DATA7202

Statistical Methods for Data Science

Unit 2 (Part 2)
Process Simulation Discrete Event Systems

Slava Vaisman

The University of Queensland
School of Mathematics and Physics

Why Process Simulation? (Example 1 — Financial Planning)

Consider a portfolio credit risk setting. Given a portfolio of k assets, the portfolio loss L is the random variable

$$L = \sum_{i=1}^{k} l_i \, X_i,$$

where l_i is the risk of asset $i \in \{1, \ldots, k\}$, and X_i is an indicator random variable that models the default of asset i. Under this setting, one is generally interested in the following.

1. *Conditional Value at Risk (CVaR).* Given a threshold (value at risk) v, calculate the conditional value at risk $c = \mathbb{E}[L \mid L \geqslant v]$.
2. *Tail probability estimation.* Given the value at risk, calculate the probability:

$$\mathbb{P}(L \geqslant v) = \mathbb{E}\left[I_{\{L \geqslant v\}} \right].$$

Example 1 — Financial Planning

$$L = \sum_{i=1}^{k} l_i X_i,$$

- For any realistic model, the defaults will be dependent.

- Moreover, additional complications such as dependent losses, market conditions, etc., should be taken into consideration.

- Consequentially, a Monte Carlo simulation may be our only resort.

Introduction to Discrete Event Systems

- Our surrounding contains complex *systems*.
- Such systems consist of related *components* or *elements* that form a complex organization.

  A bank branch can be considered as an example of complex system. Specifically, the bank tellers, the manager and clients can be viewed as system components.
- In addition, these components might have a certain characteristic called the *attributes*. The attributes can have both logical and numeric values.

  For example, the number of available tellers and individual skill level can be important component attributes.
- Finally, an interaction between the system components affects the system current state.

  For example, a customer which is leaving the branch after receiving the service affects the number of available tellers and the waiting line length. In this way, component interactions may cause a complex system behavior over time.

Example: repairmen problem (1)

- We consider a repair system consisting of m repairmen and n machines.
- At the beginning, all machines are operational. As soon a machine breaks, it is been taken by a repairman for a fix.
- In the unfortunate case that all repairmen are busy, the machine enters a repair queue.
- The next available repairman will fetch a machine from the queue and fix it.
- If there are no machine in the queue, the repairman will remain non-utilized until the next machine breaks.
- In such problems, we will be interested in the system utilization. Namely, how many machines are working and how many repairmen are busy. We are generally concerned with the average performance over a long period of time.

Example: repairmen problem (2)

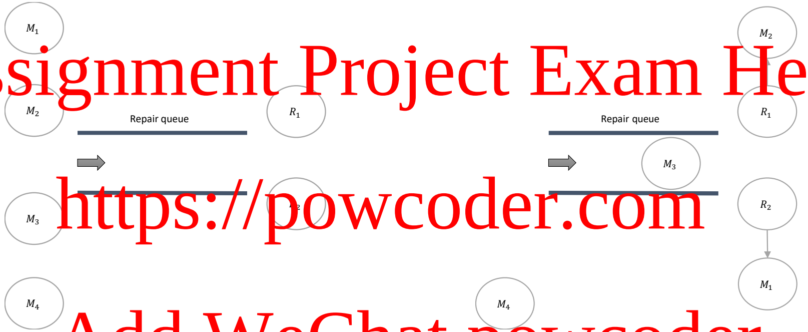


Figure: The left pannel shows the repairman system with 4 machines and 2 repairman. The right pannel shows the same system after the machines 1,2 and 3 entered the failure state. Note that in this case both repairmen are busy with machines 1 and 2, while the third machine is located in the waiting queue.

Analysis (1)

- In order to analyze complex systems, we first need to introduce a corresponding mathematical model.
- Such model should at least summarize the crucial parts of the system under consideration.
- Generally, a mathematical model for the system will involve a definition of components, parameters, and probability distributions.

