Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

PRÁCTICAS DE MODELADO Y SIMULACIÓN

BOLETÍN DE ENTREGA
Práctica 3: Simulador de colas G/G/k

INTEGRANTES DEL GRUPO:

NOMBRE Y APELLIDOS	CORREO ELECTRÓNICO
Manuel Alejandro De la Torre Ruzafa	
Jairo Peña Iglesias	**Jairopi91@gmail.com**

1. **Implementación de las funciones** *GMLC* **y** *aleatorio*

COMPLETE LOS SIGUIENTES CÓDIGOS

```
function [Z, muestra] = aleatorio(Z, tipo, param1, param2)
[Z,m]=GCLM(Z);
switch tipo

    case 0         % TIPO = 0 -> VA uniforme [0,1]
        muestra=Z/m;

    case 1         % TIPO = 1 -> VA uniforme [param1, param2]
        muestra=Z*(param2-param1)/m + param1;

    case 2         % TIPO = 2 -> VA exponencial lambda=param1
         if param1<=0
         stop('Lambda debe ser positivo')
        else
            muestra=-log(Z/m)/param1;
        end

    case 3         % TIPO = 3 -> Devuelve siempre param1 (VA "degenerada")
        muestra=param1;
end
end
```

```
function  nuevoZ  = GCLM( Z )                                [0.75 ptos]
% Usando Z como muestra previa del generador, crea la nueva muestra.
function  [nuevoZ,m]  = GCLM( Z )

% El GCLM debe usar los parámetros de referencia de Fishman y Moore
a = 48271;
q = 44488;
r = 3399;
m = a*q+r;
if(Z<m)
    nuevoZ = rem(a*Z,m);
else
  display(strcat('Z debe ser menor que el parametro m: ',int2str(m)));
 nuevoZ = Z;
end
end



% nuevoZ = mod(a*Z,(2^31)-1);
```

a. Indique las dos muestras obtenidas para el generador de VA exponencial λ (indique asimismo el λ elegido) [0.5 ptos]

```
>> Z=159753;
>> lambda=5;
>> [Z,muestra]=aleatorio(Z,2,lambda);
>> muestra

muestra =
```

```
    0.1052

>> [Z,muestra]=aleatorio(Z,2,lambda);
>> muestra

muestra =

    0.3334
```

2. Implementación del simulador

COPIE EL CÓDIGO DE SU SIMULADOR (SI REALIZA EL CÁLCULO OPCIONAL DE N INDÍQUELO TAMBIÉN)

```
%% SIMULADOR COLAS G/G/K

listaEV = [];          % Lista vacia al comienzo

t_simulacion = 0.0;    % Reloj de simulación

pasos = 100000;        % Numero de iteraciones del simulador

k = 1;                 % Numero de recursos del sistema

t_muestreoN = 1;       % Segundos entre cada muestra de N

%VARIABLES ALEATORIAS
%PARAMETROS DE SIMULACION
X=2452;          % Semilla Tiempo entre llegadas (1/lambda)
tipoX=2;
param1X=10;      % lambda (tasa de llegada)
param2X=0;

S=9876;          % Semilla Tiempo de servicio (1/mu)
tipoS=2;
param1S=15;      % mu (tasa de servicio)
param2S=0;

% TIPOS DE EVENTOS, CADA UNO UN NUMERO DIFERENTE
SALE = 0;
LLEGA = 1;
SUMAN = 2;

% ESTADO
N = 0;           %N:número de tareas en el sistema (Se inicializa a 0)
fifoTiempos = [];  %Cola FIFO del sistema

% VARIABLES PARA EL CALCULO DE LOS PROMEDIOS DE INTERES
summuestrasT = 0;
nummuestrasT = 0;

summuestrasN = 0;  % Suma de todos los valores de N muestreados
nummuestrasN = 1;  % Numero de muestras de N tomadas (La primera muestra se toma para t_simulacion=0.0)

% PRIMEROS EVENTOS
[X,taux] = aleatorio(X,tipoX,param1X,param2X);
```

```
listaEV = encolarEventoGGK(listaEV, taux, LLEGA, 0); %Programación de evento de llegada
listaEV = encolarEventoGGK(listaEV, t_muestreoN, SUMAN, 0); %Programación de evento de muestreo de N

for i=1:pasos % pasos == numero de eventos que se van a procesar

  % Salto al siguiente evento programado en la lista de eventos
  [listaEV, tiempoEvento, tipo, tiempoLLegada] = sgteEventoGGK(listaEV);

  % Actualizacion del tiempo de simulación.
  t_simulacion = tiempoEvento;



  switch tipo
    case LLEGA % EL EVENTO ACTUAL ES UNA LLEGADA
      N = N+1; % Hay una tarea más en el sistema
      [X,taux] = aleatorio(X,tipoX,param1X,param2X);
      listaEV = encolarEventoGGK(listaEV, t_simulacion + taux, LLEGA, 0);
      if N<=k % Se programa una salida si la tarea pasa a uno de los K recursos directamente
        [S,taux] = aleatorio(S,tipoS,param1S,param2S);
        listaEV = encolarEventoGGK(listaEV, t_simulacion + taux, SALE, t_simulacion);
      else % En caso contrario, la tarea entra en la cola
        fifoTiempos = pushFIFO(fifoTiempos, t_simulacion);
      end
    case SALE % EL EVENTO ACTUAL ES UNA SALIDA
      N = N-1; % Hay una tarea menos en el sistema
      if N>=k % Hay tareas en cola. Se programa la salida de la tarea que entra al recurso
        [fifoTiempos, tentrada] = popFIFO(fifoTiempos); % La tarea sale de la cola FIFO y entra al recurso
        [S,taux] = aleatorio(S,tipoS,param1S,param2S);
        listaEV = encolarEventoGGK(listaEV, t_simulacion + taux, SALE, tentrada);
      end
      summuestrasT = summuestrasT + (t_simulacion - tiempoLLegada);
      nummuestrasT = nummuestrasT + 1;
    case SUMAN % EL EVENTO ACTUAL ES DE MEDICIÓN DE N. Cada "t_muestreoN" u.t. se produce este evento.
      nummuestrasN=nummuestrasN+1;
      summuestrasN=summuestrasN+N;
      listaEV = encolarEventoGGK(listaEV, t_simulacion + t_muestreoN, SUMAN, 0);
  end

end

Tmedio = summuestrasT / nummuestrasT;   % Retardo medio del sistema
Nmedio = summuestrasN / nummuestrasN;   % Numero medio de usuarios en el sistema

% Mostramos los promedios calculador
display('########FIN DE LA SIMULACION########');
display(strcat('>Iteraciones=',num2str(i)));
display(strcat('>summuestrasT=',num2str(summuestrasT)));
display(strcat('>nummuestrasT=',num2str(nummuestrasT)));
display(strcat('>T=',num2str(Tmedio)));
display(strcat('>summuestrasN=',num2str(summuestrasN)));
display(strcat('>nummuestrasN=',num2str(nummuestrasN)));
display(strcat('>N=',num2str(Nmedio)));
```

