Demonstrador Animado das Técnicas de Programação de Simuladores de Eventos Discretos, utilizando o MATLAB®

Elizabeth Fernandez, Rui Valadas

***Resumo* – Este artigo descreve um demonstrador animado concebido para ilustrar as técnicas de programação de um modelo de simulação de eventos discretos. O demonstrador foi desenvolvido no MATLAB e a ilustração é feita com base num modelo M/M/1.**

***Abstract* – This paper describes an animated demo that ilustrates the programming techniques for discrete-event simulators. The demo is based on an M/M/1 queuing system and was developed using MATLAB.**

I. INTRODUÇÃO

A *simulação* em computador consiste na reprodução da operação de um *sistema* realizada com o objectivo de avaliar diferentes características desse sistema. Na prática, a forma como o sistema é modelado depende das características deste que se pretendem analisar. Por exemplo, se relativamente à cantina da Universidade de Aveiro, quisermos estimar o tempo médio que decorre desde que um aluno chega à bicha até que é servido por uma das senhoras, o modelo deve incluir descrições dos processos de chegada dos alunos à cantina e do serviço efectuado pelo conjunto das senhoras. No entanto, se quisermos também determinar o número de mesas necessário para garantir que a probabilidade de um aluno não encontrar um lugar vago após ser servido seja reduzida, o número de lugares e o processo que descreve o tempo de cada refeição devem também ser incluídos no modelo do sistema.

O *estado* do sistema, num dado instante de tempo, pode ser caracterizado através de variáveis de estado. Exemplos de variáveis de estado no caso da cantina, são o número de senhoras ocupadas (note que podem haver senhoras desocupadas caso, momentaneamente, não haja alunos na bicha) e o número de alunos na bicha. Podem distinguir-se dois tipos de sistemas: *discretos*, em que as variáveis de estado mudam instantaneamente em pontos separados no tempo e *contínuos*, em que as variáveis de estado mudam continuamente no tempo.

A simulação de eventos discretos (*discrete-event simulation*) [1] [2] [3] consiste na modelação da evolução temporal de um sistema, através de uma representação na qual as variáveis de estado do sistema mudam de valor em pontos separados no tempo. Estes instantes de tempo são aqueles em que ocorre um *evento*. Um evento é então definido como uma ocorrência instantânea que pode alterar o estado de um sistema.

A simulação de eventos discretos encontra aplicação em áreas tão diversificadas como sejam as redes de telecomunicações, as arquitecturas de computadores, os sistemas de tráfego rodoviário e os sistemas fabris.

A programação de simuladores de eventos discretos pode ser feita recorrendo a linguagens especializadas, designadas por linguagens de simulação, de que são exemplos, a SIMAN, a GPSS, a SLAM II e a SIMSCRIPT II.5. Como é natural, estas linguagens simplificam significativamente a tarefa de programar um simulador de eventos discretos. No entanto, no contexto do ensino da simulação de eventos discretos é importante dar a conhecer ao aluno as técnicas de programação que recorrem às linguagens de programação tradicionais (FORTRAN, PASCAL, C). Isto porque as linguagens de simulação escondem do programador o processo de gestão dos eventos, o que dificulta a percepção das consequências (por exemplo, em termos do tempo de simulação) de programar um modelo, utilizando mais eventos do que os estritamente necessários.

O MATLAB pode ser utilizado como ambiente de desenvolvimento, uma vez que dispõe das facilidades necessárias para a programação de modelos simples.

Este artigo descreve um demonstrador animado concebido para ilustrar as técnicas de programação de um modelo de simulação de eventos discretos. A ilustração é feita com base num modelo M/M/1 (no caso da cantina, o modelo M/M/1 corresponderia a um sistema onde as chegadas de alunos são de Poisson e existe apenas uma senhora que serve com um tempo exponencialmente distribuído). Na secção II são descritas as técnicas de programação para simuladores de eventos discretos, utilizando uma linguagem de programação tradicional. Na secção III é apresentado o demonstrador animado para o MATLAB. Na secção IV apresentam-se as rotinas utilizadas para programar o simulador.

II. TÉCNICAS DE PROGRAMAÇÃO

Os elementos principais de uma simulação de eventos discretos são os eventos e as variáveis. As variáveis são

normalmente de três tipos: relógio de simulação, contadores estatísticos e variáveis de estado. O *relógio de simulação* é a variável que indica, em cada instante, o tempo de simulação. Em geral, não existe qualquer relação entre o tempo de simulação e o tempo necessário para executar a simulação em computador. Os *contadores estatísticos* armazenam a informação estatística relativa ao desempenho do sistema. As *variáveis de estado* descrevem o estado do sistema. Os instantes de ocorrência dos eventos futuros são armazenados numa lista designada por *lista de eventos*. Nesta lista os eventos são ordenados por instante de ocorrência.

O programa de simulação determina qual o próximo evento da lista e avança o relógio de simulação para o instante de ocorrência desse evento. Executa então as acções associadas a esse evento (por exemplo, programação de novos eventos, actualização das variáveis de estado e dos contadores estatísticos) e volta a consultar a lista de eventos para avançar o relógio de simulação para o próximo evento.

Um programa de simulação tem normalmente os seguintes componentes:

- **Estado do sistema**: conjunto de variáveis de estado necessárias para descrever o sistema num dado instante de tempo.
- **Relógio de simulação**: variável que indica o tempo de simulação.
- **Contadores estatísticos**: variáveis utilizadas para armazenar a informação estatística acerca do desempenho do sistema.
- **Rotina de iniciação**: subprograma que inicia o modelo de simulação no instante de tempo zero.
- **Rotina de temporização**: subprograma que determina qual o próximo evento da lista de eventos e que avança o relógio de simulação para o instante de ocorrência desse evento.
- **Rotina do evento**: subprograma que actualiza as variáveis de estado quando ocorre determinado tipo de evento (existe uma rotina por cada tipo de evento).
- **Livrarias**: conjunto de sub-rogarmos utilizados para gerar observações aleatórias de distribuições probabilísticas.
- **Gerador do relatório**: subprograma que calcula (a partir dos contadores estatísticos) as estimativas das medidas de desempenho desejadas e produz um relatório quando a simulação termina.
- **Programa principal**: subprograma que invoca a rotina de temporização para determinar o próximo acontecimento e que transfere o controle para a correspondente rotina do acontecimento para actualização do estado do sistema. O programa principal pode também controlar o fim da simulação e invocar o gerador do relatório quando a simulação termina.

