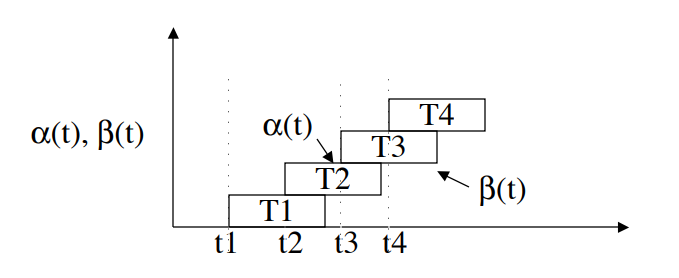
Question 1

1 - a - i

* For every second a job is in the system it contributed 1 to the integral
* The number of seconds it is in the system is what we mean by response time so the total contribution is the response time



alpha(t) = number of arrivals by time t

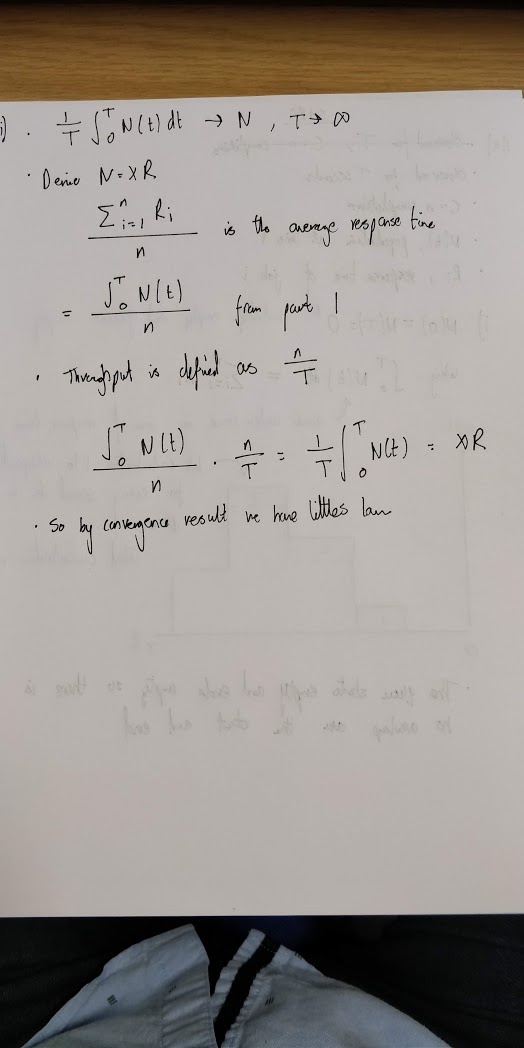
beta(t) = number of departures by time t

ti = arrival time of ith job

Ti = response time of ith job

N(t) = Number of jobs in system t = alpha(t) - beta(t)

1 - a - ii



1-b-i

* Nodes 2 and 3 have the same (the max) service demand so are both the bottleneck
* Throughput under low load from N/(D + Z) = 0.0249 jobs/second
* Throughput under high load from 1/Dmax 0.28 jobs/second

1-b-ii

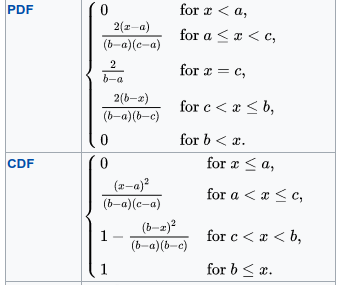
* Node 2 is now the sole bottleneck / Maximum throughput under high load stays the samet
* Mean population of node 2 queue will increase

1-b-iii

* State: Three ints representing the population of the three queues
* Events: Completion at nodes 1, 2, 3 / End of think time
* Completion at node 1
  + Decrement node 1 counter
  + Generate a uniform random number r in [0, 1]
  + If r < 180/300 = ⅗ route to node 2 otherwise to node 3
  + Increment node count for node 2/3 and schedule completion for itself from service time
* Completion at node 2 (symmetric for 3)
  + Decrement node 2 counter
  + Generate a uniform random number r in [0, 1]
  + If r < 1/300 route to users otherwise to node 1
  + If routed to users
    - Schedule end of think time
  + If routed to node 1
    - Increment node 1 count & schedule completion
* End of think time
  + Increment node 1 count and schedule completion

2-a-i

* Find the cdf
  + Results from wikipedia (take a = 0)



* We want to find x such that F(x) = U
  + If U <= c/b we use the first cdf, otherwise we use the second
  + First case: Rearrange to get x = sqrt(bcU)
  + Second case: x = b - sqrt(U \* b \* (b - c))