For the repairman example, we can identify two types of components;

1. the machine and the repairman, it is also possible to consider the waiting queue as a component;
2. the number of machines and repairman are input parameters.

Analysis (2)

- We should also make assumptions regarding the machine lifetime and fix-time distribution.
  For example, we can define

$$T_{\mathrm{lifetime}} \sim \mathrm{Exp}(\lambda), \quad \text{and}$$
$$T_{\mathrm{fix}} \sim \mathrm{Exp}(\mu).$$

- Finally, we need to specify the queuing regime. In this example, we might assume that there are no VIP machines and thus the queue regime is *first in first out* (FIFO).

Analysis (3)

As soon as the mathematical model is available, we face the following possibilities for the system analysis.

1. If the system is simple enough (very unrealistic assumption) we might be able to develop the appropriate equations and even obtain a *closed form solution*.
2. In some cases, it is possible to develop the system equations, however, one cannot obtain an exact solution due to the model complexity. This scenario is very common in most economic and queuing systems.
3. In most practical scenarios, it is generally hard to analyze complex real-life systems. The system may be very complex. That is, their formulation in terms of a simple equation is infeasible.

Analysis (4)

- A specifically important scenario is when the distributions are not available in their analytical forms.
- That is, we only have an access to samples. In the repairman problem case, for example, the life time and the service historical data will be recorded by system administrators and we will only have an access to this particular data.
- Having said that, we will need to either fit a distribution to the data, or, sample from the empirical commutative distribution function.

In the general case, stochastic simulation really shines. Moreover, simulation can provide an important insights into the problem. Namely, it can help to identify important (and not so important) variables. This task is generally called a sensitivity analysis.

Analysis (5)

Generally speaking, the bellow Figure shows the workflow for analysis of systems.

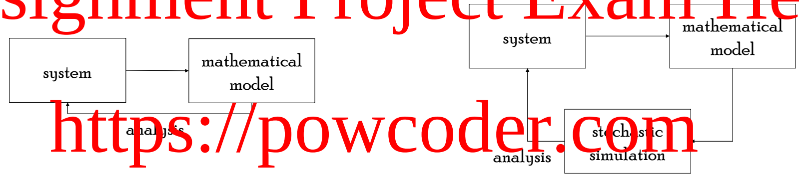


Figure: The left panel shows the fortunate case for which an analytical solution is available. The right panel shows the general case, for which we need to resort to stochastic simulation.

Discrete and Continuous System

Definition (Discrete and Continuous System)

The *discrete-event systems* (DES), are those systems in which the state variables change instantaneously through jumps at discrete points in time. The *continuous systems*, are those systems in which the state variables change continuously in time.

- A car-wash facility is an example of DES. Here, the system state changes when a customer arrives or departs.
- On the other hand, a space-ship movement is an example of continuous system. Here, the system state changes continuously over time as a function of space-ship velocity.

We continue with several examples with a view to provide an additional (important) intuition of different DESs.

Queuing theory (1)

Example (Queuing theory)

The queuing theory studies important problems involving queuing (or waiting). Typically it studies services and thus is extremely valuable for business study and optimization. In general, all queuing systems can be broken down into individual sub-systems consisting of queue and an activity as shown in Figure 3.

Figure: A queing sub-system.

Some typical examples are customers (in a bank branch, supermarket, call-center, or public transport) waiting for a service, a web-browser is waiting for response from computer servers, and reliability theory — measure an average waiting time until a failure occurs.

Queuing theory (2)

In order to analyze a queuing system, we need to consider the following components of this system.

1. The arrival regime. Are the customers arrive one at a time or in a group? What is the distribution of the arrival time? Should we consider an infinite number of arrivals?
2. The service mechanism. What are the resources needed for the service to begin? What is the service duration (namely, what is the service time distribution)? What is the number of available service stations? Is the service is parallel or sequential? Can a particular server terminate a customer processing to handle different ("emergency") customer?
3. The queue characteristics. What is the queuing regime (FIFO, LIFO)? Can a customer decide not to join the queue if the line is too long? Can a customer become "discouraged" from the proposed service if her waiting time is too long? Is the queue has an infinite capacity? Can a costumer switch between queues if she believes that the alternative one is shorter?

