function [tiempos,Ainstant]=Ainstant(ti,to);

```
% AINSTANT. Calcula y representa el trafico instantaneo (numero de organos
%         ocupados) en funcion del tiempo, dados los instantes de llegada
%         y finalizacion de las llamadas.
%
% FORMATO DE LLAMADA:
%    [tiempos,Ainstant]=Ainstant(ti,to)
%
% PARAMETROS DE ENTRADA:
%    ti: Instantes de llegada [seg]
%    to: Instantes de salida [seg]
%
% PARAMETROS DE SALIDA:
%     tiempos: Instantes iniciales de periodos de tiempo con trafico
%            instantaneo contante [seg]
%    Ainstant: Valores de trafico instantaneo [E]
%

tiempoaux=[ti,to];
signosaux=[ones(size(ti)),-ones(size(to))];
[tiempos,indexes]=sort(tiempoaux);
signos=signosaux(indexes);

Ainstant=cumsum(signos);

tiempos=[0 tiempos];    % Origen de tiempos en t=0
Ainstant=[0 Ainstant];  % Se considera que antes de llegar la primera llamada, el trafico es
cero.

figure
stairs(tiempos,Ainstant);
grid;
xlabel ('Tiempo (s)');
ylabel ('Trafico (E)')
title ('Intensidad de Trafico Instantanea');
```

Ejemplo

```
Pb=[0.5 1 1.5 2 2.5 3 5 7.5 10 15 20]/100;

for n=1:100,
   for p=1:length(Pb),
      E(n,p)=findrhoc(n,Pb(p));
   end
end




%
```

```
% B=erlangb(n,rho)
%
% This function computes the Erlang B probability that a system with n
% servers, no waiting line, Poisson arrival rate lambda, service rate
% (per server) mu, and intensity rho=lambda/mu will have all servers busy.
%
% The probability is
%
%  B=(rho^m/m!)/(sum(rho^k/k!),k=0..m)
%
% It uses a recurrence relation which is more accurate than direct evaluation
% of the formula.  This recurrence relation is a "folk theorem".  The author
% would appreciate a reference to its first publication.  The recurrence is
%
%  B(0,rho)=1
%
%  B(n,rho)=(rho*B(n-1,rho)/n)/(1+rho*B(n-1,rho)/n)
%
%
function B=erlangb(n,rho)
%
% Sanity check- make sure that n is a positive integer.
%
 if ((floor(n) ~= n) | (n < 1))
   warning('n is not a positive integer');
   rho=NaN;
   return;
 end;
%
% Sanity check- make sure that rho >= 0.0.
%
 if (rho < 0.0)
   warning('rho is negative!');
   rho=NaN;
   return;
 end;
%
% Start the recursion with B=1.
%
B=1;
%
% Run the recursion.
%
for k=1:n,
 B=((rho*B)/k)/(1+rho*B/k);
end;
```

```
%
% n=erlangbinv(p,rho)
%
% This function finds the smallest n such the Erlang B probability that
% a system with n servers, no waiting line, Poisson arrival rate lambda,
% service rate (per server) mu, and intensity rho=lambda/mu will have
% a probability <=p of having all servers busy.
%
% The Erlang B probability is given by
%
%  B=(rho^m/m!)/(sum(rho^k/k!),k=0..m)
%
% We use a recurrence relation which is more accurate than direct evaluation
% of the formula.  This recurrence relation is a "folk theorem".  The author
% would appreciate a reference to its first publication.  The recurrence is
%
%  B(0,rho)=1
%
%  B(n,rho)=(rho*B(n-1,rho)/n)/(1+rho*B(n-1,rho)/n)
%
% This routine simply loops through the recursion until B is <= p, and
% then returns n.
%
function n=erlangbinv(p,rho)
%
% Start the recursion with B=1.
%
B=1;
%
% Loop, iterating the recursion until the probability is <= p.
%
n=1;
while (1 == 1),
  B=((rho*B)/n)/(1+rho*B/n);
  if (B <= p),
    return;
  end;
  n=n+1;
end;
```

%

% p=erlangc(n,rho)

%
% This function computes the Erlang C probability that a system with n
% servers, infinite waiting line, Poisson arrival rate lambda, service rate
% (per server) mu, and intensity rho=lambda/mu will have all servers busy.
%
% It uses the formula
%
% C(n,rho)=n*B(n,rho)/(n-rho*(1-B(n,rho)))
%
% See Cooper, Introduction to Queueing Theory, 2nd Ed.
%

function C=erlangc(n,rho)

```
%
% Sanity check- make sure that n is a positive integer.
%
  if ((floor(n) ~= n) | (n < 1))
    warning('n is not a positive integer');
    rho=NaN;
    return;
  end;
%
% Sanity check- make sure that rho >= 0.0.
%
  if (rho < 0.0)
    warning('rho is negative!');
    rho=NaN;
    return;
  end;
%
% Calculate the Erlang B probability and then convert to Erlang C.
%
B=erlangb(n,rho);
C=n*B/(n-rho*(1-B));
```

```matlab
%
% n=erlangcinv(p,rho)
%
% This function finds the smallest n such the Erlang C probability that
% a system with n servers, infinite waiting line, Poisson arrival rate lambda,
% service rate (per server) mu, and intensity rho=lambda/mu will have
% a probability <=p of having all servers busy.
%
% The Erlang B probability is computed using the same recursion that the
% erlangb() function uses.  The B probability is then used to compute
% the Erlang C probability.  The Erlang C probability is given by
%
%  C(n,rho)=n*B(n,rho)/(n-rho*(1-B(n,rho)))
%
% See Cooper, Introduction to Queueing Theory, 2nd Ed.
%
% This routine simply loops through the recursion until C is <= p, and
% then returns n.
%
function n=erlangcinv(p,rho)
%
% Start the recursion with B=1.
%
B=1;
%
% Loop, iterating the recursion until the probability is <= p.
%
n=1;
while (1 == 1),
  B=((rho*B)/n)/(1+rho*B/n);
  C=n*B/(n-rho*(1-B));
  if (C <= p),
    return;
  end;
  n=n+1;
end;
```

function s=exp_service(mu,N)

```
% EXP_SERVICE. Genera tiempos aleatorios de servicio que siguen
%             una funcion de distribucion exponencial negativa.
%
% FORMATO DE LLAMADA:
%    s=exp_service(mu,N)
%
% PARAMETROS DE ENTRADA:
%    mu: Tasa o velocidad media de servicio [1/seg]
%     N: Numero de llamadas []
%
% PARAMETROS DE SALIDA:
%    s: Tiempos de servicio o duracion de las llamadas [seg]
%
%
% Antoni J. Canos. E.P.S.Gandia - U.P.Valencia. Octubre 2002.

format long

F=rand(1,N);

s=-log(1-F)/mu;
```

function [tau,ti]=exp_source(lambda,T);

```
% EXP_SOURCE. Genera tiempos aleatorios de llegadas de llamadas que
%           siguen una funcion de distribucion exponencial negativa
%           y obtiene los intantes de las llegadas en un tiempo de
%           observacion.
%
% FORMATO DE LLAMADA:
%    s=exp_source(lambda,T)
%
% PARAMETROS DE ENTRADA:
%    lambda: Tasa de llegadas o velocidad media de llamadas [1/seg]
%        T: Tiempo de observacion de las llegadas [seg]
%
% PARAMETROS DE SALIDA:
%    tau: Tiempos entre llegadas [seg]
%     ti: Instantes de llegada [seg]
%
%
% Antoni J. Canos. E.P.S.Gandia - U.P.Valencia. Octubre 2002.

format long

tn=0;
ii=0;

while tn<=T

ii=ii+1;

F=rand(1,1);

tau(ii)=-log(1-F)/lambda;

tn=tn+tau(ii);

ti(ii)=tn;

end

tau=tau(1:end-1); % Tiempos entre instantes de llegada
ti=ti(1:end-1); % Instantes de llegada (t_input)
```

% rho=findrhob(n,p)

```
% Finds an intensity rho such that B(n,rho)=p.
% Note: Must have 0<p<1.  Returns NaN if p is not in this range.
```

function rho=findrhob(n,p)

```
% Sanity check- make sure that n is a positive integer.
  if ((floor(n) ~= n) | (n < 1))
    warning('n is not a positive integer');
    rho=NaN;
    return;
  end;
%
% Sanity check- make sure that p is a probability with 0<p<1.
%
  if ((p<0.0) | (p>1.0))
     warning('Invalid p value!');
     rho=NaN;
     return;
  end;
%
% We know that at rho=0, p=0, and at rho=+Inf, p=1.   We start by finding
% an interval [0,a] containing the root.
%
a=1.0;
testp=erlangb(n,a);
while (testp < p),
  a=a*2.0;
  testp=erlangb(n,a);
end;
%
% Now, the root is somewhere between 0 and a.  Use bisection to find it.
%
%
left=0.0;
right=a;
mid=(left+right)/2;
midp=erlangb(n,mid);
while ((right-left) > 0.0001*max([1 left])),
  if (midp < p),
    left=mid;
    mid=(left+right)/2;
    midp=erlangb(n,mid);
  else
    right=mid;
    mid=(left+right)/2;
    midp=erlangb(n,mid);
  end;
end;
%
% Return the left end point of the current interval, which has prob < p.
rho=left;
```

% rho=findrhoc(n,p)

```
% Finds an intensity rho such that C(n,rho)=p.
% Note: Must have 0<p<1.  Returns NaN if p is not in this range.
%
```

function rho=findrhoc(n,p)

```
% Sanity check- make sure that n is a positive integer.
%
  if ((floor(n) ~= n) | (n < 1))
    warning('n is not a positive integer');
    rho=NaN;
    return;
  end;
%
% Sanity check- make sure that p is a probability with 0<p<1.
  if ((p<0.0) | (p>1.0))
     warning('Invalid p value!');
     rho=NaN;
     return;
  end;
%
% We know that at rho=0, p=0, and at rho=+Inf, p=1.   We start by finding
% an interval [0,a] containing the root.
%
a=1.0;
testp=erlangc(n,a);
while (testp < p),
  a=a*2.0;
  testp=erlangc(n,a);
end;
%
% Now, the root is somewhere between 0 and a.  Use bisection to find it.
%
%
left=0.0;
right=a;
mid=(left+right)/2;
midp=erlangc(n,mid);
while ((right-left) > 0.0001*max([1 left])),
  if (midp < p),
    left=mid;
    mid=(left+right)/2;
    midp=erlangc(n,mid);
  else
    right=mid;
    mid=(left+right)/2;
    midp=erlangc(n,mid);
  end;
end;
% Return the left end point of the current interval, which has prob < p.
rho=left;
```

function PB=poisson(C,Ao)

```
% POISSON. Calcula la probabilidad de bloqueo (PB) de un modelo de colas de Poisson.
%        Esto es, un sistema con infinitos servidores o canales de los cuales C
%        son reales y el resto ficticios y sin cola de espera, al que se le ofrece
%        un tráfico Ao con tiempos entre llegadas y tiempos de servicio
%        exponenciales. Se considera que hay bloqueo cuando hay C o mas servidores
%        ocupados.
%
% FORMATO DE LLAMADA:
%   PB=poisson(C,Ao)
%
% PARAMETROS DE ENTRADA:
%    C: Numero de canales. (Vector de enteros o escalar entero) []
%   Ao: Trafico ofrecido.  (Vector o escalar real) [E]
%
% PARAMETROS DE SALIDA:
%   PB: Probabilidad de bloqueo. PB(i,j)=poisson(C(i),Ao(j). []
%
%
% Antoni J. Canos. E.P.S.Gandia - U.P.Valencia. Julio 2003.

C=C(:).';
Ao=Ao(:).';
LC=length(C); % C [1xLC]
LA=length(Ao); % A [1xLA]
for ii=1:LC
  k=[0:1:C(ii)-1];   % [1xLk]
  Lk=length(k);
  factk=fact(k);     % [1xLk]
  invfactk=1./factk; % [1xLk]
  % Ao [1xLA] ; k [1xLk]
  Aok=(ones(Lk,1)*Ao).^((ones(LA,1)*k).'); %[LkxLA]
  sumat=invfactk*Aok; %[1xLA]
  PB(ii,:)=1-exp(-Ao).*sumat;
end
```

%Cuestion 2

```
clear all
close all
lambda=0.4;
t=linspace(1,15,100);
f_t=lambda*exp(-lambda*t);
F_t=1-exp(-lambda*t);
plot(t,f_t)
figure
plot(t,F_t)
pause
```

%Cuestión 3

```
T=18000;
tn=0;
ii=0;
while tn<=T %Este bucle se cumple hasta que T vale 180 sg
ii=ii+1;
F=rand;
tau(ii)=-log(1-F)/lambda;
tn=tn+tau(ii);
ti(ii)=tn;
end
```

%%Cuesion 4

```
hist(tau,51)
```

%Cuestión 5

```
%Se trata de repecir 4 veces el apartado 3
1/mean(tau)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tt=10;
centros=[tt/2:tt:T];
m=hist(ti,centros);
```

%Cuestion 6

```
tt=10;centros=[tt/2:tt:T];
m=hist(ti,centros);
k=[0:1:max(m)];
contadas=hist(m,k);
pk=contadas/sum(contadas);
lambda_prima=k*pk.'
```

%Cuesion 7

```
for ii=1:length(k)
pkfitted(ii)=lambda_prima^k(ii)*exp(-lambda_prima)/factorial(k(ii));
end
bar(k,pk); hold on;
plot(k,pkfitted,'r'); plot(k,pkfitted,'.r');
grid; hold off
%%%%%%%%%%
```

%Cuestion 8

```
clear all
T=48;
lambda=0.4;
mu=0.1;
tn=0;
ii=0;
while tn<=T %Este bucle se cumple hasta que T vale 180 sg
ii=ii+1;
tau(ii)=-log(1-rand)/lambda;
tn=tn+tau(ii);
ti(ii)=tn;
s(ii)=-log(1-rand)/mu;
end
to=ti+s; to=sort(to);
```

%%%Cuestion 9

```
stem(ti,ones(size(ti)),'.b'); hold on
stem(to,-ones(size(to)),'.r');
grid; hold off;
```

%cuesion 10

```
tiempoaux=[ti,to];
signosaux=[ones(size(ti)),-ones(size(to))];
[tiempos,indexes]=sort(tiempoaux);
signos=signosaux(indexes);
Ainstant=cumsum(signos);
tiempos=[0 tiempos];    % Origen de tiempos en t=0
Ainstant=[0 Ainstant];  % Se considera que antes de llegar la primera llamada, el trafico es cero.
figure
stairs(tiempos,Ainstant);
grid;
xlabel ('Tiempo (s)');
ylabel ('Trafico (E)')
title ('Intensidad de Trafico Instantanea');
```

%Cuesion 11

```
taus=diff(tiempos);
Ao_aprox=Ainstant(1:end-1)*taus.'/tiempos(end)
Ao=lambda/mu
```

%Cuesion 12: Repetir lo mismo

%Cuestion 14

```
B=erlangb(n,rho)
B=1;
for k=1:n,
  B=((rho*B)/k)/(1+rho*B/k);
end;
```

%Cuestion 15:

```
Pb=[0.5 1 5 10]/100;
for n=1:10,
   for p=1:length(Pb),
      E(n,p)=findrhob(n,Pb(p));
   end
end
E
```

%Cuesion 17

```
function C=erlangc(n,rho)
B=erlangb(n,rho);
C=n*B/(n-rho*(1-B));
```

%Cuestion 18:

```
Pb=[0.5 1 5 10]/100;
for n=1:10,
   for p=1:length(Pb),
      E(n,p)=findrhoc(n,Pb(p));
   end
end
E
```