代码说明

第一题

模拟产生暗电流时间序列

使用$exprnd$命令产生指数分布的随机数，以此作为暗电流事件的时间间隔。加上总时长判断和简单的加法运算$T_i = \sum_{k=1}^{i} Interval_k$则得到时间序列。代码示例：

```
while min(max(T1),max(T2))<=10*3600
    T1(i)=T1(i-1)+exprnd(1/lambda1);
    T2(i)=T2(i-1)+exprnd(1/lambda2);
    i=i+1;
end
```

二重偶然符合

基本思想是：我拥有一个长度为N_1的时间序列$T_1 = [t_{11}, t_{12}, \dots t_{1N_1}]$和长度为$N_2$的时间序列$T_2 = [t_{21}, t_{22}, \dots t_{2N_2}]$。最为保险的办法是求出其任二之差$\Delta_{ij} = (t_{1i} - t_{2j})$。矩阵化编程可以这样考虑：

$$A = [T_1^T, T_1^T \dots, T_1^T] = \begin{bmatrix} t_{11} & t_{11} \dots\dots & t_{11} \\ t_{12} & t_{12} \dots\dots & t_{12} \\ \vdots & \ddots & \\ t_{1N_1} & & t_{1N_1} \end{bmatrix} = T_1^T \times [1, 1 \dots\dots 1]_{1\times N_2}$$

$$B = \begin{bmatrix} T_2 \\ T_2 \\ \vdots \\ T_2 \end{bmatrix} = \begin{bmatrix} t_{21} & t_{22} \dots\dots & t_{2N_2} \\ t_{21} & t_{22} \dots\dots & t_{2N_2} \\ \vdots & \ddots & \\ t_{21} & & t_{2N_2} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{N_1\times 1} \times T_2$$

$$\Delta = A - B, \Delta_{ij} = t_{1i} - t_{2j}$$

因此我们就仅需关心Delta矩阵的性质就可以了。

时间标志的确定

因为我们约定为以一次二重符合中较早的暗电流为标记，所以我们还需关心差矩阵的正负问题。让我以一个sample为例：$T_1 = [1.03, 1.05, 1.07], T_2 = [1.02, 1.04, 1.08], \tau = 0.01$

	1.02	**1.04**	**1.08**
1.03	**0.01**	**-0.01**	-0.05
1.05	0.03	**0.01**	-0.03
1.07	0.05	0.03	**-0.01**

因此发生二重符合的为表中加粗标识出的格点。我们发现：当差值为正时，我们应当选取T_2项作为标记，即Delta矩阵的列序号，反之为T_1项，即Delta矩阵的行序号。因此我们使用如下的采样程序：

```
[row,col]=find(abs(delta)<=tau);
m=find(abs(delta)<=tau);
row1=row(find(delta(m)<0));
col1=col(find(delta(m)>=0));
result=[sample1(row1),sample2(col1)]
```

输出为：

```
>> samplecaiyang

result =

    1.0300    1.0700    1.0200    1.0400
```

输出了正确的结果（当然，是乱序）。

算法优化

很遗憾，直接对10h全序列操作是不可能的，因为其需要的矩阵太大（96698.5GB），因此在反复优化后，我最终采用循环节混合分块矩阵进行操作。

$$\Delta = \begin{bmatrix} t_{11}-t_{21} & t_{11}-t_{22} & & & \\ t_{12}-t_{21} & t_{12}-t_{22} & & & \\ t_{13}-t_{21} & t_{13}-t_{22} & & & \\ & & \ddots & & \\ & & & t_{1j}-t_{2j} & t_{1j+1}-t_{2j} \\ & & & t_{1j}-t_{2j+1} & t_{1j+1}-t_{2j+1} \end{bmatrix}$$

对对角上的元素设为分块矩阵，相当于将10h分为多个小块，在操作中即每3min的数据进行操作。

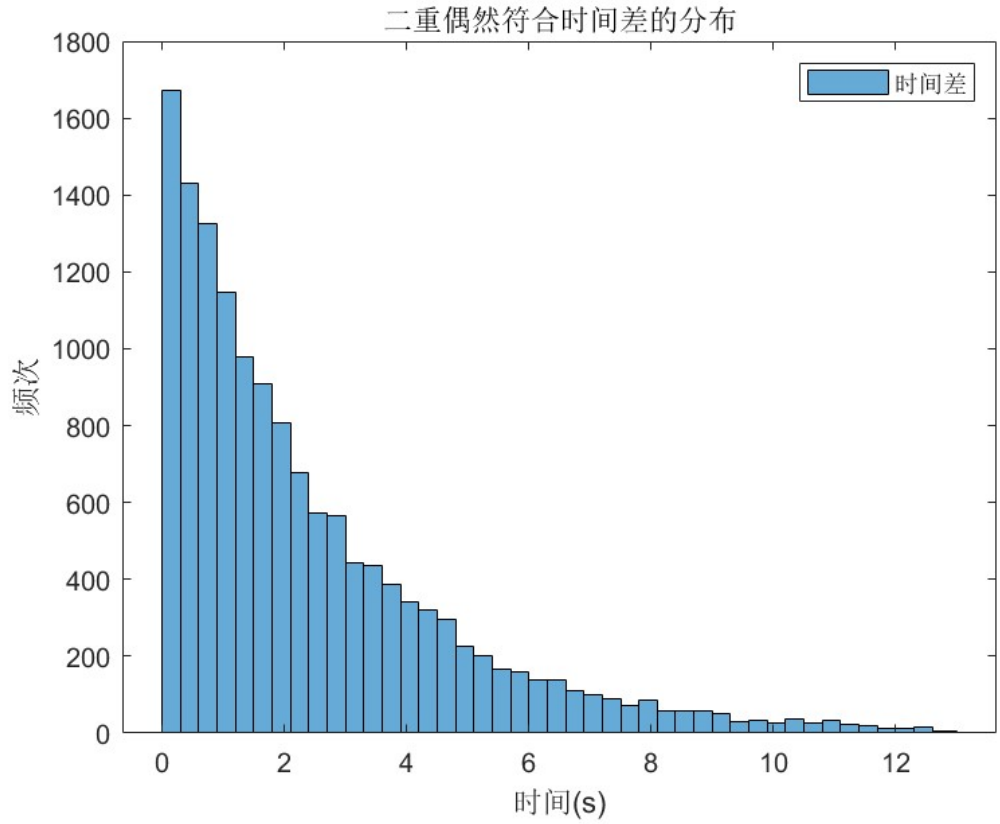
```
N=9000;
%%
for i=1:N
    Tadj1=T1adj(find(T1adj<=i*36000/N & T1adj>(i-1)*36000/N));
    Tadj2=T2(find(T2<=i*36000/N & T2>(i-1)*36000/N));
    N1=length(Tadj1);
    N2=length(Tadj2);
    B=Tadj1'*ones(1,N2);
    C=ones(N1,1)*Tadj2;
     delta=B-C;
    [row,col]=find(abs(delta)<=tau);
     m=find(abs(delta)<=tau);
    row1=row(find(delta(m)<0));
    col1=col(find(delta(m)>=0));
    result=[result,Tadj1(row1),Tadj2(col1)];
end
```

选取合适的时间分块可以有效提高效率。

事例率

共测量到14398个二重偶然符合事件，模拟事例率为$\lambda' = 0.3999444444$。而理论预测为：

$$\lambda = \lambda_1(1 - e^{-\lambda_2\tau}) + \lambda_2(1 - e^{-\lambda_1\tau}) \approx 0.3997001666$$
$$u_r \approx 0.61\%$$



坦诚地讲，这个分布并不令我满意。尤其需要注意到，在间隔时间较大的时候，分布甚至出现了紊乱。这与次数不够导致的误差不同，是上述代码优化环节的系统误差。

第二题

乙比甲先到

乙比甲先到⇒到达慢车但甲出发后10min内快车到达

$$\lambda_{fast} = \frac{1}{12}min^{-1}, \lambda = \frac{1}{3}min^{-}1, \lambda_{slow} = \frac{1}{4}min^{-1}$$

记录到达的车为快车为事件A，慢车但甲出发后10min内快车到达为事件B。

$$P(B) = \frac{3}{4}P(t \leq 10min) = \frac{3}{4}F(10) = \frac{3}{4}(1 - e^{-\lambda_{fast}*(10min)}) = \frac{3}{4}(1 - e^{-\frac{5}{6}}) \approx 42.405\%$$

甲乙同时到

则必须是到达车辆为快车。

$$\therefore P(A) = 25\%$$

等车时间

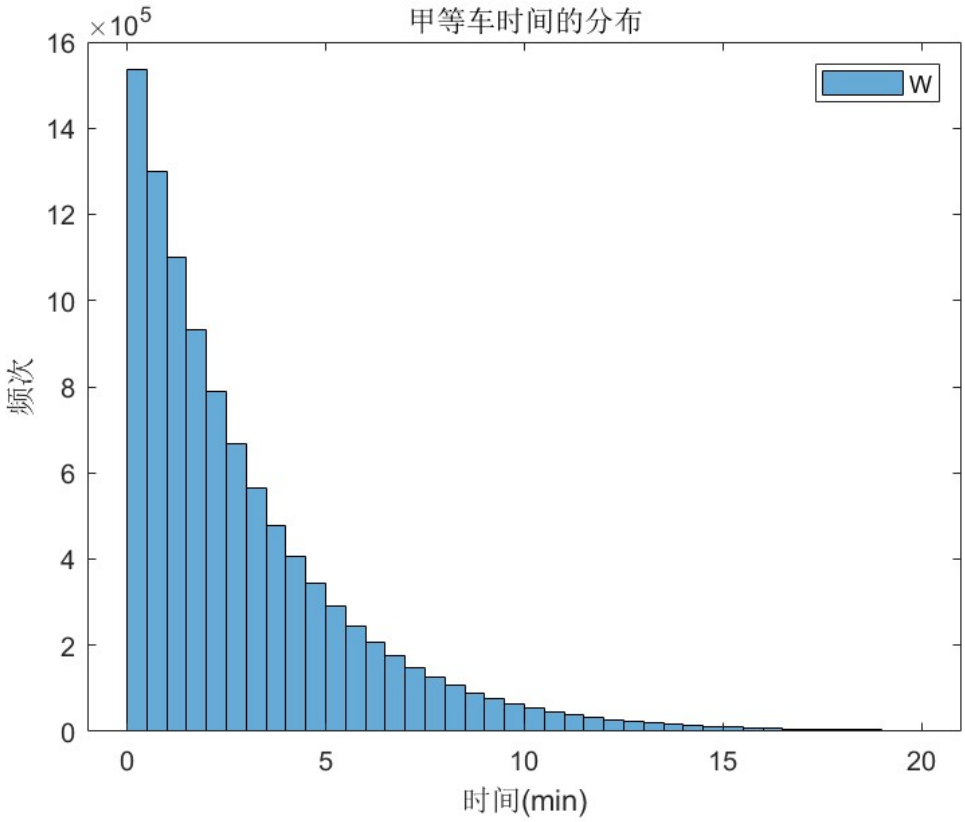
$$W_{甲} \sim E(\lambda), W_{乙} \sim E(\lambda_{fast}),\ E(E(\lambda)) = \lambda$$
$$X = W_{甲} + E(16min, 26min) = 26.5min, Y = W_{乙} + 16min = 28min$$

选择问题

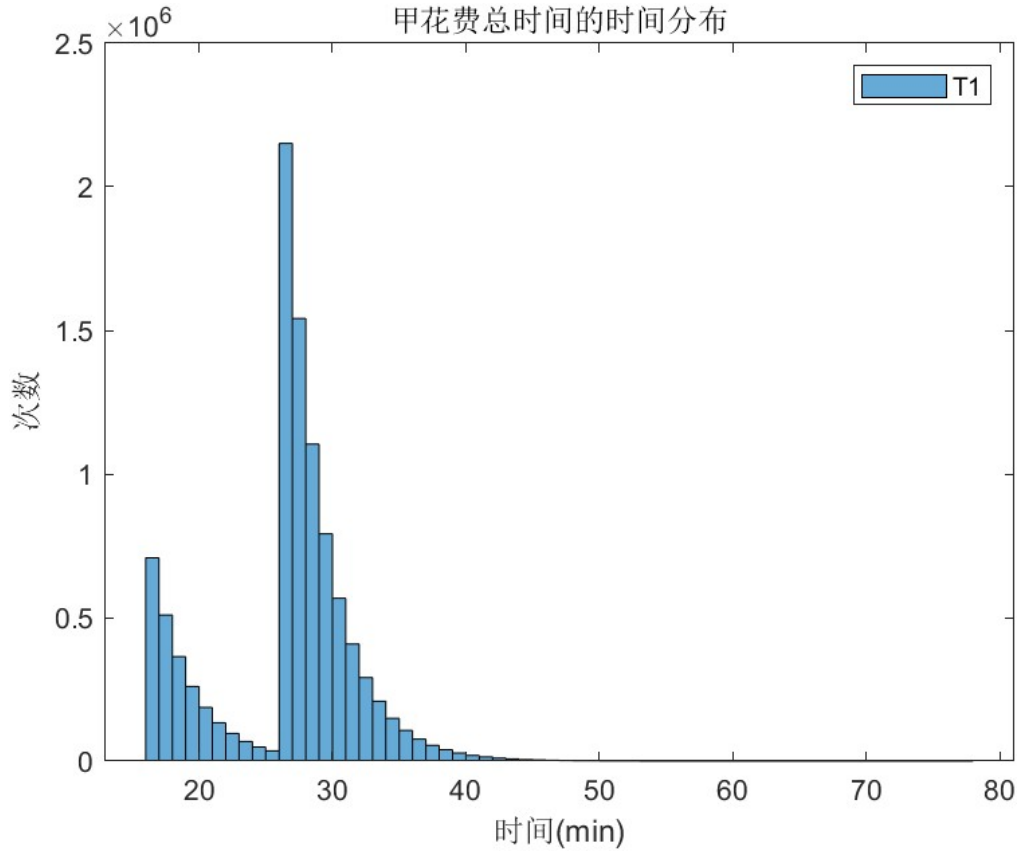
虽然甲乙同时等车时，乙及其所代表的方案似乎具有更高的先到达率。但是这其实限制于两者在同一个等车系统中，如果甲乙是完全独立的两个等车事件，那么甲方案则会期望较好。因此，如果你不是与你的朋友竞争到达目的地的先后次序，还是应当选择甲方案。

模拟结果

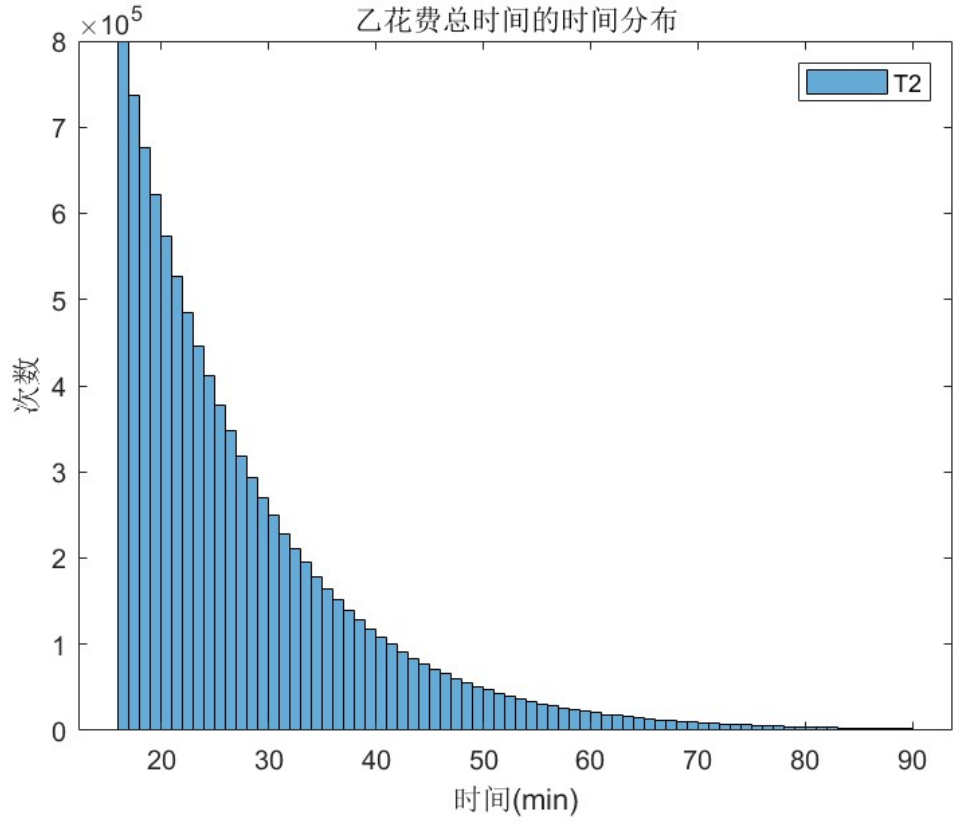
$$p_1 = 0.2500185$$
$$p_2 = 0.4238968$$
$$E(X) = 26.499454090143068min$$
$$E(Y) = 27.999989179785775min$$
$$E(W) = 2.999728458182906min$$



甲等车时间分布为$f(t) = \frac{1}{3}e^{-\frac{t}{3}}$



甲花费总时间的分布可以看作快车和慢车分别代表的两个指数分布的叠加，但是两者的权重是$1:3$



乙所花费的时间分布就是简单的快车指数分布。

第三题

理论

可瘫痪系统

视为Poisson过程。 对于频率λ的暗电流信号，单位时间内期望共有λ次数事件。因为只有$interval \geq \tau_d$的信号才能被记录，所以信号被记录的概率$P(t \geq \tau_d) = e^{-\lambda\tau_d}$，因此预期单位时间记录事件发生$\lambda e^{-\lambda\tau_d}$。此即Poisson过程的λ。由Poisson过程知道期望为$\frac{\lambda}{e^{\lambda\tau_d}}$

不可瘫痪系统

因为指数分布具有无记忆性。所以记录信号的时间间隔期望为(需要对τ_d后的概率进行归一化)：

$$E(t) = \int_{\tau_d}^{\infty} te^{-\lambda t}e^{\lambda\tau_d}dt = \frac{\lambda}{1+\lambda\tau_d}$$

所以，就可以计算

$$\lambda_1 = \frac{\lambda_0}{1+\lambda_0\tau_d}$$

探测器最大可记录事例率

- 可瘫痪系统：$\frac{d}{d\lambda_0}\frac{\lambda_0}{e^{\lambda_0\tau_d}} = e^{-\lambda_0\tau_d}(-\tau_d\lambda_0+1)$，因此$\lambda_0 = \frac{1}{\tau_d}$时，有$\max\lambda_1 = \frac{1}{e\tau_d}$
- 不可瘫痪系统：具有右饱和性，趋向于最大值$\frac{1}{\tau_d}$

模拟过程

暗电流时间序列的产生同第一题，但是：

- 可瘫痪系统中，只有$interval(i) \geq \tau$时，才能被记录：$T_1(new) = \sum^{i} interval(i)$。
- 不可瘫痪系统中，则$\forall i, \exists j_{min}, st \sum_{i}^{j} interval(k) \geq \tau$，则$T_2(new) = \sum^{j_{min}} interval(k)$

  模拟方法考虑为设置一个值delta，$delta = \sum_{i}^{j} interval(k)$ 如果$delta \geq \tau$则进行记录，并且清零delta。

```
while sum(interval)<=0.1
      interval(i)=exprnd(1/lambda(j));
         if interval(i)>=tau
             T1(length(T1)+1)=sum(interval);
         end
      delta=delta+interval(i);
         if delta>=tau
            T2(length(T2)+1)=sum(interval);
            delta=0;
      i=i+1;
end
```

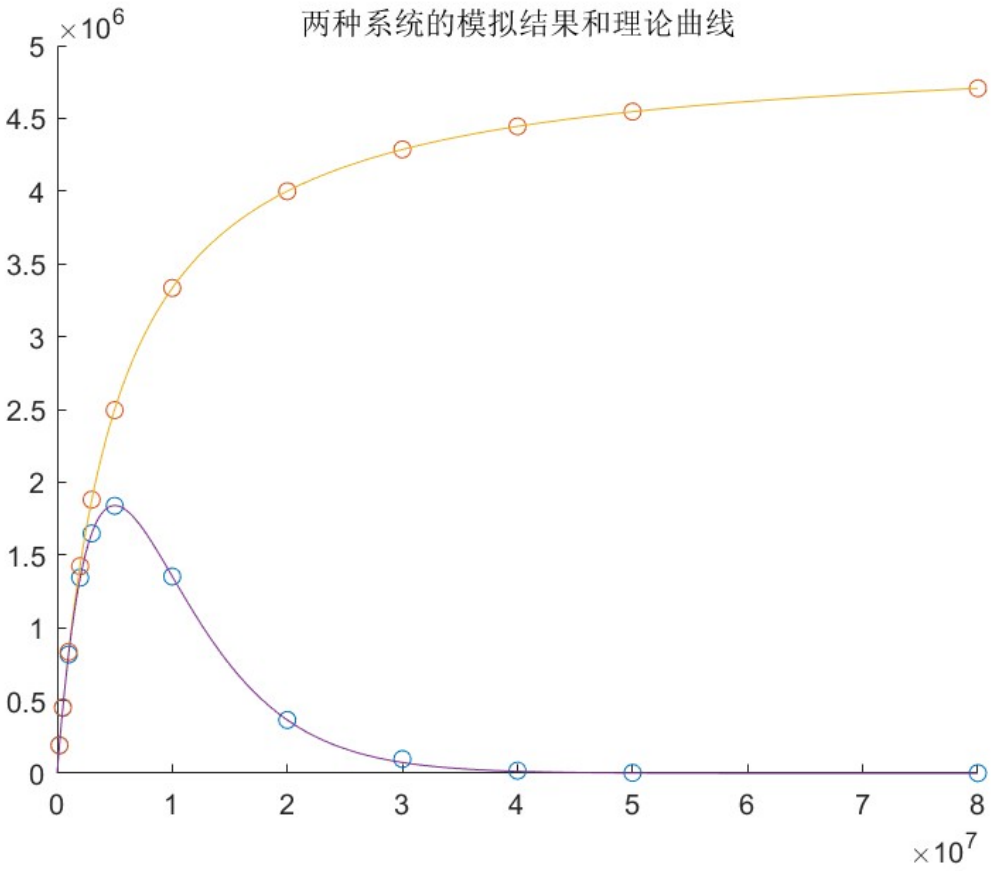
代码重要优化

- 不可瘫痪系统在高频条件中会趋向于“死时间频率”，为了减轻机器负担，所以在$\lambda = 50/80Mhz$时直接采用

$$T(k+1) = T(k) + \tau_d + exprnd(\frac{1}{\lambda})$$

这其实有违真实，但可以证明产生的偏差小于$\frac{1}{\lambda}$，相对误差为$\frac{1}{\lambda\tau_d} \leq 10\%$，因此是可以接受的（？）。

- 由于系统运行在纳秒级别，虽然不能理解，但确实如果判断语句为大于等于，就会使得可瘫痪系统在临界状态时记录的数据偏大，可能是精度不够导致的略小于死时间的信号也被记录。



Appendix

产生暗电流时间序列

```
clear
%tau=0.000001;
lambda1=100;
lambda2=200;
T1=[exprnd(1/lambda1)];
T2=[exprnd(1/lambda2)];
i=2;
while min(max(T1),max(T2))<=10*3600
    T1(i)=T1(i-1)+exprnd(1/lambda1);
    T2(i)=T2(i-1)+exprnd(1/lambda2);
    i=i+1;
end
```

二重偶然符合事件的判断

```
clear
result=[];
tau=1e-5;
load("修正后的第一时序.mat")
load("时序2.mat")
```

```
N=9000;
%%
for i=1:N
    Tadj1=T1adj(find(T1adj<=i*36000/N & T1adj>(i-1)*36000/N));
    Tadj2=T2(find(T2<=i*36000/N & T2>(i-1)*36000/N));
    N1=length(Tadj1);
    N2=length(Tadj2);
    B=Tadj1'*ones(1,N2);
    C=ones(N1,1)*Tadj2;
     delta=B-C;
    [row,col]=find(abs(delta)<=tau);
     m=find(abs(delta)<=tau);
    row1=row(find(delta(m)<0));
    col1=col(find(delta(m)>=0));
    result=[result,Tadj1(row1),Tadj2(col1)];
end
%%
result=sort(result);
A=diag(-ones(1,length(result)-1),-1)+eye(length(result));
delta2=A*result';
histogram(delta2)
legend
title("二重偶然符合时间差的分布")
xlabel("时间(s)")
ylabel("频次")
```

乘车问题

```
lambda=1/3;
p1=0;
p2=0;
T1=[];
T2=[];
W=[];
N=10000000;
for i=1:N
    interval1=exprnd(1/lambda);
    interval2=interval1;
    cho=rand(1);
    W(i)=interval1;
    while cho<=0.75
        interval2=interval2+exprnd(1/lambda);
        cho=rand(1);
    end
    if interval2>interval1
        cost=26;
    else
        cost=16;
    end
    T1(i)=interval1+cost;
    T2(i)=interval2+16;
    if abs(T1(i)-T2(i))==0
        p1=p1+1;
    end
    if T2(i)-T1(i)>0
```

```
        p2=p2+1;
    end
end
aveT1=mean(T1);
aveT2=mean(T2);
p1=p1/N;
p2=p2/N;
aveW=mean(W);
save("101.mat","T1")
save("102.mat","T2")
save("103.mat","p1")
save("104.mat","p2")
save("105.mat","aveT1")
save("106.mat","aveT2")
```

探测器死区模拟未优化整体代码

```
clear
tau=200*1e-9;
lambda=[0.2,0.5,1,2,3,5,10,20,30,40,50,80]*1e6;
interval=[];
delta=0;
Tmatrix=cell(2,length(lambda));
for j=1:12
    i=1;
    T1=[0];
    T2=[0];
    while sum(interval)<=0.1
        interval(i)=exprnd(1/lambda(j));
         if interval(i)>=tau
             T1(length(T1)+1)=sum(interval);
         end
      delta=delta+interval(i);
         if delta>=tau
             T2(length(T2)+1)=sum(interval);
              delta=0;
         else
             continue;
         end
      i=i+1;
    end
    interval=[];
    delta=0;
    Tmatrix(1,j)={T1};
    Tmatrix(2,j)={T2};
end
save("两种系统的记录","Tmatrix")
%%
close all
lambda=[0.2,0.5,1,2,3,5,10,20,30,40,50,80]*1e6;
lambdaa=[];
for i=1:12
   lambdaa(i)=length(Tmatrix{1,i})/max(Tmatrix{1,i});
   lambdab(i)=length(Tmatrix{2,i})/max(Tmatrix{2,i});
end
```

```
scatter(lambda,lambdaa)
hold on
scatter(lambda,lambdab)
hold on
lambda=1:80*1e6;
yy=lambda./(1+lambda.*tau);
plot(lambda,yy)
hold on
yyy=lambda./exp(lambda.*tau );
plot(lambda,yyy)
hold off
legend
```

不可瘫痪系统的优化

```
tau=200*1e-9;
lambda=[0.2,0.5,1,2,3,5,10,20,30,40,50,80]*1e6;
delta=0;
for j=1:8
    i=1;
    T2=[0];
    while sum(interval)<=0.1
        interval(i)=exprnd(1/lambda(j));
        delta=delta+interval(i);
        if delta>=tau
            T2(length(T2)+1)=sum(interval);
            delta=0;
        end
        i=i+1;
    end
    interval=[];
    delta=0;
    Tmatrix(2,j)={T2};
end
for j=9:12
    T2=[0];
    while max(T2)<=0.1
        T2(length(T2)+1)=max(T2)+tau+exprnd(1/lambda(j));
    end
    Tmatrix(2,j)={T2};
end
```

可瘫痪系统高频状态简化代码

```
tau=200*1e-9;
lambda=[0.2,0.5,1,2,3,5,10,20,30,40,50,80]*1e6;
interval=[];
for j=9:12
    i=1;
    T1=[0];
    T2=[0];
    while sum(interval)<=0.001
        interval(i)=exprnd(1/lambda(j));
        if interval(i)>tau
```

```
            T1(length(T1)+1)=sum(interval);
        end
        i=i+1;
    end
    TT1=[];
    for jj=1:100
        TT1=[TT1,T1+(jj-1)*0.001];
    end
    interval=[];
    Tmatrix(1,j)={TT1};
end
```