

```
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 newdata
from endpoint_maker import lightdata
from endpoint_maker import rawdata
```

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

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

```
message: Optimization terminated successfully.
success: True
  fun: 9.640888474296567
   x: [ 7.458e+02  1.323e-02  3.872e-03]
  nit: 99
 nfev: 6000
```



```

In [5]: #def tester(t,d1,k1,Kd,n,d2,k2,k3,i):
def tester(t,d2,k2,k3,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 > 6960:
            tindex = 6960
        return lightdata[i][int(tindex)]

    #def odes(z,t,d1,k1,Kd,n,d2,k2,k3):
    def odes(z,t,d2,k2,k3):
        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
        dPbdt = k1*I(t)**n/(Kd**n+I(t)**n)*Pu - k2*Pb - d1*Pb + d2*Pa
        dPadt = k2*Pb - d2*Pa
        dmRNAAdt = 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,dmRNAAdt,dmCherry1dt,dmCherry2dt,dmCherry3dt]

    #solver = odeint(odes,inivalues,t,args = (d1,k1,Kd,n,d2,k2,k3),hmax=0.1)
    solver = odeint(odes,inivalues,t,args = (d2,k2,k3),hmax=0.1)

    mCherryout = solver[:,6]
    return mCherryout

```

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

```
[7.45790516e+02 1.32273427e-02 3.87245202e-03]
```

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

```
#popt, covt = curve_fit(func,t,newdata,initialp.x, maxfev=10000000, bounds=((0  
#popt, covt = curve_fit(func,t,newdata,initialp.x, maxfev=1000000)
```

```
#d1,k1,Kd,n,d2,k2,k3 = pop  
#print('d1=',d1,'k1=',k1,'Kd=',Kd,'n=',n,'d2=',d2,'k2=',k2,'k3=',k3)
```

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

```
d2,k2,k3 = pop  
print('d2=',d2,'k2=',k2,'k3=',k3)
```

```
d2= 745.874751833565 k2= 0.01322488419488149 k3= 0.0038724606209748227
```

In [8]: `import sys`  
`import numpy`  
`#params = [1,1,1,1,1,1,1]`  
`params = [1,1,1]`  
`numpy.set_printoptions(threshold=10)`  
`#model1 = np.asarray(func(t,d1,k1,Kd,n,d2,k2,k3))`  
`model1 = np.asarray(func(t,d2,k2,k3))`  
`print(len(model1))`  
`#print(model1)`

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

```
ydata = np.asarray(newdata)  
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  
#R2_2 = 1 - ssr2/sst
```

```
print('R2 is: ', R2)  
#print(R2_2)
```

```
13922
```

```
13922
```

```
R2 is: 0.9666968168199872
```

