The Role of Advanced Math in Teaching Performance Modeling

Ziv Scully
Cornell University
School of Operations Research and Information Engineering
zivscully@cornell.edu

ABSTRACT

How should we teach performance modeling without assuming a deep mathematical background? One approach is to focus on rigorously studying relatively simple stochastic models that do not require too much math background. But this may leave students underprepared to reason about systems in practice. They have multiple servers, bursty arrivals, heavy tails, and other features that demand more complex stochastic models. Reasoning about these phenomena calls for advanced tools from performance modeling theory, but rigorously learning such tools requires more math background than many computer science and engineering students have.

In my view, the main obstacle to teaching advanced theoretical tools is the mathematical rigor. I believe we can teach such tools accessibly by dispensing with some of the rigor. In this (opinionated!) abstract, I argue that students would be well-served by advanced theoretical tools, and I outline what teaching those tools with less rigor might look like.

1. THE NEED FOR ADVANCED MATH IN PERFORMANCE MODELING

In my view, the main goals of performance modeling are to *describe* and *analyze* systems in service of improving their design. Math plays a critical role in both goals.

- We formally describe systems as mathematical models, typically stochastic models.
- We analyze said models either in theory, using tools from applied probability; or in simulation, in which case probability and statistics help us decide what to simulate and interpret results.

A typical path for performance modeling courses is to introduce students to a mathematical modeling framework, then teach strategies for analyzing models in that framework.

1.1 Markov Chains Fall Short

A modeling framework of choice for many texts on performance modeling and stochastic processes is Markov chains on discrete state spaces [4, 5, 12, 16, 23, 25, 30]. Markov chains are appealing for several reasons.

- One can define and prove fundamental theorems about Markov chains with elementary tools. For example, on finite state spaces, one can define the transition kernel as a matrix, and one can frame questions about convergence and mixing times in terms of linear algebra.
- One can often exactly analyze Markov chains with elementary tools. For example, while guessing a Markov chain's stationary distribution can be difficult, confirming a guess requires only checking balance equations.

This combination makes it feasible for students to learn to define and exactly analyze Markov chain models fully rigorously.

However, I see two main downsides to focusing a performance modeling course around Markov chains. The first downside is concrete: Markov chains have limited modeling power. It is difficult to model queueing systems with general service time distributions with Markov chains, outside of specific cases like the M/G/1 and G/M/1 where an embedded discrete-time approach is possible. Phase-type distributions can approximate general distributions, but their tails never resemble power-law tails, which are ubiquitous in computer systems and beyond [7, 22, 26, 31]. One can work around this by adding more states to the Markov chain so that jobs have infinitely many possible phases. But this yields Markov chains that are challenging or impossible to exactly analyze.

This brings us to a second downside of focusing on Markov chains: they tempt a focus on exact analysis, as opposed to approximations or bounds. It is true that many Markov-chain queueing models admit exact analysis, such as the famously general Jackson networks [16]. But exact analysis is intractable for most models, particularly multiserver models like the M/G/k [11, 17]. Why do courses focus so heavily on exact analysis? I suspect it is because exact analysis is often easier than approximations. Verifying the balance equations for Jackson networks, for example, is just algebra. Approximately analyzing systems like the M/G/k requires more advanced machinery [13, 27].

In summary: Markov chains are an appealing because they can be taught fully rigorously without assuming an extensive math background. But the set of Markov chains that can be rigorously analyzed with only elementary tools is limited. I believe this status quo can leave students unprepared to reason about the systems they will encounter in practice.

1.2 Tools Students Need to Build and Analyze Realistic Models

What features of practical systems might students need to reason about, and what mathematical tools do they need to do so? Below are four tools I think would prove useful, but this is not an exhaustive list.

The first tool is *general state spaces*. Even very simple models require complex state spaces. Consider an M/G/1

queue where we wish to track the remaining work of each job. The system state is a list $[r_1, \ldots, r_n]$, where r_i is the remaining work of job i. The state space is thus $\bigsqcup_{i=0}^{\infty} \mathbb{R}_+^i$, which is not even finite-dimensional. Computer systems, with their many layers of abstraction, have even more complicated state spaces. For instance, a data center has many physical servers, each of which hosts many virtual machines, each of which runs many programs, each of which has its own internal software state.

The second tool is *drift* of a system's state, or the drift of functions of the state. An especially important example is drift of the total remaining work of all jobs in a system, which is related to the system's *load*. Drift is important because they give students a first indication about how a system might behave. For example, if load is greater than the average work rate, the system will be unstable. Practical factors like parallelism [3] and garbage collection [14] can make figuring out a computer system's load difficult. Learning to reason about drift is one way to understand load and stability more broadly. Drift methods, as pioneered by Eryilmaz and Srikant [10], are also one of the principal tools queueing theorists have for approximately analyzing otherwise intractable systems.

The third tool is *expectations from varying perspectives*. There are many ways we might look at a system's average behavior, such as averaging over time, averaging over sample paths, and averaging over jobs. Exactly which average is appropriate depends on what metrics we are trying to analyze in theory or measure in simulation. To name two recent examples:

- Kumar et al. [18] investigate the degree to which websites rely on a small number of centralized services, such as content delivery networks. Much of their analysis is in terms of website averages, namely the fraction of websites from a given list satisfy a criterion. But there are other types of averages that one could investigate. One example is website-visit averages, namely the fraction of website visits that satisfy a criterion; or, equivalently, a website average where each website is weighted by the number of visits it gets.
- Atre et al. [1] design an algorithm for caching with bursty requests. Types of averages that could be relevant include time averages, request averages, and request-burst averages. They use request-burst averages in their algorithm, but one could imagine a different average would yield different results, and it is not a priori obvious which type of average is the right choice.

The fourth tool is *large-scale approximations*, such as *mean-field models*. Today's computer systems are certainly large-scale, and mean-field models can quickly give some insight into a large-scale system's behavior. This insight might be from numerically evaluating the mean-field model, or from proving theorems about it. Famously, mean-field models have been used to study dispatching [20]. I think mean-field models could give students insight into metastability [9], a phenomenon that is behind many recent failures in computer systems [6, 14].

1.3 What Makes These Tools Advanced

All four of the tools above are typically considered mathematically advanced. When taking a fully rigorous approach, this is certainly the case.

- Studying stochastic processes on general state spaces raises measure theoretic concerns.
- In continuous-time models, defining the drift of a Markov process involves the process's infinitesimal generator, an operator whose domain is hard to define.
- Defining expectations from certain perspectives involves what seems to be conditioning on a probability-zero event. For example, one cannot simply define an arrival average as a time average conditional on an arrival occurring. *Palm calculus* [2] is a rigorous method of overcoming this, but the formalism can be intimidating even to experts (myself included!), let alone students.
- One typically derives mean-field models as infinite-size limits of large but finite-size models. But this requires reasoning about limits of stochastic processes and raises concerns about when large-time and large-size limits may be exchanged.

Teaching these tools rigorously thus seems infeasible in the context of an engineering course.

My view is that the main thing that makes these tools advanced is the rigor. I think there is a way to teach them that dodges much of the rigor while still providing value to students. This is the subject of the next section.

2. HOW WE MIGHT TEACH THE MATH PERFORMANCE MODELING NEEDS

The conclusion of the previous section is that students would be well served by advanced theoretical tools, but that there is not time to teach them rigorously in the context of an engineering course. The natural question is: how do we teach such tools less rigorously? In this section, I outline the beginnings of an approach for doing so. Given the amount of hand-waving involved, I will refer to my proposed approach using the acronym *WAVE*.

The main issue with hand-waving without a rigorous foundation is that it can leave students unsure about exactly when hand-waving is allowed. In WAVE, I hope to state precise rules for hand-waving, which I refer to as *principles* (as distinct from theorems), that work "most of the time". To help students solidify their understanding of when principles apply, I plan to "prove" most principles in some way. This may be by picture, by computation, or even by appeal to empirical data. To further guide students, WAVE provides some common patterns for applying principles, which I refer to as *recipes*.

A downside of teaching using high-level principles and recipes, as opposed to more rigorous low-level statements, is that each individual topic might need many principles and recipes. To get around this, I hope to focus WAVE on a small number of flexible principles and recipes that can be applied in many different contexts. The first step is to introduce a flexible modeling framework to which those principles apply.

2.1 Model with Markov Processes

I propose we teach students to model systems as Markov processes on general state spaces in either discrete or continuous time. The discrete time version of this is not so different from the current Markov chain approach. The main difference is one of emphasis: rather than focusing attention on easily tractable chains like birth-death processes, the main goal is to capture the key dynamics of a system. A learning goal

for students should be to decide what aspects of a system should be tracked as part of its state in order to fully specify its transition dynamics.

I focus hereafter on continuous time, as that is the setting where technical issues arise in a fully rigorous treatment. An immediate question is: how should students specify the dynamics of a continuous-time Markov process? In WAVE, students specify dynamics in an informal but clear pseudocode. For example, Algorithm 2.1 below describes an M/G/1 with arrival rate λ and job size distribution S, tracking the size and attained service of each job as part of the state. There are two ways the state can change: *continuously*, such as a job being served; and through *jumps*, such as a job arriving or departing. This is a natural way to describe piecewise-deterministic Markov processes [8], and I suspect it will suffice for most queueing applications.

Algorithm 2.1 M/G/1 Queue with Known Job Sizes

STATE: a list $X = [(s_1, a_1), \ldots, (s_n, a_n)]$, where $n \in \mathbb{N}$ and $0 \le a_i \le s_i$ for all $i \in \{1, \ldots, n\}$
- s_i represents the size of job i
- a_i represents the attained service of job i

DYNAMICS:
(A) *Continuously* while $n \ge 1$:
  Increase a_1 at rate 1
(B) *Jump* at hazard rate λ:
  Sample s_{n+1} from S
  Set a_{n+1} to 0
  Append (s_{n+1}, a_{n+1}) to X
(C) *Jump* when $n \ge 1$ and $a_1 = s_1$:
  Delete (s_1, a_1) from X *(and shift indices)*

The main choice students make when modeling a system is what information to include in the system state. One aspect of this choice is deciding what information should be considered known or unknown. One can view Algorithm 2.1 as being an M/G/1 with known job sizes, because the job sizes s_i are tracked in the system state. A variation with unknown job sizes, which makes use of the hazard rate $h_S(\cdot)$ of the job size distribution, is given in Algorithm 2.2 below.

Algorithm 2.2 M/G/1 Queue with Unknown Job Sizes

STATE: a list $X = [a_1, \ldots, a_n]$, where $n \in \mathbb{N}$ and $a_i \ge 0$ for all $i \in \{1, \ldots, n\}$
- a_i represents the attained service of job i

DYNAMICS:
(A) *Continuously* while $n \ge 1$:
  Increase a_1 at rate 1
(B) *Jump* at hazard rate λ:
  Set a_{n+1} to 0
  Append a_{n+1} to X
(C) *Jump* at hazard rate $h_S(a_1)$:
  Delete a_1 from X *(and shift indices)*

Using pseudocode does not completely shield students from learning math. For instance, hazard rates feature in both Algorithms 2.1 and 2.2. But I believe some version of pseudocode could be intuitive, especially to students with programming background.

A question one might ask is what would need to be done to rigorize pseudocode descriptions like Algorithms 2.1 and 2.2. I suspect the main obstacle would be to formalize what conditions are admissible for jumps. One would likely want to check for non-explosiveness, meaning the process has probability zero of having infinitely many jumps in a finite time interval. I believe students can benefit from pseudocode descriptions without worrying about explosiveness in most cases.

2.2 Measure with Expectations from Varying Perspectives

Having defined a system model, how should we define metrics of interest, such as mean waiting time? WAVE is focused on metrics that can be expressed in the form

$$\mathbf{E}_\ell[f(X)],$$

in which:
- X is the system state.
- f is some numerical function of the system state.
- $\mathbf{E}_\ell[\cdot]$ is a expectation from *perspective* ℓ. We use the letter ℓ because perspectives will often be *lines* or *labels* in the pseudocode description of X's evolution.[1]

I explain all three aspects in more detail below via two examples. Both examples use the M/G/1 with known job sizes from Algorithm 2.1. The metrics of interest are mean waiting time and mean queue length.

We first consider mean waiting time. A job's waiting time as the amount of work in the system when it arrives, so we start defining a function for the amount of work:

$$w\big([(s_1, a_1), \ldots, (s_n, a_n)]\big) = (s_1 - a_1) + \cdots + (s_n - a_n).$$

That is, $w(x)$ is the amount of work when the system is in state x. The system state X evolves randomly over time, so we write $X(t)$ for the system state at time t. We suppose the system runs for a very long time interval $[0, T]$. If the times jobs arrive are $t_1, \ldots, t_N$, where N is the number of arrivals in $[0, T]$, then the mean waiting time of jobs that arrive during the interval is

$$\frac{1}{N} \sum_{i=1}^{N} w(X(t_i)).$$

More generally, we define $\mathbf{E}_{\mathsf{arrival}}[f(X)]$ for function f to be

$$\mathbf{E}_{\mathsf{arrival}}[f(X)] = \frac{1}{N} \sum_{i=1}^{N} f(X(t_i-)).$$

So mean waiting time is $\mathbf{E}_{\mathsf{arrival}}[w(X)]$.

We now consider mean queue length. We start by defining a function for the queue length:

$$q\big([(s_1, a_1), \ldots, (s_n, a_n)]\big) = (n-1)^+ = \max\{n-1, 0\}.$$

That is, $q(x)$ is the number of jobs in the queue, not counting the job in service, when the system is in state x. Mean queue length is a time average, so we next define $\mathbf{E}_{\mathsf{arrival}}[f(X)]$ for function f to be

$$\mathbf{E}_{\mathsf{time}}[f(X)] = \frac{1}{T} \int_0^T f(X(t))\, \mathrm{d}t.$$

[1] Such expectations are sometimes called *Palm expectations* in the literature [2, 21].

So mean queue length is $\mathbf{E}_{\mathsf{time}}[q(X)]$.

Above, arrival and time are two examples of a *perspectives*. A perspective is a way of taking a long-run average. I suspect that most useful perspectives will be time and perspectives associated with *jump labels*, meaning lines of pseudocode that specify jumps. The arrival perspective is a special case of this: $\mathbf{E}_{\mathsf{arrival}}[\cdot] = \mathbf{E}_{(\mathsf{B})}[\cdot]$, because (B) is the label corresponding to arrivals in Algorithm 2.1. In general, for jump labels ℓ, we define $\mathbf{E}_\ell[\cdot]$ to be the average taken over the N_ℓ times $t_{\ell,1}, \ldots, t_{\ell,N_\ell}$ a jump occurs due to label ℓ:

$$\mathbf{E}_\ell[f(X)] = \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} f(X(t_{\ell,i}-)).$$

One could imagine defining perspectives for continuous labels, too. Again using Algorithm 2.1 as an example, (A) would be the perspective of a busy system, meaning an average that excludes times the server is idle. It is not yet clear to me whether such perspectives would be useful.

2.3 Quantify with WAVE Equality

We have defined expectations as long run averages on a single sample path $X(t)$ for $t \in [0, T]$. But we have not yet said anything about the probability space underlying $X(t)$. WAVE makes no explicit mention of probability spaces, instead focusing on a single long sample path. How, then, can we use probabilistic information, such as the fact that the number of arrivals N during $[0, T]$ should be approximately λT? Traditionally, we would use the law of large numbers to deduce this.

Instead of defining a probability space, I propose we *make the law of large numbers an axiomatic principle*, or a small number of related principles. One such principle (or a special case thereof) would say that when jumps occur at constant hazard rate λ during a union of intervals of total length T, the number of jumps N is[2]

$$N \approx \lambda T.$$

In a rigorous presentation, the $\approx$ above would refer to a type of convergence, such as almost sure convergence as $T \to \infty$. Under WAVE, $\approx$ is a new equivalence relation which we call *WAVE equality*.

Intuitively, WAVE equality means "equal enough for practical purposes if the sample path is long enough". But there is no direct definition of WAVE equality. The closest we can get to giving a formal definition of WAVE equality is to say it is defined inductively via principles. That is, the only way to show that two quantities are WAVE-equal is to use a principle, and two quantities are WAVE-equal if some application of principles can show them to be. Of course, the principles are informal, so this is still not a formal definition.

In WAVE, most interesting principles hold only under WAVE equality, as opposed to ordinary equality. For example, the WAVE version of Little's law [19] is

$$\mathbf{E}_{\mathsf{arrival}}[q(X)] \approx \lambda \mathbf{E}_{\mathsf{time}}[w(X)].$$

We will soon give an argument for Little's law using WAVE. Another example is the PASTA (Poisson Arrivals See Time Averages) principle [32], whose WAVE version says that for all functions f,

$$\mathbf{E}_{\mathsf{arrival}}[f(X)] \approx \mathbf{E}_{\mathsf{time}}[f(X)].$$

[2]It may be necessary to divide both sides by T, because $N - \lambda T \approx 0$ seems more likely to lead to errors than $N/T - \lambda \approx 0$.

2.4 Analyze with the Main WAVE Principle

Having defined metrics like mean waiting time and mean queue length, how do we actually compute them? The starting point is the main WAVE principle.[3]

> For any function f, the long-run integral $\int_0^T f(X(t))\,\mathrm{d}t$ is WAVE-equal to any other sum or integral that computes the same signed area, *ignoring edge effects*.

WAVE's name is a (somewhat clumsy) acronym for this principle: *When Averaging, Vilipend Edges*.[4] I hope defining "edge effect" informally as "only affecting the area near times 0 and T" will suffice. If needed, a more formal definition could define it as a difference between the areas that scales as $o(T)$ in the $T \to \infty$ limit, but it remains unclear when that actually holds.

To demonstrate the value of the main WAVE principle, let us derive Little's law for the M/G/1. The key idea, as in the usual formal proof [19], is to look at $\int_0^T q(X(t))\,\mathrm{d}t$ in two ways. Recall our notation that N arrivals happen during $[0, T]$ at times $t_1, \ldots, t_N$.

- By definition, $\int_0^T q(X(t))\,\mathrm{d}t = T\mathbf{E}_{\mathsf{time}}[q(X)]$.
- By definition, $\sum_{i=1}^N w(X(t_i-)) = N\mathbf{E}_{\mathsf{arrival}}[w(X)]$.

By the main WAVE principle, thanks the usual trick of "slicing horizontally", these compute the same area modulo edge effects, so

$$T\mathbf{E}_{\mathsf{time}}[q(X)] \approx N\mathbf{E}_{\mathsf{arrival}}[w(X)].$$

Finally, recall that $N \approx \lambda T$ by a law of large numbers principle, which yields Little's law.

A close relative of the main WAVE principle is the *rate conservation law* [21]. One can view it as saying that for any function f,

$$\frac{f(X(T)) - f(X(0))}{T} \approx 0.$$

The power of the rate conservation law comes from expanding the left-hand side as a combination of sums and integrals which constitute all the changes in $f(X)$. This can be done in a systematic way, though it requires some more notation (which we will not define formally). Consider again the M/G/1 from Algorithm 2.1. Each of the labels contributes one way the state can change.

(A) Service yields $\mathbf{E}_{\mathsf{time}}\big[\mathbf{1}(\texttt{length}(X) \geq 1)\,\partial_{a_1} f(X)\big]$.

(B) Arrivals yield $\lambda\mathbf{E}_{\mathsf{arrival}}\big[f\big(\texttt{join}(X, [(S, 0)])\big) - f(X)\big]$.

(C) Departures yield $\lambda\mathbf{E}_{\mathsf{departure}}\big[f\big(\texttt{dropFirst}(X)\big) - f(X)\big]$.[5]

[3]As in the footnote about the law of large numbers, it may be necessary to say that the areas are WAVE-equal only after dividing by T.

[4]"Vilipend" means to regard something as having little value, a fact I learned from a thesaurus while writing this abstract.

[5]Strictly speaking, one needs to show that the average departure rate is λ. This is intuitive, but for a slightly more detailed argument, one can first write this term with a different rate $\lambda_{\mathsf{departure}}$. Applying the rate conservation law to $f(x) = \texttt{length}(X)$ then implies $\lambda_{\mathsf{departure}} \approx \lambda$.

The rate conservation law says that the sum of these three terms is WAVE-equal to zero. One can carry out the same process for essentially any pseudocode.

The rate conservation law is very powerful. Applying the rate conservation law to $f(x) = w(x)$ yields, after some computation,

$$\mathbf{E}_{\mathsf{time}}[\mathbf{1}(n(X) \geq 1)] \approx \lambda \mathbf{E}[S],$$

a well-known characterization of an M/G/1's load. Applying it to $f(x) = \frac{1}{2}(w(x))^2$ yields, when combined with PASTA, the PK formula for mean work:

$$\mathbf{E}_{\mathsf{time}}[w(X)] \approx \frac{\frac{1}{2}\lambda \mathbf{E}[S^2]}{1 - \lambda \mathbf{E}[S]}.$$

The above derivations are specific instances of the first of two more general recipes:

- To analyze $\mathbf{E}_{\ell}[(f(X))^p]$, try applying the rate conservation law to $(f(X))^{p+1}$.
- To analyze $\mathbf{E}_{\ell}[\exp(\theta f(X))]$, try applying the rate conservation law to $\exp(\theta f(X))$.

Variants of these recipes are actually used in current queueing research [15], including my own [28, 29]. For instance, applying the recipe to the M/G/k, one obtains, roughly speaking,

$$\begin{aligned}\mathbf{E}_{\mathsf{time}}[\text{M/G/}k \text{ work}] = \mathbf{E}_{\mathsf{time}}[\text{M/G/1 work}] \\ + \mathbf{E}_{\mathsf{idle}}[\text{M/G/}k \text{ work}],\end{aligned}$$

provided the server speeds are scaled such that the M/G/1 and M/G/k have the same total server speed. While queueing researchers then attempt to explicitly bound terms like $\mathbf{E}_{\mathsf{idle}}[\text{M/G/}k \text{ work}]$, one still learns something from just that expression. Whenever a server is idle in the M/G/k, there are $k-1$ or fewer jobs present, so $\mathbf{E}_{\mathsf{idle}}[\text{M/G/}k \text{ work}]$ is, roughly, the "work of at most $k-1$ jobs".

One can use the main WAVE principle and rate conservation law to prove other helpful principles. Among these are the renewal-reward theorem [12] and its more advanced cousin, the Palm inversion formula [2]. These are especially helpful because they help relate expectations from different perspectives to each other.

2.5 Unresolved Questions

There are, of course, many questions one would need to answer before teaching a course using WAVE. Below are just a few of these questions.

What does a full foundation for WAVE look like? What other definitions, principles, and recipes do we need? Perhaps we need principles that specify what counts as an edge effect. It would be valuable to have recipes for building a mean-field model given a model of one part's dynamics. It may even help to specify a more formal modeling language for writing WAVE pseudocode. Languages for modeling cyber-physical systems [24] could serve as inspiration.

For what audiences is WAVE most appropriate? I plan to teach a small part of an undergraduate stochastic processes course in a style similar to WAVE. Is this too ambitious? In the other direction, could parts of WAVE, such as the idea of expectations from varying perspectives, be valuable for audiences that are even less technical?

What does WAVE lose by sacrificing rigor? Rigorous probability theory exists for a reason. What are the most important dangers to look out for when proceeding non-rigorously? Can we teach students to identify situations that require extra attention to rigor? Considering this question might help decide whether it is worth emphasizing the distinction between WAVE equality $\approx$ and ordinary equality.

ACKNOWLEDGMENTS

Many thanks to Vittoria de Nitto Personè and Y. C. Tay for organizing TeaPACS and inviting me to participate in its 2023 edition. Thanks to Runhan Xie for several helpful discussions about his experience learning stochastic processes. Finally, thanks to Mor Harchol-Balter, my former advisor, for many discussions about teaching and presenting during my PhD years.

This work was done in part while I was an NSF FODSI postdoc at Harvard and MIT, supported by NSF grant nos. DMS-2023528 and DMS-2022448.

References

[1] Nirav Atre, Justine Sherry, Weina Wang, and Daniel S. Berger. 2020. Caching with Delayed Hits. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM 2020)*. ACM, Virtual Event, USA, 495–513. doi:10.1145/3387514.3405883.

[2] François Baccelli and Pierre Brémaud. 2003. *Elements of Queueing Theory: Palm-martingale Calculus and Stochastic Recurrences* (2 ed.). Number 26 in Applications of Mathematics. Springer, Berlin, Germany.

[3] Benjamin Berg. 2022. *A Principled Approach to Parallel Job Scheduling*. Ph. D. Dissertation. Carnegie Mellon University, Pittsburgh, PA.

[4] U. Narayan Bhat. 2008. *An Introduction to Queueing Theory*. Birkhäuser, Boston, MA. doi:10.1007/978-0-8176-4725-4.

[5] Pierre Brémaud. 2020. *Markov Chains: Gibbs Fields, Monte Carlo Simulation and Queues* (2 ed.). Number 31 in Texts in Applied Mathematics. Springer, Cham, Switzerland.

[6] Nathan Bronson, Abutalib Aghayev, Aleksey Charapko, and Timothy Zhu. 2021. Metastable Failures in Distributed Systems. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS 2021)*. ACM, Ann Arbor, MI, 221–227. doi:10.1145/3458336.3465286.

[7] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. 2009. Power-Law Distributions in Empirical Data. *SIAM Rev.* 51, 4 (Nov. 2009), 661–703. doi:10.1137/070710111.

[8] Mark H. A. Davis. 1984. Piecewise-Deterministic Markov Processes: A General Class of Non-Diffusion Stochastic Models. *Journal of the Royal Statistical Society: Series B (Methodological)* 46, 3 (July 1984), 353–376. doi:10.1111/j.2517-6161.1984.tb01308.x.

[9] Jing Dong. 2022. Metastability in Queues. *Queueing Systems* 100, 3-4 (April 2022), 413–415. doi:10.1007/s11134-022-09795-2.

[10] Atilla Eryilmaz and R. Srikant. 2012. Asymptotically Tight Steady-State Queue Length Bounds Implied by Drift Conditions. *Queueing Systems* 72, 3 (Dec. 2012), 311–359. doi:10.1007/s11134-012-9305-y.

[11] Varun Gupta, Mor Harchol-Balter, J. G. Dai, and Bert Zwart. 2010. On the Inapproximability of $M/G/K$: Why Two Moments of Job Size Distribution Are Not Enough. *Queueing Systems* 64, 1 (Jan. 2010), 5–48. doi:10.1007/s11134-009-9133-x.

[12] Mor Harchol-Balter. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, Cambridge, UK.

[13] Per Hokstad. 1978. Approximations for the $M/G/m$ Queue. *Operations Research* 26, 3 (1978), 510–523. doi:10.1287/opre.27.6.1115.

[14] Lexiang Huang, Matthew Magnusson, Abishek Bangalore Muralikrishna, Salman Estyak, Rebecca Isaacs, Abutalib Aghayev, Timothy Zhu, and Aleksey Charapko. 2022. Metastable Failures in the Wild. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2022)*. USENIX, Carlsbad, CA, 73–90.

[15] Daniela Hurtado-Lange and Siva Theja Maguluri. 2020. Transform Methods for Heavy-Traffic Analysis. *Stochastic Systems* 10, 4 (Dec. 2020), 275–309. doi:10.1287/stsy.2019.0056.

[16] Frank P. Kelly. 2011. *Reversibility and Stochastic Networks* (revised ed.). Cambridge University Press, Cambridge, UK.

[17] John F. C. Kingman. 2009. The First Erlang Century—and the Next. *Queueing Systems* 63, 1 (Nov. 2009), 3. doi:10.1007/s11134-009-9147-4.

[18] Rashna Kumar, Sana Asif, Elise Lee, and Fabián E. Bustamante. 2023. Each at Its Own Pace: Third-Party Dependency and Centralization around the World. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, 1 (Feb. 2023), 1–29. doi:10.1145/3579437.

[19] John D. C. Little. 2011. Little's Law as Viewed on Its 50th Anniversary. *Operations Research* 59, 3 (June 2011), 536–549. doi:10.1287/opre.1110.0940.

[20] Michael Mitzenmacher. 2001. The Power of Two Choices in Randomized Load Balancing. *IEEE Transactions on Parallel and Distributed Systems* 12, 10 (Oct. 2001), 1094–1104. doi:10.1109/71.963420.

[21] Masakiyo Miyazawa. 1994. Rate Conservation Laws: A Survey. *Queueing Systems* 15, 1 (March 1994), 1–58. doi:10.1007/BF01189231.

[22] Jayakrishnan Nair, Adam Wierman, and Bert Zwart. 2022. *The Fundamentals of Heavy Tails: Properties, Emergence, and Estimation*. Number 53 in Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, UK. doi:10.1017/9781009053730.

[23] James R. Norris. 1997. *Markov Chains*. Cambridge University Press, Cambridge, UK. doi:10.1017/CBO9780511810633.

[24] André Platzer. 2018. *Logical Foundations of Cyber-Physical Systems*. Springer, Cham, Switzerland. doi:10.1007/978-3-319-63588-0.

[25] Sheldon M. Ross. 2014. *Introduction to Probability Models* (11 ed.). Elsevier, Amsterdam, The Netherlands.

[26] Hiroshi Sasaki, Fang-Hsiang Su, Teruo Tanimoto, and Simha Sethumadhavan. 2017. Why Do Programs Have Heavy Tails?. In *2017 IEEE International Symposium on Workload Characterization (IISWC 2017)*. IEEE, Seattle, WA, 135–145. doi:10.1109/IISWC.2017.8167771.

[27] Alan Scheller-Wolf and Rein Vesilo. 2006. Structural Interpretation and Derivation of Necessary and Sufficient Conditions for Delay Moments in FIFO Multiserver Queues. *Queueing Systems* 54, 3 (Nov. 2006), 221–232. doi:10.1007/s11134-006-0068-1.

[28] Ziv Scully. 2022. *A New Toolbox for Scheduling Theory*. Ph. D. Dissertation. Carnegie Mellon University, Pittsburgh, PA.

[29] Ziv Scully, Isaac Grosof, and Mor Harchol-Balter. 2020. The Gittins Policy Is Nearly Optimal in the M/G/k under Extremely General Conditions. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 3, Article 43 (Nov. 2020), 29 pages. doi:10.1145/3428328.

[30] Y. C. Tay. 2014. *Analytical Performance Modeling for Computer Systems* (2 ed.). Morgan & Claypool, San Rafael, CA.

[31] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E. Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. 2020. Borg: The next Generation. In *Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys 2020)*. ACM, Heraklion Greece, 1–14. doi:10.1145/3342195.3387517.

[32] Ronald W. Wolff. 1982. Poisson Arrivals See Time Averages. *Operations Research* 30, 2 (1982), 223–231.