# Trabajo práctico para regularizar

## Maximiliano Tulian 1

1) E-mail: [k39446@frro.utn.edu.ar](mailto:k39446@frro.utn.edu.ar), Legajo: 39446

Universidad Tecnológica Nacional.

Zevallos 1341 Rosario, Santa Fe, Argentina.

## Presentación

Se debe realizar el algoritmo de una simulación para un modelo m/m/1.

Se asume que:

1. Los tiempos entre arribo son probabilísticos con distribución exponencial.
2. El tiempo de servicio es probabilístico con distribución exponencial.
3. Se posee una sola línea de espera.
4. La disciplina de colas es FIFO.
5. La población es infinita.
6. No se acepta prioridad.

Otros:

1. El modelo de programación es orientado a objetos.
2. La media de tiempo entre arribos y tiempo de servicios están fijadas en 0.5 y 0,25, con la opción de ser modificadas en la rutina de inicialización.
3. El tiempo final de simulación está fijado en diez mil unidades del reloj, con la opción de ser modificado en la rutina de inicialización.
4. Las medidas de rendimiento que se pueden calcular son:
   1. EL valor esperado de la promedio en cola y sistema
   2. El valor esperado de clientes en cola y en sistema
   3. Utilización del servidor.
5. Las gráficas no son generadas con el algoritmo, pero cada corrida devuelve una cadena de caracteres que puede ser copiada y pegada en Matlab para generar las mismas.

## Estructura del proyecto

Simulación (Nodo raíz)

Espera (Carpeta)

Arribo – Rutina de evento

Espera – Modelo de espera

Inicialización – Rutina de inicialización

Partida – Rutina de evento

Reporte – Rutina de reporte

Tiempos – Rutina de tiempos

Generadores (Carpeta)

Generador

Principal (Carpeta)

Principal

### Arribo

package espera;

import generadores.Libreria;

public class Arribo {

public static void funcion(Espera e) {

if (e.getS() == 'O') {

double aux = e.getAreaBajoQ() + (e.getReloj() -e.getTUE()) \* e.getN();

e.setAreaBajoQ(aux);

int num = e.getN() + 1;

e.setN(num);

Espera.Vta aux2 = new Espera.Vta(e.getReloj(), true);

e.getVTA().add(aux2);

}

else {

e.setS('O');

e.getDemora().add(0.0);

e.setTIOS(e.getReloj());//tiempo en que empieza a estar ocupado el servidor

int aux = e.getCli\_at() + 1;

e.setCli\_at(aux);

double tiempoServicio = Libreria.distExp(e.getMediaPartida());

Espera.Lev lev = new Espera.Lev(Partida.C(),tiempoServicio+ e.getReloj(), true);

e.getDemoraSis().add(tiempoServicio);

e.getLEV().add(lev);

}

Espera.Lev lev = new Espera.Lev(Arribo.C(), (Libreria.distExp(e.getMediaArribo()) + e.getReloj()), true);

e.getLEV().add(lev);

e.setTUE(e.getReloj());

}

public static char C(){

return 'A';

}

}

### Espera

package espera;

import java.math.BigInteger;

import java.util.ArrayList;

public class Espera {

//Variables

private char S; //D desocupado O ocupado

private double TUE;//tiempo ultimo evento

private long TFS; //tiempo final de simulacion

private double t; // tiempo

private ArrayList<Vta> VTA; //vector de tiempo de arribo

private int n;//tamaño de la cola de clientes sin atender

private int cli\_at;//numero de clientes atendidos

private double demoraTotal;

private ArrayList<Double> demora;//demora en cola

private ArrayList<Double> demoraSis;//demora en sistema

private double areaBajoQ;//acumulado de las áreas de la cola

// Se calcula como base por altura sobre 2 donde base = tiempo n que estuvo un valor constante

// y altura = tamaño de la cola n. Cada vez que n cambie un valor o un cliente entra al servidor y se // va de la cola, se calcula el área antes.

// de actualizar n a su nuevo valor. La base se calcula entre el tiempo (ahora) y la última vez que n //cambia de valor TUE

private double areaBajoB;//acumulado de la utilizacion del servidor

private double reloj;

private ArrayList<Lev> LEV;

double TIOS;

double mediaArribo;

double mediaPartida;

public Espera() {

reloj = 0;

areaBajoQ = 0;

areaBajoB = 0;

cli\_at = 0;

n = 0;

TUE = 0;

S = 'D'; //Desocupado D ocupado O

demoraTotal = 0;

VTA = new ArrayList<>();

t = 0;

LEV = new ArrayList<>();

TIOS = -1;

demora = new ArrayList<>();

demoraSis = new ArrayList<>();

}

public void setDemoraSis(ArrayList<Double> demoraSis) {

this.demoraSis = demoraSis;

}

public ArrayList<Double> getDemoraSis() {

return demoraSis;

}

public void setDemora(ArrayList<Double> demora) {

this.demora = demora;

}

public ArrayList<Double> getDemora() {

return demora;

}

public static class Lev {

char C;

double hora;

boolean usado;

public void setUsado(boolean usado) {

this.usado = usado;

}

public boolean isUsado() {

return usado;

}

public void setHora(double hora) {

this.hora = hora;

}

public double getHora() {

return hora;

}

public void setC(char C) {

this.C = C;

}

public char getC() {

return C;

}

public Lev(char C, double hora, boolean usado) {

this.C = C;

this.hora = hora;

this.usado = usado;

}

}

public static class Vta {

double hora;

boolean usado;

public Vta(double hora, boolean usado) {

this.hora = hora;

this.usado = usado;

}

public void setUsado(boolean usado) {

this.usado = usado;

}

public boolean isUsado() {

return usado;

}

public void setHora(double hora) {

this.hora = hora;

}

public double getHora() {

return hora;

}

}

public void setTIOS(double TIOS) {

this.TIOS = TIOS;

}

public double getTIOS() {

return TIOS;

}

public void setLEV(ArrayList<Lev> LEV) {

this.LEV = LEV;

}

public ArrayList<Lev> getLEV() {

return LEV;

}

public ArrayList<Vta> getVTA() {

return VTA;

}

public void setVTA(ArrayList<Vta> VTA) {

this.VTA = VTA;

}

public char getS() {

return S;

}

public void setS(char S) {

this.S = S;

}

public double getTUE() {

return TUE;

}

public void setTUE(double tUE) {

TUE = tUE;

}

public long getTFS() {

return TFS;

}

public void setTFS(long tfs) {

TFS = tfs;

}

public double getT() {

return t;

}

public void setT(double t) {

this.t = t;

}

public int getN() {

return n;

}

public void setN(int n) {

this.n = n;

}

public double getDemoraTotal() {

return demoraTotal;

}

public void setDemoraTotal(double demoraTotal) {

this.demoraTotal = demoraTotal;

}

public double getAreaBajoQ() {

return areaBajoQ;

}

public void setAreaBajoQ(double areaBajoQ) {

this.areaBajoQ = areaBajoQ;

}

public double getAreaBajoB() {

return areaBajoB;

}

public void setAreaBajoB(double areaBajoB) {

this.areaBajoB = areaBajoB;

}

public double getReloj() {

return reloj;

}

public void setReloj(double reloj) {

this.reloj = reloj;

}

public int getCli\_at() {

return cli\_at;

}

public void setCli\_at(int cli\_at) {

this.cli\_at = cli\_at;

}

public double getMediaArribo() {

return mediaArribo;

}

public double getMediaPartida() {

return mediaPartida;

}

public void setMediaArribo(double mediaArribo) {

this.mediaArribo = mediaArribo;

}

public void setMediaPartida(double mediaPartida) {

this.mediaPartida = mediaPartida;

}

}

### Inicialización

package espera;

import espera.Espera.Lev;

import generadores.Libreria;

public class Inicializacion {

public static void Inicia(Espera modelo, long TFS, double arribo, double partida){

//creamos el primer arribo y lo guardamos en LEV

//Nuestro Lev esta formado por dos colecciones una de tipos de eventos

//La otra esta formado por hora que seria la hora del tipo de evento

modelo.setMediaArribo(arribo);

modelo.setMediaPartida(partida);