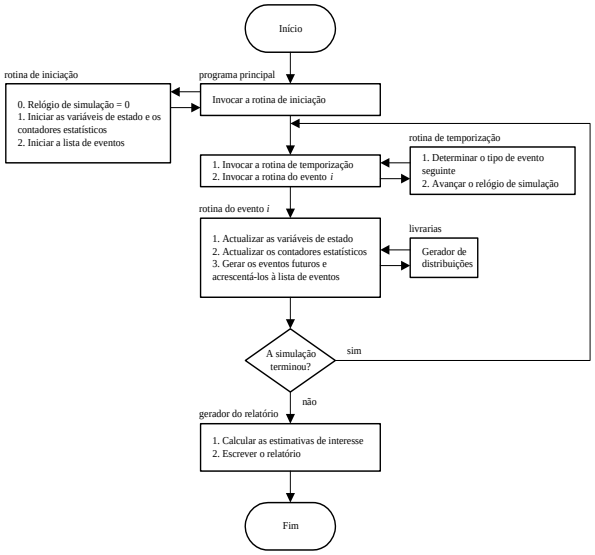

Figura 1: Interacção dos componentes de um modelo de simulação.

A Figura 1 ilustra a forma como interagem os componentes de um programa de simulação.

III. DESCRIÇÃO DO DEMONSTRADOR

A. Modelo

O demonstrador animado concebido para ilustrar as técnicas de programação de simuladores de eventos discretos é baseado num sistema M/M/1. Neste caso, os instantes entre chegadas de clientes, C_1, C_2, …, são variáveis aleatórias (v.a.) independentes e exponencialmente distribuídas, os tempos de serviço dos sucessivos clientes, S_1, S_2, …, são também v.a. independentes e exponencialmente distribuídas, sendo ainda os instantes entre chegadas e os tempos de serviço independentes entre si.

Um cliente que chega ao sistema e encontra o servidor livre entra imediatamente em serviço; se encontrar o servidor ocupado junta-se a uma fila de espera; logo que um cliente complete o seu serviço, entra em serviço o próximo cliente da fila de espera (se existir algum). Admite-se que a ordenação dos clientes na fila de espera é do tipo *First-In First-Out* (FIFO).

A simulação começará no instante t=0, no qual nenhum cliente se encontra no sistema. A chegada do primeiro cliente ocorrerá em $t=C_1$. O sistema será simulado até que um número fixo e pré-definido de n clientes completem os seus atrasos na fila de espera, ou seja, a simulação terminará quando o n-ésimo cliente entrar em serviço.

B. Medidas de desempenho

Neste demonstrador, o sistema M/M/1 é caracterizado através de três medidas de desempenho: atraso médio na

fila de espera, número médio de clientes na fila de espera e taxa de utilização média do servidor.

O atraso médio na fila de espera, *d(n)*, é estimado através de

$$\hat{d}(n) = \frac{\sum_{i=0}^{n} D_i}{n}$$

onde D_1, D_2, …, D_n são os atrasos de cada cliente na fila de espera observados pelo simulador.

Para estimar o número médio de clientes na fila de espera, *q(n)*, define-se a função *Q(t)* como sendo o número médio de clientes na fila de espera no instante *t*, para $t \geq 0$, e *T(n)* o tempo requerido para observar os *n* atrasos na fila de espera. Uma estimativa de *q(n)* é dada por

$$\hat{q}(n) = \frac{\int_0^{T(n)} Q(t)dt}{T(n)}$$

Note-se que o numerador de $\hat{q}(n)$ representa a área sob a curva *Q(t)* entre o início e o fim da simulação.

A taxa de utilização média do servidor, *u*(*n*), pode ser vista como a proporção do tempo de simulação em que o servidor está ocupado. Para estimar a taxa de utilização do servidor define-se a função

$$B(t) = \begin{cases} 1 \text{ se servidor ocupado no instante } t \\ 0 \text{ se servidor livre no instante } t \end{cases}$$

Agora, *û(n)* pode ser considerado como a proporção de tempo em que *B(t)* é igual a 1, ou seja,

$$\hat{u}(n) = \frac{\int_0^{T(n)} B(t)dt}{T(n)}$$

Tal como anteriormente, o denominador de *û(n)* corresponde à área sob a curva *B(t)* entre o início e o fim da simulação.

C. Eventos, variáveis de estado e contadores estatísticos

Para simular este sistema só é necessário conhecer o estado do sistema após as chegadas e as partidas de clientes. Assim, os eventos necessários para programar este sistema são: a chegada de um cliente e a partida de um cliente.

As variáveis de estado necessárias para estimar as medidas de desempenho descritas anteriormente são: o `estado do servidor` (livre ou ocupado), o `número de clientes na fila de espera`, o `instante de chegada` de cada cliente à fila de espera e o `instante do último evento`. O `estado do servidor` corresponde, em cada instante, a *B*(*t*); o `número de clientes na fila de espera` corresponde, em cada instante, a *Q*(*t*); o `instante de chegada` de cada cliente permite calcular o atraso na fila de espera.

Os contadores estatísticos são o `número de atrasos`, o `atraso total`, a `área sob Q(t)` e a `área sob B(t)`.