La siguiente función nos devuelve los valores teóricos de retardo y número medio de usuarios en un sistema dado. Los parámetros de entrada definen el sistema:

```
function [retardo, usuarios] = calculaPromedios( tipoCola, lambda, mu, k )
    switch tipoCola
        case 0 %Sistema M/M/1
            rho = lambda/mu;
            retardo = 1/(mu*(1-rho));
            usuarios = rho/(1-rho);
        case 1 %Sistema M/M/k
            i = lambda/mu;
            rho = i/k;
            sum = 0;
            for n = 0:k-1
                sum = sum+i^n/factorial(n);
            end
            p0 = (i^k/factorial(k)*(1-rho)+sum)^(-1);
            retardo = (k*rho)^(k+1)*p0/(lambda*k*factorial(k)*(1-rho)^2)+1/mu;
            usuarios = k*rho+(k*rho)^(k+1)*p0/(k*factorial(k)*(1-rho)^2);
        case 2 %Sistema M/D/1
            rho = lambda/mu;
            retardo = rho/(2*mu*(1-rho))+1/mu;
            usuarios = rho+rho^2/(2*(1-rho));
    end
    display(strcat('Tiempo medio teórico-->',num2str(retardo)));
    display(strcat('Número medio de eventos teórico-->', num2str(usuarios)));
end
```

a) Elija una configuración M/M/1 (es decir, valores λ y μ) y compruebe que el resultado es coherente. INDIQUE CONFIGURACIÓN ELEGIDA Y RESULTADOS OBTENIDOS [0.75 ptos]

NOTA: Se puede calcular analíticamente T, como:

$$T = \frac{1}{\mu * (1 - \lambda / \mu)}$$

```
Configuración:
λ=15;
μ=20;
pasos=100000;
X=2452;
S=9876;
t_muestreoN=1;

Resultado de la simulación:

>> simGGK
#######FIN DE LA SIMULACION########
>Iteraciones=100000
>summuestrasT=10327.5261
>nummuestrasT=48387
```

```
>T=0.21344
>summuestrasN=10267
>nummuestrasN=3224
>N=3.1846
```

b) Elija una configuración M/M/3 (es decir, valores λ y μ) y compruebe que el resultado es coherente. INDIQUE CONFIGURACIÓN ELEGIDA Y RESULTADOS OBTENIDOS `[0.75 ptos]`
NOTA: Se puede calcular analíticamente T, como:

$$
\begin{aligned}
&I = \lambda/\mu \\
&\rho = I/k \\
&p_0 = \left[\frac{I^k}{k!(1-\rho)} + \sum_{n=0}^{k-1} \frac{I^n}{n!}\right]^{-1} \\
&T = 1/\mu + \frac{I^k}{\mu k!(1-\rho)^2} p_0
\end{aligned}
$$

```
Configuración:
λ=15;
μ=20;
pasos=100000;
X=2452;
S=9876;
t_muestreoN=1;

Resultado de la simulación:
>> simGGK
#######FIN DE LA SIMULACION########
>Iteraciones=100000
>summuestrasT=2476.5003
>nummuestrasT=48388
>T=0.05118
>summuestrasN=2404
>nummuestrasN=3224
>N=0.74566
```

c) Elija una configuración M/D/1 (es decir, valores λ y s –s es constante-) y compruebe que el resultado es coherente. INDIQUE CONFIGURACIÓN ELEGIDA Y RESULTADOS OBTENIDOS `[0.5 pto]`