//primer arribo

Lev a = new Lev(Arribo.C(), Libreria.distExp(modelo.getMediaArribo()), true);

modelo.getLEV().add(a);

//creamos la primer partida y la guardamos en LEV

//Seteamos el tiemp -1 para indicar que no puede ocurrir

a = new Lev(Partida.C(), -1,false);

modelo.getLEV().add(a);

//setiamos el tiempo final de simulación

modelo.setTFS(TFS);

}

}

### Partida

package espera;

import espera.Espera.Vta;

import generadores.Libreria;

import java.util.ArrayList;

public class Partida {

public static void funcion(Espera e)

{

if (e.getN() == 0) {

e.setS('D');

double aux = e.getAreaBajoB() + (e.getReloj() - e.getTIOS());

e.setAreaBajoB(aux);

Espera.Lev lev = new Espera.Lev(Partida.C(), -1, false);

e.getLEV().add(lev);

}

else{

double aux = e.getAreaBajoQ() + ((e.getReloj() - e.getTUE()) \* e.getN());

e.setAreaBajoQ(aux);

int ayuda = e.getN() - 1;

e.setN(ayuda);

double hora = 0;

ArrayList<Vta> auxiliar = e.getVTA();

for (int i = 0; i < e.getVTA().size(); i++) {

if (auxiliar.get(i).isUsado()) {

hora = auxiliar.get(i).getHora();

auxiliar.get(i).setUsado(false);

i=e.getVTA().size();

}

}

aux = e.getDemoraTotal()+(e.getReloj() - hora);

e.setDemoraTotal(aux);

e.setCli\_at(e.getCli\_at()+1);

double tiempoServicio = Libreria.distExp(e.getMediaPartida());

Espera.Lev lev = new Espera.Lev(Partida.C(),tiempoServicio + e.getReloj(), true);

e.getDemora().add(e.getReloj() - hora );//demora individual en cola

e.getDemoraSis().add(e.getReloj() - hora + tiempoServicio );//demora en sistema

e.getLEV().add(lev);

}

e.setTUE(e.getReloj());

}

public static char C(){

return 'P';

}

}

### Reporte

package espera;

import espera.Espera.Vta;

import generadores.Libreria;

import java.util.ArrayList;

public class Reporte {

public static double[] funcion(Espera e){

double[] aux = new double[3];

double q = e.getAreaBajoQ() / e.getT();

double b = e.getAreaBajoB() / e.getT();

double d = e.getDemoraTotal() / e.getCli\_at();

aux[0] = q; //número medio en cola

aux[1] = b; //utilización del servidor

aux[2] = d; //demora promedio

muestraSeparador();

System.out.println(" Parametros utilizados");

System.out.println("Media arribo: " + e.getMediaArribo());

System.out.println("Media partida: " + e.getMediaPartida());

System.out.println("Tiempo final de simulación: " + e.getTFS());

muestraSeparador();

System.out.println(" Medidas de rendimiento");

muestra(q,b,d);

muestraSeparador();

System.out.println(" Valores para graficar");

/\*modelo, parametro, booleano para la media, booleano para la varianza

parametro 0 demora media en sistema

parametro 1 demora media en cola

\*/

graficar(e,0,false);

graficar(e,1,false);

muestraSeparador();

return aux;

}

public static double calculaError (double valorExacto, double valorComparar) {

double aux = (valorExacto > valorComparar) ? (valorExacto - valorComparar) / valorExacto : (valorComparar - valorExacto) / valorExacto ;

return aux;

}

public static void muestra (double mediaCola, double mediaUtilizacionServidor, double mediaDemora){

double errorCola = calculaError(0.5, mediaCola);

double errorUtilizacionServidor = calculaError(0.5, mediaUtilizacionServidor);

System.out.println("Número medio en cola: " + Libreria.darFormato(mediaCola, 5) + " Error: " + Libreria.darFormato(errorCola,5));

System.out.println("Utilización del servidor: " + Libreria.darFormato(mediaUtilizacionServidor, 5)+ " Error: " + Libreria.darFormato(errorUtilizacionServidor,5));

System.out.println("Demora promedio: " + Libreria.darFormato(mediaDemora, 5));

}

public static void muestraSeparador() {

System.out.println("------------------------------------------");

}

public static void graficar(Espera e, int tipo, boolean v) {

String auxMedia = "";

String auxVarianza = "";

ArrayList parametro = new ArrayList<Double>();

double media = 0;

double mediaVar = 0;

switch(tipo){

case 0: auxMedia += "Demora media en sistema \n {";

auxVarianza += "Varianza de la demora en sistema \n {";

parametro = e.getDemoraSis();

mediaVar = e.getMediaArribo();

break;

case 1: auxMedia += "Demora media en cola \n {";

auxVarianza += "Varianza de la demora cola \n {";

parametro = e.getDemora();

mediaVar = e.getMediaPartida();

break;

case 2: auxMedia += "Utilización esperada del servidor \n {";

auxVarianza += "Varianza de la utilización del servidor \n {";

parametro = e.getDemora();

mediaVar = e.getMediaArribo();

break;

}

for(int i = 0; i < parametro.size(); i += 1){

if(i%500 == 0 && i != 0){

auxMedia += "\n";

auxVarianza += "\n";

}

media += (double) parametro.get(i);

if(i != 0){

auxMedia += Libreria.darFormato(media/i, 4);

auxVarianza += Libreria.darFormato(Math.pow((media/i) - mediaVar,2), 4);

if(i + 1 != parametro.size()){

auxMedia += ", ";

auxVarianza += ", ";

}

}

else {

auxMedia += Libreria.darFormato(media, 4) + ", ";

auxVarianza += Libreria.darFormato(media, 4) + ", ";

}

}

auxMedia += "}";

auxVarianza += "}";

if(v) {

System.out.println(auxMedia + "\n" + auxVarianza);

} else {

System.out.println(auxMedia);

}

}

public static void tiempoEA(Espera e){

ArrayList<Vta> aux = e.getVTA();

ArrayList<Double> tiempos = new ArrayList<>();

tiempos.add(aux.get(0).getHora());

for (int i = 1; i < aux.size(); i++) {

double help = aux.get(i).hora - aux.get(i-1).hora;

tiempos.add(help);

}

}

}

package espera;

import java.util.ArrayList;

### Tiempos

public class Tiempos {

public static char busca(Espera modelo){

char evento = 'z';

ArrayList<Espera.Lev> jeje = modelo.getLEV();

int indice = 0;

double hora = modelo.getTFS() + 100;

for (int i = 0; i < jeje.size(); i++) {

if (jeje.get(i).usado && jeje.get(i).hora < hora && modelo.getReloj() < jeje.get(i).hora) {

indice = i;

hora = jeje.get(i).hora;

}

}

jeje.get(indice).usado = false;

modelo.setReloj(jeje.get(indice).hora);

evento = jeje.get(indice).C;

return evento;

}

}

### Librería

package generadores;

import java.math.RoundingMode;

import java.text.NumberFormat;

import java.util.Random;

public class Libreria {

//Metodos de clase

public static String darFormato(double numero,int longitud){

NumberFormat nf = NumberFormat.getInstance();

nf.setMaximumFractionDigits(longitud);

nf.setRoundingMode(RoundingMode.DOWN);

return nf.format(numero);

}

//Metodos de Instancia privados

public static double Uniforme() {

//Nos devuelve un valor uniforme con dos digitos decimales

Random rnd= new Random();

double aux = rnd.nextDouble();

//darFormato(aux); nos muestra el n�mero generado con el formato de 4 d�gitos nomas

return aux;

}

private double UniformeRango(int rango,int inicio){

Random rnd = new Random();

//System.out.println("N�mero aleatorio entre["+inicio+":"+(rango+inicio)+"] : "+(rnd.nextDouble()\*rango+inicio));

return rnd.nextDouble()\*rango+inicio;

}

//Metodos de Instancia publicos

public double distUniforme(int a,int b) {

double x = a +(b-a) \* Uniforme();

return x;

}

public static double distExp(double media){

double x = -(media \* Math.log(Uniforme()));

return x;

}

public double distNormal(double media,double var,double kValores){

double k = 0;

double x=0;

for (int i = 0; i < kValores; i++) {

k = k + Uniforme();

