Lesson 6. Introduction to Stochastic Processes

1 Overview

- A **stochastic process** is a sequence of random variables ordered by an index set
- Examples:
    - $\{S_n; n = 0, 1, 2, \dots\} = S_0, S_1, S_2, \dots$ with discrete index set $\{0, 1, 2, \dots\}$
    - $\{Y_t; t \geq 0\}$ with continuous index set $\{t \geq 0\}$
- The indices n and t are often referred to as "time"
    - $\{S_n; n = 0, 1, 2, \dots\}$ is a **discrete-time process**
    - $\{Y_t; t \geq 0\}$ is a **continuous-time process**
- The **state space** of a stochastic process is the range (possible values) of its random variables
    - State spaces can be discrete or continuous
      (i.e. the random variables of a stochastic process are discrete or continuous)
- A stochastic process can be described by the joint distribution of its component random variables
- Working with joint distributions can be unwieldy and have technical issues
- Instead, we can describe a stochastic process via an algorithm for generating its sample paths
- Recall: a **sample path** is a record of the time-dependent behavior of a system
    - A stochastic process generates sample paths
        - e.g. a sequence of random variates of $S_0, S_1, S_2, \dots$
- Today: an example

2 The Case of the Leaky Bit Bucket

Bit Bucket Computers specializes in installing and maintaining highly reliable computer systems. One of its standard configurations is to install a primary computer, an identical backup computer that is idle until needed, and provide a service contract that guarantees complete repair of a failed computer within 48 hours. If it has not fixed a computer within 48 hours, then it replaces the computer.

Computer systems are rated in terms of their "time to failure" (TTF). The engineers at Bit Bucket Computers have developed a probability distribution for the TTF of the individual computers and a probability distribution for the time required to complete repairs. They would like to have a TTF rating for the entire system. A failure of the system is when both computers are down simultaneously.

Some additional details from the engineers:

- TTF for a computer
  - Let X_i denote the TTF of the ith computer in service
  - $X_1, X_2, \dots$ are independent and **time-stationary** (i.e. identically distributed) random variables with common cdf F_X
    - ⇒ A new computer and a computer that has just been repaired have the same TTF
  - F_X is the Weibull distribution with parameters $\alpha = 2$, $\beta = 812$
    - ⇒ Expected TTF is 720 hours (30 days) with standard deviation 376 hours (16 days)
      - ◇ Due to their flexible nature, Weibull distributions are commonly used for failure times
- Service time
  - Let R_i denote the time required to repair the ith computer failure
  - $R_1, R_2, \dots$ are independent and time-stationary random variables with common cdf F_R
  - Based on service records, F_R is the uniform distribution on $[4, 48]$
- $X_1, X_2, \dots$ and $R_1, R_2, \dots$ are independent
  - ⇒ Repair time of a computer is not affected by its TTF or the number of times it has been repaired

3 Simulating the Leaky Bit Bucket

- We're interested in D, the time the entire system fails
- D is a random variable: a (complex) function of random variables $X_1, X_2, \dots$ and $R_1, R_2, \dots$
- Let's generate values of $X_1, X_2, \dots$ and $R_1, R_2, \dots$ and use these to simulate values of D
- We can describe this simulation algorithmically
- System logic from Bit Bucket engineers:
  - After a system is installed, the primary computer is started
  - When it fails, the backup computer is immediately started and a service call is made to Bit Bucket
  - If the primary computer is repaired before the backup computer fails, then the primary computer becomes the backup computer, and the former backup computer remains the primary computer
  - If at any time neither computer is available, the entire system fails
  - Only one computer can be repaired at a time, and are repaired first-come-first-served

- System state: the critical variable that characterizes system status

  - State space:

- System events of interest
  - $e_1 =$
  - $e_2 =$
- The **clock time** C_i of system event e_i is the time the next system event of type e_i occurs
  - When no type e_i event is pending, $C_i \leftarrow \infty$
- The ***n*th event epoch** T_n is the time at which the nth system event occurs
- At T_{n+1}, the time of the $(n+1)$st event, two things can happen:
  - The system state can change
  - The clocks can be reset
- How exactly?
- Let `random()` be a function that generates variates for Uniform$[0, 1]$
- Subroutine for system event e_1:

- Subroutine for system event e_2:

- Let's also create an "initial" system event e_0 representing the installation of the computer system:

$e_0()$:
1: $S_0 \leftarrow 0$ (initially no computers down)
2: $C_1 \leftarrow F_X^{-1}(\texttt{random()})$ (set clock for first computer TTF)
3: $C_2 \leftarrow +\infty$ (no pending repair)

- Putting this all together:

algorithm BitBucketSimulation:
1: $n \leftarrow 0$ (initialize system event counter)
2: $T_0 \leftarrow 0$ (initialize event epoch)
3: $e_0()$ (execute initial system event)

4: $T_{n+1} \leftarrow \min\{C_1, C_2\}$ (advance time to next pending system event)
5: $I \leftarrow \arg\min\{C_1, C_2\}$ (find index of next system event)
6: $C_I \leftarrow \infty$ (event I no longer pending)

7: $e_I()$ (execute system event I)
8: $n \leftarrow n + 1$ (update event counter)

9: go to line 4

Example 1. Suppose the first five values generated by $F_X^{-1}(\texttt{random()})$ are 877, 1041, 612, 36, and 975. In addition, suppose the first four values generated by $F_R^{-1}(\texttt{random()})$ are 17, 8, 39, and 9. Generate the sample path using the algorithm BitBucketSimulation.

Event counter n	System event I	Time T_n	State S_n	Failure clock C_1	Repair clock C_2

- S_n is the number of down computers when the nth system event occurs
- T_n is the time of the nth system event
- Let's combine these:

$$Y_t = \text{number of down computers at time } t \quad \text{for } t \geq 0$$

or equivalently,

- The **time average** of Y_t up to the nth event epoch is

Example 2. Using your simulated sample path from Example 1, graph Y_t. What is the time average of Y_t up to the 8th event epoch?

- Recall: we're interested D, the time of total system failure, which is:

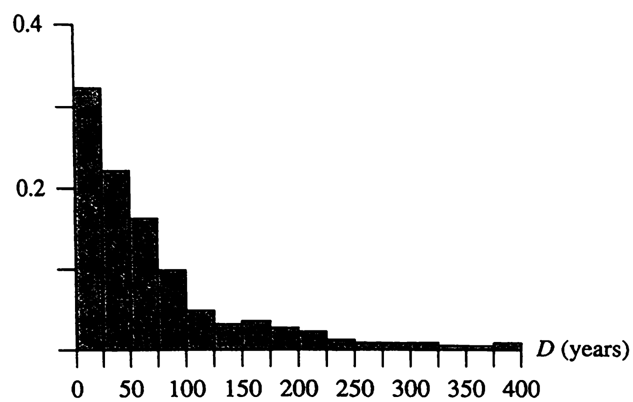
- The value of D generated by our simulation in Example 1 is

Example 3. Modify the algorithm BitBucketSimulation to record the value D generated by the simulation.

- To get information about the distribution of D, we run this simulation many times, say $m = 500$:

```
for r = 1 to m do
    algorithm BitBucketSimulation
end for
```

- Sample results:
  - Average of generated values of D: 551606 hours ≈ 63 years
  - Histogram of generated values of D:



  - 2% of the generated values of D are less than 2 years
  - Is this acceptable or unacceptable?