D. Interface com o utilizador

O demonstrador consiste numa interface gráfica onde se mostram os parâmetros de entrada, o relógio de simulação, as variáveis de estado, a lista de eventos, as curvas *Q*(*t*) e *B*(*t*), as estimativas das medidas de desempenho e uma representação gráfica animada da fila de espera e do servidor, conforme representado na Figura 2. Todos os valores são actualizados imediatamente após o processamento de cada evento. A passagem dos clientes pelo sistema é animada da seguinte forma: os clientes são representados por pequenas superfícies quadradas; a chegada de um cliente é representada através do deslocamento no monitor, da esquerda para a direita, de uma dessas superfícies. Se o servidor não estiver ocupado, a superfície desloca-se para dentro da moldura azul que representa o servidor, e a moldura muda a sua cor para vermelho; caso contrário junta-se a uma fila de espera. A partida de um cliente é representada através do deslocamento no monitor, da esquerda para a direita, da superfície que terminou o seu serviço.

O utilizador pode introduzir os parâmetros de entrada (`taxa de chegada`, `taxa de serviço` e `número de atrasos requeridos`) ou pode optar por valores por defeito. Tem também a opção de executar o programa evento-a-evento.

E. Execução passo-a-passo

No parágrafos seguintes iremos explicar detalhadamente a evolução do modelo de simulação nos instantes de tempo t=0, t_1, t_2, …, t_{10} nos quais são observados 4 atrasos na fila de espera, utilizando, para o efeito, o demonstrador desenvolvido em MATLAB®. Assumiremos para instantes entre chegadas e tempos de serviço dos clientes os seguintes valores: C_1=0.6354, C_2=0.3988, C_3=0.9586, C_4=2.7054, C_5=0.8735, C_6=0.0721, C_7=0.1670, C_8=0.6407; S_1=4.3805, S_2=0.3382, S_3=0.4765, S_4=2.1479. Note-se que, num programa de simulação, estes valores são gerados por geradores de números aleatórios.

Iniciação (t = 0, Figura 2). A simulação começa e a rotina que inicia as variáveis necessárias à simulação do modelo é invocada. Inicialmente o sistema está vazio e o servidor está livre. As variáveis de estado que caracterizam esta situação são iniciadas: o `estado do servidor` é zero, *B*(*t*)=0, e o `número de clientes na fila de espera` é zero, *Q*(*t*)=0. Os contadores estatísticos `número de atrasos`, `atraso total` e a `área sob Q(t)`, e `a área sob B(t)`, são todos colocados à zero. O `relógio` de simulação é colocado a

zero e à `lista de eventos` são atribuídos os instantes de tempo da próxima ocorrência de cada um dos tipos de eventos. O instante de tempo da primeira chegada é $0+C_1=0.6354$. Uma vez que não existem clientes em serviço o evento partida é eliminado, colocando-o a infinito para garantir que o primeiro evento seja uma chegada. Após a iniciação do sistema a rotina de temporização que determina o instante de tempo e o tipo do próximo evento é invocada. Visto que $0.6354<\infty$, o próximo evento é uma chegada e ocorrerá no instante 0.6354, pelo que a rotina avança o relógio de simulação para este instante de tempo.

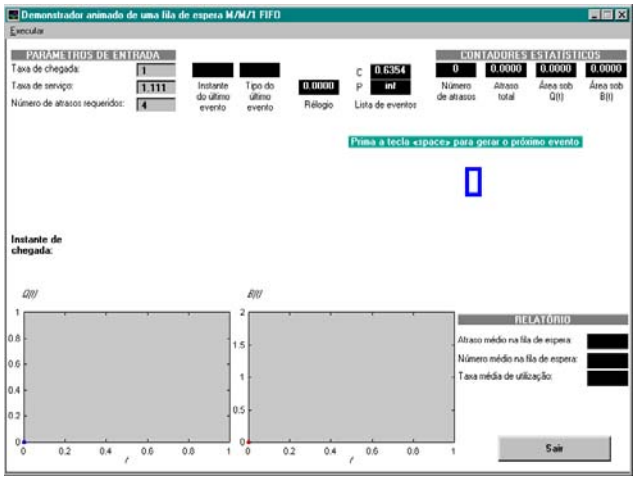

Figura 2: Iniciação do modelo do sistema.

Chegada do cliente 1 (t = 0.6354). A Figura 3 mostra as alterações feitas ao sistema até ao instante imediatamente antes ao processamento do evento. Uma vez que o cliente encontra o servidor livre, ele inicia imediatamente o seu serviço e tem um `atraso na fila de espera` nulo, $D_1=0$. O `relógio` foi avançado para o instante de tempo actual e a `lista de eventos` actualizada. A próxima chegada ocorrerá dentro de $C_2=0.3988$ unidades de tempo, ou seja, em $t=0.6354+0.3988=1.0342$, e a próxima partida será dentro de $S_1=4.3805$ unidades de tempo, ou seja, em $t=0.6354+4.3805=5.0159$. O `número de atrasos` é incrementado de 1 e $D_1=0$ é adicionada ao `atraso total`. A `área sob` $Q(t)$ é actualizada adicionando ao seu valor anterior, 0, o produto do `número de clientes na fila de espera`, 0, com o intervalo de tempo desde o último evento que é calculado subtraindo o valor do `relógio` de simulação, 0.6354, ao valor do instante de `tempo do último evento`, 0, isto é, 0.6354-0=0.6354. Analogamente, a `área sob` $B(t)$ é actualizada, adicionando ao seu valor anterior, 0, o produto do `estado do servidor`, 0, com o intervalo de tempo desde o último evento. Tal como anteriormente, é invocada a rotina de temporização que escolhe a menor das quantidades da `lista de eventos` e determina que o próximo evento será outra chegada que ocorrerá no instante de tempo t=1.0342 e avança o `relógio` de simulação para esse instante.

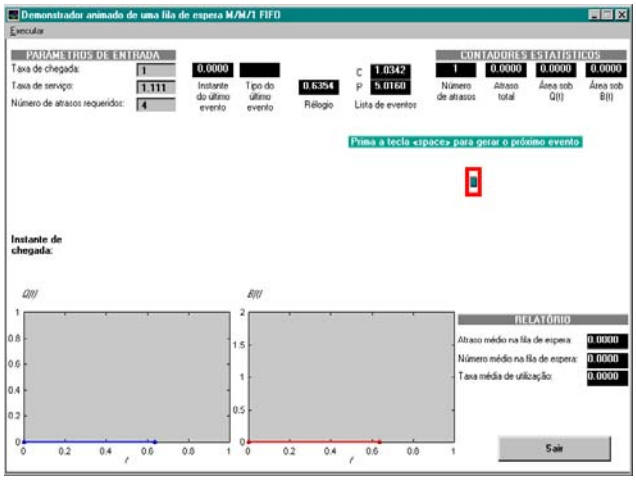

Figura 3: Caracterização do sistema do instante de tempo t=0.6354.

Chegada do cliente 2 (t = 1.0342). A Figura 4 mostra as alterações feitas ao sistema até o instante imediatamente antes do processamento do evento. Este cliente encontra o servidor ocupado e a fila de espera vazia pelo que ocupa a primeira posição na fila de espera e o seu `instante de chegada`, t=1.0342, é armazenado para posterior cálculo do atraso na fila de espera, D_2. O instante de tempo do próximo evento chegada é incrementado de $C_3=0.9586$ unidades de tempo, e será $t=1.0342+0.9586=1.9928$; o próximo evento será esta chegada, uma vez que, a partida do cliente em serviço será apenas no instante t=5.0159. O `instante do último evento` é actualizado para 0.6354. A `área sob` $Q(t)$ é incrementada de 0 resultante do produto de $Q(t)=0$ com o intervalo de tempo desde o último evento, 1.0342-0.6354=0.3988. A `área sob` $B(t)$ é incrementado de 0.3988 resultante do produto de $B(t)=1$ com 0.3988.

Chegada do cliente 3 (t = 1.9928). A Figura 5 mostra o estado do sistema até o instante imediatamente antes do

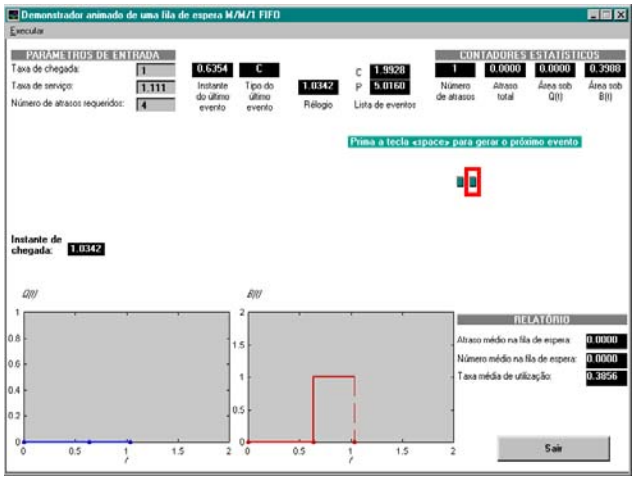

Figura 4: Caracterização do sistema no instante de tempo t=1.0342.

processamento do evento. O cliente encontra o servidor ocupado e junta-se à fila de espera. O seu `instante de chegada`, 1.9928, tal como o sucedido para o cliente anterior, é armazenado com o propósito de calcular o seu

atraso na fila de espera, D_3. O instante de tempo da próxima chegada é incrementado de C_3=2.7054 e actualizado para t=1.9928+2.7054=4.6982 e o instante de tempo da próxima partida mantém-se inalterado. O `instante do último evento` é actualizado para 1.0342. Os acumuladores das áreas são actualizados adicionado-lhes o produto de 1 (o valor de $B(t)$ e $Q(t)$ no instante t=1.0342) com o tempo desde o último evento, 1.9928-1.0342=0.9586.

Figura 5: Caracterização do sistema no instante de tempo t=1.9928.

Chegada do cliente 4 (t = 4.6982) . As alterações ao estado do sistema inerentes a esta chegada são semelhantes àquelas feitas aquando da chegada do cliente 3 no instante de tempo t=1.9928, discutidas anteriormente. O próximo evento será uma partida no instante de tempo t=5.0159.

Partida do cliente 1 (t = 5.0159). A Figura 6 mostra o sistema e a sua caracterização imediatamente após o processamento do evento. O servidor mantém-se ocupado, uma vez que o cliente 2 entra em serviço; os restantes 2 clientes movem-se um lugar na fila de espera. O cliente que acaba de entrar em serviço requer um tempo de serviço S_2=0.3382 pelo que o instante de tempo da próxima partida é actualizado para t=5.3541. O instante de tempo da próxima chegada mantém-se inalterado, uma vez que os eventos de chegada são agendados apenas quando se dão chegadas de clientes ao sistemas. O `atraso total` é actualizado, adicionando ao seu valor anterior, 0, o atraso que o cliente 2 sofreu na fila de espera, que é dado pela diferença entre o instante de tempo actual e o `instante de chegada` do cliente ao sistema, D_2=5.0159-1.0342=3.9817. O `instante do último evento` é actualizado para 4.6982. O `número de atrasos` é incrementado de 1 e os contadores estatísticos são actualizados, adicionando 3×(5.0159-4.6982)=0.9531 à `área sob` $Q(t)$ e 1×(5.0159-4.6982)=0.3177 à `área sob` $B(t)$. A rotina de temporização determina que o próximo evento é uma partida no instante de tempo t=5.3541.

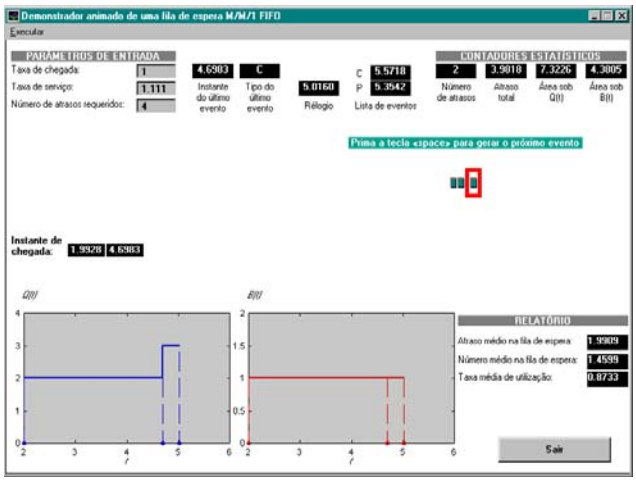

Figura 6: Caracterização do sistema no instante de tempo t=5.0159.

Partida do cliente 2 (t = 5.3541). As alterações ao estado do sistema inerentes a esta partida são semelhantes àquelas que foram feitas aquando da partida do cliente 1 no instante de tempo t=5.0159, discutidas anteriormente. Note-se que os próximos três eventos são do tipo chegada e ocorrerão nos instantes t=5.5717, 5.6438 e 5.8108.

Chegada dos clientes 5, 6 e 7 (t = 5.5717, 5.6438, 5.8108). As alterações ao estado do sistema inerentes as estas chegadas são semelhantes àquelas que foram feitas aquando da chegada do clientes 3. O próximo evento será uma partida no instante t=5.8306.

Partida do cliente 3 (t = 5.8306). As alterações ao estado do sistema inerentes a esta partida são apresentadas na Figura 7 e são semelhantes àquelas que foram feitas aquando das partidas dos cliente 1 e 2. O `atraso total` é actualizado, adicionando ao seu valor anterior, 7.3430, o atraso que o cliente 4, que acabou de entrar em serviço, sofreu na fila de espera, D_4=5.8306-4.6982=1.1324, para 7.3430+1.1324=8.4754. Com a partida do cliente 3, o cliente 4 deixa a fila de espera e entra em serviço, instante em que o `número de atrasos` é incrementado para 4 e a simulação termina. Neste momento o programa de simulação invoca a rotina que gera o relatório para o cálculo das medidas de desempenhos do sistema: o `atraso médio dos clientes na fila de espera`, $\hat{d}(4)$ = 8.4754/4 = 2.1189, o `número médio de clientes na fila de espera`, $\hat{q}(4)$ = 8.9409/4 = 1.5334 e a `taxa média de utilização do servidor`, $\hat{u}(4)$ = 5.1952/5.8306 = 0.8910.

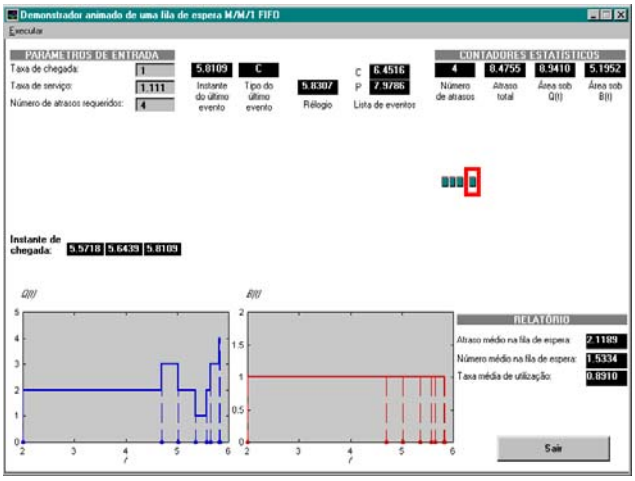

Figura 7: Caracterização do sistema no instante de tempo *t*=5.8306.

IV. PROGRAMA EM MATLAB

Nesta secção faz-se uma descrição da implementação do programa de simulação da fila de espera M/M/1, utilizando o MATLAB.

O programa principal *mm1.m* é formado pelas seguintes rotinas: *inicia.m*, *temp.m*, *chegada.m*, *partida.m*, *relator.m* e *actcont.m*. A Tabela 1 contém uma descrição das principais variáveis e rotinas utilizadas na programação do modelo. O código do programa principal é apresentado na Figura 9 e segue o fluxograma da Figura 8. As constantes `Ocupado` e `Livre` são definidas para serem utilizadas com a variável `EstServ`, por forma a tornar o código de fácil compreensão. O programa começa por invocar a rotina *inicia* que, tal como o nome indica, inicia as variáveis necessárias à simulação. O ciclo `while` é executado até que um número requerido de `NumAtrasReq` clientes completem a sua passagem pela fila de espera. Esta é a condição de paragem da simulação. Dentro deste ciclo a rotina *temp* é invocada para determinar o tipo do próximo evento e avançar o relógio de simulação para o instante em que esse ocorrerá. Antes do processamento deste evento a rotina *actcont* é invocada para actualizar as áreas sob as curvas *B(t)* e *Q(t)*. O comando `if`, cujo teste é baseado na variável `TipProxEvent` que assume o valor 1 quando o evento é do tipo chegada e o valor 2 quando é do tipo partida, passa o controlo à rotina do evento adequado. Após o ciclo `while` terminar a rotina *relator* é invocada e a simulação termina.

O código da rotina *inicia* é apresentado na Figura 10. Esta rotina é invocada apenas uma vez no início da simulação, avançando o relógio de simulação para o instante de tempo de chegada do primeiro cliente, `TempProxEvent(1)`. Este instante é determinado adicionando uma v.a. exponencial com média 1/`lambda` ao relógio de simulação, `Relogio=0`. Para esse efeito é utilizada a função *exprnd.m* da *Statistics Toolbox* do MATLAB que gera v.a. exponencialmente distribuídas.

Tabela 1: Rotinas e variáveis do programa em MATLAB do modelo fila de espera M/M/1.

Rotina	**Definição**
inicia	Rotina de iniciação das variáveis de simulação
temp	Rotina que actualiza o relógio de simulação
chegada	Rotina do evento chegada
partida	Rotina do evento partida
relator	Rotina que gera o relatório final
actcont	Rotina que actualiza os contadores estatísticos
Variável	**Definição**
Parâmetros de entrada:	
`lambda`	Taxa de chegada dos clientes ao sistema
`niu`	Taxa de serviço
NumAtrasReq	Número total de clientes atrasados a serem observados
Variáveis auxiliares:	
`Livre`	Estado livre do servidor (= 0)
`Ocupado`	Estado ocupado do servidor (=1)
`TipProxEvent`	Tipo do próximo evento (determinado pela rotina *temp*)
`TempDesdUltEvent`	Armazena o intervalo de tempo desde que ocorreu o último evento (utilizado pela rotina *actcont*)
`Relogio`	Relógio de simulação
`Atraso`	Atraso na fila de espera de um cliente
`TotalAtras`	Atraso total dos clientes na fila de espera
Variáveis de estado:	
`EstServ`	Estado do servidor (0 se livre, 1 se ocupado)
`NumNaQ`	Número actual de clientes na fila de espera
`TempUltEvent`	Instante de tempo do último evento
`InstCheg`	Vector que armazena o instante de chegada dos clientes ao sistema
Contadores estatísticos:	
`NumClieAtras`	Número de clientes que completaram os seus atrasos na fila de espera
`AreaNumNaQ`	Área sob a curva *B*(*t*)
`AreaEstServ`	Área sob a curva *Q*(*t*)
Lista de eventos:	
`TemProxEvent(1)`	Instante de tempo de ocorrência do próximo evento chegada
`TemProxEvent(2)`	Instante de tempo de ocorrência do próximo evento partida
Parâmetros de saída:	
`NumMedNaQ`	Número médio de clientes na fila de espera
`AtrasMedNaQ`	Atraso médio do clientes na fila de espera
`TempMedUtilServ`	Tempo médio de utilização do servidor

Visto que, inicialmente, não existem clientes no sistema o próximo evento partida é eliminado, fazendo

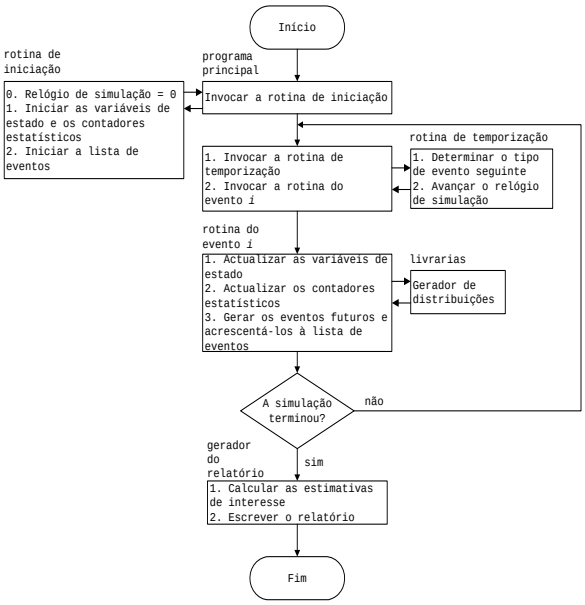

Figura 8: Fluxograma do programa principal.

`TempProxEvent(2)=inf`, de modo a garantir que o primeiro evento seja uma chegada.

A rotina *temp* é apresentada na Figura 11 e tem como função avançar o relógio de simulação para o instante de tempo do próximo evento. O programa compara `TempProxEvent(1)` e `TempProxEvent(2)` e atribui à variável `Relogio` o instante de tempo do evento cujo instante de ocorrência é menor e à variável `TipProxEvent` o tipo desse evento. Para esse efeito é utilizada a função *min.m* do MATLAB que calcula o mínimo de um vector e retorna o seu valor e o respectivo índice no vector. No caso em que as quantidades `TempProxEvent(1)` (uma chegada) e `TempProxEvent(2)` (uma partida) são iguais o próximo evento é do tipo chegada. No entanto, note-se que da próxima vez que a rotina *temp* for invocada o próximo evento é do tipo partida, ou seja, a rotina é invocada duas vezes consecutivas sem que o relógio de simulação seja avançado.

O código da rotina do evento *chegada* está representado na Figura 13 e segue o fluxograma da Figura 12. Note-se que variável `Relogio` armazena o instante de chegada do cliente que acabou de chegar ao sistema.

A rotina do evento *partida* é apresentada na Figura 15 e segue o fluxograma da Figura 14. Esta é invocada pelo programa principal quando um cliente completa o seu serviço e, consequentemente, ocorre a sua partida. A instrução `TempProxEvent(2)=inf` evita que o programa entre em ciclo infinito quando se verifica que, após a partida de um cliente, a fila de espera fica vazia.

O atraso do cliente na fila de espera é obtido subtraindo `InstCheg(1)` ao valor actual do relógio de simulação, `Relogio`.