}

x = ( var \* Math.pow((12/kValores), 0.5) \*(k-(kValores/2)) )+media;

return x;

}

}

### Principal

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package principal;

import espera.\*;

import generadores.Libreria;

/\*\*

\* @author MaximilianoDaniel

\*/

public class principal {

public static void main(String[] args) {

Espera modelo = new Espera();

/\*

Inicia recibe como parámetros el modelo creado,

el tiempo final de simulación,

la media de arribo

la media de partida

\*/

Inicializacion.Inicia(modelo, 10000, 0.5, 0.25);

char aux = Tiempos.busca(modelo);

while (modelo.getReloj() < modelo.getTFS()) {

if (aux != 'z') {

switch(aux){

case 'A': {Arribo.funcion(modelo);}

//System.out.println("Arribo: "+ modelo.getReloj());

break;

case 'P': {Partida.funcion(modelo);}

// System.out.println("Partida: " + modelo.getReloj());

break;

default: break;

}

}

else{

System.out.println("Error");

}

modelo.setT(modelo.getReloj());

aux = Tiempos.busca(modelo);

}

//agregamos las medias de la réplica

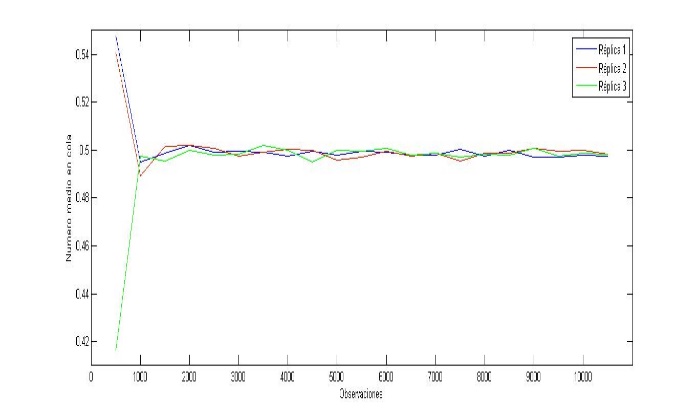
Reporte.funcion(modelo);

}

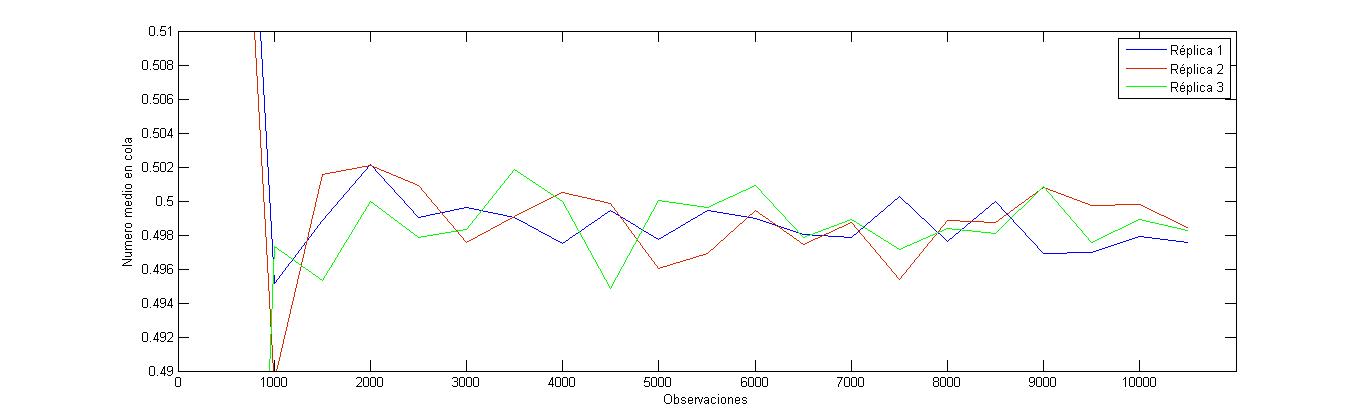
}

## Parámetros calculados con gráficas

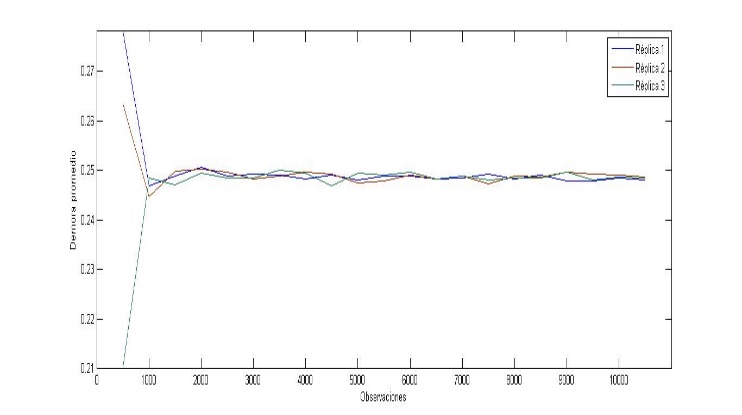
**Número medio en cola**



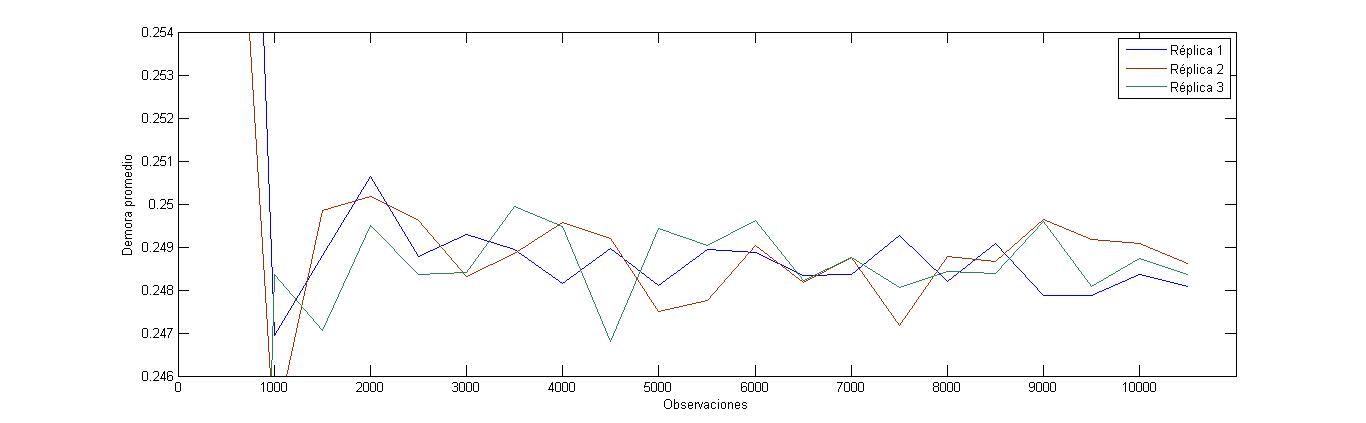
**Número medio en cola con zoom**

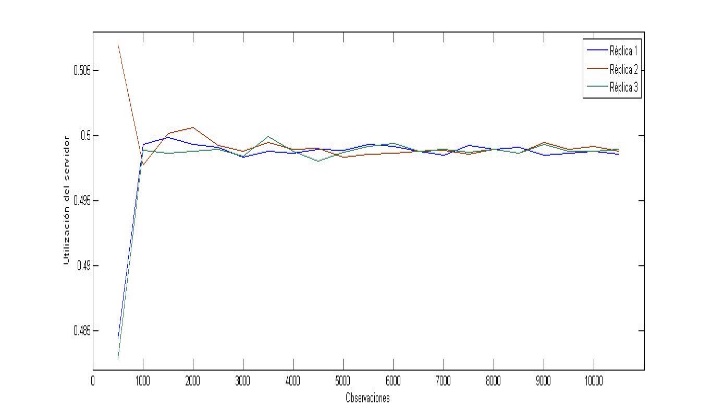


**Demora promedio**



**Demora promedio con zoom**

**Utilización del servidor**



**Utilización del servidor con zoom**

