SFWR ENG 4E03

Kemal Ahmed

Fall 2015

Instructor: Vincent Maccio

Note: material covered in [Stats 3Y03 Summary](https://drive.google.com/file/d/0BxW61uJyyN8TNy1iUFE0ZlRMLTg/view) will not be covered in this summary

Contents

[Statistics 1](#_Toc435551734)

[Variance 2](#_Toc435551735)

[Exponential 2](#_Toc435551736)

[Uniform 2](#_Toc435551737)

[Binomial 2](#_Toc435551738)

[Operations Analysis 2](#_Toc435551739)

[Visitation Trick 4](#_Toc435551740)

[Summation Equations 4](#_Toc435551741)

[DTMC 5](#_Toc435551742)

[Matrices 5](#_Toc435551743)

[CTMC 5](#_Toc435551744)

[Poisson Process 5](#_Toc435551745)

[Kendall notation 6](#_Toc435551746)

[Variations 6](#_Toc435551747)

[M/M/C Provisioning 7](#_Toc435551748)

[Traffic Equations 7](#_Toc435551749)

[Jackson Networks 8](#_Toc435551750)

[Questions 8](#_Toc435551751)

# Statistics

**Poisson parameter** [λ]: rate

**Service rate** [μ]:

**Continuous Random Variable (CRV)**:



*Think chemistry, i.e. cancelling units*

## Variance



* Don’t change probability, but square X for calculation only



## Exponential

* **Mean** [E[X]]: 1/λ
  + a.k.a. Expected value
* **Variance**: 1/λ2
* **Probability Distribution Function (PDF)** [P(X=x)]: λe–λx/x!
* **Cumulative Distribution Function (CDF)** [f(x)]: CDF = ∫PDF, i.e. 1 – e–λx
* Memoryless
* not always for time

## Uniform

* **Variance**: (b–a)2/12
* **Mean**: (a+b)/2
* **PDF**: 1 / (b–a) , a ≤ x ≤ b
* **CDF**: 1
* **Uniform Distribution**: no memoryless property

## Binomial

* **Mean** [E[X]]: n × probability
* **Variance**: n × p × (1 – p)
* **Probability Distribution Function (PDF)** [P(X)]: (n c x)px(1–p)n–x
* **Cumulative Distribution Function (CDF)** [f(x)]: 

# Operations Analysis

**Device** [i]: units that are in terms of *i* are specific to an individual device or node within a system

**Total devices** [k]:

**Service Time** [S]: time per specific job

1/μ

**Visitation** [V]: given or projected visits/jobs (closed system); cannot be calculated; basically a probability

[E(V)]: calculated visit/job ratio

P(visit)∙total visits in previous node

**Demand** [D]: total service time for all jobs





**Bottleneck** [Dmax]: device with largest demand

**Time in system** [T]: time the job is in the system





If E[Z] = 0, T = R

**Response Time** [R]: time the job is *being processed* in the system

If E[Z] = 0, R = T

E[R] = E[RQ] + E[S]

**Users** [M]:

**Optimal users** [M\*]:



**Total Jobs** [N]: N=M in a closed system

* *Little’s Law*:
* 
* Steady state probability
  + M/M/1:
    - E[N] = λ/(μ–λ) = *ρ*/(1–*ρ*), if you have overall system λ
    - E[N] = ← probability × #jobs, if your λ or μ is different for each state
  + M/M/C: E[N] = ΣE[Ni] = ΣPλ/(μi–Pλ) = Σ(λi/(μi–λi))

**Think time** [Z]: time it takes the user to put a request in and start, it’s kinda like the frequency that users put in requests (seconds / request)



**Throughput** [X]: out-rate, max jobs / hour of full system



Note: andconverge at their lowest point, so equate them



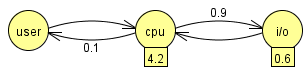
**Utilization** [ρ]: ratio that the time is busy



ρ = λ/ciμ

## Visitation Trick

If determining visitation at a node, establish a reference node from one of the incoming nodes, usually the user node, that has a returning percentage



Vuser = 1 = 0.1 ∙ VCPU

## Summation Equations

**Geometric Series**: , where 0 ≤ r ≤ 1 (because otherwise it would be unstable)



**Geometric Sequence**: 



Removing the annoying factors:



Take out a value so the integral takes out the i and i+1



# DTMC

**Discrete Time Markov Chains (DTMC)**:

[n]: number of tasks in queue / system



^think of it like series / parallel, where you add multiple connections out in different directions (parallel) and multiply connections stacked onto each other (series)

OR jobsin = jobsout

Steady state: n->∞

For discrete: use the sum of the X’s, so E[X] = Σ(P(X=i)∙Xi) and E[X2] = Σ(P(X=i)∙Xi2)

## Matrices

Rows: equations for nodes going out (add up to 1)

Columns: equations for nodes coming in

# CTMC

## Poisson Process

**Counting Process**: a way of determining the time between consecutive occurrences of an event

**Poisson Process**: a *counting process*, whose time between arrivals uses Exponential Distribution

* λtotal = Σλi
  + you can also split up λ into multiple λs
* Not only do you see each second as time independent, each stream of probabilities is independent
* P(x;λ) = e–λλx/x!
  + [x]: things will happen
  + [λ]: rate; λ = αt
* [α]: expected number of events during unit interval
* [t]: time interval length
* 

## Kendall notation

**Job Processing time** [μ]: rate of jobs leaving system (jobs/sec)

μ = 1/ processing\_time\_per\_job

M/M/1 Queue

[M]: time between arrivals is Markovian (Memoryless) ~ exp(λ)

[M]: job processing times are Markovian (Memoryless) ~ exp(μ)

[1]: single server

(Σpout)×πi = Σpjπj, j=0..n, j≠i

π0: percent of time that the queue is empty

Attributes:

* FIFO
* Infinite buffer

### Variations

* M/M/2 Queue: same, except 2 servers
* M/M/C Queue: *C* servers
* M/Ek/C: Erlang k, i.e. series of exponential
* H()/M/C: hyperexpontial distribution
* PH/M/C: phase type, i.e. any combination of any number of exponentials with any rate
* M/G/C: General distribution
* G/G/1: has not been solved yet
* M/M/1/1: 1 server, 1 job

[c]: number of servers

Think: one queue goes to multiple servers







### M/M/C Provisioning

Useful if multiple jobs are sharing the same queue

**Blocking Probability** [PQ]: probability that a process will be blocked when entering the system and be placed in the queue

Erlang-C Equation: 

Given λ and μ, what should c be so PQ < ρ

**Waiting time in queue** [RQ]: response time of queue



M/M/1: 

M/M/C: 

[Q]: transition matrix



Replace i <--> j to get qjj and qji.

## Traffic Equations

For each node, what is the number of jobs entering?

λx = R + ΣPi,entering ∙ λi,entering

response rate + probability of each job entering

# Jackson Networks

hi

# Questions

* Assignment 5, Q2 states in ready queue??
* Assignment 6, Q3, M/M/C