OKPESEYI SIMILOLUWA BLOSSOM

16/SCI01/033

CSC 410

QUESTIONS AND SOLUTIONS

1. Explain the concepts of operational laws as applied to computer and network system performance evaluation.

Operational laws are simple equations which may be used as an abstract representation or model of the average behaviour of almost any system. One of the advantages of the laws is that they are very general and make almost no assumptions about the behaviour of the random variables characterising the system1 . Another advantage of the laws is their simplicity: this means that they can be applied quickly and easily by almost anyone.

Based on a few simple observations of the system the performance analyst can, by applying these simple laws, derive more information. Using this information as input to further laws the analyst gradually builds up a more complete picture of the behaviour of the system. Note that although we will talk in this section about operational laws in the context of systems, the laws are equally applicable to the observations obtained from models and we will have occasion to use the laws in this way later in the course. The foundation of the operational laws are observable variables. These are values which we could derive from watching a system over a finite period of time. We assume that the system receives requests from its environment. Each request generates a job or customer within the system. When the job has been processed the system responds to the environment with the completion of the corresponding request. An Abstract System If we observed such an abstract system we might measure the following quantities: T, the length of time we observe the system; A, the number of request arrivals we observe; C, the number of request completions we observe; B, the total amount of time during which the system is busy (B ≤ T); N, the average number of jobs in the system. From these observed values we can derive the following four important quantities:

λ = A/T, the arrival rate;

X = C/T, the throughput or completion rate,

U = B/T, the utilisation;

S = B/C, the mean service time per completed job.

In contrast, Markovian analysis relies on very strong assumptions about the distribution function of the random variables which are used. Performance Modelling LN-2 We will assume that the system is job flow balanced. This means that the number of arrivals is equal to the number of completions during an observation period, i.e. A = C. Obviously this assumption is not true in all observation periods, but it is a testable assumption because an analyst can always test whether the assumption holds. Note that if the system is job flow balanced the arrival rate will be the same as the completion rate, that is, λ = X.

1. Exhaustively describe at least eight operational laws that are widely employed in computer system performance evaluation.
2. Little’s Law

This is the best known and most commonly used operational law is Little’s law. It is named after the man who published the first formal proof of the law in 1961, although it had been widely used before that time. Little’s law is usually phrased in terms of the jobs in a system and relates the average number of jobs in the system N to the residence time W, the average time they spend in the system. Let X be the throughput, as above. Then Little’s law states that N = XW The average number of jobs in a system is equal to the product of the throughput of the system and the average time spent in that system by a job. Given a computer system, Little’s law can be applied at many different levels: to a single resource, to a subsystem or to the system as a whole. A little care may be necessary if the law is applied in this way, as the definitions of the number of jobs, throughput and residence time used at the different levels must be compatible with each other. At different levels of detail, different definitions of “request” are appropriate. For example, when considering a disk, it is natural to define a request to be a disk access, and to measure throughput and residence time on this basis. When considering an entire transaction processing system, on the other hand, it is natural to define a request to be a user-level transaction, and to measure throughput and residence time on this basis. Each such transaction may generate several disk accesses. We will return to this idea of systems, or subsystems, within a system in the following subsection.

1. Forced Flow Law

It is often natural to regard a system as being made up of a number of devices or resources. Each of these resources may be treated as a system in its own right as far as the operational laws are concerned, with the rest of the system forming the environment of that resource. A request from the environment generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource. 10 Performance Modelling LN-2 Suppose that during an observation interval we count not only completions external to the system, but also the number of completions at each resource within the system. We define the visit count, Vi , of the ith resource to be the ratio of the number of completions at that resource to the number of system completions Vi ≡ Ci/C. More intuitively, we might think of this as the average number of visits that a system-level job makes to that resource. For example, if, during an observation interval, we measure 10 system completions and 150 completions at a specific disk, then on the average each system-level request requires 15 disk operations. The forced flow law captures the relationship between the different components within a system. It states that the throughputs or flows, in all parts of a system must be proportional to one another. In other words, it relates the throughput at the individual resources (Xi = Ci/T) to the throughput at the complete system (X = C/T). It is stated as follows Xi = XVi The throughput at the ith resource is equal to the product of the throughput of the system and the visit count at that resource. An informal interpretation of this law is that, since the visit count defines the number of visits to a resource or device that each job needs in order to complete its processing, the resource must keep up a correspondingly scaled completion rate to ensure that the system completion rate is maintained.

1. Utilisation Law

If we know the amount of processing that each job requires at a resource then we can calculate the utilisation of the resource. Let us assume that each time a job visits the ith resource the amount of processing, or service, time it requires is Si . Note that service time is not necessarily the same as the residence time of the job at that resource: in general a job might have to wait for some time before processing begins. The total amount of service that a system job generates at the ith resource is called the service demand, Di : Di = SiVi The utilisation of a resource, the percentage of time that the ith resource is in use processing to a job, is denoted Ui . The utilisation law states that 11 Performance Modelling LN-2 Ui = XiSi = XDi The utilisation of a resource is equal to the product of the throughput of that resource and the average service requirement at that resource.

1. General Residence Time Law

One method of computing the mean residence or response time per job in a system is to apply Little’s law to the system as a whole. However, if the mean number of jobs in the system, N, or the system level throughput, X, are not known an alternative method can be used. Applying Little’s law to the ith resource we see that Ni = XiWi , where Ni is the mean number of jobs at the resource and Wi is the average response time of the resource. From the forced flow law we know that Xi = XVi . Thus we can deduce that Ni/X = ViWi . The total number jobs in the system is clearly the sum of the number of jobs at each resource, i.e. N = N1 + · · · + NM if there are M resources in the system. We know from Little’s law that W = N/X and from this we arrive at the general residence time, or general response time law: W = X M i=1 WiVi The average residence time of a job in the system will be the sum of the product of its average residence time at each resource and the number of visits it makes to that resource.

1. Interactive Response Time Law

The name of this law dates back to the time when most of the systems which were being modelled were mainframes processing both interactive jobs and batch jobs. The think time, Z, was quite literally the length of time that a programmer spent thinking at his terminal before submitting another job. More generally interactive systems are those in which jobs spend time in the system not engaged in processing, or waiting for processing: this may be because of interaction with a human user, or may be for some other reason. For example, if we are studying a cluster of PCs with a central file server to investigate the load on the file server, the think time might represent the average time that each PC spends processing locally without access to the file server. At the end of this nonprocessing period the job generates a fresh request. The key feature of such a system is that the residence time can no longer be taken as a true reflection of the response time of the system. The think time represents the time between processing being completed and the job becoming available as a request again. Thus the residence time of the job, as calculated by Little’s law as the time from arrival to completion, is greater than the system’s response time. The interactive response time law reflects this: it calculates the response time, R as follows: R = N/X − Z The response time in an interactive system is the residence time minus the think time. Note that if the think time is zero, Z = 0 and R = W, then the interactive response time law simply becomes Little’s law.

1. Bottleneck analysis

The resource within a system which has the greatest service demand is known as the bottleneck resource or bottleneck device, and its service demand is maxi{Di}, denoted Dmax. The bottleneck resource is important because it limits the possible performance of the system. This will be the resource which has the highest utilisation in the system. The residence time of a job within a system will always be at least as large as the total amount of processing that each job requires—this will be the time that the job takes even if it never has to wait for a resource. The total amount of processing that a job requires is D, the total service demand,

D = PM i=1 Di .

In general, there will be some contention in the system meaning that jobs have to wait for processing so the residence time will be larger than this, i.e. W ≥ D The throughput of a system will always be limited by the throughput at the slowest resource (think of the forced flow law); this is the bottleneck device. By the utilisation 13 Performance Modelling LN-2 law, at this resource, let’s call it b, Ub = XDmax. Therefore, since Ub ≤ 1 X ≤ 1/Dmax It follows that if we wish to improve throughput we should first concentrate on this resource—improving throughput at other resources in the system might have little effect on the overall performance. Using Little’s law or the interactive response time law, we can derive a tighter bound on the response time which applies when the system is heavily loaded (i.e. the mean number of jobs, N, is high). Applying the interactive response time law to the throughput bound,

X ≤ 1/Dmax   
we obtain: R = N/X − Z ≥ NDmax − Z

Applying Little’s law we obtain W ≥ NDmax.

Thus the asymptotic bound for residence time or response time is: W ≥ max{D, NDmax} R ≥ max{D, NDmax − Z} Similarly the bound on the throughput of an interactive system may be made tighter when the system is lightly loaded (i.e. the mean number of jobs, N, is small). From the interactive response time law:

X = N/(R + Z) ≤ N/(D + Z)

Applying Little’s law (when Z = 0) we obtain

X ≤ N/D. X ≤ min{1/Dmax, N/(D + Z)}

Notice that the bottleneck depends on both resource parameters (Xi or Si) and the workload parameters (Vi). If we change the number of visits that each job makes to a resource we might move the bottleneck.

1. The Flow Balance Assumption

Frequently it will be convenient to assume that systems satisfy the flow balance property, namely, that the number of arrivals equals the number of completions, and thus the arrival rate equals the throughput: The Flow Balance Assumption: A = C, therefore A = X The flow balance assumption can be tested over any measurement interval, and it can be strictly satisfied by careful choice of measurement inter: val. When used in conjunction with the flow balance assumption, Little’s law and the forced flow law allow us to calculate device utilizations for systems whose workload intensities are described in terms of an arrival rate. In Figure 3.5 we show a queueing network model similar to that used to represent the VAX-11/780 in the case study described in Section

1. Response Time Law

The Little’s law tells us that the system throughput must be

10 - = 0.5 interactions/second.

If we 15+5 denote think time by 2 then we can write this incarnation of Little’s law as N = X(R + Z>. As with the utilization law, this application is so ubiquitous that we give it its own name and notation, expressing R in terms of the other quantities:

The Response Time Law: R = $ - Z

As an example application of the response time law, suppose that a system has 64 interactive users, that the average think time is 30 seconds, and that system throughput is 2 interactions/second. Then the response 64 time law tells us that response time must be - - 30 = 2 seconds

Suppose we were to construct a queueing network model of the system in the previous example. The number of users (64) and the average think time (30 seconds) would be parameters of the model, along with the service demands at the various resources in the system. Throughput and response time would be outputs of the model. Suppose that the model projected a throughput of 1.9 interactions/second, an error of just 5%. Since the response time law must be satisfied by the queueing network model, a compensating error in projected response time must result: R = 64-30 1.9 Thus the model must project a response time of 3.7 seconds, an error of 85%

1. Distinguish between the Forced Flow Law and the Residence Time Law from a systems perspective (not by definition).

The forced flow time law; It is often natural to regard a system as being made up of a number of devices or resources. Each of these resources may be treated as a system in its own right as far as the operational laws are concerned, with the rest of the system forming the environment of that resource. A request from the environment generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource. The forced flow law, which states that the flows (throughputs) in all parts of a system must be proportional to one another. Suppose that during an observation interval we count not only system completions, but also the number of completions at each resource. We define the visit count of a resource to be the ratio of the number of completions at that resource to the number of system completions, or, more intuitively, to be the average number of visits that a system-level request makes to that resource. If we let a variable with the subscript k refer to the k-th resource (a variable with no subscript continues to refer to the system as a whole), then we can write this definition as: V,, the visit count of resource k: ck V,

S 7 If during an observation interval we measure 10 system completions and 150 completions at a specific disk, then on the average each system-level request requires 150/10 = 15 disk operations. If we rewrite this definition as ck = vk c and recall that the completion count divided by the length of the observation interval is defined to be the throughput, then the throughput of resource k is given by:

The Forced Flow Law: Xk = V,X .

And while the response time law on the other hand;

One method of computing the mean residence or response time per job in a system is to apply Little’s law to the system as a whole. However, if the mean number of jobs in the system, N, or the system level throughput, X, are not known an alternative method can be used. Applying Little’s law to the ith resource we see that

Ni = XiWi ,

where Ni is the mean number of jobs at the resource and Wi is the average response time of the resource. From the forced flow law we know that

Xi = XVi . Thus we can deduce that,

Ni/X = ViWi

The total number jobs in the system is clearly the sum of the number of jobs at each resource, i.e. N = N1 + · · · + NM if there are M resources in the system. We know from Little’s law that W = N/X and from this we arrive at the general residence time, or general response time law:

W = X M i=1 WiVi

4. Discuss some basic queuing models and basic queuing disciplines.

I. Different models in queuing theory are classified by using special (or standard) notations described initially by D.G.Kendall in 1953 in the form (a/b/c). Later A.M.Lee in 1966 added the symbols d and c to the Kendall notation. Now in the literature of queuing theory the standard format used to describe the main characteristics of parallel queues is as follows:

                   {(a/b/c) : (d/c)}

Where

a = arrivals distribution

b = service time (or departures) distribution

c = number of service channels (servers)

d = max. number of customers allowed in the system (in queue plus in service)

e = queue (or service) discipline.

Certain descriptive notations are used for the arrival and service time distribution (i.e. to replace notation a and b) as following:

M = exponential (or markovian) inter-arrival times or service-time distribution (or    equivalently poisson or markovian arrivel or departure distribution)

D = constant or deterministic inter-arrival-time or service-time.

G = service time (departures) distribution of general type, i.e. no assumption is made about the type of distribution.

GI = Inter-arrival time (arrivals) having a general probability distribution such as as normal, uniform or any empirical distribution.

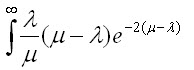
Ek = Erlang-k distribution of inter-arrival or service time distribution with parameter k (i.e. if k= 1, Erlang is equivalent to exponential and if k = , Erlang is equivalent to deterministic).

For example, a queuing system in which the number of arrivals is described by a Poisson probability distribution, the service time is described by an exponential distribution, and there is a single server, would be designed by M/M/I.

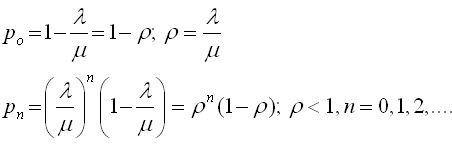
The Kendall notation now will be used to define the class to which a queuing model belongs. The usefulness of a model for a particular situation is limited by its assumptions.

**Model 1 :{( M/M1): (/FCFS)} single server, unlimited queue model**

The derivation of this model is based on certain assumptions about the queuing system:

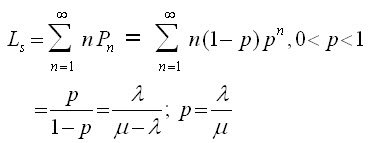
1. Exponential distribution of inter-arrival times or poisson distribution of arrival rate.
2. Single waiting line with no restriction on length of queue (i.e. infinite capacity) and no banking or reneging.
3. Queue discipline is ‘first-come, first-serve
4. Single serve with exponential distribution of service time

**Performance characteristics**

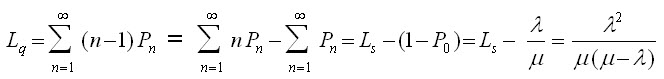


       Pw   = probability of server being busy (i.e. customer has to wait) =  1-Ρo= λ / µ

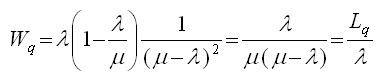
1. Expected (or average) number of customer in the system (customers in the line plus the customer being served)



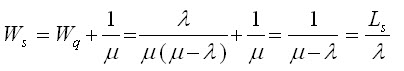
1. Expected (or average) queue length or expected number of customers waiting in the queue



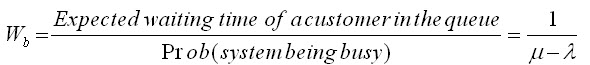
1. Expected (or average) waiting time of a customer in the queue



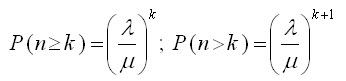
1. Expected (or average) waiting time of a customer in the system (waiting and service)



1. Expected (or average) waiting time in the queue for busy system



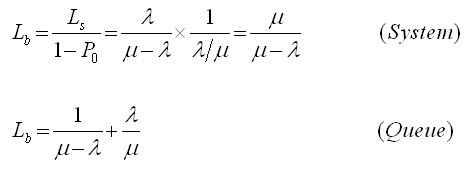
1. Probability of k or more customers in the system



7. The variance (fluctuation) of queue length

 Var (n) = 161239

8. Expected non-empty queue length



9. Probability that waiting time is more than

1612315.jpg

1.Average number of patients in the queue

1612316.jpg

2.Fraction of the time for which there no patients 1612317.jpg

3.When the average queue size is decreased from 4/3 patient, the new service rate is determined as

1612318.jpg  patients per minute.

Average rate of treatment required is: 1612319.jpg= 7.5 minutes, i.e.  a decrease in the average rate of treatment is 2.5(= 10 – 7.5) minutes.

Budget per patient = Rs (100 + 2.5 x 10) = Rs 125 per patient.

1. Queuing Disciplines

* Class Based Queue (CBQ): is a queuing discipline for the network scheduler that allows traffic to share bandwidth equally, after being grouped by classes. The classes can be based upon a variety of parameters, such as priority, interface, or originating program.
* Token Bucket Flow (TBF):  is an algorithm used in packet switched computer networks and telecommunications networks. It can be used to check that data transmissions, in the form of packets, conform to defined limits on bandwidth and burstiness (a measure of the unevenness or variations in the traffic flow).
* First In First Out (FIFO): is an accounting method in which assets purchased or acquired first are disposed of first. FIFO assumes that the remaining inventory consists of items purchased last. An alternative to FIFO, LIFO is an accounting method in which assets purchased or acquired last are disposed of first.
* Stochastic Fair Queuing (SFQ): is a family of [scheduling algorithms](https://en.wikipedia.org/wiki/Scheduling_algorithm) used in some [process](https://en.wikipedia.org/wiki/Scheduling_(computing)) and [network schedulers](https://en.wikipedia.org/wiki/Network_scheduler). The algorithm is designed to achieve [fairness](https://en.wikipedia.org/wiki/Fairness_measure) when a limited resource is shared, for example to prevent flows with large packets or processes that generate small jobs from consuming more throughput or CPU time than other flows or processes. Fair queuing is implemented in some advanced [network switches](https://en.wikipedia.org/wiki/Network_switch) and [routers](https://en.wikipedia.org/wiki/Router_(computing)).
* Asynchronous Transfer Mode (ATM):  is a switching technique used by telecommunication networks that uses asynchronous time-division multiplexing to encode data into small, fixed-sized cells. This is different from Ethernet or internet, which use variable packet sizes for data or frames.
  1. Discuss how to resolve some basic queuing problems.

1. Reduce Wait Time by Going Transparent

We’re going on a limb here: You’re probably running a business and not an Area 51 operation. So why all the secrecy?

Put yourself in your customer’s shoes. In a situation where you already have little control, it can be difficult to deal with lack of information.

How long is each customer going to take? Why has the queue not moved in so long? Am I stuck here?

If you don’t equip your staff with the knowledge to answer these questions for your customers, you’re in for a big eye-opener — in the form of hundreds of customers walking out through your door.

The problem, however, is that **businesses often don’t have a way of accurately answering these questions**. When managers are manually handling queues, there’s only a limited amount of predictability to the flow of your queues.

That is a very different situation compared to an [automated retail queue management system](https://www.qminder.com/queuing-solution/).

With a QMS, it’s not possible to be ambiguous about things like the length of the queue. The interface of Qminder is specifically designed to provide clear and detailed information.

[Setting up Qminder at your location](https://www.qminder.com/how-to-set-up-Qminder-at-your-location/) is extremely easy. A wireless system connects the check-in counter (an iPad or a tablet), the store backend, and a big TV display which updates information in real time.

All the information your customers need, right where they can see it.

2. Reducing Queues the Disney Way: Give Customers Something to Do

The biggest source of frustration when standing in a queue is inactivity. From the perspective of a typical queue-stander, they’re simply wasting their time.

And the longer the wait time, the stronger this feeling.

For some insight on how to deal with queues, let’s take a look at Disney. Disney is certainly no stranger to long queues, as [Disney amusement parks deal with tens of millions visitors every year](http://disneynews.us/disney-parks-attendance/).

People are there to have a good time, and **waiting isn’t what “good time” entails**. That’s why Disney has designed its queues to be not the tedious pre-fun part of the amusement park, but an integral aspect of entertainment.

A typical queue in the Disney park has cameras and large interactive screens that allow visitors to see themselves and play games. This simple trick makes us feel as though we’re already being entertained — and, in a way, we are.

Another clever trick used by Disney is for people waiting to see [Monster’s Inc. Laugh Floor](http://www.disneylists.com/2016/06/8-facts-secrets-monsters-inc-laugh-floor-disneys-magic-kingdom-park/), a light-hearted comedy show.

When waiting to enter the theater, audience members are asked to text their favorite jokes. Some of these jokes end up featured in the actual show, and the submitter even gets a mention in the credits.

This level of interaction makes for an unforgettable experience that easily trumps long waiting.

Obviously, not every business has the luxury of providing amusement park-level of entertainment. But despite all that, the lesson is simple — **boredom is the biggest mood-killer**.

Again, put yourself in your customer’s shoes. Would you want to wait for minutes, if not hours, without anything to occupy your addled, bored-out-your-skull mind?

For better or for worse, we’re seeing the birth of the ADHD generation. When all people care about is distraction, anything goes.

Customer is always right? Try “Customer is always entertained!”

3. A Fair Queuing System Keeps Everyone Happy

One of the most important characteristics of any queue management method is the queuing discipline used. Simply put, the queuing discipline is the rule used to decide who goes next in a queue.

Two of the most commonly used rules are:

First in, First out.

Last in, First out.

Bottom line, **people expect queues to be fair**. It’s not like they’re happy to be stuck waiting in line, to begin with. But when everyone abides by the same rules, we can’t help but follow them too.

Don’t believe me? Let’s look at Exhibit A.

[A group of sociologists helmed by Stanley Milgram](https://www.theguardian.com/lifeandstyle/2008/nov/08/healthandwellbeing) took a ticketing counter as their base of experiments, planting several stooges to cut in line. Despite visible irritation from the other people, the protests were mild to non-existent.

However, with every additional line-cutter, the dissent grew ever louder. This means that **people are willing to ignore outliers as long as they remain the exception**.

Once people start routinely bypass the queue, all hell breaks loose.

The problem of line-cutting and queue-jumping is a major one for unmanaged queues, but not so much for queues controlled by automated QMS. There, the possibility of someone jumping in ahead simply doesn’t exist.

Once you [adopt a queue management system for your retail](https://www.qminder.com/everything-retail-business-queue-management-system/), the queuing discipline is out there for everyone to see. (Remember what we told you about transparency?)

Social fairness is a major component of peaceful queuing experience. All queue-standers were created equal, and it’s your job that it stays the same.

4. Reduce Waiting Time by Making Queuing Experience Enjoyable

Unless your customers are, for whatever reason, inmates or cattle, **you need to reconsider the use of token numbers**. Paper-based queue systems were all the rage back when we’ve just figured out a more civilized way to queue up.

But now? It’s nothing short of CX embarrassment.

If you want the personality of your business to stand out, you have to make it personal.

[People respond really well to seeing strangers remember their name](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1647299/) — even better when they refer to them using said names! It is a simple gesture that goes a long way in making your business appear people-friendly.

This simple scientific fact is the backbone of Qminder’s name-based queuing solution. All you need is a tablet and a customer eager enough to join a queue.

Once they put down their name, the system remembers them and continues to refer to them by this name until the end of the service session.

Best thing about this system? Since you let your customers check themselves into the queue, **you give them a sense of agency and control**.

Also, because there is clear information available for everybody, physical queues become obsolete. Customers are given an accurate estimate of their wait time, so they’re free to wander about and busy themselves with other things.

Thus, we take the worst part of queues out of the equation — that nagging feeling that you’re wasting your precious time.

[Improving customer service with a QMS](https://www.qminder.com/improve-customer-service-through-queue-management-system/) is a surefire way to success as a business. Once you transform the worst part of the customer journey — queues, waiting, idleness — your first-time visitors will quickly turn into long-time customers.

6. You have been presented with some systems performance evaluation report; the first was done using measurement technique only, the second was done with simulation technique only, the third was done using analytical technique only, then the fourth was done using measurement and simulation technique only, while the fifth was done using simulation and analytical technique only and the sixth was done using measurement and analytical technique only. You are the director of information technology infrastructure in your firm and your firm is about to acquire the information infrastructure concerned in this report

i. What specific motive will you consider imperative in general?

Don’t trust the result of a simulation model until they have been validated by other performance evaluation techniques

ii. Why do you consider this metric important?

It’s important so as to know if the system is correctly defined and if the goals are clearly stated.

iii. If they are not in this report, what will you do in such a report?

Such report is invalid

iv. If they are part of the report, what is the first action to take with that report and why?

Go through the report to check if there are any mistakes

v. Which reports or combination of report will you adopt and why?

The forth report (measurement and simulation) because measurement technique requires more effort and it is the most accurate; simulation method requires relatively lesser effort and it is averagely accurate while analytical method is very quick and often very less accurate.