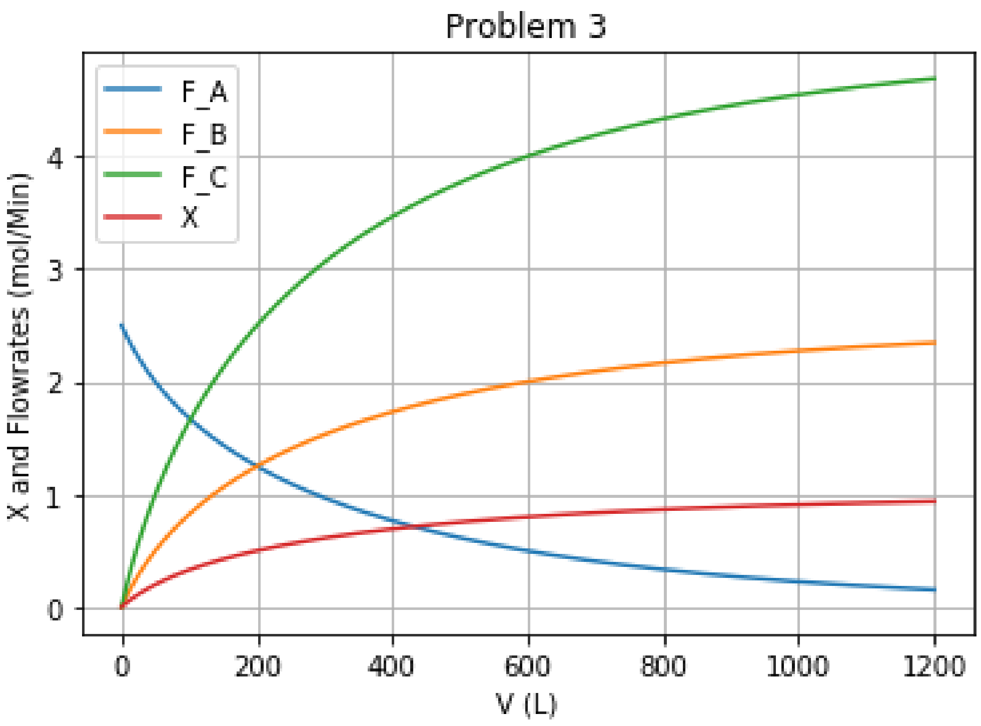
Reactor Design Python Code

April 1, 2019

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        from scipy.integrate import odeint

In [60]: #Initial Conditions
         Fao=2.5
         Fbo=0
         Fco=0
         inivalues=[Fao,Fbo,Fco]

         #Parameters
         k = 0.0442784555801
         T=400
         Pao =10
         R = 0.08206
         c_To=Pao/(0.08206*T)  #total concentration


         def pfr(z,V):
             [Fa,Fb,Fc]=z #assignment of dependent variables to convenient variable names

             #Stoichiometry

             Ft=(Fa+Fb+Fc) #Total molar flowrate
             Ca=c_To*(Fa/Ft)
             Cb=c_To*(Fb/Ft)
             Cc=c_To*(Fc/Ft)

            #Rate
             ra = -k*Ca
             rb = -ra
             rc = -2*ra

             #MB
             dFadV = ra
             dFbdV = rb
             dFcdV = rc
```

```
    return dFadV, dFbdV, dFcdV


Vspan = np.linspace(0,1200,1500) # independent variable array: 200 pts from V=0 to V=
solver = odeint(pfr,inivalues,Vspan) # solver output has format [X,y]


# Plot results
plt.plot(Vspan,solver[:,0], label='F_A')
plt.plot(Vspan,solver[:,1], label='F_B')
plt.plot(Vspan,solver[:,2], label='F_C')

plt.xlabel('V (L)')
plt.ylabel('X and Flowrates (mol/Min)')

plt.title('Problem 3')


X_target=solver[:,0]
X=[]
for i in range(1500):
    X.append((2.5-X_target[i])/2.5) #(intial-final)/initial
plt.plot(Vspan, X, label='X')
plt.grid(True)
plt.legend(loc = 'best')
plt.show()

Vol=np.interp(0.9,X,Vspan)
index=int(Vol) #integer form of the Volume at which 90% conversion happens

A=np.array(solver[:,0])
B=np.array(solver[:,1])
C=np.array(solver[:,2])
print (A[index])#Flowrate of A at 90% conv.
print (B[index])#flowrate of A at 90% conv.
print (C[index])#flowrate of A at 90% conv.

Ftotal=A[index]+B[index]+C[index]
SpaceTime= Vol/Ftotal

print("Volume needed for 90% conversion is {:.2f} L. SpaceTime needed for 90% convers
```

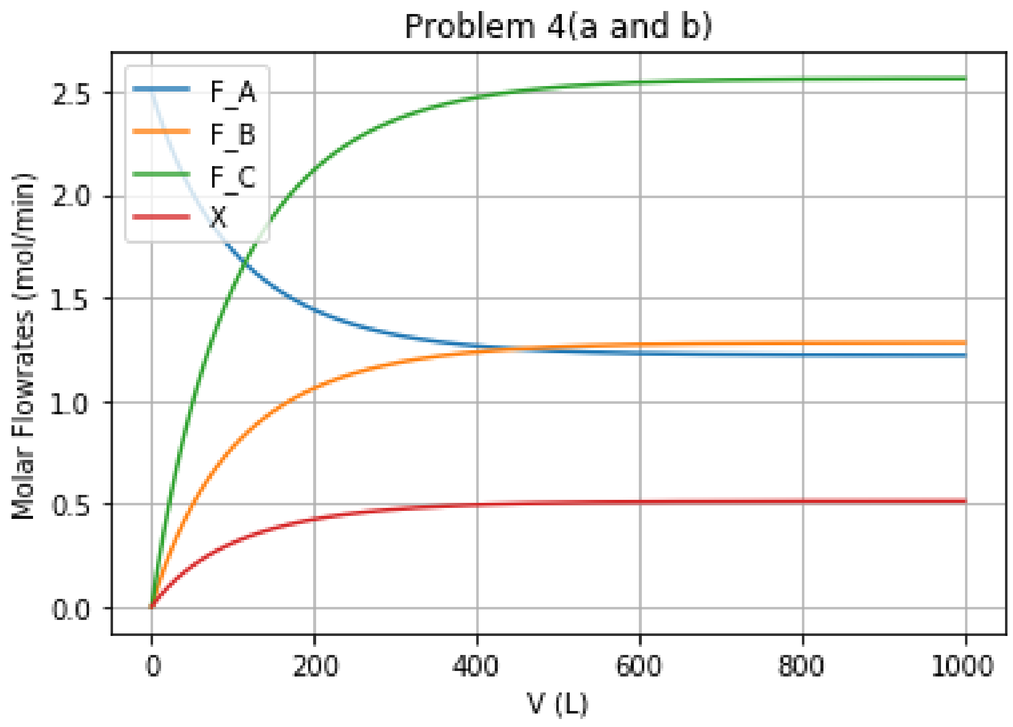


```
0.362065645038
2.13793435496
4.27586870992
Volume needed for 90% conversion is 946.61 L. SpaceTime needed for 90% conversion is 139.70 mi
```

```
In [77]: #4a

         #Initial conditions
         Fao=2.5 #[mol/min]
         Fbo=0
         Fco=0
         inivalues=[Fao,Fbo,Fco]
         #Parameters
         T=400
         P_ao=10
         R=0.08206
         kc=0.08*60 #MTC
         k = 0.0442784555801 #same as in Problem 3


         def memreact(z,V):
             c_To=P_ao/(R*T) #total concentration
             Kc=0.025
```

```
    [Fa,Fb,Fc]=z

    #Stoichiometry
    Ft=Fa+Fb+Fc
    ca=c_To*Fa/Ft
    cb=c_To*Fb/Ft
    cc=c_To*Fc/Ft

    #rate laws
    ra = -k*(ca-cb*(cc**2)/Kc)
    rb = -ra
    rc = -2*ra

    #MB
    dFadV = ra
    dFbdV = rb
    dFcdV = rc

    return dFadV, dFbdV, dFcdV

Vspan = np.linspace(0,1000,1500)
solver = odeint(memreact,inivalues,Vspan)
# Plot results
plt.plot(Vspan,solver[:,0], label='F_A')
plt.plot(Vspan,solver[:,1], label='F_B')
plt.plot(Vspan,solver[:,2], label='F_C')
plt.xlabel('V (L)')
plt.ylabel('Molar Flowrates (mol/min)')
plt.title('Problem 4(a and b)')


Xout=(Fao-solver[-1,0])/Fao #get's final value
print('Maximum Conversion (Xe) = {:.2f}'.format((Xout))
X_target=solver[:,0]
X=[]
for i in range(1500):
    X.append((2.5-X_target[i])/2.5)
plt.plot(Vspan, X, label='X')
plt.grid(True)
plt.legend(loc = 'best')
plt.show()

#4b

conv90=np.interp(Xout*0.9,X,Vspan)
print('The volume needed for 90% of the maximum conversion is = {:.1f} L.'.format(conv
```

```
Maximum Conversion (Xe) = 0.51
```

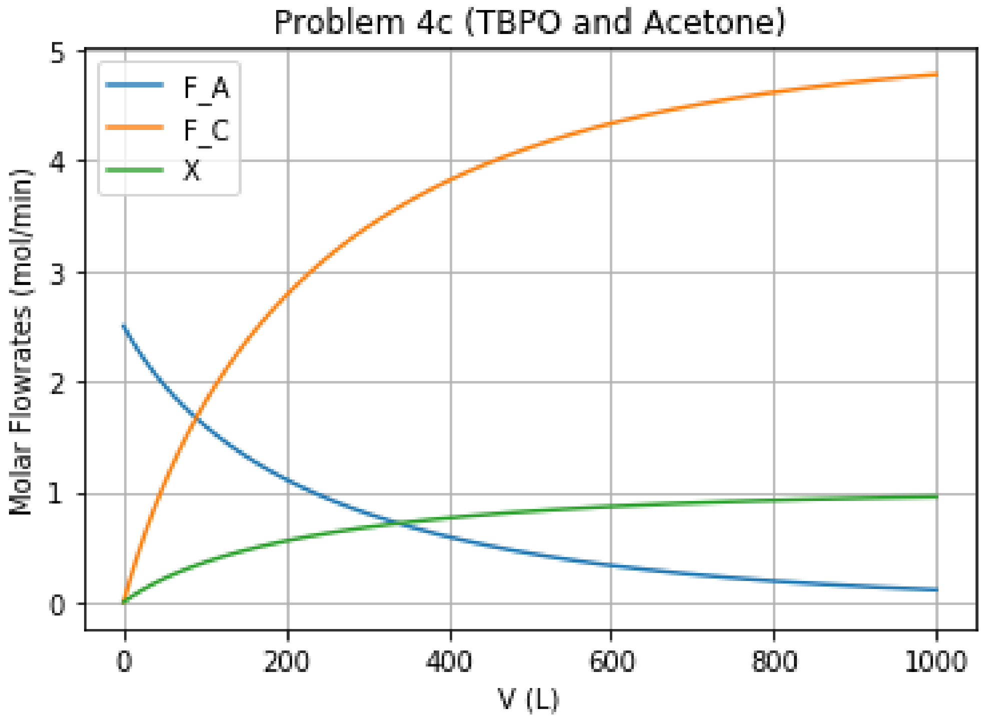


```
The volume needed for 90% of the maximum conversion is = 268.4 L.
```

```
In [78]: #4c

         #Initial conditions
         Fao=2.5 #[mol/min]
         Fbo=0
         Fco=0
         inivalues=[Fao,Fbo,Fco]
         #Parameters
         T=400
         P_ao=10
         R=0.08206
         kc=0.08*60 #MTC
         k = 0.0442784555801 #same as in Problem 3


         def memreact(z,V):
             c_To=P_ao/(R*T) #total concentration
             Kc=0.025
             [Fa,Fb,Fc]=z
```

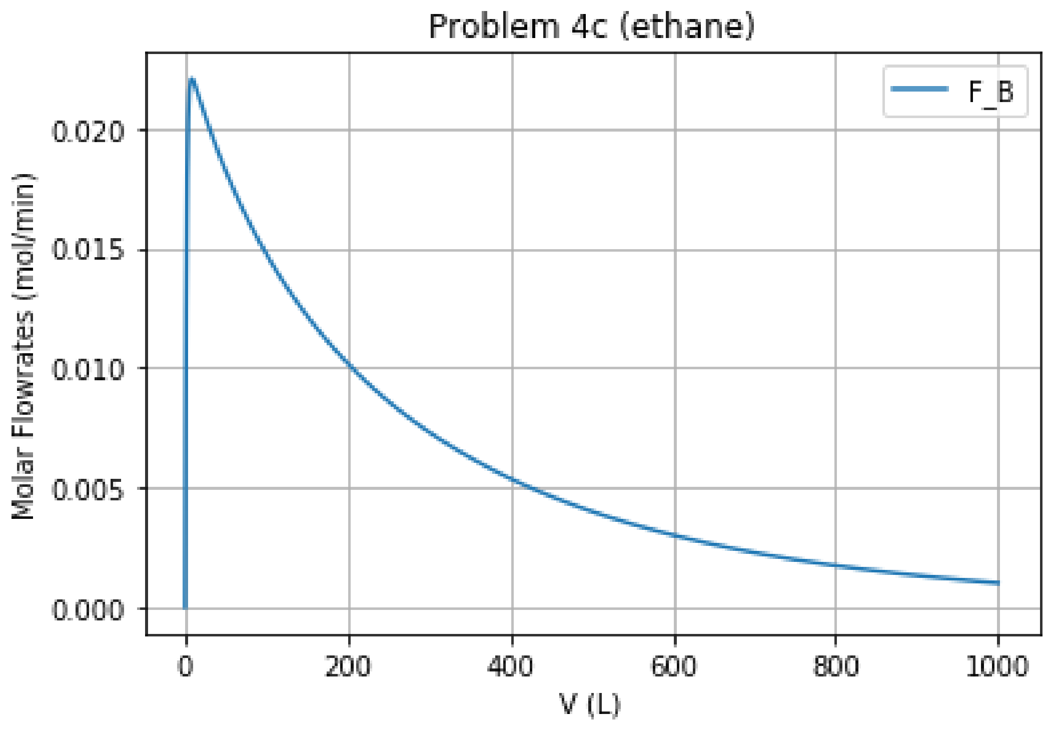
```
    #Stoichiometry
    Ft=Fa+Fb+Fc
    ca=c_To*Fa/Ft
    cb=c_To*Fb/Ft
    cc=c_To*Fc/Ft

    #rate laws
    ra = -k*(ca-cb*(cc**2)/Kc)
    rb = -ra
    rc = -2*ra
    #MB
    dFadV = ra
    dFbdV = rb - kc*cb
    dFcdV = rc

    return dFadV, dFbdV, dFcdV

# Setup ODE solver
Vspan = np.linspace(0,1000,1500) # independent variable array: 200 pts from V=0 to V=
solver = odeint(memreact,inivalues,Vspan) # solver output has format [X,y]
#print(solver)

# Plot results
plt.plot(Vspan,solver[:,0], label='F_A')
plt.plot(Vspan,solver[:,2], label='F_C')
plt.xlabel('V (L)')
plt.ylabel('Molar Flowrates (mol/min)')
plt.title('Problem 4c (TBPO and Acetone)')


Xout=(Fao-solver[-1,0])/Fao #(initial-final)/initial
print('At reactor outlet, X_A ={:.2f}'.format((Xout)))
X_target=solver[:,0]
X=[]
for i in range(1500):
    X.append((2.5-X_target[i])/2.5)
plt.plot(Vspan, X, label='X')
plt.grid(True)
plt.legend(loc = 'best')
plt.show()
conv80= np.interp(0.8,X,Vspan)
print('The volume needed for reaching 90% of the maximum conversion is',conv80,'L')
```

```
At reactor outlet, X_A =0.96
```



```
The volume needed for reaching 90% of the maximum conversion is 457.2671936581845 L
```

```
In [79]: plt.plot(Vspan,solver[:,1], label='F_B')
         plt.xlabel('V (L)')
         plt.ylabel('Molar Flowrates (mol/min)')
         plt.legend(loc = 'best')
         plt.title('Problem 4c (ethane)')
         plt.grid(True)
         plt.show()
```



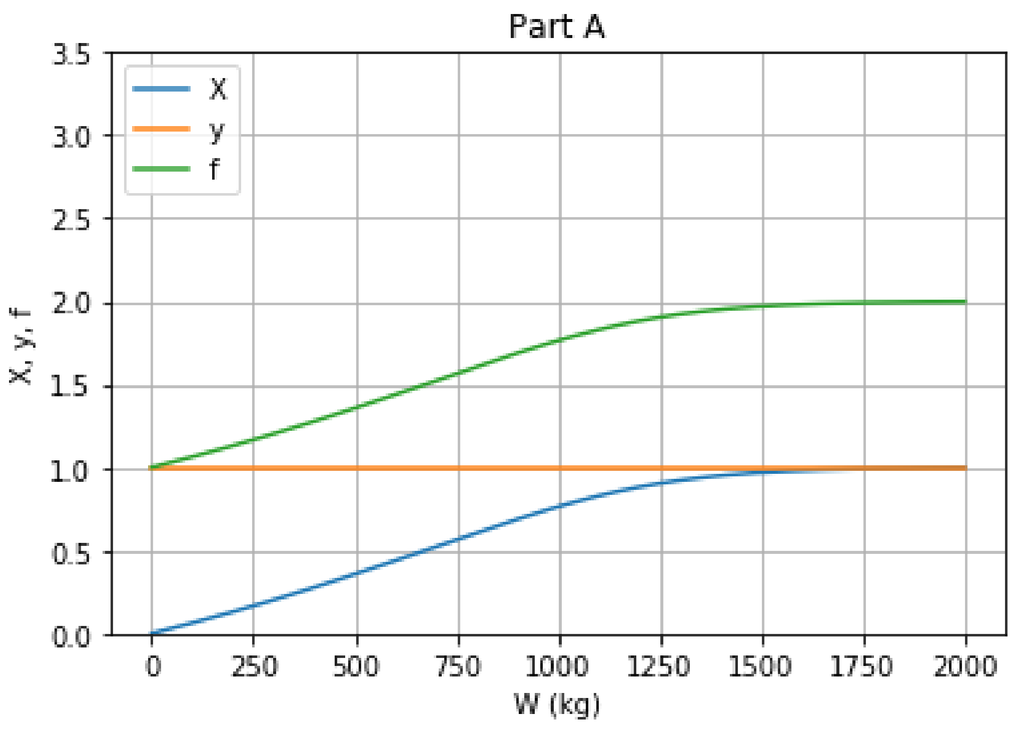
```
In [24]: print('Maximum Flow rate of TBPO = {:.1f} mol/min.'.format(np.max(solver[:,0])))
         print('Maximum Flow rate of Ethane = {:.4f} mol/min.'.format(np.max(solver[:,1])))
         print('Maximum Flow rate of Acetone = {:.1f} mol/min.'.format(np.max(solver[:,2])))
```

```
Maximum Flow rate of TBPO = 2.5 mol/min.
Maximum Flow rate of Ethane = 0.0221 mol/min.
Maximum Flow rate of Acetone = 4.8 mol/min.
```

The maximum flow rate of TBPO (A) happens at the very beginning which is to be expected, as it will decrease as volume increases. The opposite is true for Acetone, having it's maximum flow rate be at the very end of the 1000 L process, as it's Rate increase with volume. The oddity is with ethane, and that is because of the mass transfer coefficient. The permeability makes the ethane diffuse out of the reactor, so it's flowrate decreases once the mass of the ethane is big enough to have it's MTC play a large role, ie, after 0.0221 mol/min

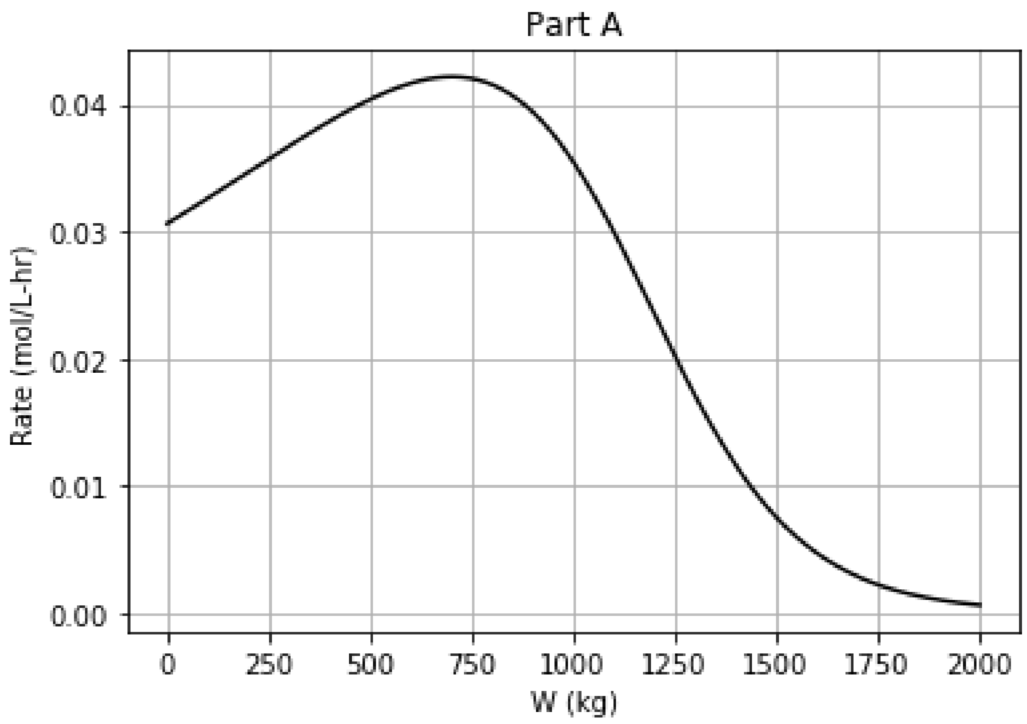
```
In [3]: #2a

        #Define all parameters
        T = 680
        Fao = 50
        Pao = 10
        R = 0.08206
        Cao = Pao/(R*T)
```

```python
KBu=0.32
k = 0.054
alpha = 0.000
Wspan = [0,2000]
inivalues=[0,1.0]

# Set up the differenial equations
def pbr(z,V):
    [X,y]=z

    ca = Cao*(1 - X)/(1 + X) #stoichiometry
    Pa=ca*R*T
    neg_ra = (k*Pa)/((1+KBu*Pa)**2) #rate law

    #ODEs:
    dXdW = neg_ra/Fao
    dydW = -(alpha*(1+X))/(2*y) #Ergun equation
    return dXdW, dydW

# Setup ODE solver
V = np.linspace(Wspan[0],Wspan[1],200)
solver = odeint(pbr,inivalues,V)
X=solver[:,0]
y=solver[:,1]
f=(1+X)/y #Calculate f for each element in X and y vectors

# Plot results
plt.plot(V,X, label='X')
plt.plot(V,y, label='y')
plt.plot(V,f, label='f')
plt.xlabel('W (kg)')
plt.ylabel('X, y, f')
plt.title('Part A')
plt.legend(loc = 'upper left')
plt.grid(True)
plt.ylim(0,3.5)
plt.show()

#From X vs. W, find W required for 80% conversion
V_requirement=np.interp(0.8,X,V)
print('Reactor reaches a conversion of X = 80% at W =',str(V_requirement),'kg')
```



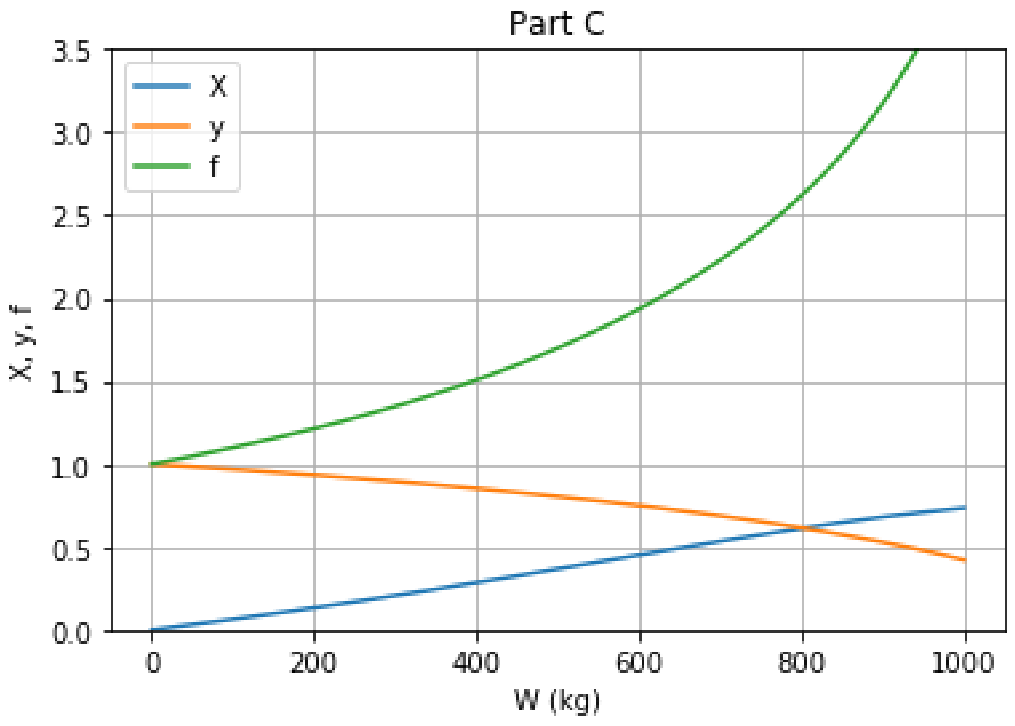
```
Reactor reaches a conversion of X = 80% at W = 1054.1518509611706 kg
```

In [4]:
```
ca = Cao*(1 - X)/(1 + X) #stoichiometry
Pa=ca*R*T
neg_ra = (k*Pa)/((1+KBu*Pa)**2) #rate law


plt.plot(V,neg_ra,'black')
plt.xlabel('W (kg)')
plt.title('Part A')
plt.ylabel('Rate (mol/L-hr)')
plt.grid(True)
plt.show()
```



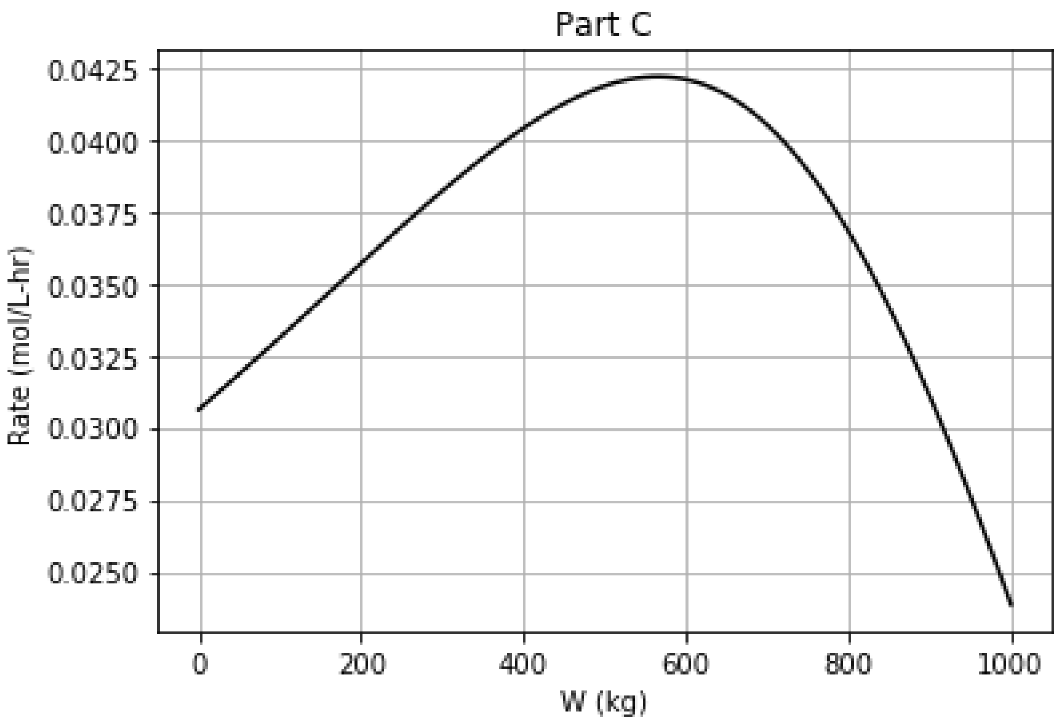
```
In [5]: #2c

        #Define all parameters
        T = 680
        Fao = 50
        Pao = 10
        R = 0.08206
        Cao = Pao/(R*T)
        KBu=0.32
        k = 0.054
        alpha = 0.0006
        Vspan = [0,1000]
        inivalues=[0,1.0]

        # Set up the differenial equations
        def pbr2(z,V):
            [X,y]=z

            ca = (Cao*(1 - X)/(1 + X))*y #stoichiometry
            Pa=ca*R*T
            neg_ra = (k*Pa)/((1+KBu*Pa)**2) #rate law

            #ODEs:
            dXdV = neg_ra/Fao  #mole balance
```

```
    dydV = -(alpha*(1+X))/(2*y) #Ergun equation
    return dXdV, dydV

# Setup ODE solver
V = np.linspace(Vspan[0],Vspan[1],200) # independent variable array, increment 1
solver = odeint(pbr2,inivalues,V) # solver has format [X,y]

X=solver[:,0]
y=solver[:,1]
f=(1+X)/y #Calculate f for each element in X and y vectors

# Plot results
plt.plot(V,X, label='X')
plt.plot(V,y, label='y')
plt.plot(V,f, label='f')
plt.xlabel('W (kg)')
plt.ylabel('X, y, f')
plt.legend(loc = 'upper left')
plt.grid(True)
plt.title('Part C')
plt.ylim(0,3.5)
plt.show()

#From X vs. W, find W required for 70% conversion
V_requirement=np.interp(0.7,X,V)
print('Reactor reaches a conversion of X = 70% at W =',str(V_requirement),'kg')
```



```
Reactor reaches a conversion of X = 70% at W = 932.2674246272811 kg
```

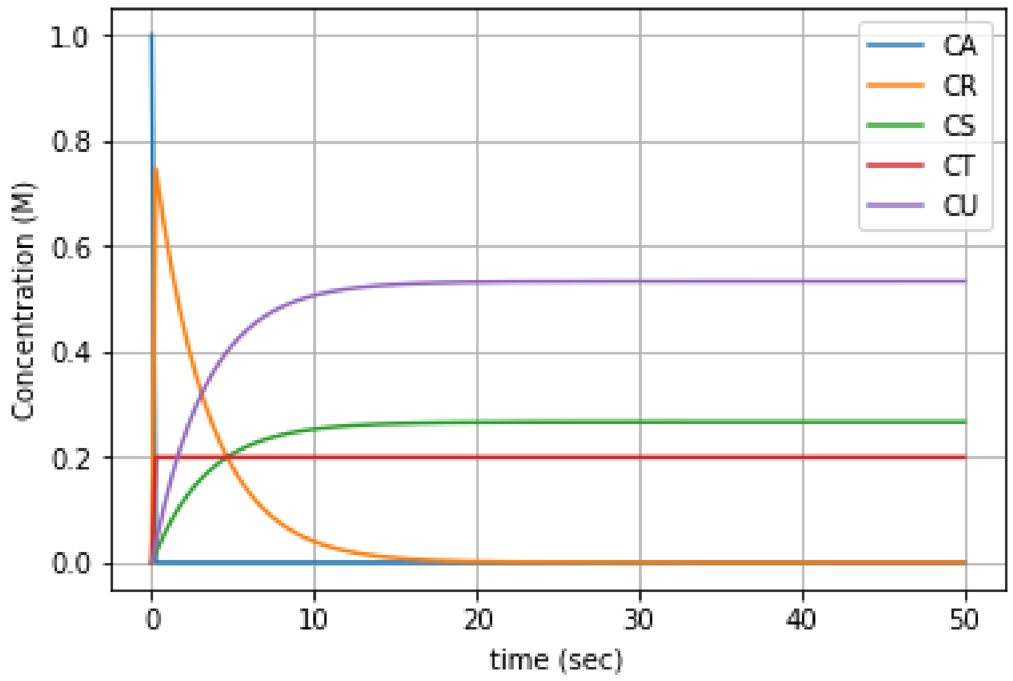
```
In [6]: ca = (Cao*(1 - X)/(1 + X))*y #stoichiometry
        Pa=ca*R*T
        neg_ra = (k*Pa)/((1+KBu*Pa)**2) #rate law


        plt.plot(V,neg_ra,'black')
        plt.xlabel('W (kg)')
        plt.ylabel('Rate (mol/L-hr)')
        plt.title('Part C')
        plt.grid(True)
        plt.show()
```



```
In [7]: #Define all parameters
        Cao=1
        Cro=0
        Cso=0
        Cto=0
        Cuo=0
```

```
k1=40
k2=10
k3=0.1
k4=.2



inivalues=[Cao,Cro,Cso,Cto,Cuo]


# Set up the differenial equations
def pfr(z,t):
    [Ca,Cr,Cs,Ct,Cu]=z #assignment of dependent variables to convenient variable names

    r1a=-k1*Ca
    r1b=-k2*Ca
    r1r=-k3*Cr
    r2r=-k4*Cr


    ra=r1a+r1b
    rr=-r1a+r2r+r1r
    rt=-r1b
    rs=-r1r
    ru=-r2r

    dCadt=ra
    dCrdt=rr
    dCtdt=rt
    dCsdt=rs
    dCudt=ru

    return dCadt, dCrdt, dCsdt,dCtdt, dCudt


T = np.linspace(0,50,200) # independent variable array, increment 1
solver = odeint(pfr,inivalues,T) # solver has format [X,y]

CA=solver[:,0]
CR=solver[:,1]
CS=solver[:,2]
CT=solver[:,3]
CU=solver[:,4]


# Plot results
```

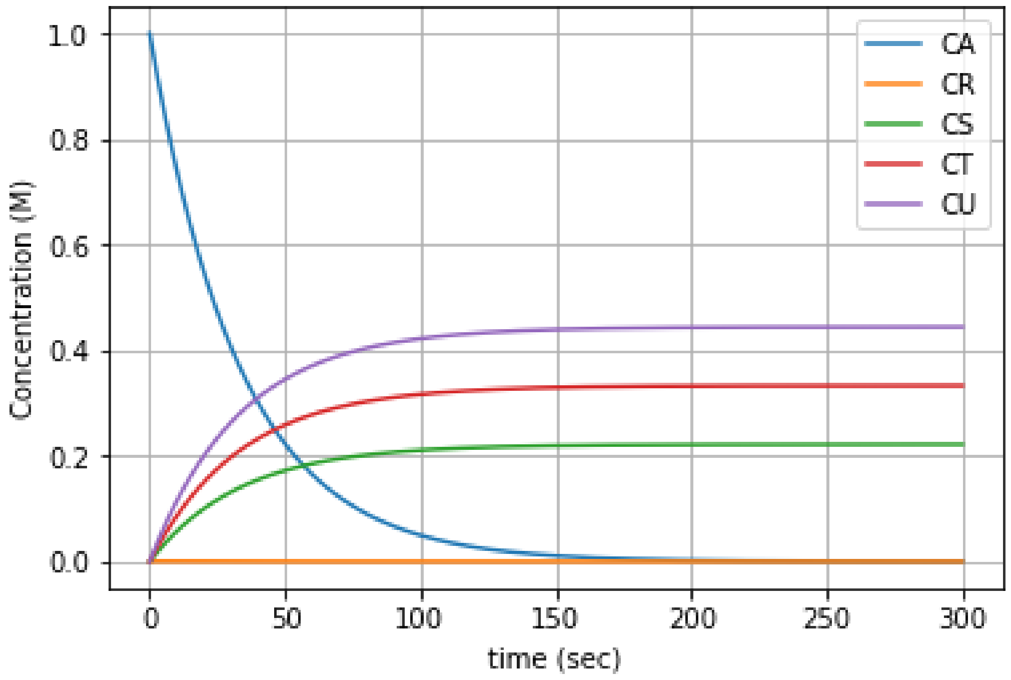
```
plt.plot(T, CA, label='CA')
plt.plot(T,CR, label='CR')
plt.plot(T,CS, label='CS')
plt.plot(T,CT, label='CT')
plt.plot(T,CU, label='CU')



plt.xlabel('time (sec)')
plt.ylabel('Concentration (M)')
plt.legend(loc = 'best')
plt.grid(True)
plt.show()

time=np.interp(0.2,CS,T)
print(time)
A_Final= np.interp(time,T,CA)
R_Final= np.interp(time,T,CR)
S_Final= np.interp(time,T,CS)
T_Final= np.interp(time,T,CT)
U_Final= np.interp(time,T,CU)

print("Final Concentration of A is {:.3f} M".format(A_Final))
print("Final Concentration of R is {:.3f} M".format(R_Final))
print("Final Concentration of S is {:.3f} M".format(S_Final))
print("Final Concentration of T is {:.3f} M".format(T_Final))
print("Final Concentration of U is {:.3f} M".format(U_Final))
#arg=np.argmax(CB)
#time=T[arg]
#vo=V/time

#print(MaxB)

#From X vs. W, find W required for 50% conversion
#V_requirement=np.interp(0.7,X,V)
#print('Reactor reaches a conversion of X = 70% at W =',str(V_requirement),'kg')
```



```
4.643402772774782
Final Concentration of A is -0.000 M
Final Concentration of R is 0.200 M
Final Concentration of S is 0.200 M
Final Concentration of T is 0.200 M
Final Concentration of U is 0.400 M
```

```
In [8]: #Define all parameters
        Cao=1
        Cro=0
        Cso=0
        Cto=0
        Cuo=0


        k1=0.02
        k2=.01
        k3=10
        k4=20


        inivalues=[Cao,Cro,Cso,Cto,Cuo]
```

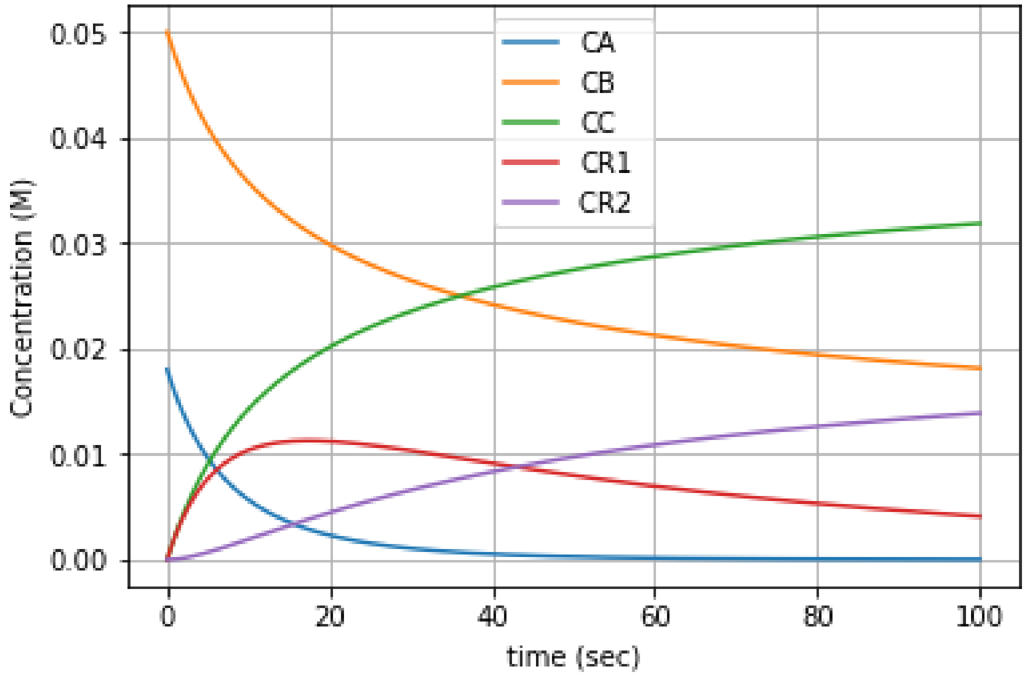
```
# Set up the differenial equations
def pfr(z,t):
    [Ca,Cr,Cs,Ct,Cu]=z #assignment of dependent variables to convenient variable names

    r1a=-k1*Ca
    r1b=-k2*Ca
    r1r=-k3*Cr
    r2r=-k4*Cr


    ra=r1a+r1b
    rr=-r1a+r2r+r1r
    rt=-r1b
    rs=-r1r
    ru=-r2r

    dCadt=ra
    dCrdt=rr
    dCtdt=rt
    dCsdt=rs
    dCudt=ru

    return dCadt, dCrdt, dCsdt,dCtdt, dCudt


T = np.linspace(0,300,200) # independent variable array, increment 1
solver = odeint(pfr,inivalues,T) # solver has format [X,y]

CA=solver[:,0]
CR=solver[:,1]
CS=solver[:,2]
CT=solver[:,3]
CU=solver[:,4]


# Plot results
plt.plot(T, CA, label='CA')
plt.plot(T,CR, label='CR')
plt.plot(T,CS, label='CS')
plt.plot(T,CT, label='CT')
plt.plot(T,CU, label='CU')



plt.xlabel('time (sec)')
plt.ylabel('Concentration (M)')
plt.legend(loc = 'best')
plt.grid(True)
plt.show()

time=np.interp(0.2,CS,T)
print(time)
A_Final= np.interp(time,T,CA)
R_Final= np.interp(time,T,CR)
S_Final= np.interp(time,T,CS)
T_Final= np.interp(time,T,CT)
U_Final= np.interp(time,T,CU)

print("Final Concentration of A is {:.3f} M".format(A_Final))
print("Final Concentration of R is {:.3f} M".format(R_Final))
print("Final Concentration of S is {:.3f} M".format(S_Final))
print("Final Concentration of T is {:.3f} M".format(T_Final))
print("Final Concentration of U is {:.3f} M".format(U_Final))


#arg=np.argmax(CB)
#time=T[arg]
#vo=V/time

#print(MaxB)

#From X vs. W, find W required for 50% conversion
#V_requirement=np.interp(0.7,X,V)
#print('Reactor reaches a conversion of X = 70% at W =',str(V_requirement),'kg')
```



```
76.78827015068796
Final Concentration of A is 0.100 M
Final Concentration of R is 0.000 M
Final Concentration of S is 0.200 M
Final Concentration of T is 0.300 M
Final Concentration of U is 0.400 M
```

```
In [9]: #Define all parameters
        Cao=0.018
        Cbo=0.05
        Cco=0
        Cr1o=0
        Cr2o=0

        R=1.98588
        T=80+273
        k1=(4.87*10**6)*np.exp(-10080/(R*T))
        k2=(3.49*10**3)*np.exp(-5965/(R*T))

        L=6*12 #in
        A=((1.25**2)/4)*np.pi
        V=L*A #in^3
        V=V/61.0237#L



        inivalues=[Cao,Cbo,Cco,Cr1o,Cr2o]


        # Set up the differenial equations
        def pfr(z,t):
            [Ca,Cb,Cc,Cr1,Cr2]=z #assignment of dependent variables to convenient variable nam

            r1a=-k1*Ca*Cb
            r2b=-k2*Cr1*Cb

            ra=r1a
            rb=r1a+r2b
            rc=-r1a-r2b
            rR1=-r1a+r2b
            rR2=-r2b

            dCadt=ra
            dCbdt=rb
```

```
    dCcdt=rc
    dcR1dt=rR1
    dcR2dt=rR2

    return dCadt, dCbdt, dCcdt, dcR1dt, dcR2dt

# Setup ODE solver
T = np.linspace(0,100,200) # independent variable array, increment 1
solver = odeint(pfr,inivalues,T) # solver has format [X,y]

CA=solver[:,0]
CB=solver[:,1]
CC=solver[:,2]
CR1=solver[:,3]
CR2=solver[:,4]

# Plot results
plt.plot(T, CA, label='CA')
plt.plot(T,CB, label='CB')
plt.plot(T,CC, label='CC')
plt.plot(T,CR1, label='CR1')
plt.plot(T,CR2, label='CR2 ')

plt.xlabel('time (sec)')
plt.ylabel('Concentration (M)')
plt.legend(loc = 'best')
plt.grid(True)
plt.show()

MaxR1= np.max(CR1)
arg=np.argmax(CR1)
time=T[arg]
vo=V/time

print("A) The Flowrate that maximizes R1 is {:.4f} L/s".format(vo))
print("B) The Maximum effluent Concentration of R1 is {:.4f} mol/L".format(MaxR1))

#From X vs. W, find W required for 50% conversion
#V_requirement=np.interp(0.7,X,V)
#print('Reactor reaches a conversion of X = 70% at W =',str(V_requirement),'kg')
```

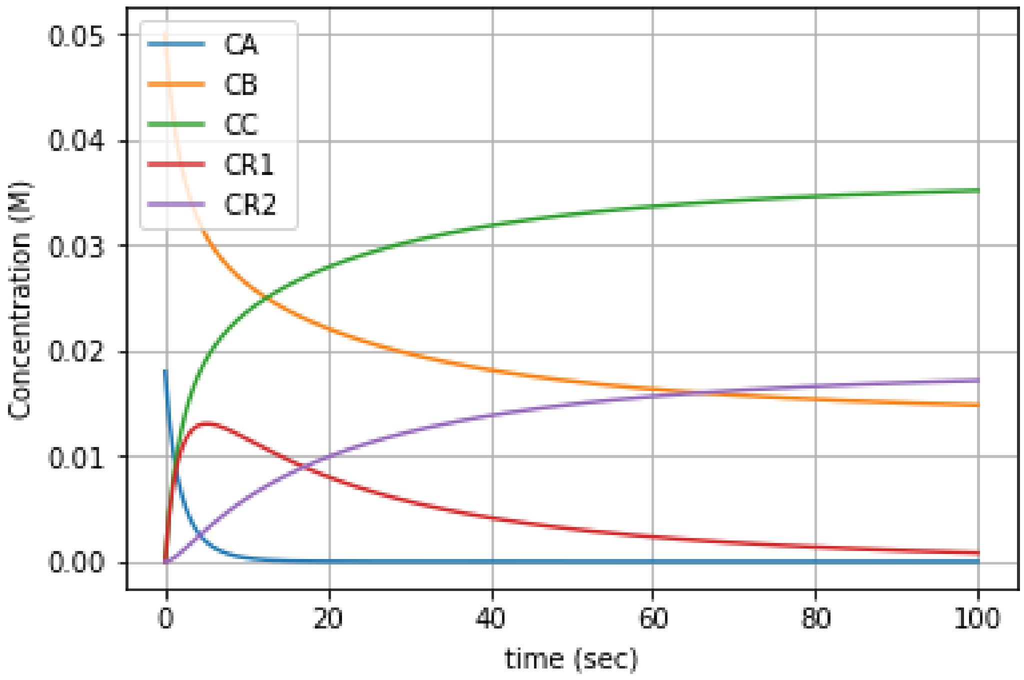


```
A) The Flowrate that maximizes R1 is 0.0823 L/s
B) The Maximum effluent Concentration of R1 is 0.0113 mol/L
```

```
In [10]: #Define all parameters
         Cao=0.018
         Cbo=0.05
         Cco=0
         Cr1o=0
         Cr2o=0

         R=1.98588
         T=120+273
         k1=(4.87*10**6)*np.exp(-10080/(R*T))
         k2=(3.49*10**3)*np.exp(-5965/(R*T))

         L=6*12 #in
         A=((1.25**2)/4)*np.pi
         V=L*A #in^3
         V=V/61.0237#L



         inivalues=[Cao,Cbo,Cco,Cr1o,Cr2o]
```

```
# Set up the differenial equations
def pfr(z,t):
    [Ca,Cb,Cc,Cr1,Cr2]=z #assignment of dependent variables to convenient variable na

    r1a=-k1*Ca*Cb
    r2b=-k2*Cr1*Cb

    ra=r1a
    rb=r1a+r2b
    rc=-r1a-r2b
    rR1=-r1a+r2b
    rR2=-r2b

    dCadt=ra
    dCbdt=rb
    dCcdt=rc
    dcR1dt=rR1
    dcR2dt=rR2

    return dCadt, dCbdt, dCcdt, dcR1dt, dcR2dt

# Setup ODE solver
T = np.linspace(0,100,200) # independent variable array, increment 1
solver = odeint(pfr,inivalues,T) # solver has format [X,y]

CA=solver[:,0]
CB=solver[:,1]
CC=solver[:,2]
CR1=solver[:,3]
CR2=solver[:,4]

# Plot results
plt.plot(T, CA, label='CA')
plt.plot(T,CB, label='CB')
plt.plot(T,CC, label='CC')
plt.plot(T,CR1, label='CR1')
plt.plot(T,CR2, label='CR2 ')

plt.xlabel('time (sec)')
plt.ylabel('Concentration (M)')
plt.legend(loc = 'best')
plt.grid(True)
plt.show()

MaxR1= np.max(CR1)
arg=np.argmax(CR1)
time=T[arg]
```

```
vo=V/time

print("At 120 degrees Celsius,the Maximum effluent Concentration of R1 is {:.4f} M and
print("So increasing temperature will increase yield, but you will also need to increa
```



```
At 120 degrees Celsius,the Maximum effluent Concentration of R1 is 0.0131 M and the Flowrate th
So increasing temperature will increase yield, but you will also need to increase the flowrate
```

```
In [ ]:
```