Discrete stochastic simulation

Project notes

Introduction (1)

There are several types of system simulation:

- **Analog simulation**: with physical models, for instance, the reduced model of a bridge.
- **Digital simulation**: done in computers.
  - Possible models:
    - **Continuous models**: differential equations which describe the evolution of continuous variables.
    - **Discrete models**: phenomenon that can be described by events and its sequences.
  - Classification:
    - **Stochastic**: the variables follow random laws, for instance, the time between events.
    - **Deterministic**: the variables follow deterministic laws.

Introduction (2)

- **In the following, we illustrate the technique of digital, discrete and stochastic simulation**.
- Problem: **Simulation of toll highway**.
  - Simulate the traffic in a toll highway, considering only one way.
  - Vehicles enter the the highway with a random frequency and, at the end, arrive at a tollhouse (only with one tollkeeper), stay in the queue until payement and, afterwards, get out of the toll highway.
  - We intend to study the evolution of the number of vehicles in the queue, in function of the following random laws:
    - Time between arrivals;
    - Travel time;
    - Payment time.

Introduction (3)

- The adopted technique is based on **chronological sequencing pending events**:
  1. **Identify the relevant events to model the system.**
     - It is necessary to categorize the events:
       - It is essential to categorize events in order to distinguish them.
       - It is essential to set a time attribute, that time-stamps the instant when the event occurs.
  2. **Define random laws to drive the events.**

Introduction (4)

3. **Define procedures to simulate the observation of the random variables in question.**
4. **Simulate the events**.
   - The core of the simulator is the **pending event container** (PEC), which contains all events waiting to happen (future events). During the simulation:
     - Events can be removed from the PEC.
     - New events can be added to the PEC.

Identifying the events (1)

- In the simulation of the toll highway there exist the following events:
  - **Arrival**: arrival of a vehicle in the toll higway in study (entrance in the tollway).
  - **Tollhouse**: arrival of a vehicle to the tollhouse in the end of the tollway (it stays in the tollhouse queue or starts paying at the tollhouse).
  - **Departure**: exit of the vehicle from the tollway after paying.

Identifying the events (2)

- In the simulation of the toll highway there are no other attributes besides the time-stamp attribute.
- In more complicated simulations more attributes may be necessary.
  - For instance, one might need to associate to each event the corresponding vehicle, if one wants to study the total time spent by the vehicles in the tollway, including payment at the tollhouse.

Identifying the events (3)

- Class diagram with the events of the system (without any detail):

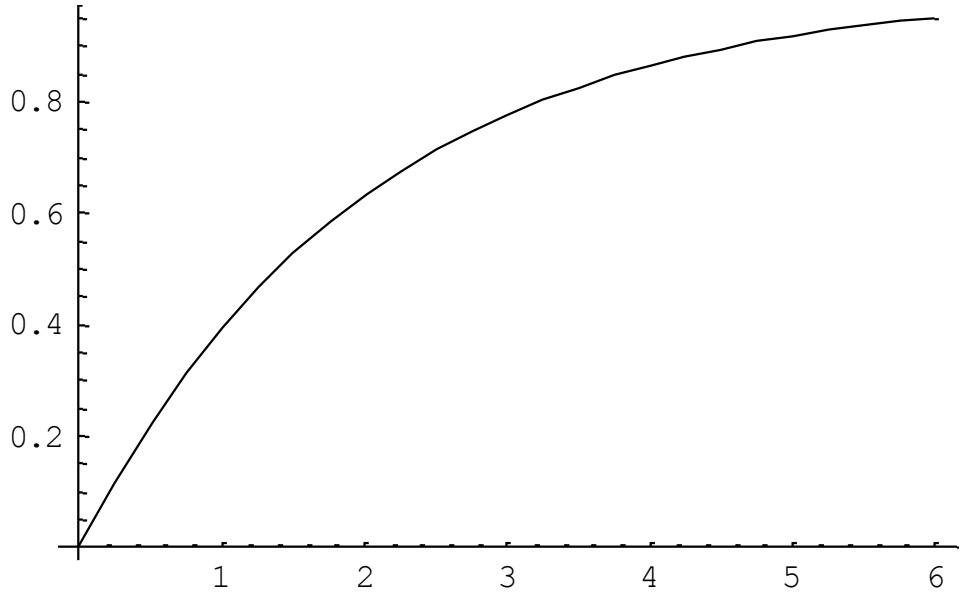
Random laws (1)

- In the simulation of the toll highway we need to consider the following random laws:
  - **Arrivals**: we assume that the time spent between consecutive arrivals is a random variable with exponential distribution with mean value ***arrival***.
  - **Traversals**: we assume that the time spent by a vehicle traversing the tollway is a random variable with exponential distribution with mean value ***traversal***.
  - **Payments**: we assume that the time spent by each vehicle paying (time between start paying, after being at the first place in the tollhouse queue, and exit) is a random variable with exponential distribution with mean value ***payment***.

Random laws (2)

- A random variable with **exponential distribution** with **mean value *m*** has the following cumulative distribution: ***1-exp{-t/m}***

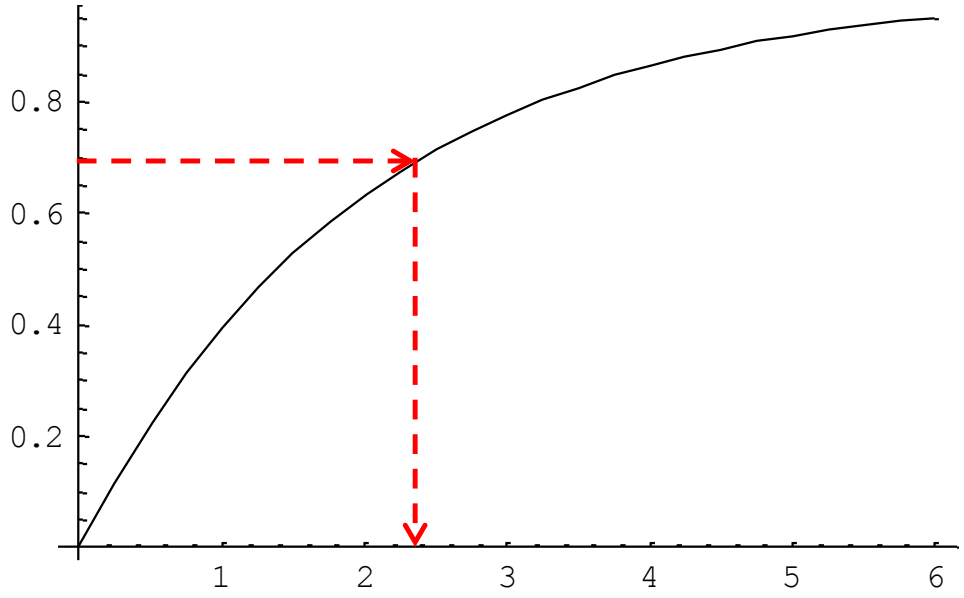
For instance, for m=2:



Random variable observation (1)

- An **observation of an exponential random variable** is generated by using the inverse of the cumulative distribution function: ***-m log(1-y)***

For instance, for m=2:



Random variable observation (2)

- If $Y \sim U(0,1)$ then $-m \log(1-Y) \sim Exp(m)$

```
static Random random = new Random();

public static double expRandom(double m) {
     double next = random.nextDouble();
     return -m*Math.log(1.0-next);
}
```

Random variable observation (3)

- For a random variable with exponential distribution with mean 2:

```
for (int i=0; i<5; i++) {
    System.out.println(expRandom(2));
}
```

In the terminal is printed:

```
3.611626562700762
0.545405686927192 2
0.969504648970717 1
3.763521233737085 6
0.045461122700387 36
```

Event simulation – PEC (1)

- To understand PEC, consider the problem of traffic simulation in study when:
  - There are vehicles in the tollway,
  - There are vehicles in the tollhouse queue, and
  - There is a vehicle paying in the tollhouse.
- **How should the simulation continue from this point on?**

Event simulation – PEC (2)

- The simulation is event-driven, where a cycle iterates over the events wainting in the PEC to happen.
  - A question arises: **which is the next event to simulate**?
  - A simple answer: we get from the PEC the next event, where by next we mean the event with smallest time-stamp.
  - Another question arises: **how do we feed the PEC with new events**?
  - The answers is quite more complicated…

Event simulation – PEC (3)

- All events we know that will happen in the future should be at PEC (they have not been simulated because their time has not yet been reached).
  - **Each time an arrival to the toll highway is simulated** we should add to the PEC:
    1. The next arrival: if sufficies to add to the PEC the event

       ```
       new EvArrival(time+expRandom(arrival))
       ```

    2. The end of traversal of the vehicle being simulated: it sufficies to add to the PEC the event

       ```
       new EvTollHouse(time+expRandom(traversal))
       ```

    where `time` is the attribute with the time-stamp of the current event (the current time), in this case an arrival event.

Event simulation – PEC (4)

- **Each time an end of traversal of the tollway is simulated** we have to check if the tollkeeper is occupied.
  1. If the tollkeeper is occupied we add the vehicle to the tollhouse queue.
  2. Otherwise the payment starts. In this case, we can compute the event of end of payment and, thus, we should add to the PEC the event

     ```
     new EvDeparture(time+expRandom(payment))
     ```
- Finally, **each time an exit of the tollway is simulated**, that is, end of payment, we have to examine the tollhouse queue. If the queue is not empty we remove an element from the queue (the first one) and we schedule the end of payment of this vehicle adding to the PEC the event

  ```
  new EvDeparture(time+expRandom(payment))
  ```

Event simulation – PEC (5)

- Indeed, the tollhouse queue is not needed, being only necessary to save the number of vehicles in the queue…

Simulation (1)

- The class **`TrafficSimulator`** contains:
  - The attribites `simulationTime`, `currentTime` and `currentEvent`;
  - The PEC;
  - The counter `numCarsTollHouseQueue`;

Simulation (2)

- The classes related with the events contain:
  - The `arrival`, `traversal` and `payment` values received as parameters;
  - The `time` of the event;
  - The method: **`simulateEvent();`**

Simulation (3)

- Assuming that **pec** attribute and **expRandom** method are accessible, the method **simulateEvent** related to the vehicle arrival to the tollway is defined as follows:

```
pec.addEvPEC(new EvArrival(time+expRandom(arrival)));
pec.addEvPEC(new EvTollHouse(time+expRandom(traveesal)));
```

- Note: be careful about the way access is given to the **pec** attribute and **expRandom** method!

Simulation (4)

- Assuming that **pec** and **numCarsTollHouseQueue** attributes, and **expRandom** method are accessible, the method **simulateEvent** related to the arrival at the tollhouse is defined as follows:

```
if (numCarsTollHouseQueue==0)
     pec.addEvPEC(new EvDeparture(time+expRandom(payment)));
numCarsTollHouseQueue++;
```

- Note: be careful about the way access is given to the **pec** and **numCarsTollHouseQueue** attributes, and **expRandom** method!

Simulation (5)

- Assuming that **pec** and **numCarsTollHouseQueue** attributes, and **expRandom** method are accessible, the method **simulateEvent** related to the arrival at the tollhouse is defined as follows:

```
numCarsTollHouseQueue--;
if (numCarsTollHouseQueue>0)
     pec.addEvPEC(new EvDeparture(time+expRandom(payment)));
```

- Note: be careful about the way access is given to the **pec** and **numCarsTollHouseQueue** attributes, and **expRandom** method!

Simulation (6)

- The **`PEC`** is a class with two methods:
  - **`public void addEvPEC(Event ev)`**
    adds the received event to the `PEC`.
  - **`public Event nextEvPEC()`**
    removes the first event from the `PEC` and returns it.

Simulation (7)

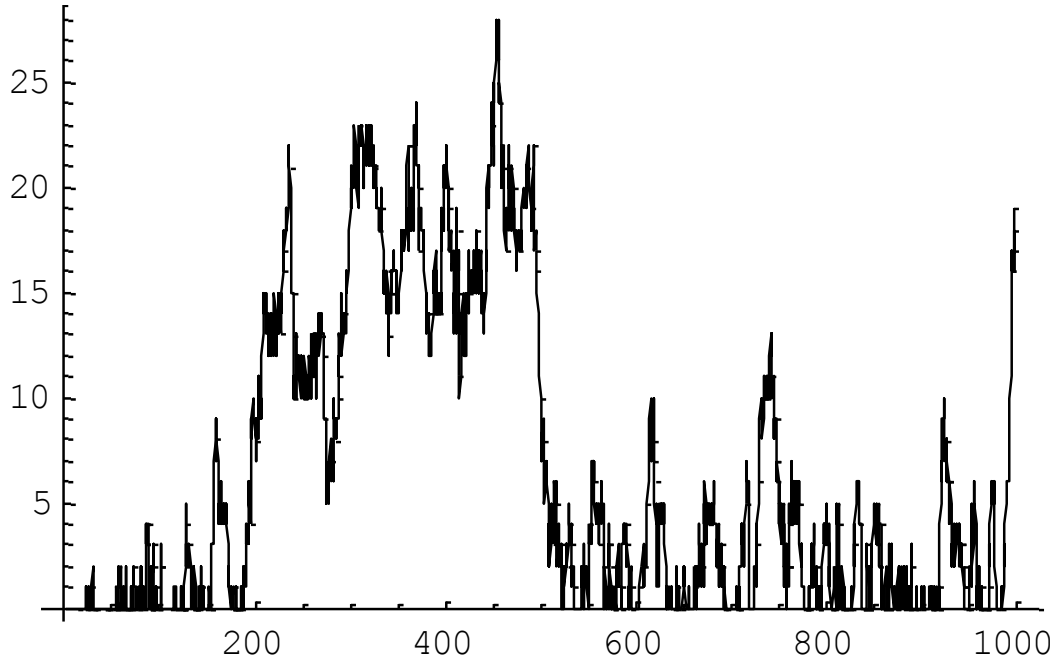
- **The simulator is the `main` method of the class `TrafficSimulator` and consists basically in a cycle, where in each iteration it is simulated the next event in the `PEC`.**
- The simulator has as input:
  - `arrival`
  - `traversal`
  - `payment`
  - `simulationTime`

Simulation (8)

```
public static void main(String[] args) {
     // TODO: input reading and actualization of the
     // attributes: arrival, traversal, payment and
     // simulationTime
     //initialization
     numCarsTollHouseQueue = 0;
     pec = new PEC();
     currentEvent = new EvArrival(expRandom(arrival));
     currentTime = currentEvent.time;
     //simulation cycle
     while (currentTime<simulationTime) {
          //TODO: simulate currentEvent
          currentEvent = pec.nextEvPEC();
          currentTime = currentEvent.time;
     }
}
```

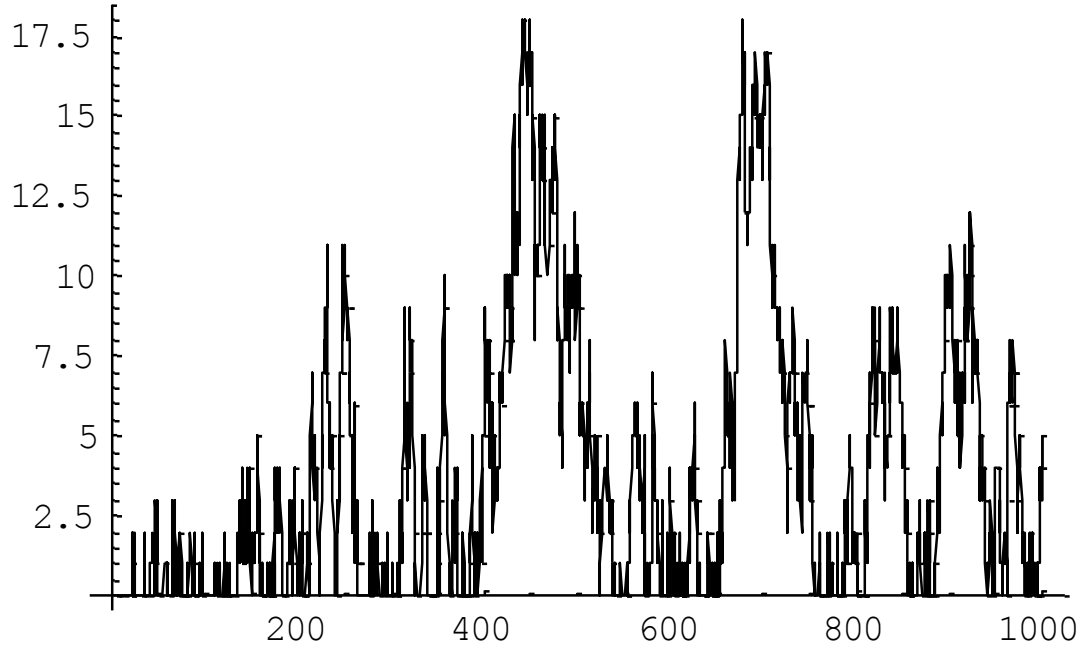
Examples (1)

- The following graphic is obtained for the length of the tollhouse queue running `TrafficSimulator` with:
  - `arrival=1`
  - `traversal=50`
  - `payment=0.9`
  - `simulationTime=1000`



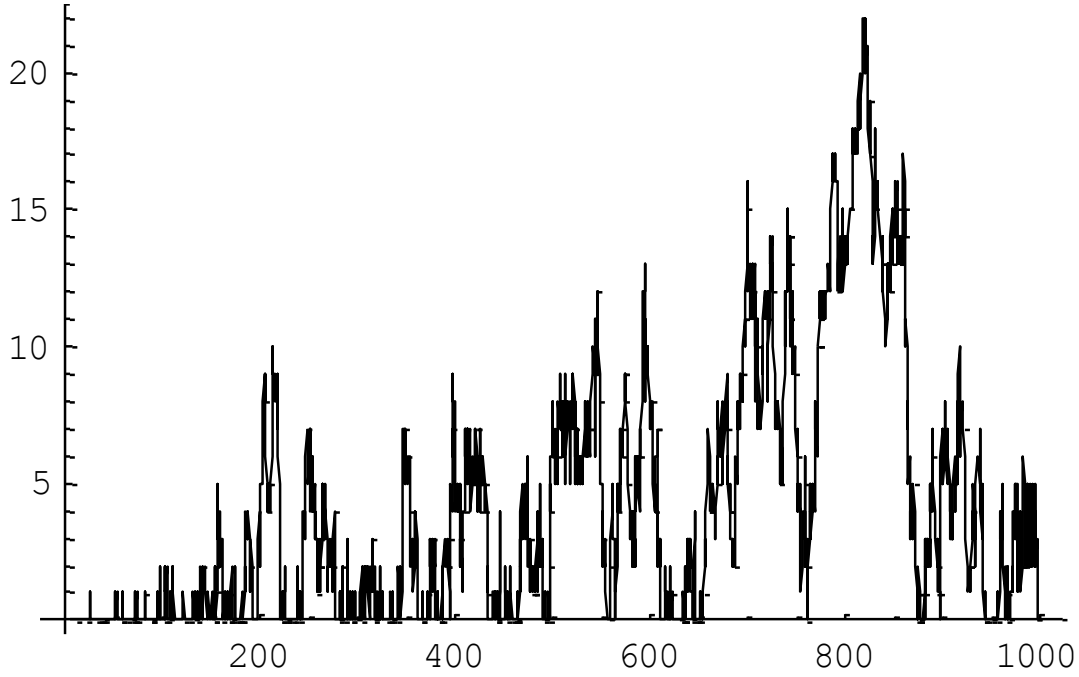
Examples (2)

- Another simulation with the same input:



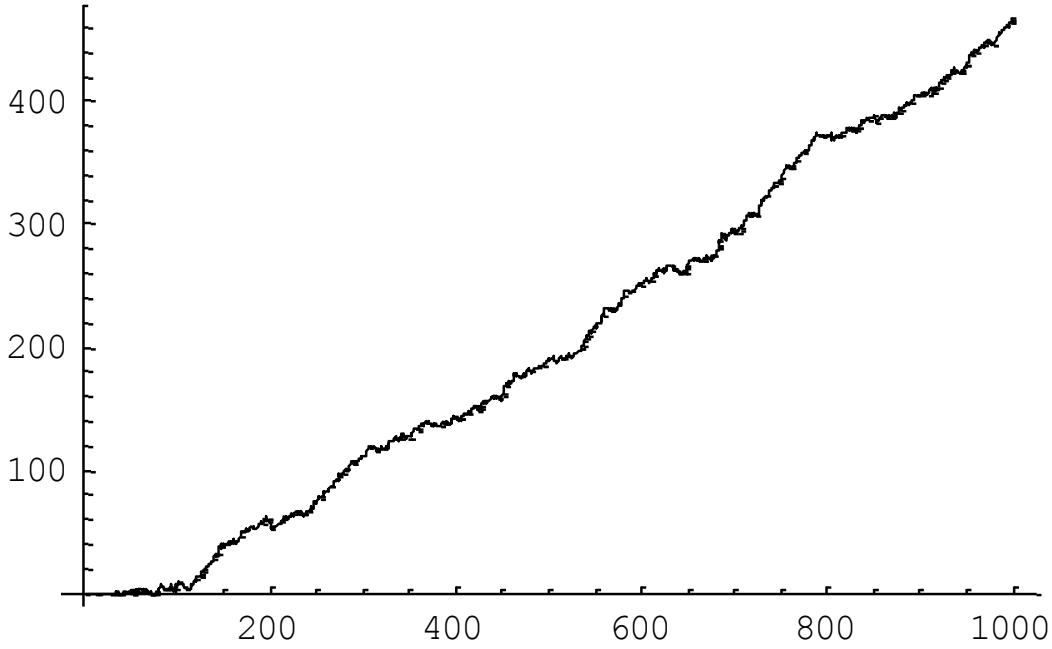
Examples (3)

- Another simulation with the same input:



Examples (4)

- Increasing only `payment` to `2`:



Examples (5)

- Decreasing only `payment` to `0.7`: