UNIVERSIDAD DE GRANADA

Simulación de Sistemas

grado en ingeniería informática

Práctica 3

Modelos de Simulación Dinámicos y Discretos

Autor
José María Sánchez Guerrero

Rama
Computación y Sistemas Inteligentes

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2019-2020

Índice general

Capítulo 1

Mi segundo modelo de simulación Discreto

Nuestro modelo de simulación consistirá en un servidor que presta un determinado servicio a una serie de clientes, los cuales solicitarán dicho servicio periódicamente. Cuando llega un cliente y el servidor no está ocupado, será atendido inmediatamente; en caso contrario, el cliente tendrá que esperar en la cola. Cuando se completa un servicio, el servidor elegirá al siguiente en una forma FIFO.

Al empezar la simulación, no habrá clientes esperando y el servidor está libre. Utilizaremos el mismo generador exponencial tanto para el tiempo que tardarán en llegar los clientes, como el tiempo que tardará el servidor en atender a cada uno.

1.1. Simulación con incremento fijo de tiempo

En esta simulación, vamos a tratar al tiempo incrementándolo de unidad en unidad. Para evitar problemas con el manejo del tiempo, tendremos que modificar los generadores de datos para que nos devuelvan los valores redondeados al entero más próximo. Si obtenemos un valor igual a 0, devolveremos 1 en su lugar, ya que el suceso generado quedaría en un tiempo anterior al actual, que generamos al incrementar en una unidad.

Este será nuestro código resultante, que nos servirá tanto para generar el tiempo de llegada del cliente como para generar el timepo del servicio (sólo tendremos que modificar la variable *tlleg* por *tserv*):

```
float generallegada(float tlleg){
  float u = random();          // o tambien rand() en lugar de random()
  u = ( u / (RAND_MAX+1.0) ); //RAND_MAX es una constante del sistema
  u = round( -tlleg * log(1-u) );

  if (u != 0)
    return u;
  else
    return 1.0;
}
```

Para la simulación vamos a emplear diferentes unidades de medida de tiempo (horas, minutos, segundos...) y con un número de clientes a atender bastante alto, de unos 10.000, para que los resultados sean robustos. Este es el resultado que obtenemos:

tiempo	**tlleg**	**tserv**	**% tiempo ocioso**	**Media clientes en cola**
horas	0.15	0.1	0.019994	0.000000
medias horas	0.3	0.2	0.625621	0.078153
cuartos de hora	0.6	0.4	7.224885	0.356633
minutos	9	6	31.501728	1.647961
segundos	540	360	33.934223	1.282934

Como podemos ver, los porcentajes de tiempo ocioso del servidor tienen una variación bastante grande. Esto se debe a que los valores pequeños (unidades de tiempo más altas como las horas o la medias horas) producen unos valores muy próximos a cero en los generadores y, en consecuencia, son redondeados. Como ya hemos visto, estos valores no serán devueltos como 0, si no como 1.

Con esto estamos diciendo que un suceso dura más de lo que realmente es, y vamos acumulando este error en toda la simulación. Esta es la razón por la cual los modelos de incremento fijo no son los más apropiados, ya que sus valores son bastante diferentes a los que obtendríamos sobre el papel.

A continuación, vamos a mostrar unas gráficas con diversas ejecuciones para comprobar que los resultados obtenidos son coherentes y ver mejor las diferencias que obtenemos al utilizar distintas medidas tiempo:

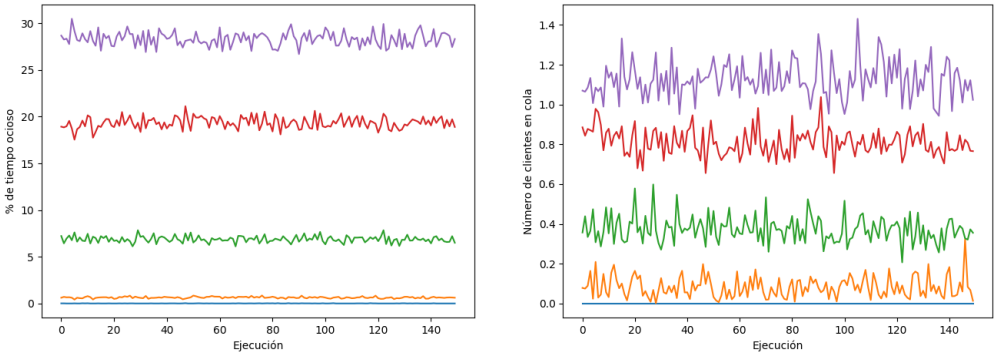

Figura 1.1: Porcentaje de tiempo ocioso del servidor y media de clientes en cola

Aqui podemos observa cómo las gráficas van teniendo un porcentaje de ocupación y un número de clientes en cola más alto a medida que disminuimos las valores *tlleg* y *tserv*. También podemos observar que, pese a que cada una de las líneas rondan un rango de valores, existe cierta variabilidad en los resultados; por lo cual sería un poco irresponsable fiarnos de un solo dato como el de la tabla.

1.2. Simulación con incremento variable de tiempo

Ahora vamos a tratar el mismo problema pero con un incremento de tiempo variable, para hacerlo más eficiente y preciso. En este caso, la variable no tendrá ni por qué ser un entero ni tendremos que parsear los generadores de datos, ya que se incrementará su valor hasta el suceso más cercano. Decimos que es más preciso porque porque no tiene que dar los saltos que daba el anterior modelo y, en consecuencia, gana en eficiencia ya que va suceso a suceso.

Esta es la nueva línea de código que tendremos que añadir (también quitaremos la que aumenta el tiempo de manera uniforme $reloj++$):

```
1 reloj = min(tiempo_llegada, tiempo_salida)
```

A continuación, vamos a ejecutar este modelo de la misma forma que hemos hecho con el modelo anterior y óbservar los cambios que se obtienen:

tiempo	tlleg	tserv	% tiempo ocioso	Media clientes en cola
horas	0.15	0.1	33.993752	1.260754
medias horas	0.3	0.2	33.555038	1.250663
cuartos de hora	0.6	0.4	34.353592	1.382941
minutos	9	6	34.451107	1.211860
segundos	540	360	33.695103	1.261086

Vemos que los resultados esta vez han sido más regulares, ya que para todas las medidas de tiempo se han obtenido unos valores entre el 33 y el 34 por ciento. A diferencia del modelo anterior, en este podríamos decir que los resultados son bastante más fiables independientemente de si utilizamos horas, minutos, segundos... Esto es debido a que ahora no tenemos la acumulación del error que teníamos anteriormente, y los sucesos se producen siempre en el momento declarado, sin ponderar.

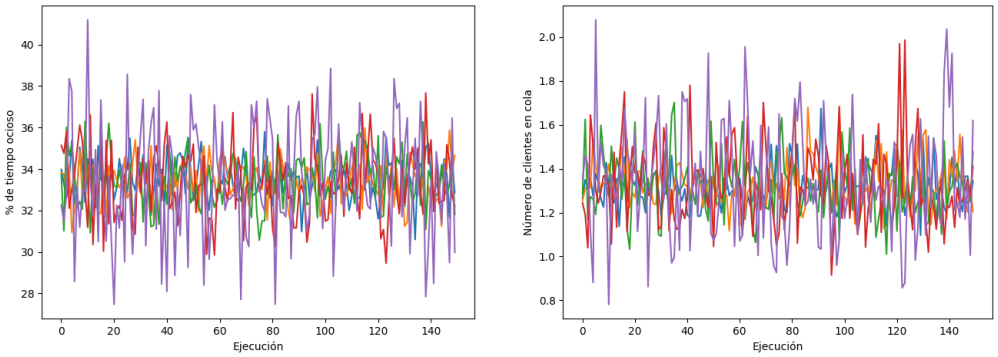

Figura 1.2: Porcentaje de tiempo ocioso del servidor y media de clientes en cola

Si observamos ahora las gráficas con las diversas ejecuciones, vemos que están prácticamente todas unas encima de otras. Como acabamos de decir, ya no tenemos este error en las distintas medidas de tiempo. No obstante, vemos que aún tenemos bastante variabilidad en los datos, llegando algunos a máximos como 42 o a mínimos como 27.

