Experimentación y Análisis

**G2 - Modelos Estocásticos y Simulación**

***Primer Parcial, Quinto Punto***

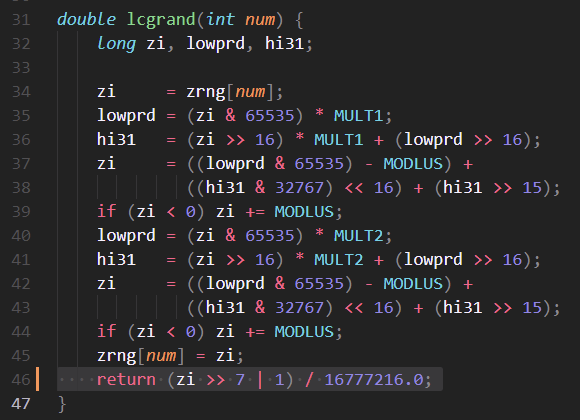
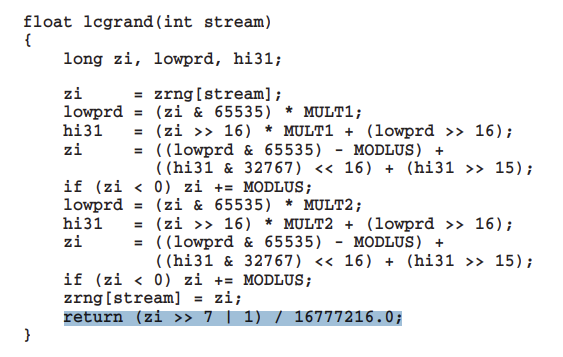
*Juan Esteban Alarcón Bravo*

*David Sneider Ovalle Pineda*

*Gustavo Alberto Puentes Romero*

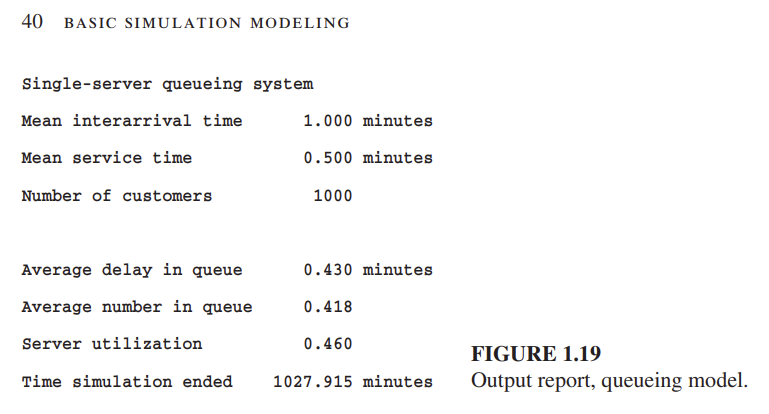
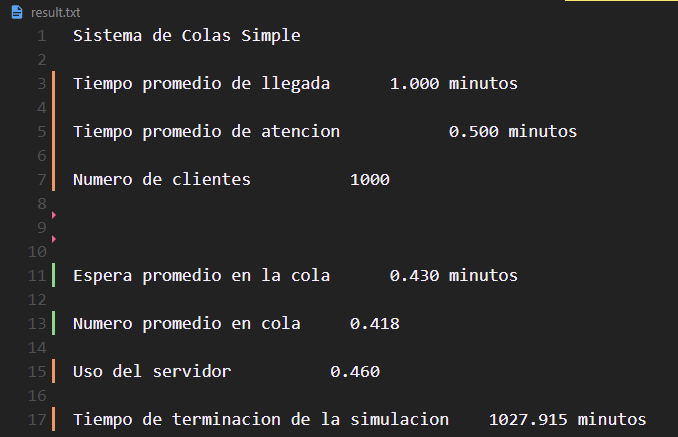
# Compilación y corrección inicial

Inicialmente se realizaron correcciones menores al código original. Primero, se corrigió un error tipográfico en el archivo lcgrand.cpp. En la línea 46 de la imagen, entre el 7 y el 1 había dos barras verticales, pero el libro de Law muestra solo una:

Las semillas del generador de números aleatorios **no** se cambiaron.

