COLA MMKK - ERLANG B

Autores

ANDRES LOPEZ 20191195005

WILMAR RENGIFO 20191195007

IVÁN BELLO 20191195002

HERRAMIENTAS MATEMÁTICAS PARA EL MANEJO DE LA INFORMACIÓN

HANS IGOR LÓPEZ CHÁVEZ

Universidad Distrital Francisco José De Caldas

Maestría en Ciencias de la Información y las Comunicaciones

Bogotá, Colombia

Junio de 2019

TABLA DE CONTENIDO

OBJETIVO

Diseñar y construir una herramienta de simulación que permita aplicar los conceptos de independencia, estimación de parámetros y generación de números aleatorios, por medio de un ejemplo asociado a la Teoría de Colas

1.MODELO MATEMÁTICO

A continuación se presenta el modelo matemático

Palabras claves

Erlang

La intensidad de tráfico está medida en Erlangs, donde 1 Erlang es un circuito en uso por 3600 segundos, la intensidad de tráfico también está medida en "Circuit Centum Seconds" CCS, 1 CSS es un circuito en uso por 100 segundos[1].

Grado de Servicio

GoS está definido como la probabilidad de que una llamada falle. Por lo tanto, un sistema de comunicación con todos los canales ocupados rechazará, debido a la congestión de cualquier llamada adicional a las anteriores, el rango es entre 0 y 1 donde 0 es el valor ideal en un sistema de comunicación[2].

Capacidad de Canal

Se puede definir como la capacidad del sistema para ofrecer canales libres a sus suscriptores, es decir, los servidores.

Erlang B

Las llamadas que son bloqueadas toman una nueva ruta y nunca regresan a la troncal original. Es decir, lo que diferencia este tipo de fórmula de bloqueo con las demás fomulas es que el usuario realiza un único intento de llamada, el cuál si no logra establecerlo será enrutado otra vez de manera inmediata.

Lambda

Número promedio de llegadas en una unidad de tiempo

Mu

Tasa medio de servicio para todo el sistema

Transición de Estados

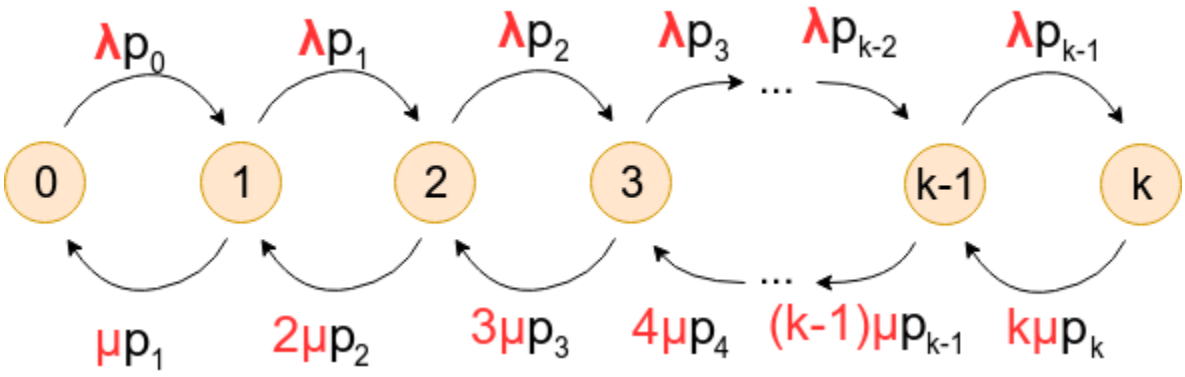

Ilustración 1 - Transición de Estados para Cola MMKK

Ecuaciones de Estado Estacionario.

Evento	Tasa Promedio salida del estado	=	Tasa Promedio llegada del estado	P
0	λP_0	=	μP_1	$P_1 = \frac{\lambda}{\mu}.P_0$
1	$\lambda P_1 + \mu P_1$	=	$\lambda P_0 + 2\mu P_2$	$P_2 = \frac{\lambda}{2\mu}P_1$
2	$\lambda P_2 + 2\mu P_2$	=	$\lambda P_1 + 3\mu P_3$	$P_3 = \frac{\lambda}{3\mu}P_2$
3	$\lambda P_3 + 3\mu P_3$	=	$\lambda P_2 + 4\mu P_4$	$P_4 = \frac{\lambda}{4\mu}P_3$
4	$\lambda P_4 + 4\mu P_4$	=	$\lambda P_3 + 5\mu P_5$	$P_5 = \frac{\lambda}{5\mu}P_4$
$k-1$	$\lambda P_{k-1} + (k-1)\mu P_{k-1}$	=	$\lambda P_{k-2} + k\mu P_k$	$P_k = \frac{\lambda}{k\mu}P_{k-1}$
k	$k\mu P_k$	=	λP_{k-1}	$P_k = \frac{\lambda}{k\mu}P_{k-1}$

Ahora P_n en términos de P_0:

$$P_1 = \frac{\lambda}{\mu}.P_0$$

$$P_2 = \frac{\lambda}{2\mu}.P_1 \Longrightarrow P_2 = \frac{1}{2}\left(\frac{\lambda}{\mu}\right)^2.P_0$$

$$P_3 = \frac{\lambda}{3\mu}.P_2 \Longrightarrow P_3 = \frac{1}{2}.\frac{1}{3}\left(\frac{\lambda}{\mu}\right)^3.P_0$$

$$P_4 = \frac{\lambda}{4\mu}.P_3 \implies P_3 = \frac{1}{2}.\frac{1}{3}.\frac{1}{4}\left(\frac{\lambda}{\mu}\right)^4.P_0$$

$$\text{(I) } P_n = \frac{1}{n!}.\left(\frac{\lambda}{\mu}\right)^n.P_0$$

Entonces, la suma de todas las probabilidades es uno, por lo tanto

$$P_0 + P_1 + P_2 + ... + P_k = 1$$

$\sum_{n=0}^{\infty}.P_n = 1$ remplazando P_n

$$\sum_{n=0}^{\infty}.\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n.P_0 = 1$$

Entonces

$$\text{(II) } P_0 = \frac{1}{\sum_{n=0}^{\infty}.\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n}$$

Remplazando (II) en (I) $r = \frac{\lambda}{\mu}$

$$P_n = \frac{\frac{1}{n!}.r^n}{\sum_{l=0}^{c}\frac{1}{l!}.r^l}$$

$$P_1 = \frac{\lambda}{\mu}.P_0$$

$$P_2 = \frac{\lambda}{\mu}.P_1 \implies P_2 = \left(\frac{\lambda}{\mu}\right)^2.P_0$$

$$P_3 = \frac{\lambda}{\mu}.P_2 \implies P_3 = \left(\frac{\lambda}{\mu}\right)^3.P_0$$

para nuestro caso:

$$P_1 = \frac{\lambda}{\mu}.P_0$$

$$P_2 = \frac{\lambda}{2\mu}.P_1 \implies P_2 = \frac{1}{2}\left(\frac{\lambda}{\mu}\right)^2.P_0$$

$$P_3 = \frac{\lambda}{3\mu}.P_2 \text{ - > } P_3 = \frac{1}{2}.\frac{1}{3}\left(\frac{\lambda}{\mu}\right)^3.P_0$$

$$P_4 = \frac{\lambda}{4\mu}.P_3 \text{ -> } P_3 = \frac{1}{2}.\frac{1}{3}.\frac{1}{4}\left(\frac{\lambda}{\mu}\right)^4.P_0$$

$$P_n = \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n.P_0$$

Función de perdida de MMKK (dónde n=c):

Probabilidad de pérdida $B(c.r) = P_c = \frac{\frac{r^c}{c!}}{\sum_{l=0}^{c}\frac{r^l}{l!}}$

dónde c será en número máximo de servidores

Función Recursiva

$$r = \frac{\lambda}{\mu} \text{ -> } B(c.r) = \frac{rB(c-1,r)}{c + rB(c-1,r)}$$

2. FUNCIÓN MMKK

Para simular la función MMKK propuesta se diseñó y codificó una función usando el software MATLAB. La función se prueba con parámetros de entrada de la función deben ser: un vector con los tiempos entre arribos consecutivos y un vector con los tiempos de servicio. Como resultado se obtuvo la medida de desempeño.

Notación Kendall Lee

M

Distribución llegada de los clientes

M

Distribución de servidores

K

Número de servidores

K

Tamaño del sistema

Función Teórica

Se han comparado los valores con la función teórica recursiva:

```
lambda = 3;
mu = 4;
servidores = 2;
mmkkteorico(lambda,mu, servidores)
```

```
ans = 0.1385
```

Velocidad de función Teórica

Ahora se comprobará la velocidad de la función teorica

```
tic
lambda = 3;
mu = 4;
servidores = 2;
mmkkteorico(lambda,mu,servidores)
```

```
ans = 0.1385
```

```
toc
```

```
Elapsed time is 0.008974 seconds.
```

Función MMKK prueba con valores fijos

Para verificar el funcionamiento de la cola, se realizó la función mmkk, que toma como parámetros de entrada los vectores:

```
tea = [2,7,5,3,2,4,8,5,2,1];
tds = [1,8,14,2,9,3,6,4,3,4];
servidores = 2;
```

y retorna todas las variables consideradas durante la simulación:

```
tic
mmkk( tea, tds, servidores, 1 )
```

```
tea = 1×10
     2     7     5     3     2     4     8     5     2     1
tds = 1×10
     1     8    14     2     9     3     6     4     3     4
reloj = 1×10
     2     9    14    17    19    23    31    36    38    39
salida = 1×10
     0    17    28    19    28    28    37    40    41    41
bloqueo = 1×10
     0     0     0     0     0     1     0     0     0     1
quienatiende = 1×10
     1     1     2     1     1     0     1     2     1     0
p_bloqueo = 0.2000
ans = 0.2000
```

```
toc
```

```
Elapsed time is 0.037891 seconds.
```

Función MMKK prueba con valores enteros aleatorios

Ahora se realizará la prueba con valores enteros aleatorios

```
usuarios = 20;
tea = randi(3,1,usuarios);
tds = randi(4,1,usuarios);
servidores = 2;
tic
mmkk( tea, tds, servidores, 1)
```

```
tea = 1×20
     3     3     1     3     2     1     1     2     3     3     1     3 ⋯
tds = 1×20
     3     1     4     4     3     4     3     2     3     1     3     1 ⋯
reloj = 1×20
     3     6     7    10    12    13    14    16    19    22    23    26 ⋯
salida = 1×20
     0     7    11    14    15    15    17    18    22    23    26    27 ⋯
bloqueo = 1×20
     0     0     0     0     0     1     0     0     0     0     0     0 ⋯
quienatiende = 1×20
     1     1     1     2     1     0     2     1     1     1     1     1 ⋯
p_bloqueo = 0.0500
ans = 0.0500
```

```
toc
```

```
Elapsed time is 0.735275 seconds.
```

Validación Función MMKK prueba con tiempos exponencialmente distribuidos

En la siguiente prueba se muestra

```
lambda = 3;
mu = 4;
servidores = 2;
usuarios = 100000;
tea = exprnd(1/lambda,1,usuarios);
tds = exprnd(1/mu,1,usuarios);
tic
bloqueo = mmkk(tea, tds, servidores)
```

```
bloqueo = 0.1373
```

```
toc
```

```
Elapsed time is 0.780084 seconds.
```

```
mmkkteorico(lambda,mu,servidores)
```

```
ans = 0.1385
```

```
figure ();
pdfcdfcontinua(tea)
```

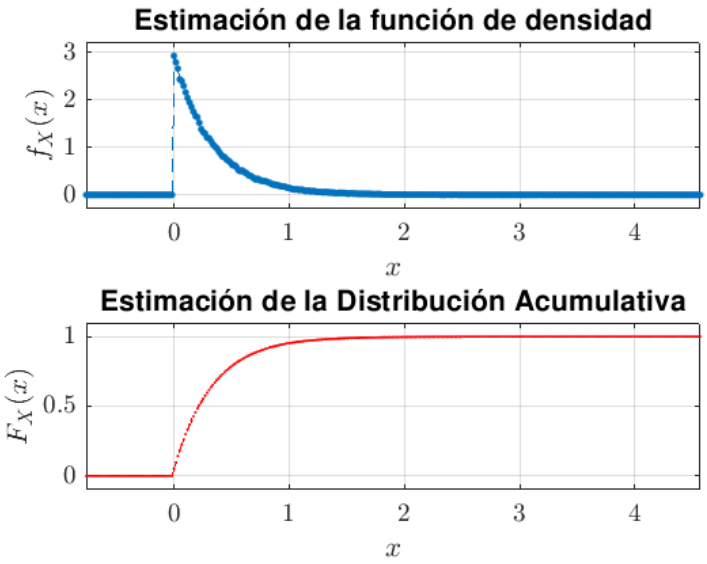

Ilustración 2 - Función MMKK prueba con tiempos exponencialmente distribuidos (TEA)

```
figure();
pdfcdfcontinua(tds)
```

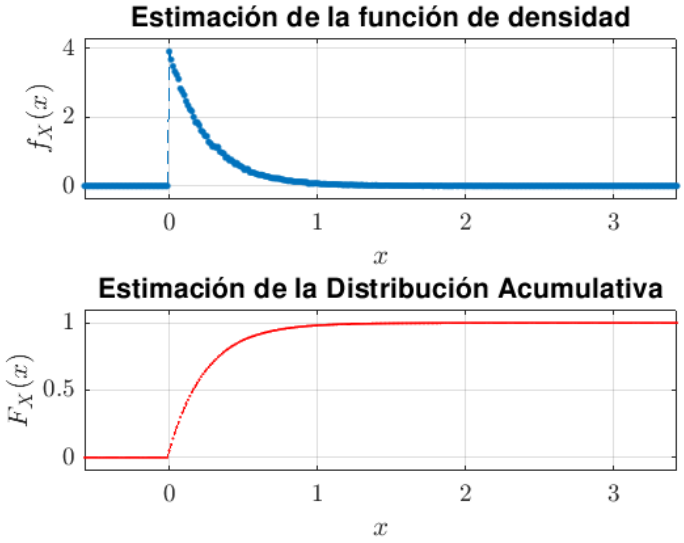

Ilustración 3 - Función MMKK prueba con tiempos exponencialmente distribuidos (TDS)

```
figure();
mmkk(tea, tds, servidores,2);
```

```
bloqueo = 1×100000
     0     0     0     0     1     0     0     0     0     1     0     0 ···
```

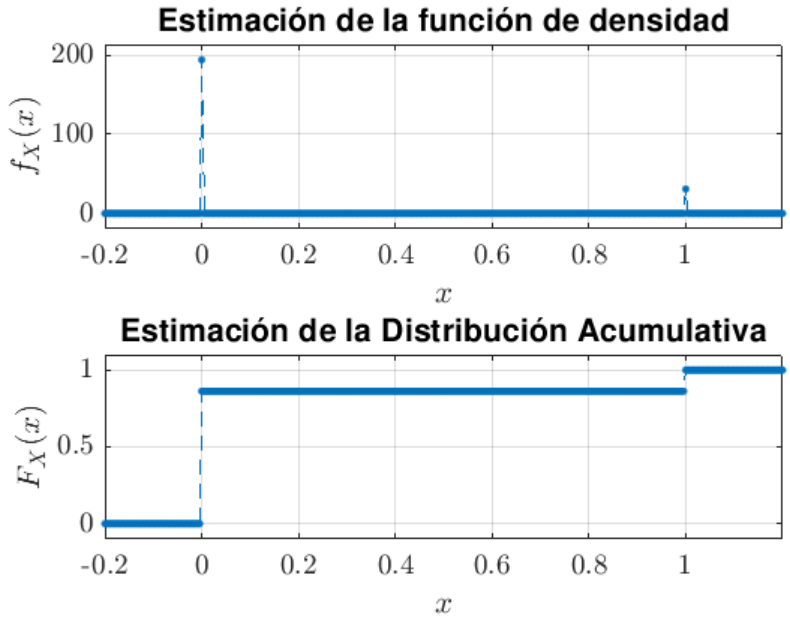

Ilustración 4 - Estimación de PDF y CDF

Función super MMKK

Esta función se implementa para hacer visible ciertos indicadores de desempeño de la cola necesarios para tener una imagen completa de la eficiencia de la misma. Los indicadores que se calcularon con esta función son:

- Cual servidor esta ocupado en un momento especifico
- Cuantos servidores estan en ocio en un momento especifico
- Porcentaje de ocupacion en un momento especifico
- Probabilidad de ocio en el sistema
- Usuarios simultaneos en un momento especifico
- Promedio de usuarios simultaneos en el sistema
- Total de usuarios atendidos
- Total de usuarios bloqueados
- Tiempo medio de atención

Flujograma de la función super_MMKK:

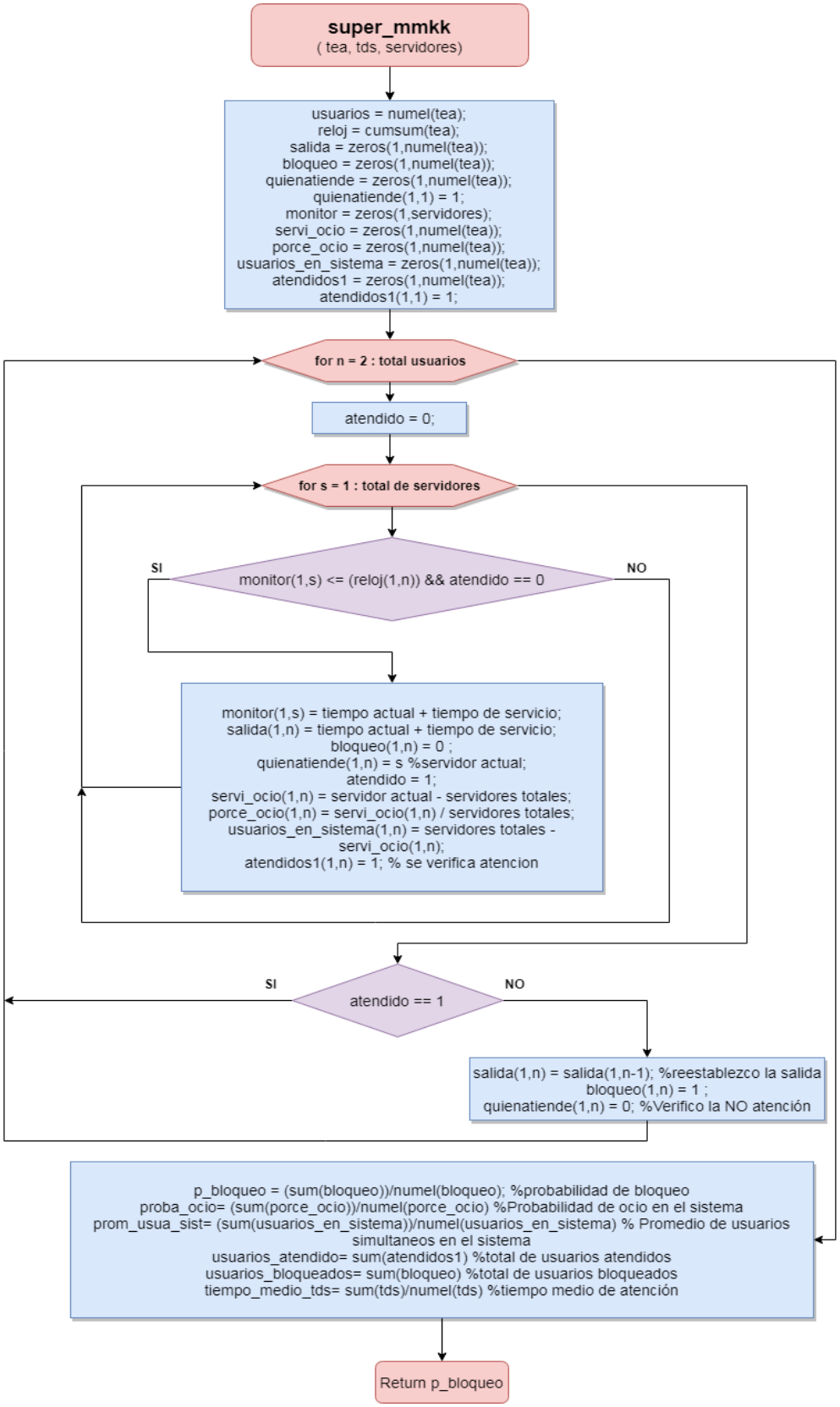

Ilustración 5 - Flujograma función super MMKK

```
clc
```

```
clear all
lambda = 3;
mu = 4;
usuarios = 1000000;
experimentos = 2;
servidores = 3;
resultados = zeros(1,experimentos);

tea = exprnd((1/lambda),1,usuarios);
tds = exprnd((1/mu),1,usuarios);

% tea = [2 7 5 3 2 4 8];
% tds = [1 8 14 2 9 3 6];

p_bloqueo = super_mmkk(tea,tds,servidores,1)
```

```
proba_ocio = 0.4777
prom_usua_sist = 1.4660
usuarios_atendido = 966342
usuarios_bloqueados = 33658
tiempo_medio_tds = 0.2500
p_bloqueo = 0.0337
```

3. SIMULACIÓN DE MMKK

3. Realice la simulación de la cola asignada variando $A = \frac{\lambda}{\mu}$. Considere que los elementos del vector de tiempos entre arribos consecutivos se distribuyen exponencialmente con parámetro: λ, y que los elementos del vector de tiempos de servicio se distribuyen exponencialmente con parámetro: μ. Como resultado principal, realice la gráfica de la medida de desempeño versus A. Verifique que los resultados de simulación coincidan con el valor teórico de la medida de desempeño.

Flujograma de la validación

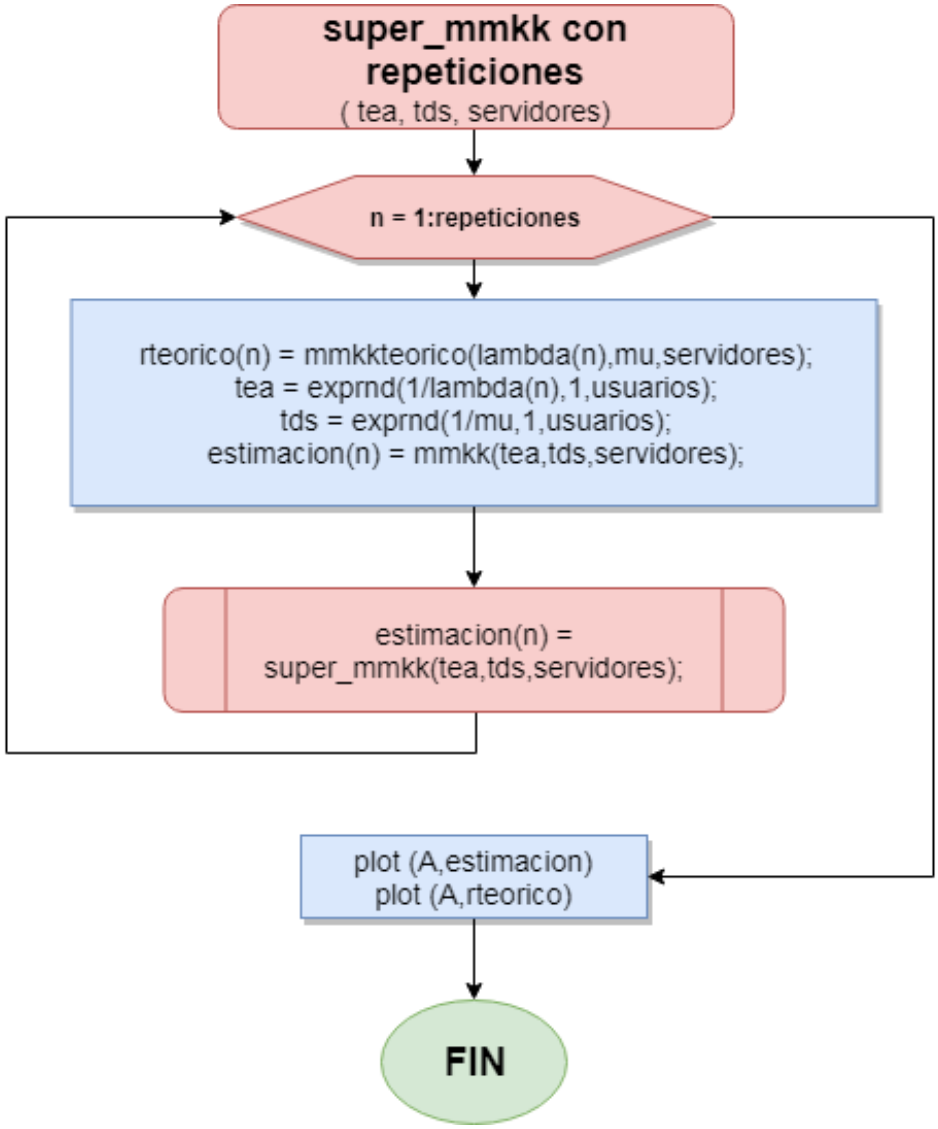

Ilustración 6 - Flujograma de la validación

```
repeticiones = 100;
A = linspace(0.05,9.95,repeticiones);
mu = 3;
lambda = mu.*A;
servidores = 2;
usuarios = 10000;
mmkkA(A, servidores, repeticiones, lambda, mu, usuarios);
```

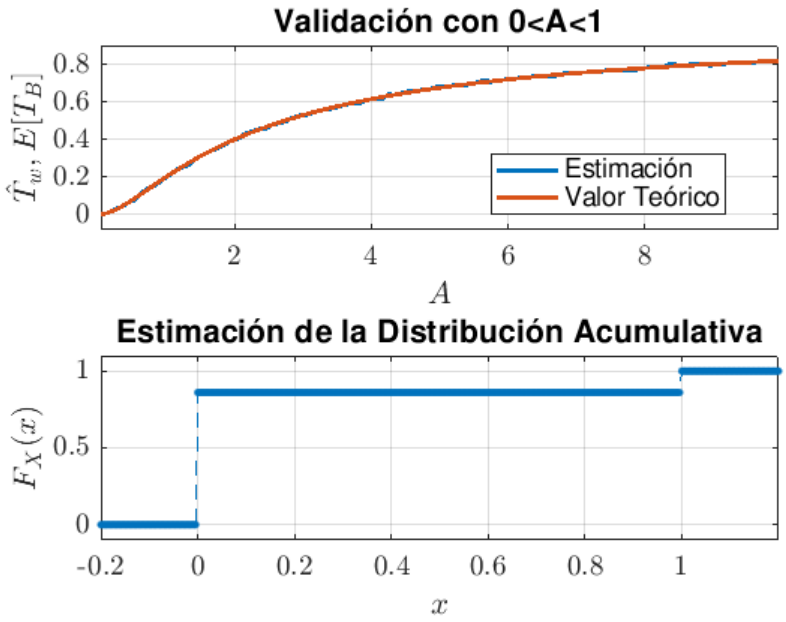

Ilustración 7 - Validación 0<A<1 / Estimación CDF (100 repeticiones)

```
repeticiones = 1000;
A = linspace(0.05,9.95,repeticiones);
mu = 3;
lambda = mu.*A;
servidores = 2;
usuarios = 10000;
mmkkA(A, servidores, repeticiones, lambda, mu, usuarios);
```

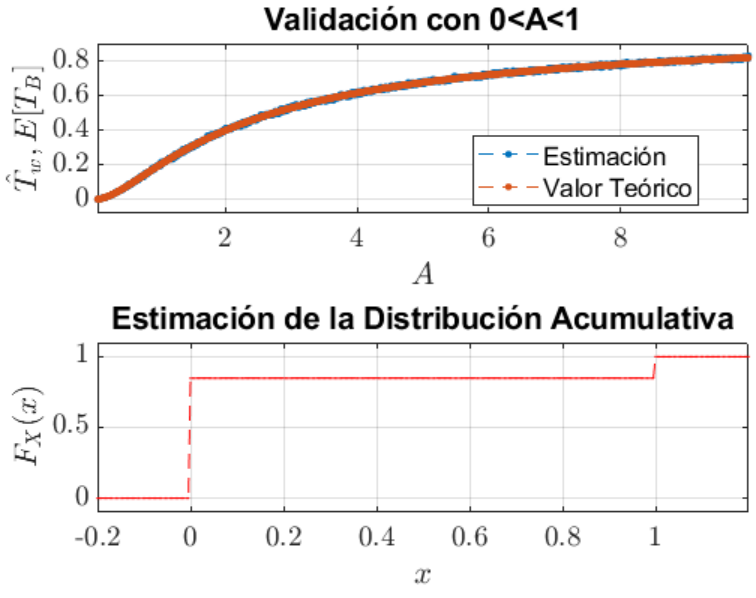

Ilustración 8 - Validación 0<A<1 / Estimación CDF (1000 repeticiones)

4. GENERACIÓN DE FUNCIONES

4. Utilice la función de la tarea 13 (que aún no se ha asignado), para generar cuatro funciones asociadas al conjunto de datos suministrados en clase. Muestre evidencias de la validez de sus funciones (comparación de las gráficas de las distribuciones de los datos y gráficas qqplot, por ejemplo).

```
datosBCOct89Ext = load('BCOct89Ext.TL');
datosBCOct89Ext4= load('BCOct89Ext4.TL');
datosBCpAug89 = load('BCpAug89.TL');
datosBCpOct89 = load('BCpOct89.TL');
diffBCOct89Ext = diff(datosBCOct89Ext(:,1)');
diffBCOct89Ext4= diff(datosBCOct89Ext4(:,1)');
diffBCpAug89= diff(datosBCpAug89(:,1)');
diffBCpOct89= diff(datosBCpOct89(:,1)');


pdfcdfcontinua(diffBCOct89Ext,0,0.005);
```

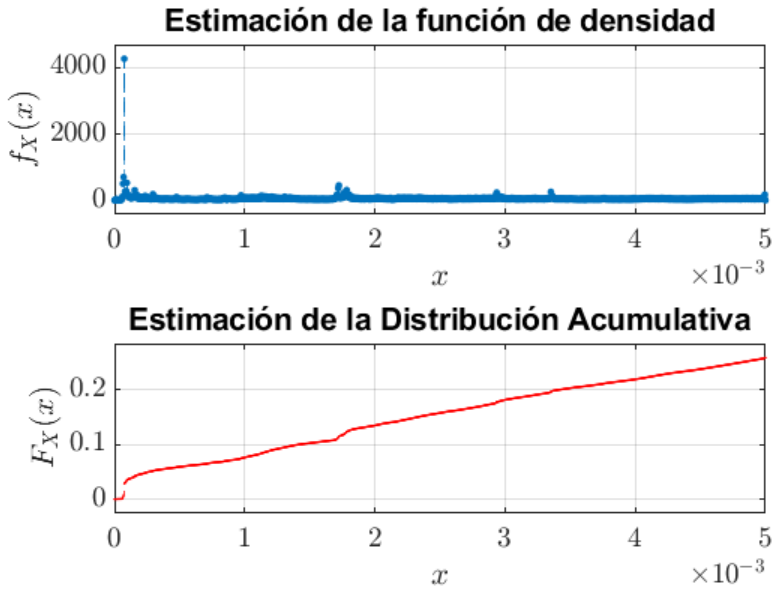

Ilustración 9 Estimación PDF / CDF diffBCOct89Ext

```
figure
pdfcdfcontinua(diffBCOct89Ext4,0,0.005);
```

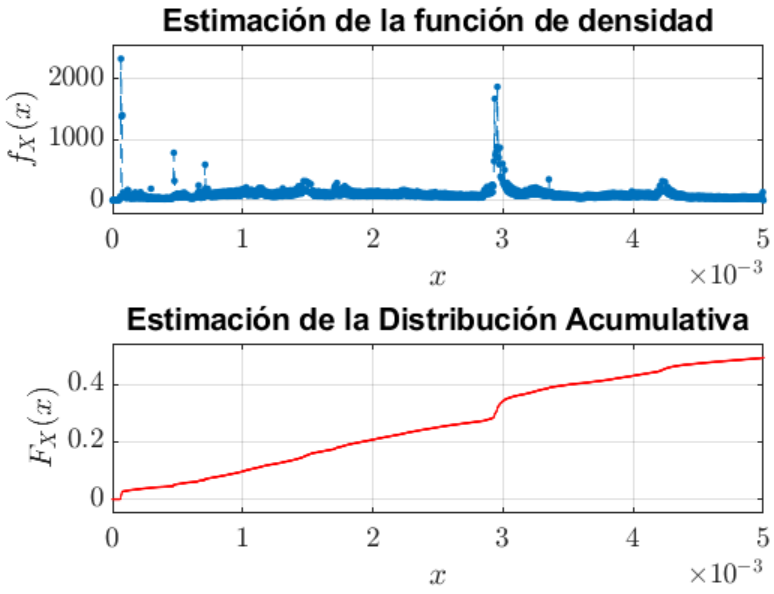

Ilustración 10 - Estimación PDF / CDF diffBCOct89Ext4

```
figure
pdfcdfcontinua(diffBCpAug89,0,0.005);
```

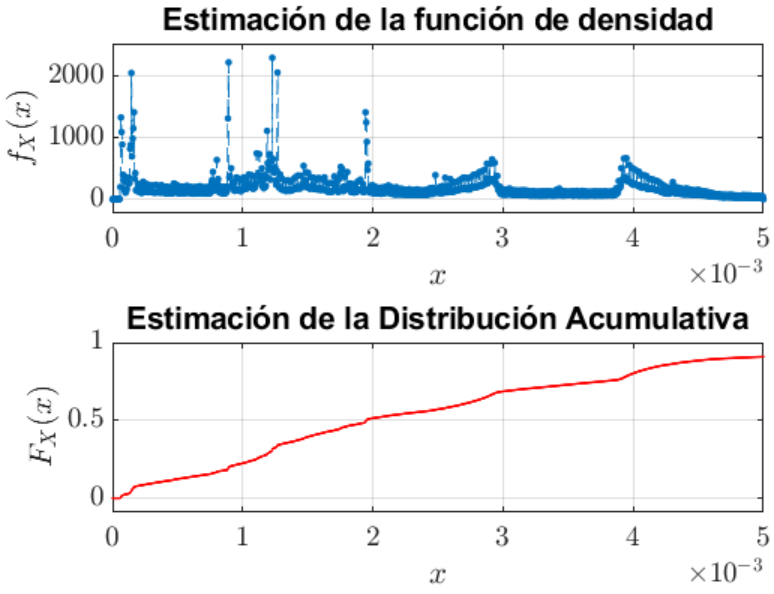

Ilustración 11 - Estimación PDF / CDF diffBCpAug89

```
figure
pdfcdfcontinua(diffBCpOct89,0,0.005);
```

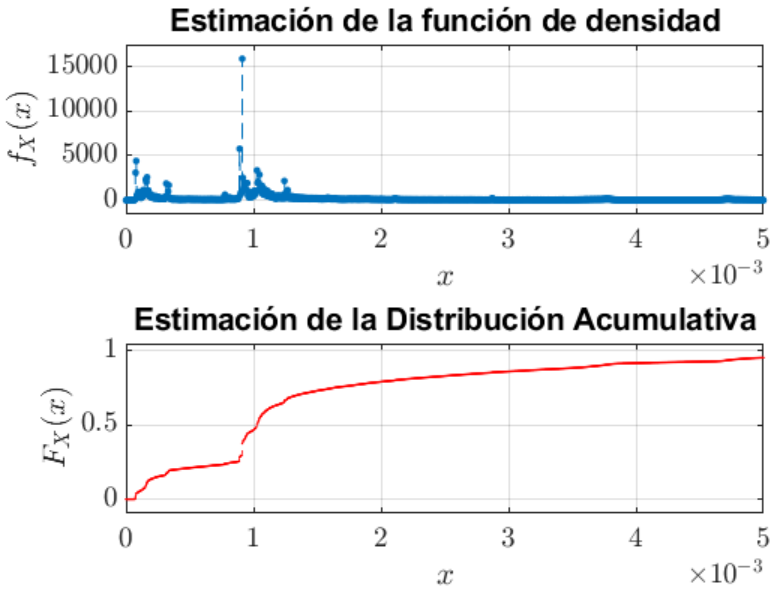

Ilustración 12 - Estimación PDF / CDF diffBCpAug89

```
superfuncion(diffBCOct89Ext,'FuncionBCOct89Ext');
superfuncion(diffBCOct89Ext4,'FuncionBCOct89Ext4');
superfuncion(diffBCpAug89,'FuncionBCpAug89');
superfuncion(diffBCpOct89,'FuncionBCpOct89');
```

QQplot de la función BCOct89Ext

```
figure;
qqplot(diffBCOct89Ext,FuncionBCOct89Ext(1,1000000));
```

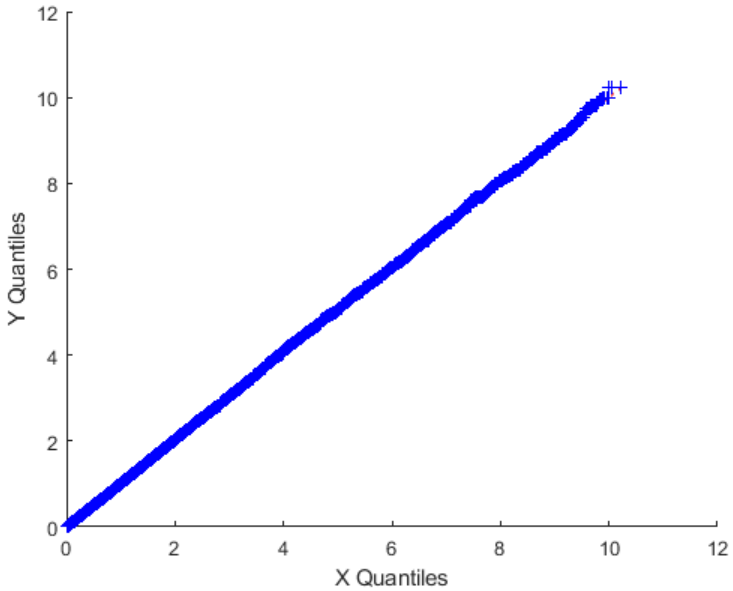

Ilustración 13 - QQplot BCOct89Ext

El ajuste en la función BCOct89Ext va hasta 10.

QQplot de la función BCOct89Ext4

```
figure;
qqplot(diffBCOct89Ext4,FuncionBCOct89Ext4(1,1000000));
```

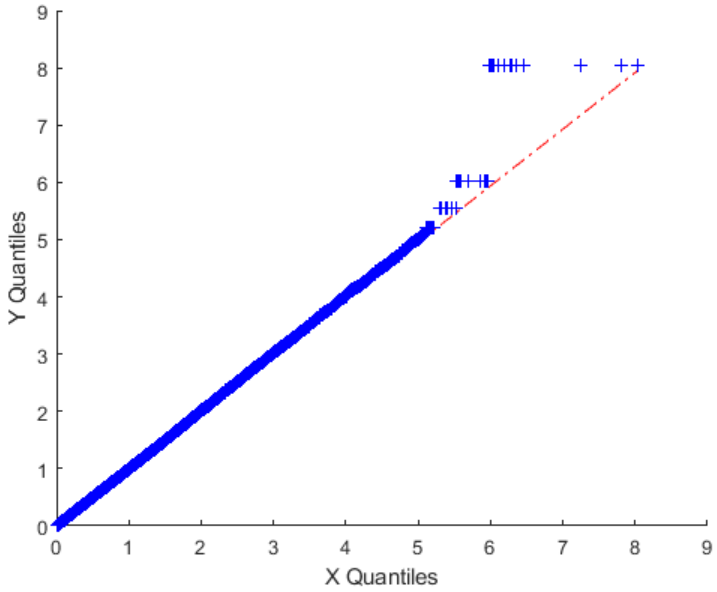

Ilustración 14 - QQplot de la función BCOct89Ext4

```

```

QQplot de la función BCpAug89

```
figure;
qqplot(diffBCpAug89,FuncionBCpAug89(1,1000000));
```

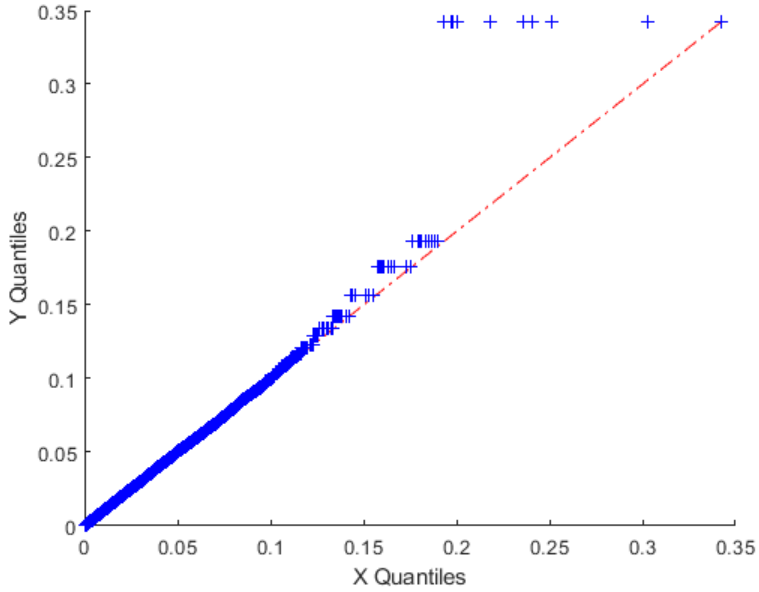

Ilustración 15 - QQplot de la función BCpAug89

QQplot de la función BCpOct89

```
figure;
qqplot(diffBCpOct89,FuncionBCpOct89(1,1000000));
```

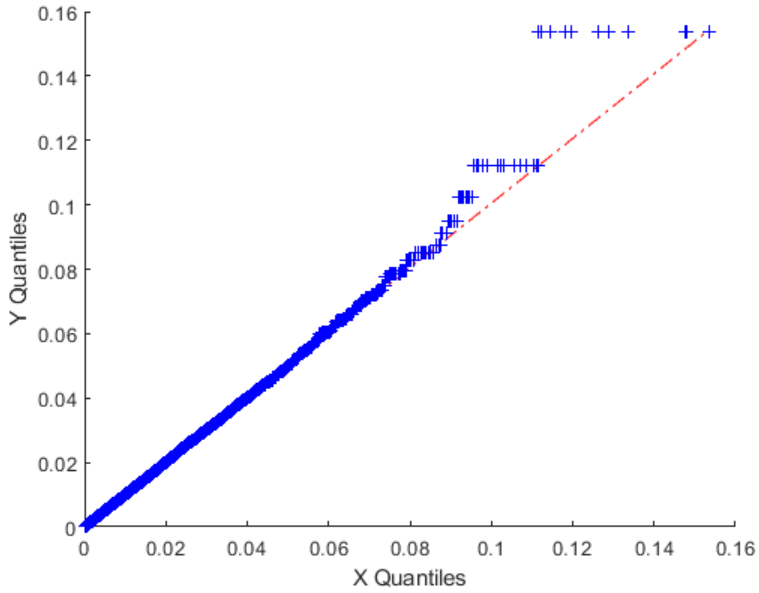

Ilustración 16 - QQplot de la función BCpOct89

5. SIMULACIÓN CON FUNCIONES ASOCIADAS

5. Realice la simulación de la cola asignada variando . Considere que los elementos del vector de tiempos entre arribos consecutivos se obtienen con cada una de las cuatro funciones de la cuarta actividad (para

cada función considere que el parámetro: λ, se estima con el inverso multiplicativo de la media aritmética de los elementos del vector generado con cada una de las cuatro funciones:), y que los elementos del vector de tiempos de servicio se distribuyen exponencialmente con parámetro: μ. Como resultado principal, realice las cuatro gráficas de la medida de desempeño versus A. Compare los resultados de simulación con el valor teórico de la medida de desempeño.

Simulación con función BCOct89Ext

```
datos = FuncionBCOct89Ext(1,1000000);
lambda = 1/mean(datos)
```

```
lambda = 8.1217
```

```
mumin = 8;
mumax = 20;
incremento = 0.5;
servidores = 2;

mu = mumin:incremento:mumax;
usuarios = 1000000;
tiemposentrearribos = zeros(1,usuarios);
tiempomedio = zeros(1,numel(mu));
tic;
for n = 1:numel(mu)
    tiemposentrearribos = FuncionBCOct89Ext(1,usuarios);
    tds = exprnd (1/mu(n),1,usuarios);
    tiempomedio(n) = mmkk(tiemposentrearribos', tds,servidores);
end
toc;
```

```
Elapsed time is 1.391159 seconds.
```

```
mu = linspace(mumin,mumax,1000);
latexplot(mu,1./(mu-lambda)-1./mu,...
    'Tiempos Promedios de Espera','$\mu$',...
    '$\overline{P_B}$');
hold on;
mu = mumin:incremento:mumax;
superstem(mu,tiempomedio);
hold off;
legend('Exponencial','BCOct89Ext4','Interpreter','latex');

xlim([7.50 20.50])
ylim([-0.010 0.803])
```

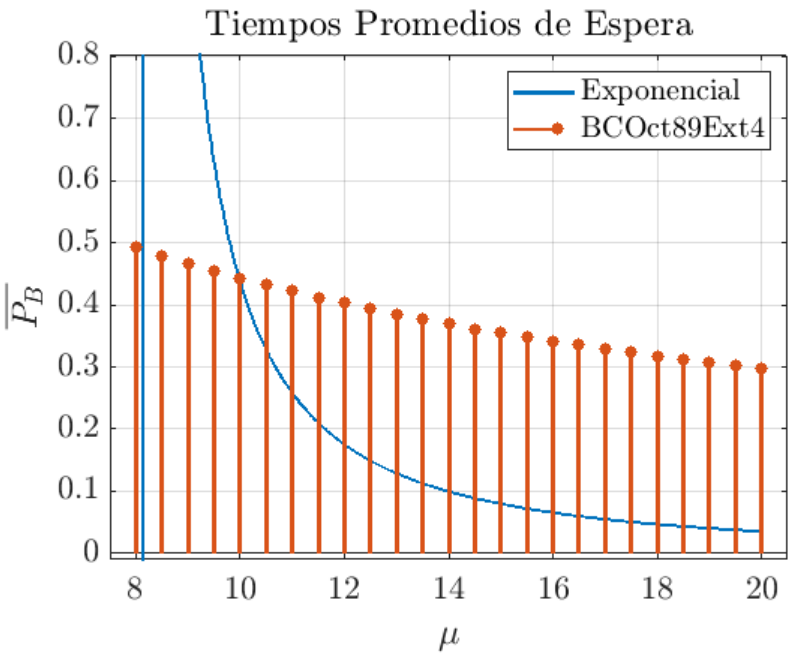

Ilustración 17 - Tiempos promedio de espera

Simulación con función BCOct89Ext4

```
datos = FuncionBCOct89Ext4(1,1000000);
lambda = 1/mean(datos)
```

```
lambda = 13.2540
```

```
mumin = 13;
mumax = 23;
incremento = 0.5;
servidores = 2;

mu = mumin:incremento:mumax;
usuarios = 1000000;
tiemposentrearribos = zeros(1,usuarios);
tiempomedio = zeros(1,numel(mu));
tic;
for n = 1:numel(mu)
    tiemposentrearribos = FuncionBCOct89Ext4(1,usuarios);
    tds = exprnd (1/mu(n),1,usuarios);
    tiempomedio(n) = mmkk(tiemposentrearribos', tds,servidores);
end
toc;
```

```
Elapsed time is 1.580839 seconds.
```

```
mu = linspace(mumin,mumax,1000);
superplot(mu,1./(mu-lambda)-1./mu,...
    'Tiempos Promedios de Espera','$\mu$',...
```

```
    '$\overline{P_B}$');
hold on;
mu = mumin:incremento:mumax;
superstem(mu,tiempomedio);
hold off;
legend('Exponencial','BCOct89Ext4','Interpreter','latex');

xlim([11.81 21.81])
ylim([0.465 0.577])

xlim([12.96 23.15])
ylim([0 1.096])
```

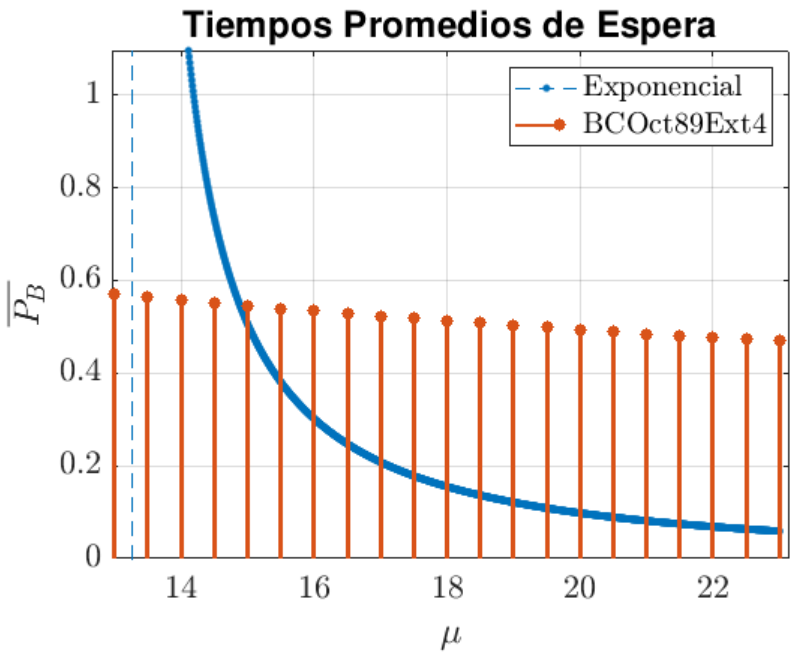

Ilustración 18 - Tiempos promedios de espera FuncionBCOct89Ext4

Simulación con función BCpAug89

```
datos = FuncionBCpAug89(1,1000000);
lambda = 1/mean(datos)
```

```
lambda = 318.0105
```

```
mumin = 318;
mumax = 340;
incremento = 1;
servidores = 2;

mu = mumin:incremento:mumax;
usuarios = 1000000;
tiemposentrearribos = zeros(1,usuarios);
```

```
tiempomedio = zeros(1,numel(mu));
tic;
for n = 1:numel(mu)
    tiemposentrearribos = FuncionBCpAug89(1,usuarios);
    tds = exprnd (1/mu(n),1,usuarios);
    tiempomedio(n) = mmkk(tiemposentrearribos', tds,servidores);
end
toc;
```

```
Elapsed time is 1.269140 seconds.
```

```
mu = linspace(mumin,mumax,1000);
latexplot(mu,1./(mu-lambda)-1./mu,...
    'Tiempos Promedios de Espera','$\mu$',...
    '$\overline{P_B}$');
hold on;
mu = mumin:incremento:mumax;
superstem(mu,tiempomedio);
hold off;
legend('Exponencial','BCpAug89','Interpreter','latex');

xlim([317.4 340.4])
ylim([-0.022 0.968])
```

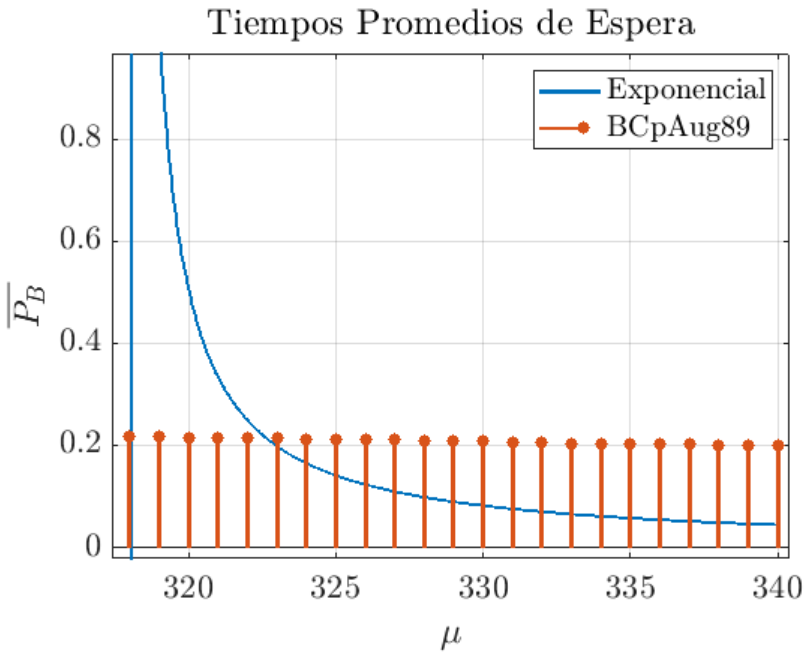

Ilustración 19 - tiempos promedio de espera BCpAug89

Simulación con función BCpOct89

```
datos = FuncionBCpOct89(1,1000000);
lambda = 1/mean(datos)
```

```
lambda = 565.8700
```

```
mumin = 568;
mumax = 600;
incremento = 1;
servidores = 2;

mu = mumin:incremento:mumax;
usuarios = 1000000;
tiemposentrearribos = zeros(1,usuarios);
tiempomedio = zeros(1,numel(mu));
tic;
for n = 1:numel(mu)
    tiemposentrearribos = FuncionBCpOct89(1,usuarios);
    tds = exprnd (1/mu(n),1,usuarios);
    tiempomedio(n) = mmkk(tiemposentrearribos', tds,servidores);
end
toc;
```

```
Elapsed time is 1.829607 seconds.
```

```
mu = linspace(mumin,mumax,1000);
latexplot(mu,1./(mu-lambda)-1./mu,...
    'Tiempos Promedios de Espera','$\mu$',...
    '$\overline{P_B}$');
hold on;
mu = mumin:incremento:mumax;
superstem(mu,tiempomedio);
hold off;
legend('Exponencial','BCpOct89','Interpreter','latex');

xlim([567.5 600.5])
ylim([0.027 0.351])
```

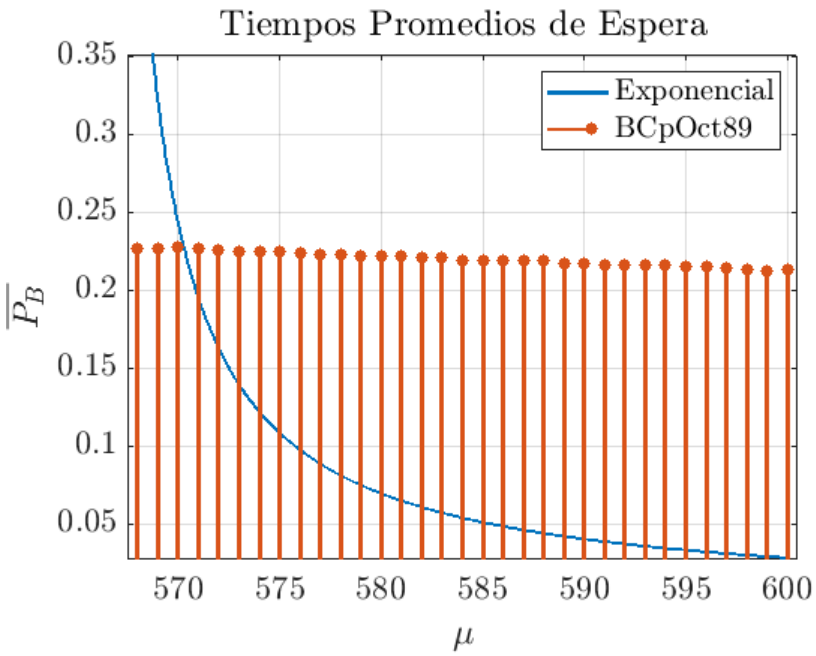

Ilustración 20 - tiempos promedio de espera BCpOct89

6. ESCENARIO REAL

Archivo de datos

	A	B	C	D	E	F	
1	daFECLLAHORA	daHORA	Fecha	Anexo	Nombre Empleado	Duración Real	
2	03/06/2019 09:18:14 tt	09:18:14	03-06-2019	1108	ADMISIONES	00:00:46	365671
3	03/06/2019 10:42:30 tt	10:42:30	03-06-2019	1108	ADMISIONES	00:00:30	771401
4	03/06/2019 11:45:09 tt	11:45:09	03-06-2019	1108	ADMISIONES	00:00:51	315700
5	03/06/2019 11:47:12 tt	11:47:12	03-06-2019	1108	ADMISIONES	00:00:48	315700
6	04/06/2019 07:57:18 tt	07:57:18	04-06-2019	1108	ADMISIONES	00:01:42	305816
7	04/06/2019 07:59:11 tt	07:59:11	04-06-2019	1108	ADMISIONES	00:00:49	305816
8	04/06/2019 08:08:27 tt	08:08:27	04-06-2019	1108	ADMISIONES	00:00:33	683186
9	04/06/2019 08:09:27 tt	08:09:27	04-06-2019	1108	ADMISIONES	00:00:33	683186
10	04/06/2019 08:09:56 tt	08:09:56	04-06-2019	1108	ADMISIONES	00:00:04	771401
11	04/06/2019 08:10:11 tt	08:10:11	04-06-2019	1108	ADMISIONES	00:00:49	319608
12	04/06/2019 08:12:28 tt	08:12:28	04-06-2019	1108	ADMISIONES	00:00:32	319576
13	04/06/2019 08:12:30 tt	08:12:30	04-06-2019	1108	ADMISIONES	00:00:30	661650
14	04/06/2019 08:12:30 tt	08:12:30	04-06-2019	1108	ADMISIONES	00:00:30	683186

Ilustración 21 - Datos del experimento

Configurar las opciones de importación

```
opts = spreadsheetImportOptions("NumVariables", 2);
opts.Sheet = "Hoja1";
opts.DataRange = "A1:B294";
```

```
opts.VariableNames = ["tea", "tds"];
opts.SelectedVariableNames = ["tea", "tds"];
opts.VariableTypes = ["double", "double"];
tbl = readtable("teatdsAdmi.xlsx", opts, "UseExcel", false);
```

Convertir a tipo de salida

```
tea = tbl.tea;
tds = tbl.tds;
clear opts tbl
tea = tea(:)';
tds = tds(:)';
servidores = 2;
tic
mmkk( tea, tds, servidores,2)
```

```
bloqueo = 1×294
     0     0     0     0     0     0     0     1     0     0     0     0 ···
```

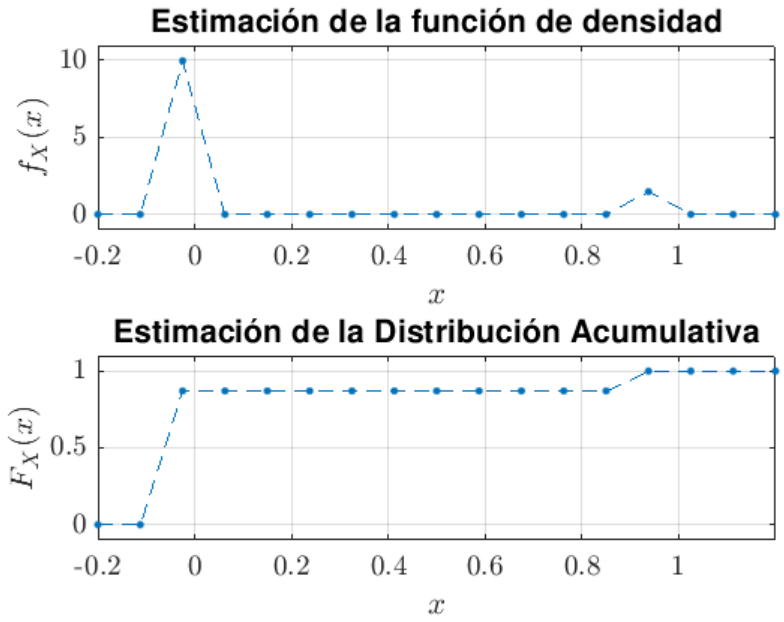

Ilustración 22 - Estimación CDF / PDF

```
ans = 0.1293
```

```
toc
```

```
Elapsed time is 0.211057 seconds.
```

```
tic
super_mmkk( tea, tds, servidores,1)
```

```
proba_ocio = 0.2959
prom_usua_sist = 1.1429
usuarios_atendido = 256
usuarios_bloqueados = 38
```

```
tiempo_medio_tds = 72.9558
ans = 0.1293
```

```
toc
```

```
Elapsed time is 0.029805 seconds.
```

```
figure();
pdfcdfcontinua(tea);
```

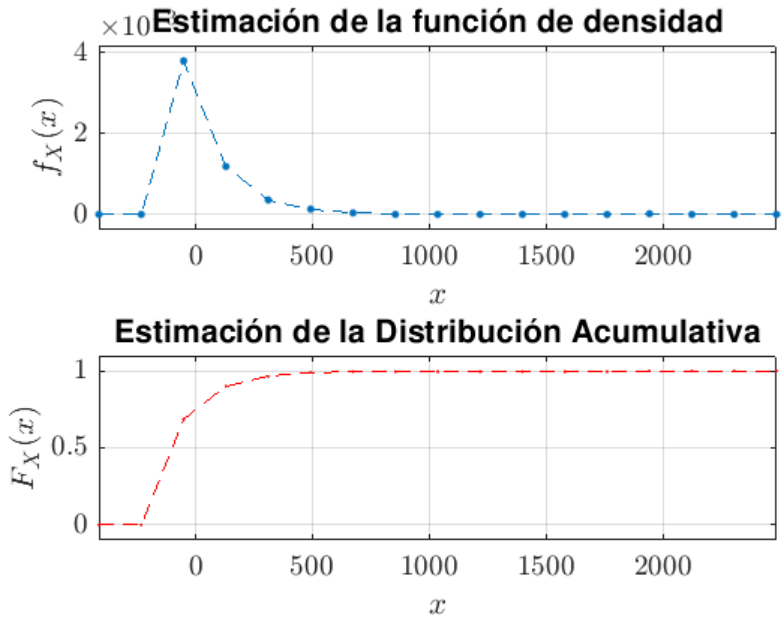

Ilustración 23 - Estimación PDF / CDF

```
figure();
qqplot(tea);
```

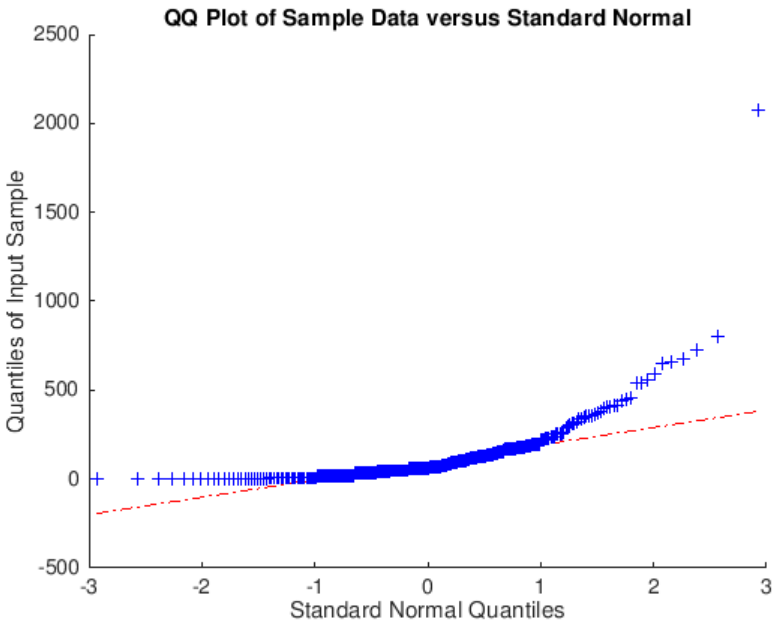

Ilustración 24 - QQPlot

CONCLUSIONES

Según los resultados del estudio se puede inferir la probabilidad de bloqueo como el punto de eficiencia del sistema de cola, ya que alcanzar el mínimo valor positivo que converge en cero, es evidencia de un sistema que aprovecha al máximo los servidores y tiene una mínima probabilidad de ocio de sus servidores.

En lo referente al diseño de la simulación de la cola, se evidencia que el algoritmo realiza una exigencia mayor cuando se encuentran ocupados los servidores, es por ello que en cuanto más valor tenía la probabilidad de bloqueo, mayor seria el tiempo que el algoritmo tomaría en terminar y simular los valores sintéticos de la cola.

En el tipo de cola MMKK hay un número mínimo de servidores donde la probabilidad de bloqueo es cero, por eso es necesario hacer este tipo de análisis en un sistema con estas condiciones, dado que tener el mínimo de servidores necesarios es igual que tener muchos más servidores y serían innecesarios y mucho más costoso.

La probabilidad de ocio de los servidores es un indicador que es necesario a tener en cuenta, ya que llegar a una probabilidad de bloqueo cercana a 0 no implica que la cola sea eficiente, por ejemplo, si tenemos una probabilidad de bloqueo 0 pero la probabilidad de ocio de nuestro servidores esta por encima de 0.5, existirá una evidente subutilización de los servidores, lo que dependiendo del sistema generara una mal adecuado uso de recursos.

BIBLIOGRAFÍA

[1]https://www.tlmat.unican.es/siteadmin/submaterials/923.pdf7

[2] http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/garduno_a_f/capitulo1.pdf

FUNCIONES

Código de función teórica:

```
function B = mmkkteorico(lambda,mu,servidores)
    r = lambda / mu;
    B=1;
    for k=1:servidores
        B=((r*B)/k)/(1+r*B/k);
    end
end
```

Código MMKK

```
% Comentada para hacer simulaciones....
% function [p_bloqueo] = mmkk( tea, tds, servidores, mostrar )
%     switch nargin
%         case 3
%             mostrar = 0;
%     end
%     usuarios = numel(tea);
%     reloj = cumsum(tea);
%     salida = zeros(1,numel(tea));
%     bloqueo = zeros(1,numel(tea));
%     quienatiende = zeros(1,numel(tea));
%     quienatiende(1,1)  = 1;
%     monitor = zeros(1,servidores);
%
%     for n = 2:numel(tea)
%
%         atendido = 0;
%         for s = 1:servidores
%
%             if monitor(1,s) <= (reloj(1,n)) && atendido == 0
%                 monitor(1,s) = reloj(1,n) + tds(1,n);
%                 salida(1,n) = reloj(1,n) + tds(1,n);
%                 bloqueo(1,n) = 0 ;
%                 quienatiende(1,n)  = s;
%                 atendido = 1;
%             end
%         end
%
%         if atendido == 1
```

```
%             continue
%         end
%
%         if atendido == 0
%             salida(1,n) = salida(1,n-1);
%             bloqueo(1,n) = 1 ;
%             quienatiende(1,n)  = 0;
%         end
%
%
%     end
%
%     p_bloqueo = (sum(bloqueo))/numel(bloqueo);
%     if mostrar == 1
%         tea
%         tds
%         reloj
%         salida
%         bloqueo
%         quienatiende
%         p_bloqueo = (sum(bloqueo))/numel(bloqueo)
%     end
%     if mostrar == 2
%
%         bloqueo
%         pdfcdfcontinua(bloqueo)
%     end
%
% end
```

Código PDFCDFCONTINUA

```
%   Elaborada por Hans López, hanslop@gmail.com

function [] = pdfcdfcontinua(datos,minimo,maximo)
    switch nargin
        case 1
            rango = max(datos(:)')-min(datos(:)');
            minimo = min(datos(:)')-0.2*rango;
            maximo = max(datos(:)')+0.2*rango;
    end
    datos = datos(:)';
    total = numel(datos);
    particiones = floor(sqrt(total));
    x = linspace(minimo,maximo,particiones);

    fr = histc(datos,x)/total;
```

```
    cdf = cumsum(fr);
    subplot(2,1,2);
    superplot(x,cdf,...
    'Estimación de la Distribución Acumulativa',...
    '$x$','$F_X(x)$',3,'r');
    subplot(2,1,1);

    pdf = fr/(x(2)-x(1));

    superplot(x,pdf,...
    'Estimación de la función de densidad',...
    '$x$','$f_X(x)$');
end
```

Código SuplerPlot

```
%   Elaborada por Hans López, hanslop@gmail.com

function [] = superplot(x,y,titulo,ejex,ejey,grosor,colorlinea)
%   Utiliza la funci□n plot para realizar la gr□fica de f(x). SUPERPLOT
%   cambia el fondo de la figura a color blanco y el tama□o de las fuentes.
%   Adicionalmente ajusta de forma autom□tica los l□mites de las gr□ficas.
%
%   La funci□n permite agregar el t□tulo de la gr□fica, y el nombre de los
%   ejes como par□metros de entrada.
%
%   Ejemplo:
%
%   x = linspace(-5,5,1000);
%   SUPERPLOT(x,sinc(x),'Funci□n Seno Cardinal','$x$','sinc($x$)',10);
%
%   Funci□n SUPERPLOT, Versi□n 4.0, 23 de marzo de 2019
%   - se agreg□un s□timo par□metro para modificar el color de la l□nea.
%   Versi□n 3.0, 18 de mayo de 2017
%   - se modific□el interprete de los ejes a LaTeX. El interprete del
%   t□tulo no se modific□
%   Versi□n 2.0, 2 de diciembre de 2016:
%   - Se modific□el tama□o del punto a 10, para mejorar la visualizaci□n.
%   - El tama□o del punto se puede modificar indic□ndolo en el sexto
%   par□metro, como muestra el ejemplo.
%   Versi□n 1.0, 29 de septiembre de 2015
%   Elaborada por Hans López, hanslop@gmail.com
%
switch nargin
    case 0
        disp('Faltan argumentos de entrada');
        return;
    case 1
```

```
        plot(x,'.--','MarkerSize',10);
        grid on;
        set(gca,'FontSize',14);
        set(gcf,'Color','white');
        axis([1 length(x) min(x)-0.1*(max(x)-min(x)) max(x)+0.1*(max(x)-min(x))]);
        return;
    case 2
        plot(x,y,'.--','MarkerSize',10);
    case 3
        plot(x,y,'.--','MarkerSize',10);
        title(titulo,'FontSize',16);
    case 4
        plot(x,y,'.--','MarkerSize',10);
        title(titulo,'FontSize',16);
        xlabel(ejex,'FontSize',16,'Interpreter','latex');
    case 5
        plot(x,y,'.--','MarkerSize',10);
        title(titulo,'FontSize',16);
        xlabel(ejex,'FontSize',16,'Interpreter','latex');
        ylabel(ejey,'FontSize',16,'Interpreter','latex');
    case 6
        plot(x,y,'.--','MarkerSize',grosor);
        title(titulo,'FontSize',16);
        xlabel(ejex,'FontSize',16,'Interpreter','latex');
        ylabel(ejey,'FontSize',16,'Interpreter','latex');
    case 7
        plot(x,y,'.--','Color',colorlinea,'MarkerSize',grosor);
        title(titulo,'FontSize',16);
        xlabel(ejex,'FontSize',16,'Interpreter','latex');
        ylabel(ejey,'FontSize',16,'Interpreter','latex');
end
grid on;
set(gca,'FontSize',14);
set(gca,'TickLabelInterpreter','latex');
set(gcf,'Color','white');
axis([min(x) max(x) min(y)-0.1*(max(y)-min(y)) max(y)+0.1*(max(y)-min(y))]);
end
```

Código Función MMKK con repeticiones

```
function mmkkA(A, servidores, repeticiones, lambda, mu, usuarios)
estimacion = zeros(1,repeticiones);

parfor n = 1:repeticiones
    rteorico(n) = mmkkteorico(lambda(n),mu,servidores);
    tea = exprnd(1/lambda(n),1,usuarios);
    tds = exprnd(1/mu,1,usuarios);
    estimacion(n) = mmkk(tea,tds,servidores);
```

```
end
latexplot(A,estimacion,...
    'Validación con 0<A<1',...
    '$A$','$E[T_B]$');
hold on;
latexplot(A,rteorico,...
    'Validación con 0<A<1',...
    '$A$','$\hat{T}_w,E[T_B]$');
hold off;
legend('Estimación','Valor Teórico','Location', 'Best');
end
```

Código superfuncion

```
function [] = superfuncion(datos,nombre)
switch nargin
    case 0
        disp('Falta el vector de datos');
        return;
    case 1
        nombre = 'aleatorio';
end

p = linspace(0,1,100001);
datos = datos(:)';
numero = quantile(datos,p);

encabezado = ['function y = ',nombre,'(m,n)'];
cuerpo = 'numero = [';
final = [']; ','y = numero(randi(numel(numero),m,n));','end'];

fid = fopen([nombre,'.m'],'w');

fprintf(fid,'%s\n%s\n%10.8f\n%s\n',encabezado,cuerpo,numero,final);
fclose(fid);
disp([nombre,'.m codificado. Revisa tu carpeta de MATLAB, por favor.']);
disp(['Luego puedes ejecutar: ',nombre,'(m,n).']);
end
```

Función supermmkk

```
function [p_bloqueo] = super_mmkk( tea, tds, servidores, mostrar )
    switch nargin
        case 3
            mostrar = 0;
    end
    usuarios = numel(tea);
    reloj = cumsum(tea);
```

```
salida = zeros(1,numel(tea));
bloqueo = zeros(1,numel(tea));
quienatiende = zeros(1,numel(tea));
quienatiende(1,1)  = 1;
monitor = zeros(1,servidores);
servi_ocio = zeros(1,numel(tea));
porce_ocio = zeros(1,numel(tea));
usuarios_en_sistema = zeros(1,numel(tea));
atendidos1 = zeros(1,numel(tea));
atendidos1(1,1)  = 1;

for n = 2:numel(tea)

    atendido = 0;
    for s = 1:servidores

        if monitor(1,s) <= (reloj(1,n)) && atendido == 0
            monitor(1,s) = reloj(1,n) + tds(1,n);
            salida(1,n) = reloj(1,n) + tds(1,n);
            bloqueo(1,n) = 0 ;
            quienatiende(1,n)  = s;
            atendido = 1;
            servi_ocio(1,n) = servidores - s;
            porce_ocio(1,n) = servi_ocio(1,n) / servidores;
            usuarios_en_sistema(1,n) = servidores - servi_ocio(1,n);
            atendidos1(1,n) = 1;
        end
    end

    if atendido == 1
        continue
    end

    if atendido == 0
        salida(1,n) = salida(1,n-1);
        bloqueo(1,n) = 1 ;
        quienatiende(1,n)  = 0;
    end


end

p_bloqueo = (sum(bloqueo))/numel(bloqueo); %probabilidad de bloqueo

if mostrar == 1
    tea;
    tds;
    reloj;
```

```
        %inicios
        salida;
        bloqueo; %bloqueados
        quienatiende; %Cual servidor atiende
        servi_ocio; % Cuantos servidores estan en ocio en el momento
        porce_ocio; % Porcentaje de ocupacion en el momento
        proba_ocio = (sum(porce_ocio))/numel(porce_ocio) %Probabilidad de ocio en el
sistema
        usuarios_en_sistema; % Usuarios simultaneos en el momento
        prom_usua_sist = (sum(usuarios_en_sistema))/numel(usuarios_en_sistema) %
Promedio de usuarios simultaneos en el sistema
        usuarios_atendido = sum(atendidos1) %total de usuarios atendidos
        usuarios_bloqueados = sum(bloqueo) %total de usuarios bloqueados
        tiempo_medio_tds = sum(tds)/numel(tds) %tiempo medio de atención

    end

 end
```