A instrução `InstCheg(1)=[]` remove o primeiro cliente, avançando os restantes (se existir algum) um lugar na fila de espera, assegurando que, o instante de chegada do próximo cliente a entrar em serviço, esteja sempre armazenado em `InstCheg(1)`. Uma alternativa a esta instrução é a utilização de um ciclo `for`. No entanto esta solução, no caso de existirem muitos clientes na fila de espera, resulta em tempos computacionais consideravelmente elevados.

```
function mm1(lambda, niu, NumAtrasReq)
% MM1       fila de espera M/M/1 - sistema de uma
% fila de espera de um servidor, com intervalos entre
% chegadas exponenciais IID com média 1/lambda,
  tempos
% de serviço exponenciais IID com média 1/niu, e
% clientes servidos em modo First-In, First-Out(FIFO)
%
% Sintaxe:  mm1(lambda, niu, NumAtrasReq)
%       lambda = taxa de chegada
%       niu = taxa de serviço
%       NumAtrasReq = Número de Atrasos Requeridos

Ocupado = 1;
Livre = 0;

% Iniciar a simulação
inicia

% Executar a simulação enquanto forem necessários
% atrasos
while (NumClieAtras < NumAtrasReq),

  % Determinar o próximo evento
  temp

  % Actualizar os tempos médios dos contadores
  % estatísticos
  actcont

  % Invocar a rotina evento adequado
  if (TipProxEvent == 1),
    chegada
  else
    partida
  end
end

% Invocar o gerador de relatório e acabar simulação
relator
```

Figura 9**:** Código MATLAB do programa principal do modelo da fila de espera M/M/1, *mm1*.

O código da rotina *relator*, invocada quando o programa principal termina a simulação, é apresentado na Figura 16. O atraso médio é calculado, dividindo o total de atrasos pelo número de clientes cujos atrasos na fila de espera foram observados e o número médio de atrasos na fila de espera é obtido, dividindo a área sob *Q(t)* pelo

```
% INICIA     Rotina que INICIA as variáveis de
% simulação

% Iniciar o Relógio de Simulação
Relogio = 0;

% Iniciar as Variáveis de Estado
EstServ = Livre;         % Estado do Servidor
NumNaQ = 0;              % Número de Clientes da
  fila
                         % de espera
TempUltEvent = 0;        % Tempo do Último Evento
InstCheg = [];           % Instante de Chegada

% Iniciar os Contadores Estatísticos
NumClieAtras = 0;% Número de Clientes  Atrasados
TotalAtras = 0;  % Total de Atrasos
AreaNumNaQ = 0;  % Área do Número de Clientes na fila
                 % de espera
AreaEstServ = 0; % Área do Estado do Servidor

% Iniciar a lista de eventos.
% Desde que não haja nenhum cliente presente o evento
% partida é eliminado
TempProxEvent(1) = Relogio + exprnd(1/lambda);
TempProxEvent(2) = inf;
```

Figura 10: Código MATLAB da rotina *inicia*.

valor final do relógio de simulação, `Relogio`. A taxa de utilização do servidor é calculada, dividindo a área sob *B(t)* pelo `Relogio`. Todas as medidas de desempenho são mostrada no écran. É mostrado também o valor de `Relogio` por forma a sabermos quanto tempo é necessário para observar `NumAtrasReq` atrasos de clientes na fila de espera.

```
% TEMP       Rotina de TEMPorização, actualiza o
% relógio de simulação e determina o tipo do próximo
% evento

[Relogio,TipProxEvent] = min(TempProxEvent);
```

Figura 11: Código MATLAB da rotina *temp*.

A rotina *actcont* apresentada na Figura 17 é invocada imediatamente antes de processar cada evento (de qualquer tipo) e actualiza as áreas sob as funções *Q(t)* e *B(t)* necessárias ao cálculo das médias temporais. Para esse efeito é calculado o intervalo de tempo desde o último evento. Este é atribuído a variável `TempDesdUltEvent`. A variável `TempUltEvent` é actualizada, atribuindo-lhe o valor actual do relógio de simulação. *Q(t)* é aumentado da área do rectângulo de largura `TempDesdUltEvent` e altura `NumNaQ`. A área sob *B(t)* é aumentada da área do rectângulo de largura `TempDesdUltEvent` e altura `EstServ`.

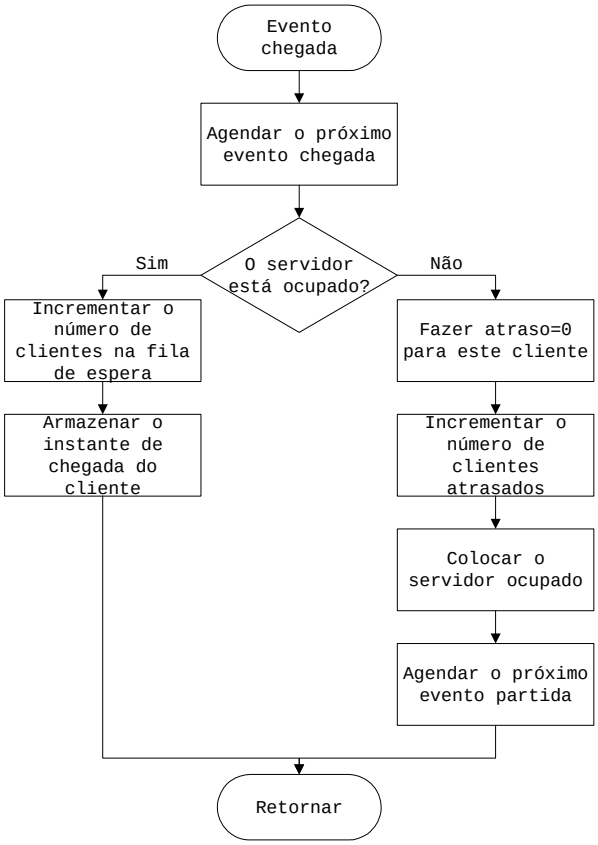

Figura 12: Fluxograma da rotina do evento *chegada*.

```
% CHEGADA        Rotina do evento CHEGADA

% Agendar o instante da próxima chegada
TempProxEvent(1) = Relogio + exprnd(1/lambda);

% Verificar se o Servidor está ocupado
if (EstServ == Ocupado),
  % O Servidor está ocupado, então incrementa o
  número
  % de clientes da fila de espera
  NumNaQ = NumNaQ + 1;

  % É necessário armazenar o instante de chegada do
  % cliente no fim da fila de espera
  InstCheg(NumNaQ) = Relogio;
else
  % Incrementar o número de clientes atrasados e
  % colocar o Servidor ocupado
  NumClieAtras = NumClieAtras + 1;
  EstServ = Ocupado;

  % O Servidor está livre, então o cliente que chegar
  % tem um tempo de atraso nulo
  Atraso = 0;

  % Agendar o instante da próxima Partida
  TempProxEvent(2) = Relogio + exprnd(1/niu);
end
```

Figura 13: Código MATLAB da rotina do evento *chegada*.

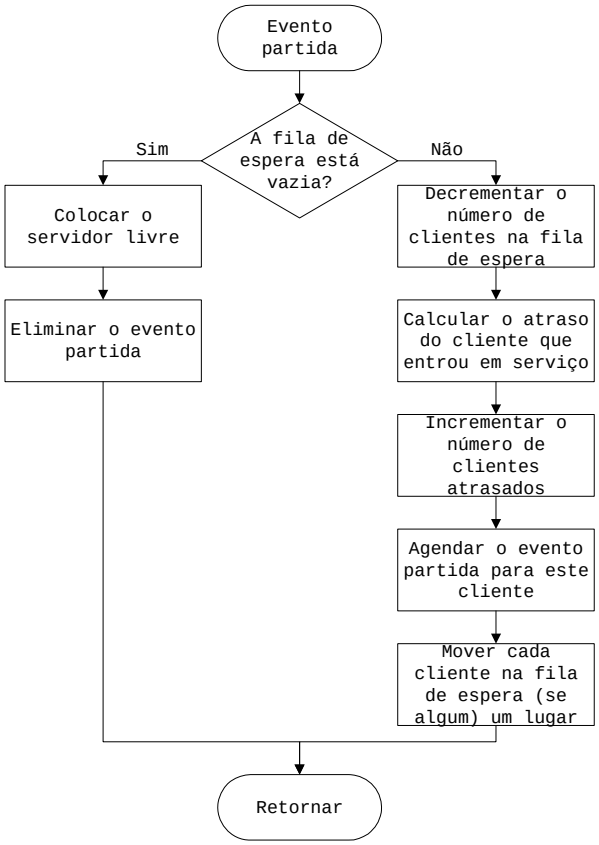

Figura 14: Fluxograma da rotina do evento *partida*.

```
% PARTIDA        Rotina do evento PARTIDA

% Verifica se a fila está vazia
if (NumNaQ == 0),
  % A fila está vazia, então coloca o Servidor livre
  e
  % elimina o evento partida
  EstServ = Livre;
  TempProxEvent(2) = inf;
else
  % A fila não está vazia, então decrementa o número
  % de clientes na fila de espera
  NumNaQ = NumNaQ - 1;

  % Incrementar o número de clientes atrasados e o
  % instante de partida
  NumClieAtras = NumClieAtras + 1;
  TempProxEvent(2) = Relogio + exprnd(1/niu);

  % Calcular o atraso do cliente que está começando o
  % serviço
  Atraso = Relogio - InstCheg(1);
  % Actualizar o acumulador do atraso total
  TotalAtras = TotalAtras + Atraso;

  % Subir cada cliente na fila (se existir algum) um
  % lugar na fila de espera
  InstCheg(1) = [];
end
```

```
% RELATOR        Rotina que gera o RELATÓRio final

% Calcular e escrever as estimativas das medidas de
% desempenho desejadas:

% Atraso Médio na Fila de Espera
AtrasMedNaQ = TotalAtras / NumClieAtras;
% Número Médio de atraso na Fila de Espera
NumMedNaQ = AreaNumNaQ / Relogio;
% Tempo Médio de Utilização do Servidor
TempMedUtilServ = AreaEstServ / Relogio;

disp(sprintf('\n\t\t== RELATÓRIO ==\n'));
disp(sprintf('\tAtraso médio na Fila de Espera: %6.3f
  minutos\n', AtrasMedNaQ));
disp(sprintf('\tNúmero médio de Clientes na Fila de
  Espera: %6.3f\n', NumMedNaQ));
disp(sprintf('\tTempo médio de utilização do
  Servidor: %6.3f minutos\n', TempMedUtilServ));
disp(sprintf('\tTempo de simulação: %6.3f
  minutos\n\n', Relogio));
```

Figura 16: Código MATLAB da rotina *relator*.

```
% ACTCONT        ACTualização do tempo médio dos
% CONTadores estatísticos

% Calcular o Tempo Desde o Último Evento
TempDesdUltEvent = Relogio - TempUltEvent;

% Actualizar o Tempo do Último Evento assinalado
TempUltEvent = Relogio;

% Actualizar a área sob a função número clientes na
% fila de espera
AreaNumNaQ = AreaNumNaQ + NumNaQ * TempDesdUltEvent;

% Actualizar a área sob a função estado do servidor
AreaEstServ = AreaEstServ + EstServ*TempDesdUltEvent;
```

Figura 17: Código MATLAB da rotina *actcont*.

V. CONCLUSÕES

Este artigo apresenta uma descrição das técnicas de programação de um modelo de simulação de eventos discretos. É feita uma explicação detalhada da evolução da simulação de um modelo do tipo M/M/1, ilustrada passo-a-passo através de um demonstrador animado desenvolvido no MATLAB.

VI. BIBLIOGRAFIA

[1] A. Law, D. Kelton, “Simulation, modeling and analysis”, McGraw-Hill, 1991.

[2] J. Banks, J. Carson, B. Nelson “Discrete-Event System Simulation”, Academic Press, 1996.

[3] S. Ross, “Simulation”, Academic Press, 1997.