Discrete Stochastic Simulation

Pedro Barahona
DI/FCT/UNL

Métodos Computacionais
1st Semestre 2018/2019

Example: Queueing Systems

- We may now address a slightly more complicated queueing system where in addition to one server, there is a buffer of size 1 where requests can be maintained while the server is busy. We keep the same arrival distributions but adopt a different serving time, as follows:
  - One single server
  - Service time following an ***Erlang*** *distribution (2, 1.5);*
  - *One buffer:*
    - *if a request arrives when both the server and the buffer are empty, the request enters the server.*
    - *if a request arrives when the server is full but the buffer is empty, the request stays in the buffer, until the server is free.*
    - *if a request arrives when the buffer is full the request is rejected.*
  - Requests arrive with an exponential distribution with mean time of 3 minutes.
- Again, simulation (for a sufficient large time) may be used to estimate the behaviour of this system.

Example: Queueing Systems

- We will be interested in obtaining the likely behaviour of the system, namely
  - What is the percentage of time the server is busy.
  - What is the percentage of requests that are rejected;
  - *What is the average waiting time of a request in the queue.*

- Now the state, **s**, of the system should indicate not only whether a request is being served, and at what time it arrived, but also whether a request is in the queue, and and at what time it arrived. Hence, **s** may be encoded as a structure with three fields:
  - **s.latest_system_time (lst)**: the time elapsed since the beginning of the simulation;
  - **s.entry_server_time (est):** a number specifying whether the server is busy. If the server is busy it should represent the time the request has been accepted. Otherwise, the value is encoded as +inf.
  - **s.entry_buffer_time (ebt):** a number specifying whether a request is in the the queue, represent the time the request was been accepted (Otherwise , the value is encoded as +inf).

Example: Queueing Systems

- The event, **e**, should still indicate the timing of the next arrival of a request, as well as the timing of the next completion of a served request:
  - **e.next_arrival_time (nat)**: the timing of the next arrival of a request;
  - **e.next_exit_time (net)**: the timing of the next exit from the server.
  - If the server is empty, *and the buffer is also empty*, the **next_exit_time** should be encoded as **+inf.**
  - *However, if the server becomes empty, but the buffer is not empty, the request from the buffer is moved to the sderver a new* ***next_exit_time*** *should be computed.*

- To monitor the system a new variable should maintain the timing when the buffer has been busy (to compute the mean waiting time), and **m** may be encoded as a structure with 4 fields:
  - **m.server_busy_time (sbt)**: the time the server has been busy so far;
  - **m.buffer_wait_time (qwt)**: the time requests have been waiting in the queue;
  - **m.number_accepted_requests (nar)**: Number of requests accepted so far;
  - **m.number_rejected_requests (nrr)**: Number of requests rejected so far;

Example: Queueing Systems

- Given the above assumptions the initial state should be encoded as
  - **s.latest_system_time = 0;**
  - **s.entry_server_time: = inf.**
  - **s.buffer_server_time: = inf.**
- The initial events should be as before
  - **e.next_arrival_time = x;**
  - **e.next_exit_time = inf;**

  where x is obtained from the exponential distribution
- The initial monitoring data should be
  - **m.server_busy_time = 0;**
  - **m.buffer_busy_time = 0;**
  - **m.number_accepted_requests = 0**
  - **m.number_rejected_requests = 0.**

Example: Queueing Systems

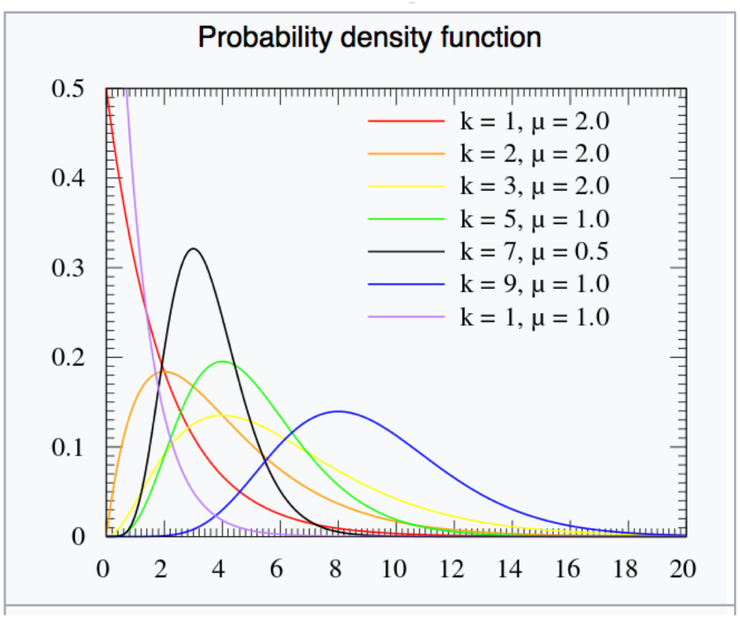
- The stopping condition could be specified as before, namely by allowing the simulation of the system to last until some final_time. i.e. until
    - **s.latest_system_time > final_time**
- Finally, the state transitions can be caused by the arrival of requests or exit from servers, and can be described in the following transition table

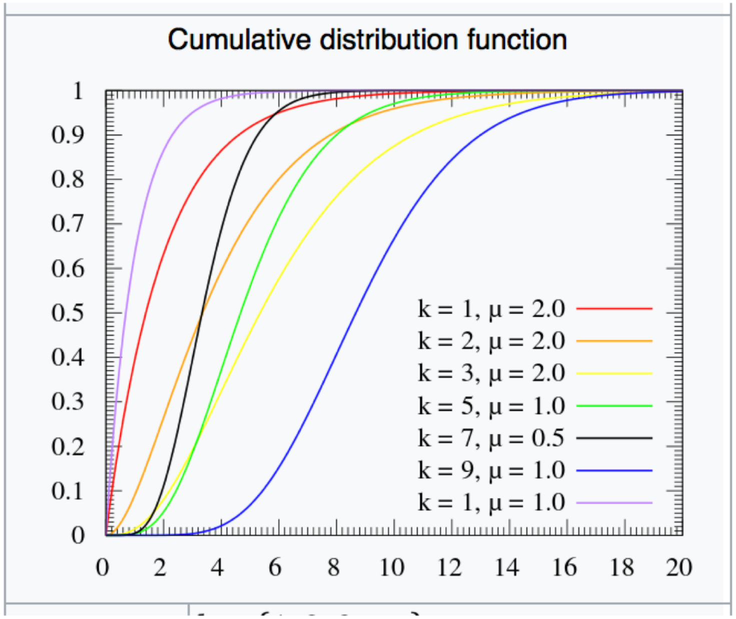
	event		current state			next state			next event		monitor			
	nat	net	lst	est	ebt	lst	est	ebt	nat	net	nar	nrr	sbt	qwt
arrival (server empty, queue empty)	a	inf	-	inf	inf	a	a	inf	exp a	erl a	+1	=	=	=
arrival (server busy, queue empty)	a	b (> a)	-	c	inf	a	c	a	exp a	b	+1	=	=	a
arrival (server busy, queue busy)	a	b (> a)	-	c	d	a	c	d	exp a	b	=	+1	=	=
departure (queue empty)	a	b (< a)	-	c	inf	b	inf	inf	a	inf	=	=	+(b-c)	=
departure (queue busy)	a	b (< a)	-	c	d	b	b	inf	a	erl b	=	=	+(b-c)	+(b-d)

- But before encoding this example, let us analyse the serving times that follow an Erlang(k,m) distribution (with k = 2, m = 1.5).

Erlang distribution

- The Erlang distribution is the distribution of the sum of ***k*** *independent and identically distributed random variables*, each having an *exponential distribution with mean **m***.





- Source: https://en.wikipedia.org/wiki/Erlang_distribution

Erlang distribution

- The Erlang distribution is the distribution of the sum of ***k*** *independent and identically distributed random variables*, each having an *exponential distribution with mean* ***m***.
- Its pdf (probability density function) is the following:

$$\boldsymbol{f(x;k,m)} = \frac{\boldsymbol{x^{k-1}e^{-x/m}}}{\boldsymbol{m^k(k-1)!}}$$

- Hence, a significant difference with respect to the uniform and exponential distribution is that it cannot be generated by the inverse method (that requires obtaining **x** as a function of **f**).
- Hence it can be obtained by the general accept-reject method, assuming that it is truncated at some convenient **x** (for example, $\mathbf{x_{max}}$ **= 10*k*m**) and max value (it depends on **k** and **m**, but for $k > 1$ and $m > 0.2$, $\mathbf{f_{max}}$ **= 2** is a “safe” value).
- Of course, given the definition above it can be simulated as the sequence of **k** exponential distributions, each with a mean **m**.

Erlang distribution

$$f(x;k,m)=\frac{x^{k-1}e^{-x/m}}{m^{k}(k-1)!}$$

- Adopting the accept-reject method the distribution can be obtained by adapting the generic ar function (seen before) to the Erlang pdf, as follows

```
function x = erlang_ar(k,m);
% generates events with an Erlang (k,m) distribution.
% it uses the generic accept-reject method
   accept = false;
   while ! accept
      x = 10 * k * m * rand();           % x = 10*k*m
      r = 2  * rand();                   % fdp < 2
      y = (x^(k-1)*exp(-x/m))/((m^k)*fact(k-1))
      accept = (r <= y)
   end
end
```

- In this case, we generate values of x, up to a maximum 10*k*m. In this range of values for x, the values of the pdf are all below 2 (as discussed)

Erlang distribution

- Since the the Erlang distribution corresponds to the the sum of *k independent and identically distributed random variables*, each having an *exponential distribution with mean m/k*, its generator can be also obtained alternatively as:

```
function x = erlang_sp(k,m);
% generates events with an Erlang (k,m) distribution.
% it takes into account that this distribution
% corresponds to a sequence of k independent
% exponential distibutions with mean m.
   x = 0;
   for i = 1:k
      x = x + expo_distr(m/k);
   end
end
```

Example: Queueing Systems

- Given the above specifications we can now implement the queueing system, with 1 server and one buffer as follows:

```
function s = initial_s1q1_state()
   s.latest_system_time = 0;
   s.entry_server_time = inf;
   s.entry_buffer_time = inf;      % new variable
end
```

```
function e = initial_s1q1_event(mean)
   e.next_arrival_time = expo_distr (mean);
   e.next_exit_time = inf;
end
```

```
function e = initial_s1q1_monitor()
   m.number_rejected_services = 0;
   m.number_accepted_services = 0;
   m.server_busy_time = 0;
   m.queue_wait_time = 0;         % new variable
end
```

Example: Queueing Systems

- The stopping condition emains the same (apart from the signature):

```
function e = stop_s1q1(s,max_t)
    s.latest_system_time > max_t;
end
```

- Finally, transition function should now encode 5 different types of events as described in the previous table

	event		current state			next state			next event		monitor			
	nat	net	lst	est	ebt	lst	est	ebt	nat	net	nar	nrr	sbt	qwt
arrival (server empty, queue empty)	a	inf	-	inf	inf	a	a	inf	exp a	erl a	+1	=	=	=
arrival (server busy, queue empty)	a	b (> a)	-	c	inf	a	c	a	exp a	b	+1	=	=	a
arrival (server busy, queue busy)	a	b (> a)	-	c	d	a	c	d	exp a	b	=	+1	=	=
departure (queue empty)	a	b (< a)	-	c	inf	b	inf	inf	a	inf	=	=	+(b-c)	=
departure (queue busy)	a	b (< a)	-	c	d	b	b	inf	a	erl b	=	=	+(b-c)	+(b-d)

Example: Queueing Systems

	event		current state			next state			next event		monitor			
	nat	net	lst	est	ebt	lst	est	ebt	nat	net	nar	nrr	sbt	qwt
arrival (server empty, queue empty)	a	inf	-	inf	inf	a	a	inf	exp a	erl a	+1	=	=	=
arrival (server busy, queue empty)	a	b (> a)	-	c	inf	a	c	a	exp a	b	+1	=	=	a
arrival (server busy, queue busy)	a	b (> a)	-	c	d	a	c	d	exp a	b	=	+1	=	=
departure (queue empty)	a	b (< a)	-	c	inf	b	inf	inf	a	inf	=	=	+(b-c)	=
departure (queue busy)	a	b (< a)	-	c	d	b	b	inf	a	erl b	=	=	+(b-c)	+(b-d)

```
function [s,e,m] = transition_s1q1(s,e,m,mean,ke,me);

% arrival while server and buffer empty
 if e.next_exit_time == inf && s.entry_buffer_time == inf
      s.latest_system_time = e.next_arrival_time;
      s.entry_buffer_time = e.next_arrival_time;
      e.next_arrival_time = s.latest_system_time + expo_distr(mean);
      e.next_exit_time = s.latest_system_time + erlang_distr(ke,me);
      m.number_accepted_services = m.number_accepted_services + 1;
....

end
```

Example: Queueing Systems

	event		current state			next state			next event		monitor			
	nat	net	lst	est	ebt	lst	est	ebt	nat	net	nar	nrr	sbt	qwt
arrival (server empty, queue empty)	a	inf	-	inf	inf	a	a	inf	exp a	erl a	+1	=	=	=
arrival (server busy, queue empty)	a	b (> a)	-	c	inf	a	c	a	exp a	b	+1	=	=	a
arrival (server busy, queue busy)	a	b (> a)	-	c	d	a	c	d	exp a	b	=	+1	=	=
departure (queue empty)	a	b (< a)	-	c	inf	b	inf	inf	a	inf	=	=	+(b-c)	=
departure (queue busy)	a	b (< a)	-	c	d	b	b	inf	a	erl b	=	=	+(b-c)	+(b-d)

```
function [s,e,m] = transition_s1q1(s,e,m,mean,ke,me);


  ....
% arrival when server busy and buffer empty
 elseif e.next_arrival_time <= e.next_exit_time && ...
                                       s.entry_buffer_time == inf
      s.latest_system_time = e.next_arrival_time;
      s.entry_buffer_time = e.next_arrival_time;
      e.next_arrival_time = s.latest_system_time + expo_distr(mean);
      m.number_accepted_services = m.number_accepted_services + 1;
....

end
```

Example: Queueing Systems

	event		current state			next state			next event		monitor			
	nat	net	lst	est	ebt	lst	est	ebt	nat	net	nar	nrr	sbt	qwt
arrival (server empty, queue empty)	a	inf	-	inf	inf	a	a	inf	exp a	erl a	+1	=	=	=
arrival (server busy, queue empty)	a	b (> a)	-	c	inf	a	c	a	exp a	b	+1	=	=	a
arrival (server busy, queue busy)	a	b (> a)	-	c	d	a	c	d	exp a	b	=	+1	=	=
departure (queue empty)	a	b (< a)	-	c	inf	b	inf	inf	a	inf	=	=	+(b-c)	=
departure (queue busy)	a	b (< a)	-	c	d	b	b	inf	a	erl b	=	=	+(b-c)	+(b-d)

```
function [s,e,m] = transition_s1q1(s,e,m,mean,ke,me);


   ....
% arrival when server busy and queue full
 elseif e.next_arrival_time <= e.next_exit_time &&...
                                     s.entry_buffer_time < inf
      s.latest_system_time = e.next_arrival_time;
      e.next_arrival_time = s.latest_system_time + expo_distr(mean);
      m.number_rejected_services = m.number_rejected_services + 1;
....

end
```

Example: Queueing Systems

	event		current state			next state			next event		monitor			
	nat	net	lst	est	ebt	lst	est	ebt	nat	net	nar	nrr	sbt	qwt
arrival (server empty, queue empty)	a	inf	-	inf	inf	a	a	inf	exp a	erl a	+1	=	=	=
arrival (server busy, queue empty)	a	b (> a)	-	c	inf	a	c	a	exp a	b	+1	=	=	a
arrival (server busy, queue busy)	a	b (> a)	-	c	d	a	c	d	exp a	b	=	+1	=	=
departure (queue empty)	a	b (< a)	-	c	inf	b	inf	inf	a	inf	=	=	+(b-c)	=
departure (queue busy)	a	b (< a)	-	c	d	b	b	inf	a	erl b	=	=	+(b-c)	+(b-d)

```
function [s,e,m] = transition_s1q1(s,e,m,mean,ke,me);


 ....
% departure when queue is empty
 elseif e.next_exit_time <= e.next_arrival_time &&...
                                     s.entry_buffer_time == inf
      aux = e.next_exit_time - s.entry_server_time;
      s.latest_system_time = e.next_exit_time ;
      s.entry_server_time = inf;
      m.server_busy_time = m.server_busy_time + aux;
      e.next_exit_time = inf;
 ....
end
```

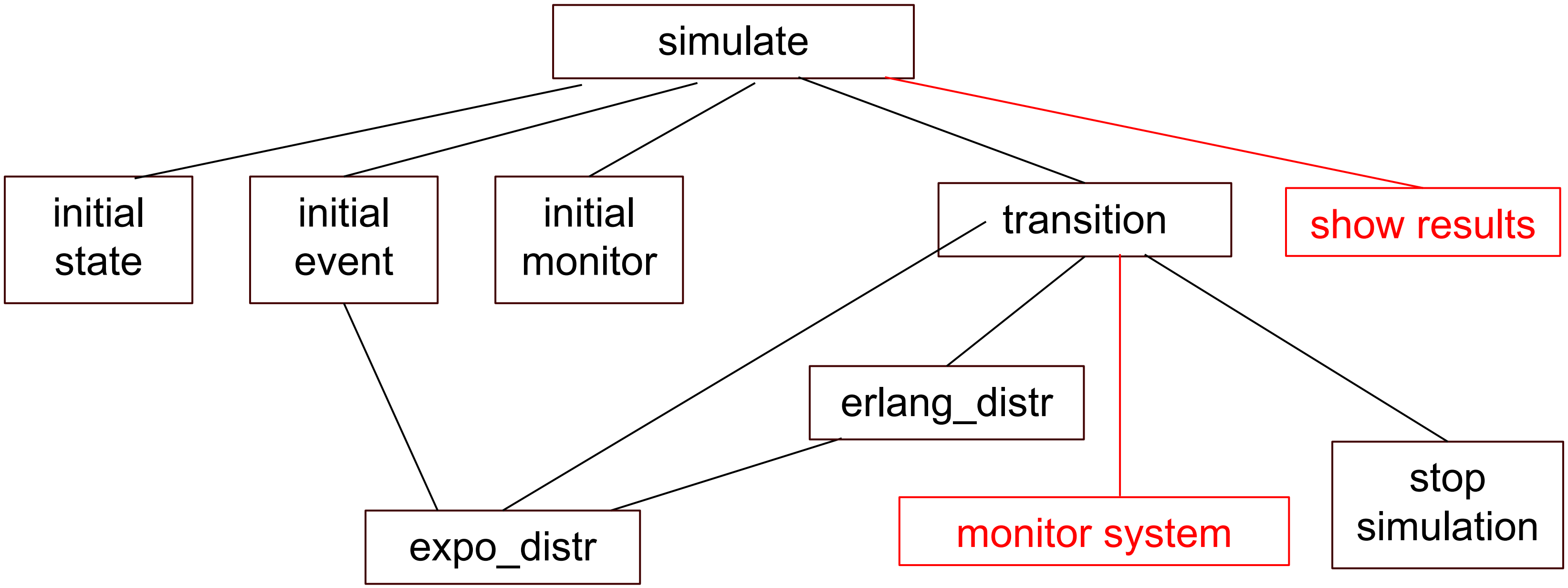
Example: Queueing Systems

	event		current state			next state			next event		monitor			
	nat	net	lst	est	ebt	lst	est	ebt	nat	net	nar	nrr	sbt	qwt
arrival (server empty, queue empty)	a	inf	-	inf	inf	a	a	inf	exp a	erl a	+1	=	=	=
arrival (server busy, queue empty)	a	b (> a)	-	c	inf	a	c	a	exp a	b	+1	=	=	a
arrival (server busy, queue busy)	a	b (> a)	-	c	d	a	c	d	exp a	b	=	+1	=	=
departure (queue empty)	a	b (< a)	-	c	inf	b	inf	inf	a	inf	=	=	+(b-c)	=
departure (queue busy)	a	b (< a)	-	c	d	b	b	inf	a	erl b	=	=	+(b-c)	+(b-d)

```
function [s,e,m] = transition_s1q1(s,e,m,mean,ke,me);
....
% departure when queue is full
 elseif e.next_exit_time <= e.next_arrival_time &&...
                                     s.entry_buffer_time < inf
      aux_1 = e.next_exit_time - s.entry_server_time;
      aux_2 = e.next_exit_time - s.entry_buffer_time;
      s.latest_system_time = e.next_exit_time;
      s.entry_server_time = s.latest_system_time;
      s.entry_buffer_time = inf;
      e.next_exit_time = s.latest_system_time + erlang_distr(ke,me);
      m.server_busy_time = m.server_busy_time + aux_1;
      m.queue_wait_time = m.queue_wait_time + aux_2;
   else  printf("unforeseen situation!!!"); end
end
```

Debugging Programs

- Simulation of a queueing process is an example of a program with some degree of complexity, that poses difficulties in debugging.
- A general rule in a program structured by means of nested functions is to guarantee that no function is used before it is fully debugged.
- In. addition, auxiliary functions may be (temporarily) used to obtain generated in the process so as to be analysed and give clues to potential mistakes.



Debugging Programs

- The progress of the simulation may be monitored during the transitions, to check whether they are modelling correctly the system intended behaviour:

```
function monitor_s1q1_transitions(s,e,m)
   printf("time = %i, server = %i, buffer = %i\n", ...
                       s.latest_system_time, ...
                       s.entry_server_time, ...
                       s.entry_buffer_time)
   printf("arrival = %i, exit = %i\n", ...
                       e.next_arrival_time, ...
                       e.next_exit_time)
   printf("accept = %i, reject = %i, busy = %i, wait = %i\n", ...
                       m.number_accepted_services, ...
                       m.number_rejected_services, ...
                       m.server_busy_time, ...
                       m.queue_wait_time)
end
```

- Note that the information should be presented in an “ergonomic” way, so as to be easily understood.

Debugging Programs

- The results from simulation may be shown in an “ergonomic form”. , for example by means of function **show_s1q1_results**, shown below (first the data to show):

```
function show_s1q1_results(s,e,m);
   final_simul_time  = s.latest_system_time;
   tot = m.number_accepted_services + m.number_rejected_services;
   total_nb_requests = tot;
   accepted_requests = m.number_accepted_services;
   fraction_accepted = 100 * accepted_requests / total_nb_requests;
   rejected_requests = m.number_rejected_services;
   fraction_rejected = 100 * rejected_requests / total_nb_requests;
   mean_service_time = m.server_busy_time / accepted_requests;
   mean_arrival_time = final_simul_time / total_nb_requests;
     total_busy_time = m.server_busy_time;
  fraction_busy_time = 100 * total_busy_time / final_simul_time;
  mean_waiting_time  = m.queue_wait_time / accepted_requests;
  ...
end
```

Debugging Programs

- The data is then shown in the terminal:

```
function show_s1q1_results(s,e,m);
   ...
   printf("\n")
   printf("\n---Results of Simulation:\n");
   printf("   total_nb_requests = %i\n", total_nb_requests);
   printf("    total_simul_time = %i\n", final_simul_time);
   printf("   total_nb_accepted = %i (%4.1f of total)\n",...
                                      accepted_requests,...
                                      fraction_accepted);
   printf("   total_nb_rejected = %i (%4.1f of total)\n",...
                                      rejected_requests,...
                                      fraction_rejected);
   printf("    server_busy_time = %i (%4.1f of total)\n",...
                                      total_busy_time,...
                                      fraction_busy_time);
   printf("   mean_service_time = %4.2f\n", mean_service_time);
   printf("   mean_arrival_time = %4.2f\n", mean_arrival_time);
   printf("   mean_waiting_time = %4.2f\n", mean_waiting_time);
   printf("\n")
end
```