2-a-ii

We generate X using bU(0, 1) and we generate Y using (2/b)U(0,1). We can get the success rate p to be the area under the distribution f(x) over the rectangle which is 1/2

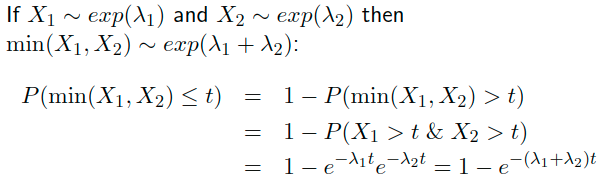
Consider the random variable R to be the number of failed attempts. So we have

P(R = r) = (1 - p)^{r} \* p

E(R) = (1 - p) / p from general result of geometric distribution

we then calculate 2 (we need to generate both X and Y) \* (E(R) + 1(we need at least one iteration)) hence we need 4 U(0, 1)

2-b-i



The Holding Time of s is the minimum of the Transition Distribution from s to any other state i.e.

(for restricted range)

2-b-ii



Explanation comes from expanding pQ=0 and remembering pi\*qii = pi\*(-sum(qij))

2-b-iii

* Time having already elapsed doesn’t affect the expected time to an event
* State transitions aren’t affected by state history

4-a-i

* Use M/M/1 queue formulas

\rho = 0.6 / 1 = 0.6

We get mean queue-length by \rho / (1 - \rho) = 1.5

We get mean buffer length by (\rho)^{2} / (1 - \rho) = 0.9

4-a-ii

\sum\_{k = 0}^{m} P(t{1} = k) P(N\_{2} = m - k)

= \sum\_{k = 0}^{m} \rho ^ {k} (1- \rho) \rho ^ (m - k) \* (1 - \rho)

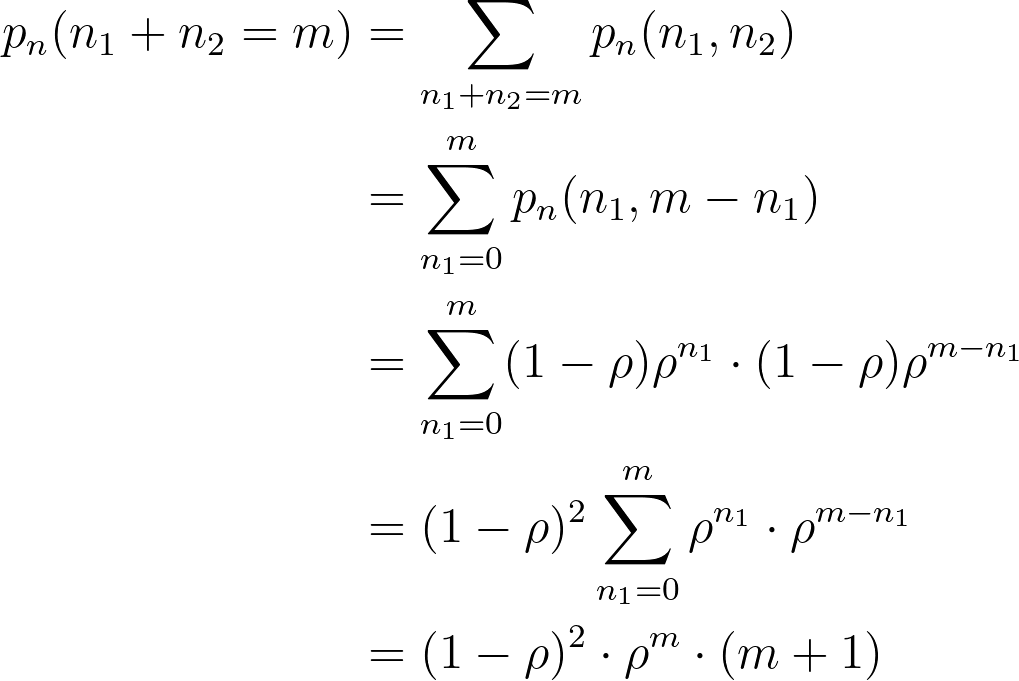
= \sum\_{k = 0}^{m} \rho ^ {m} (1- \rho) ^ {2}

= (1- \rho) ^ {2} \sum\_{k = 0}^{m} \rho ^ {m}

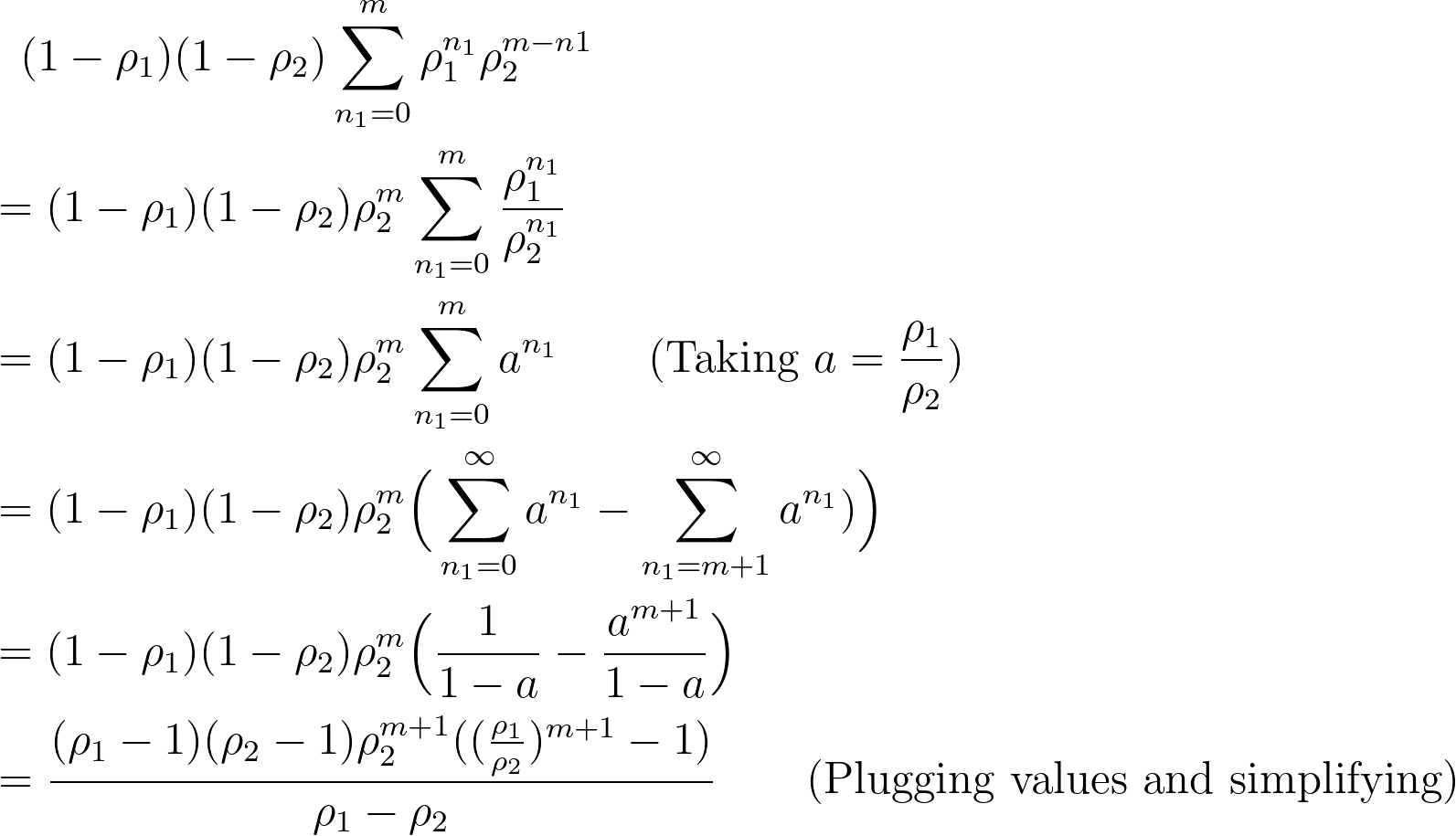
= (1- \rho) ^ {2} \* (1 - \rho ^ {m + 1}) / (1 - \rho)

= (1 - \rho) \* (1 - \rho ^ {m + 1})

Giuliano’s Solution during 2020/2021 Revision Lecture:



Now if we have different service rates, we have \rho\_1 != \rho\_2, and we have the new formula:

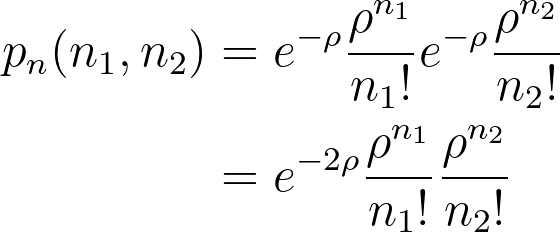


Note: We can assume a<1 <=> \rho\_1 < \rho\_2 because if that’s not the case, we would just take p(m-n2, n2) (instead of p(n1, m-n1)) and have the same formula with \rho\_1 and \rho\_2 being inverted.

4-a-iii

please help

Giuliano’s Solution during 2020/2021 Revision Lecture:



(We take the joint probability by taking the individual p\_ns for the M/M/infinity queue)

Here, \rho = \gamma / \mu (original case with identical service rates)