Queuing theory (3)

Given a queuing system, there are many interesting questions one can ask.

1. What is the expected waiting time of a system customer (in the queue and overall)?
2. What fraction of the customers will wait longer than a pre-specified time threshold before they are served?
3. How long is the queue on average?
4. What is the probability that the queue exceeds a pre-defined length?
5. What is the expected time the server is busy? (Server not busy means that we loose money!)

Queuing theory (4)

Questions to ask for a system design or optimization.

1. How many servers do we need to optimize a certain aspect of the system? For example, we might need to comply with a regulation that the waiting time should not exceed 3 minutes. In some countries, there is a requirement for a maximum waiting time.
2. Should we introduce a (higher) priority for some types of arriving customers?
3. Should we invest resources to reduce the service time?

Queuing theory (5)

Example (Jackson Tandem Queue)

Consider the Jackson Tandem Queue in the Figure bellow.

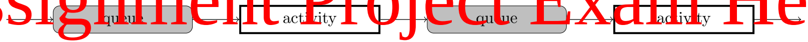


Figure: A Jackson Tandem Queue.

- The arrival to the first subsystem is Poisson with intensity λ.
- The service times of the first and the second activity are μ_1 and μ_2, respectively.
- What do we want to find out? For example, it can be average time of the client in the system, expected waiting time — time at queue 1 or 2, average length of the queue, etc.

Next, we are going to deal with DES systems. To set the stage, we consider the formal setting next.

DES Formal setup (1)

In order to formally discuss discrete event systems (DES), we need to consider two important ingredients.

1. The *system state* is the collection of variables that describe the process.
   Generally speaking, the system states are described by a stochastic time dependent process $\{\boldsymbol{X}_t\}_{\{t \geqslant 0\}}$.
2. The *system event* is a collection of events that change the system state.
   Generally, an event includes the event occurrence time (when this event happened), and the event type (how the system is affected after the event occurrence).

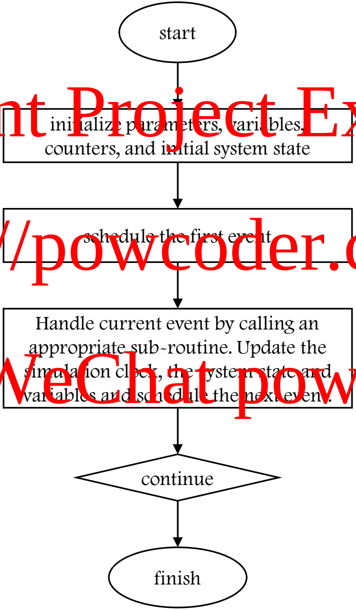
- It is important to note that from the simulation point of view, the system is observed only at times where an event occurred.
- We do not observe the system between event times, since we assume that the system is either not changing or can only change in a deterministic way.

DES Formal setup (2)

The following steps should be performed in order to carry out DES study.

1. Determine the system state space.
2. Determine the system components and all possible events and the corresponding interactions.
3. Determine the distributions (such as arrival, service, etc.)
4. Determine the required data structures such as queues, counter, etc. When defining variables, keep in mind the parameters that we would like to investigate.
5. Implement sub-procedures for every possible event. These sub-procedures will generally update the system state, variables and counters. In addition, these methods will schedule future events according to the system's behavior logic.
6. Report the results along with the corresponding statistical analysis.

A general DES process flow



The repairman problem

In the repairman example, in which we are interested in the number of utilized machines and repairman, the following holds.

1. The system state $\{\boldsymbol{X}_t\}_{\{t \geqslant 0\}}$ is $\boldsymbol{X}_t = (X_{m,t}, X_{r,t}, W)$, where $X_{m,t}$ and $X_{r,t}$ stand for the number of utilized machines and repairman, respectively, and W represents the waiting queue.
2. The system components are *machine* and *repairman*. In addition, there are two possible events: *machine breaks*, and *machine repaired*. Note that the system state does not change between the events.
3. We can assume, for example, that the life and the fix times are distributed according to $\text{Exp}(\lambda)$ and $\text{Exp}(\mu)$, respectively.
4. An essential data structure is the waiting queue, in which the machine wait until the first repairman becomes available. We might need additional flags for the repairman (free/busy).

The repairman problem (5) the main DES procedure

Algorithm 1: The Main DES procedure for repaireman problem

input : The rates λ, μ, n, m and the simulation time T.

output: —

Set $t \leftarrow 0$ and schedule a life-time for machine $1 \leqslant i \leqslant n$ to be $t + L_i \sim \text{Exp}(\lambda)$.

while $t < T$ **do**

    Get the first (earliest) event in the event list, and let t be its time.

    switch *event type* **do**

        case *Machine break* **do**

            Call Machine break.

        end

        case *Machine repair* **do**

            Call Machine repair.

        end

    end

end

The repairman problem (5) the main DES procedure — subroutines

1. If the event is *machine break*, we just check if there are available free repairman. If so, we assigned this machine to the repairman and schedule the machine fix time according to the current time + Exp(μ). If there are no available repairman, we just push the machine into the waiting queue.
2. If the event is *machine repair*, we schedule the current machine life time to current time + Exp(λ). Then, we check if there are machines waiting in the waiting queue. If so, we fetch the machine from the waiting queue, assign it to the repairman and schedule the machine fix time according to the current time + Exp(μ).

6. The corresponding statistical analysis will be discussed later.

The event queue (1)

- We still have a single technical issue that should be addressed.
- How do we schedule future events?
- Clearly, we can store all scheduled events in some data structure (like a list).
- Then, we can just advance the system clock with some Δt, that is, we can consider all times $0, \Delta t, 2\Delta t, \ldots$, and in each iteration, just check if such time exists in the stored event list.
- While the above approach is conceptually correct, it is extremely inefficient from the computational point of view.
- Recall that we are dealing with DESs, and no activity happens between the events.
- The above observation implies, that it is wasteful to iterate trough $0, \Delta t, 2\Delta t, \ldots,$ times.

The event queue (2)

- Instead, we can store the events in a data-structure called the priority-queue.
- Specifically, all the events are stored in the queue **sorted by the system time (in ascending order)**, of these events.
- In this case, the simulation clock will just jump from one event to another according to the event times.
- To summarize, when we say that we schedule an event for time t, the event (event type and event time t), will be stored in the priority queue according to t.
- The event will then be fetched from the priority queue during the *handle current event* node in the DES flow Figure.

As soon as we have the simulation data in hand, we need to consider the system analysis, which is discussed next.

Reporting the results

First, let us consider a sampling from the empirical cdf. This sampling is required in the data-science setting, namely, when analytical distributions of the system are not available.

Definition (Empirical cdf (1))

Let $X_1, \ldots, X_n \sim F$ be an iid samples; note that F is not known. Then, F can be approximate with the empirical distribution function, which is defined as follow

$$\widehat{F}_n(x) = \frac{\sum_{i=1}^{n} I\{X_i \leqslant x\}}{n}.$$

This is equivalent to an assignment of weight $1/n$ to every observation. In addition, $\widehat{F}_n$ is approaching F for large n.

Empirical cdf (2)


```
empiricalcdf.m
N = 100;
mu = 10;
X = exprnd(mu,N,1);
x = linspace(0,max(X),100);
y = expcdf(x,mu);
plot(x,y);
hold on
ecdf(X);
legend('Theoretical CDF','Empirical CDF',
                    'Location','best')
hold off
```

Empirical cdf (2)

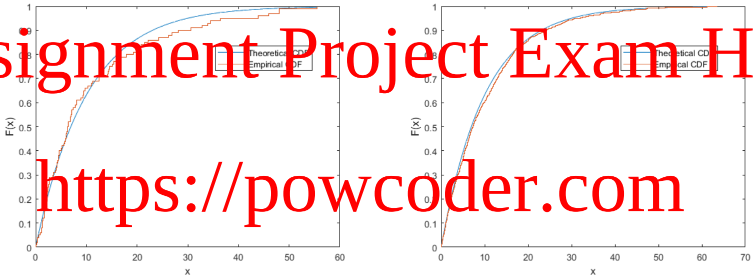


Figure: Empirical cdf versus true cdf of exponential distribution with $\mathbb{E}[X] = 10$. Left panel for 100 samples, and right pannel for 1000 samples.

Finite Horizon Simulation (1)

- Many times, we are interested in a process simulation and analysis for a fixed time interval $[t_{\text{start}}, t_{\text{end}}]$.
- Suppose that $T = t_{\text{end}} - t_{\text{start}}$.
- As in the static setting, we considered the estimation of $\ell_T = \mathbb{E}Y$, where $Y = H(\boldsymbol{X})$ and

$$\ell_T = \mathbb{E}\left[\frac{1}{T}\int_0^T Y_t \mathrm{d}t\right]$$

via

$$\widehat{\ell}_T = \frac{1}{N}\sum_{i=1}^{N} Y_i = \overline{Y}. \tag{1}$$

Finite Horizon Simulation (2)

- In additional, we provided the corresponding confidence interval. In particular, since for large N, $\widehat{\ell}$ is distributed approximately $\mathsf{N}(\ell, \sigma^2/N)$, where σ is unknown, but can be estimated (without bias) via

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(Y_i - \overline{Y}\right)^2. \tag{2}$$

- The approximate $1-\alpha$ confidence interval is give by

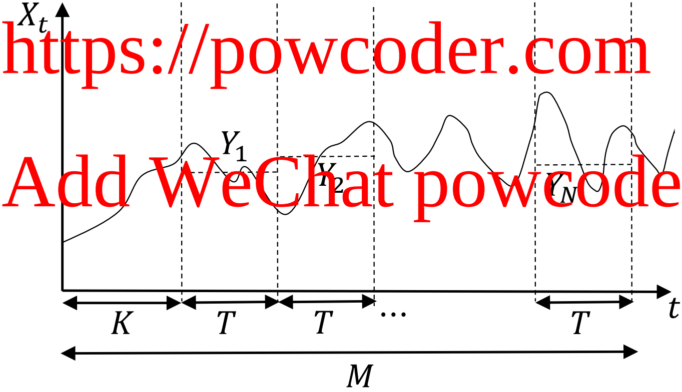
$$\left(\overline{Y} - z_{1-\alpha/2}\frac{S}{\sqrt{N}}, \overline{Y} + z_{1-\alpha/2}\frac{S}{\sqrt{N}}\right). \tag{3}$$

Finite Horizon Simulation (3)


```
%finitehorizon.m
N = 1000; % # of samples

% sample from N(5,9)
ell = randn(N,1)*3 + 5;

mu = mean(ell);
S = std(ell);
CI = [mu-1.96*S/sqrt(N),mu+1.96*S/sqrt(N)];
fprintf("0.95 CI for mu is (%d, %d)\n",CI(1),CI(2));

>> finitehorizon
0.95 CI for mu is (4.793920e+00, 5.165342e+00)
```

Steady-State Simulation — the Batch Means Method (1)

In dynamic settings, we sometimes will be interested in the steady state simulation. In particular, there will be some initial period of time, for which we would like to throw the samples away. This is the so-called system burn-in. The Batch Means Method idea is sleight-forward.



Steady-State Simulation — the Batch Means Method (2)

1. Perform a long simulation run while obtaining M samples.
2. Discard the first K observations corresponding to the burn-in period.
3. Divide the remaining $M - K$ samples into N batches. Note that each batch is of length

$$T = \frac{M - K}{N}.$$

4. Let $X_t^{(i)}$ be the t-th observation from the i-th batch for $i = 1, \ldots, N$. Let Y_i be the mean of the i-th batch, that is,

$$Y_i = \frac{1}{T} \sum_{t=1}^{T} X_t^{(i)}.$$

5. Apply (1), (2), and (3) to deliver the estimator and the confidence interval.

```
%batchmeans.m
M = 10000; % # of samples
K = 1000;  % discard
N = 50;    % # of batches
% sample from N(5,9)
ell = randn(M,1)*3 + 5;
T = floor((M-K)/N);
ell = ell(K+1:length(ell));
batch_ell = zeros(N,1);
for i=1:N
   startid =T*(i-1)+1;
   endid = startid + T-1;
   tmp = ell(startid:endid);
   batch_ell(i) = mean(tmp);
end
```

```
mu = mean(batch_ell);
S = std(batch_ell);
CI = [mu-1.96*S/sqrt(N),mu+1.96*S/sqrt(N)];
fprintf("batch means 0.95 CI for mu is (%d, %d)\n"
                    ,CI(1),CI(2));

>> batchmeans
batch means 0.95 CI for mu is
               (4.972924e+00, 5.079908e+00)
```

Typical types of estimators (1)

In practice, we con encounter different types of desired estimator. Here, we discuss two such types.

Type (i).

- For example, if we need to estimate the average time a client spends in the system, a convenient practice is to store a departing client array, where each client contains the system entry time and the system departure time.

- Define Y_i to be the client's i departure time minus the entry time.
- The corresponding estimators from the departure events are straight forward.

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} Y_i, \quad N \text{ is the number of clients,}$$

(via (1), (2), and (3) regardless of the finite-horizon or steady-state setting) .

Typical types of estimators (2)

Type (ii).

- An additional problem that should be considered is the estimation of the step function.
- For example, consider the average utilization of the server.
- We should be careful in this case, since the samples that we have from the DES procedure are not uniformly distributed (we just have a record of system states at the *specific* times where an event happened).
- In this case, we need to estimate the so-called step function.

Typical types of estimators (3) — Type (ii)

Generally , we need to record the step lengths. That is, we cannot directly use the estimator

$$\widehat{\ell}_T = \frac{1}{N}\sum_{i=1}^{N} Y_i = \overline{Y}.$$

for

$$\ell_T = \mathbb{E}\left[\frac{1}{T}\int_0^T Y_t \mathrm{d}t\right].$$

Instead, we should create a two-dimensional array having values $(\boldsymbol{X}_i, \Delta_i)$, where

$$\Delta_i = t_i - t_{i-1}.$$

The estimator $\widehat{\ell}_T$ can now be calculated via

$$\widehat{\ell}_T = \frac{\sum_{i=1}^{n} \boldsymbol{X}_i \times \Delta_i}{\sum_{i=1}^{n} \Delta_i},$$

where n is the length of the two-dimensional array.

Worked example

- We consider and implement a full example DES system with explanation.
- Understanding this example should allow a relatively easy implementation of other systems.
- We assume an M/M/1 queue. The system has a single server, where arrivals are determined by a Poisson process with rate λ and job service times have an exponential distribution with parameter μ.
- The model is elementary enough to allow closed-form expressions for some metrics of interest.

M/M/1 queue (1)

To set the stage, we consider the following facts.

- The inter-arrival times are exponentially distributed with mean $1/\lambda$. The average service time is exponentially distributed with mean $1/\mu$.
- The model stability determined by the condition: $\lambda < \mu$. Note that if (on average), arrivals happen faster than the service, the queue will grow indefinitely long. As a consequence, the system will not have a stationary distribution.
- We define $\varrho = \lambda/\mu$. The parameter ϱ is the average proportion of time for which the server is utilized.
- The average number of customers in the system is $\varrho/(1-\varrho)$ and the variance of number of customers in the system is $\varrho/(1-\varrho)^2$.

M/M/1 queue (1)

Our objective is to carry out DES simulation study for the M/M/1 queuing system. In this study, we would like to estimate the following quantities.

- Estimate $\widehat{\varrho}$ — the average proportion of time for which the server is utilized. The analytical solution is $\varrho = \lambda/\mu$.
- Estimate $\widehat{L}$ — the mean number of customers in the system. The analytical solution is $L = \varrho/(1-\varrho)$.

- $\widehat{L}_q$ — the mean number of customers in the queue. The analytical solution is $L_q = \frac{\varrho^2}{(1-\varrho)}$.

- $\widehat{W_q}$ mean wait in the queue. The analytical solution is $\frac{\varrho^2}{\lambda(1-\varrho)}$.
- $\widehat{W}$ mean wait in the system. The analytical solution is $W = W_q + \frac{1}{\mu} = \frac{\varrho^2}{\lambda(1-\varrho)} + \frac{1}{\mu}$.

M/M/1 queue (2)

- First, we define the system state space. Let $\{\boldsymbol{X}_t\}_{\{t \geqslant 0\}}$ be the corresponding process, where $\boldsymbol{X}_t = (t, W_t, X_{\text{busy}})$, where t is the current time, W_t is the number of customers waiting in the queue, and X_{busy} stand for the server status 0-free, 1-busy.
- Note that both the number of customers in the queue and in the system at time t is given by W_t and $W_t + X_{\text{busy}}$ respectively.
- In order to estimate the mean waiting times, it will be convenient to store the departure client data which contains the client's arrival time, service start time and service end time.
- That is, the clients wait time in the queue and the client overall wait time in the system is given by service start time - arrival time, and service end time - arrival time, respectively.

Algorithm 2: The main DES procedure for the M/M/1 queue.

input : The rates λ, μ, and the simulation time T.
output: —

Set $t \leftarrow 0$ and generate the first arrival at $t + \text{Exp}(\lambda)$.
while $t < T$ **do**
  Get the first (earliest) event in the event list, and let t be its time.
  switch *event type* **do**
    case *Arrival* **do**
      Call Arrival: (Schedule next arrival, and handle service and the queue).
    end
    case *Departure from system* **do**
      Call Departure: (If physical queue is not empty, pop an event and schedule its departure).
    end
  end
end

M/M/1 queue (4)

- The set of possible events is $\{A, D\}$, where A and D stand for arrival and departure, respectively.
- Note that we will have a single physical queue and the event queue. The sub-procedures for every possible event are as follows.
  - For arrival, schedule the next arrival and check if the server is empty. If so, enter the client to service. Otherwise, insert the customer to the physical queue.
  - For departure, Check if the physical queue is not empty. If it is not empty, send the first waiting customer to service and schedule its departure in the event queue.

Running the `mm1queue.m` code, results in the following output.

M/M/1 queue (5)

```
ANALYTICAL RESULTS
----------------------------------------------
the average proportion of time for which the server is utilized is
                                                 6.666667e-01
the average number of customers in the queue is 1.333333e+00
the average number of customers in the system is 2.000000e+00
the average wait time the queue is 6.666667e-01
the average wait time the system is 1.000000e+00
----------------------------------------------
STEADY STATE SIMULATION RESULTS
average proportion of time for which the server is utilized 6.616047e-01
          CI=(6.433565e-01, 6.798530e-01)
average number of customers in the queue 1.284452e+00
          CI=(1.076862e+00, 1.492042e+00)
average number of customers in the system 1.946057e+00
          CI=(1.723913e+00, 2.168201e+00)
average wait time in the queue 6.341173e-01
          CI=(5.449580e-01, 7.232765e-01)
average wait time in the system 9.661445e-01
           CI=(8.738474e-01, 1.058442e+00)
```