Al archivo principal de código se le cambió el nombre por main.cpp y se ejecutó con los parámetros del primer ejemplo contemplado en el libro:



Podemos apreciar que los resultados de la simulación son los mismos del libro, por lo que entendemos que el código funciona adecuadamente hasta este punto.

# Verificación de módulos

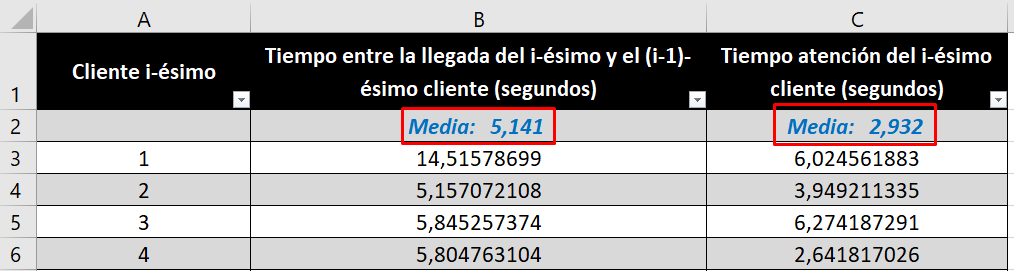
Los módulos del programa están referenciados en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| MÓDULO | FUNCIÓN QUE LO REPRESENTA | NOTAS |
| Inicialización |  | Funcionalidades ya incluidas y puestas en el módulo con la excepción de las funciones *fopen()* y *fscanf()* que estaban en *main()* y fueron reubicadas. |
| Manejo de Espacio-Tiempo |  | Funcionalidades ya incluidas y ubicadas en el módulo. |
| Eventos |  | Los dos eventos básicos de un sistema de colas, *llegada()* y *salida()*. Funcionalidades ya incluidas y ubicadas en el módulo. |
| Función Percentil |  | La función percentil fue una nueva implementación en el código. Se definió como , en donde es el tiempo de la simulación, es la función exponencial ya implementada y es el parámetro poblacional. |
| Generador de Reporte |  | Funcionalidades ya incluidas y ubicadas en el módulo. |
| Simulador Principal |  | Funcionalidades ya incluidas en el programa. Llama las funciones *inicializar()*, *controltiempo()* y *reporte()* y las funciones de eventos *llegada()* y *salida()*. |

# Verificación del simulador

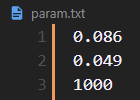
El archivo de Excel incluido con el código traía dos columnas de datos: Tiempo entre llegadas y tiempo de atención. En esta sección se mostrará la fiabilidad del simulador para reproducir estos datos.

Primero, para poder usar los datos del archivo en el simulador, se obtuvo la media de las dos columnas de datos:



Estas medias, que están en segundos, deben pasarse a minutos para ser usada como parámetros del simulador:

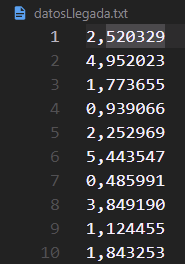
Se introducen estos datos en el archivo de parámetros del simulador:



En el archivo principal del simulador se agregó una línea de código que imprime en el archivo datosLlegada.txt el valor del tiempo entre llegadas que se va almacenando con cada iteración de la simulación:



El resultado son estos 1000 datos:

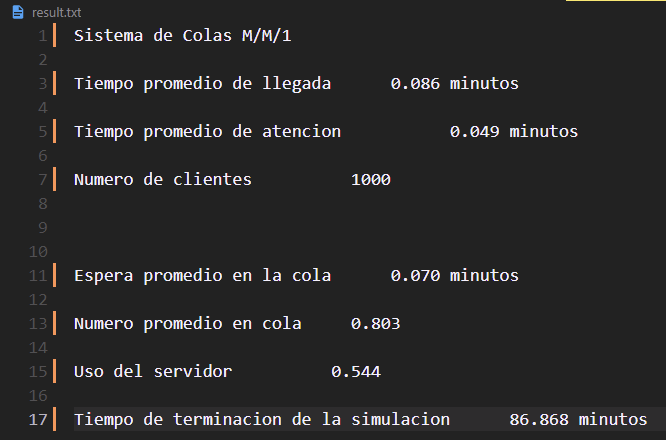


Transcribiendo esos datos a Excel y comparándolos con los datos originales, obtenemos la siguiente gráfica:

Ampliando una sección arbitraria (intervalo 700-800), podemos ver en mejor detalle los resultados:

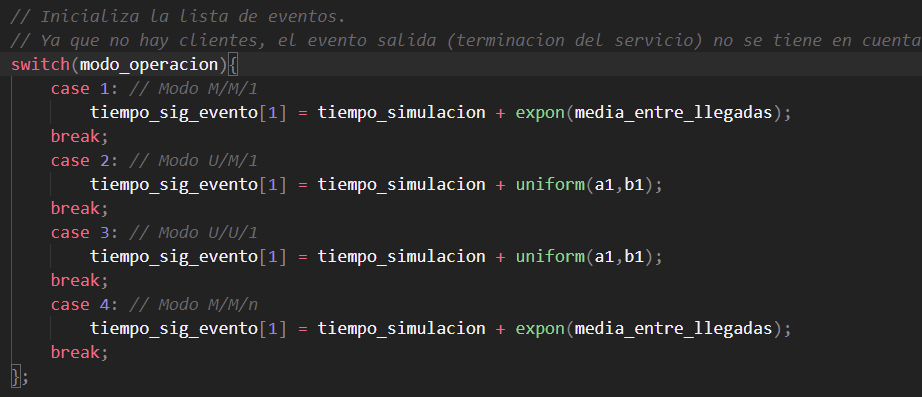
Podemos entonces observar que el simulador puede representar de manera aproximada los datos teóricos del archivo de datos proporcionado.

Estos son los resultados del generador de reportes:

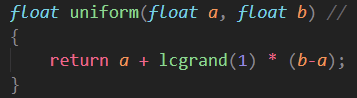
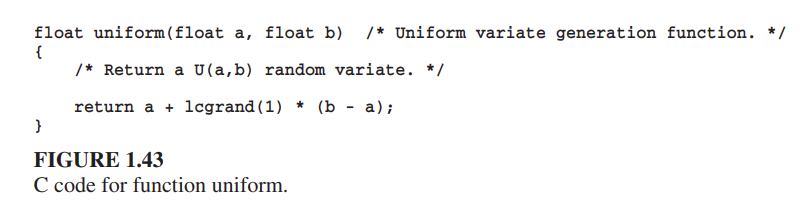


# Modificación para modelos con intervalos uniformes

Para poder utilizar el simulador en estos nuevos modos de operación, se creó la variable modo\_operacion. Esta variable modifica el flujo del programa para que pueda utilizar distintos valores de entrada. En este caso, para el “tiempo promedio de llegada” no se utilizará la función exponencial con media aleatoria que estaba implementada por defecto, sino que se usará un intervalo uniforme que hace uso del generador de números aleatorios para escoger un valor de llegada:



*En vez de usar la función expon() para ajustar el tiempo de la siguiente llegada, se usa uniform(), por ejemplo.*

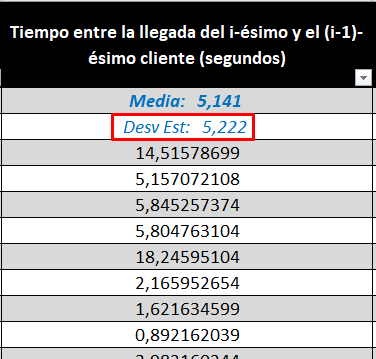
 

*Detalle de la función uniform(), copiada del libro de Law.*

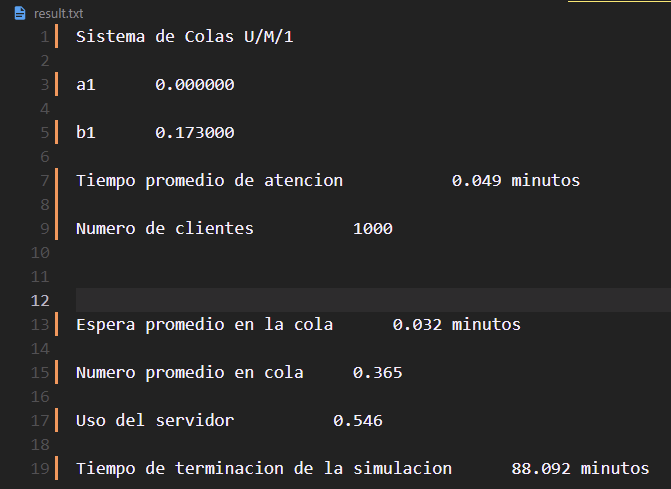
Estas dos nuevas implementaciones nos ayudan también a resolver los siguientes dos casos, en donde se implementa un modelo U/M/1 y un modelo U/U/1.

## Modelo

Nuevamente vamos a comparar los datos teóricos con los prácticos, pero para este modelo necesitamos definir valores adecuados para usar en el intervalo . Para esto, calculamos ahora también la desviación estándar de los datos:



El intervalo entonces lo definimos arbitrariamente como (en minutos, nuevamente). Estos parámetros los pasamos al simulador, lo ejecutamos y obtenemos los siguientes resultados:



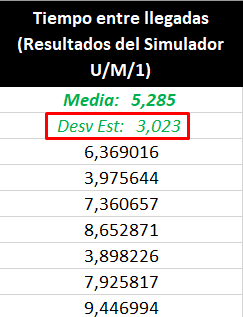
*Nota: el valor mínimo del intervalo lo establecimos como 0 porque es un número negativo, y esto en términos de tiempo no es posible*

Lo primero que vemos es que el tiempo de espera y el número promedio de personas en la cola se redujo a aproximadamente la mitad que en el caso anterior (M/M/1). Por otro lado, el uso del servidor y el tiempo de terminación de la simulación son similares.

Veamos ahora el comportamiento del tiempo entre llegadas:

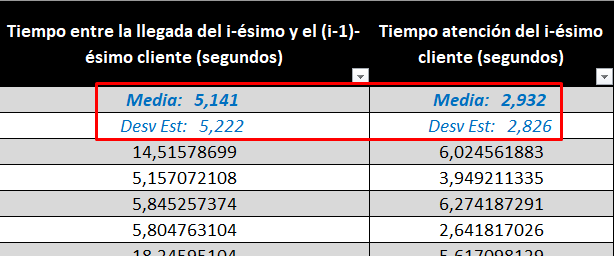
Lo que más resalta de esta simulación es la uniformidad de los datos: parecen bien distribuidos en un intervalo aproximado de . Esto era de esperarse, ya que la función uniforme no genera tantos datos atípicos como una distribución exponencial, incluso cuando ambas hacen uso del generador de números aleatorios.

La desviación estándar también es menor en este caso, aunque la media es similar:

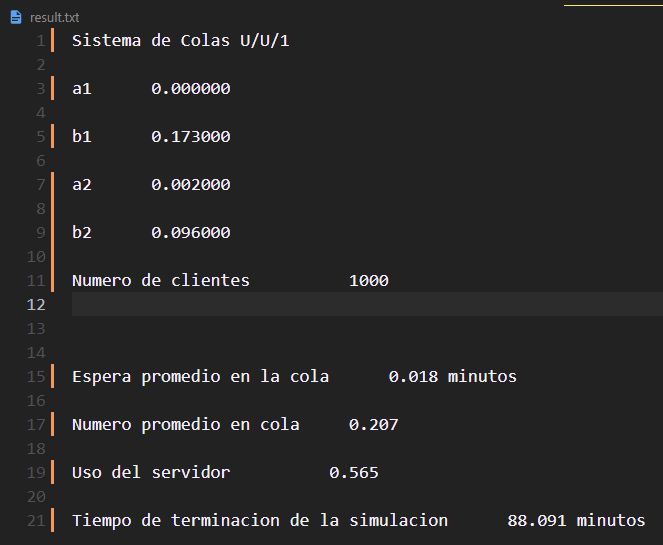


## Modelo

Aplicando el mismo procedimiento del modelo anterior, calculamos la media y la desviación estándar de las dos variables:



Convertimos estos valores a minutos y los ingresamos en el simulador:



Nuevamente los valores de espera promedio y número promedio en la cola se redujeron a la mitad de la simulación anterior (y, por ende, a una cuarta parte de la simulación por defecto M/M/1).

Nuevamente vemos una distribución de datos uniforme, muy similar a la anterior.

# Modificación para modelos con múltiples servidores (M/M/n)

Para generalizar el simulador a todos los casos M/M/n desde su estado actual (que es un caso particular de este modelo, donde ), debemos hacer ajustes en el control de la variable estado\_servidor. Originalmente, esta tenía dos estados definidos así:



Esto solo permitía caracterizar un servidor con el estado “libre” u “ocupado”, pero al cambiar esta variable para que utilice más valores, podemos hacer que sea interpretada como “el número de servidores disponibles”.

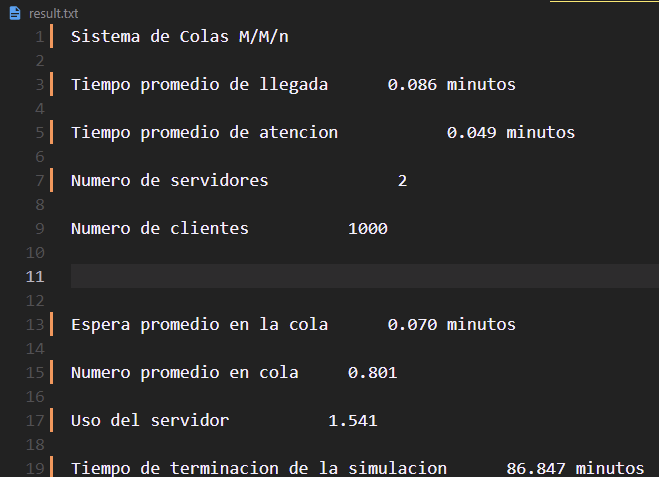
Por ejemplo, en la función llegada(), la línea 224 fue reemplazada por la línea 225: Ahora, ya no se revisa si “el servidor está ocupado” sino que se revisa si “en número de servidores disponibles es menor o igual al número total de servidores”, o, en palabras más sencillas, “¿hay servidores disponibles?”:

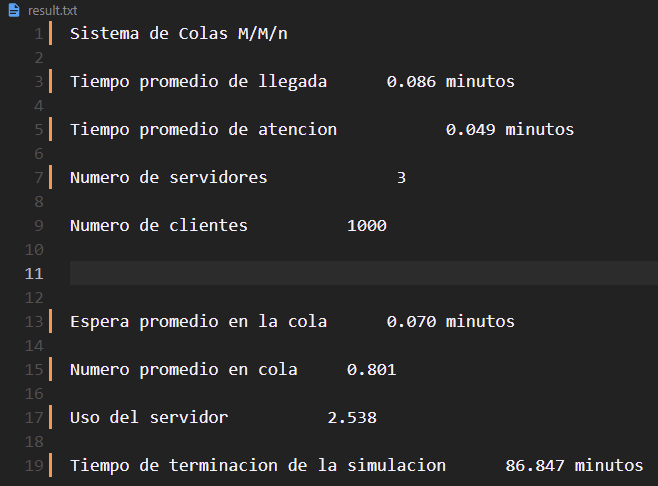


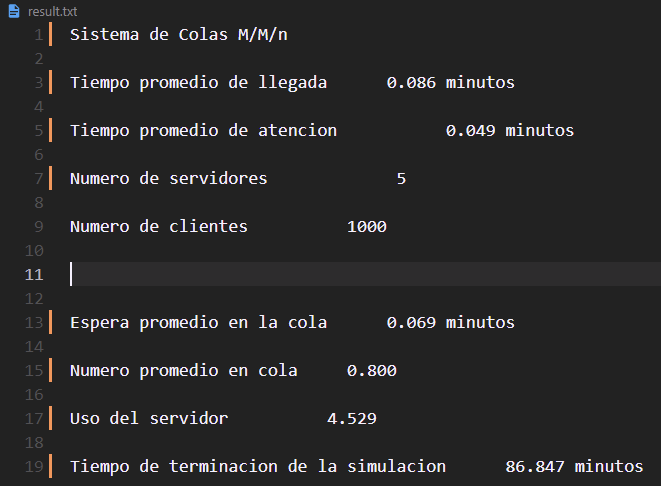
La variable num\_servidores es un nuevo parámetro de entrada que se refiere a la del modelo M/M/n que queremos trabajar. Si esta fuera dejada con un valor de 1, por ejemplo, obtendríamos los mismos resultados que en el caso trivial del modelo M/M/1.

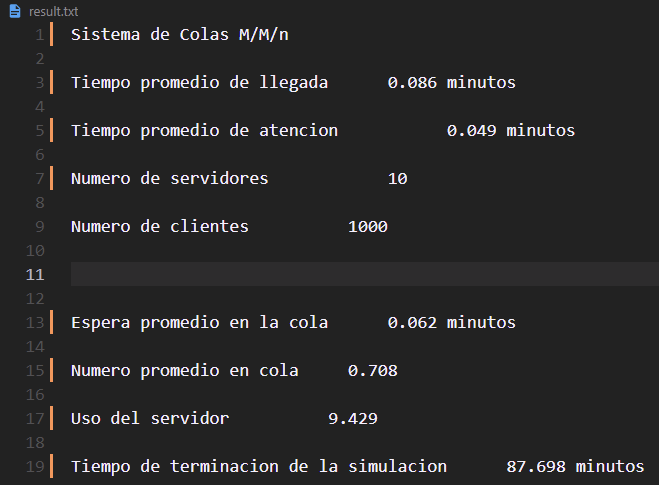
## Modelo

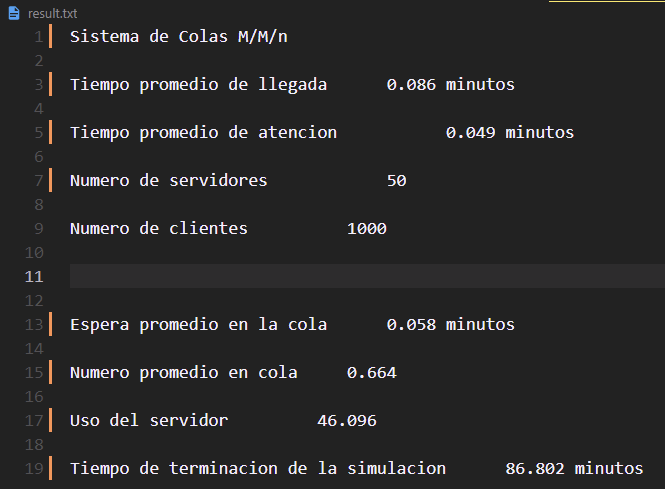
Ya con el simulador modificado, probamos su funcionalidad con diferentes valores de . Específicamente, se probó con .











Vemos entonces que el uso del servidor ahora representa la cantidad promedio de servidores que estuvieron en uso durante la simulación. Este dato refleja fidedignamente el parámetro de entrada , lo que significa que el sistema está usando todos los servidores en la simulación.