NOTA: Se puede calcular analíticamente T, como:

$$
\begin{aligned}
&\rho = \frac{\lambda}{\mu} \\
&T = \frac{1}{\mu} + \frac{\rho}{\mu(1-\rho)}
\end{aligned}
$$

```
Configuración:
λ=15;
s=1/20;
pasos=100000;
X=2452;
S=9876;
t_muestreoN=1;

Resultado de la simulación:
>> simGGK
#######FIN DE LA SIMULACION########
>Iteraciones=100000
>summuestrasT=6117.9646
>nummuestrasT=48388
>T=0.12644
>summuestrasN=5995
>nummuestrasN=3224
>N=1.8595
```

d) Si ha realizado el ejercicio opcional compruebe que se verifica

$N = T * \lambda$ (LEY DE LITTLE)

En todos los escenarios anteriores.
INDIQUE RESULTADO TEÓRICO PARA N Y RESULTADO POR SIMULACIÓN
[2 ptos CODIGO Y RESULTADO OK]

Para cada caso, los valores teóricos aplicando la Ley de Little son:

Caso a): N = 0.2*15 = 3, [En simulación: N=3.1846]

```
Resultado analítico:
>> calculaPromedios(0,15,20);
Tiempo de respuesta medio teórico-->0.2
Número medio de tareas teórico-->3
```

Caso b): N = 0.051*15 = 0.765, [En simulación: N=0.74566]

```
Resultado analítico:
>> calculaPromedios(1,15,20,3);
Tiempo de respuesta medio teórico-->0.051
Número medio de tareas teórico-->0.765
```

Caso c): N = 1.875*15 = 1.875, [En simulación: N=1.8595]

```
Resultado analítico:
>> calculaPromedios(2,15,20);
Tiempo de respuesta medio teórico-->0.125
Número medio de tareas teórico-->1.875
```

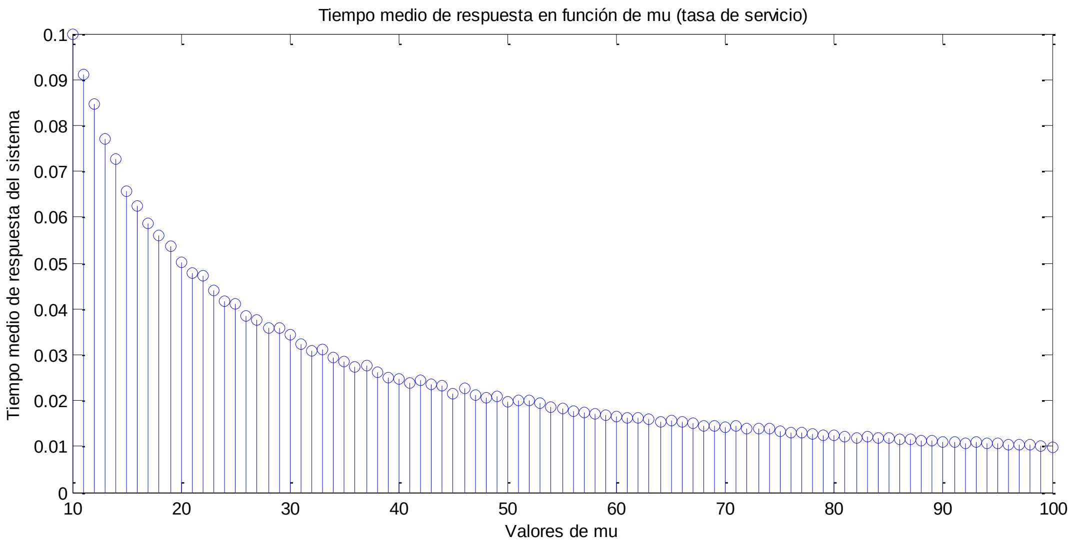
e) G/G/k, con X~U[2,5], S~U[1,4], y k=1..5. Indique el promedio de T obtenido en la memoria [2 pto]

```
K=1 => T=2.7536
K=2 => T=2.4987
```

```
K=3 => T=2.4988
K=4 => T=2.4998
K=5 => T=2.4995
```

f) G/M/5, con X=40 (constante) S~exp(λ), con λ=10…100 (dibuje en una gráfica de matlab el resultado) `[2 pto]`

Para 10000 iteraciones:



Para 100000 iteraciones: