CS 482

Aiden Sirotkine

Spring 2025

Contents

Chapter 1

CS 482

This'll be an interesting class lets hope I can figure out stats.

- building complex models
- algorithmic randomness
- statistically analyzing data

Chuck data into a black box of modelling code and get result data out.

Simulations give the most realistic answers for complex systems that cannot be linearly solved.

Simulation is only good for basically unsolvable problems.

Chapter 2

Terminology

2.1 System

A collection of *entities* that interact with a common purpose according to sets of *laws* and *policies*

example: a store, airport terminal, etc.   most things

2.2 Entities

The components/objects that define a system

Physical or Logical objects

Temporary or Permanent

2.3 Attributes

The traits that define an entity

static or dynamic

qualitative or quantitative
A cashier in a store :

- has a high school diploma (static, qualitative)
- has an IQ of 104 (static, quantitative)
- can be busy or idle (dynamic, qualitative)
- can process customers at a rate from 6/hour to 20/hour (dynamic, quantitative)

2.4 Entity-Attribute Hierarchy

determines the level of detail in the simulation

- Regional Plants
- Production Lines
- Work Areas
- Machines, Tools, Operators

The *level of detail* in the simulation model is set by the entity/attribute hierarchy boundary, which is determined by the *objectives of the study.*

Attributes become entities if you want more detail in a system

The boundary between attribute/entity determines the level of detail in the simulation.

2.5 Laws and Policies

Both Laws and Policies govern the behavior of the system, but Laws cannot be changed while Policies can be changed.

Laws are followed, Policies are set.

2.6 Model

The thing that we're trying to do in this class

A simplification of a system

there are many ways to model a system

1. Events List
2. Difference Equations
3. Markov Chains

2.6.1 State Space

A Collection of variables that represent and measure the condition of the system (busy, idle, broken, etc.)

The state space is the *film* for a photo snapshot of a system.

2.7 Event

An instant of time when one of the following occurs

- the state(s) change(s)

- other events are caused (scheduled) or prevented (cancelled) (i.e. other states change)
- data is collecte dand statistics may be compiled (based on or uses states)

examples:

- a part arrives
- A machine starts or stops or breaks down
- The end of day of operation

2.8 Process

An indexed set of states of events

Let N_t be hte number of customers in the system at time t

Let B_j be a $0-1$ indicator of whether or not that jth customer does or doesn't get on hold.

2.9 Discrete-Event Simulation

A model where the state S changes at *discrete* points in time

what/when/how/what impact of changes

2.10 Single Server Queue

λ is the rate at which parts arrive

μ is the rate at wich the server can process parts
Number in system = number in queue + number in service.
events schedule other events.

2.11 Building Simulation Models

- define states
- identify when and how the state change
- define events
- define initialization
- ALL simulation models have an initialization event.
- assume a random number generator is available
- a new random variable value must be generated each time a random variable is used or called
- arrival events schedule more arrival events (self-generating)

2.11.1 When executing an event:

- State changes occur first
- Events scheduled or cancelled occur second

Once you have all of the states and changes and whatever, coding the simulation itself is not actually that hard.

hop from most recent event to most recent event until done, changing the event queue as needed.

2.11.2 Dynamic Simulation

- Events are scheduled and executed in time sequence
- States are changed as each event is executed
- Data can be colelcted as events to do stats and math stuff
- Know where to collect data and what type of data to collect.

2.12 Data Tracking

- If we want to determine the customer waiting times in the systme, we need to record both the arrival time ARRTIME and the departure time DEPTIME. We can then calculate WAITINSYS = DEPTIME - ARRTIME
- What we do is at an event, we also record data,
- So for example, when a customer arrives, we record the time in ARRTIME at the point of that event.
- You need to make sure that you have indexes for each arrival and exit so that you can correlate waiting times to entities.

2.13 Simulataneous Events

- It doesn't happen super often but it does happen and we still have to do things sequentially

- For example, if we want to track the number of times the server becomes idle then if the server completes a task as a new customer arrives, we don't want the server to be open for 0 seconds, so the simultaneity of the task actually makes a difference in things.
- You can also randomize what events happen first and that might fix whatever issues might occur.
- Just be aware that it is an issue

2.14 Static Simulation Models

- Easier than dynamic models
- time is not really a factor
- Event lists are not needed (though can still be used)
- Events are naturally sequential

2.15 Dice Game

- Is it better to roll 1 dice and try to get 5 twice, or roll 2 die and try to get 7, 11, or 12 three times in a row
- can be done mathematically, but is trivial for a simulation
- demonstration of how simulations can be useful over analytical mathing.

2.16 Buffer Allocation

- Consider a production line with 3 machines that flow into each other and process a certain amount of a parts per second
- How large capacity of a buffer in between each machine do we need in order for machines to stop as little as possible?
- Machines can be busy, idle, or blocked (meaning the buffer in front of it is full, so it cant make more stuff without overflowing)

2.16.1 Policies

- Parts are indistinguishable
- parts are first come first served
- Parts begin processing at a machine only if there is space availbale in the buffer
- the buffer at the very end is infinity, so the last machine is on forever.

2.16.2 Events

- Arrival at Buffer
- Exiting buffer

- Arrival at Machine
- Exiting machine
- End of day

2.16.3 Priority

When making simultaneous events, we have to figure out what happens first.

Does the machine pick up an item from the buffer first? Does the buffer decriment first? Does the previous machine try to add an item to the buffer first?

2.17 Verification and Validation

- Does the simulation represent the real system?
- Is the code correct

2.17.1 Techniques

- Study the code carefully
- Make a trace through the simulation model to ensure that even adn state changes are realistic
- Look at the simulation output. Doe sit agree with known anlytic (theoretical) results? Does it work under certain restrictions?

- Fault/failure insertion testing: Put in bad data and see if it breaks correctly
- Set inputs at extreme values and see if the obvious happens
- Use historical data with known outputs and compare what the simulation yields.

Model validation tells us roughly how close our simulation is to the real system.

2.18 The Modeling Process

With a real system you can create a conceptual model and then that gets turned into a computer model that returns a bunch of data.

2.18.1 Detail

If we add too much detail, we lose the big picture and our simulation because obnoxious.

If we don't have enough detail, we lose accuracy.

Keep It Simple, Stupid

If we're trying to make a restaurant, the number of customers depends on the time of day.

The servers might or might not be distinguishable from each other

What are we measuring from the simulation?

What foods are avaialable in the simulation?

are there different table sizes?

Start with the minimum amount of detail, and then go up from there.

Use the minimal amount of detail to answer your questions.

2.19 Queuing Models

Seen in manufacturing, service, all sorts of stuff.

2.19.1 M/M/1 Model

Markovian / Markovian / 1 server (single service queue)

M/M queues are well studied and we have a good idea of how they work.

- M/M/1
- M/M/c
- M/M/c Feedback
- M/M/1 Priority
- M/M/1/K

2.19.2 Service Mechanims

- First in, First out
- Last in, First out
- random

2.19.3 Potential Measurements

- average wait time per customer
- fraction of customers who wait more than 10 minutes
- average idle time per worker
- average number of idle workers

2.19.4 G/G/1

Let A_i be the inter-arrival time betwee ncustomers i and $i+1$

Let S_i be the servie time of customer i

Then D_i is the delay time (waiting time in queue) of customer i

This type of queue ends up being governed by Lindsley's Equation

$$D_i = max\{0, D_i + S_i - A_i\} \qquad D_1 = 0$$

Only holds true for single server queuing system.

Chapter 3

Simulation (Event) Graphs

Graphical represenatation of entities, attributes, and state changes.

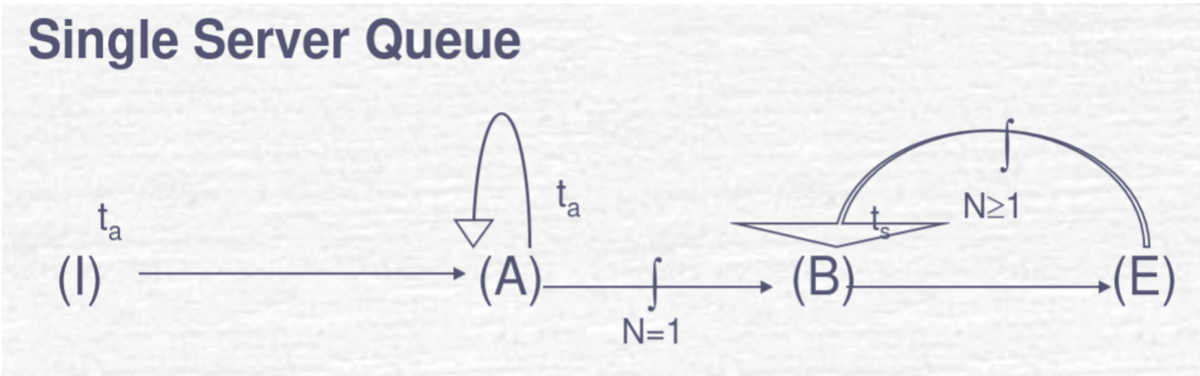
Provides a universal representation for discrete-event simulation models

IT'S NOT A FLOW CHART

State changes occur at each node.

- Scheduling edge
- Cancelling edge
- Self-scheduling event
  Event A is scheduled every event A
- Simultaneous Sequential Event

3.1 Single Server Queue



Where t_a is the time between arrivals and t_s is the service time.

It shows what is scheduled when and in what sequence.

$\int$ means if a condition is met.

The system is initialized, a things goes in the queue, if there is 1 server in the queue, begin service event, end service event, if there is someone in the queue, being servie event again.

you can compress the graph a little more.

Every simulation's event graph can be made with 1 initialization event and 1 node

sometimes simplifying a graph is very stupid because it gets rid of too much detail.

3.2 Edge Reduction

An event node can be removed if there are no condition exit edges and oen of the following

- zero delay times entering the edge

- zero delay times exiting the edge
- State changes are not associated with any edge conditions

if i) holds, then there can be condition exit edges

reduced graphs are NOT isomorphic. The same system is being modeled but with a different level of detail.

3.3 Variations of a Single Server Queue

Consider multiple indistinguishable servers (bowling alley)

Number of idle servers is a state variable S (number of idle servers)

Arrival is normal, begin service event is dependent on if $S > 0$,

end service event increments S.

Event B (begin service) needs to have higher priority than events E and A to avoid "phantom" customers/servers.

3.3.1 Phanton Customer

Supposed A and E begin at the same time.

