In [2]:
```
import numpy as np
```

In [3]:
```
#1.b System Reliability-Expected time until system fails
def System_reliability():
    t=0
    shape=2
    scale=1
    C=5
    alpha=0.5
    Acc_Shock=0
    Shock=np.array([])
    T=np.array([])
    while(Acc_Shock<=C):
        t=t-np.log(np.random.uniform())
        T=np.append(T,[t])
        x=np.random.gamma(shape,scale)
        Shock=np.append(Shock,[x])
        Acc_Shock=sum(Shock*np.exp(alpha*(T-t)))

    return(t)

S_T=np.array([])
for i in range(0,1000):
    S=0
    for j in range(0,1000):
        S=S+System_reliability()
    S_T=np.append(S_T,[S/1000])
```

In [4]:
```
np.mean(S_T)
```

Out[4]:
```
5.070479582271947
```

In [5]:
```
np.std(S_T)
```

Out[5]:
```
0.14054515153092517
```

In [17]:
```
#2.
def limorder_model():
    t=0
    T=1000
    V_t=0
    W_t=0
    Pa=0
    Pb=0
    CT=0
    LT=0
    P_Vec=np.zeros((101,), dtype=np.int)
    P_Vec[45:53]=[0,-1,-2,-3,0,3,2,1]
    A=0
    B=0
    while(t<T):
        Pa_t=min(np.where(P_Vec>0)[0])

        Pb_t=max(np.where(P_Vec<0)[0])

        lam_b=1/np.array(range(1,Pa_t+1))
        lam_s=1/np.array(range(1,101-Pb_t))
        Cap_Lam_s=sum(lam_s)
        Cap_Lam_b=sum(lam_b)
        M=2
        J=0
        Cap_theta_s=0.1*sum(P_Vec[Pa_t:])
        Cap_theta_b=-0.1*sum(P_Vec[:Pb_t+1])
        S=Cap_Lam_s+Cap_Lam_b+Cap_theta_b+Cap_theta_s+M
        U=np.random.uniform()
        U2=np.random.uniform()
        if(U<=Cap_Lam_s/S):
            J=1
            Pa=Pa_t+Pa
            X1=np.cumsum(lam_s)
            t1=[ n for n,i in enumerate(X1) if i>U2*Cap_Lam_s ][0]
            P_Vec[Pb_t+t1+1]=P_Vec[Pb_t+t1+1]+1
            A=A+1
        elif(U<=(Cap_Lam_b+Cap_Lam_s)/S):
            J=2
            Pb=Pb_t+Pb
            X2=np.cumsum(lam_b)
            t2=[ n for n,i in enumerate(X2) if i>U2*Cap_Lam_b ][0]
            P_Vec[Pa_t-t2-1]=P_Vec[Pa_t-t2-1]-1
            B=B+1
        elif(U<=(2+Cap_Lam_b+Cap_Lam_s)/S):
            J=3
            if U2>0.5:
                P_Vec[Pa_t]=P_Vec[Pa_t]-1
                W_t=W_t+1*(Pa_t+1)
            else:
                P_Vec[Pb_t]=P_Vec[Pb_t]+1
                W_t=W_t+1*(Pb_t+1)
            V_t=V_t+1

        elif(U<(Cap_theta_s+2+Cap_Lam_b+Cap_Lam_s)/S):
            J=4
```

```
            X3=np.where(P_Vec>0,0.1*P_Vec,0)
            X3=np.cumsum(X3)
            t3=[ n for n,i in enumerate(X3) if i>U2*Cap_theta_s ][0]
            P_Vec[t3]=P_Vec[t3]-1
            CT=CT+1
            LT=LT+t3+1
        else:
            J=5
            X4=np.where(P_Vec<0,-0.1*P_Vec,0)
            X4=np.cumsum(X4)
            t4=[ n for n,i in enumerate(X4) if i>U2*Cap_theta_b ][0]
            P_Vec[t4]=P_Vec[t4]+1
            CT=CT+1
            LT=LT+t4+1
        t=t-np.log(np.random.uniform())/S

    return(V_t,W_t,W_t/V_t,CT,V_t/T,CT/T,LT,Pa_t,Pb_t)
```

In [18]:
```
limitorder_model()
```

Out[18]:
```
(2055, 111973, 54.48807785888078, 6975, 6.975, 375134, 2.055)
```

In [ ]:
```

```