Course Manual Queueing Theory and Simulation

EBB074A05, 2021-2022

Nicky D. van Foreest

2022-02-01

Contents

1 Schedule as used during the course

The schedule is at the top for easy reference.

Week	Lecture	Sections
1	1	1.1, 1.2, 2.1, 22
	2	2.3, 2.4
2	3	3.1, 3.2
	4	3.3, 3.4
3	5	4.1, 4.2
	6	4.3, 4.4
4	7	4.5, 5.1
	8	5.2, 5.3
5	9	5.4, 6.1
	10	6.2, 6.3
6	11	6.4, 6.5
	12	6.5, 6.6
7	13	7.1, 7.2
	14	7.3, 7.4

2 Overview of the Course

Queueing systems pervade our society, at shops, supermarkets, call centers and so on. It is common in such systems to make trade-offs between the costs of operation (e.g., hiring personnel), and service as perceived by customers (e.g., mean waiting time).

To make such trade-offs one should be able to analyze the behavior of queueing systems and compute relevant key performance indicators. For this purpose we need models of queueing systems. The simple models can typically be solved with mathematical tools, but the analysis of difficult models requires simulation. Although simulation is very powerful, the development of useful simulators is based on profound insight into the behavior of queueing systems, in particular bottlenecks. Moreover, simulators require extensive testing, as (programming) mistakes are easily made. Hence, mathematical models are also key ingredients in the development process of a simulator, to provide insight and suitable testing grounds.

The aim of this course is to equip the students with a set of tools to analyze and improve queueing systems by means of simple mathematical models on the one hand, and simulation on the other. The students should realize that both approaches complement each other: the simulations to analyze realistic queueing systems, the mathematical models to provide intuition, insight and tests.

The course starts with providing a set of simple numerical models (recursions) to analyze and simulate realistic queueing systems. Then we develop mathematical models to analyze single server queueing systems and queueing networks, and show how simulation and analysis interact in the understanding of queueing systems and queueing networks.

3 Course Goals

See ocasys.

4 Reading material

In the course we use the `queueing_book.pdf` which is available at github. The exercises form an integral part of the text; they provide examples, derivations, and theoretical results. As such, the main text uses, explicitly and implicitly, material developed in the exercises. It is therefore mandatory to make and understand *all* exercises in the book. Exercises assigned per week/lecture relate directly to the reading material per week/lecture. In the lectures I assume that you solved the exercises of previous lectures; hence, it is important to keep up with doing exercises.

We will open a discussion board on nestor for questions.

You are free to use old exams for extra questions (although the book contains plenty). I will *not* answer questions about old exams; for that you have to consult your fellow students.

5 Lectures and exercises

The course consists of seven weeks with two on-line lectures per week. In the lectures we focus on the, perhaps, more difficult parts of the reading material. The rest of the reading material is for self study. I will not record the lectures as I will not discuss material that is not in the eueueing book or the assignments.

6 Assignments

There are a number of assignments which you have to complete with another student. The assignments consist of explaning simulation code and redoing a number of simulations. All information is available in the `assignments` directory at github. I made a number of youtube movies to explain how the simulations work.

The rules are like this:

1. You have to do each assignment with another student. You can sign up for a group once we opened the groups on nestor.

2. For each assignment you have to hand in a pdf file, which is typeset in LaTeX. It should include both your names, student ids, a title, and a date. You can find a template on github

3. Each assignment contains exercises. Many of these exercises ask you to explain how the simulation code works. The intention is to get you started with simulation (hence a bit of programming) and help you keep up with the course. Much of programming is 'monkey see, monkey do', reading and copying the code of others is more of a rule than an exception.

4. With respect to programming language, you can use my python code, but you are also allowed to build your simulation in R, or C++, or whatever other language you like. As long as you can *read* my python code, all is ok.

5. Assignment n is due at Wednesday 18h00 in week $n+1$ of the course.

Note specifically that the python code developed in the book and the simulation is part of the course. You should be able to understand the code and find mistakes if you are presented with modifications of the code. For instance, at the exam we can include a question like: "what is the value of `a` after the completion of this loop:"

```
a = 3
for i in range(3):
    a += 5
```

And then you have to provide the answer: "18".

7 Tutorials

During the tutorials you work as a group on the assignment for that week. Besides this, you can ask questions about the book or other material.

8 Entry Conditions

We will heavily use probability theory, calculus, linear algebra, and programming concepts.

9 Exam

The exam and the resit will be on campus. The regular exam duration is 3 hours. The exam resit) will consist of 2 parts.

The first part:

1. Takes 1 h
2. Is closed book
3. We ask you about (small variations) of the exercises of the queueing book; it will mostly focus on derivations.

The second part:

1. Takes 2h.
2. Is open book and consist of about 13 questions. You are allowed to use the book, the solutions, the material for the tutorial, the internet, anything, but NOT fellow students. You have to make the exam on your own.
3. All problems have the same weight.
4. Assumptions and data presented within a problem only apply to that problem. Definitions and symbols will not be explained in the exam; you can find them in the course book.
5. The problems ask you to provide the result of a computation. For instance, "Let $a = 4$ and $b = 7$. Compute $a + b$." Then you are supposed to provide the answer 11 in nestor.
6. The exam questions will be based on exercises of the book. However, in the exam you have to provide numerical answers. You have to use the computer to carry out the computations; unless you have unprecedented calculation skills. You are free to use whatever language suits you best. I like python, but if you prefer some other language, no problem.
7. Hence, you have to bring a laptop to the exam to make this part.

8. Ensure that you know how to copy numbers from a pdf file and paste them into your programming environment or excel for further processing. Try this as a test:

6.84	7.50	7.77	8.43	8.71	9.25	9.92
10.17	10.32	10.96	11.65	12.20	13.17	13.66
14.34	15.23	15.77	16.56	17.06	17.08	17.86
18.81	19.20	19.95	20.93	21.67	22.49	22.92
23.26	23.78	24.48	25.30	26.20	26.79	26.86

9. You should be comfortable with using the python code (or something similar in, e.g., R) of Section 3.4 of the book and the assignments.

10. You should be capable of programming simple recursions such as those of Section 7.1 and 7.2 of the book.

11. When you are asked to compute a standard deviation or variance, divide by n, not by $n-1$.

12. If you need to use Sakasegawa's formula, you should assume that the results are exact, rather than approximate.

13. You should provide your answer in the same unit of time as given in the problem.

14. Rouding to 3 significant digits is sufficient.

15. You will not be penalized for small deviations in precision from the expected answer. Specifically, suppose your answer is x and ours (the correct) is y. Whenever $x/y \in [0.95, 1.05]$ we accept your answer as correct.

16. The exam is personalized: you have your own set of questions (in a random sequence) from a pool of questions and you have to use the data as specified in your exam. Your exam will be provided via Nestor.

17. You are not allowed to distribute the exam until 1 hour after the closing time of the exam, and we rely on your common sense and honesty to comply with this rule. To help you resist the temptation to share your exam: the questions will be in random sequence, the question formulations will be different, e.g., "$a = 3, b = 4$, what is $a + b$?", "$a = 3, b = 4$, what is $a \cdot b$?", "What is $a \times b$ if $a = 3, b = 6$?", "What is the product of 7 and 8?". In fact, it is very easy to find many different ways to formulate the same type of question, or formulate questions that are seemingly the same, but differ in the details. So, if you plan to cheat, you will most surely waste a lot of time just figuring out what precise overlap you have with fellow cheaters. And if you don't get the details right, your answer will be wrong anyway.

18. After 1 hour after the closing time of the exam on Nestor, you are of course allowed to share and discuss your exam.

19. You provide your answers on Nestor in the directory 'Course Documents/Exam'. Answers are strictly numerical, so we expect no technical problems with this. As long as you have access to Nestor (via computer, mobile phone or tablet), you are safe.

10 Grading

We advise you to go the lectures and the assignments, but there is no obligation.

Assignments will be graded as a 1, 4, 7, 9, or 10. (For a 10 your work needs to be flawless.) If you don't turn in an assignment, the grade will default to 1. Let e be points for the exam or the resit. Then we compute your final grade g with the code:

```
from sigfig import round

tot = 25

def grade(a, e):
    a = round(sum(a) / len(a), sigfigs=2)
    e = round(10 * e / tot, sigfigs=2)
    if e < 5:
        g = max(e, 1)
    elif a >= 6:
        g = max(0.75 * e + 0.25 * a, e)
    else:
        g = 0.75 * e + 0.25 * a

    return round(g, sigfigs=1)


# some tests
print(grade(a=[10, 10, 10, 10, 10, 10, 10], e=5))
print(grade(a=[10, 10, 10, 10, 10, 10, 10], e=12))
print(grade(a=[10, 10, 10, 10, 10, 10, 10], e=13))
print(grade(a=[8, 8, 8, 8, 10, 1, 1], e=11))
print(grade(a=[8, 8, 8, 8, 10, 1, 2], e=12))
print(grade(a=[8, 8, 8, 8, 10, 1, 1], e=23))
print(grade(a=[1, 1, 1, 1, 10, 1, 10], e=25))
print(grade(a=[7, 7, 8, 8, 9, 10, 6], e=15))
print(grade(a=[7, 7, 7, 7, 7, 6, 4], e=25))
```

```
2.0
5.0
6.0
4.0
5.0
9.0
8.0
6.0
10.0
```

11 Contact Info

- dr. N. (Nicky) D. Van Foreest, Duisenberg 666, 050-363 51 78, n.d.van.foreest@rug.nl.
- dr. X. (Stuart) Zhu, 050 36 38960, x.zhu@rug.nl
- J. (Joost) Doornbos, j.doornbos@rug.nl