If E is executed first, then event B is scheduled

If A is scheduled second, another B is scheduled because that server is still considered as "idle"

if we execute A first, we increment Q by 1 and then schedule B

if E is executed second, we increment S and because $Q > 0$, we schedule another B event for the same A.

This is why whenever B is scheduled, B needs to have priority.

3.3.2 non-empty initial conditions

Let $Q = q$ the initial number of people in the system.

consider a number Q' Which is all the intitial people.

We need to add a new event P if there are people in the queue initially

P puts the initial people in the queue 1 by 1 until all the servers are full or there are no more people.

P needs to have higher priority than B because we want everyone to be set to a server before any begin service queues begin.

Consider N the number of people in the system.

3.4 Batched Service/Arrivals

β is the batch of things in service and α is the number of customers that arrived.

The server cannot serve an amount of customers under β

make sure that B has a higher priority that E and A and then you'll be set.

It's pretty close to the same as a regular queue.

However, a busy period only begins as long as

$\beta \leq N \leq \beta + \alpha$, So the number in the system cannot be too too large or else we are already in a busy period.

However, we can use a variable L to determine whether or not we are in a busy period.

if $N \geq \beta$ and $L = 1$ then the B event begins, and if E occurs and $N < \beta$, then L goes to 1 and it will wait for a large enough batch of customers.

3.5 Rework (Feedback)

After E, if a certain event (rand(0, 1)) happens, then a REWORK event happens which queues another E event after a certain amount of time.

Because there's no begin service event, these simulations are clunky

3.5.1 Discard Rework

There's a certain probability that a part gets thrown out after E.

Nothing happens after the discard.

3.6 Limited Buffer Space

page 127

Consider an ENTER event.

N is number in the system. M is the number of customers that arrived. J is the number of customers that enter the queue.

$M - J$ is the number of people that get turned away

Set event E to have a higher priority than event A so that the customers go through the system correctly and no one gets incorrectly turned away.

3.6.1 Indicator Functions

You can use them instead of using the ENTER event.

let K be our buffer space

if $N > K$, then we turn em away

if $N \leq K$, then we add them to the queue.

3.7 0 Buffer Space + Retrial Orbiting

cars circling a parking lot.

If a customer arrives, but all servers are busy, then that same customers arrives again, but after a certain amount of time.

repeat until the customer finally gets in.

BEGIN should have the highest priority.

END should have a higher priority than ARR and RETRY

ARR should have a higher priority than RETRY

a BUSY event exists to count the amount of customers that do not get served on their first try.

3.8 Kit Assembly System

We have 2 separate assembly events that have to become combined for a third event to occur.

Let $A1$ and $A2$ both lead to the KIT event that then schedules B which schedules E

$A1$ schedules KIT if $Q1 = r \,\&\&\, Q2 \geq p$

$A2$ schedules KIT if $Q2 = p \,\&\&\, Q1 \geq r$

$E > B > KIT$ to avoid phantom things.

3.9 2 Different Prioritized Servers

after than A event, the item will go to $B1$ first, but if that's full, we go to $B2$

$S = 0$ means busy, $S = 1$ means idle.

need to make sure $E1$ and $B1$ have higher piority than $B2$ and $E2$

3.10 2 Different Randomized Servers

If both servers are idle, one will be picked at random.

Otherwise, we pick whatever server is available.

Randomize the priority events $E1$ and $E2$

3.11 Closing Time

After a closing event, all arrival events afterwards become null make sure that A has higher priority than $CLOSE$ because if the customer comes directly at closing time, they should still be treated.

Use a cancelling edge on any scheduled arrival events.

3.11.1 Closing Time (Empty the Queue)

set Q to 0

3.11.2 Closing Time (Empty the Queue, Empty the Server)

set Q to 0 and the servers to busy

3.11.3 Queue with Breakdowns

Certain random chance that the whole server can fail at the initialization stage.

If the server breaks, all end events are cancelled (or nothing happens if the server is idle (You can save S before the system breaks and then check that to determine if there are end service events or not))

REPAIR needs to have a higher priority than A

3.11.4 Breakdown with Fresh Parts

After the REPAIR event, if the server was busy at the time of breaking, put the WIP part in the queue again instead of throwing it away.

3.11.5 Breakdown with Partially Processed Part

If the server broke while it was busy, make a calculation to track how much time was left to process the part, and then make that the new time to the next end service event AFTER the server is done REPAIR-ing

3.11.6 Priority Queue (Non-preemptive)

Non-preemptive means if a low-priority is already in service, they will not be kicked out if a high-priority customer arrives.

3.11.7 Priority Queue (Preemptive)

If a high priority enters the queue, we cancel any low priority end service event and we place that low priority person back in the queue. The high priority person immediately gets served if there not are high priority customers in front of him.

The low priority customer can either be placed in the back of the queue or thrown out.

Real simulations have a mix of a bunch of these different details will be together in one system.

3.12 Inventory Systems

3.12.1 (s, S)

You start with a certain amount of goods S that gets continually depleted until it reaches s, in which it replenishes back to S.

3.12.2 (s, S) Reorder Delay

It takes a certain amount of time for the inventory to replenish.

Add a state order function C that is 1 if there is an order in process and 0 if not.

3.12.3 (s, S) Reorder Delay, No Order Event

you can make the arrive event dependent on the demand event depending on it $C = 1$ or not.

It just makes the graph smaller.

Realisticaly, interarrival times and order times are going to be random.

3.12.4 Inventory Management

Single units arrive with a certain interarrival time.

Orders arrive with a different interarrival time.

Each order requests β units.

If inventory goes to 0, you make an external order.

Make sure that input always has a higher priority than demand.

3.12.5 Perishable Item

Orders arrive with a time stamp. Units have a shelf life of R days, after which it is discarded.

You have to index every item and their arrival time, and then you can parse through the inventory and remove items if their indexed arrival time says that they're expired.

Now the whole inventory is indexed and sorted via arrival time.

Add a new DemandCheck self-scheduling event (demand is no longer self-scheduling). Check what's expired, remove the expired inventory,

After that, do the regular demand (because you know that none of your inventory is expired).

Because of how the indexing is done, the demand event always grabs the oldest bananas

3.12.6 Final Points

- Make sure that all nodes in an event graph can be reached.
- If a node can't be reached, schedule an event to reach it
- Event scheduling priorities are important

3.13 Dice Game Thing

1 2 3 4 5 6

1×1

2 3 4 5 6

5×6

2 3 4 5

6×5

2 3 4

5×4

2 3

2×2

3×3

$1 + 30 + 30 + 20 + 4 + 6$

1 2 3 4 5 6

1×1

2 3 4 5 6

6×6

6×5

5×4

1×2

6×3

$1 + 36 + 30 + 20 + 2 + 18$

107

Chapter 4

Midterm 1

idk what I need to study I'll be so honest.

4.1 Events, Entities, Laws, Policies

4.2 Queues

4.2.1 State Changes

Literally everything is a state change and you have to get al of them all the time

4.2.2 Lindley's Equaiton

It's pretty self-explanatory but it looks very fancy

The delay time of the next customer is the first customer's service + wait time minus the amount of time it took for the next customer to walk over

$$D_{i+1} = \max\{0, D_i + S_i - A_i\}$$

Where A_i is the time between customer I and $i + 1$

4.2.3 Dynamics vs Static

A static system is one that can be solved with a nice formula

A dynamic system has more moving parts that are harder

This is related to the tractability vs realism

tractability means that a problem can be hard-solved, so the most realistic solution will be just a mathematical equation.

A not-tractable problem cannot be easily mathed out, so a simulation is what will give the most realistic answer.

4.3 Level of Detail

Keep it simple stupid.

Only add the amount of detail necessary to measure what you want to measure in the simulation.

add detail only as needed

4.3.1 Simulation Verification

Set values to fake numbers

Set values to extremes

Compare simulation data to historical data.

4.3.2 Entity/Attribute Hierarchy

This is essentially the words that tell you what should be its own entity and what should be just an attribute that the entity has.

For example, a bank and a teller should both be entities for a certain system, but the service time of a teller only needs to be an attribute, but we don't need a ton of detail in the personality and work ethic of a bank teller.

You could also have just the bank itself be an entity, and the teller is the service time of the bank, but that's a little absurd.

4.3.3 Event Priorities

Event priorities are important to stop customers from being double counted at various points in the system.

Most of the time, a server starting to serve a customer should be the highest priority event.

Sometimes the event priority affects how data is collected.

4.4 All Queues

customers arrive with an interrarival time

they get served if there's a server.

They go to a queue if there's a queue.

- Multiple Servers or Multiple Server Types

- Batched Service and Arrivals
- Rework
- Limited Waiting Space
- Orbiting
- Assembly System
- Two Different Servers
- Closing Time
- Breakdowns
- Priority Queue

They all do different stuff and are all talked about in decent detail in the beautiful notes that we have.

I think the main thing I need to do is have a decent understanding of how to draw out the event graphs for any inventory or single server queue system.

4.5 Inventory Systems

They're pretty cool

People order stuff from your inventory with an interrarival time, and when you get below a certain amount of stuff, you order new stuff.

4.6 Event Graphs

Event graphs make like everything do everything.

4.6.1 Graph Reduction

Technically, all event graphs only need 1 node and 1 initialization node.

Initialize, self repeating customer arrival, end service.

Chapter 5

Stats

I missed the first 3 minutes of lecture hopefully I'm not cooked

5.1 Random Variables

Computers can't generate random numbers right off the bat, you need algorithms that use externally random things to create numbers

5.2 Linear Congruential Generator

This generates a random number from a set random state seed

$$V_i = (aV_{i-1} + c) \mod (m)$$

Where a, c, m are multiplier, increment, and modulus.

There's a proof by induction that an LCG can be generated as a function of a seed V_0

5.2.1 Potential Exam Question

Write a proof by induction for an LCG

5.2.2 Faulty LCG

Some LCG's will get caught into separate loops.

5.2.3 Theorem

An LCG only works well if these conditions are met:

- if q is a prime that divides m, then q divides $a - 1$
- The only positive integer that divides both m and c is 1
- if 4 divides m, then 4 divides $a - 1$

The last condition is related to the first part condition, but 4 is not prime.

My first guess is because 4 is 2×2, and 2 is an even prime?

Genuinely have no idea.

m should equal 2^B where B is the number of bits in the machine in order to make the most random random number generator possible.

If $c = 0$, then the LCG is called a power residue or a multiplicative generator.

Don't invent your own LCG. We've found all the good ones.

5.3 Testing RNG's

You can look at the mean and variance of an RNG

for a $U(0, 1)$ RNG, the expected value should be $1/2$, and the variance should be $1/12$ for a uniform distribution.

The values of an LCG approach these values at $m \to \infty$

5.3.1 Emprical Testing for Uniformity

You can do a chi-squared goodness of fit test with an alternative and null hypothesis

Compute the frequency count of an LCG and compute the test statistic.

You can get a multinomial distribution.

If the test statistic is less than the chi-squared, then you do not reject the null hypothesis.

You should get a type 1 error 5% of the time because that's how stats work.

5.3.2 Trouble Spots

Choose the intervals for your test statistic evenly

Choose the intervals such that you would expect each class to contain at least 5 or 10 observations.

p_i should (ideally) be small.

There's an example of a test stat in the lecture slides.

5.4 Kolmogorov-Smirnov Goodness-of-Fit Test

Use the CDF rather than the PDF (which the chi-squared test uses) (cumulative distribution function vs probability density function)

Construct an emperical CdF for the n ordered values

Construct a hypothesized CDF for the n IID U(0, 1) variates

Compute D = max (D+, D-), where those variables are some math

There's another nice illustration in the lecture slides.

This can be tested for all possible distributions

The smallest possible value that you can get for D is $1/2n$

The smallest possible value is $1/8$

We have a table to figure out the critical values to reject the null hypothesis.

Reject H_0 in favor of H_a if $D < D_\alpha$

5.4.1 Anderson-Darling Test

You take a weighted average of the squared distances between the ideal CDF and the sampled CDF.

5.5 Testing for Independence Using Data

This is important because IID means independent

5.5.1 Sign Test

$S =$ the runs of numbers above or below the median

For large N, S is distributed with a certain mean and standard deviation

If S is large, then you have positive dependency, and that's bad

If S is small, then you have negative dependency, and that's also bad

5.5.2 Runs Up and Down Test

Runs of increasing and decreasing numbers

Calculate the numbers of runs of +'s and -'s

You should have a very specific means and standard deviation.

5.6 How Do We Drive a Simulation?

Driving means creating the random variables to run a simulation

5.6.1 Trace

Use historical data files and directly plug them in.

This works, but its very difficult, and it's bad at showing all scenarios

5.6.2 Empirical or Nonparametric Distributions

Histograms

Just make up data and hope that it's good

5.6.3 Parametric Probability Models

- Use normal or exponential distributions
- This is usually what is used like all the time.
- There's a framework for choosing parametric distributions
- Hypothesize various probability models
- Estimate parameters by matching moment
- Test adequacy of model using some goodness-of-fit test
- something else

There's more nice examples in the lecture slides.
Your statistical tests might not work perfectly
All models are wrong, but some are useful.

Statistical tests work well until they don't.

If we make α much smaller instead of 0.05, then maybe that changes something about your conclusion.

5.6.4 Another Approach

What you can do in taking sampled data and connect all the dots to make a continuous probability distribution.

5.6.5 Comparison

There are a number of ways that you can compare two algorithms, and it basically comes down to how valid is the algorithm and how computationally expensive is it.

5.7 Transformations

Given U(0, 1) variables, we want variables in a different distribution function.

5.8 Inversion

Theorem : If F is invertible, then Y = F(X) is disributed U(0, 1).

Basically you can invert a CDF in order to get a different distribution function.

5.8.1 How to Invert a CDF

Given a CDF denoted at F(X), we set F(X) to a U(0, 1) random variable denoted as u, and we solve for x

$$u = F(X) = 1 - e^{-x/\mu} \Rightarrow\Rightarrow\Rightarrow x = -\mu \ln(1-u)$$

That's the inversion of an exponential CDF

5.8.2 INVERSION IS ON THE EXAM 2

5.8.3 Trangular Distribution

These are probability distributions that have 2 separate parts.
To get the CDF, you integrate the PDF

5.9 Geometric Random Variable

Number of Bernoulli trials until the first success (with probability p)

You get a probability mass function and a cumulative distribution function.

Set the CDF to u and get an inversed function.

You have to round up to get

$$k = \lceil \ln(1-u)/ln(1-p) \rceil$$

5.10 Advantages and Disadvantages

Inversion of a function is not always known.

However.

It is very good for variance reduction, and it only uses a $U(0,1)$ variable, and its great for handling truncated distributions.

5.11 Order Statistics

Given a bajillion observations, you want to put them in a certain order.

If we consider that, the first item is the failure of a serial system, while the last element is the failure time of a parallel system.

This is on page 66 because I'm confused

There are algorithms that get you the CDFs for serial and parallel system failures.

As we get more and more U(0, 1) elements, the maxima and minima will get closer and closer to 1 and 0.

"With order statistics, inversion is Nirvana.

5.12 Special Properties of Random Variables

Erlang is a sum of r IID exponentials (convolution)

There are Gamma and Beta (a ratio of Gamma)

There are all sorts of Beta distribution shapes given a Gamma random variable.

We get Gammas from Exponentials and Betas from Gammas.

5.13 Binomial Random Variable

The main way to calculate a binomial random variable is just count the number of successes of a Bernoulli trial.

This uses n Bernoulli random variables to brute force a binomial distribution.

There's a similar brute force method for Geometric Random Variables.

5.14 Acceptance-Rejection

You take a uniform U(0, 1) variable, and you accept it if it fits in your other distribution, and you reject it if it does not fit in your distribution.

Generate a point (y, x), and if $y < f(x)$, report x, and if $y > f(x)$ try again.

$f(x)$ is the probability distribution function that we want to mimic.

5.14.1 Minorizing Function

It's just a heuristic of your PDF. If you PDF sucks, then the minorizing function is much easier to compute for any x.

5.14.2 Generalized Acceptance Rejection

Its a majorizing function that is greater than $f(x)$ for all x

Generate x from the majorizing pdf and generate y uniformly between 0 and $g(x)$.

Now you have a new method for acceptance rejection

- Generate w with pdf $k(w) = g(w)/A_g$
- Generate $y = U(0, g(x))$
- if y ¡ f(x), accept x, otherwise, go to 1

Where A_g is the normalizing constant which is the area under the $g(x)$ majorizing curve.

This just makes it faster.

The majorizing function is whatever you want it to be.

5.14.3 Biggest Problems

How do we actually choose the majorizing and minorizing functions suc hthat

- It is easy to geenrate points under the majorizing function
- the minorizing functions must be easy to compute

Also, how do we adapt this acceptance-rejection algorithm to an infinite domain?

The answer is you can, but you need to choose a majorizing function that also converges to 0 as $\pm x \to \infty$.

You should also pick a very nice integrable function so that you can normalize it to make an easy pdf and cdf.

You can then generate from that majorizing function using a $U(0, 1)$ variable and the inverse math that we discussed earlier.

5.15 Poisson Random Variable

It's just another probability distribution function.

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!} \qquad E(X) = \lambda$$

A Poisson random variable is the number of IID exponential random variables whose sum is under some value.

But how can we make a neffieicnet algorithm of a Poisson(λ)?

Let$\{Y_i\}$ be IID exp(λ) and $\{U_i\}$ be IID U(0, 1).

Then X is distributed Possion(λ) iff

$$\sum_{i=1}^{X} Y_i \leq 1 < \sum_{i=1}^{X+1} Y_i$$

If you do some math, you get

$$\prod_{i=1}^{X} U_i \geq e^{-\lambda} > \prod_{i=1}^{X+1} U_i$$

5.16 Poisson Process

Times at which events occur, where the interarrival times of the events ar edistribution exponential.

A homogeneous poisson Process is defined as

$$t_i = t_{i-1} - \frac{1}{\lambda}\ln(U_{i-1})$$

5.16.1 Inhomogeneous Poisson Process

The rate changes over time, but has a maximum.

- Generate a homogeneous Poisson process with rate λ_{max}
- Accept arrivals with probability $\lambda(t)/\lambda_{max}$
- Accepted arrivals are the Inhomogeneous Poisson Process

The easiest way to do this is with a self-schedulign event

$$t_a \approx \exp(\lambda_{max}) \qquad Q = Q + I(u \leq \lambda(t)/\lambda_{max})$$

5.17 Generating Random Permutations

You basically just do a for loop and from 1 to N swap A(N) with a random index determined by a U(0, 1) value.

5.18 General Discrete Random Variables

Discrete variables can be hard to generate

Imagine a Binomial(n, p) random variable with very large N and very small p

If a large number of observations are needed, how can they be generated efficiently?

5.18.1 Alias Method

Use when

- there are a large number of discrete values
- you want to generate many variates from this distribution

It requires only a single U(0, 1) variable

Transforms a discrete random variable into a whole distribution.

What you do is take a $U(0, 1)$ random variable, multiply is by you range, round up, and boom you now have a uniformly discrete random variable.

However, what if you non-uniform random variable

Define Q_i is the probaility thta i is actually chosen given that i is first selected = P(i chosen — i selected)

What you do is partition the probabilities with only either 0 or 1 cut point per value such that now it follows your non-uniform distribution

- 25% for 1 becomes 20% for 1 and 5% for 2
- 25% for 2 becomes 10% for 2 and 15% for 3
- 25% for 3 stays that way

- 25% for 4 gets cut up to something

So, if the U(0, 1) variable ends up in any of those 4 sectors, you do another U(0, 1) variable

If N is really really large, the algorithm is O(1) instead of O(n), which is what an inversion algorithm would do.

The algorithm is pretty neat.

5.18.2 Alias Algorithm

LOOK AT SLIDES AND FIGURE IT OUT

5.19 Marsaglia's Method

For discrete random variables with denominators as powers of 2

Use when there's a large number of values and we need alot of them.

Requires a single U(0, 1) variable

5.19.1 Algorithm

for urns 1/2, 1/4, 1/8, etc. place segments in each urn for the amount of items with probabilities in those segments.

Then apply the law of total probabilty to the segments in each urn.

- Pick an urn with probability q_i

- Pick a value from the urn using a discrete uniform

THAT'S WHY YOU NEED THE PROBABILITY TO HAVE A DENOMINATOR THATS A POWER OF 2.

There's some math to turn the one variable into the variable for the alias probability.

5.20 Normal Random Variable

There is a better way to do it than the acceptance-rejection method we went over earlier

5.20.1 Crude Method

This is also known as the Central Limit Theorem Method

If you take a bunch of uniform variables, you get a standard normal distribution

consider taking $n = 12$ U(0, 1) variables

$$(Y - n/2)/(n/12)^{1/2} \to N(0,1) \qquad n \to \infty$$

This doesn't work well for $n = 12$ because n is too small.

5.20.2 Bivariate Method

If we take a standard normal and square and sum them, it'll be distributed chi squared with 2 degrees of freedom, which is distributed exponentially with mean 1/2.

You use inversion to get an exponential generator

$$D^2 = X_1^2 + X_2^2 = \chi_2^2 \equiv \text{exponential}(1/2)$$
$$D^2 = -2\ln(U)$$

So you do some math and get some stuff that yield's you a good normal distribution.

The issue is the variables are not super independent which sucks.

5.20.3 Polar Method

Let U_1 and U_2 be U(0, 1) variables

$$V_i = 2U_i - 1 \qquad W = V_1^2 + V_2^2$$

If $W > 1$, then we do a thing
If $W < 1$, then

$$Y = (-2\ln(W)/W)^{1/2} \qquad X_1 = V_1 Y, X_2 = V_2 Y = N(0,1)$$
$$X_1^2 + X_2^2 = -2\ln(W)$$

5.21 ALIAS AND URNS MAYBE ON EXAM 2

5.22 BASIC CALCULUS ON MIDTERM PROBABLY FOR A PROOF OR 2

That's the end of the Unit

Chapter 6

Statistical Analysis

The slides are going over a bunch of terms that we should know.

6.1 Variance

A random variable with a variance of 0 is deterministic. It's just a value.

$$Var(cX) = c^2 Var(X)$$

Random Sampling is also important
Confidence Intervals, convergence, etc etc etc
Normal stats only works if all your variables are IID
That doesn't work in simulation stats.

6.2 Project

You have to write an 8-10 page paper and an 8-10 minute youtube video.

The goal is to estimate, with a 95% confidence interval, the number of card draws necessary to get a matching pair.

But the cards are banknotes and a match is the serial numbers.

You CANNOT just check every card to every other card. Your time complexity will be absurd because you're working with a bajillion cards.

6.2.1 Intended Solution Hints

- What information is necessary to store? Can we store it more efficiently?
- How can we estimate variance efficiently? (Keep this one in mind for the next set of slides)
- Can we substitute empirical work with theory?

6.3 Confidence Intervals

Because simulations can be repeated, we can repeat tests multiple times to create a variety of confidence intervals to figure out the true mean.

6.4 Correlation Coefficient

-1 is inverse correlation, +1 is positive correlation, 0 is no correlation.

The fun thing about doing stats with simulation data is that our numbers are absolutely correlated, while in stats, we're meant to be taking random samples.

Simulation data tends to be positively correlated and biased.

6.5 Simulation Stats

What you need to do is treat full simulations as just singular observations.

Once you have a full sample size of simulations, then you can start getting real numbers.

6.6 Performance Measures

What are the numbers that we want to take out of the simulation

6.6.1 Transient

Measures that change over time

6.6.2 Steady-State

Measures that converge to a data point given more and more time.

6.7 Types of Simulations

6.7.1 Terminating

Simulations that close after a day and thus "end" at some point.

6.7.2 Steady-State

Simulations that can run forever if they really want to.

6.8 Performance Stats

If we take data from individual runs, it will be correlated, but if we take the mean of multiple datapoint from multiple runs, our numbers will be IID and the stats should work fine.

6.8.1 Ratio Estimates

If we're looking at the ratios of two different numbers, there are two ways to get your data.

You can find the average of each ratio, which gives you the average ratio in a certain time frame.

You can add up all the numerators and denominators, which will then give you a long running average.

6.9 Steady State Simulation Issues

- The simulation can run forever
- How long do you need it to run to stabilize?
- Initial condition shouldn't affect the steady state
- a Steady State does NOT mean a single value, but instead it means that the distribution of values becomes time invariant

6.10 Steady State Measures

Given any set of initial conditions, all the steady state values should converge to the same steady state distribution.

Many systems do NOT have a steady-state. Steady states only exist in systems that do not terminate and have the potential to reach a steady state.

Steady state features have to last forever (no closing), but transient measures can have steady-state like features.

For a terminating system. The wait time for a certain time of day is never a steady system, but the average wait time on any given day can converge to a steady state system.

6.10.1 Steady State Initial Conditions

It would work in theory, but we don't know those conditions.

What we do instead is leave a "warm up" time and discard the first few values so that the system is in a steady state by the time we start recording.

The sum is still biased because of correlations in adjacent customers/datapoints.

6.11 Transient Behavior

The effect that initial conditions have on the output.

If the initial transient approaches 0 over a very long time, then the simulation can reach a steady state.

6.12 Welch's Test

Run the simulation n times with the same initial conditions and calculate m datapoints.

We take the average of the datapoints at a specific time from every simulation. (The vertical average)

This essentially gives us a smoothing effect of the transient.

All of the randomness of the simulation will cancel out so the steady state becomes far more clear (because the simulation itself is the only thing bringing in noise).

6.12.1 Steady State Point Estimate

Consider μ as the long-run expected value for Y_i.

Set

$$\bar{Y}(m) = \frac{1}{m}\sum_{i=1}^{m} Y_i$$

The simulation output is typically truncated to get rid of the initial data.

$\bar{Y}(m)$ is a consistent estimator for μ

6.12.2 Steady State Confidence Interval

$\bar{Y}(m)$ is an observation of dependent variables, so there is not a good way to directly calculate the variance of the data.

6.13 Methods

6.13.1 Method of Replication

Make b independently seeded runs with m observations per run. (removing transient data)

Take the sum of the sum of the observations

$$E(Y_{ij}) = \bar{Z} \pm t_{b-1,0.25} s/b^{1/2}$$

where t is the t-value of the confidence interval.

The total number of observations is $bm = n$

increase m to increase normality

Each run has initialization bias and there is correlation within each individual run.

What you can do is warm up a single run and use those steady state conditions as the initial conditions.

But will using the same steady state conditions for every run cause biases and correlations?

6.13.2 Batch Means Method

A variation of the replication method

consider a sequence of dependent data Y_i

If, instead of using i and $i+1$, we use $i+k$ for some large-ish k instead, then our steady state datapoints are going to be almost independent.

We take only a single run, and we split it up into batches $1, 1+b, \ldots$, and $2, 2+b, \ldots, \ldots$

The main bias will be within adjacent batches.

You use the batch means to estimate the variance of the steady state.

The optimal way to do it is a small number of very large batches.

Some possible batch variants are

- put gaps between batches
- have batches overlap (which somehow works)

There's some math to calculate the covariance between two batches to double check if they are correlated or not.

6.14 COVARIANCE AND CORRELATION WILL BE ON THE EXAM

6.15 Regenerative Method

- Takes advantage of the regenerative structure of a system
- A system that regenerates has points in which the system resets and future events are not affected by past events.
- (Image when a queue empties out for a single server queue, then the future runs don't impact the past runs because the queue emptied out)
- THe processes between regenerative points are IID
- Consider a sequence of correlated data.
- For each sample of data between regenerative cycles. Take the total simulation output Z and the total number of samples N from each cycle.
- You can then use both Z and N to make a confidence interval for the expected value of the system.
- Because Z and N are IID, you can find $E(Z)$ and $E(N)$ trivially, and then $E(Y) = E(Z)/E(N)$
- as the number of regenerative cycles $r \to \infty$, the bias and variance of the system both go to 0.

6.15.1 Pros and Cons

Advantages :

- No initial transient
- Independent observations
- Simple
- Asymptotically Exact

  Distadvantages

- Unknown cycle lengths
  What do you do with the last unfinished cycle?
- Simulation may have veery very long cycles
- Strong bias of estimates
  This is an issue for finite r

6.16 Blood Bank Example

Donors and Recipients arrive and donors give a consistent amount while recipient request a poisson distribution.
You can estimate the data with any method.
The regenerative cycle occurs when the inventory is 0
Replication uses 40000 days and discards 10000
Batch uses 20000 and discards 500

Regenerative method uses 7793 cycles (because thats the number of days it took to get 50 regenerative cycles)

Each gives a different daily inventory but the average is roughly 23 pints

The reason the daily inventory is so large is because blood expires.

6.17 Variance Reduction Techniques

VRT improves the quality of estimators (reduces variance while remaining unbiased)

Variance can usually be decreased by increasing N. This may be slow (Var($\bar{Y}$) = O(1/N))

The simulation is a blackbox that takes in arrivals and services and returns output processes (wait times)

Generally, the more people that show up, the more you'll have to wait.

6.17.1 Antithetic Random Numbers

Just turn U_i into $(1 - U_i)$ to flip the dependency

6.18 CAN BE ON EXAM 2

The relationship between random variables and their outputs(?)

If you use both the random variance and its Antithetic variate then you have double the data to make some numbers.

The covariance between U and $1-U$ is perfect negative correlation -1

6.18.1 EXAM QUESTION MAYBE

Find the covariance between $F^{-1}(U)$ and $F^{-1}(1-U)$

Let $F = 1 - \exp(-\lambda x)$

There's a bunch of math in the slides but you should get -1, which makes sense.

6.19 Control Variates

You can do thing to induce negative dependency.

Let $E(Y) = \Theta$ and a correlated variable M with $E(M) = \mu$

Let $R = Y + \alpha(M - \mu)$ With $E(Y) = E(R)$ and some math for the covariance

A good choice for M is to have the correlation be either 1 or -1

6.19.1 How to Choose

Since we know that the arrival time the wait times are negative correlated, we can use that.

S and W are positively correlated, so we can use those as well.

You should be able to use Control Variate (CV) shenanigans as unbiased estimators.

Control Variates can be combined together.

6.19.2 Variance Reduction Through Conditioning

let Y be a simulation output and V be a random vraiable that can be used to decompose the output.

Let Y be the waiting times for a queue with 2 distinguishable servers V.

$$E(Y) = E(E(Y|V))$$

The conditioned estimator has lower variance than the unconditioned estimator.

You can get free lunch (something for nothing) just by messing with the numbers.

6.19.3 Bank Example

Look at the slides

Basically you get look at the exact same data but from the variance from 2.7 to like 0.05

The solution to the project can be done analytically.

6.20 Monte Carlo Simulation

Characteristics of Monte Carlo Techniques

- Underlying problem is deterministic or stochastic

- There's a stochastic model with the same expected value as the original
- can solve problems that are really hard/impossible to solve analytically
- Can be more efficient than deterministic techniques
- Allows you to sample from a pdf (probability density function) previously unknown

6.20.1 Applications

Literally everything

Any sort of estimates of anything

You literally use this for your research.

6.20.2 Buffon Needle Problem

If you randomly drop a needle, what are the chances that it intersects a line given a random position and angle.

Find the number of crosses / the number of tosses.

You can estimate pi with a monte carlo simulation

There are a bunch of other simulation examples in the slides.

6.21 EXAM 2

Using common and antithetic variance to calcualte covariance.