In [1]:
```
import numpy as np
from scipy.integrate import odeint
from scipy.optimize import curve_fit
from scipy.optimize import differential_evolution
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
import pandas as pd
```

In [2]:
```
from endpoint_maker import sqres
from endpoint_maker import func
from endpoint_maker import t
from endpoint_maker import new
from endpoint_maker import lightdata
from endpoint_maker import rawdata
```

In [3]:
```
if __name__ == '__main__':
    def initparams():
        #bounds = ([0.00001,10000],[0,4],[0.00001,1000],[0.00001,1000],[0.00001,100
        bounds = ([0.00001,1000],[0.00001,1000],[0.00001,1000],[0.00001,1000])
        result = differential_evolution(sqres,bounds,maxiter=100,popsize=20,polish=
        return result
```

In [4]:
```
initialp = initparams()
print(initialp)
```

```
message: Maximum number of iterations has been exceeded.
success: False
    fun: 0.38314242749142585
      x: [ 4.820e+02  2.804e-02  1.371e-04  1.552e+01]
    nit: 100
   nfev: 8080
```

In [5]:
```
#def tester(t,Kd,n,d2,k2,k3,i):
def tester(t,d2,k2,k3,k7,i):

    inivalues = [1,0,0,0,0,0,0]
    arrayvalues = np.asarray([])

    #for i in range(len(lightdata[:,0])):
    def I(t):
        tindex = t/5
        if tindex > 12241:
            tindex = 12240
        return lightdata[i][int(tindex)]

    #def odes(z,t,Kd,n,d2,k2,k3):
    def odes(z,t,d2,k2,k3,k7):
        Pu,Pb,Pa,mRNA,mCherry1,mCherry2,mCherry3 = z
        d1 = 0.019905
        k1 = 0.08299
        Kd = 90.41
        n = 0.964487
        #d2= 486.67
        #k2= 6.597
        #k3= 0.0539


        d3 = 0.000077
        k4 = 1.25
        d4 = 0.000031
        k5 = 0.00283
        k6 = 0.00283

        Pu = z[0]
        Pb = z[1]
        Pa = z[2]
        mRNA = z[3]
        mCherry1 = z[4]
        mCherry2 = z[5]
        mCherry3 = z[6]

        dPudt = d1*Pb - k1*I(t)**n/(Kd**n+I(t)**n)*Pu - k7*I(t)**n/(Kd**n+I(t)**n)*
        dPbdt = k1*I(t)**n/(Kd**n+I(t)**n)*Pu - k2*Pb - d1*Pb + d2*Pa
        dPadt = k2*Pb - d2*Pa + k7*I(t)**n/(Kd**n+I(t)**n)*Pu
        dmRNAdt = k3*Pa - d3*mRNA
        dmCherry1dt = k4*mRNA-(d4 + k5)*mCherry1
        dmCherry2dt = k5*mCherry1-(d4+k6)*mCherry2
        dmCherry3dt = k6*mCherry2 - d4*mCherry3



        return [dPudt,dPbdt,dPadt,dmRNAdt,dmCherry1dt,dmCherry2dt,dmCherry3dt]

    #solver = odeint(odes,inivalues,t,args = (Kd,n,d2,k2,k3),hmax=0.1)
    solver = odeint(odes,inivalues,t,args = (d2,k2,k3,k7),hmax=0.1)

    mCherryout = solver[:,6]
    #mCherryout = mCherryout[0:24480:240]
    return mCherryout
```

In [6]:
```
print(initialp.x)
```

```
[4.81958516e+02 2.80370210e-02 1.37062868e-04 1.55200190e+01]
```

In [7]:
```
popt, covt = curve_fit(func,t,new,initialp.x,maxfev=1000000)

#popt, covt = curve_fit(func,t,newdata,initialp.x, maxfev=10000000, bounds=((0.0000
#popt, covt = curve_fit(func,t,newdata,initialp.x, maxfev=1000000)

#Kd,n,d2,k2,k3 = popt
#print('Kd=',Kd,'n=',n,'d2=',d2,'k2=',k2,'k3=',k3)

#d3,k4,d4,k5,k6 = popt
#print('d3=',d3,'k4=',k4,'d4=',d4,'k5=',k5,'k6=',k6)

d2,k2,k3,k7 = popt
print('d2=',d2,'k2=',k2,'k3=',k3,'k7=',k7)
```

```
C:\Users\Leandra\anaconda3\Lib\site-packages\scipy\integrate\_odepack_py.py:248: ODE
intWarning: Excess work done on this call (perhaps wrong Dfun type). Run with full_o
utput = 1 to get quantitative information.
  warnings.warn(warning_msg, ODEintWarning)
d2= 481.95851622241594 k2= 0.08536428809294827 k3= 0.00013554990806318004 k7= 15.520
0194300376
```

In [8]:

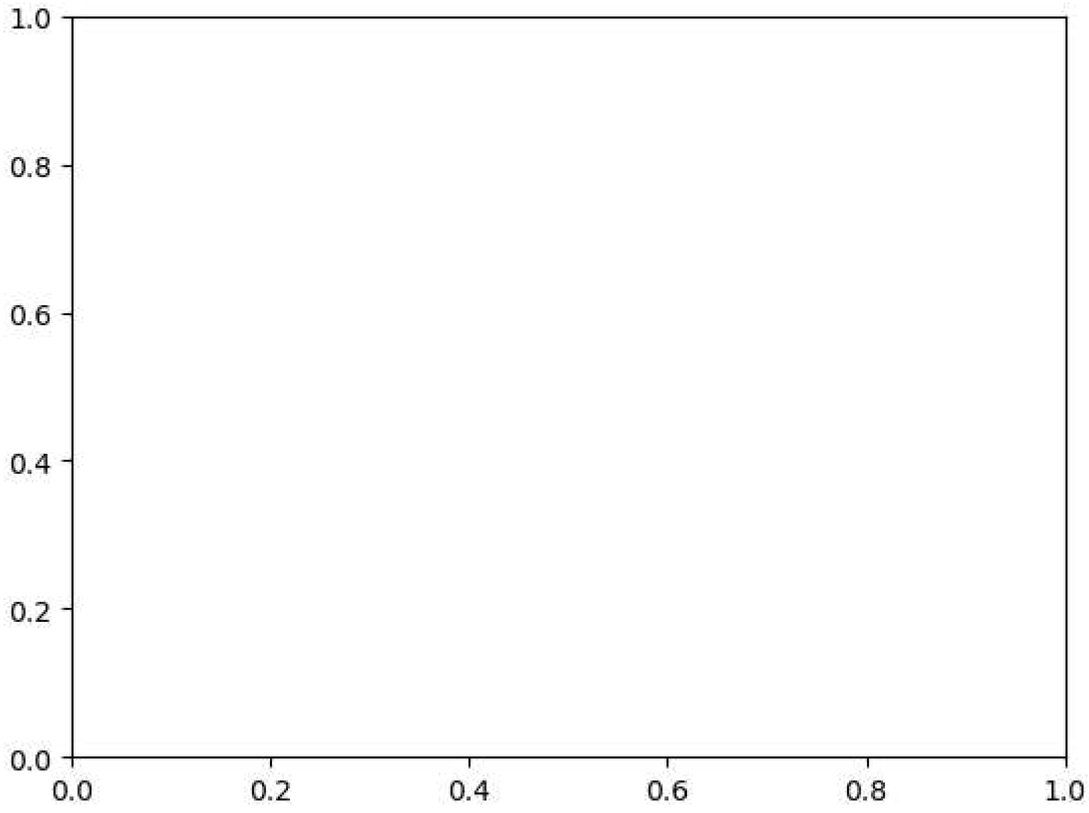
```
import sys
import numpy
#params = [1,1,1,1,1,1,1]
params = [1,1,1,1]
numpy.set_printoptions(threshold=10)
#model1 = np.asarray(func(t,Kd,n,d2,k2,k3))
model1 = np.asarray(func(t,d2,k2,k3,k7))
print(len(model1))
#print(model1)

#a,b,c,d,e,f,g = params
a,b,c,d= params

ydata = np.asarray(new)
print(len(ydata))

ssr = np.sum((ydata-model1)**2)
#ssr2 = np.sum((ydata-model2)**2)
sst = np.sum((ydata-np.mean(ydata))**2)
R2=1-ssr/sst
print('R2 is:', R2)
```

```
102
102
R2 is: -2.383722105946195
```

In [9]:

```
pp = PdfPages('multipage.pdf')
ydata = np.asarray(new)

#condition = [1,2,3,4,5,6,7,8,9]
for i in range(2):
    #model = tester(t,Kd,n,d2,k2,k3,i)
    model = tester(t,d2,k2,k3,k7,i)

    #print(model)
    #a,b,c,d,e,f,g= params
    a,b,c,d= params

    plt.plot(t,model[i],'.', label = 'model')
    #print(model)
    #t = np.linspace(0,34800, num=6961)
    #raw = rawdata[0:13920:240]
    plt.plot(t,rawdata[i],'.',label = 'data')
    #print(rawdata[i])
    plt.legend()
    pp.savefig()
    plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[9], line 13
      9 #print(model)
     10 #a,b,c,d,e,f,g= params
     11 a,b,c,d= params
---> 13 plt.plot(t,model[i],'.', label = 'model')
     14 #print(model)
     15 #t = np.linspace(0,34800, num=6961)
     16 #raw = rawdata[0:13920:240]
     17 plt.plot(t,rawdata[i],'.',label = 'data')

File ~\anaconda3\Lib\site-packages\matplotlib\pyplot.py:3578, in plot(scalex, scale
y, data, *args, **kwargs)
   3570 @_copy_docstring_and_deprecators(Axes.plot)
   3571 def plot(
   3572     *args: float | ArrayLike | str,
   (...)
   3576     **kwargs,
   3577 ) -> list[Line2D]:
-> 3578     return gca().plot(
   3579         *args,
   3580         scalex=scalex,
   3581         scaley=scaley,
   3582         **({"data": data} if data is not None else {}),
   3583         **kwargs,
   3584     )

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:1721, in Axes.plot(self,
scalex, scaley, data, *args, **kwargs)
   1478 """
   1479 Plot y versus x as lines and/or markers.
   1480
   (...)
   1718 (``'green'``) or hex strings (``'#008000'``).
   1719 """
   1720 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1721 lines = [*self._get_lines(self, *args, data=data, **kwargs)]
   1722 for line in lines:
   1723     self.add_line(line)

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:303, in _process_plot_va
r_args.__call__(self, axes, data, *args, **kwargs)
    301     this += args[0],
    302     args = args[1:]
--> 303 yield from self._plot_args(
    304     axes, this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datakey)

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:499, in _process_plot_va
r_args._plot_args(self, axes, tup, kwargs, return_kwargs, ambiguous_fmt_datakey)
    496     axes.yaxis.update_units(y)
    498 if x.shape[0] != y.shape[0]:
--> 499     raise ValueError(f"x and y must have same first dimension, but "
    500                      f"have shapes {x.shape} and {y.shape}")
    501 if x.ndim > 2 or y.ndim > 2:
    502     raise ValueError(f"x and y can be no greater than 2D, but have "
    503                         f"shapes {x.shape} and {y.shape}")

ValueError: x and y must have same first dimension, but have shapes (12241,) and
(1,)
```



```
In [ ]:
```

```
In [ ]:
```