Por otra parte, esta mejora no sólo afecta a las medidas de tiempo, si no que también afecta a la eficiencia. En el modelo anterior, los incrementos en el reloj dependen de la medida del tiempo; mientras que en el modelo actual, el número de incrementos dependerá del número de sucesos. Vamos a comprobarlo extrayendo una tabla que compare los tiempos de ejecución para todas las medidas con incremento fijo, con las mismas medidas en incremento variable:

tiempo	tlleg	tserv	tiempo fijo	tiempo variable
horas	0.15	0.1	0.001671	0.002071
medias horas	0.3	0.2	0.002937	0.002562
cuartos de hora	0.6	0.4	0.003491	0.001119
minutos	9	6	0.003358	0.001464
segundos	540	360	0.020543	0.002939

Podemos ver que el tiempo de ejecución en el modelo con incremento fijo va aumentando lo que parece lineal o exponencialemente, mientras que en el modelo con incremento variable tenemos unos tiempos prácticamente iguales en unas ejecuciones y otras.

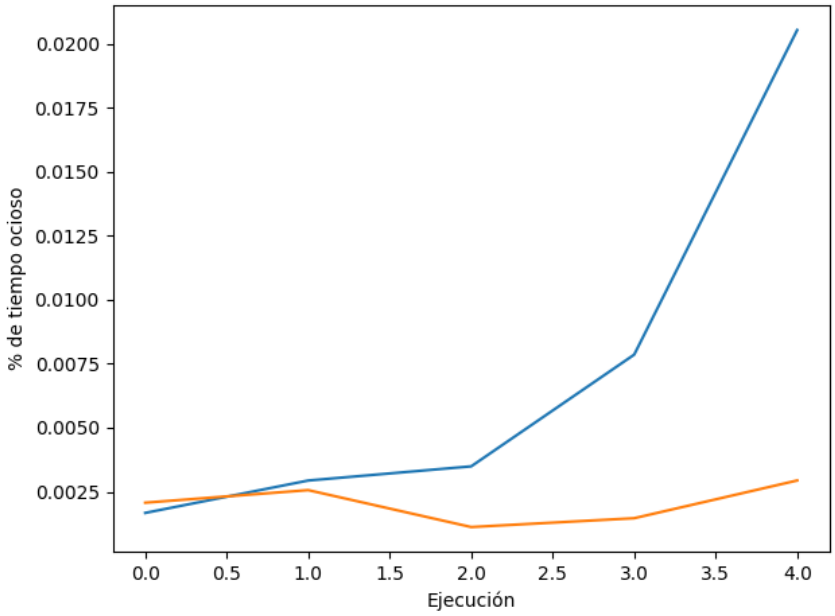

Figura 1.3: En azul los tiempos con incremento fijo y, en naranja, variable.

En la gráfica lo podemos corroborar. Tras estudiar los dos modelos, concluímos que utilizar uno de incremento variable sería lo acertado, no sólo por la mayor eficiencia en tiempo que nos da, si no por algo más importante como es la precisión y calidad al trabajar con distintos tipos de datos o unidades de medida.

1.3. Programa de simulación dinámico y discreto

En esta sección vamos a abordar un problema similar al anterior, pero con una cantidad de m servidores idénticos trabajando en paralelo. La cola de clientes seguirá siendo la misma, tratando a los clientes del mismo método FIFO. Si utilizamos más de un servidor en complicado, pero con $m = 1$ es posible obtener los valores teóricos de casi todas las medidas de rendimiento.

Vamos a ejecutar el programa para ver qué resultamos obtenemos. Lo haremos con unos parámetros como los anteriores ($parada = 10.000$, $tlleg = 9$ y $tserv = 6$):

```
jose@jose-Lenovo-G50-70:~/Universidad/SS (Simulacion de Sistemas)/SS_Practicas/P3$
Tiempo medio de espera en cola = 12.840
Tiempo medio de estancia en el sistema = 18.840
Numero medio de personas en cola = 1.425
Numero medio de personas en el sistema = 2.086
Longitud media de colas no vacias = 3.261
porcentaje medio de tiempo de ocio por servidor = 33.960
Longitud maxima de la cola = 29
```

Figura 1.4: Ejecución del programa colammk con $m = 1$.

Una vez obtenidos estos valores, vamos a contrastarlos con los teóricos. Lo primero es calcularlos:

- Tiempo medio de espera en cola $= \frac{tserv^2}{tlleg - tserv} = \frac{6^2}{9-6} = \frac{36}{3} = 12$
- Tiempo medio de estancia en el sistema $= \frac{tserv \cdot tlleg}{tlleg - tserv} = \frac{6 \cdot 9}{9-6} = \frac{54}{3} = 18$
- Número medio de clientes en cola $= \frac{tserv^2}{tlleg(tlleg - tserv)} = \frac{6^2}{9 \cdot (9-6)} = \frac{36}{27} = 1.3$
- Número medio de clientes en el sistema $= \frac{tserv}{tlleg - tserv} = \frac{6}{9-6} = \frac{6}{3} = 2$
- Longitud media de colas no vacías $= \frac{tlleg}{tlleg - tserv} = \frac{9}{9-6} = \frac{9}{3} = 3$
- Porcentaje de tiempo de ocio del servidor $= \left(1 - \frac{tserv}{tlleg}\right) \cdot 100 = \left(1 - \frac{6}{9}\right) \cdot 100 = \frac{100}{3} = 33.3$

Como podemos ver, los resultados se ajustan bastante bien a los que esperábamos teóricamente. Esto se debe, en parte, a que la simulación se ha realizado con un tiempo de parada muy alto, es decir, la simulación ha durado bastante tiempo y las medias que se obtienen son lo suficientemente representativas.

Si probamos con un tiempo de parada bajo, los resultados serán más impredecibles o variables, como se muestra a continuación:

```
jose@jose-Lenovo-G50-70:~/Universidad/SS (Simulacion de Sistemas)/SS_Practicas/
Tiempo medio de espera en cola = 4.842
Tiempo medio de estancia en el sistema = 10.842
Numero medio de personas en cola = 0.996
Numero medio de personas en el sistema = 1.740
Longitud media de colas no vacias = 1.947
porcentaje medio de tiempo de ocio por servidor = 25.629
Longitud maxima de la cola = 5
```

Figura 1.5: Ejecución del programa colammk con $m = 1$ y tiempo bajo.

Ahora vamos a aumentar el número de servidores m, pero manteniendo un equilibrio. El tiempo de servicio dividido por el número de servidores tiene que permanecer constante. Vamos a modificar el programa para que repita varias veces la simulación y calcule las medias y desviaciones típicas de las medidas de rendimiento.

Las líneas más importantes que se han modificadas son las siguientes:

```
void fin(){
  ....
  // Variables nuevas para medias y desviaciones
  retrasoMedio += retrasomedio;
  estanciaMedia += estanciamedia;
  enColaMedio += encolamedio;
  enSistemaMedia += ensistemamedio;
  colasMedia += colasnovaciasmedio;
  porcentajeOcioMedio += porcentajemedioocio;
  longMaxCola += maximacola;

  retrasoMedioDes += pow(retrasomedio, 2);
  estanciaMediaDes += pow(estanciamedia, 2);
  enColaMedioDes += pow(encolamedio, 2);
  enSistemaMedioDes += pow(ensistemamedio, 2);
  colasMediaDes += pow(colasnovaciasmedio, 2);
  porcentajeOcioMedioDes += pow(porcentajemedioocio, 2);
  longMaxColaDes += pow(maximacola, 2);
}
```

```
float calcularDesviacion(float suma, float media)
{
    float desviacion = suma - (n_simulaciones * pow(media, 2));
    desviacion = desviacion / (n_simulaciones - 1);

    return sqrt(desviacion);
}
```

Por último ya tenemos el bucle for, la división entre el número de simulaciones para hacer la media y desviaciones típicas, y posteriormente imprimimos los resultados.

