
            ylabel= ("1/(a-x)")

            title= ('Third order reaction')
            show_plt(C,T,title,xlabel,ylabel,unite )


            print("The rate constant is", k,'', unite, '\nThe half-life is ', t,'m
```

```
        return
    except:
        print("Fill all requirement boxes , if this massage still appears meaning t

interact(RDd,unite_con='mol',unite_time='s', C1='Type only concentration number',ti

##################Nth Order
#####################################################################
btnrdd = widgets.Button(description='Click here for another third Reaction Order(al
display(btnrdd)

def my_eventrd_handler(btnrd_object):

    w=widgets.Text(value='Third  Reaction Order', disabled=True)
    display(w)
    print('\nThe simplest kind of third-order reaction is one whose rate is proport

    interact(RDd,unite_con='mol',unite_time='s', C1='Type only concentration numbe

btnrdd.on_click(my_eventrd_handler)
####################################
```

(B3) A third-order reaction obeys the rate law

$$\frac{d[A]}{dt} = -k[A]^3$$

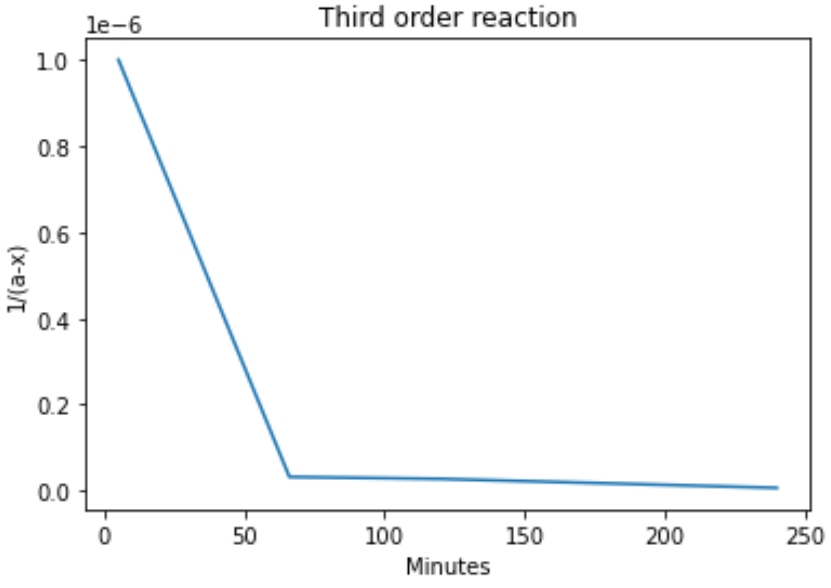
Prove that the *integrated rate law* is

$$\frac{1}{2[A]_t^2} = \frac{1}{2[A]_0^2} + kt$$

C1, C2, C3, and C4 indicate to [A]t concentration of the reactant which is wanted to find its order and their units must be Mole in this progra m

t1, t2, t3, and t4 indicate to (t) time from begining reaction until the trial to find rest concenteration and their units must be Minutes in t his program

C1	1000
t1	5
C2	5555
t2	66
C3	6000
t3	120
C4	12000
t4	240



```
The rate constant is -6.869853963246135e-09  Mol⁻²min⁻¹
The half-life is  -278.1262432109379 min
The initial conccenteration is  1253.042755658759 Mol
Tthe cofficent correlation is -0.701835023015162
The stander deviation is  2.4652133902880107e-09
```

Click here for another third Reaction Order(all the reactant affecting on rate const…

Third Reaction Order two reactants hacing different concentration affect on its rate

In [66]:

```
########################################
w=widgets.Text(value='Third  Reaction Order', disabled=True)
display(w)

print('k_A and k_B indicate to costant rate for A and B raector, respectively ')
print('A_0 and B_0 indicate to initial concenteration of A and B reactants, respec
print('C_Ax and C_Bx inidcate to A and B concentration of A and B in time tx thus

print('\nThe simplest kind of third-order reaction is one whose rate is proportion

def THt(n,conc_unit,time_unit, k_A,k_B,A_0,B_0, C_A1,C_B1,t1,C_A2,C_B2,t2,C_A3,C_B
    try:


        C0=[]
        C=[]#C.A
        C1=[]#1/C
        C2=[]
        B0=[]
        F=[]
        x=[]

        B=[]#C.B
        B1=[]#1/C
        T=[]
        n=float((n))
        time_unit=str(time_unit)
        conc_unit=str(conc_unit)
        k_A=float((k_A))
        k_B=float((k_A))
        A_0=float(A_0)
        A_10=1/A_0
        B_0=float(B_0)
        B_0=1/B_0
        CC=((k_A*B_0)-(k_B*A_0)**2)
        CC0=((k_A*B_0)-(k_B*A_0))

        C_A1 =float(C_A1)
        C0.append(C_A1)
        m= (1/(C_A1))
        p=m-A_10
        f1=p/CC0

        C_B1 =float(C_B1)
        B0.append(C_B1)
        B.append(1/(C_B1))

        ###################################
        b=(C_A1*B_0)/(C_B1*A_0)
        b=math.log(b)
```

```
        b=f1+(k_B*b)
        x.append(b)


        t1= float(t1)
        T.append((n-1)*t1)
###################################
        C_A2=float((C_A2))
        C0.append(C_A2)
        C.append(1/(C_A2))
        m= (1/(C_A2))
        p=m-A_10
        f2=p/CC0

        C_B2 =float(C_B2)
        B0.append(C_B2)
        B.append(1/(C_B2))

        b=(C_A2*B_0)/(C_B2*A_0)
        b=math.log(b)
        b=f2+(k_B*b)
        x.append(b)

#####################################################################
        t2=float(((t2)))
        T.append((n-1)*t2)

        C_A3 =float((C_A3))
        C0.append(C_A3)
        m= (1/(C_A3))
        p=m-A_10
        f3=p/CC0


        C.append(1/(C_A3))

        C_B3 =float(C_B3)
        B.append(1/(C_B3))
        B0.append(C_B3)
        b=(C_A3*B_0)/(C_B3*A_0)
        b=math.log(b)
        b=f3+(k_B*b)
        x.append(b)


    ###################################################################
        t3=float((t3))
        T.append((n-1)*t3)

        C_A4=float((C_A4))
        C0.append(C_A4)
        m= (1/(C_A4))
        p=m-A_10
```

```
        f4=p/CC0

        C.append(1/(C_A4))

        C_B4 =float(C_B4)
        B.append(1/(C_B4))
        B0.append(C_B4)
        b=(C_A4*B_0)/(C_B4*A_0)
        b=math.log(b)
        b=f4+(k_B*b)
        x.append(b)
        t4=float((t4))
        T.append((n-1)*t4)


        #slope, intercept, r_value1, p_value, std_err = stats.linregress(F,T)
        slope, intercept, r_value1, p_value, std_err = stats.linregress(x,T)

        r1= (r_value1)

        k= float(slope)

        t1=float(k*(A_0)*(B_0*k_A)-(A_0*k_B))
        t1=1/t1

        t2=float(((k*((k_A*(B_0)-k_B*(A_0))**2))))
        t3=float(((2-k_B)*(A_0))/(k_A*(B_0)))
        t3=math.log(abs(t3))
        t3=k_B*t3

        t=float(t3/t2)
        t=float(t1-t)

        plt.figure()
        plt.xlabel(time_unit)
        plt.ylabel(conc_unit)

        plt.plot(x, T )
        plt.title('nth order reaction')

        print ("The rate constant is", k , '\nThe Half-Life is ', t,' min \nThe co

        return
    except:

        print("Fill all requirement boxes , if this massage still appears meaning

interact(THt,n='3',conc_unit='mol', time_unit='sec', k_A='Type A reactant constant

btnTht = widgets.Button(description='Click here for another third Reaction Order(t

display(btnTht)

```

```
157 def my_eventTht_handler(btnTht_object):
158
159     w=widgets.Text(value='Third  Reaction Order', disabled=True)
160     display(w)
161
162     interact(THt,n='3',conc_unit='mol', time_unit='sec', k_A='Type A reactant cons
163
164 btnTht.on_click(my_eventTht_handler)
165
166
```

Third Reaction Order

```
k_A and k_B indicate to costant rate for A and B raector, respectively
A_0 and B_0 indicate to initial concenteration of A and B reactants, res
pectively
C_Ax and C_Bx inidcate to A and B concentration of A and B in time tx th
us x indicates to trial number

The simplest kind of third-order reaction is one whose rate is proportio
nal to the power three of the concentration of one reactant. A second ki
nd of third-order reaction has a reaction rate that is proportional to t
he product of the concentrations of three reactants.
```

n	3
conc_unit	mol
time_unit	sec
k_A	Type A reactant constant rate
k_B	Type B reactant constant rate
A_0	Type A initial concenteration
B_0	Type B initial concenteration
C_A1	Type A reactant concenteration
C_B1	Type A reactant concenteration
t1	Type time
C_A2	Type A reactant concenteration
C_B2	Type B reactant concenteration
t2	Type time
C_A3	Type A reactant concenteration

C_B3	Type B reactant concenteration
t3	Type time
C_A4	Type A reactant concenteration
C_B4	Type B reactant concenteration
t4	Type time

```
Fill all requirement boxes , if this massage still appears meaning ther
e is mistake in your details
```

Click here for another third Reaction Order(two reactants affecting on rate consta…

nTH order

In [67]:

```
print('\nThis for  Reaction order  which is bigger than first order to calculate ra
print('\nC1, C2, C3, and C4 represent rest concentration of the reactor which is wa
print('\nt1, t2, t3, and t4 represent time from begining reaction until the trial t

def TH(n, C1,t1,C2,t2,C3,t3,C4,t4):
    try:
        C=[]
        T=[]
        n=float((n))
        C1 =float(C1)
        C.append(1/(C1**(n-1)))
        t1= float(t1)
        T.append((n-1)*t1)

        C2=float((C2))
        C.append(1/(C2**(n-1)))
        t2=float(((t2)))
        T.append((n-1)*t2)

        C3 =float((C3))
        C.append(1/(C3**(n-1)))
        t3=float((t3))
        T.append((n-1)*t3)

        C4=float((C4))
        C.append(1/(C4**(n-1)))
        t4=float((t4))
        T.append((n-1)*t4)
        if C1 and t1 and C2 and t2 and C3 and t3 and C4 and t4>=1:

            slope, intercept, r_value, p_value, std_err = stats.linregress(T,C)
            a=(1/intercept)
            a=math.exp(a)
            a=(n-1)*a
            k=(n-1)*slope
            t=((2**(n-1))-1)/((n-1)*(k)*(a**(n-1)))

            plt.figure()
            plt.xlabel("Minutes")
            plt.ylabel("(a-x)")

            plt.plot(T, C )
            plt.title('nth order reaction')

            print ("The rate constant is", k , '\nThe Half-Life is ', t,' min \nThe

            return
    except:
        print("Fill all requirement boxes , if this massage still appears meaning t

```

```
interact(TH,n='Type only number', C1='Type only number',t1='Type only number',C2='
```

```

btnNTH = widgets.Button(description='NTH  Reaction Order\n(all the same reactant a
display(btnNTH)



def my_eventFF_handler(btnNTH_object):

    w=widgets.Text(value='Click here for another nTH  Reaction Order', disabled=Tru
    display(w)

    interact(TH,n='2', C1='1',t1='2',C2='3',t2='4',C3='4',t3='5',C4='5',t4='5');

btnNTH.on_click(my_eventFF_handler)

```

This for Reaction order which is bigger than first order to calculate rate constant, Half-life time, correlation coefficient, and standard dev iation

C1, C2, C3, and C4 represent rest concentration of the reactor which is wanted to find its order and their units must be Mole in this program

t1, t2, t3, and t4 represent time from begining reaction until the trial to find rest concenteration and their units must be Minutes in this pr ogram

n	Type only number
C1	Type only number
t1	Type only number
C2	Type only number
t2	Type only number
C3	Type only number
t3	Type only number
C4	Type only number
t4	Type only number

Fill all requirement boxes , if this massage still appears meaning ther e is mistake in your details

NTH Reaction Order (all the same reactant affecting on rate constant have sa…

actiation energy activation energy arrhenius equation with two temoerature and two constant rates points

In [79]:

```
print(' Activation Energy is the energy difference between the reactants and the ac
listOfImageNames = [ "ea2.png"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))

print('\nk1,k2 are constant rate of reaction which occured in different temperature
print('\nT1,T2 are temperatures degrees in kelvin degree')

def Ea(k1,T1,k2,T2):

    try:
        k1=float((k1))
        T1=float((T1))
        k2=float((k2))
        T2=float((T2))
        k1 and T1 and k2 and T2>1
        k=math.log(k2/k1)
        t=(1/T1)-(1/T2)
        Ea=float((k/t)*(8.314))
        return ('The actiavtion energy for this reaction is', Ea,'KJ/mol')
    except:
        print("Fill all requirement boxes , if this massage still appears meaning t


interact(Ea,k1="Type only number",T1="Type only number",k2="Type only number",T2="T









btnEa = widgets.Button(description='"Two-Point Form" of the Arrhenius Equation', la
display(btnEa)
def my_eventEa_handler(btnEa_object):



        w=widgets.Text(value='Activation Energy', disabled=True)
        display(w)

        print('\nThis will calculate actiavation energy \n')

        interact(Ea,k1="1",T1="2",k2="3",T2="4");

btnEa.on_click(my_eventEa_handler)
```

```
51
```

Activation Energy is the energy difference between the reactants and the activated complex, also known as transition state. In a chemical reaction, the transition state is defined as the highest-energy state of the system. If the molecules in the reactants collide with enough kinetic energy and this energy is higher than the transition state energy, then the reaction occurs and products form. In other words, the higher the activation energy, the harder it is for a reaction to occur and vice versa.

$$\ln\frac{k_2}{k_1} = \frac{-Ea}{R}\left(\frac{1}{T_2} - \frac{1}{T_1}\right) \qquad \ln\frac{k_2}{k_1} = \frac{Ea}{R}\left(\frac{1}{T_1} - \frac{1}{T_2}\right)$$

k1,k2 are constant rate of reaction which occured in different temperatures degres

T1,T2 are temperatures degrees in kelvin degree

k1: 1
T1: 2
k2: 2
T2: 6

```
('The actiavtion energy for this reaction is', 17.288476977526155, 'KJ/mol')
```

"Two-Point Form" of the Arrhenius Equation

actiation energy activation energy arrhenius equation with four temoerature and four constant rates points

In [80]:

```
print('Activation Energy is the energy difference between the reactants and the act
listOfImageNames = [ "ea.JPG"]
for imageName in listOfImageNames:
    display(Image(filename=imageName))

print('\nk1,k2,k3,4 are constant rate of reaction which occured in different temper
print('\nT1,T2,T3,T4 are temperatures degrees in kelvin degree')
def Eaa(k1,T1,k2,T2, k3,T3,k4,T4):

    try:

        T0=[]
        k0=[]
        T=[]
        k=[]

        k1=float((k1))
        k1=math.log(k1)
        k0.append(k1)
        T1=float((T1))
        T1=(1/T1)
        T0.append(T1)


        k2=float((k2))
        k2=math.log(k2)
        k0.append(k2)
        T2=float((T2))
        T2=(1/T2)
        T0.append(T2)



        k3=float((k3))
        k3=math.log(k3)
        k0.append(k3)

        T3=float((T3))
        T3=(1/T3)
        T0.append(T3)


        k4=float((k4))
        k4=math.log(k4)
        k0.append(k4)

        T4=float((T4))
        T4=(1/T4)
        T0.append(T4)

```

```
51
52
53
54
55          slope, intercept, r_value, p_value, std_err = stats.linregress(T0,k0)
56          Ea= slope*8.314
57
58          x= ("1/T")
59          y= ("Ln(k)")
60          title=('The Arrhenius Equation')
61
62          title=('Zero order reaction')
63          unit_k= 'KJ/mol'
64
65
66          show_plt(T0, k0,title,x,y,unit_k )
67
68
69
70          print('The actiavtion energy for this reaction is', Ea,'KJ/mol')
71
72          return
73      except:
74          print("Fill all requirement boxes , if this massage still appears meaning t
75  interact(Eaa,k1="Type only number",T1="Type only number",k2="Type only number",T2="
76  btnEaa = widgets.Button(description='"Variation of the rate constant with temperatu
77  display(btnEaa)
78  def my_eventEaa_handler(btnEaa_object):
79
80
81          print('The activation energy ( Ea ) is the energy difference between the re
82
83          w=widgets.Text(value='Activation Energy', disabled=True)
84          display(w)
85
86          print('\nThis will calculate actiavation energy \n')
87
88          interact(Eaa,k1="1",T1="2",k2="3",T2="4", k3="1",T3="2",k4="3",T4="4");
89
90  btnEaa.on_click(my_eventEaa_handler)
91
92
```

Activation Energy is the energy difference between the reactants and the activated complex, also known as transition state. In a chemical reaction, the transition state is defined as the highest-energy state of the system. If the molecules in the reactants collide with enough kinetic energy and this energy is higher than the transition state energy, then the reaction occurs and products form. In other words, the higher the activation energy, the harder it is for a reaction to occur and vice versa.

$$k = Ae^{-\frac{E_a}{RT}} \quad or \quad ln\ k = -\frac{E_a}{RT} + ln\ A$$

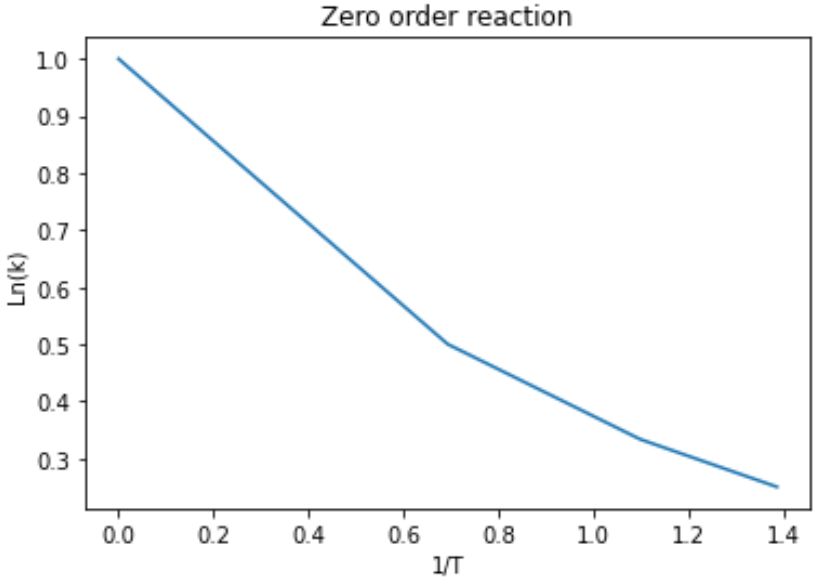
Where:

k =	Chemical Reaction Rate
A =	Pre-exponential Factor
E_a =	Activation Energy
R =	Gas Constant
T =	Temperature in Kelvin

k1,k2,k3,4 are constant rate of reaction which occured in different temp eratures degres

T1,T2,T3,T4 are temperatures degrees in kelvin degree

k1	1
T1	1
k2	2
T2	2
k3	3
T3	3
k4	4
T4	4



```
The actiavtion energy for this reaction is -14.63391113289941 KJ/mol
```

"Variation of the rate constant with temperature" of the Arrhenius Equation

find how much current is used to produce specific amount of specific metal in Galvanic Cell

In [81]:

```
w=widgets.Text(value='find period time to electroplate a flute', disabled=True)
display(w)

print("This program will find how much current is used to produce specific amount o
print("\nch_element box: The symbol of a chemical element. " )
print("\nmoles_tr box:  The moles of electrons transferred. " )
print("\nAmount_gram box: type amount of chemcial material in gram unit. " )
print("\nTime box: type time in second unit. " )


def GTV( ch_element,  Amount_gm,moles_tr, time):


    try:
        ch_element=str(ch_element)
        moles_tr= float(moles_tr)
        Amount_gm = float(Amount_gm )
        time=float(time)
        #chemlib.electrochemistry.electrolysis(element: str, n: int, **kwargs)
        z= electrolysis(ch_element, float(moles_tr),  grams= float(Amount_gm) , se

        return   z

    except:
        print("Fill all requirement boxes with their requirements , if this massage
interact(GTV,ch_element='Cu', moles_tr= '2', time= '25 ', Amount_gm= '600');
btnelecttGTV = widgets.Button(description='click here to Find how much current is u
display(btnelecttGTV)
def my_eventbtnelecttGTV_handler(btnelecttGTV_object):



        w=widgets.Text(value='find period time to electroplate a flute', disabled=T
        display(w)

        interact(GTV,ch_element='Cu', moles_tr= '2', time= '25 ', Amount_gm= '600'

btnelecttGTV.on_click(my_eventbtnelecttGTV_handler)
# Reference: https://chemlib.readthedocs.io/en/latest/electrochemistry.html#galvan
```

find period time to electroplate a flute

```
This program will find how much current is used to produce specific am
ount of specific metal  in Galvanic Cell.

ch_element box: The symbol of a chemical element.

moles_tr box:  The moles of electrons transferred.
```

```
Amount_gram box: type amount of chemcial material in gram unit.

Time box: type time in second unit.
```

ch_element: Cu

Amount_gm: 600

moles_tr: 2

time: 25

```
{'element': 'Cu',
 'n': 2.0,
 'seconds': 25.0,
 'amps': 72880.74780473986,
 'grams': 600.0}
```

click here to Find how much current is used to produce specific amount of metal i…

#This program will find period time to electroplate a flute

In [82]:

```
#This program will find period time to electroplate a flute
from chemlib import electrolysis
from chemlib import Galvanic_Cell
from ipywidgets import interact, interactive, fixed, interact_manual
w=widgets.Text(value='find period time to electroplate a flute', disabled=True)
display(w)

print("This program will find period time to electroplate a flute")
print("\nch_element box: The symbol of a chemical element. " )
print("\nmoles_tr box:  The moles of electrons transferred. " )
print("\nAmount_gram box: type amount of chemcial material in gram unit. " )

def GTt( ch_element,moles_tr, applying_current, Amount_gm):

    try:
        ch_element=(str(ch_element))
        moles_tr= float(moles_tr)
        applying_current =float(applying_current)
        Amount_gm = float(Amount_gm )
        #chemlib.electrochemistry.electrolysis(element: str, n: int, **kwargs)

        return   electrolysis(ch_element, float(moles_tr), amps = float(applying_cu

    except:
        print("Fill all requirement boxes with Integer numbers , if this massage st


interact(GTt,ch_element='Cu', moles_tr= '2', applying_current= '100', Amount_gm =
btnelectt = widgets.Button(description='Find period time to electroplate a flute',
display(btnelectt)
def my_eventbtnelectt_handler(btnelectt_object):



        w=widgets.Text(value='Click here to find period time to electroplate ', dis
        display(w)

        interact(GTt,ch_element='Cu', moles_tr= '2', applying_current= '100', Amou

btnelectt.on_click(my_eventbtnelectt_handler)
# Reference: https://chemlib.readthedocs.io/en/latest/electrochemistry.html#galvan
```

find period time to electroplate a flute

```
This program will find period time to electroplate a flute

ch_element box: The symbol of a chemical element.
```

```
moles_tr box:  The moles of electrons transferred.

Amount_gram box: type amount of chemcial material in gram unit.
```

ch_element	Cu
moles_tr	2
applying_c…	100
Amount_gm	600

```
{'element': 'Cu',
 'n': 2.0,
 'seconds': 18220.186951184965,
 'amps': 100.0,
 'grams': 600.0}
```

Find period time to electroplate a flute

```
# All these previous pictures from google
```