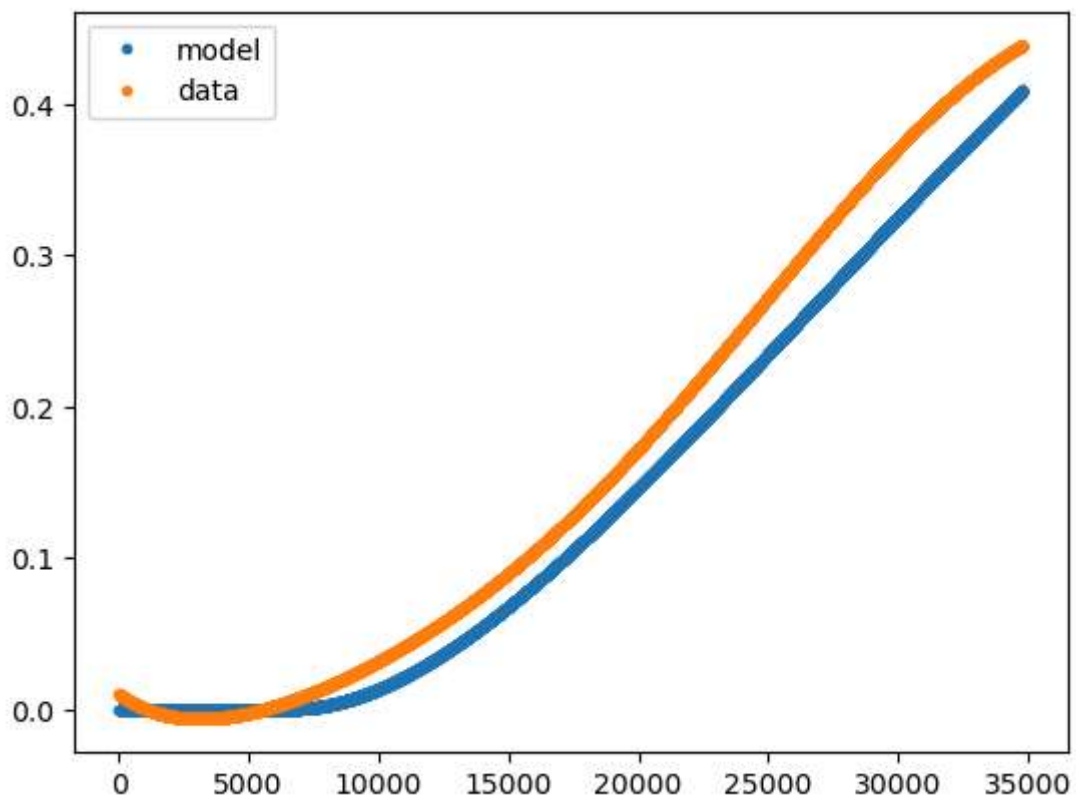
```
In [9]: pp = PdfPages('multipage.pdf')
ydata = np.asarray(newdata)

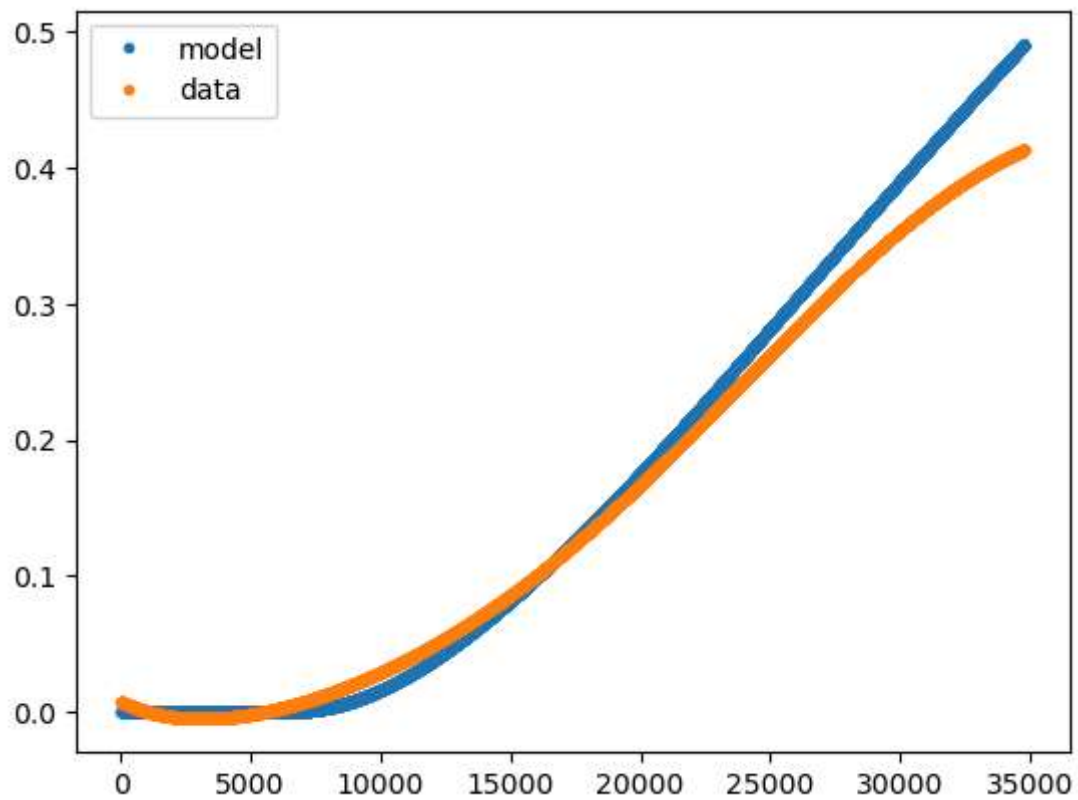
#condition = [1,2,3,4,5,6,7,8,9]
for i in range(2):
    #model = tester(t,d1,k1,Kd,n,d2,k2,k3,i)
    model = tester(t,d2,k2,k3,i)

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

    plt.plot(t,model,'.', label = 'model')
    #print(model)
    plt.plot(t,rawdata[i],'.',label = 'data')
    #print(rawdata[i])
    plt.legend()
    pp.savefig()
    plt.show()

pp.close
```





Out[9]: <bound method PdfPages.close of <matplotlib.backends.backend\_pdf.PdfPages object at 0x000002166EDA5DD0>>

In [ ]:

In [ ]: