Señales aleatorias y ruido.
Simulacion de un proceso de llegadas y salidas en tiempos determinísticos

Arcadio Alexis Calvillo Madrid 159702

03 Abril 2019

1 Procesos de nacimiento y muerte (Birth-death)

Sea un proceso de Markov en el que las transiciones entre estados sólo se da entre estados vecinos, entonces este proceso se denomina como *un proceso de vidas y muertes*. Ahora se presentan dos casos; uno en el que la cantidad de "vidas" que puede haber es infinita y el que veremos en esta simulación en el cual la cantidad de "vidas" es fija y finita.

En (Fig. 1) se muestra la representación de un proceso de vidas y muertes con la particularidad de que no se puede quedar en el mismo estado.

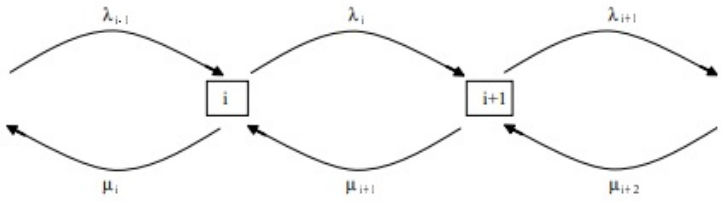


Fig. 1: Representación de un proceso de vidas y muertes

2 Proceso de un servidor en tiempos discretos

Sea un *proceso de vidas y muertes* con las siguientes características:

1) Llegan al servidor con una probabilidad p_a

2) No llega nadie con una probabilidad $1 - p_a$

3) Sale alguien con una probabilidad p_b

4) No sale nadie con una probabilidad $1 - p_b$

5) Si el servidor está lleno ya no llega nadie

6) Si el servidor está vacío no sale nadie

Entonces (Fig. 2) el diagrama de transición de estados cambia un poco ya que ahora sí se puede quedar en el mismo estado y, entre otras particularidades, en el primer estado y el último las probabilidades de transición cambian.

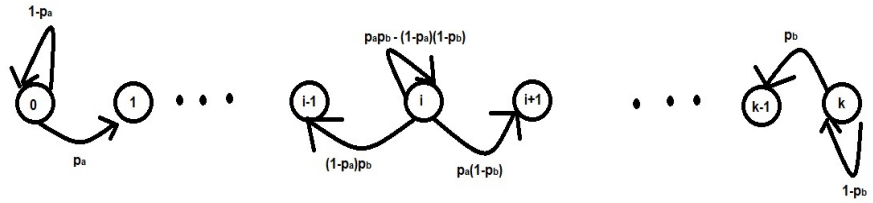


Fig. 2: Representación de llegadas y salidas de un servidor (discreto)

3 Código

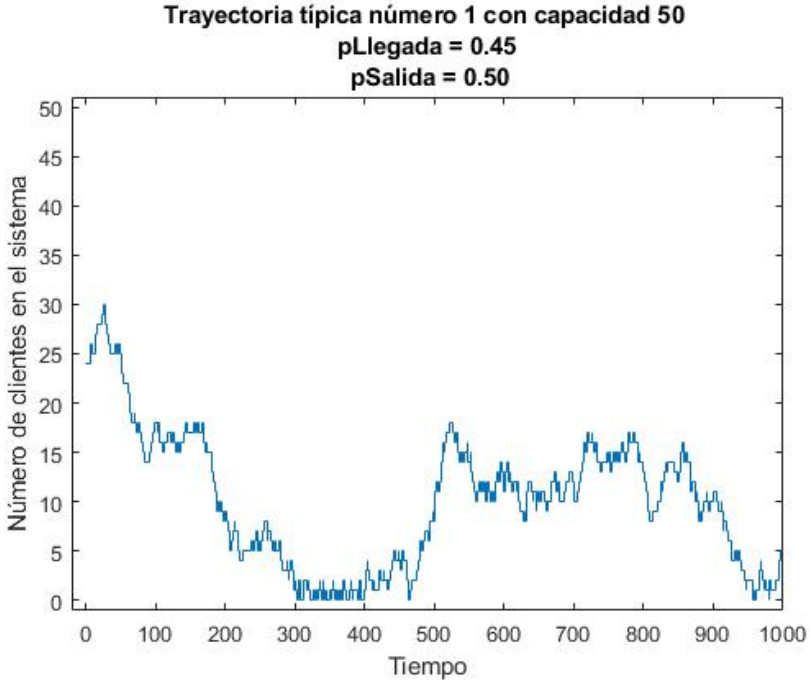
El objetivo de la tarea es simular un *proceso de llegadas y salidas* con una capacidad fija en *MatLab*. Para esto se propone el siguiente código

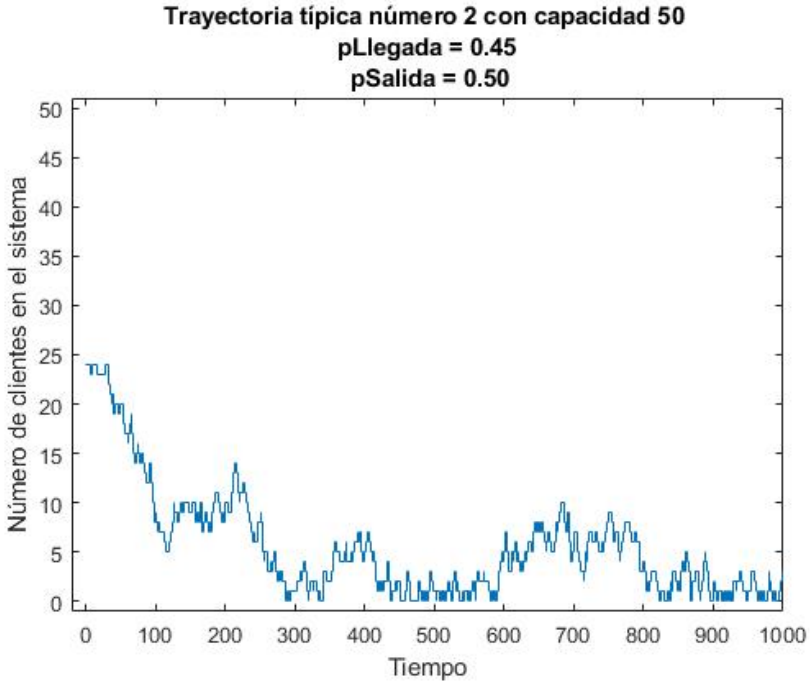
```
function simulacionVidMuer(pa,pb,nT,k,nTray)
%pa      Probabilidad de llegada
%pb      Probabilidad de salida
%nT      N mero de tiempos
%k       Capacidad del "servidor"
%nTray   N mero de trayectorias
close all;

numK=zeros(nTray,nT);

%Estados iniciales del servidor
numK(:,1)=randi(k,nTray,1);

for i=1:nTray
    for j=1:nT-1
        numK(i,j+1)=numK(i,j)+((rand()<pa).*(numK(i,j)<k))-((rand()
            <pb).*(numK(i,j)>0));
    end
end
```

```
xt=linspace(1,nT,nT);
%Gr fica de tres trayectorias t picas
for i=1:3
    figure(i);
    ssf=sprintf('Trayectoria t pica n mero %d',i);
    stairs(xt,numK(i,:)),title(ssf);
    axis([-nT*.02,nT+1,-1,k+1])
    xlabel('Tiempo');
    ylabel('N mero de clientes en el sistema');
end

%Distribuci n de estados
ppi=zeros(1,k+1);
for i=1:k+1
    ppi(i)=sum(sum(numK==i-1))/(nTray*nT);
end
%Comprobaci n de que es una distribuci n
sum(ppi)

%Vector de estado, s lo para graficar
xk=[0:1:k];

figure(4)
stem(xk,ppi,'x')
hold on;
ssf=sprintf('Distribuci n de estados de un servidor con %d de
    capacidad y \n %d Pasos \n %d Trayectorias \n pLlegada = %f
    pSalida = %f',...
    k,nT,nTray,pa,pb);
plot(xk,ppi,'--'),title(ssf);
xlabel('N mero de clientes');
ylabel('Pi');
```

4 Resultados de la simulación

4.1 Resultados para $p_a = .45$ y $p_b = .5$





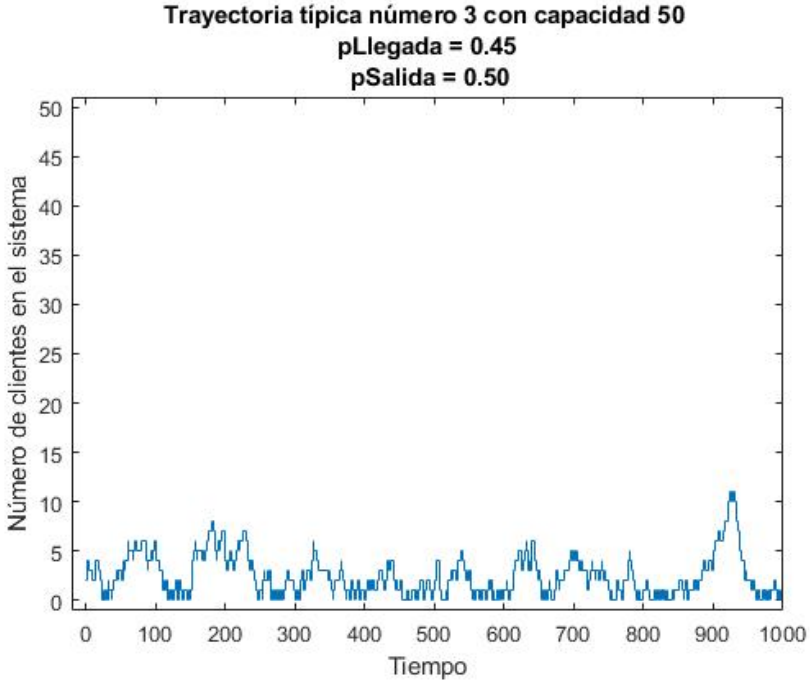


Fig. 3: Trayecorias típicas

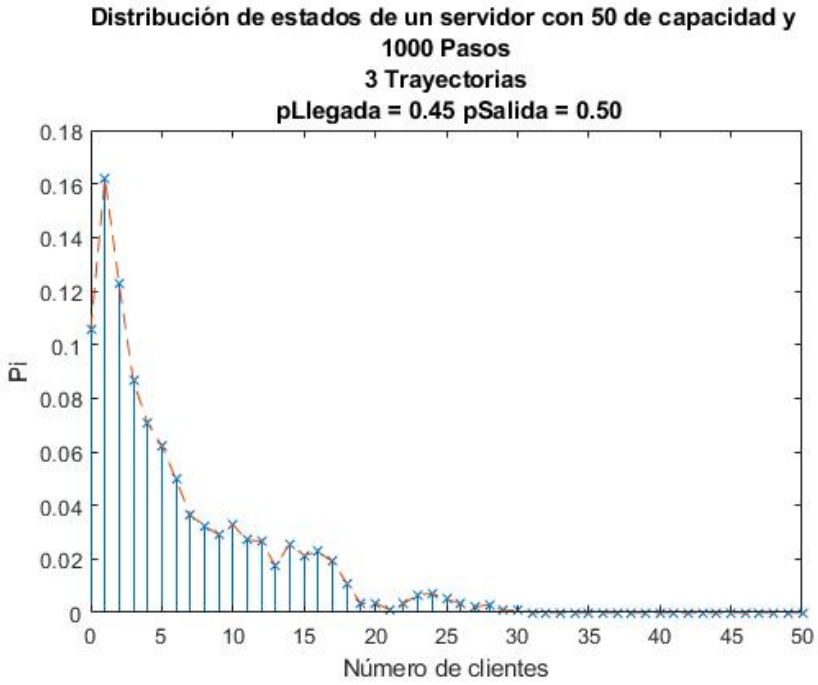
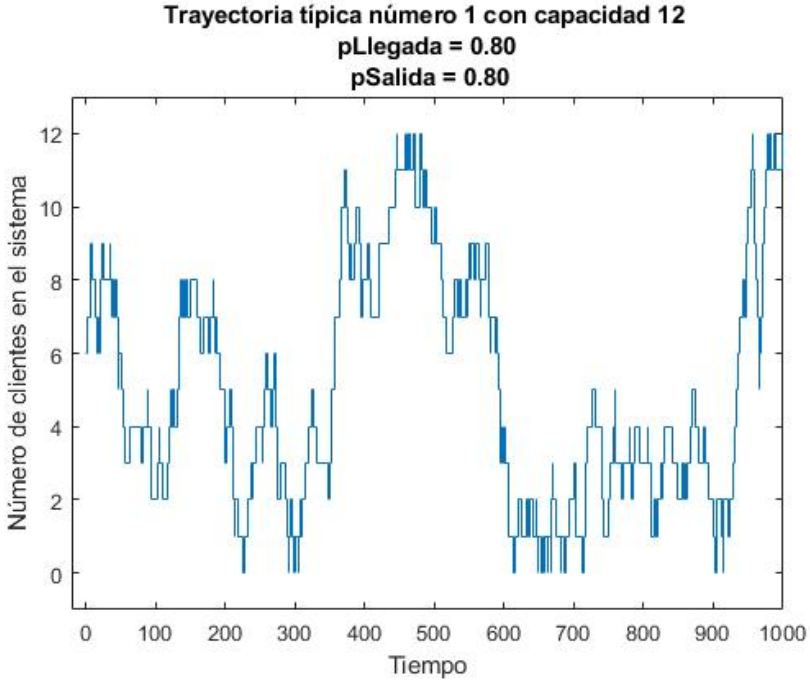
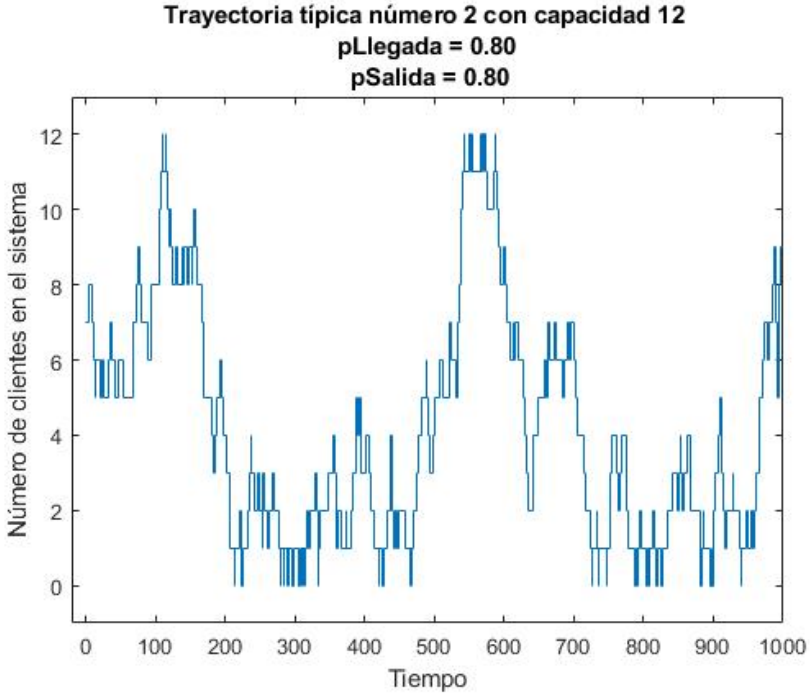


Fig. 4: Estimador de distribución de estados

4.2 Resultados para $p_a = .8$ y $p_b = .8$





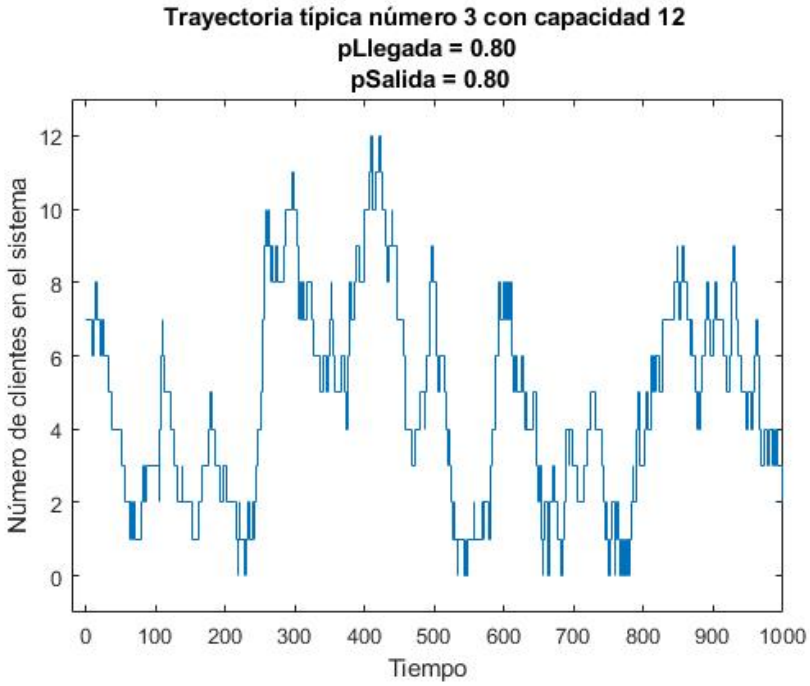


Fig. 5: Trayecorias típicas con $p_a = p_b = .8$

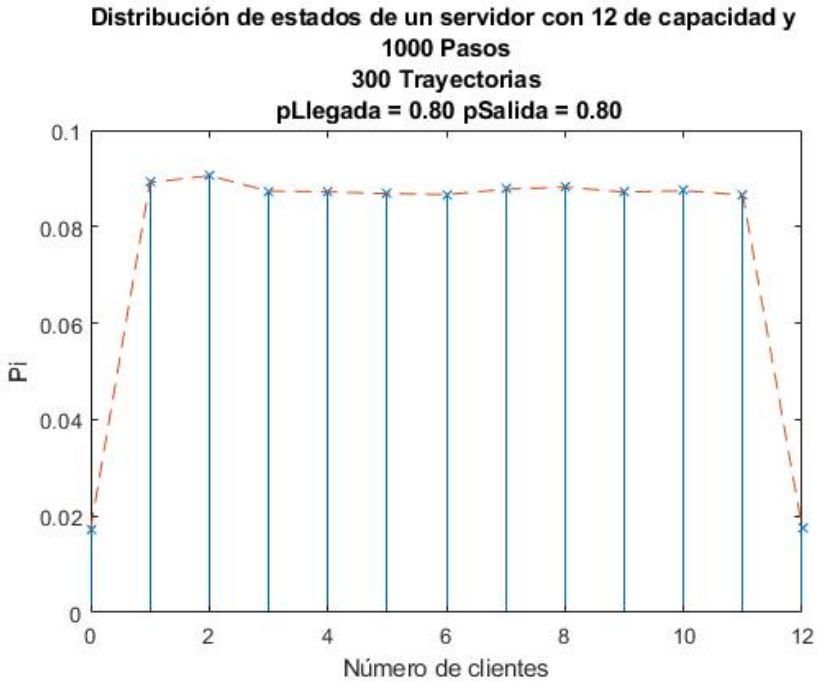


Fig. 6: Estimador de distribución de estados con $p_a = p_b = .8$

4.3 Otros parámetros

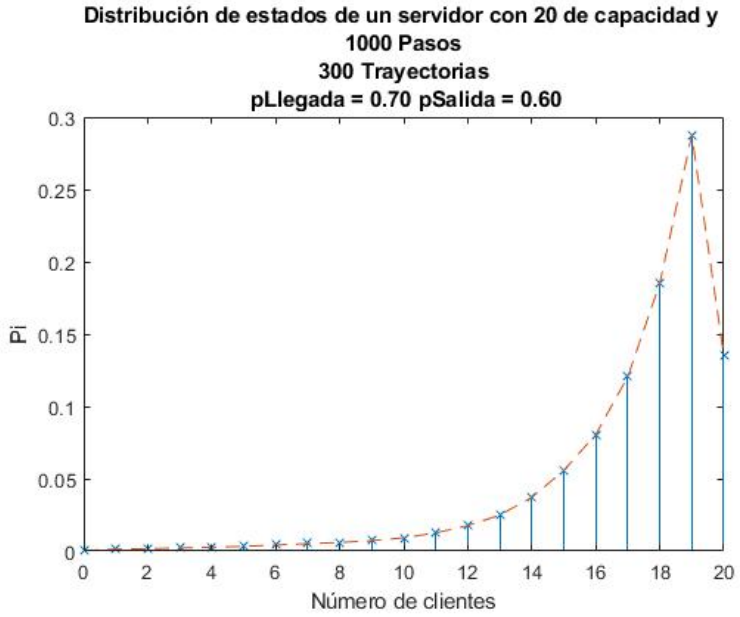


Fig. 7: Estimador de distribución de estados con $p_a = .7$ y $p_b = .6$

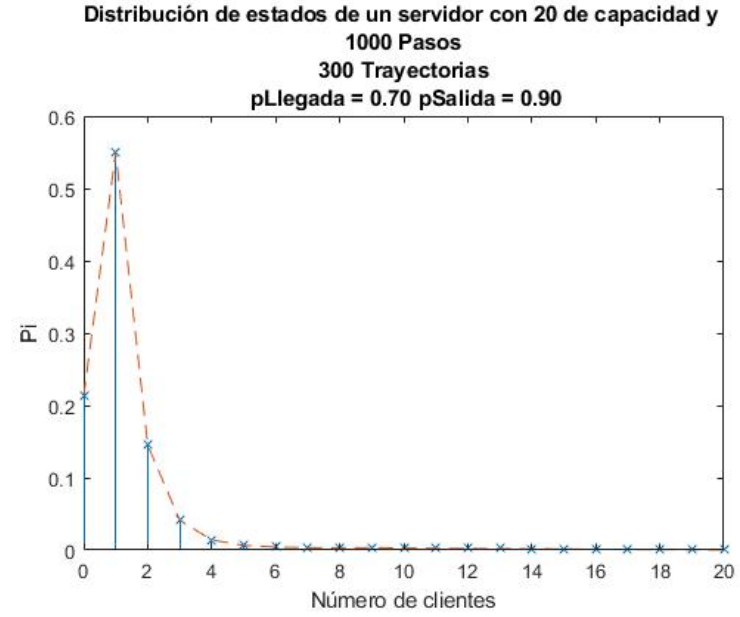


Fig. 8: Estimador de distribución de estados con $p_a = .7$ y $p_b = .9$

5 Conclusiones

Como primera conclusión viendo los tipos de comportamiento se tiene que en un *proceso de llegadas y salidas de un servidor (de tiempos deterministas)* el servidor sólo se vaciará si y sólo si la probabilidad de salida es menor que la probabilidad de llegadas, es decir:

$$p_b > p_b$$

Por otro lado, si nos encontramos en el caso en que ambas probabilidades son iguales, es decir:

$$p_b = p_a$$

el servidor tiende a estar más tiempo ocupado que lleno o vacío (Fig. 6).

Como último caso, y menos deseable, tenemos el caso en el que la probabilidad de llegada es mayor a la de salida, es decir:

$$pa > pb$$

En este caso si el servidor tuviera cola de espera esta se haría infinitamente grande.