La prueba de este programa la vamos a realizar con varios números de servidores, sin embargo, el resto de parámetros que usaremos serán los mismos que los de la ejecución con $m = 1$, ya que los resultados obtenidos eran muy similares a los valores teóricos. En este caso vamos a realizar la ejecución con 2, 3, 4 y 5 servidores, adaptando el *tserv* a cada uno de ellos para mantener el equilibrio. Este ha sido el resultado:

Número de servidores	**2** **tserv: 12**	**3** **tserv: 18**	**4** **tserv: 24**	**5** **tserv: 30**
Tiempo medio espera en cola	med: 9.41335 des: 0.730856	med: 7.80226 des: 1.19625	med: 6.5818 des: 1.59282	med: 5.78543 des: 0.709112
Tiempo medio estancia en el sistema	med: 21.4133 des: 0.730836	med: 25.8023 des: 1.19633	med: 30.5818 des: 1.59289	med: 35.7854 des: 0.70908
Numero medio personas en cola	med: 1.05059 des: 0.0854933	med: 0.883231 des: 0.0896291	med: 0.746949 des: 0.0693109	med: 0.649897 des: 0.0780725
Número medio personas en el sistema	med: 2.37887 des: 0.0963648	med: 2.87863 des: 0.103085	med: 3.40728 des: 0.0925651	med: 3.98036 des: 0.118443
Longitud media colas no vacías	med: 2.97911 des: 0.172741	med: 2.99961 des: 0.209258	med: 2.98825 des: 0.184951	med: 2.97684 des: 0.214132
Porcentaje medio tiempo de ocio	med: 33.5856 des: 0.744918	med: 33.4867 des: 0.718379	med: 33.4917 des: 0.822586	med: 33.3909 des: 1.01696
Longitud máxima cola	med: 18.4 des: 3.43452	med: 17.88 des: 3.86317	med: 17.16 des: 3.32191	med: 16.5 des: 2.74234

Podemos ver que a medida que aumentamos el número de servidores, los resultados van siendo mejores. Los tiempos medios de espera en cola se reducen, con menos clientes en cola y, a su vez, un número mayor de clientes en el sistema.

Por otro lado, podemos ver que el porcentaje medio de tiempo de ocio del servidor es prácticamente el mismo en todos los casos (teniendo en cuenta siempre un cierto error) y el tiempo medio de estancia en el sistema se va incrementando. Esto quiere decir que el sistema está bien equilibrado y, a su vez, deducimos que estamos desperdiciando potencia del servidor; es decir, que pese a tener un número alto de servidores, no tenemos los suficientes clientes para que funcionen eficientemente o, simplemente, no los distribuímos de una forma eficaz.

Para finalizar con este modelo de simulación, vamos a estudiar que sucede si reemplzamos los generadores actuales por generadores determinísticos y uniformes. El nuevo código implementado para el determinístico y para el uniforme, respectivamente, es:

```
// Deterministico. Devuelve siempre el valor medio introducido
float generador_deterministico(float media)
{
  return media;
}

// Uniforme. Devuelve un valor uniformemente generado con media
// en el valor introducido
float generador_uniforme(float media)
{
  float u;
  u = (float) random();
  u = u/(float)(RAND_MAX+1.0);
  return(media*2*u);
}
```

Las pruebas también se van a hacer con una cantidad $m = 1$ de servidores y con el resto de parámetros igual que en las pruebas anteriors. El resultado es el siguiente:

Tipo de generador	**Generador determinístico**	**Generador uniforme**
Tiempo medio espera en cola	0	3.498
Tiempo medio estancia en el sistema	6	9.498
Numero medio personas en cola	0	0.387
Número medio personas en el sistema	0.667	1.051
Longitud media colas no vacías	0	1.519
Porcentaje medio tiempo de ocio	33.339	33.534
Longitud máxima cola	0	6

Como podemos ver, el generador determinístico ha obtenido los mejores resultados posibles, con un tiempo medio de espera en la cola de 0 minutos y un tiempo de servicio medio de 6 minutos. Esto se debe a que el generador de tiempos de servicio no produce las variaciones que se pueden dar en la vida real, y por lo tanto,

como el tiempo de llegadas es 9 minutos y el de servicio 6, nunca se pisrán el uno con el otro.

En cuanto al generador uniforme, vemos que nos da resultados diferentes a los del generador determinístico y que, en un principio, parecen bastante buenos. No obstante, si los comparamos con los valores teóricos calculados anteriormente, vemos que son muy diferentes, lo que quiere decir que son unos valores bastante alejados de la realidad.

Capítulo 2

Mi tercer modelo de simulación Discreto

En este modelo de simulación vamos a estudiar el funcionamiento de un puerto y posteriormente, realizar mejoras en el sistema para poder hacerlo más eficiente. El puerto consta de tres puntos de atraque, por lo que podrá cargar tres petroleros simultáneamente. También tendremos tres tipos de petroleros según el tiempo de carga que necesiten, a lo que hay que añadirle las horas que tardan los petroleros en llegar al puerto.

Tanto para atracar como para salir del puerto, cada petrolero necesitará los servicios de un remolcador. Tendremos sólo un remolcador en el puerto, el cual tarda 1 ± 0.25 horas en realizar la actividad más 0.25 horas si tiene que ir desde la bocana del puerto hasta los puntos de atraque. El remolcador tratará a los barcos de forma FIFO, tanto para los de atracan como para los que se van.

En la zona donde está situado el puerto se producen tormentas frecuentes, por lo que el remolcador quedará inactivo durante estas (excepto si ya está realizando una actividad). Si la actividad que está viajando hasta un punto de atraque, también se cancelará la actividad.

2.1. Configuración inicial del remolcador

Haciendo uso de la implementación ya proporcionada, vamos a realizar una serie de ejecuciones con distintos números de simulaciones, para asi determinar cual es un valor adecuado de ellas.

Número de simulaciones	10	50	100	150	500
Media barcos cola de llegadas	media: 1.333533 des: 0.523857	media: 1.135396 des: 0.411275	media: 1.166943 des: 0.468569	media: 1.221241 des: 0.439310	media: 1.201712 des: 0.461860
Media barcos cola de salidas	media: 0.029232 des: 0.004155	media: 0.028225 des: 0.002669	media: 0.028771 des: 0.003588	media: 0.028802 des: 0.003547	media: 0.028596 des: 0.003577
Tiempo medio puerto tipo 0	media: 34.803562 des: 5.445097	media: 33.173023 des: 4.127820	media: 33.263069 des: 4.946590	media: 33.819340 des: 4.702908	media: 33.678764 des: 4.920179
Tiempo medio puerto tipo 1	media: 41.308720 des: 5.769780	media: 38.570274 des: 4.494416	media: 39.113869 des: 4.792210	media: 39.822876 des: 4.737566	media: 39.477352 des: 5.019176
Tiempo medio puerto tipo 2	media: 52.762413 des: 5.920279	media: 50.656128 des: 4.696106	media: 51.064156 des: 5.299624	media: 51.613480 des: 4.874365	media: 51.393242 des: 5.082584
% tiempo remolcador desocupado	media: 80.564148 des: 0.210406	media: 80.600021 des: 0.183851	media: 80.609428 des: 0.160885	media: 80.639183 des: 0.167643	media: 80.619522 des: 0.205144
% tiempo remolcador viajando vacío	media: 1.170690 des: 0.236674	media: 1.296613 des: 0.218616	media: 1.309277 des: 0.237519	media: 1.260584 des: 0.239027	media: 1.272699 des: 0.227300
% tiempo remolcador remolcando	media: 18.265163 des: 0.181919	media: 18.103365 des: 0.220881	media: 18.081308 des: 0.246783	media: 18.100248 des: 0.256010	media: 18.107714 des: 0.241916
% tiempo puntos atraque libres	media: 12.410981 des: 1.272044	media: 13.255801 des: 1.253805	media: 13.299381 des: 1.415099	media: 13.025707 des: 1.442802	media: 13.064457 des: 1.378616
% tiempo puntos atraque ocupados (sin carga)	media: 0.974415 des: 0.138513	media: 0.940827 des: 0.088957	media: 0.959018 des: 0.119604	media: 0.960067 des: 0.118232	media: 0.953195 des: 0.119229
% tiempo puntos atraque ocupados (cargando)	media: 86.614601 des: 1.206176	media: 85.803375 des: 1.242067	media: 85.741608 des: 1.397373	media: 86.014229 des: 1.409460	media: 85.982361 des: 1.347268

Como podemos ver, los resultados han sido muy parecidos para todas las pruebas, incluso para las que tienen un número de simulaciones muy bajo todos los parámetros son muy similares. El único parámetro que varía más es el número medio de barcos en cola de llegadas, que con bajas simulaciones puede tomar variaciones de hasta ± 0.2 o ± 0.25 barcos de media. En los valores más altos, este valor toma aproximadamente una media de 1.20, por lo que se tomarán los valores cercanos a este como los correctos.

Como lo que tarda la simulación no es un tiempo excesivo, vamos a realizar futuras ejecuciones con un número de simulaciones de 150, pero en caso de tener un computador menos potente o realizar modificaciones que relenticen mucho el tiempo, podremos bajarlo, ya que no será excesivamente impreciso.

2.2. Mejoras propuestas para el remolcador

La primera mejora propuesta es la de aumentar la cantidad de puntos de atraque a 4 o 5. En mi caso, lo vamos a realizar con 4, 5 y 50, una cantidad exagerada de puntos de atraque para comprobar si existe cuello de botella y en que punto estaría situado.

Para cambiar el número de puntos de atraque, simplemente tendremos que situarnos en el archivo *puerto.h* y cambiar el parámetro *num_atraques* a 4, 5 o 50. Los resultados son los siguientes:

Puntos de atraque	**3**	**4**	**5**	**50**
Media barcos cola de llegadas	1.221241	0.087470	0.047531	0.045330
Media barcos cola de salidas	0.028802	0.028978	0.029302	0.029909
Tiempo medio puerto tipo 0	33.819340	21.290621	20.853863	20.841713
Tiempo medio puerto tipo 1	39.822876	27.292864	26.873707	26.824234
Tiempo medio puerto tipo 2	51.613480	39.280964	38.843136	38.826294
% tiempo remolcador desocupado	80.639183	78.232468	77.879196	77.831390
% tiempo remolcador viajando vacío	1.260584	3.640723	3.984176	4.021134
% tiempo remolcador remolcando	18.100248	18.126814	18.136642	18.147490
% tiempo puntos atraque libres	13.025707	34.647606	47.665897	94.769493
% tiempo puntos atraque ocupados (sin carga)	0.960067	0.724439	0.586037	0.059817
% tiempo puntos atraque ocupados (cargando)	86.014229	64.627922	51.748062	5.170702

Como podemos observar, los resultados van en consonancia a la cantidad de puntos de atraque que hay. El número medio de barcos que hay en la cola de llegadas se aproxima cada vez más a 0, o el porcentaje de tiempo de puntos de atraque libres es casi del 100 %. Esto se debe a que los barcos apenas tendrán que esperar a ser colocados, porque siempre va a haber un lugar donde ponerlos. También se aumentará el tiempo en que están estos puntos libres ya que no siempre

va a existir una cantidad de barcos muy grande en espera como para tenerlos llenos constantemente.

Si nos fijamos en la columna del 50, nos podemos dar cuenta de que la media de barcos en cola no mejora mucho más, mientras que varios de los parámetros obtenidos adquieren valores exagerados (lo lógico por la cantidad excesiva de puntos). Lo que nos dice esto es que no merece la pena aumentar más de 4 o 5 el número de puntos de atraque, ya que el cuello de botella, una vez llegado a este punto, estará en los reolcadores, las tormentas o simplemente en la aleatoriedad del sistema.

Capítulo 3

Análisis de Salidas y Experimentación

3.1. ¿Cuánto hay que simular?

3.2. Intervalos de confianza

3.3. Comparación de más de dos sistemas