

**HANDOUTS | MATERIAL
HIGHLIGHTED AND PROVIDED BY :**

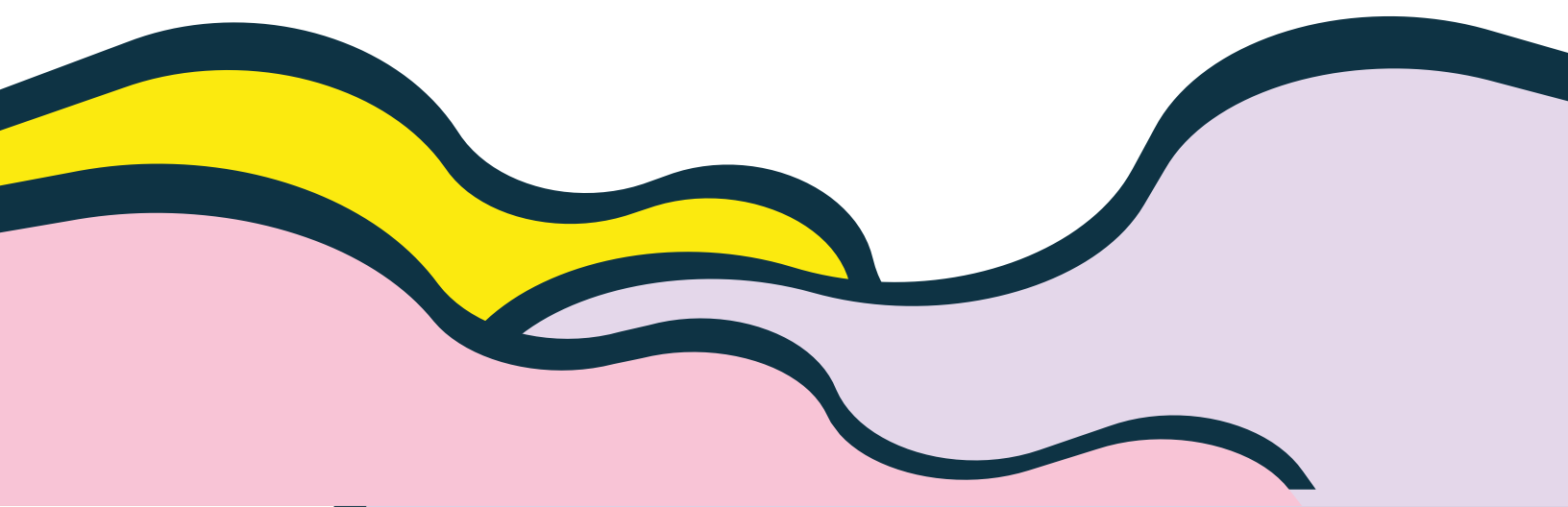
VU TALK

CLICK TO JOIN OUR WHATSAPP GROUP : [click me](#)

VU TALK YouTube CHANNEL : [click me](#)

For Any Help Contact Me On Whatsapp : [click me](#)

Follow VU TALK on Whatsapp Channel : [click me](#)



CS620 Modeling & Simulation

Handouts Organized by Scammer404

Module – 1

- Modeling & Simulation is an extremely important part of our daily lives.
- Our entire living experience is based on the concept of “Consciousness”
- The state of being aware of and responsive to one's surroundings

Module – 2

Analysis of Complex Systems:

- Simulation enables the study of, and experimentation with, the internal interactions of a complex system or of a subsystem within a complex system.

Defining Changes:

- Informational, organizational, and environmental changes can be simulated, and the effect of these alterations on the model's behavior can be observed.

Knowledge from Simulation:

- The knowledge gained during the designing of a simulation model could be of great value toward suggesting improvement in the system under investigation.

Learning from Change:

- Changing simulation inputs and observing the resulting outputs can produce valuable insight into which variables are the most important and into how variables interact.

Pedagogical Value:

- Simulation can be used as a pedagogical device to reinforce analytic solution methodologies.

Trial & Error:

- Simulation can be used to experiment with new designs or policies before implementation, so as to prepare for what might happen.

Verification of Mathematical Models:

Simulation can be used to verify analytic solutions.

Learning Requirements for a Machine:

- Simulating different capabilities for a machine can help determine the requirements on it.

Replacement for Training:

- Simulation models designed for training make learning possible without the cost and disruption of on-the-job instruction.

Animation of Simulation:

- Animation shows a system in simulated operation so that the plan can be visualized.

Visually Understanding Modern Factories:

- The modern system (factory, wafer fabrication plant, service organization, etc.) is so complex that its internal interactions can be treated only through simulation.

Other reasons!

- Explain (very distinct from predict)
- Guide data collection
- Illuminate core dynamics
- Suggest dynamical analogies
- Discover new questions
- Promote a scientific habit of mind
- Bound (bracket) outcomes to plausible ranges
- Illuminate core uncertainties.
- Offer crisis options in near-real time
- Demonstrate tradeoffs / suggest efficiencies
- Challenge the robustness of prevailing theory through perturbations
- Expose prevailing wisdom as incompatible with available data
- Train practitioners
- Discipline the policy dialogue
- Educate the general public
- Reveal the apparently simple (complex) to be complex (simple)

Module – 3

Simulation: When isn't an Appropriate Tool?

Simulation is NOT a Silver Bullet!

- There is a typical misconception often in graduate students that everything can be, or should be simulated.
- Simulation is NOT for everything!

Banks and Gibson (1997):

- Banks and Gibson present ten rules for evaluating when simulation is not appropriate

Rule Number 1:

- Simulation should not be used when the problem can be solved by common sense.

Example:

- E.g. In an automobile tag facility serving customers who arrive randomly at an average rate of 100/hour and are served at a mean rate of 12/hour.
- To determine the minimum number of servers needed, simulation is not necessary.
- Just compute $100/12 = 8.33$ indicating that nine or more servers are needed.

Rule Number 2:

- Simulation should not be used if the problem can be solved analytically.
- For example, under certain conditions, the average waiting time in the example above can be found from curves that were developed by Hillier and Lieberman [2002].

Rule Number 3:

- Simulation should not be used if it is easier to perform direct experiments.

Example:

- In a fast-food drive-in restaurant it can be less expensive to stage a person taking orders using a hand-held terminal and voice communication to determine the effect of adding another order station on customer waiting time.

Rule Number 4:

- Don't use simulation if the costs exceed the savings.
- If a simulation study costs \$20,000 and the savings might be \$10,000, simulation would not be appropriate.

Rule 5 & 6:

- Simulation should not be performed if the resources or time are not available.

Examples:

- If the simulation is estimated to cost \$20,000 and there is only \$10,000 available, the suggestion is not to venture into a simulation study.
- Similarly, if a decision is needed in two weeks and a simulation will take a month, the simulation study is not advised.

Rule 7:

- Simulation takes data, sometimes lots of data.
- If no data is available, not even estimates, simulation is not advised.

Rule 8:

- The next rule concerns the ability to verify and validate the model.
- If there is not enough time or if the personnel are not available, simulation is not appropriate.

Rule 9:

- If managers have unreasonable expectations, if they ask for too much too soon, or if the power of simulation is overestimated, simulation might not be appropriate.

Rule 10:

- Last, if system behavior is too complex or can't be defined, simulation is not appropriate.
- Human behavior is sometimes extremely complex to model.

Module – 4

Simulation: Advantages and Disadvantages?

Simulation is NOT a Silver Bullet!

- Simulation is intuitively appealing to a client because it mimics what happens in a real system or what is perceived for a system that is in the design stage.
- The output data from a simulation should directly correspond to the outputs that could be recorded from the real system.
- Additionally, it is possible to develop a simulation model of a system without dubious assumptions (such as the same statistical distribution for every random variable) of mathematically solvable models.
- For these and other reasons, simulation is frequently the technique of choice in problem solving.
- For these and other reasons, simulation is frequently the technique of choice in problem solving.

Simulation vs. Optimization:

- In contrast to optimization models, simulation models are "run" rather than solved.
- Given a particular set of input and model characteristics, the model is run and the simulated behavior is observed.
- This process of changing inputs and model characteristics results in a set of scenarios that are evaluated.
- A good solution, either in the analysis of an existing system or in the design of a new system, is then recommended for implementation.

Simulation:

– Simulation 'has many advantages, but some disadvantages.

– Pegden, Shannon, and Sadowski [1995] summarize these.

Benefits:

- New policies, operating procedures, decision rules, information flows, organizational procedures, and so on can be explored without disrupting ongoing operations of the real system.
- New hardware designs, physical layouts, transportation systems, and so on can be tested without committing resources for their acquisition.

- Hypotheses about how or why certain phenomena occur can be tested for feasibility.
- Time can be compressed or expanded to allow for a speed-up or slow-down of the phenomena under investigation.
- Insight can be obtained about the interaction of variables.
- Insight can be obtained about the importance of variables to the performance of the system.
- Bottleneck analysis can be performed to discover where work in process, information, materials, and so on are being delayed excessively.
- A simulation study can help in understanding how the system operates rather than how individuals think the system operates.
- "What if" questions can be answered. This is particularly useful in the design of new systems.

Module – 5

Disadvantages of Simulation:

- Model building requires special training.
- It is an art that is learned over time and through experience.
- Furthermore, if two models are constructed by different competent individuals, they might have similarities, but it is highly unlikely that they will be the same.
- Simulation results can be difficult to interpret
- Most simulation outputs are essentially random variables (they are usually based on random inputs), so it can be hard to distinguish whether an observation is a result of system interrelationships. or of randomness.
- Simulation modeling and analysis can be time consuming and expensive.
- Skimping on resources for modeling and analysis could result in a simulation model or analysis that is not sufficient to the task.
- Simulation is used in some cases when an analytical solution is possible, or even preferable.
- This might be particularly true in the simulation of some waiting lines where closed-form queueing models are available.

In Defense of . . .

- Vendors of simulation software have been actively developing packages that contain models that need only input data for their operation.
- Such models have the generic tag "simulator" or "template."
- Many simulation software vendors have developed output-analysis capabilities within their packages for performing very thorough analysis.
- Simulation can be performed faster today than yesterday and will be even faster tomorrow, because of advances in hardware that permit rapid running of scenarios and because of advances in many simulation packages.
- For example, some simulation software contains constructs for modeling material handling that uses such transporters as fork-lift trucks, conveyors, and automated guided vehicles.

- Closed-form models are not able to analyze most of the complex systems that are encountered in practice.
- In many years of consulting practice, not one problem was encountered that could have been solved by a closed-form solution.

Module – 6

Technical Societies Co-Sponsoring WSC:

- American Statistical Association (ASA),
- Association for Computing Machinery/Special Interest Group on Simulation (ACM/SIGSIM),
- Institute of Electrical and Electronics Engineers: Computer Society (IEEEICS),
- Institute of Electrical and Electronics Engineers: Systems, Man and Cybernetics Society (IEEEISMCS),
- Institute of Industrial Engineers (IIE),
- Institute for Operations Research and the Management Sciences: College on Simulation (INFORMS/CS) and
- The Society for Computer Simulation (SCS)

Manufacturing Applications:

- Dynamic modeling of continuous manufacturing systems, using analogies to electrical systems
- Benchmarking of a stochastic production planning model in a simulation test bed
- Paint line color change reduction in automobile assembly
- Modeling for quality and productivity in steel cord manufacturing
- Shared resource capacity analysis in biotech manufacturing
- Neutral information model for simulating machine shop operations

Semiconductor Manufacturing:

- Constant time interval production planning with application to work-in-process control
- Accelerating products under due-date oriented dispatching rules

- Design framework for automated material handling systems in 300 mm wafer fabrication factories

Construction Engineering & Project Management:

- Impact of multitasking and merge bias on procurement of complex equipment
- Application of lean concepts and simulation for drainage operations maintenance crews
- Building a virtual shop model for steel fabrication
- Simulation of the residential lumber supply chain

Module – 7

Military Applications:

- Frequency-based design for terminating simulations: A peace-enforcement example
- A multi-based framework for supporting military-based interactive simulations in 3D environments
- Specifying the behavior of computer-generated forces without programming
- Fidelity and validity: Issues of human behavioral representation
- Assessing technology effects on human performance through trade-space development and evaluation
- Impact of an automatic logistics system on the sortie-generation process
- Research plan development for modeling and simulation of military operations in urban terrain

Logistics, Supply Chain & Distribution Applications:

- Inventory analysis in a server-computer manufacturing environment
- Comparison of bottleneck detection methods for AGV systems
- Semiconductor supply-network simulation
- Analysis of international departure passenger flows in an airport terminal
- Application of discrete simulation techniques to liquid natural gas supply chains
- Online simulation of pedestrian flow in public buildings

Transportation Modes and Traffic:

- Simulating aircraft-delay absorption

- Runway schedule determination by simulation optimization
- Simulation of freeway merging and diverging behavior
- Modeling ambulance service of the Austrian Red Cross.
- Simulation modeling in support of emergency firefighting in Norfolk
- Modeling ship arrivals in ports
- Optimization of a barge transportation system for petroleum delivery
- Iterative optimization and simulation of barge traffic on an inland waterway

Business Process Simulation:

- Agent-based modeling and simulation of store performance for personalized pricing
- Visualization of probabilistic business models
- Modeling and simulation of a telephone call center
- Using simulation to approximate sub-gradients of convex performance measures in service systems
- Simulation's role in baggage screening at airports
- Human-fatigue risk simulations in continuous operations
- Optimization of a telecommunications billing system
- Segmenting the customer base for maximum return

Healthcare:

- Modeling front office and patient care in ambulatory health care practices
- Evaluation of hospital operations between the emergency department and a medical telemetry unit
- Estimating maximum capacity in an emergency room
- Reducing the length of stay in an emergency department
- Simulating six-sigma improvement ideas for a hospital emergency department
- A simulation-integer-linear-programming-based tool for scheduling emergency room staff

Module – 8

System and Environment:

- In this module, we shall examine the concepts of system and environment
- Before we are able to model anything, we need to have some basic concepts.
- There are two basic concepts of importance.
- System
- System Boundary
- A system is defined as a group of objects joined together in some regular interaction or interdependence toward the accomplishment of some purpose

Manufacturing Applications:

- Dynamic modeling of continuous manufacturing systems, using analogies to electrical systems
- Benchmarking of a stochastic production planning model in a simulation test bed
- Paint line color change reduction in automobile assembly
- Modeling for quality and productivity in steel cord manufacturing
- Shared resource capacity analysis in biotech manufacturing
- Neutral information model for simulating machine shop operations

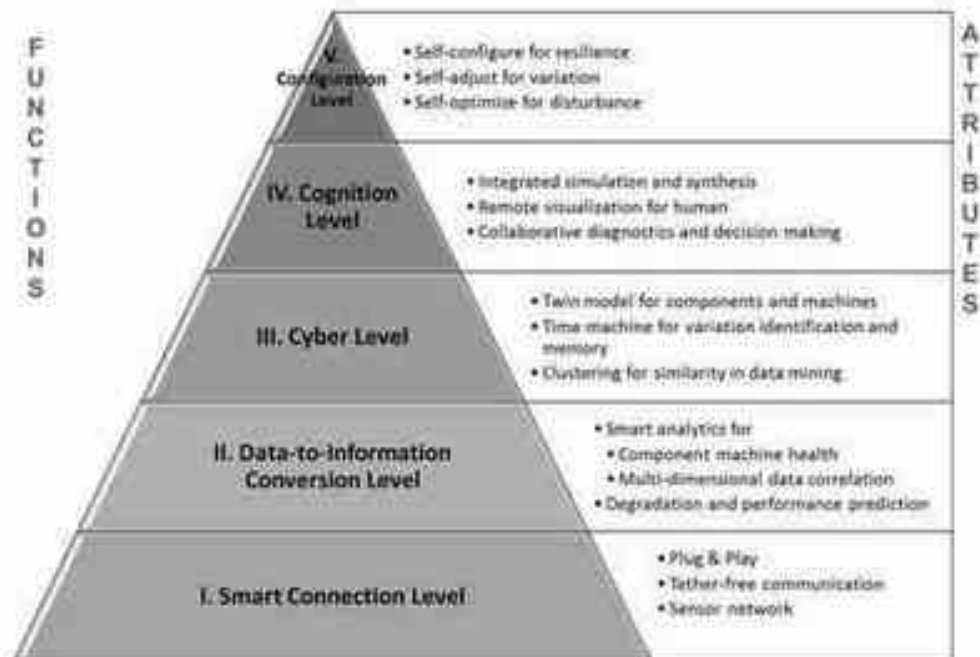
Example: BMW Car Production System:



Example: Aircraft Arresting System:



Example: Cyberphysical System:

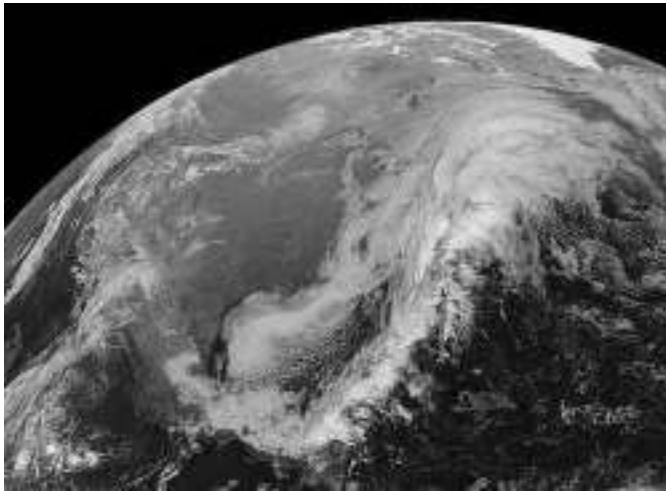


System Environment & System Boundary:

- Changes in a “system environment” occur based on changes outside the system
- Modeling system requires identification of the concept of “System Boundary”

Examples:

- In a factory system, arrival of orders can be considered as outside of the influence of the factory.
- Therefore it can be a part of the environment
- If the effects of supply on demand are important, then this must be part of the system

Example: Storm System:**Summary:**

In this module, we have started to understand the concept of a system.

Module –9

System Concepts:

- **System:** An organized entity made up of interrelated and interdependent parts.
- **Boundaries:** Barriers that define a system and distinguish it from other systems in the environment.
- **Homeostasis:** The tendency of a system to be resilient towards external factors and maintain its key characteristics.
- **Adaptation:** The tendency of a self-adapting system to make the internal changes needed to protect itself and keep fulfilling its purpose.
- **Reciprocal Transactions:** Circular interactions that systems engage in such that they influence one another.
- **Feedback Loop:** The process by which systems self-correct based on reactions from other systems in the environment.
- **Throughput:** Rate of energy transfer between the system and its environment during the time it is functioning.
- **Microsystem:** The system closest to the client.
- **Mesosystem:** Relationships among the systems in an environment.
- **Exosystem:** A relationship between two systems that has an indirect effect on a third system.
- **Macrosystem:** A larger system that influences clients, such as policies, administration of entitlement programs, and culture.
- **Chronosystem:** A system composed of significant life events that can affect adaptation.

Summary:

Here, we have examined various concepts related to systems.

Module – 10

Components of a system:

In this module, we shall examine the concepts of system components.

Motivation:

Once we know about system and system boundary, there are some other concepts that we need to define

Entity and more!:

- Any object of interest in the system is called an entity
- An attribute is the property of an entity
- An activity represents a time period of a specific length

State:

Collection of variables necessary to describe the system at any time, relative to the objectives of the study

Event:

Instantaneous occurrence that might change the state of the system

Endogenous vs. Exogenous:

- Endogenous is used for activities and events occurring within the system
- Exogenous is used for activities and events occurring in the environment that affect the system

Examples:

System	Entities	Attributes	Activities
System	Entities	Attributes	Activities
Banking	Customers	Checking-account balance	Making deposits
Rapid rail	Riders	Origin; destination	Traveling
Production	Machines	Speed; capacity; breakdown rate	Welding; stamping
Communications	Messages	Length; destination	Transmitting
Inventory	Warehouse	Capacity	Withdrawing

Examples:

System	Events	State Variables
Banking	Arrival; departure	Number of busy tellers; number of customers waiting
Rapid rail	Arrival at station; arrival at destination	Number of riders waiting at each station; number of riders in transit
Production	Breakdown	Status of machines (busy, idle, or down)
Communications	Arrival at destination	Number waiting to be transmitted
Inventory	Demand	Levels of inventory; backlogged demands

Summary:

- We talked about some basic concepts of systems
- We also discussed some examples.

Module – 11

Discrete and Continuous:

In this module, we shall examine the concepts of discrete and continuous systems.

Motivation:

Once we know about system and system boundary, there are some other concepts that we need to define.

How computers handle the real world?

- In the real world, things are complex.
- Measuring the exact values is important for us to develop the most appropriate model
- Computers by themselves are based on digital circuits.
- As such, intrinsically, their internal representation is based on discrete (clear steps)

Event:

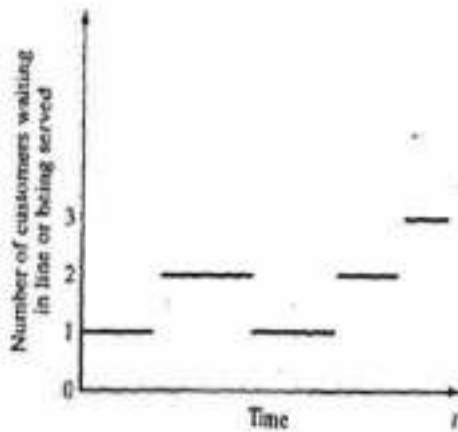
Few systems in practice are wholly discrete or continuous, but since one type of change predominates for most systems, it will usually be possible to classify (Law and Kelton , 2009)

Discrete System:

A discrete system is one in which the state variables change ONLY at discrete set of points in time.

Example:

- Customer arrival in a bank.
- The state variable 'Number of customers' changes only when a customer arrives or when the service is completed.

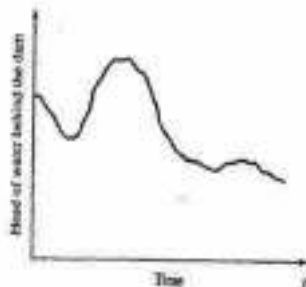


Continuous System:

A Continuous system is one in which the state variables change CONTINUOUSLY over time.

Example:

- Head of water behind a dam.
- The state variable 'Head of Water' changes as rain comes in or dam is emptied etc.



Summary:

- We talked about discrete and continuous systems
- We also discussed some examples

Module – 12**Model of a System:**

In this module, we shall examine what it means by developing a system model.

Motivation:

How do we model a system?

Why Model?

- Sometimes it is of interest to study a system to understand the relationships between its components or to predict how the system will operate under a new policy.

Ideally . . .

- To study the system, it is sometimes possible to experiment with the system itself.
- However, this is not always possible.

Why not the Real System?

- Maybe the new system might not yet exist; it could be in only hypothetical form or at the design stage.
- E.g. Space flight, going to distance planets, rescue system, anything revolutionary etc.

Would Having an Existing System Work?

- Even if the system exists, it might be impractical to experiment with it.
- E.g.
- It may be too costly
- It may be impractical

Example

- Would it make sense to double the unemployment rate to discover the effect of employment on inflation?

- In the case of a bank, reducing the numbers of tellers to study the effect on the length of waiting lines might infuriate the customers so greedy that they move their accounts

Example: Discrete System:

- In case of a consulting job for the simulation of a redesigned port in western Australia.
- At \$200 millions for a loading/unloading berth, it's not advisable to invest that amount only to find that the berth is adequate or not.

Model: What it is?

- A model is defined as a representation of a system for the purpose of studying the system.
- For most studies, it is only necessary to consider those aspects of the system that affect the problem under investigation.
- These aspects are represented in a model of the system; the model, by definition, is a simplification of the system.
- On the other hand, the model should be sufficiently detailed to permit valid conclusions to be drawn about the real system

Variety of Models: Same System:

- A system may need to be described by several models
- Model is described similar to components.

Summary:

- We talked about discrete and continuous systems
- We also discussed some examples

Module – 13

Model Types

- In this module, we shall examine different types of models
- What are the different model types?

Key Types:

- Mathematical
- Physical

Mathematical:

A mathematical model uses symbolic notation and mathematical equations to represent a system.

What is a Simulation Model?

- A special type of Mathematical model
- Simulation models may be further classified as being static or dynamic, deterministic or stochastic, and discrete or continuous.

Monte Carlo Model

A static simulation model, sometimes called a Monte Carlo simulation, represents a system at a particular point in time.

Dynamic Models:

- Dynamic simulation models represent systems as they change over time
- E.g. Simulation of a Bank form 9 AM to 2PM

Deterministic Models:

- Simulation models that contain no random variables are classified as deterministic.
- Deterministic models have a known set of inputs, which will result in a unique set of outputs.
- Deterministic arrivals would occur at a dentist's office if all patients arrived at the scheduled appointment time.

Stochastic Models:

- A stochastic simulation model has one or more random variables as inputs.
- Random inputs lead to random outputs
- Random Outputs can be considered only as estimates of the true characteristics of a model.
- The simulation of a bank would usually involve random interarrival times and random service times.
- Thus, in a stochastic simulation, the output measures the average number of people waiting, the average waiting time of a customer
- Output must be treated as statistical estimates of the true characteristics of the system.

Discrete vs. Continuous Model Definition:

- Discrete and continuous models are defined in an analogous manner.
- However, a discrete simulation model is not always used to model a discrete system, nor is a continuous simulation model always used to model a continuous system.
- Tanks and pipes are modeled discretely by some software vendors, even though we know that fluid flow is continuous.

What if Both?

- In addition, simulation models may be mixed, both discrete and continuous. The choice of whether to use a discrete or continuous {or both discrete and continuous} simulation model is a function of
 - the characteristics of the system
 - the objective of the study.
- E.g. , a communication channel could be modeled discretely if the characteristics and movement of each message were deemed important.
- Conversely, if the flow of messages in aggregate over the channel were of importance, modeling the system via continuous simulation could be more appropriate.

Cont'd

We shall be studying discrete, dynamic, and stochastic models primarily.

Summary:

We have examined the basic types of models and seen how systems can be modeled.

Module – 14

Discrete Event Simulation:

In this module, we shall examine Discrete Event Simulation

- Why is Discrete Event Simulation important?
- What is it?

Key Types:

- Discrete-event systems simulation is the modeling of systems in which the state variable changes only at a discrete set of points in time.

- The simulation models are analyzed by numerical methods rather than by analytical methods.

What does it mean?

- Analytical methods employ the deductive reasoning of mathematics to "solve" the model
- For example, differential calculus can be used to compute the minimum-cost policy for some inventory models.

Numerical Methods:

- Numerical methods employ computational procedures to "solve" mathematical models.
- In the case of simulation models, which employ numerical methods, models are "run" rather than solved-that is
- An artificial history of the system is generated from the model assumptions,
- Observations are collected to be analyzed and to estimate the true system performance measures.

Characteristics of Real World Models:

- Real-world simulation models are rather large,
- The amount of data stored and manipulated is vast, so such runs are usually conducted with the aid of a computer.
- However, much insight can be obtained by simulating small models manually.

Summary:

- Our key focus is on discrete-event system simulation
- The models of interest are analyzed numerically, usually with the aid of a computer

Module – 15

Steps in a Simulation Study:

In this module, we shall examine steps in a Simulation study

- How is a Simulation study conducted?
- What are the steps?

Key Steps:

- Problem Formulation
- Setting of Objectives and Overall project plan
- Model Conceptualization
- Data Collection
- Model Translation
- Verification
- Validation
- Experimental Design
- Production Runs and Analysis
- More Runs?
- Documentation and Reporting
- Implementation

Problem Formulation:

- Every study should begin with a statement of the problem.
- If the statement is provided by the policymakers, or those that have the problem, the analyst must ensure that the problem being described is clearly understood.

Formulation:

- If a problem statement is being developed by the analyst, it is important that the policymakers understand and agree with the formulation.
- There are occasions where the problem must be reformulated as the study progresses.
- In many instances, policymakers and analysts are aware that there is a problem long before the nature of the problem is known.

Step 1:

Objectives and Project Plan:

- The objectives indicate the questions to be answered by simulation.
- Without the objectives, it makes little sense to develop a simulation model

Objectives:

- The first thing to verify is: Is Simulation the right tool for the problem at hand?
- This will be based on the actual problem statement

If Simulation is the Right Tool . . .

- Assuming that it is decided that simulation is appropriate, the overall project plan should include benchmark Systems evaluation
- i.e. a statement of alternative systems to be considered
- Also, a method for evaluating the effectiveness of these alternatives should be noted.

Project Planning:

- It should also include the plans for the study in terms of
- The number of people involved
- The cost of the study
- The number of days required to accomplish each phase of the work
- The results expected at the end of each stage.

Summary:

We have examined the importance of basic planning and project management.

Module – 16

Steps in a Simulation Study: Conceptualization!

Step 2: Model Conceptualization:

- The construction of a model of a system is probably as much art as science.
- Pritsker [1998] provides a lengthy discussion of this step.
- Although it is not possible to provide a set of instructions that will lead to building successful and appropriate models in every instance, there are some general guidelines that can be followed" [Morris, 1967].
- The art of modeling is enhanced by
 - An ability to abstract the essential features of a problem
 - To select and modify basic assumptions that characterize the system,
 - To enrich and elaborate the model until a useful approximation results.

- Thus, it is best to start with a simple model and build toward greater complexity.
- However, the model complexity need not exceed that required to accomplish the purposes for which the model is intended.
- Violation of this principle will only add to model-building and computer expenses.
- It is not necessary to have a one-to-one mapping between the
- model and the real system.
- Only the essence of the real system is needed.

User Involvement:

- It is advisable to involve the model user in model conceptualization.
- Involving the model user will both
 - enhance the quality of the resulting model
 - increase the confidence of the model user in the application of the model.
- While several models will be presented in the course – only real-world experience can “teach” the art of modeling

Summary:

Till now we have taken a look at the first 2 steps of a simulation study.

Module – 17

Steps in a Simulation Study: Data Collection and Model Translation

In this module, we shall continue to examine 2 key steps in a Simulation study

Data Collection:

- There is a constant interplay between the construction of the model and the collection of the needed input data [Shannon, 1975].
- As the complexity of the model changes, the required data elements can also change.
- Data collection takes a large portion of the total time required to perform a simulation

- It is therefore necessary to begin it as early as possible
- Data collection is usually conducted together with the early stages of model building.
- The objectives of the study dictate, in a large way, the kind of data to be collected.

Example:

- In the study of a bank, for example, if the desire is to learn about the length of waiting lines as the number of tellers change, the types of data needed would be
- the distributions of interarrival times (at different times of the day),
- The service-time distributions for the tellers,
- The historic distributions on the lengths of waiting lines under varying conditions.
- See also an excellent discussion by Henderson [2003].)

Model Translation:

- Most real-world systems result in models that require a great deal of information storage and computation,
- So the model must be entered into a computer-recognizable format
- We use the term "program" even though it is possible to accomplish the desired result in many instances with little or no actual coding.
- The modeler must decide whether to program the model in a simulation language
- GPSSIH, or to use special-purpose simulation software.
- For manufacturing and material handling, tools include Arena®, AutoMod™, Extend™, Flexsim, MicroSaint, ProModel®, Quest®, SIMUL8®, and WITNESS™
- Simulation languages are powerful and flexible.
- However, if the problem is amenable to solution with the simulation software, the model development time is greatly reduced.
- Furthermore, most of the simulation-software packages have added features that enhance their flexibility, although the amount of flexibility varies greatly.

Module – 18

Steps in a Simulation Study: Verification and Validation

Has the Model been Verified?

- Verification pertains to the computer program prepared for the simulation model.
- The basic question is “Is the computer program performing properly?”
- With complex models, it is difficult, if not impossible, to translate a model successfully in its entirety without a good deal of debugging

Testing Verification:

- If the input parameters and logical structure of the model are correctly represented in the computer, verification has been completed.

Validation:

- Validation is usually achieved through
 - the calibration of the model,
 - an iterative process of comparing the model against actual system behavior
 - using the discrepancies between the two, and the insights gained, to improve the model.
 - This process is repeated until model accuracy is judged acceptable
 - In the example of a bank, data was collected concerning the length of waiting lines under current conditions.
 - Does the simulation model replicate this system measure?
 - This is one means of validation.

Summary:

- Till now we have taken a look at the several steps of the simulation study.
- We have examined how verification and validation are performed.

Module – 19

Steps in a Simulation Study: Experimental Design

Experimental Design:

- In this module, we shall continue to examine steps in a Simulation study
- The alternatives that are to be simulated must be determined.
- Often, the decision concerning which alternatives to simulate will be a function of runs that have been completed and
- analyzed.
- For each system design that is simulated, decisions need to be made concerning
- the length of the initialization period,
- the length of simulation runs,
- the number of replications to be made for each run.

Production Runs and Analysis:

- Production runs, and their subsequent analysis, are used to estimate measures of performance for the system designs that are being simulated.
- There is also possibility to use software to help in this step

Example Software:

E.g. AutoStat (in AutoMod), OptQuest (in several pieces of simulation software), SimRunner (in Pro Model), WITNESS Optimizer (in WITNESS), behavior space in NetLogo etc.

More Runs?

- Given the analysis of runs that have been completed, the analyst determines whether additional runs are needed
- Also what design those additional experiments should follow.
- Simulation languages are powerful and flexible.
- However, if the problem is amenable to solution with the simulation software, the model development time is greatly reduced.
- Furthermore, most of the simulation-software packages have added features that enhance their flexibility, although the amount of flexibility varies greatly.

Summary:

We have seen how simulation steps can be simplified by means of specific software

Module – 20

Steps in a Simulation Study:

Documentation and Reporting:

- There are two types of documentation:
- Program
- Progress.

Documentation:

- Program documentation is necessary for numerous reasons.
- If the program is going to be used again by the same or different analysts, it could be necessary to understand how the program operates.

Documentation Benefits:

- This will create confidence in the program, so that model users and policymakers can make decisions based on the analysis.
- Also, if the program is to be modified by the same or a different analyst, this step can be greatly facilitated by adequate documentation.
- One experience with an inadequately documented program is usually enough to convince an analyst of the necessity of this important step.
- Another reason for documenting a program is so that model users can change parameters at will in an effort to learn the relationship in
 - input parameters and
 - output measures of performance
- or to discover the input parameters that optimize some output measure of performance.

Frequency of Progress Reports:

- Progress reports should be frequent (monthly, at least)
- They help even those not involved in the day to day operation to keep abreast
- The awareness of these others can often enhance the successful completion of the project by surfacing misunderstandings early, when a problem can be solved easily.

Summary:

- We have seen how progress reports can be used in a study

Module – 21

Lesson 21 : Steps in a sim. Study: Logs & Reports

Logs and Reports: Project reports give a chronology of work done and decisions made. This can prove to be of great value in keeping the project on course.

Musselman suggests frequent reports (monthly, at least) so that even those not involved in the day-to-day operation can be kept abreast. The awareness of these others can often enhance the successful completion of the project by surfacing misunderstandings early when the problem can be solved easily. Musselman also suggests maintaining a project log to provide a comprehensive record of accomplishments, change requests, key decisions, and other items of importance.

On the reporting side, Musselman suggests frequent deliverables. These may or may not be the results of major accomplishments. His maxim is that “it is better to work with many intermediate milestones than with one absolute deadline.” Possibilities before the final report include a model specification, prototype demonstrations, animations, training results, intermediate analyses, program documentation, progress reports, and presentations. He suggests that these deliverables should be timed judiciously over the life of the project.

The results of all the analyses should be reported clearly and concisely in a final report. This will allow the model users (now the decision-makers) to review the final formulation, the alternative systems that were addressed, the criteria by which the alternatives were compared, the results of the experiments, and the recommended solution(s) to the problem. Furthermore, if decisions have to be justified at a higher level, the final report should provide a vehicle of certification for the model user/decision-maker and add to the credibility of the model and the model-building process.

12. Implementation: The success of the implementation phase depends on how well the previous eleven steps have been performed. It is also contingent upon how thoroughly the analyst has involved the ultimate model user during the entire simulation process. If the model user has been involved during the entire model-building process and if the model user understands the nature of the model and its outputs, the likelihood of a vigorous implementation is enhanced [Pritsker, 1995].

Conversely, if the model and its underlying assumptions have not been properly communicated, implementation will probably suffer, regardless of the simulation model's validity.

It actually involves putting a plan, policy or design into action or we can say that installing a software is also called implementation.

Module – 22

Lesson 22 : Steps in a sim. Study: Phases Overview

Phase I

- Phase I consists of Problem Formulation and an Orientation/discovery Period
- The initial statement of the problem is usually quite "fuzzy," the initial objectives will usually have to be reset, and the original project plan will usually have to be fine-tuned.
- These recalibrations and clarifications could occur in this phase, or perhaps will occur after or during another phase (i.e. , the analyst might have to restart the process).

Phase II

- It involves model building and data collection
- This includes steps 3 (Model
- Conceptualization), 4 (Data Collection), 5 (Model Translation), 6 (Verification), and 7 (Validation).
- A continuing interplay is required among the steps.
- Exclusion of the model user during this phase can have dire implications at the time of implementation.
- Once we are done with the basic simulation experiments, it is imperative to involve the end user for exiting the case study.
- This is a continuing step.

Summary

- We have seen how typical case studies are structured in the form of different phases

Module – 23

Lesson 23 : Simulation Case Studies Basic steps

- In this module, we shall see an overview of various case studies

Variety of Examples

- We shall be taking a look at a couple of examples.
- The variety will include hands on as well as spreadsheet examples.
- Key objective: Take a look at Discrete Event Simulation and also descriptive statistics for prediction of system performance

General Steps

- Every simulation that we observe will be based on 3 key steps
- The idea is to examine the examples using a methodological approach

First Step

- Determine the characteristics of each of the inputs to the simulation.
- Quite often, these are modeled as probability distributions, either continuous or discrete.

Second Step

- Construct a simulation table.
- Each simulation table is different, for each is developed for the problem at hand.

Example: Simulation Table

- An example of a simulation table is shown in Table.
- In this example, there are p inputs, $x_{ij}, j=1, 2, \dots, p$,
- One response y_i for each of repetitions (or, trials)
- $i=1, 2, \dots, n$.
- Initialize the table by filling in the data for repetition 1

Table

Repetitions	Inputs						Response
	x_{i1}	x_{i2}	...	x_{ip}	...	x_{iq}	y_i
1							
2							
3							
...							
n							

Step 3

- For each repetition i , generate a value for each of the p inputs, and evaluate the function
- Calculate a value of the response y_i .
- The input values may be computed by sampling values from the distributions chosen in step 1
- A response typically depends on the inputs and one or more previous responses.
- Also, if the program is to be modified by the same or a different analyst, this step can be greatly facilitated by adequate documentation.
- One experience with an inadequately documented program is usually enough to convince an analyst of the necessity of this important step.

Overview of Examples

- Queueing
- Inventory
- Reliability
- Network analysis.
- The two queueing examples provide
 - single-server
 - two-server system

First Example Overview

- The first of the inventory examples involves a problem that has a closed-form solution;
- What it means is that the simulation solution can be compared to the mathematical solution.

Second Inventory Example

- The second inventory example pertains to the classic order-level model .

More examples

- Next, there is an example that introduces the concept of random normal numbers
- Also, there is a model for the simulation of lead-time demand.
- The examples conclude with the analysis of a network.

Summary

- In this module, we have examined how simulation examples give us a clear picture of how to do simulations
- We also took a look at the type of examples that we shall be examining in our next module

Module – 24

Lesson 24 : Basics of a Spreadsheet Simulation: Simulating Randomness

- In this module, we shall see an overview of spreadsheet simulation
- We shall also take a look at simulating randomness

How to Simulate Randomness

- Almost every simulation we encounter needs at least one random variable
- E.g. Coin Toss (Head vs. Tail)



Another Example: Queue

- Arrival and departure rate



Taking a Step Back: What is Random?

- In the real world, the number of factors involved in any event are so numerous that it can be very difficult to predict such events.
- In a computer, this can be very hard to generate.
- Hence the concept of “pseudo randomness”

General Steps

- Generate a number between 0 and 1 – referred to as the “random number”
- “Random variable” is any randomly generated quantity with a specified statistical distribution.

- A method of generating a sequence of random numbers is called a Random Number Generator (RNG)

Important Statistical Properties

- Number should be uniformly distributed between 0 and 1
- Subsequent numbers should be statistically independent of all previous numbers

VBA in MS Excel

- We use the RAND formula = RAND()
- More complex formulas can be used e.g. IF(RAND() <= 0.5, 0.1)
- Here IF checks the first condition and if true, returns the second one
- Another option is RANDBETWEEN()
- Rnd01
- Discrete Uniform(min, max)
- DiscreteEmp(rCumProb, rValues)
- Uniform(low, high)
- NORMSINV(Rnd01())
- For manual simulation, we can use coin toss or dice (Impractical for anything large however)

Summary

- In this module, we have examined the basics of simulation in spreadsheet

Module – 25

Lesson 25 : Simulating a Coin Toss

- In this module, we shall see how we can simulate a coin toss using Excel

The Problem

- We want to simulate a sequence of 10 coin tosses.
- Coin is a “fair” coin
- Fair means same chance of heads vs. tail



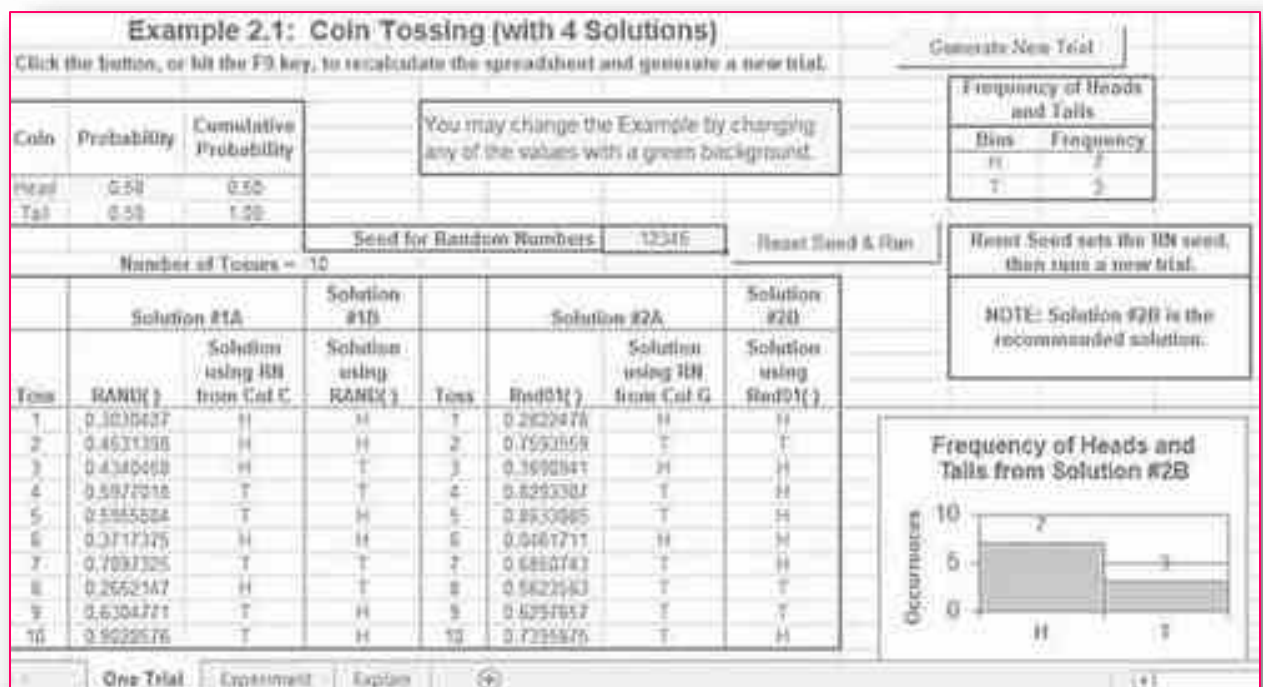
Type of Simulation: Monte Carlo Simulation

- There are no events
- No clock times being tracked

Description: Solutions

- The spreadsheet example has 4 solutions.
- Two use RAND() and other two use Rnd01 function

Simulation



Summary

- In this module, we have examined how we can use VBA to simulate a coin toss.

Module – 26

Lesson 26 : Simulating a Random Service Time

- In this module, we shall see how we can simulate a Random service time

The Problem

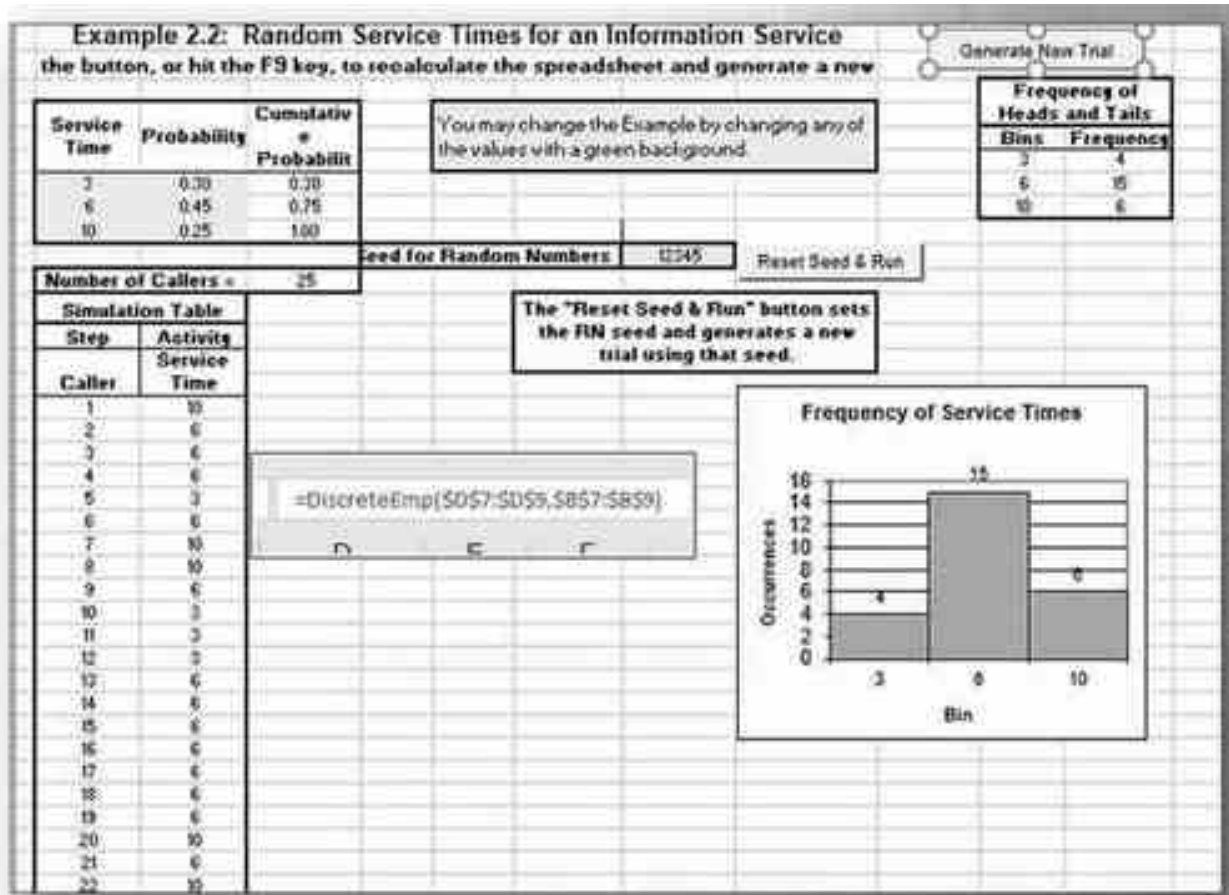
- We want to simulate a set of discrete values from continuous variables.
- Callers are assigned 3 different times with different probabilities.
- 3 with a probability of 30%
- 6 with a probability of 45%
- 10 with a probability of 25%

Type of Simulation: Monte Carlo Simulation

- There are no events
- No clock times being tracked

Description: Solution

- The spreadsheet example converts the continuous number to discrete using cumulative probability



Summary

- In this module, we have examined how we can use Excel VBA functions to simulate a discrete distribution.
- The key idea is to assign limits based on cumulative probability for each value

Module – 27

Lesson 27 : Simulating a Random Arrival Time

- In this module, we shall see how we can simulate a Random Arrival time

The Problem

- We want to simulate a random interarrival time (Time between successive arrivals) and compute arrival times of customers or other entities

Interarrival Time

- Interarrival times and service times are examples of simulation-generated time durations
- These are called activity times or activities

Simulation Clock

- Simulation clock is a key component in a dynamic discrete-event simulation.
- Clock times represent the time of occurrence of an event
- E.g. arrival, service beginning, completion etc.

Simulating a Random Arrival Time

- Telephone calls to a Telephone Info service occur at random times
- These are defined by a discrete distribution with values 1, 2, 3, or 4 min – all equal probability.
- Goal is to generate both interarrival times and arrival times
- Dynamic here means time-based – with system state changing over time
- Dynamic event-based means model tracks progression of event occurrences over time

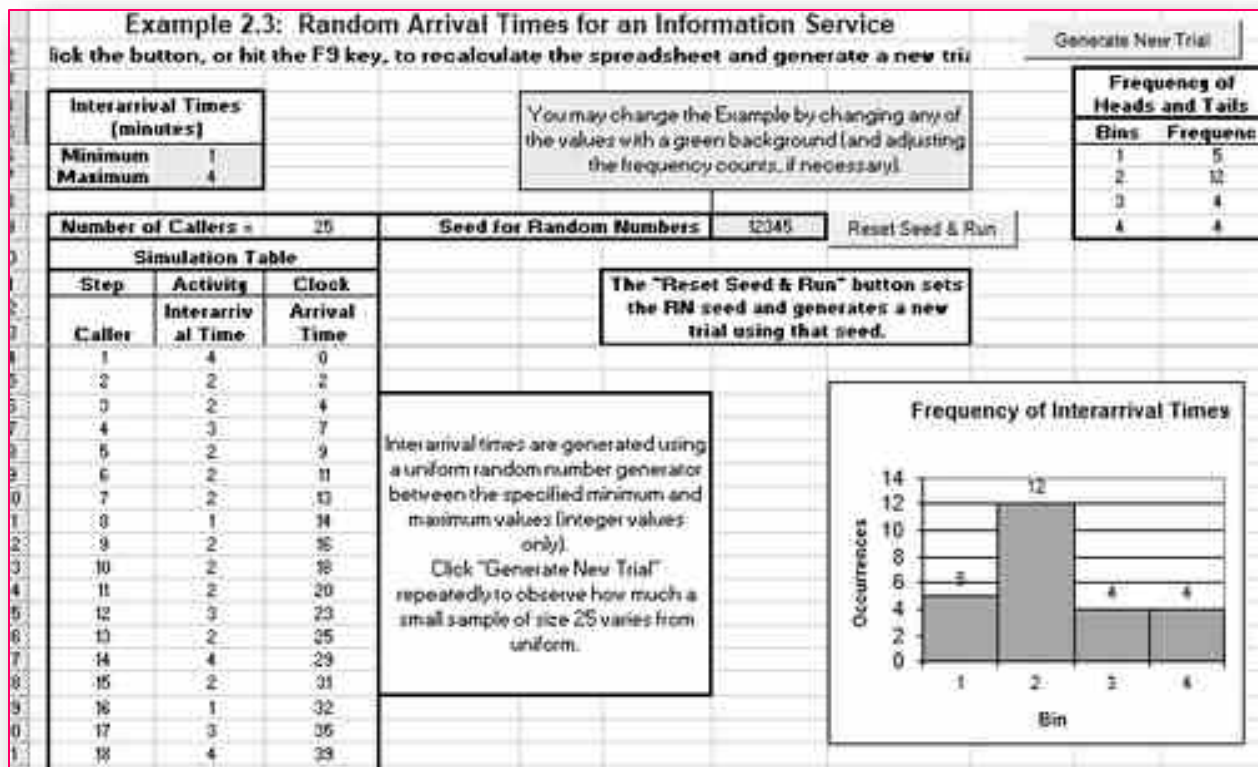
Simulation Details

- Simulation is based on the “Uniform” discrete distribution
- Uniform means all events have equal probabilities
- VBA Function Discrete Uniform()
- The function takes 2 arguments – minimum and maximum values

Simulation Clock

- Simulation clock is a key component in a dynamic discrete-event simulation.
- Clock times represent the time of occurrence of an event
- E.g. arrival, service beginning, completion etc.

Example2.3ArrivalTimes.xls



Summary

- In this module, we have examined how we can use Excel VBA functions to simulate events such as interarrival times and arrival times.
- The key idea is to use a discrete uniform distribution

Module – 28

Lesson 28 : Framework for Spreadsheet Simulation

- In this module, we shall examine a framework for Spreadsheet simulation
- We have seen several examples of simulation.
- Can we convert those examples into a framework?

Definitions

- First of all, model developer must define:

- Model Inputs
- System States
- Model outputs

What are Inputs

- Inputs are the exogenous variables that are (usually) defined independently of other system characteristics
- E.g. Probability of heads in a coin toss
- Service times
- Interarrival time distribution in a queuing system
- Demand distribution in an inventory model

Implications of Input Variables

- These input specifications are useful
- They can help generate:
 - Random activity times and other random variables
 - Whereas other inputs may be constants
 - E.g. policy or decision parameters

Output variables

- They compute measures of system performance (also known as model responses)
- E.g. an output could be:
 - An individual customer's waiting time (in a queue)
 - The cost of an individual transaction in an inventory system

Associated Responses

- Associated responses could be:
 - Average wait time in a queue
 - Average cost per unit time in an inventory system

Summary

- In this module, we have examined a generic framework for developing simulation models (spreadsheet-based)
- It essentially follows a step-by-step procedure

Module – 29

Framework: Simulation Table Design

- Each sim table is custom designed for the problem at hand.
- Each column can have various types

Column Types

- An activity time associated with the input
- Any other random variable defined by an input
- A system state
- An event or clock time of an event
- Model output
- Sometimes, a model response

General Guidelines: Input

- Input Characteristics
- Probability distributions
- Constants

General Guidelines: Activities

- Activities
- Events
- System states
- Sorting the relevant from the irrelevant

General Guidelines: Model Responses

- Summary performance measures
- Correlate with objectives and specific questions

General Guidelines: Model Outputs

- Determine the outputs needed to compute the model response

General Guidelines: Simulation Table

- Construct the simulation table

Repetitions	Inputs						Response
	x_{i1}	x_{i2}	...	x_{ij}	...	x_{ip}	y_i
1							
2							
3							
.							
.							
.							
n							

- For each step i , generate a value for each activity.
- Next, calculate the system states
- Finally calculate the outputs
- In the end, use the outputs to compute the model responses or measures of performance

Summary

- In this module, we have examined a generic framework for developing simulation models (spreadsheet-based)
- It essentially follows a step-by-step procedure

Module – 30

Framework Using Discrete-Event

Simulation: General Idea

General Principles

- This Module develops a common framework for the modeling of complex systems by using discrete-event simulation
 - It covers the basic building blocks of all discrete-event simulation models: entities and attributes, activities, and events

Discrete-event Simulation

- In Discrete-event Simulation,
 - A system is modeled in terms of its state at each point in time; of the entities that pass through the system
 - And the entities that represent system resources; and of the activities and events that cause the system state to change.
- Discrete-event models are appropriate for those systems for which changes in system state occur only at discrete points in time

Summary

- This Module introduces and explains the fundamental concepts and methodologies underlying all discrete-event simulation packages.

Module – 31

Framework Using Discrete-event

Simulation-I

Concepts In Discrete-event Simulation

- This section expands on these concepts and proposes a framework for the development of a discrete-event model of a system.
- The major concepts are illustrated by examples

- The major concepts are briefly defined
 - **System** : A collection of entities (e.g., people and machines) that interact together over time to accomplish one or more goals.
 - **Model**: An abstract representation of a system, usually containing structural, logical, or mathematical relationships that describe a system in terms of state, entities and their attributes, sets, processes, events, activities, and delays.
 - **System state**: A collection of variables that contain all the information necessary to describe the system at any time.
 - **Entity**: Any object or component in the system that requires explicit representation in the model (e.g., a server, a customer, a machine).
 - **Attributes**: The properties of a given entity (e.g., the priority of a waiting customer, the routing of a job through a job shop).
 - **List**: A collection of (permanently or temporarily) associated entities, ordered in some logical fashion (such as all customers currently in a waiting line, ordered by "first come, first served," or by priority).
 - **Event notice**: A record of an event to occur at the current or some future time, along with any associated data necessary to execute the event; at a minimum, the record includes the event type and the event time.
 - **Event list**: A list of event notices for future events, ordered by time of occurrence; also known as the future event list (FEL).

Summary

- Here we have discussed some basic concepts related to a framework for DES

Framework For Discrete-event

Simulation-II

Concepts In Discrete-event Simulation

- This section continues to expand on the framework concepts
 - **Activity:** A duration of time of specified length (e.g., a service time or interarrival time), which is known when it begins (although it may be defined in terms of a statistical distribution).
 - **Delay:** A duration of time of unspecified indefinite length, which is not known until it ends (e.g., a customer's delay in a last-in-first-out waiting line which, when it begins, depends on future arrivals).
 - **Clock:** A variable representing simulated time, called CLOCK in the examples to follow

Different Simulation, Packages Use Different Terminology

- Different simulation, packages use different terminology for the same or similar concepts—for example, lists are sometimes called sets, queues, or chains. Sets or lists are used to hold both entities and event notices

The Entities On A List Are Always Ordered By Some Rule

- The entities on a list are always ordered by some rule, such as first-in-first-out or last-in-first-out, or are ranked by some entity attribute, such as priority or due date. The future event list is always ranked by the event time recorded in the event notice

An Activity Typically Represents A Service Time

- An activity typically represents a service time, an interarrival time, or any other processing time whose duration has been characterized and defined by the modeler. An activity's duration may be specified in a number of ways:

Activity's Duration

- **1. Deterministic:** For example, always exactly 5 minutes;
- **2. Statistical:** For example, as a random draw from among 2, 5, 7 with equal probabilities;

- 3. A function depending on system variables and/or entity attributes—for example, loading time for an iron ore ship as a function of the ship's allowed cargo weight and the loading rate in tons per hour.

Duration Of An Activity is Computable

- However it is characterized, the duration of an activity is computable from its specification at the instant it begins. Its duration is not affected by the occurrence of other events (unless, as is allowed by some simulation packages, the model contains logic to cancel an event)

Summary

- Here we have given some additional concepts related to DES

Module – 33

Framework For Discrete-event

Simulation-III

Concepts In Discrete-event Simulation

- This section discusses keeping track of activities

To Keep Track Of Activities

- For example, if the current simulated time is $CLOCK = 100$ minutes and an inspection time of exactly 5 minutes is just beginning, then an event notice is created that specifies the type of event (an end-of-inspection event), and the event time ($100 + 5 = 105$ minutes)

Conditional Wait

- A delay is sometimes called a conditional wait, an activity an unconditional wait.
- The completion of an activity is an event, often called a primary event, that is managed by placing an event notice on the FEL.

Conditional Or Secondary Event

- The completion of a delay is sometimes called a conditional or secondary event, but such events are not represented by event notices, nor do they appear on the FEL.

Systems Considered Here Are Dynamic

- The systems considered here are dynamic, that is, changing over time. Therefore, system state, entity attributes and the number of active entities

- The contents of sets, and the activities and delays currently in progress are all functions of time and are constantly changing over time. Time itself is represented by a variable called CLOCK

Example 3.1: Call Center, Revisited

- Consider the Able-Baker call center system of Example 2.2. A discrete-event model has the following components:
- System state

– $L_q(t)$: the number of callers waiting to be served at time t ;

System State

– $L_a(t)$: 0 or 1 to indicate Able as being idle or busy at time t ;

– $L_b(t)$: 0 or 1 to indicate Baker as being idle or busy at time t .

Entities

- Entities:
 - Neither the callers nor the servers need to be explicitly represented, except in terms of the state variables, unless certain caller averages are desired

Events

Events:

- Arrival event;
- Service completion by Able;
- Service completion by Baker.

Summary

- This Module deals exclusively with dynamic, stochastic systems (i.e., involving time and containing random elements) that change in a discrete manner

Module – 34

Framework For Discrete-event

Simulation-IV

Activities

- **Activities**
 - Interarrival time, defined in Table 2.11;
 - Service time by Able, defined in Table 2.12;
 - Service time by Baker, defined in Table 2.13.

Delay

- **Delay:**
 - A caller's wait in queue until Able or Baker becomes free .
- The definition of the model components provides a static description of the model.

Prototype System Snapshot At Simulation Time t

<i>Clock</i>	<i>System state</i>	<i>Entities and attributes</i>	<i>Set 1</i>	<i>Set 2</i>	<i>...</i>	<i>Future event list, FEL</i>	<i>Cumulative statistics and counters</i>
t	$(x, y, z, \dots)$					$(3, t_1)$ – Type 3 event to occur at time t_1 $(1, t_2)$ – Type 1 event to occur at time t_2 . . .	

Prototype system snapshot at simulation time t .

Some Questions That Need Answers Include

- How does each event affect system state, entity attributes, and set contents?

- How are activities defined (i.e., deterministic, probabilistic, or some other mathematical equation)?
- What event marks the beginning or end of each activity?
- What is the system state at time 0? What events should be generated at time 0 to prime the model that is, to get the simulation started?
- Which events trigger the beginning (and end) of each type of delay? Under what conditions does a delay begin or end?

Discrete-event Simulation

- A discrete-event simulation is the modeling over time of a system all of whose state changes occur at discrete points in time-those points when an event occurs .
- A discrete-event simulation (hereafter called a simulation) proceeds by producing a sequence of system snapshots (or system Images) that represent the evolution of the system through time

Summary

- This Module presents further concepts of DES

Module – 35

The Event Scheduling/Time Advance Algorithm

- The mechanism for advancing simulation time and guaranteeing that all events occur in correct chronological order is based on the future event list (FEL)
- This list contains all event notices for events that have been scheduled to occur at a future time.

Scheduling A Future Event

- **Scheduling a future event means that,**
 - At the instant an activity begins, its duration is computed or drawn as a sample from a statistical distribution;
 - And that the end-activity event, together with its event time, is placed on the future event list

Random Breakdowns Or Random Arrivals

- In the real world, most future events are not scheduled but merely happen

- such as random breakdowns or random arrivals. In the model, such random events are represented by the end of some activity, which in turn is represented by a statistical distribution

FEL Is Ordered By Event Time

- At any given time t , the FEL contains all previously scheduled future events and their associated event times (called t_1, t_2)
- The FEL is ordered by event time,

Imminent Event

- meaning that the events are arranged chronologically-that is, the event times satisfy

$$t < t_1 \leq t_2 \leq t_3 \leq \dots t_n$$

- Time t is the value:
 - of CLOCK, the current value of simulated time. The event associated with time t_1 is called the imminent event
 - Imminent event; that is, it is the next event that will occur
- After the system snapshot at simulation time $\text{CLOCK} = t$ has been updated, the CLOCK is advanced to simulation time $\text{CLOCK} = t_1$ the imminent event notice is removed from the FEL, and the event is executed

Execution Of The Imminent Event

- Execution of the imminent event means that a new system snapshot for time t_1 is created, one based on the old snapshot at time t and the nature of the imminent event
- At time t_1 , new future events may or might not be generated, but if any are, they are scheduled by creating event notices and putting them into their proper position on the FEL
- After the new system snapshot for time t_1 has been updated, the clock is advanced to the time of the new imminent event and that event is executed. This process repeats until the simulation is over.

Summary

- To summarize the discussion, when using the *Event-scheduling Approach*, a simulation analyst concentrates on events and their effect on system state

Module – 36

The Event Scheduling Time Advance Algorithm-II

- The sequence of actions that a simulator (or simulation language) must perform to advance the clock and build a new system snapshot is called the *Event-scheduling/Time-advance Algorithm*, whose steps are listed in Advancing simulation time and updating system image

Old System Snapshot At Time t

Old system snapshot at time t

<i>CLOCK</i>	<i>System state</i>	<i>...</i>	<i>Future event list</i>	<i>...</i>
t	(5, 1, 6)		$(3, t_1)$ – Type 3 event to occur at time t_1 $(1, t_2)$ – Type 1 event to occur at time t_2 $(1, t_3)$ – Type 1 event to occur at time t_3 $\cdot$ $\cdot$ $\cdot$ $\cdot$ $\cdot$ $\cdot$ $\cdot$ $\cdot$ $\cdot$ $(2, t_n)$ – Type 2 event to occur at time t_n	

Event-scheduling/time-advance algorithm

- Step 1. Remove the event notice for the imminent event (event 3, time t_1) from FEL.*
- Step 2. Advance CLOCK to imminent event time (i.e., advance CLOCK from t to t_1).*
- Step 3. Execute imminent event: update system state, change entity attributes, and set membership as needed.*
- Step 4. Generate future events (if necessary) and place their event notices on FEL, ranked by event time. (Example: Event 4 to occur at time t^* , where $t_2 < t^* < t_3$.)*
- Step 5. Update cumulative statistics and counters.*

Advancing simulation time and updating system image

New System Snapshot At Time t_1

New system snapshot at time t_1

CLOCK	System state	...	Future event list	...
t_1	(5, 1, 5)		$(1, t_2)$ – Type 1 event to occur at time t_2 $(4, t^*)$ – Type 4 event to occur at time t^* $(1, t_3)$ – Type 1 event to occur at time t_3 $(2, t_n)$ – Type 2 event to occur at time t_n	

Advancing simulation time and updating system image

List Processing

- The length and contents of the FEL are constantly changing as the simulation progresses, and thus its efficient management in a computerized simulation will have a major impact on the efficiency of the computer program representing the model. The management of a list is called **list processing**

Cancellation Of An Event

- The major list processing operations performed on a FEL are removal of the imminent event, addition of a new event to the list, and occasionally removal of some event (called **Cancellation of an Event**). As the imminent event is usually at the top of the list, its removal is as efficient as possible.

Removal And Addition Of Events From The FEL

- The removal and addition of events from the FEL is illustrated in Advancing simulation time and updating system image.
 - Event 3 with event time t_1 represents, say, a service completion event at server 3

- Since it is the imminent event at time t , it is removed from the FEL in step 1 of the *event-scheduling/time-advance algorithm*:
- When event 4 (say, an *Arrival Event*) with event time t^* is generated at step 4, one possible way to determine its correct position on the FEL is to conduct a *top-down* search:

If $t^* < t_2$,	place event 4 at the top of the FEL.
If $t_2 \leq t^* < t_3$,	place event 4 second on the list.
If $t_3 \leq t^* < t_4$,	place event 4 third on the list.
$\vdots$	$\vdots$
If $t_n \leq t^*$,	place event 4 last on the list.

Least Efficient Way To Maintain The FEL

- The least efficient way to maintain the FEL is to leave it as an unordered list (additions placed arbitrarily at the top or bottom), which would require at step 1 (of a Figure) complete search of the list for the imminent event before each clock advance

Summary

- To summarize the discussion, when using the *Event-scheduling Approach*, a simulation analyst concentrates on events and their effect on system state

Module – 37

The Event Scheduling Time Advance Algorithm-III

Exogenous Events

- If $t = 0$, then the state (5, 1, 6) might represent the initial number of customers at three different points in the system
- An *Exogenous Event* is a happening "*outside the system*" that impinges on the system
- An important example is an arrival to a queueing system
- At time 0, the first arrival event is generated and is scheduled on the FEL (meaning that its event notice is placed on the FEL).
- The interarrival time is an example of an activity

- When the clock eventually is advanced to the time of this first arrival, a second arrival event is generated. First, an interarrival time is generated, a^* ;
- It is added to the current time, $CLOCK = t$; the resulting (future) event time, $t + a^* = t^*$, is used to position the new arrival event notice on the FEL

Bootstrapping

- This method of generating an external arrival stream is called bootstrapping ; it provides one example of how future events are generated in step 4 of the event scheduling time-advance algorithm.

The Event Scheduling Time Advance Algorithm II

- The method of generating an external arrival stream is called bootstrapping
- Bootstrapping is illustrated in Figure given. The first three interarrival times generated are 3.7, 0.4, and 3.3 time units.
- The end of an interarrival interval is an example of a primary event.
- A second example of how future events are generated is provided by a service completion event in a queueing simulation

Service Completion

- When one customer completes service, at current time $CLOCK = t$, if the next customer is present, then a new service time, s^* , will be generated for the next customer.
- The next Service completion event will be scheduled to occur at future time $t^* = t + s^*$, by placing onto the FEL a new event notice, of type service completion, with event time t^*

Summary

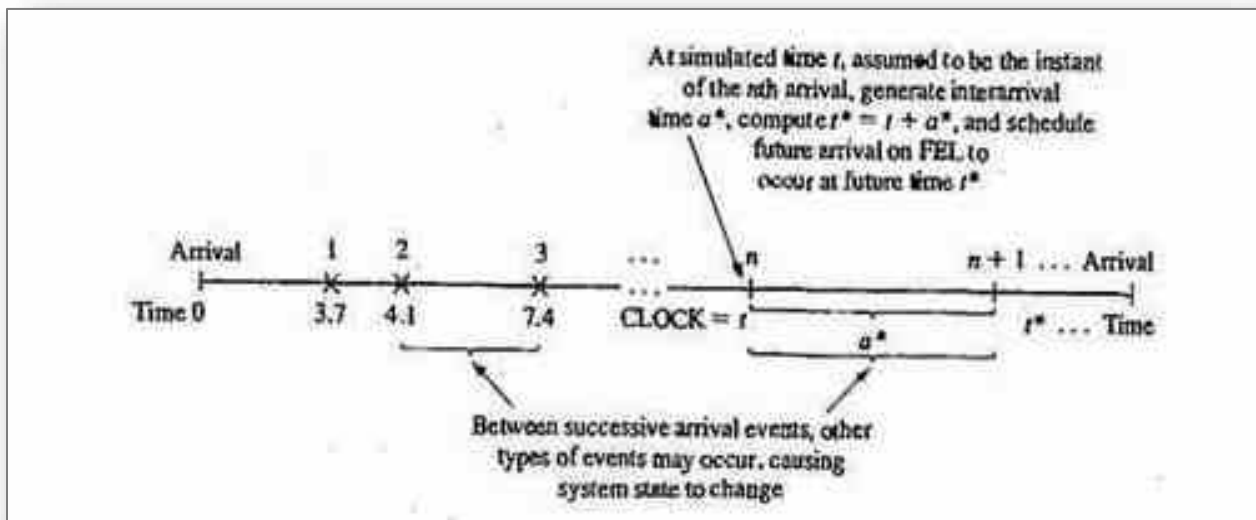
- To summarize the discussion, when using the *Event-scheduling Approach*, a simulation analyst concentrates on events and their effect on system state

The Event Scheduling Time Advance Algorithm-IV

Service Completion

- Beginning service is a conditional event because its occurrence is triggered only on the condition that a customer is present and a server is free
- Service completion is an example of a primary event

Generation Of An External Arrival



Generation of an external arrival stream by bootstrapping.

Runtimes And Downtimes

- A third important example is the alternate generation of runtimes and downtimes for a machine subject to breakdowns.
- At time 0, the first runtime will be generated, and an end-of-runtime event will be scheduled.

Stopping Event, E

- Every Simulation must have a stopping event, here called E, which defines how long the simulation will run.
- There are generally two ways to stop a simulation
- At time 0, schedule a stop simulation event at a specified future time TE'.
 - Thus, before simulating, it is known that simulation will run over the time interval[0, TE]. Example: Simulate a job shop for TE -40 hours.
- Run length TE is determined by the simulation itself.
- Generally, TE is the time of occurrence of some
- specified event E. Examples: T E is the time of the 100th service completion at a certain service center
- TE is the time of break down of a complex system. TE is the time of disengagement or total kill (whichever _ occurs first) in a combat simulation. TE is the time at which a distribution center ships the last carton in a day's orders.

Summary

- To summarize the discussion, when using the *Event-scheduling Approach*, a simulation analyst concentrates on events and their effect on system state

Module – 39

World Views-I

- When using a simulation package or even when doing a manual simulation, a modeler adopts a world view or orientation on for developing a model.

The Most Prevalent World View

- The most prevalent world views are
 - The event-scheduling world view,
 - The process-interaction world view,
 - The activity-scanning world view

Simulation Model In Terms of Analyst

- When using a package that supports the process-interaction approach, a simulation analyst thinks in terms of processes.
- The analyst defines the simulation model in terms of :

- Entities or objects
- And their life cycle as they flow through the system, demanding resources and queueing to wait for resources

Life Cycle

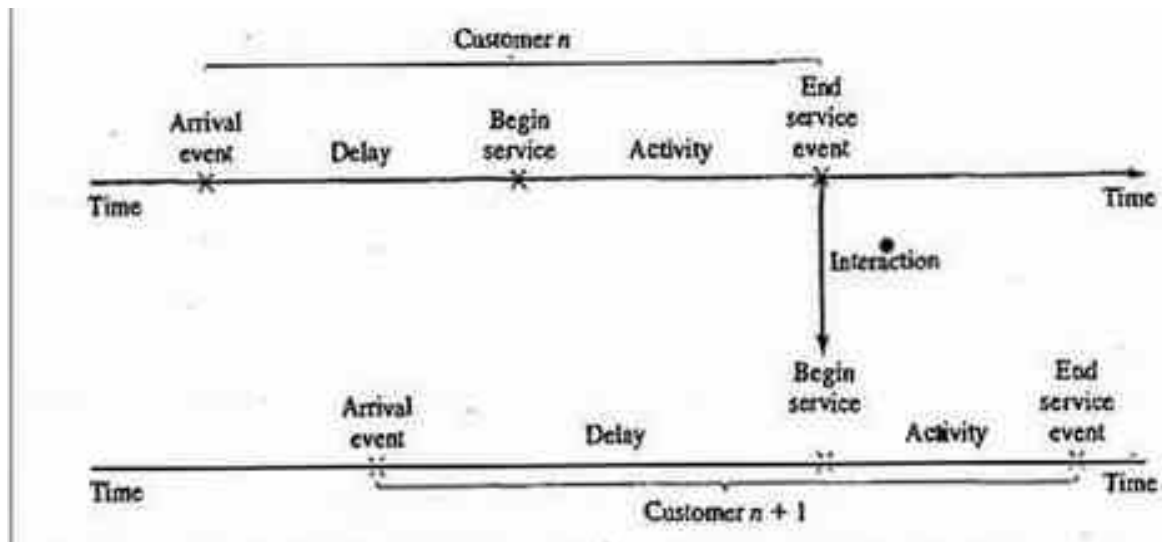
- A process is the life cycle of one entity.
- This life cycle consists of various events and activities.
- Some activities might require the use of one or more resources whose capacities are limited.

End Service Event

- In more precise terms, a process is a time-sequenced list of
 - Events
 - Activities
 - Delays
- Including demands for resources, that define the life cycle of one entity as it moves through a system

Customer

- ProcessAn example of a "customer process" is shown in Following Figure.
- In this figure, we see the interaction between two customer processes as customer $n + 1$ is delayed until the previous customer's "end service event" occurs.



Two interacting customer processes in a single-server queue

Activity-scanning Approach

- The pure activity-scanning approach has been modified (and made conceptually somewhat more complex) by what is called the three-phase approach

Summary

- In this module we learnt Proponents claim that the activity scanning and three-phase approaches are particularly good at handling complex resource problems in which various combinations of resources are needed to accomplish different tasks.
- These approaches guarantee that all resources being freed at a given simulated time will all be freed before any available resources are reallocated to new tasks.

World Views-II

Three-phase Approach

- The three-phase approach, which combines some of the features of event scheduling with activity scanning to allow for variable time advance and the avoidance of scanning when it is not necessary, but keeps the main advantages of the activity-scanning approach.

Duration Zero Time Units

- In the three-phase approach, events are considered to be activities of duration zero time units. With this definition, activities are divided into two categories, which are called B and C.

B activities & C activities

- **B activities:**
 - Activities bound to occur; all primary events and unconditional activities.
- **C activities:**
 - Activities or events that are conditional upon certain conditions-being true

B-type Activities

- The B-type activities and events can be scheduled ahead of time, just as in the event-scheduling approach.
- This allows variable time advance.
- The FEL contains only B-type events. Scanning to learn

C-type Activities

- C-type activities can begin or C-type events occur happens only at the end of each time advance, after all B-type events have completed.

Three-phase Approach

- In summary, with the three-phase approach, the simulation proceeds with repeated execution of the 3 phases until it is completed;
- Phase A :

- Remove the imminent event from the FEL and advance the clock to its event time.
 - Remove from the FEL any other events that have the same event time.
- Phase B:
 - Execute all B-type events that were removed from the FEL. (This could free a number of resources or otherwise change system state.)
- Phase C :
 - Scan the conditions that trigger each C-type activity and activate any whose conditions are Smet.
 - Rescan until no additional C-type activities can begin and no events occur.

Summary

- In this module we learnt Proponents claim that the activity scanning and three-phase approaches are particularly good at handling complex resource problems in which various combinations of resources are needed to accomplish different tasks.
- These approaches guarantee that all resources being freed at a given simulated time will all be freed before any available resources are reallocated to new tasks.

Module – 41

Manual Simulation Using Event Scheduling

- In the conducting of an event-scheduling simulation, a simulation table is used to record the successive system snapshots as time advances.

Single-Channel Queue

- Example 3 3: Single-Channel Queue
- Reconsider the grocery store with one checkout counter that was simulated in (previous example) by an ad hoc method.
- The system consists of those customers in the waiting line plus the one (if any) checking out.
- A stopping time of 60 minutes is set for this example. The model has the following components:

Model's Component

System state $(LQ(t), LS(t))$, where $LQ(t)$ is the number of customers in the waiting line, and $LS(t)$ is the number being served (0 or 1) at time t .

Entities The server and customers are not explicitly modeled, except in terms of the state variables.

Events

- Arrival (A);
- Departure (D);
- Stopping event (E), scheduled to occur at time 60.

Event notices

- (A, t) , representing an arrival event to occur at future time t ;
- (D, t) , representing a customer departure at future time t ;
- $(E, 60)$, representing the simulation stop event at future time 60.

Activities

- Interarrival time, defined in Table 2.6;
- Service time, defined in Table 2.7

Delay Customer time spent in waiting line.

Event Type, Event Time

- The event notices are written as (event type, event time).
- In this model, the FEL will always contain either two or three event notices.
- The effect of the arrival and departure events was first shown in previous Figures and is shown in more detail in next Figures .

Simulation Table

- The simulation table for the checkout counter is given in Table 3.1.
 - The reader should cover all system snapshots except one, starting with the first,
 - and attempt to construct the next snapshot from the previous one and the event logic in figure below and the interval time and service time will be identical to those used in table below:

Interarrival Times	1	1	6	3	7	5	2	4	1...
Service Times	4	2	5	4	1	5	4	1	4...

The simulation table for the checkout counter

Summary

- Here we have examined the basics of manual simulation

Execution Of The Arrival Event

- Initial conditions are that the first customer arrive at time 0 and begin service. This is reflected in table 3.1

Module – 42

Manual Simulation Using Event Scheduling-II

Execution Of The Arrival Event

- Initial conditions are that the first customer arrive at time 0 and begin service. This is reflected in table 3.1

Simulation Table for Checkout Counter

Table 3.1 Simulation Table for Checkout Counter (Example 3.3)

Clock	System State		Future Event List	Comment	Cumulative Statistics	
	LQ(t)	LS(t)			B	MQ
0	0	1	(A, 1) (D, 4) (E, 60)	First A occurs ($a^* = 1$) Schedule next A ($s^* = 4$) Schedule first D	0	0
1	1	1	(A, 2) (D, 4) (E, 60)	Second A occurs: (A, 1) ($a^* = 1$) Schedule next A (Customer delayed)	1	1
2	2	1	(D, 4) (A, 8) (E, 60)	Third A occurs: (A, 2) ($a^* = 6$) Schedule next A (Two customers delayed)	2	2
4	1	1	(D, 6) (A, 8) (E, 60)	First D occurs: (D, 4) ($s^* = 2$) Schedule next D (Customer delayed)	4	2
6	0	1	(A, 8) (D, 11) (E, 60)	Second D occurs: (D, 6) ($s^* = 5$) Schedule next D	6	2

(continued overleaf)

Simulation Table for Checkout Counter

Table 3.1 (continued)

Clock	System State		Future Event List	Comment	Cumulative Statistics	
	LQ(t)	LS(t)			B	MQ
8	1	1	(D, 11) (A, 11) (E, 60)	Fourth A occurs: (A, 8) ($\alpha^* = 3$) Schedule next A (Customer delayed)	8	2
11	1	1	(D, 15) (A, 18) (E, 60)	Fifth A occurs: (A, 11) ($\alpha^* = 7$) Schedule next A Third D occurs: (D, 11) ($\lambda^* = 4$) Schedule next D Customer delayed	11	2
15	0	1	(D, 16) (A, 18) (E, 60)	Fourth D occurs: (D, 15) ($\lambda^* = 1$) Schedule next D	15	2
16	0	0	(A, 18) (E, 60)	Fifth D occurs: (D, 16)	16	2
18	0	1	(D, 23) (A, 23) (E, 60)	Sixth A occurs ($\alpha^* = 5$) Schedule next A ($\lambda^* = 5$) Schedule next D	16	2
23	0	1	(A, 25) (D, 27) (E, 60)	Seventh A occurs: (A, 23) ($\alpha^* = 2$) Schedule next Arrival Sixth D occurs: (D, 23)	21	2

Execution Of The Arrival Event

- b the s stem snap shot at time zero (CLOCK = 0), with LQ = 0) = 0, LS(0) = 1, and both a departure event and arrival event on the FEL
- The simulation is scheduled to stop at time 60. Only two statistics, server utilization and maximum queue length, will be collected

Server Utilization

- Server utilization is defined by total server busy time (B) divided by total time (Te).
- Total busy time B, and maximum queue length, MQ will be accumulated as the simulation progresses

- A column headed “comments” is included to aid the reader
- (a^* and s^* are the generated interarrival and service times, respectively.)
- As soon as the system snapshot at times $CLOCK = 0$ is complete, the simulation begins. At time 0, the imminent event is (A,1).
- The $CLOCK$ is advanced to time 1, and (A, 1) is removed from the FEL

Execution of the arrival event

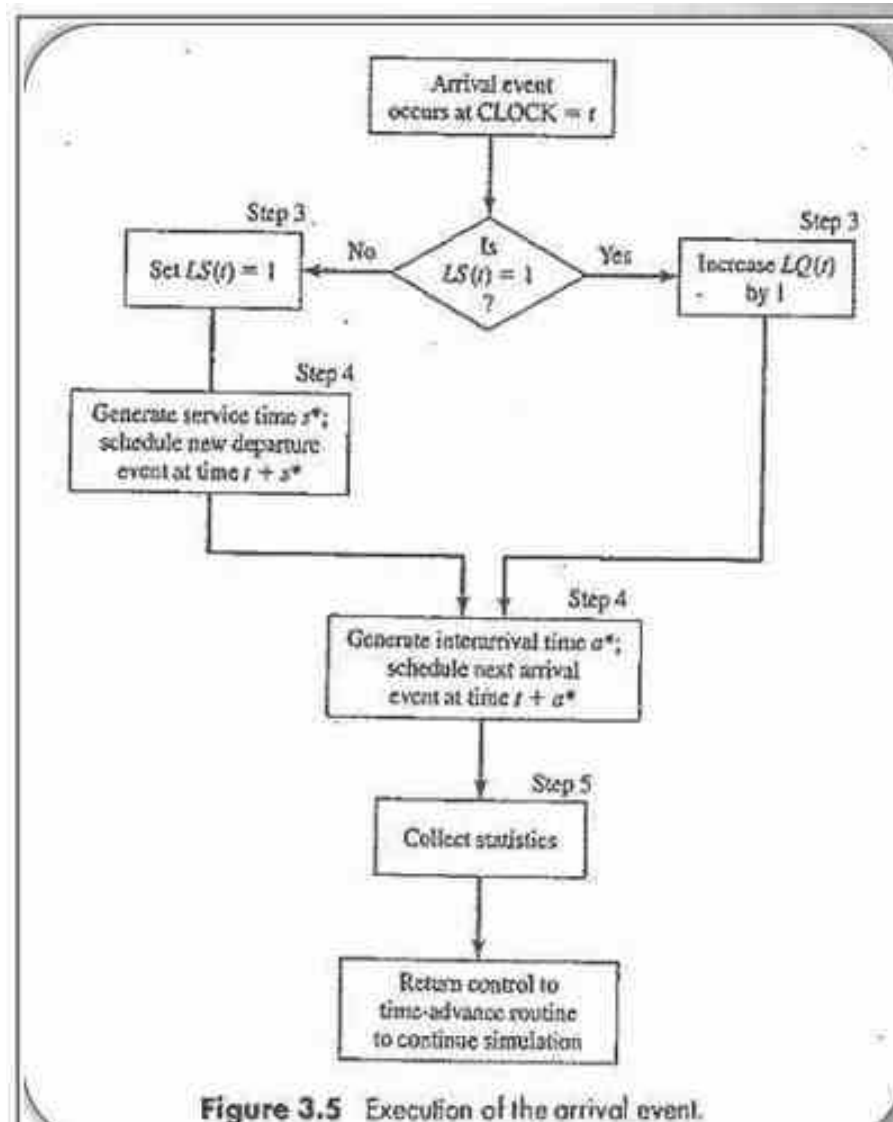
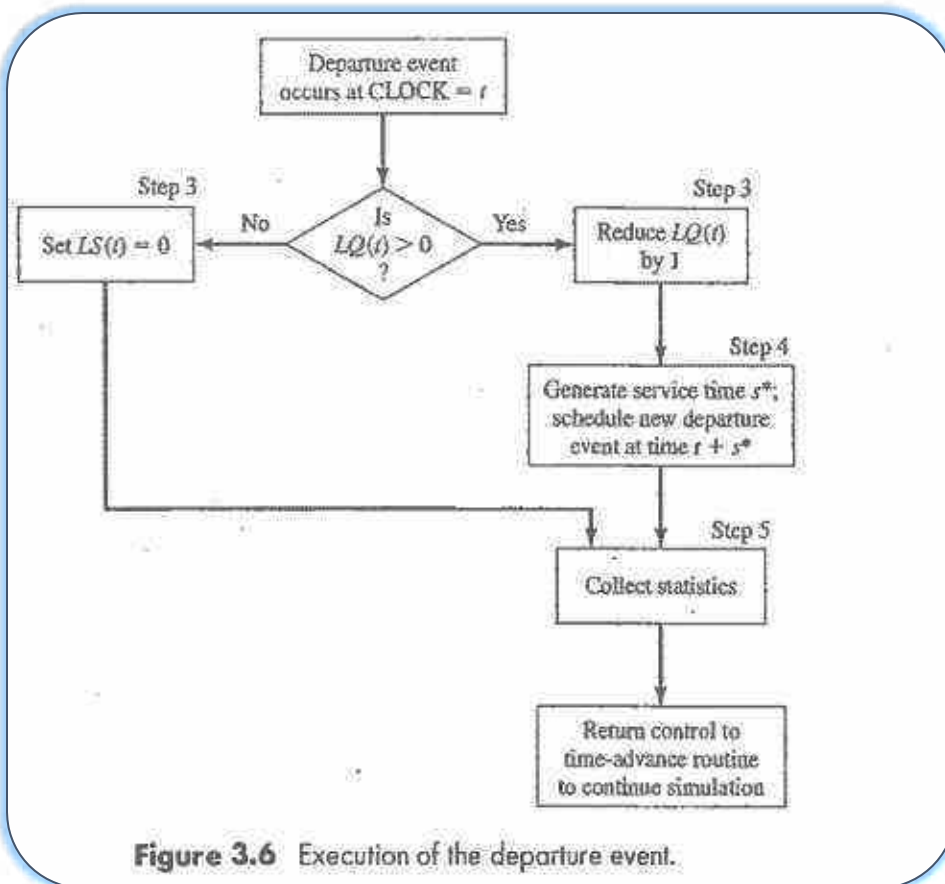


Figure 3.5 Execution of the arrival event.

Execution of the departure event



Summary

- Here we have examined the details of manual simulation

Simulation for Checkout Counter

- The simulation in Table 3.1 covers the time interval $\{0, 23\}$. At simulated time 23, the system empties, but the next arrival also occurs at time 23.
- The server was busy for 21 of the 23 time units simulated, and the maximum queue length was two.
- This simulation is, of course, too short to draw any reliable conclusions.
- Exercise 1 asks the reader to continue the simulation and to compare the results with those in Example 2.1.

- Note that the simulation table gives the system state at all times, not just the listed times.
- For example, from time 11 to time 15, there is one customer in service and one in the waiting line.

The Dump-Truck Problem

- Six dump trucks are used to haul coal from the entrance of a small mine to the railroad. Figure 3.7 provides a schematic of the dump-truck operation. Each **Here we have examined the details of manual simulation**
- truck is loaded by one of two loaders. After a loading, the truck immediately moves to the scale, to be weighed as soon as possible.
- Both the loaders and the scale have a first-come-first-served waiting line (or queue) for trucks.
- Travel time from a loader to the scale is considered negligible. After being weighed, a truck begins a travel time (during which time the truck unloads) and then afterward returns to the loader queue.
- The distributions of loading time, weighing time, and travel time are given in Tables 3.3, 3.4, and 3.5, respectively, together with the random digit assignment for generating

Summary

- Here we have examined the details of manual simulation

Module – 43

Manual Simulation Using Event Scheduling-III

Simulation for Checkout Counter

- The simulation in Table 3.1 covers the time interval $\{0,23\}$. At simulated time 23, the system empties, but the next arrival also occurs at time 23.
- The server was busy for 21 of the 23 time units simulated, and the maximum queue length was two.
- This simulation is, of course, too short to draw any reliable conclusions.
- Exercise 1 asks the reader to continue the simulation and to compare the results with those in Example 2.1.
- Note that the simulation table gives the system state at all times, not just the listed times.

- For example, from time 11 to time 15, there is one customer in service and one in the waiting line.

The Dump-Truck Problem

- Six dump trucks are used to haul coal from the entrance of a small mine to the railroad. Figure 3.7 provides a schematic of the dump-truck operation. Each truck is loaded by one of two loaders. After a loading, the truck immediately moves to the scale, to be weighed as soon as possible.
- Both the loaders and the scale have a first-come-first-served waiting line (or queue) for trucks.
- Travel time from a loader to the scale is considered negligible. After being weighed, a truck begins a travel time (during which time the truck unloads) and then afterward returns to the loader queue.
- The distributions of loading time, weighing time, and travel time are given in Tables 3.3, 3.4, and 3.5, respectively, together with the random digit assignment for generating

Summary

- Here we have examined the details of manual simulation

Module – 44

Statistical Models in Simulation

“Review Of Terminology And Concepts”

Discrete Random Variable

- **Discrete random variables**
 - Let X be a random variable. If the number of possible values of X is finite, or countably infinite, X is called a discrete random variable
 - The possible values of X may be listed as $x_1, x_2, \dots$

Discrete random variables

- In the finite case, the list terminates; in the countably infinite case, the list continues indefinitely.

Example # 1

- The number of jobs arriving each week at a job shop is observed. The random variable of interest is X , where
 - X = number of jobs arriving each week
- The possible values of X are given by the range space of X , which is denoted by R_x
 - Here $R_x = \{ 0, 1, 2, \dots \}$

Examples:

Let X be a discrete random variable. With each possible outcome x_i in R , a number $p(x_i) = P(X = x_i)$ gives the probability that the random variable equals the value of x_i .

- Example (Probability Distribution) The numbers $p(x_i)$, $i = 1, 2, \dots$, must satisfy the following two conditions :

1. $p(x_i) \geq 0$, for all i
2. $\sum_{i=1}^{\infty} p(x_i) = 1$

- The collection of pairs $(x_i, p(x_i))$, $i = 1, 2, \dots$ is called the probability distribution of X , and $p(x_i)$ is called the probability mass function (pmf) of X

Example # 2

- Consider the experiment of tossing a single die. Define X as the number of spots on the up face of the die after a toss.
 - Then $R_x = \{ 1, 2, 3, 4, 5, 6 \}$
- Assume the die is loaded so that the probability that a given face lands up is proportional to the number of spots showing.

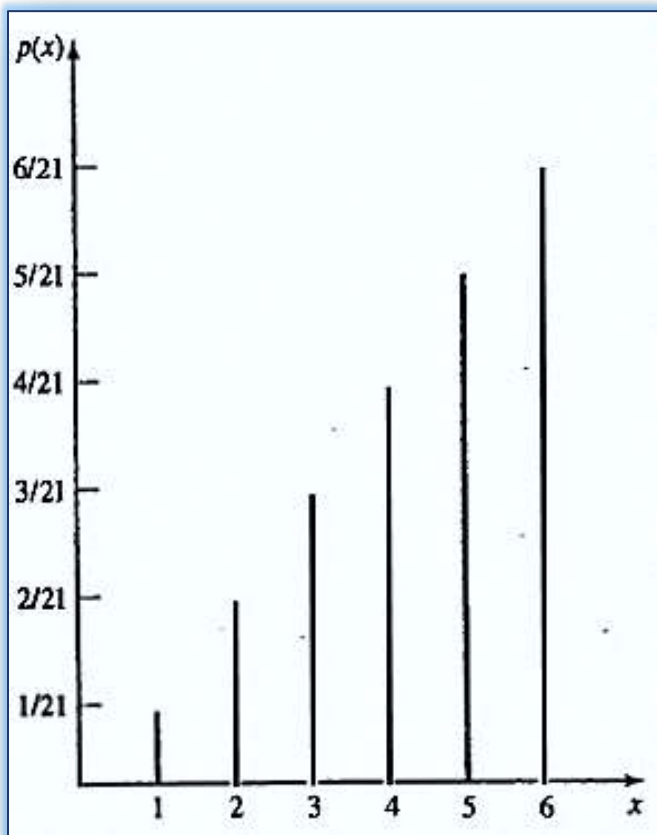
Example

- The *Discrete Probability Distribution* for this random experiment is given by

x_i	1	2	3	4	5	6
$p(x_i)$	1/21	2/21	3/21	4/21	5/21	6/21

- The conditions stated earlier are satisfied-that is, $p(x_i) \geq 0$ for $i = 1, 2, \dots, 6$ and $1/21 + \dots + 6/21 = 1$

Probability Mass Function For Loaded-die Example



Probability mass function for loaded-die example.

Summary

- In this Module we have seen about more properties of a random variable i.e. Probability Distribution and probability mass function

Module – 45

Review of Terms-I

Continuous Random Variable

- If the range space R_x of the random variable X is an interval or a collection of intervals, X is called a *Continuous Random Variable*. For a continuous random variable X , the probability that X lies in the interval $[a, b]$ is given by

$$P(a \leq X \leq b) = \int_a^b f(x) dx \quad (5.1)$$

- a. $f(x) \geq 0$ for all x in R_x
- b. $\int_{R_x} f(x) dx = 1$
- c. $f(x) = 0$ if x is not in R_x

- As a result of Equation (5.1), for any specified value
– x_0 , $P(X = x_0) = 0$, because

$$\int_{x_0}^{x_0} f(x) dx = 0$$

- $P(X = x_0) = 0$ also means that the following equations hold:

Continuous Random

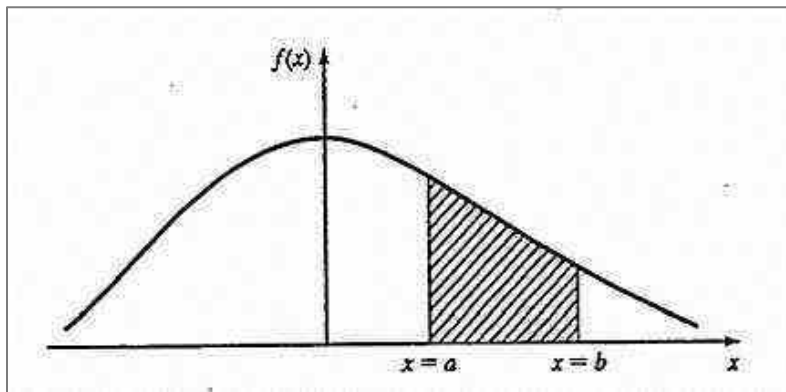
$$P(a \leq X \leq b) = P(a < X \leq b) = P(a \leq X < b) = P(a < X < b)$$

$$P(a \leq X \leq b) = \int_a^b f(x) dx$$

- The graphical interpretation of Equation is shown in Figure

Graphical Interpretation

- The shaded area represents the probability that X lies in the interval [a, b]



Graphical interpretation of $P(a < X < b)$.

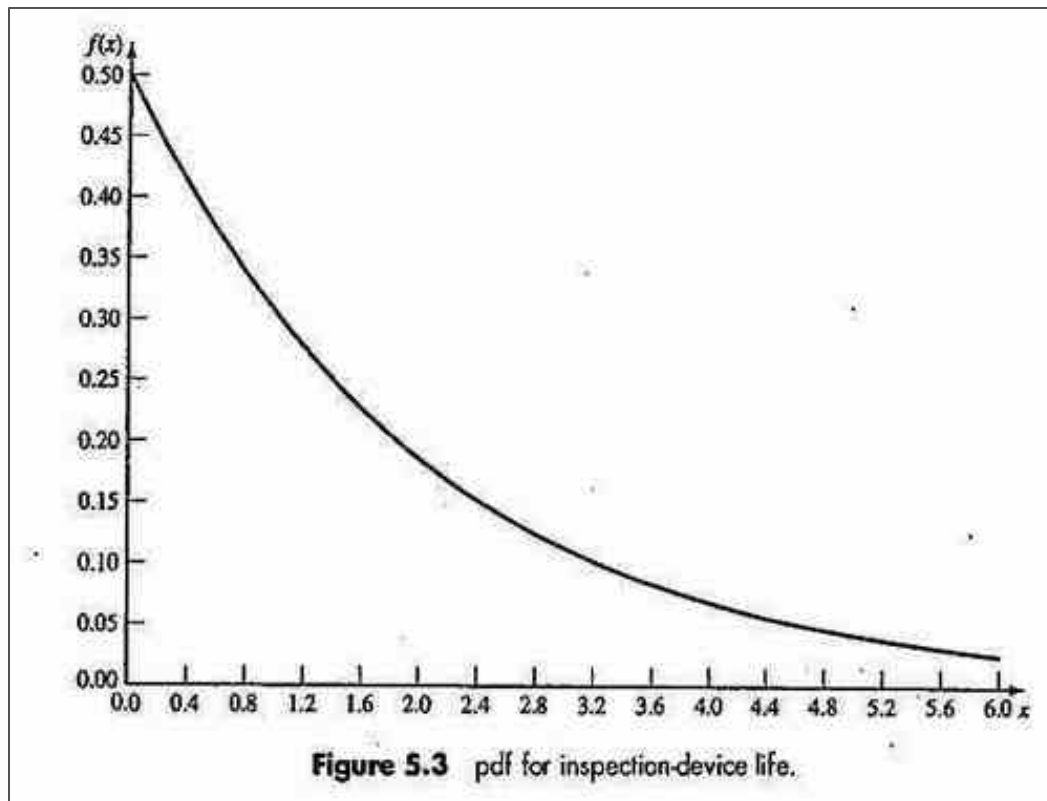
Example

- The life of a device used to inspect cracks in aircraft wings is given by X, a continuous random variable assuming all values in the range $x \geq 0$. The pdf of the lifetime, in years, is as follows:

$$f(x) = \begin{cases} \frac{1}{2}e^{-x/2}, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Pdf

- pdf is shown graphically in below in Figure . The random variable X is said to have an exponential distribution with mean 2 years



Probability Of Life Of The Device

- The probability that the life of the device is between 2 and 3 years is calculated as

$$\begin{aligned}
 P(2 \leq X \leq 3) &= \frac{1}{2} \int_2^3 e^{-x/2} dx \\
 &= -e^{-3/2} + e^{-1} = -0.223 + 0.368 = 0.145
 \end{aligned}$$

Summary

- In This Module we discussed about Continuous Random Variable and its probability of occurrences in time interval [a, b]

Module – 46

Review of Terms-II

Cumulative Distribution Function

- The cumulative distribution function (cdf), denoted by $F(x)$, measures the probability that the random variable X assumes a value less than or equal to x , that is, $F(x) = P(X \leq x)$

– If X is discrete, then

$$F(x) = \sum_{\substack{\text{all} \\ x_i \leq x}} p(x_i)$$

– If X is continuous, then

$$F(x) = \int_{-\infty}^x f(t) dt$$

Properties of CDF

- Some properties of the cdf are listed here:

- F is a nondecreasing function. If $a < b$, then $F(a) \leq F(b)$.
- $\lim_{x \rightarrow \infty} F(x) = 1$
- $\lim_{x \rightarrow -\infty} F(x) = 0$

- All probability questions about X can be answered in terms of the cdf. For example,

$$P(a < X \leq b) = F(b) - F(a) \quad \text{for all } a < b$$

- For continuous distributions, not only does Equation hold, but

$$P(a < X \leq b) = F(b) - F(a) \quad \text{for all } a < b$$

- Also the probabilities in Equation are equal to $F(b) - F(a)$

$$P(a \leq X \leq b) = P(a < X \leq b) = P(a \leq X < b) = P(a < X < b)$$

Loaded-die Example

- The die-tossing experiment described in Example previously discussed has a cdf given as follows:

x	$(-\infty, 1)$	$[1, 2)$	$[2, 3)$	$[3, 4)$	$[4, 5)$	$[5, 6)$	$[6, \infty)$
$F(x)$	0	1/21	3/21	6/21	10/21	15/21	21/21

- where $[a, b) = (a \leq x < b)$. The cdf for this example is shown graphically in Figure given in next slide.
- If X is a discrete random variable with possible values $x_1, x_2, \dots$, where $x_1 < x_2 < \dots$, the cdf is a step function.
- The value of the cdf is constant in the interval (x_{i-1}, x_i) and then takes a step, or jump, of size $p(x_i)$ at x_i
- Thus, in previous Example, $p(3) = 3/21$, which is the size of the step when $x = 3$.
- The cdf for the device described in Example is given by

$$F(x) = \frac{1}{2} \int_0^x e^{-t/2} dt = 1 - e^{-x/2}$$

- The probability that the device will last for less than 2 years is given by

$$P(0 \leq X \leq 2) = F(2) - F(0) = F(2) = 1 - e^{-1} = 0.632$$

- The probability that the life of the device is between 2 and 3 years is calculated as

$$\begin{aligned} P(2 \leq X \leq 3) &= F(3) - F(2) = (1 - e^{-3/2}) - (1 - e^{-1}) \\ &= -e^{-3/2} + e^{-1} = -0.223 + 0.368 = 0.145 \end{aligned}$$

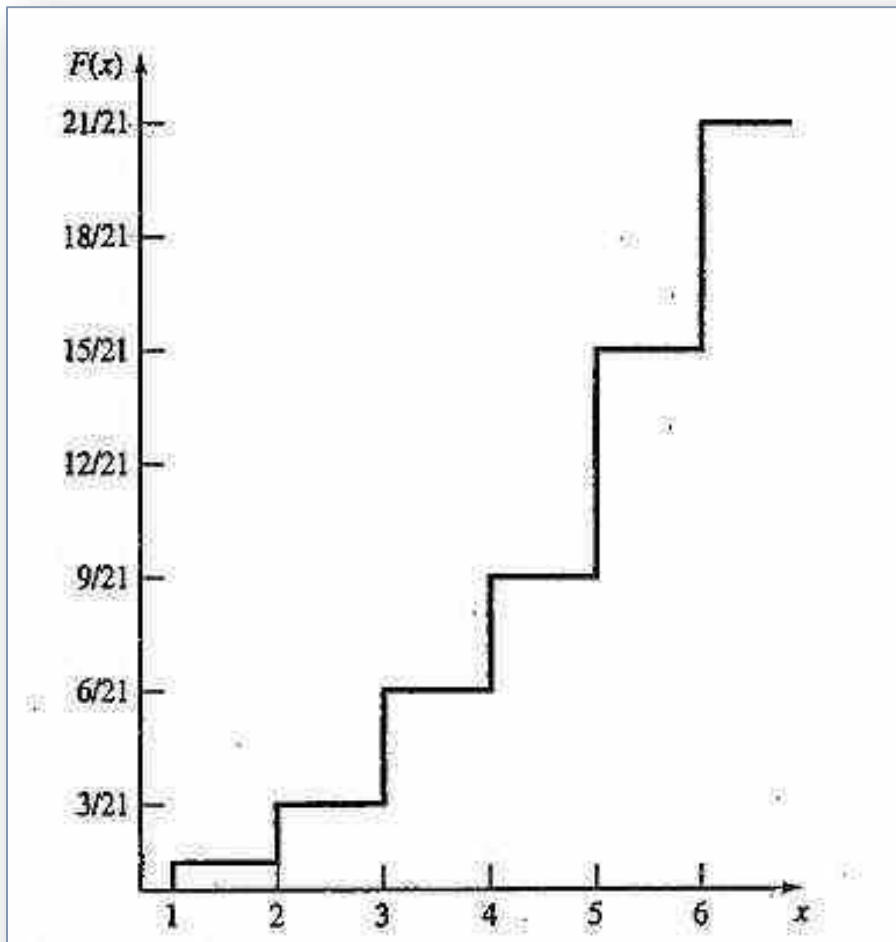
Summary

- In this Module we discussed about Cumulative Distribution Function of random variable X and its properties (e.g.) Standard Deviation Mean and Variance

Module – 47

Review of Terms-III

Cdf For Loaded-die Example



cdf for loaded-die example.

Expectation

- An important concept in probability theory is that of the expectation of a random variable.

- If X is a random variable, the expected value of X , denoted by $E(X)$, for discrete and continuous variables is defined as follows:

$$E(X) = \sum_{\text{all } i} x_i p(x_i) \quad \text{if } X \text{ is discrete} \quad (5.6)$$

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx \quad \text{if } X \text{ is continuous} \quad (5.7)$$

nth Moment Of X

- The expected value $E(X)$ of a random variable X is also referred to as the mean, μ , or the first moment of X .

- The quantity $E(X^n), n \geq 1$, is called the n th moment of X , and is computed as follows:

Variance of X

The variance of a random variable, X , denoted by $V(X)$ or $\text{var}(X)$ or σ^2 , is defined by

$$V(X) = E[(X - E(X))^2]$$

A useful identity in computing $V(X)$ is given by

$$V(X) = E(X^2) - [E(X)]^2$$

- The mean $E(X)$ is a measure of the central tendency of a random variable. The variance of X measures the expected value of the squared difference between the random variable and its mean.
- Thus, the variance,
 - $V(X)$, is a measure of the spread or variation of the possible values of X around the mean $E(X)$. The standard deviation, σ , is defined to be the square root of the variance

Mean And The Standard Deviation

- The mean, $E(X)$, and the standard deviation **have same units**

Thus,

$$V(X) = 21 - \left(\frac{91}{21}\right)^2 = 21 - 18.78 = 2.22$$

and

$$\sigma = \sqrt{V(X)} = 1.49$$

Example

- The mean and variance of the life of the device described in Example 5.3 are computed as follows:

$$\begin{aligned} E(X) &= \frac{1}{2} \int_0^{\infty} x e^{-x/2} dx = -x e^{-x/2} \Big|_0^{\infty} + \int_0^{\infty} e^{-x/2} dx \\ &= 0 + \frac{1}{1/2} e^{-x/2} \Big|_0^{\infty} = 2 \text{ years} \end{aligned}$$

- To compute $V(X)$ from Equation (5.10), first compute $E(X^2)$ from Equation (5.9) as follows:

Thus,

$$E(X^2) = -x^2 e^{-x/2} \Big|_0^{\infty} + 2 \int_0^{\infty} x e^{-x/2} dx = 8$$

giving

$$V(X) = 8 - 2^2 = 4 \text{ years}^2$$

and

$$\sigma = \sqrt{V(X)} = 2 \text{ years}$$

Summary

- In this Module we discussed about Cumulative Distribution Function of random variable X and its properties (e.g.) Standard Deviation Mean and Variance

Module – 48

Review of Terms-IV

The Mode

- The Mode is used in describing several statistical models that appear in this chapter

The Mode (Discrete case)

- In the discrete case, the mode is the value of the random variable that occurs most frequently.

The Mode (Continuous case)

- In the continuous case, the mode is the value at which the pdf is maximized
- The mode might not be unique

Bimodal

- If the modal value occurs at two values of the random variable, the distribution is said to be *Bimodal*

Summary

- In this module we discussed Mode is used in describing several statistical models

Module – 49

Useful Statistical Models

- Numerous situations arise in the conduct of a simulation where an investigator may choose to introduce probabilistic events
- In an inventory model, the time between demands and the lead times (time between placing and receiving an order) can be probabilistic
- In a *reliability model*, the time to failure could be probabilistic. In each of these instances, the *simulation analyst* desires to generate random events and to use a known statistical model if the underlying distribution can be found

Queueing systems

- In the queueing examples, interarrival- and service-time patterns were given. In these examples, the times between arrivals and the service times were always probabilistic, as is usually the case.
- However, it is possible to have a constant interarrival time (as in the case of a line moving at a constant speed in the assembly of an automobile), or a constant service time (as in the case of robotized spot welding on the same assembly line).
- The following example illustrates how probabilistic interarrival times might occur.

Example # 1

- Mechanics arrive at a centralized tool crib as shown in Table 5.1. Attendants check in and check out the requested tools to the mechanics. The collection of data begins at 10:00 A.M. and continues until 20 different interarrival times are recorded.

Example

- Rather than record the actual time of day, the absolute time from a given origin could have been computed.
- Thus, the first mechanic could have arrived at time zero, the second mechanic at time 7:13 (7 minutes, 13 seconds), and so on.

Table (Arrival Data)

Arrival Number	Arrival (Hour:Minutes:Seconds)	Interarrival Time (Minutes:Seconds)
1	10:05:03	—
2	10:12:16	7:13
3	10:15:48	3:32
4	10:24:27	8:39
5	10:32:19	7:52
6	10:35:43	3:24
7	10:39:51	4:08
8	10:40:30	0:39
9	10:41:17	0:47
10	10:44:12	2:55
11	10:45:47	1:35
12	10:50:47	5:00
13	11:00:05	9:18
14	11:04:58	4:53
15	11:06:12	1:14
16	11:11:23	5:11
17	11:16:31	5:08
18	11:17:14	0:47
19	11:21:26	4:08
20	11:24:43	3:17
21	11:31:19	6:36

Summary

- In this Module we discussed about useful Statistical models and relation between distribution of time arrivals and number of arrivals per time period

Module – 50

Useful Statistical Models-II

Example # 2

- Another way of presenting interarrival data is to find the number of arrivals per time period.
- Here, such arrivals occur over approximately 1 1/2 hours; it is convenient to look at 10-minute time intervals for the first 20 mechanics.

Example

- In the second time period, two mechanics arrived, and so on. The results are summarized in Table 5.2.
- This data could then be plotted in a histogram, as shown in Figure 5.5.

Table (Arrivals in Successive Time Periods)

Table 5.2 Arrivals in Successive Time Periods

<i>Time Period</i>	<i>Number of Arrivals</i>	<i>Time Period</i>	<i>Number of Arrivals</i>
1	1	6	1
2	2	7	3
3	1	8	3
4	3	9	2
5	4	—	—

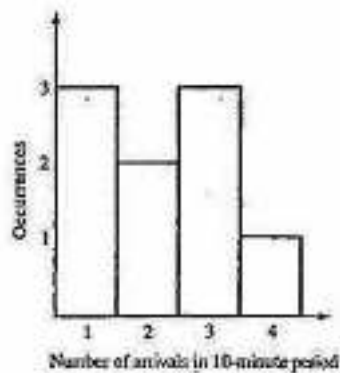


Figure 5.5 Histogram of arrivals per time period.

Distribution Of Time

- The distribution of time between arrivals and the distribution of the number of arrivals per time period are important in the simulation of waiting-line systems.
- "Arrivals" occur in numerous ways; as machine breakdowns, as jobs coming into a job shop, as units being assembled on a line, as orders to a warehouse, as data packets to a computer system, as calls to a call center, and so on

Summary

- In this Module we discussed about useful Statistical models and relation between distribution of time arrivals and number of arrivals per time period

Why Agent-Based Modeling?

- Before we discuss why *Agent-based Modeling* is important, we briefly describe what agent-based modeling is.
 - An agent is an autonomous computational individual or object with particular properties and actions.
- Agent-based modeling is a form of computational modeling whereby a phenomenon is modeled in terms of agents and their interactions
- In the past two decades,
- scientists have increasingly used agent-based modeling methods to conduct their research

ABM

- ABM is a species of computation, growing up alongside the maturation of computer technology.
- The combination of large data, cheap computation, and high connectivity allows agent-based models to be constructed with millions of individual agents

Agent-based Representations

- Agent-based representations are easier to understand than mathematical representations of the same phenomenon.
 - This is because agent-based models are constructed out of individual objects
 - and simple rules for their movement of behavior, as opposed to equational models that are constructed from mathematical symbols.

Individual Agents

- In thinking about individual agents, we can make sense of them by projecting our bodily experience onto the agents.
- Thus, the language and concepts we use in agent-based modeling is much closer to natural language and our natural thinking.

Step To Presenting This Conceptualization

- As a first step to presenting this conceptualization, we look back historically at changes to representational tools and practices used in science and the significant benefits they had for both scientists and learners.

Summary

- The main argument of this introductory module is

- That ABM is a transformative representational technology that enables us to better understand familiar topics

Module – 52

A Thought Experiment

- Imagine a country (let 's call it Foo) where people represented numbers as the Romans did, using symbols such as MCMXLVIII.
- Fooian scientists worked laboriously to quantify and more accurately calculate basic science such as planetary motion and dynamical forces.

Number Of Different Approaches

- Business people had great difficulties with their financial calculations, tradespeople struggled with their measurements, and consumers labored to assess their purchases.
- Educational researchers and policymakers in this imaginary country were very concerned with the difficulty of learning to handle numbers,
- They worked hard to make these skills accessible to more of their citizens.
- They engaged in a number of different approaches.

Name The Sort Of Innovation

- A first step is to name the sort of innovation associated with the shift from Roman to Hindu-Arabic representations of number
- It is not sufficient,
 - for example, to say that we have a new approach to learning and working with numbers

Structuration

- By Structuration we mean :
 - The encoding of the knowledge in a domain as a function of the representational infrastructure used to express the knowledge.

Restructuration

- A change from one structuration of a domain to another resulting from such a change in representational infrastructure we call a restructuring .

Examples Of Restructuration

- The development of Arabic numerals and the transformation of kinematics via algebra were empowering and democratizing, enabling significant progress in science and widening the range of people who could make sense of previously formidable topics and skills

Summary

- If the problems with which we struggle today could be so transformed, think of the new domains we could enter and conquer.
- This Module was about the
 - “who can do what, ”

Module – 53

Complex Systems and Emergence

- What areas are widely thought to be difficult for people to comprehend and potentially ripe for restructuration? One such area is complex systems.
- Its very name suggests that it is a difficult area for comprehension.
- However, even if the level of complexity in our life remained constant over the ages, our continual quest for knowledge would ultimately lead us to study complex systems.

From Simple to Complex System

- As we gain facility and more complete understanding of simple systems, we naturally progress to trying to make sense of increasingly complex systems.

Simple Population Dynamics Models

- Simple population dynamics models,
 - For example, make the implicit assumption that all members of a species are the same,
 - But later, it becomes important to explore the manifold complexity of the food web
 - And how each individual interacts with every other individual.
- Thus, our need to understand more complex systems is also a natural result of the growth of human knowledge.

Complex Systems Theory

- Complex systems theory develops principles and tools for making sense of the world ' s complexity and defines complex systems:
 - as systems that are composed of multiple individual elements that interact with each other
 - yet whose aggregate properties or behavior is not predictable from the elements themselves.

Emergent Phenomenon

- Through the interaction of the multiple distributed elements an “ emergent phenomenon ” arises.
- The phenomenon of emergence is characteristic of complex systems.
- The term “ emergent ” was coined by the British philosopher and psychologist G. H. Lewes

Emergence

- Since Lewes ' s time, scholars have struggled with how to best define emergence
- For our purposes, we define emergence as :
 - The arising of novel and coherent structures, patterns, and properties through the interactions of multiple distributed elements .

Features Of Emergence

- Important features of emergence include:
 - The global pattern ' s spontaneous arising from the interaction of elements,
 - And the absence of an orchestrator or centralized coordinator — the system “ self-organizes. ”

Macrostructures Are Emergent

- Structure (or rules) at the micro-level leads to ordered pattern at the macro-level.
 - Because the macrostructures are emergent, composed of many elements, they are dynamic,
 - And perturbing them often results in them dynamically reforming

Summary

- In This Module we discussed about complex systems theory which is ever finding more complex systems that may at first seem irreducible but are found to be self-organized or evolved rather than intelligently designed by a designer.

Module – 54

Understanding Complex Systems and Emergence

- We have said that understanding complex systems and emergence is hard for people.

Fundamental And Distinct Challenges

- Emergence, in particular, presents two fundamental and distinct challenges.
 - The first difficulty lies in trying to figure out the aggregate pattern when one knows how individual elements behave.

Integrative Understanding

- We sometimes call the first fundamental and distinct challenge “integrative” understanding,
 - As it parallels the cumulative integration of small differences in calculus.

Fundamental And Distinct Challenges

- A second difficulty in fundamental and distinct challenge arises
 - when the aggregate pattern is known and one is trying to find the behavior of the elements that could generate the pattern

Differential Understanding

- We sometimes call the second fundamental and distinct challenge a “differential” understanding (aka compositional understanding),
 - As it parallels the search in calculus for the small elements that produce an aggregate graph when accumulated

Summary

- In this Module we have discussed about why it is hard to understand complex systems and emergence

Module – 55

Example 1:

Integrative Understanding

- Figure presents a system composed of a few identical elements following one rule.



Some arrows moving clockwise around a circle of radius 20

- Each element is a small arrow.
- We imagine a clock ticking and at each tick of the clock the arrows follow their rule.
- We initialize the system so that each individual arrow starts on a circle (of radius 20 units).
- We start them all facing clockwise on the circle.
- Now, we give them one movement behavior (or rule).
- At every tick of the clock, they move forward 0.35 units then turn right one degree.
- As the clock ticks, they continue to move and turn, move and turn, moving clockwise along the circle
- Now suppose that we slightly alter these rules.
 - Instead of moving forward 0.35 units, we have them move 0.5 units while still turning one degree. What will be the aggregate pattern that we see??
- In fact, the pattern that emerges is a pulsating circle.
 - All the arrows stay in a circle, but the circle changes its radius,
 - First expanding, then contracting
 - And repeating this cycle forever.

Summary

- In this Module we analyzed certain pattern of agents and their behavior while changing factors

Module – 56

Example 2:

Differential Understanding

- There are many coherent, powerful, and beautiful patterns we observe in the world. What accounts for their prevalence? How do they originate?
- The secret to understanding the formation of these patterns is to understand
 - That they are emergent, arising from the interactions of distributed individual elements.
- Flocking Of Birds One such prevalent (and often beautiful) pattern is the flocking of birds.
 - Birds fly together in many different formations, from the classic V formation of goose flocks to the large, very dense flocks of starlings that resemble insect swarms.

How And Why Do Birds Flocks Emerge?



Deterministic-centralized Mindset

- In the 1980s and early '90s, Wilensky and Resnick interviewed people from a wide range of backgrounds, asking them to explain how such patterns arise.

Cognitive Pattern

- The results suggested a phenomenon, or cognitive pattern, called the deterministic-centralized mindset, or DC mindset.
 - The pattern stemmed from two main empirical findings:

Empirical Findings (The D Component)

1. Most subjects did not see any role for randomness in creating these structures, randomness was seen as destructive of pattern, not a force for creating pattern (the D component);

Empirical Findings (The C Component)

Most subjects described these patterns as arising from the actions of a centralized controller or orchestrator (the C component).

Summary

- In this module we have discussed that in the past two decades, researchers have found that emergent phenomena are endemic to the natural and social worlds and that using an emergent lens to make sense of complex patterns is a vital need in a twenty-first-century world.

Module – 57

Differential Understanding-II

Thinking In Levels

- In further analyses of the interviews, Wilensky and Resnick (1999) identified a key component of the DC mindset and an obstacle to thinking about emergent phenomena:
- The problem of “ thinking in levels. ”

Emergent phenomena’s two levels

- Emergent phenomena can be described as existing on
 - at least two levels: the level of the individual elements (cars, birds, people, etc.); and the
 - level of system or aggregate patterns (flocks, traffic jams, housing patterns, etc.).
- Most people fail to distinguish between these levels, instead “ slipping ” between levels to attribute the properties of one level to the other

Appearance Of Stability

- The appearance of stability often leads people to conclude that the individual elements of the flock (the birds) are stable and have a consistent place in the flock.

Complex Phenomena

- Wilensky and Resnick also showed a host of examples where this levels slippage interfered with both integrative and differential understanding of complex phenomena in the natural and human social worlds.

Summary

- In this module we have discussed that in the past two decades, researchers have found that emergent phenomena are endemic to the natural and social worlds and that using an emergent lens to make sense of complex patterns is a vital need in a twenty-first-century world.

Module – 59

Agent-Based Modeling as Representational Infrastructure for Restructurations

- With the aid of new computer-based modeling environments, we can simulate complex patterns and better understand how they arise in nature and society.
- Whereas in many areas of science we have relied on simplified descriptions of complexity

Agent-based modeling

- Agent-based modeling is a computational methodology that enables one to model complex systems.
- As the name suggests, agent-based models are composed of agents:
 - computational entities that have properties, or state variables and values

Agents

- Agents usually also have a graphical component so you can see them on the computer screen.
 - An agent can represent any element of a system.

Gas Molecule Agent

- A gas molecule agent, for instance, might have properties such as
 - “Mass ” with value 30 atomic mass units,
 - “Speed ” with value 10 meters per second,
 - and “ Heading ” with a value of the angle it is facing

A Sheep Agent

- A sheep agent, by contrast, might have properties such as
- “ speed ” with value 3 mph,
- “ weight” with value 30 lbs.,
- and fleece with a value of “ full ” (a discrete-textual rather than numerical value).

Agents Also Have Rules Of Behavior

- In addition to their properties, agents also have rules of behavior.
 - A gas molecule agent might have a rule to collide with another molecule;
 - A sheep agent might have a rule to eat grass if there is grass available nearby

Universal Clock

- In an agent-based model, we imagine a universal clock.
- When the clock ticks, all agents invoke their rules.
- If the conditions of the rules are satisfied, (e.g., they are at the edge of a box, or grass is nearby), they enact the behavior (i.e., bounce or eat grass)

Summary

- In this module we discussed While different fields have different degrees of structuration inertia, there is no end to the domains of application for ABM.
- However, differences in structuration inertia render some fields more easily adaptable to ABM restructuration than others.

Module – 59

Agent-Based Modeling as Representational Infrastructure for Restructurations

Agents And Rules

- The goal of agent-based modeling is to create agents and rules that will generate a target behavior.
- Sometimes the rules are not well known, or you just want to explore the system’s behavior.

ABM’s Enable Restructurations Of Complex Systems

- We contend that ABM ’s enable restructuration of complex systems so that the
 - (a) understanding of complex systems can be democratized and
 - (b) the science of complex systems can be advanced.

NetLogo

- The computer language used in this text, NetLogo (Wilensky, 1999), was developed by Uri Wilensky for these express purposes.
 - It is a general-purpose agent-based modeling language designed to be “ low-threshold ”
 - but also “ high-ceiling ”

Turtle

- NetLogo calls its prototypical agent a “ turtle. ”
- NetLogo was first developed in the late 1990s, and it is now in use by hundreds of thousands of users worldwide.

Netlogo To Construct And Explore Models

- Thousands of scientific papers have utilized NetLogo to construct and explore models in a wide variety of disciplines.
- It has also been employed by policymakers to model policy choices, business practitioners to model business decisions

Summary

- In this module we discussed While different fields have different degrees of structuration inertia, there is no end to the domains of application for ABM.
- However, differences in structuration inertia render some fields more easily adaptable to ABM restructuring than others.

Module – 60

Example:

Predator-Prey Interactions

- To illustrate the potential power of widespread agent-based modeling literacy and restructuring,
- we will look briefly at two examples derived from different content domains:
 - predator-prey interactions
 - the spread of forest fires

- Let us start with the study of predator-prey interactions.
- This domain is often first introduced qualitatively in high school, then quantitatively at the university level

Lotka-volterra Differential Equations

- In its quantitative form, the population dynamics of a single predator and prey are introduced by the classic Lotka-Volterra differential equations, a pair of coupled differential equations that proceed as follows:

- Lotka-Volterra differential equations

- $$\frac{dPred}{dt} = K_1 * Pred * Prey - M * Pred$$

- $$\frac{dPrey}{dt} = B * Prey - K_2 * Pred * Prey$$

- The first equation says that the number of predators increases as

First DFE

- Predators interact with prey

(by fixed constant K1)

- and decreases by a constant mortality rate (M).

Second DFE

- The second equation says that the number of prey
- Increase by a constant birthrate (B) and
- Decreases in interaction with predators (by a fixed constant K 2).

Summary

- In This module we described
 - In general, exploring complementary approaches to a single problem provides us with deeper insight.
 - If multiple different approaches find a similar pattern of behavior at different levels of analysis, then there is better confirmation of the underlying result.

Module – 61

Example:

Predator-Prey Interactions

Agent-based Representation Of Predator And Prey

- An agent-based representation of predator and prey, such as the one illustrated, would typically employ simple algorithmic models.



An agent-based Wolf Sheep Predation model.

- They might give each predator and prey a store of energy that is depleted when they move and increased when they eat.
 - If their energy dips too low, they die.
 - If it is high enough, they might reproduce.
- And when they move, if they encounter food (which, for the predator, is the prey), they eat it
- These instructions are explicitly stated in an easy-to-read language that instructs each agent in how to behave, and accompanies the visual model.

Explicit Mechanism

- This kind of representation makes its mechanisms explicit; as such, they are usually quite understandable by even young children.
- They also can be challenged more easily and tested.
- We will explore such predator-prey models in detail

Advantages To Agent-based Representations

- There are many advantages to agent-based representations.
- First, it does not require knowing calculus.
- Requiring calculus to be able to reason about predator-prey interactions sets the entry threshold quite high.

Advantages To ABM Representations

- The content of the predator-prey model can be made quite accessible to a non-calculus audience through ABM.
- Agent-based models can serve as powerful complements to equation-based models.
- They are particularly effective entry points into scientific domains.

Disadvantage Of ABM Representation

- ABM also have some disadvantages as compared to equations. For an expert, an equation can more compactly represent a phenomenon than can an ABM.
- Moreover, when the equation is solvable, it enables the direct calculation of results without the need to run a model.

Disadvantages of ABM

- We now know from the works of biologists such as Gause (1936) that the equational model is less accurate than an ABM in the isolated predator-prey situation for which it was intended.
- In particular, the equational model underrepresents extinctions, since the model uses real numbers to represent the population densities.

Summary

- In This module we described
 - In general, exploring complementary approaches to a single problem provides us with deeper insight.
 - If multiple different approaches find a similar pattern of behavior at different levels of analysis, then there is better confirmation of the underlying result.

Module – 62

Example:

Forest Fires

- Our second example is about the spread of a forest fire.
- It typically falls under the subject matter of physics, described in terms of two classic partial differential equations.
 - Classic heat equation
 - Spread of a forest fire

Classic Heat Equation

- The first is the classic heat equation,
 - which describes the distribution of heat in a given region over time, where θ represents the thermal diffusivity of the material through which the heat is traveling.

$$\frac{dH(x,t)}{dt} = \theta \frac{d^2 H(x,t)}{dx^2}$$

- The second equation physicists use to describe :
 - The spread of a forest fire treats the fire as if it were a potentially turbulent fluid, thus using the Reynolds equation of fluid flow.

$$\frac{dU_i}{dt} + U_j \frac{dU_i}{dX_j} = \frac{1}{\rho} \frac{dP}{dx_i} + \nu \frac{d^2 U_i}{dx_j dx_j} - \frac{d}{dx_j} \overline{u_i^t u_j^t}$$

ABM Approach To Modeling Forest Fires

Contrast these two equations with the ABM approach to modeling forest fires (illustrated in

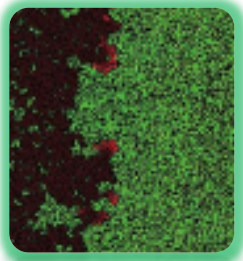


figure below),

ABM Approach To Modeling Forest Fires

- This would typically model the environment as a grid of cells with trees occupying certain cells.
- Modeling the spread of fire consists simply of giving rules to the cells that are on fire as to when to spread to neighboring tree cells.
- A very simple ABM of forest fire spread would not correctly model a particular fire,
 - But it does give insight into the dynamics of any fire and once we know the details of a particular fire,
 - we can add in whatever data or rules that apply to the situation.

ABM Provides Several Representational Benefits

- ABM methods are starting to be used to model and fight real forest fires
- The restructuration of these systems using ABM provides several representational benefits.

ABM uses Discrete than Continuous Representations

- They make use of discrete rather than continuous representations,
 - which are more easily comprehensible, more closely match real-world situations and require much less formal mathematics to employ.

Summary

- In this Module we discussed how ABM is more preferable than any mathematical equations
- ‘Agent-based models’ are now used in the professions to do research in medicine and law and by policymakers to help them explore effects of alternative policies.
- In this Module we discussed how ABM is more preferable than any mathematical equations

- ‘Agent-based models’ are now used in the professions to do research in medicine and law and by policymakers to help them explore effects of alternative policies.

Module – 63

What Is Agent-Based Modeling?

Ants (Foraging For Food)

- The ant opens her eyes and looks around. There are many of her siblings nearby, but there is no food.
- The ant is hungry, so she heads out from the ant colony and starts to wander around.
- She sniffs a little to the left and a little to the right,
- and still she cannot smell any food. So she keeps wandering.
- She passes by several of her sisters, but they do not interest her right now;
 - she has food on her mind.
- She keeps wandering. Sniff! Sniff!
- She gets a whiff of some of that delightful pheromone stuff.
- She heads in the direction of the strongest pheromone scent;
 - in the past there has been food at the end of that trail.
- Sure enough, as the ant proceeds along the trail, she arrives at some delicious food.
- She grabs some food and heads back to the colony, making sure to drop some pheromone along the way.
- On the way back, she runs into many of her sisters, each sniffing her way along pheromone trails, repeating the same process they will carry out all day.

Model Of Ant Behavior

- What we have just fancifully described is an ant foraging for food.
- The opening paragraph of this Module is in itself a model of ant behavior.

Model

- By a model ,
 - we mean an abstracted description of a process, object, or event.
- Models can take many distinct forms.

- However, certain forms of models are easier to manipulate than other forms.

Textual Description

- The textual description in our first paragraph cannot easily be manipulated to answer questions about the ant colony ' s behavior

— for example, what would we see if all the ants in the colony followed the behavior we described ?

Textual Model Is Not Sufficiently Generative

- It is difficult to extrapolate from the textual description of one ant to a description of an ant colony.
- The textual model is not sufficiently generative to answer such questions.
- It is a fixed description

— always “ behaving ” the same

Summary

- In this module we have described that the methodology of ABM :
 - Encodes the behavior of individual agents in simple rules so that we can observe the results of these agents ' interactions.
- This technique can be used to model and describe a wide variety of processes, phenomena, and situations

Module – 64

What Is Agent-Based Modeling?

Computational Model

- One way to make the preceding model more generalizable and gain insight into the behavior of an ant colony would be to implement the model in a
 - computational form
- A computational model is a model :
 - that takes certain input values, manipulates those inputs in an algorithmic way, and generates outputs.
- In computational form, it would be easy to run the “ Ants ” model

- with large numbers of ants and to observe the model's outputs given many possible different inputs.

Model implementation

- We use the term model implementation to refer to this process of transforming a textual model
 - into a working computational simulation
 - (written in some form of computer “ code ”)

Conceptual Model

- Besides textual models, there are other forms of conceptual models that describe processes, objects, or events but are not computational;
 - conceptual models can also be diagrammatic or pictorial.

Agent

- By the word agent , we mean an autonomous individual element of a computer simulation.
- These individual elements have properties, states, and behaviors.

Agent (Ant)

- The ant is an agent.
- It has properties such as:
 - Its appearance and its rate of movement.
- It has characteristic behaviors such as:
 - Moving, sniffing, picking up food, and dropping pheromone.
- It has states such as whether or not it is carrying food (a binary state)
- and whether it can sense how much pheromone is in the environment around it (a multi-valued state).

Agent-based modeling (ABM)

- Agent-based modeling (ABM) is a computational modeling paradigm that enables us to describe how any agent will behave.
- it is most useful when describing these phenomena as complex systems .

Summary

- In this module we have described that the methodology of ABM :

- Encodes the behavior of individual agents in simple rules so that we can observe the results of these agents ' interactions.
- This technique can be used to model and describe a wide variety of processes, phenomena, and situations

Module – 65

Creating the Ant Foraging Model

Ant Foraging Model

- Many biologists and entomologists have observed ants in the wild (Hölldobler & Wilson, 1998 ; Wilson, 1974)
- And have described how ants seemed to form trails to and from food sources and their nest.

Complex System

- The study of complex systems has become an important scientific frontier.
- By the term complex system we mean a system composed of many distributed interacting parts .
- The field of complex systems or complexity science arose in the mid-1980s
- It was born out of a variety of disparate fields from economics to physics to ecology.
- Complex systems science provides a set of tools and frameworks for viewing phenomena.
- Any phenomenon can be viewed as a complex system, and choosing when to do so depends on assessing when it is most useful to use the lens and/or methodologies of complex systems.

Complexity Science

- One important methodology of complexity science is agent-based modeling.
- The history of agent-based modeling and complex systems is explored

Ant Foraging Model (Hypothesis)

- After an ant finds food, it goes back to the nest, drops the food off, and communicates to the queen ant, and this queen ant tells the other ants where the food is and sends them off to collect it.
- Suppose a researcher notices :
 - that the food-finding ant never communicates with any kind of “ boss ” ant before leaving the nest again,
 - so the researcher might reason that food gathering was not working through a centralized leader or controller but rather through distributed control

An Other Hypothesis

- Another hypothesis that could be advanced is that perhaps the ant did not communicate with a central leader,
- But rather simply communicated with other ants, and those other ants were able to “ diffuse ” the information about the food source throughout the nest.
- There is reason to believe such a hypothesis might be true, since bees work in a similar way;
- when a bee finds a food source, it returns to the hive and conducts a complicated dance that tells the other bees where to find the food (Gould & Gould,1988)
- However, though ants do have some methods of communicating information directly (Hölldobler & Wilson, 1998)
- Most species do not communicate the exact route to a food source in this way.

Summary

- In this module we explored about ant’s behavior and have described about different Hypothesis , By the end of chapter 3, you will be able to make substantive modifications to this model,(Ant Foraging Model)

Creating the Ant Foraging Model-II

Basic Hypothesis

- Now that we have a basic hypothesis (in the form of a textual model) that describes the behavior of the ant,
- how do we implement the model so that we can test out this hypothesis
- We also check if our computational model adequately models nature?

Implemented Model

- The first step is to create a more algorithmic description of the preceding textual model.
- This is just another model itself, but one that is more easily translated into an implemented model.

Set Of Rules

- Here is one set of rules that an individual ant could follow in order to operate in accordance with the model described earlier.
- We describe the rules from the point of view of an individual ant.
 1. If I am not carrying food, I check if there is food where I am;
 2. if there is, I pick it up; if there isn't food right here,
 3. I try to sense a pheromone trail nearby; if I find one, then I face “uphill” along the pheromone gradient toward its strongest source.
 4. If I am carrying food, I turn back toward the nest and drop pheromone on the ground below me.
 5. I turn randomly a small amount and move forward a step.

NetLogo

- These rules are easily implemented in a computer language.
- There are many computer languages in use for many purposes,
 - but only a few are especially tailored to work with agent-based modeling.
- One of these is NetLogo (Wilensky, 1999a).
- NetLogo is both a modeling language and an integrated environment designed to make agent-based models easy to build.

- In fact, NetLogo is so easy to use that, rather than describe algorithms and models in pseudo-code,

NetLogo code:

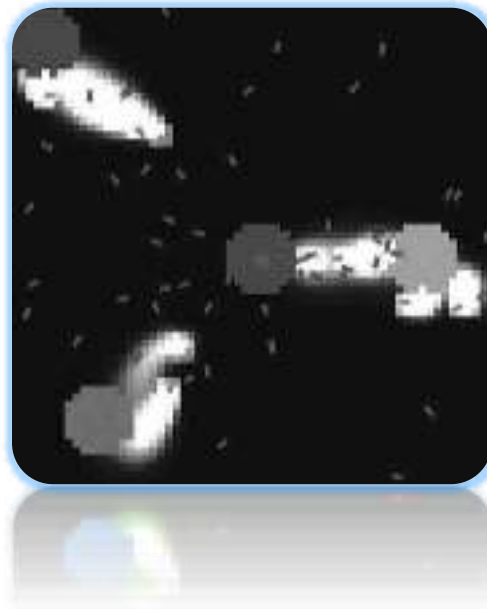
- throughout this course we will use NetLogo instead.
- Describing the preceding rules in the NetLogo language is straightforward.
- Here, for example, is one translation of rules 1 – 3 into NetLogo code:

```

If not carrying-food? [ look-for-food ]      ;; if not carrying food, look for it
if carrying-food? [ move-towards-nest ]      ;; if carrying food turn back towards the nest
wander                                       ;; turn a small random amount and move forward

```

NetLogo “Ants” model of foraging behavior



NetLogo “Ants” model of foraging behavior.

Summary

- In this module we have explored how to model the ant’s behavior and have described it using different hypotheses

Module – 67

Creating the Ant Foraging Model-III

NetLogo Videos

Forest Fire

Results and Observations (Ant Model)

- If we look closely at the model running, we observe that initially the ants wander around at random.
- Then some ants will wander into a nearby food source.

Results and Observations

- Once they find that food source, they will start to bring food back to the nest, laying a pheromone trail beneath them.
- If just one lone ant finds the food source, the pheromone trail will not be strong enough for other ants to follow it;
- As more and more ants find the food source, the trail will become stronger and stronger.
- The ants as a group appear to exploit food sources in an optimal manner.
- That is, they first gather food from the nearest food source, then the second nearest, and so on.
- This appears to be a conscious plan of the ant colony; but as we know from the ant rules, this is not the case.
- In fact, as you observe the model closely, you will note that sometimes ants operate almost at cross-purposes with other ants
- Ants create additional pheromone trails to farther food sources and distracting some of the ants that are currently harvesting the closer food source.
- When the food source has been completely consumed, the ants no longer lay pheromone;

Summary

- In this Module we have seen Ant's behavior model. Having run and used the model, it may seem obvious that these low-level rules can create these rich and optimal global patterns.

Module – 68

Lesson 68 : When Is ABM Most Beneficial?

Exploration & Exploitation

This optimal exploitation of food sources could be placed within a larger context.

In many ways, the colony of ants seems to balance exploration and exploitation (Dubins & Savage, 1976).

In any situation in which an entity is operating in an unknown environment, the entity must spend some time exploring the environment to understand how its actions affect its rewards, and sometimes exploiting the environment, that is taking actions that it knows have produced the best rewards in the past.

By allocating a large number of ants to exploit the current nearest food source while other ants continue to explore, the ant colony as a whole successfully balances exploration and exploitation.

Emergent Phenomenon

These “trails” that the ants build to the food source, the “optimal” behavior that they exhibit, and the “balance” between exploration and exploitation are not coded into anyone ant.

There is nothing that tells the ant to build a trail; there is nothing that tells the ant to go to the nearest food source first; there is nothing that tells some ants to explore while others exploit.

The “trails,” “optimal,” and “balance” behavior of the ants is not coded into any of the ants but is instead an emergent phenomenon of the model (Holland, 1998; Anderson, 1972; Wilensky & Resnick, 1999).

Module – 69

What Good Is an Ant Model?

- The Ant model is our first example of an agent-based model.
- Now that we have gone through the process of understanding how a model of ant foraging works,
- what knowledge have we gained from that process and how can we profitably use the model?

Eight Main Uses For Agent-based Models

- Description,
- Explanation,
- Experimentation,
- Providing Sources Of Analogy,
- Communication/Education,
- Providing Focal Objects Or Centerpieces For Scientific Dialogue,
- As Thought Experiments,
- Prediction.

A Model Is Descriptive

- A model is descriptive of a real-world system.
- Granted, it is a simplification of the real world and does not contain all of the details and inconsistencies that are present in the real world.
- Models that are not coarse-grained descriptions are useless as descriptions because they are indistinguishable from the real world and therefore do not assist in our understanding of complex systems (Korzybski, 1990)

Models Are Explanatory

- Models are explanatory in that they point out the essential mechanisms underlying a phenomenon.
- They can function as a proof that hypothesized mechanisms are sufficient to account for an observation.
- Models provide us with a proof of concept that something is possible.

- For example, once we built the Ants model and observed the results, we proved that an ant colony could exhibit characteristics such as “ trails, ” “ optimality, ” and “ balance ” without a centralized

Summary

- We have been discussing the Ants model and its results for some time now.
- This model is one example of an agent-based model.
- But there is a wide diversity of models that are called agent-based models

Module – 70

What Good Is an Ant Model?-II

Key function of ABMs

- A key function of ABMs is to explicate the power of the emergence.
- In general, it is difficult for people to understand how such simple rules can lead to complex observed phenomena, and ABMs make this connection explicit

Models Facilitate Experimentation

- Models can be run repeatedly to note variations in their dynamics and in their outputs.
- Some models have very little variation from run to run.
- Others exhibit path dependency and can therefore vary tremendously from run to run.

Experimentation

- Model parameters can be varied to see their effect on model behavior and outputs.
- For example, in the Ants model, we could change the evaporation rate of the pheromone and see what effect that has on the performance of the ants
- Alternatively, we could model the return to nest behavior in more detail, by placing a sun in the environment and have the ants calculate their path back based on the sun.

Effects of modifications on system

- Thus agent-based models enable us to easily examine different mechanisms and attributes of a system and see what effect those modifications have on the overall behavior of the system.

Model's Behavior Classification

- By varying these different attributes or parameters and observing their effect on the behavior of the system, we can classify model behavior into output regimes , or characteristic output behaviors of the model.

Models Provide Us With Analogies

- Since models are simplifications of reality, they enable us to find similarities with other such simplifications even if the modeled phenomena are apparently very different.
- In this way, we can apply reasoning gained in one domain of knowledge to other knowledge domains.
- For instance, in 1996, Schoonderwoerd, extending Dorigo's work on ant colonies as optimization devices, found that the problem of ants efficiently finding food sources is similar to the problem of packets on networks efficiently seeking out their destinations (Schoonderwoerd et al., 1996).

Summary

- We have discussed various benefits of Agent-based Models

Module – 71

What Good Is an Ant Model?-III

Communication And Education

- Agent-based models can be used as a vehicle for communication and education .
- We can show the model to people who have never seen an ant colony before and they can explore how ant colonies behave.
- Models provide us with an educational tool that encapsulates
- knowledge that may not be readily available from real world observation.
- Moreover, an agent based model allows us to expand beyond static knowledge, enabling learners to conduct experiments much as scientists do.
- If someone has a hypothesis as to how a particular mechanism works, they can implement that mechanism and see whether it can account for observed behavior.

Thought Experiments

- Models can sometimes present new phenomena that are not necessarily about some real-world phenomenon, but are thought experiments on possible computations.

Prediction

- A common conception of computer modeling is that its major purpose is prediction .
- It is true that we often use modeling to think about possible future scenarios, and to the extent that any modeling tool can be used for discussing the future, agent-based modeling can be used in this way

Predictions Depend On..

- However, like any modeling tool, an agent-based model's predictions depend on the accuracy of its input data.
- This is especially true when the events we are interested in are the results of complex systems, for which small changes in inputs can often lead to very different results.

Model To Predict The Outcome Of A System,

- Many times, while modelers may claim to want to use a model to predict the outcome of a system,
- Such as an ant colony, they actually use the model to describe past patterns of behavior,.

Prediction

- Such as what food source was first eaten, and to explain future patterns that might arise, such as how ants might move around on the landscape.
- Sometimes modelers start off motivated by generating predictions but end up finding greater power for explanation and description

Summary

- We have been discussing the Ants model and its results for some time now.
- This model is one example of an agent-based model.
- But there is a wide diversity of models that are called agent-based models

What is Agent-based Modeling?

The Reality of Agents

- Agents vs. Objects
- Why agent-based modeling is so intuitive?

What is an Agent?

An agent is an autonomous individual or object with particular properties, actions, and possibly goals.

What is the environment?

- The environment is the landscape on which agents interact and can be geometric, network-based, or drawn from real data.
- The interactions that occur between these agents or with the environment can be quite complex.

What about the interactions?

- The interactions that occur between these agents or with the environment can be quite complex.
- Agents can interact with other agents or with the environment
- Not only can the agent's interaction behaviors change in time, but so can the strategies used to decide what action to employ at a particular time.

ABM vs. Other Modeling Forms

- The most common form of scientific models is the equation form.
- One distinction is that because ABM models individuals it can model a heterogeneous population
- Equational models typically must make assumptions of homogeneity.
- Furthermore, when you model individuals, the interactions and results are typically discrete and not continuous.
- Continuous models do not always map well onto real-world situations.
- For instance, equation-based models of population dynamics treat populations as if they are continuous quantities

- In fact they are populations of discrete individuals.
- When simulating population dynamics it is very important to know if you have a sustainable population.
- After all, a wolf population cannot continue if there are fewer than two wolves left
- In reality, a millionth of a wolf cannot exist and certainly cannot reproduce, but it can result in increased wolf population in EBM.
- The mismatch between the continuous nature of EBM and the discrete nature of real populations causes this “ nano-wolf ” problem
(Wilson, 1998).

Summary

- Here we have compared ABMs with EBM

Module – 73

Randomness vs. Determinism

Randomness

- One important feature of agent-based modeling, and of computational modeling in general, is that it is easy to incorporate randomness into your models

Benefits of Randomness

- Many equation-based models and other modeling forms require that each decision in the model be made deterministically.
- In agent-based models this is not the case
- Instead, the decisions can be made based on a probability.

Example of Randomness

- For instance, in the Ants model, as the ants move around the landscape, their decisions are not completely determined; instead, at each time step they change their heading a small amount based on a random number.

Ant Model and Randomness

- As a result, each ant follows a unique, irregular path.

- In reality, ants might be affected by small changes in elevation, the presence or absence of twigs and stones, and even the light of the sun.

Other cases

- Finally, there are often times when we simply do not know enough about how a complex system works in order to build a completely deterministic model.
- In many of these cases the only type of model that we can build is a model with some random elements. Agent-based modeling
- Or other modeling forms that allow you to incorporate random features are essential to studying these kinds of systems.

Summary

- Here, we have analyzed the benefits of randomness

Module – 74

When is ABM beneficial?

ABM Benefits

- Agent-based modeling has some benefits over other modeling techniques, but, as with any tool, there are contexts in which it is more useful than others.
- ABM can be used to model just about any natural phenomenon (e.g., you could describe any phenomenon by describing the interaction of its subatomic particles).
- However, there are some contexts for which the cost of building an ABM exceeds the benefits, and there are other times when the benefits are extraordinary given the costs.

Examples

- Some problems with large number of homogenous agents are often better modeled (i.e., they will provide more accurate solutions to aggregate problems faster) using an aggregate solution like mean field theory or system dynamics modeling (Oppen & Saad, 2001 ; Forrester, 1968).
- For instance, if you are concerned about the temperature in a room, then tracking every individual molecule and its history is not necessary.

General Rules

- On the other hand, if a problem has only a handful of interacting agents, then you usually do not need to bring to bear the full power of ABM and instead can write detailed equations describing the interaction — two billiard balls colliding, for example, does not require ABM

- As a rule of thumb, agent-based models are most useful when there are a medium number (tens to millions) of interacting agents (Casti, 1995).

Summary

- Here, we have analyzed the benefits of randomness

Module – 75

Tradeoffs of ABM

ABM Tradeoffs

- Agent-based modeling has some benefits over other modeling techniques, but, there are also tradeoffs
- Sometimes agent-based modeling is not the right tool for the job.
- For example, ABM can be computationally intensive.
- Simulating thousands or millions of individuals can require great computing power.
- Equation-based models, by contrast, are often very simple to run and essentially just require repetitive mathematical calculations.
- This is true only for simple equation-based models
- Numerically solving complicated equation-based models may take as much computational time as agent-based models.

Still ABM may be Better

- The computational expense of running an ABM is a price one pays for having the benefits of rich individual-level data.
- The additional computational power needed for running ABMs is the same power that allows the tracking and development of rich histories of individuals.

Example

- For example, in the Ants model we could write a simple equation that describes the rate at which food is gathered based on its distance from the nest.
- However, through ABM, we are able to observe the behavior of the individual ants and
- understand how they form trails to the food source.

Other cases

- ABM does require us to gain an understanding of the microbehavior of a system, but without modeling the microbehaviors, ABM would not provide as elaborate a model as it does.
- While ABM does require us to know or guess about the individual-level mechanisms operating, it does not require knowledge of the aggregate level mechanism.

Summary

- Here, we have analyzed the tradeoffs of ABMs

Module – 76

Tradeoffs of ABM

- Agent-based modeling has some benefits over other modeling techniques, but, there are also tradeoffs
- Sometimes agent-based modeling is not the right tool for the job.
- For example, ABM can be computationally intensive.
- Simulating thousands or millions of individuals can require great computing power.
- Equation-based models, by contrast, are often very simple to run and essentially just require repetitive mathematical calculations.
- This is true only for simple equation-based models
- Numerically solving complicated equation-based models may take as much computational time as agent-based models.

Still ABM may be Better

- The computational expense of running an ABM is a price one pays for having the benefits of rich individual-level data.
- The additional computational power needed for running ABMs is the same power that allows the tracking and development of rich histories of individuals.

Example

- For example, in the Ants model we could write a simple equation that describes the rate at which food is gathered based on its distance from the nest.
- However, through ABM, we are able to observe the behavior of the individual ants and
- understand how they form trails to the food source.

Other cases

- ABM does require us to gain an understanding of the microbehavior of a system, but without modeling the microbehaviors, ABM would not provide as elaborate a model as it does.
- While ABM does require us to know or guess about the individual-level mechanisms operating, it does not require knowledge of the aggregate level mechanism.

Summary

- Here, we have analyzed the tradeoffs of ABMs

Module – 77

Creating Simple Agent-Based Models: Life-I

Overview

- Toy Models are very important
- “Real world” is better understood as a thought experiment
- Game of Life is an interesting first model

Toy Models?

- Humans learn by using toys
- Abstract Models can be made for the real world



Thought Experiments

- Science is filled with thought experiments



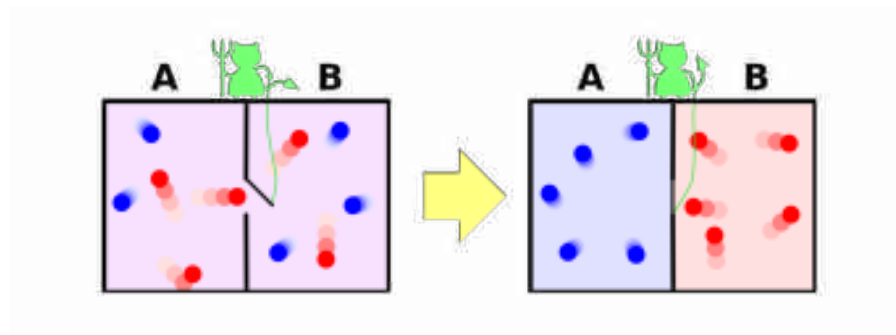
Physics: Thought Experiments

- Physics



More Thought Experiments

- Maxwell's Demon



Game of Life: History

- British Mathematician John Horton Conway – 1970
- A cellular automaton
- It was made popular by a columnist “Martin Gardner”



http://adeptis.ru/vinci/m_part3_3.html

Game of Life

- The game is played on a large grid, such as a checkerboard or graph paper.
- Let's say we are playing on a square grid with 51 squares 1 (or “cells”) on a side.
- Each cell can be either “alive” or “dead.”
- This is called the “state” of the cell.
- Every cell is surrounded by eight “neighbor” cells.
- The grid is considered to “wrap around” so that a cell on the left edge has three (3) neighbor cells on the right edge and, similarly, a cell on the top edge has three (3) neighbor cells on the bottom edge.
- There is a central clock. The clock ticks establish a unit of time.
- In the game of Life, the unit of time is called a generation.
- More generally, in agent-based models, the unit of time is referred to a tick.

- Whenever the clock ticks, each cell updates its state

Rules of Life

- (1) If the cell has less than two (2) alive neighbors, it dies.
- (2) If it has more than three (3) alive neighbors, it also dies
- (3) If it has exactly two (2) alive neighbors, the cell remains in the state
- (4) If it has exactly three (3) alive neighbors, the cell becomes alive if it is dead, or stays alive if it is already alive

Summary

- We have seen how Toy Models can be an important exercise
- We have looked at the history and rules of the Game of Life

Module – 78

Life-II

Overview

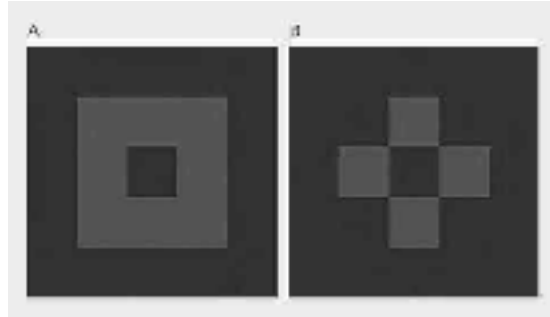
- Game of Life is interesting to code
- It is developed using NetLogo patches
- Overview of implementation

Game of Life

- Game of Life involves a Cellular Automaton
- Every cell is a square
- Every cell has neighbors

Cell: Neighbors

- A. Moore Neighborhood
- B. Von Neumann Neighborhood



Understanding the rules

- Each cell checks the state of itself and its eight surrounding neighbors and then sets itself to either alive or dead.
- If there are less than two alive neighbors, then the cell dies.
- If there are more than three alive neighbors, the cell dies.
- If there are 2 alive neighbors, the cell remains in the state it is in.
- If there are exactly three alive neighbors, the cell becomes alive.
- This is done in parallel and continues forever.
- patches-own [live-neighbors]

Code of Life

```
to setup
  clear-all
  ask patches [
    ;; create approximately 10% alive patches
    set pcolor blue - 3 ;; dark blue cells are dead
    if random 100 < 10 [
      set pcolor green ;; green cells are alive
    ]
  ]
  reset-ticks
end
```

```
(to go
  ask patches [
    ;; each patch counts its number of green neighboring patches
    ;; and stores the value in its live-neighbors variable
    set live-neighbors count neighbors with [ pcolor = green ]
  ]
  ask patches [
    ;; patches with 3 green neighbors, turn (or stay) green
    if live-neighbors = 3 [ set pcolor green ]
    ;; patches with 0 or 1 green neighbors turn (or stay) dark blue (from isolation)
    if live-neighbors = 0 or live-neighbors = 1 [ set pcolor blue - 3 ]
    ;; patches with 2 or more green neighbors turn (or stay) dark blue (from overcrowding)
    if live-neighbors >= 4 [ set pcolor blue - 3 ]
    ;; patches with exactly 2 green neighbors keep their color
  ]
  tick
end)
```

Summary

- We have seen the importance of toy models
- We have examined game of life
- We have seen how to start coding in NetLogo using patches

Module – 79

Life-III

Overview

- Observing how Game of Life Works
- Examining emergent patterns

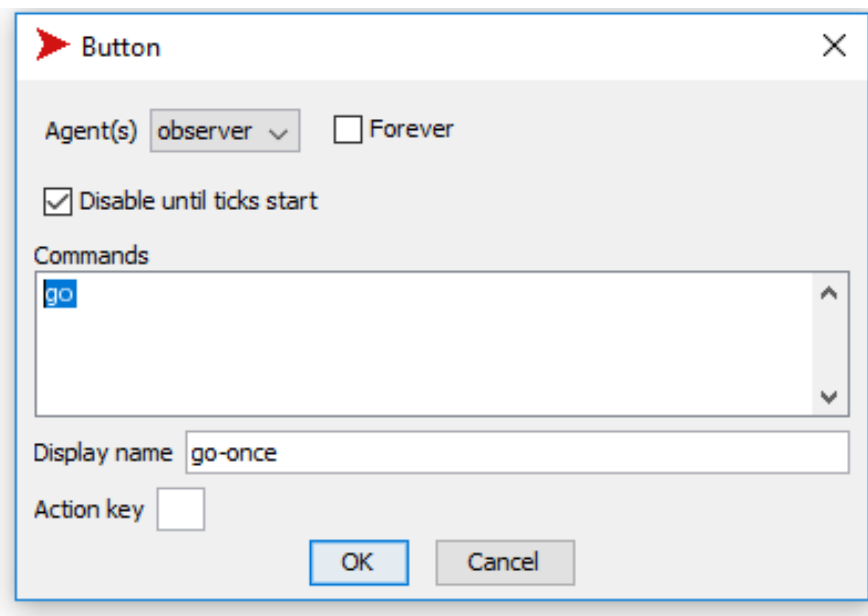
User Interface

- Adding a Button



Go-Once button

- Add another button

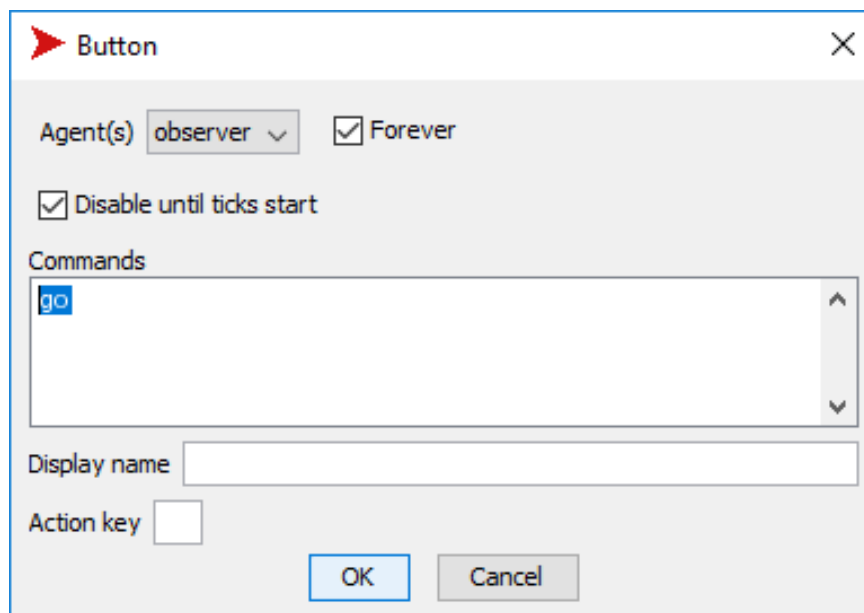


The screenshot shows a 'Button' dialog box with the following settings:

- Agent(s):** A dropdown menu showing 'observer'.
- Forever:** An unchecked checkbox.
- Disable until ticks start:** A checked checkbox.
- Commands:** A text area containing the command 'go'.
- Display name:** A text field containing 'go-once'.
- Action key:** An empty text field.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

Go button

- Add yet another button



The screenshot shows a 'Button' dialog box with the following settings:

- Agent(s):** A dropdown menu showing 'observer'.
- Forever:** A checked checkbox.
- Disable until ticks start:** A checked checkbox.
- Commands:** A text area containing the command 'go'.
- Display name:** An empty text field.
- Action key:** An empty text field.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

Understanding the Output

Simulation of Life

- Emergent Pattern: Glider



Output

- Another emergent pattern



Summary

- We have examined outcome from the game of life
- We have seen how emergent patterns emerge unexpectedly

Module – 80

Simple Economy-I

Overview

- Understand how economy model works
- Learn about more ABM concepts

Simple Economy

- In spite of being primarily of interest to Computer Sciences, there has been a growing interest in the use of ABM in the social sciences.

Why ABM in Other Sciences?

- Agent-based methods may be particularly valuable in the social sciences
- Mathematical descriptions often do not offer sufficient descriptive power
- Heterogeneous agents help a lot in developing an understanding

Organizations

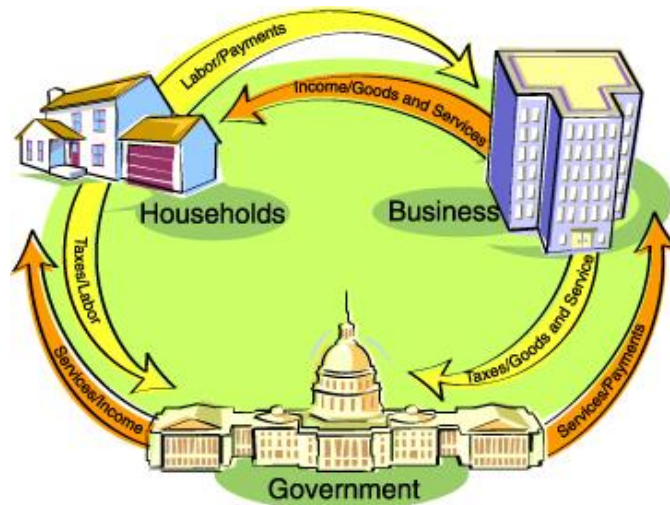
- Complex Systems Social Sciences Association (CSSSA)
- World Congress on Social Simulation (WCSS)

Others

- American Education Research Association (AERA),
- Marketing Science,
- Eastern Economics Association
- American Association of Geographers (AAG)
- These have hosted sessions on the intersection of agent-based modeling and social science.

Economics Using ABM

- Economies consist of heterogeneous actors such as buyers and sellers
- There is a natural mapping of ABM methods to economics.
- Figure credit: Wikimedia



- In 1996, the economists, Josh Epstein and Robert Axtell published a book
- It depicted a world titled SugarScape
- This world was populated by economic agents

Simple Economy Description

- A very simple economics model that has some surprising results.
- Suppose you have a fixed number of people, say 500, each starting out with the same amount of money, say \$100.
- At every tick, each person gives one of his or her dollars to any other person at random.
- What will happen to the distribution of money?
- An important constraint is that the total amount of money remains fixed, so no one can have less than zero money.
- If you run out of money, you cannot give any away until you get some back.
- Refining the question a little further:
- Is there a stable limiting distribution of the money?
- If so, what is it? For instance, will all the wealth be concentrated in a few hands or will it be equitably distributed?

Simple Economy Intuition

- Intuitively, we can think about the model
- There should be a limiting flat distribution
- The reasoning:

- No person starts off with an advantage
- Selection of people to whom money is transferred is random,
- No person should have much of an advantage over any others.
- Thus, the resulting wealth distribution should be relatively flat
- Everyone should wind up with roughly the same amount of money he or she started with.

Alternate Intuition

- Wealth will be normally distributed

Summary

- We have examined the role of ABM in the social science
- We have seen how economics can benefit from ABM
- We have seen the basis of the “Simple Economy” model

Module – 81

Simple Economy-II

Overview

- Go in more detail about the “Simple Economy” model
- Understand the code for Simple Economy

Simple Economy

```
turtles-own [ wealth ]

to setup
  clear-all
  create-turtles 500 [
    set wealth 100
    set shape "circle"
    set color green
    set size 2

    ;; visualize the turtles from left to right in ascending order of wealth
    setxy wealth random-ycor
  ]
  reset-ticks
end
```

```

≡ to go
  ;; transact and then update your location
  ask turtles with [ wealth > 0 ] [ transact ]
  ;; prevent wealthy turtles from moving too far to the right
  ask turtles [ if wealth <= max-pxcor [ set xcor wealth ] ]
  tick
end

≡ to transact
  ;; give a dollar to another turtle
  set wealth wealth - 1
  ask one-of other turtles [ set wealth wealth + 1 ]
end

≡ to-report top-10-pct-wealth
  report sum [ wealth ] of max-n-of (count turtles * 0.10) turtles [ wealth ]
end

≡ to-report bottom-50-pct-wealth
  report sum [ wealth ] of min-n-of (count turtles * 0.50) turtles [ wealth ]
end

```

Summary

- We have examined the implementation of the Simple Economy Model
- We have seen how turtles can be used to develop a model

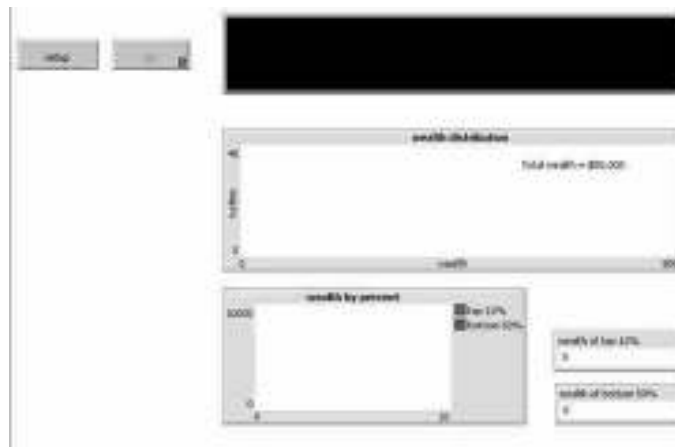
Module – 82

Simple Economy-III

Overview

- Understanding the User Interface of Simple Economy
- Understand the execution of Simple Economy

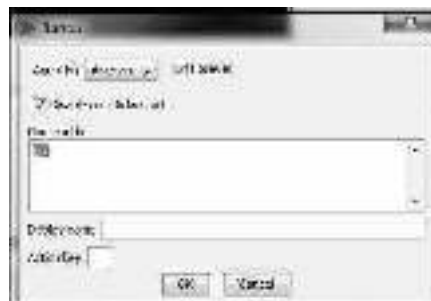
Simple Economy



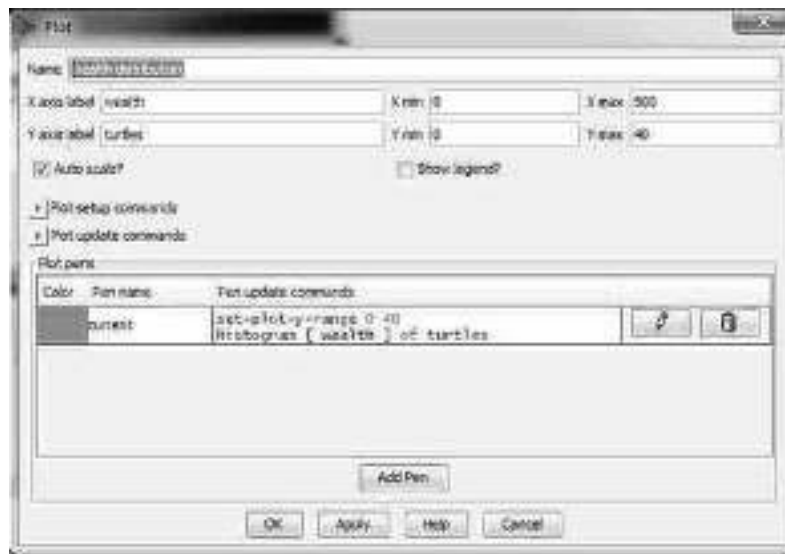
- First we add a button



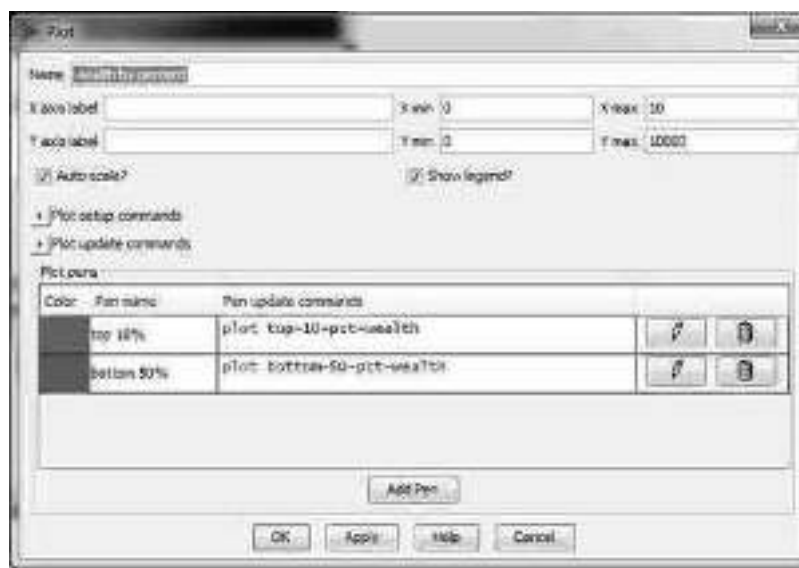
- Another button



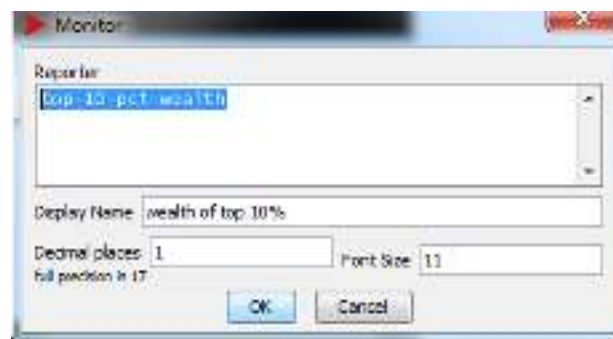
- Now, we add a plot



- Now, we add another plot



- Now, we add a monitor



- Now, we add another monitor



Summary

- We have examined the design of the User interface for the Simple Economy Model

Module – 83

Simple Economy-IV

Overview

- Understand the Complexity dynamics of the Simple Economy Model

Simulation Run

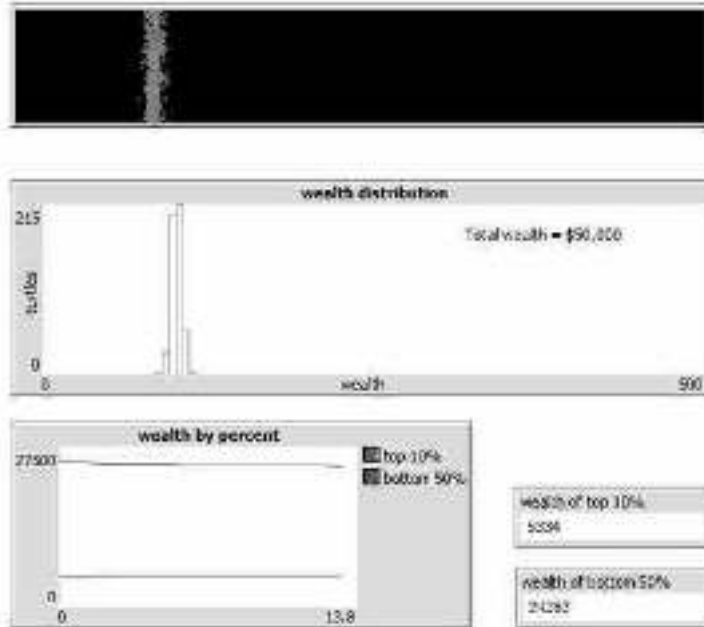
- Let us see how this works in NetLogo

Running the Model

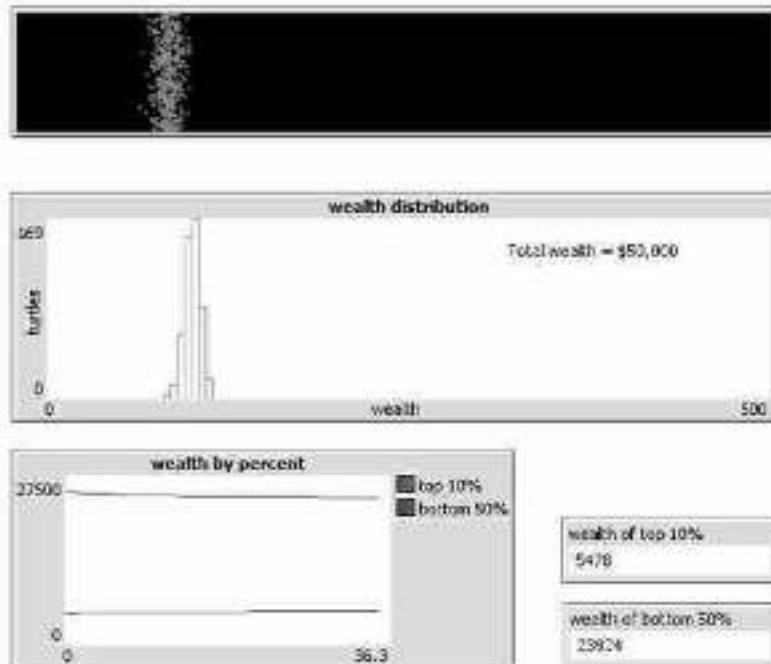
- Let us see the initial condition



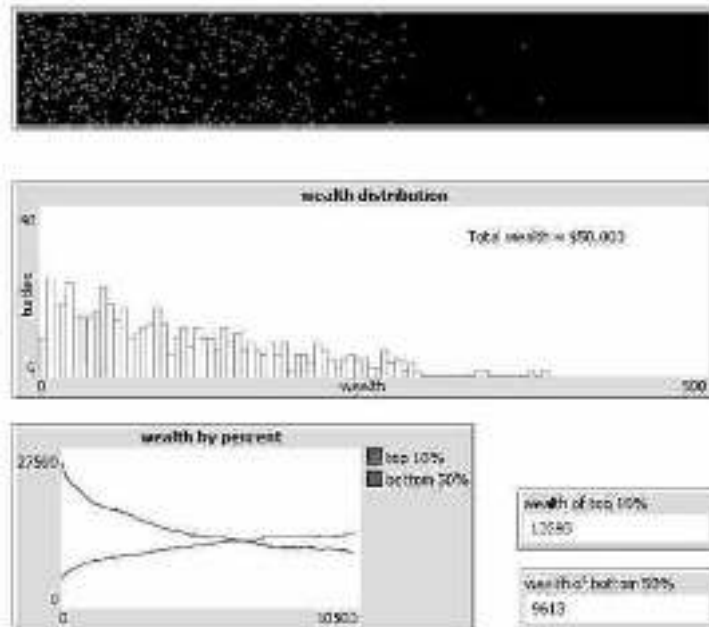
- After running



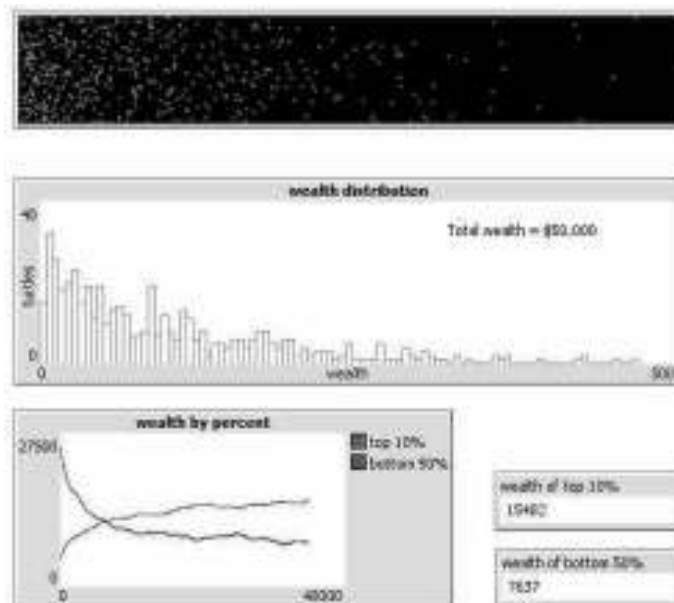
- After running



- After long run: 10,000



- After long run: 35,000



Understanding Results

- We can see that the distribution is not at all flat.
- There are a few very wealthy individuals and many poor agents.
- At this time step in this model run, the wealth of the top 10 percent of the agents is a total of \$12,633, or an average of \$253 per person
- Whereas the total wealth of the entire bottom 50 percent is only \$10,166, or an average of \$41 per person.
- The crossover point at which the wealth of the top 10 percent of all agents exceeded the wealth of the bottom half, or 50 percent, of all agents was at 5,600 ticks.
- If we run it for 25,000 more ticks, the gap continues to grow and now the top 10 percent have more than twice as much in total (and more than 10 times as much per individual) than the bottom 50 percent.
- The distribution of money eventually converges to a stationary distribution.
- This distribution has been shown to be exponential, which means that there is great inequality in monetary wealth.
- The key condition that creates this stationary distribution is the conservation of money.

Summary

- We have examined the Execution of the Simple Economy Model
- We have seen how surprising results can appear in relatively simple models

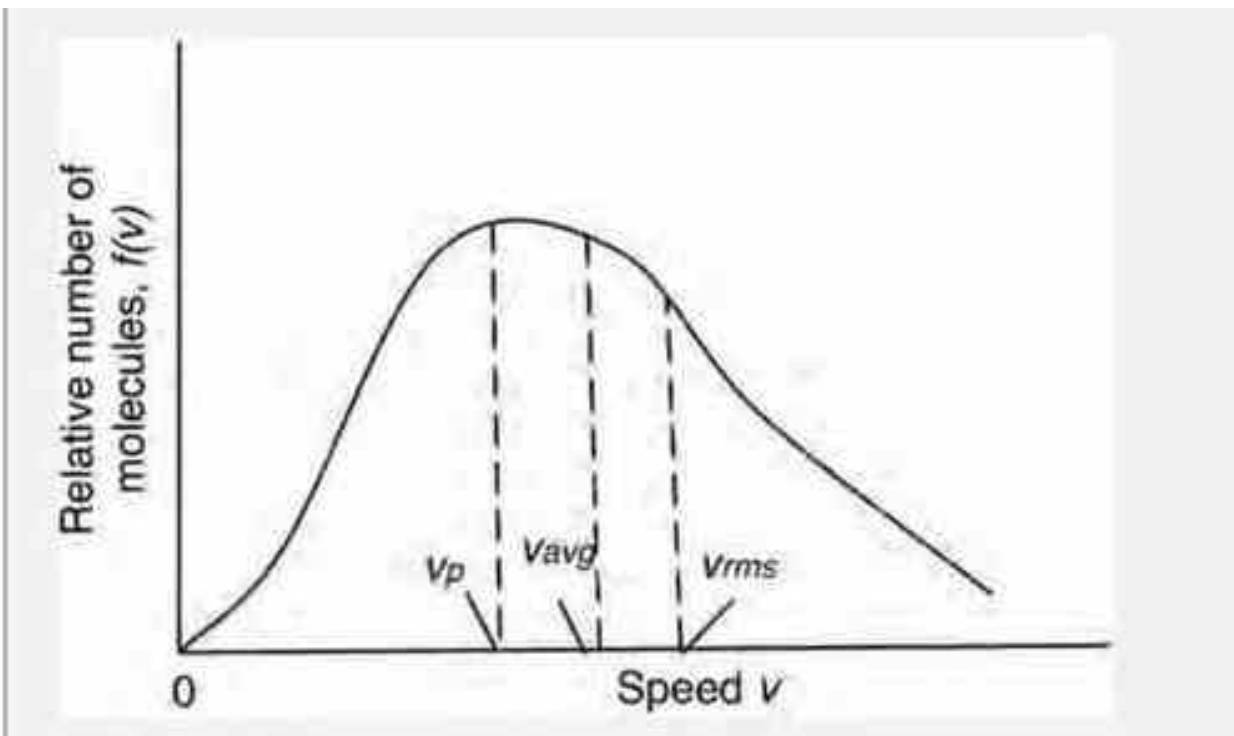
Econophysics

Overview

- Understand the basis of Complexity dynamics of the Simple Economy Model

The basis

- The distribution we have just seen in the Simple Economy model is an instance of a broader class of models
- In these models a conserved quantity is exchanged in a closed system.
- The great nineteenth-century Austrian physicist Ludwig Boltzmann first examined such models when he worked out the physics of gas molecules in a box.
- He imagined these molecules as “ billiard balls ” moving and colliding.
- When they collide, they change speed and direction,
- Because of the conservation of energy, the total energy after a collision remains the same.
- He derived an equation, now known as the Boltzmann equation , for the distribution of the energies of the billiard balls
- He solved the equation to show that their energies formed a stationary exponential distribution
- It is known as Boltzmann distribution, the Boltzmann-Gibbs distribution, and the Maxwell-Boltzmann distribution.
- The Boltzmann distribution is a central component of statistical mechanics.
- Statistical mechanics uses probability theory to understand the thermodynamic behavior of systems composed of a large number of particles.
- In many ways, agent-based modeling can be seen as extending this perspective to content domains beyond physics.



Recent Work

- In recent years, theories and methods from physics have been applied to economics.
- The physicist Gene Stanley (1996) coined the term "econophysics" for this interdisciplinary research field.
- In their 2000 paper, Dragulescu and Yakovenko describe several variants of the kind of monetary exchange that we have explored in this model.
- They show that if agents are allowed to go into debt (subject to a fixed credit limit), there is still a stationary exponential distribution
- However in this case it results in even greater inequality.
- A more specific term for what we are modeling is monetary wealth.

Dragulescu and Yakovenko caution:

It is tempting to identify the money distribution with the distribution of wealth. However, money is only one part of wealth, the other part being material wealth. Material products have no conservation law: They can be manufactured, destroyed, consumed, etc. Moreover, the monetary value of a material product (the price) is not constant. The same applies to stocks, which economics textbooks explicitly exclude from the definition of money. So, in general, we do not expect the Boltzmann-Gibbs law for the distribution of wealth.

Summary

- We have examined the basis of complex results in the Simple Economy Model
- We have seen how concepts from one domain can be used in a completely different domain

Module – 85

NetLogo: Agentsets and Lists

Overview

- Understand some commonly used NetLogo concepts

Understanding Complex Results

- Understanding complex results often requires advanced modeling techniques
- E.g. in Simple economy, we have seen complex results
- Some of these results need correlation with the real world
- Empirical studies have shown that wealth distribution in most countries is not distributed exponentially,
- Instead it follows a power law known as the Pareto distribution (Pareto, 1964).
- This distribution is even more unequal than an exponential distribution.
- Power law distributions arise in a surprising number of contexts
- They are typically associated with preferential attachment processes (Barabási, 2002)
- The amount received is proportional to the existing amount.

Lists and Agentsets

- To model such complexity we often need advanced programming concepts
- Some of the most used constructs (data structures) in NetLogo are:
- Agentsets
- Lists
- An agentset is a set of agents.
- Agentsets can have turtles, patches, or links, but they cannot mix agent types.

- You can ask an agentset to perform some commands.
- NetLogo comes with three special agentsets built in: " turtles, " " patches, " and " links"
- What makes agentsets so

powerful is that you can create your own agentsets

- For example, an agentset of all the red turtles,
- Or an agentset of all the patches in the upper right quadrant.
- Agentsets are always in random order.
- So if you ask an agentset, several different times, to execute some commands, each time the order in which the agents take turns executing the commands will be different.

- Lists are ordered collections of data.
- We can have a list of numbers
- A list of words, even a list of lists, also even a list of agents.
- Since they are in a list, these agents will have a particular order, so you can use the list to execute their commands in any order you might like.

Summary

- We have examined the basis of complex results in the Simple Economy Model
- We have seen how concepts from one domain can be used in a completely different domain

Module – 86

Exploring and Extending Agent-Based Models

Overview

- Understand how we do not have to develop models from scratch
- Understand how existing models can be extended easily

Part of the inhumanity of the computer is that, once it is competently programmed and working smoothly, it is completely honest.

—Isaac Asimov

The perfect journey is never finished, the goal is always just across the next river, round the shoulder of the next mountain. There is always one more track to follow, one more mirage to explore.

—Rosita Forbes

Sailors on a becalmed sea, we sense the stirring of a breeze.

—Carl Sagan

Characteristic Features

- **There are four characteristic features of agent-based modeling:**

1. Simple rules can be used to generate complex phenomena
2. Randomness in individual behavior can result in consistent patterns of population behavior
3. Complex patterns can “ self-organize ” without any leader orchestrating the behavior
4. Different models emphasize different aspects of the world

- Simple rules can be used to generate complex phenomena
- Many of the models that we will look into have very simple rules that do not require complex mathematical formulas or a deep understanding of the knowledge domain that they are attempting to model.
- Nonetheless, they are able to reproduce complex phenomena that are observed in the real world.
- For instance, a model of fire spread may have only a simple rule to describe fire spread from one tree to another
- Still it may have interesting things to say about how likely a fire is to spread across an entire forest.

- Randomness in individual behavior can result in consistent patterns of population behavior
- It is easy to misunderstand this
- E.g. Seeing an ordered population level behavior such as a flock of birds, it is easy to assume that there must be deterministic processes that govern the behavior of the individuals (Wilensky & Resnick, 1999)
- Many times the individual level rules are quite simple
- The rules do not necessarily tell the bird where to position itself in the flock.
- Instead, the rules often contain a certain amount of nondeterminism and are robust to perturbations in the initial conditions.
- Despite the stochastic nature, they can still result in the generation of predictable high-level behavior like the flocking of birds.

Summary

- We have a couple of characteristic features of ABM

Module – 87

Characteristic Features of ABM

Overview

- Understand some more characteristic features of ABMs

No Leader Needed!

- Complex patterns can " self-organize " without any leader orchestrating the behavior
- Nature has surprises in store
- A population of individuals, each following very simple rules, can " self-organize, " generating complex and beautiful patterns without any orchestrator or centralized controller

Different Models – different aspects

- Different models emphasize different aspects of the world.
- Finishing a model does not mean that the work is all done
- Even after we have completed a good working model of a particular phenomenon, we are not done with the modeling process.

- Every model foregrounds certain aspects of the world
- It also backgrounds other aspects.
- This is the idea behind abstraction and narrowness of focus
- There can be many models of the same phenomenon
- They each may have something interesting to say about the way the world works
- For instance, a model of residential location preferences may emphasize
- How likely is it for a person is to move into a neighborhood based on how she likes the other people in the neighborhood.
- Such a model may have very interesting things to say about how urban populations develop,
- However it may say nothing about school districts, location of retail businesses, or development of parks
- All of these are also affected by residential location preferences.

Summary

- We have examined some very interesting characteristic features of ABM
- Understanding these features allows us to reuse and extend existing models rather than develop them from scratch

Module – 88

Fire Model

Overview

- Understand the dynamics of the Forest Fire Model

Forest Fire

- Credit: Pixabay



- Why study Forest fires?
- Forest fires are inherently an extremely complex phenomena
- Some countries have developed a complex mathematical index to predict them.
- The mathematical model (Fire Weather Index) involves a very large number of variables

Tipping Points

- Different models emphasize the concept of tipping points.
- Threshold or tipping point

Forest Fire

- A tipping point occurs when a small change in one parameter results in a large change in an outcome.

Good example – forest fire

- This simple model is highly sensitive to one parameter.
- When observing the resultant outcome of whether or not a fire will burn from one side of a forest to another (percolate), the output is mainly dependent on the percentage of the ground that is covered by trees
- As this parameter increases, there will be little to no effect on the system for a long time,
- Then all of a sudden the fire will leap across the world.
- This is a " tipping point " in the system

Benefits of Knowing

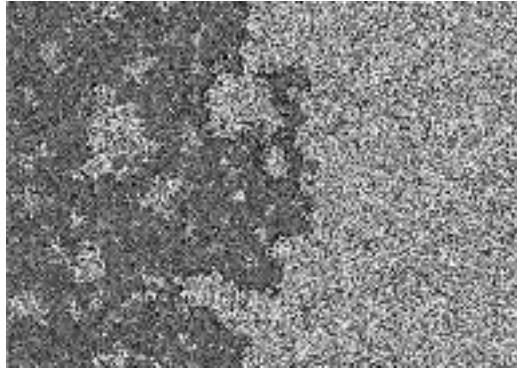
- First, if you know a system exhibits a tipping point, you know that continuing to put effort into the system, even if you are not seeing any results yet, may yet bear fruit.
- Second, if you know where the tipping point is, and if you know how close you are to it, then you can determine whether or not it is worth putting additional effort into the system.
- If far away from the tipping point, then it might not be worthwhile trying to change the state of the system
- Whereas if close to the tipping point it may take only a small amount of effort to make a big change in the state of the system.

Forest Fire Model

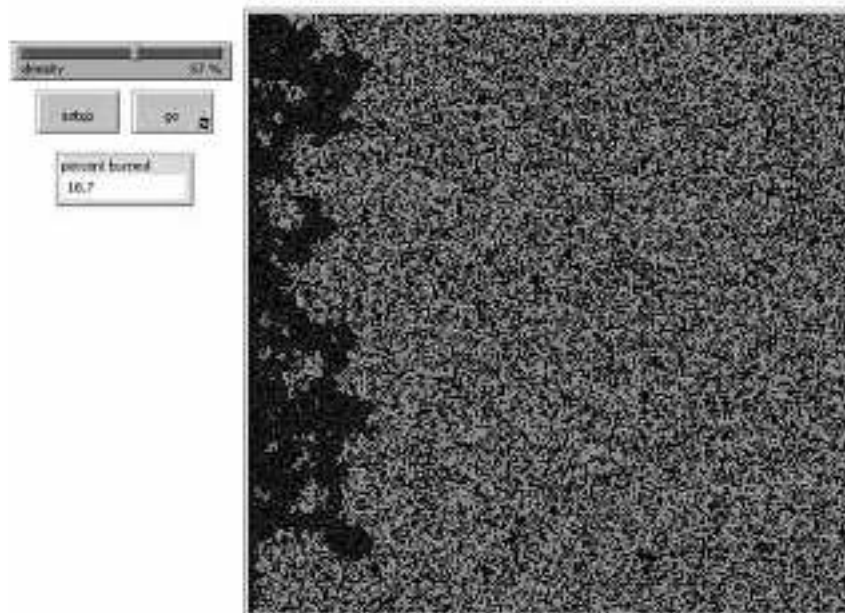
- The Fire model arose from a number of independent efforts to understand percolation phenomena.
- In percolation, a substance (such as oil) moves through another material (such as rock), which has some porosity.

Forest Fire Model

- Percolation
- Credit: Wikimedia Commons



Forest Fire Model



Summary

- We have examined the basics of Forest Fire model

Module – 89

Fire Model: Basic Model

Overview

- Understand the code of the Forest Fire Model

Forest Fire Model

```
globals [
  initial-trees ;; how many trees (green patches) we started with
]

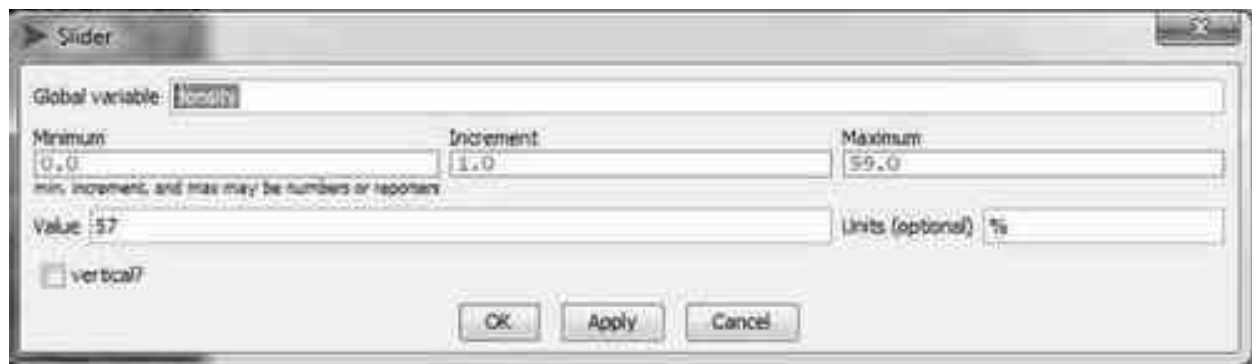
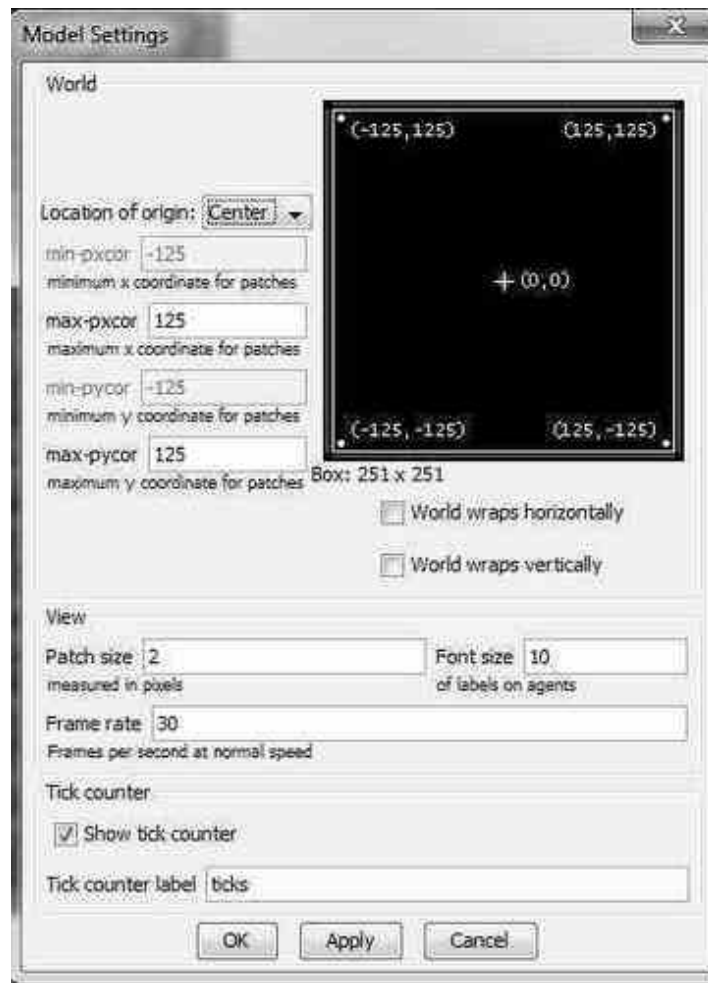
to setup
  clear-all
  ;; make some green trees
  ask patches [
    if (random 100) < density [
      set pcolor green
    ]
    ;; make a column of burning trees at the left-edge
    if pxcor = min-pxcor [
      set pcolor red
    ]
  ]
  ;; keep track of how many trees there are
  set initial-trees count patches with [pcolor = green]
  reset-ticks
end
```

```

to go
  ;; stop the model when done
  if all? patches [ pcolor != red ] [
    stop
  ]
  ;; ask the burning trees to set fire to any neighboring non-burning trees
  ask patches with [ pcolor = red ] [ ;; ask the burning trees
    ask neighbors4 with [pcolor = green] [ ;; ask their non-burning neighbor trees
      set pcolor red ;; to catch on fire
    ]
    set pcolor red - 3.5 ;; once the tree is burned, darken its color
  ]
  tick ;; advance the clock by one "tick"
end

```

Forest Fire



Button

Agent(s) ☐ Forever

☐ Disable until ticks start

Commands

Display name

Action key

OK Cancel

Button

Agent(s) ☒ Forever

☒ Disable until ticks start

Commands

Display name

Action key

OK Cancel

Monitor

Reporter

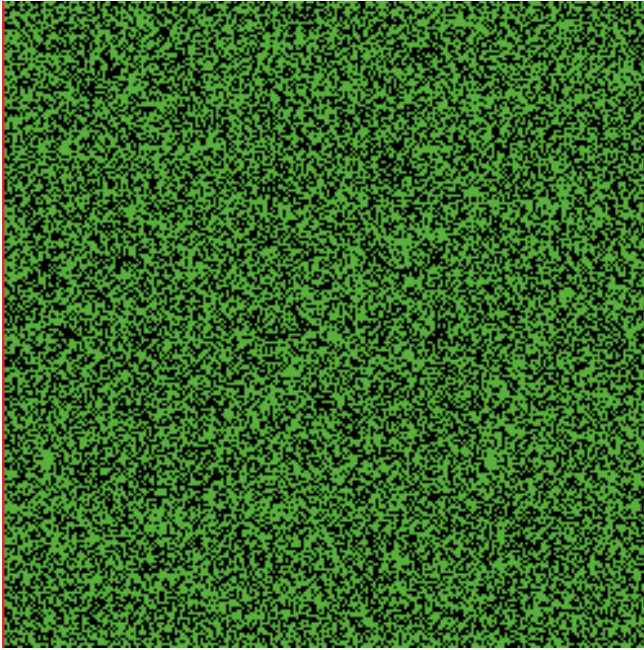
Display Name

Decimal places Font Size

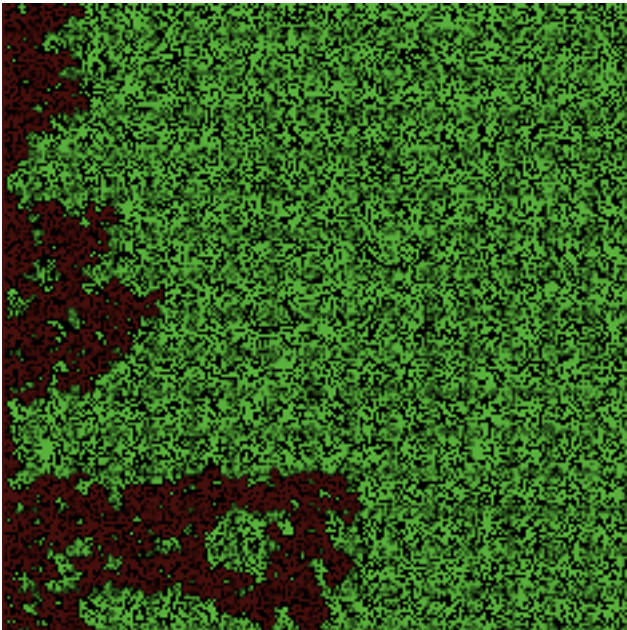
full precision is 17

OK Cancel

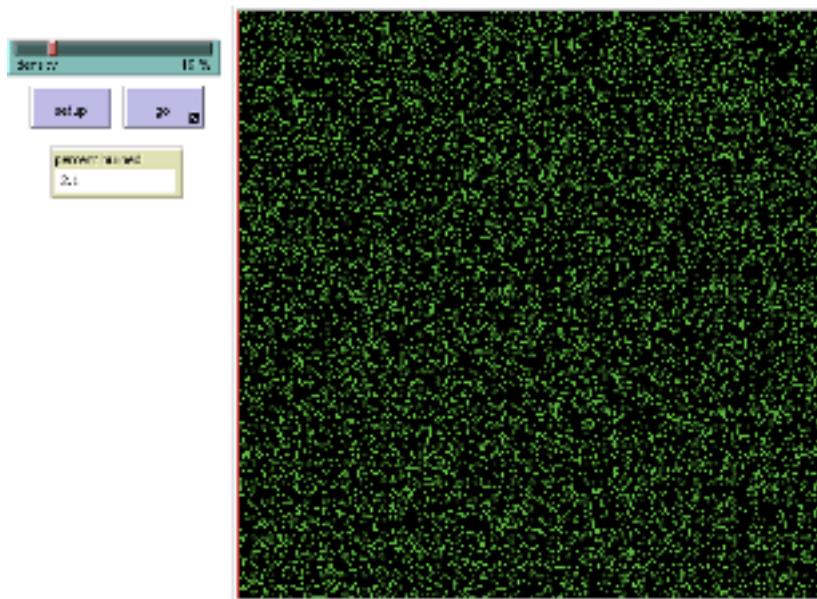
Forest Fire: Initial



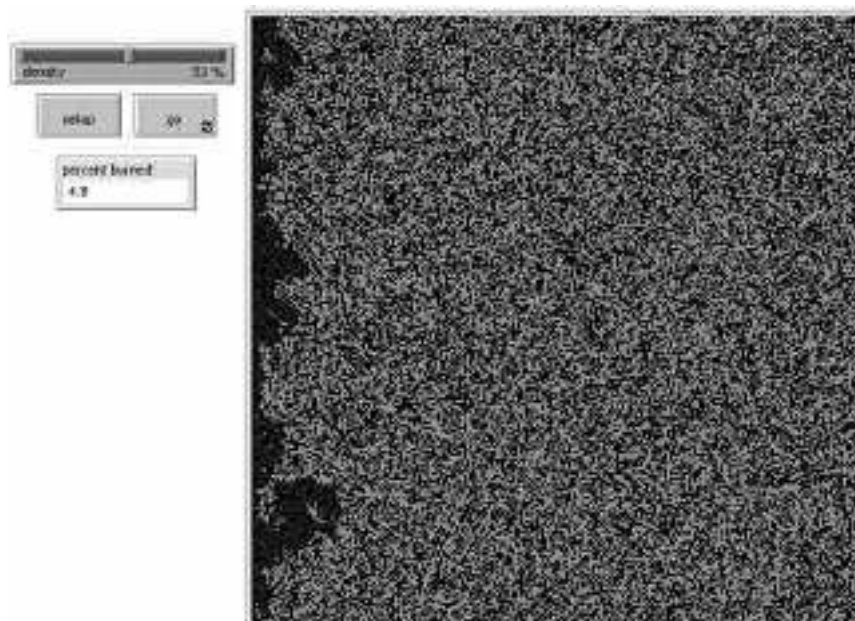
Forest Fire: Final



Another Run: 19% cover



Another Run: 53% cover



Summary

- We have examined the code and various runs of the Forest Fire model

Module – 90

First Extension: Probabilistic Transitions

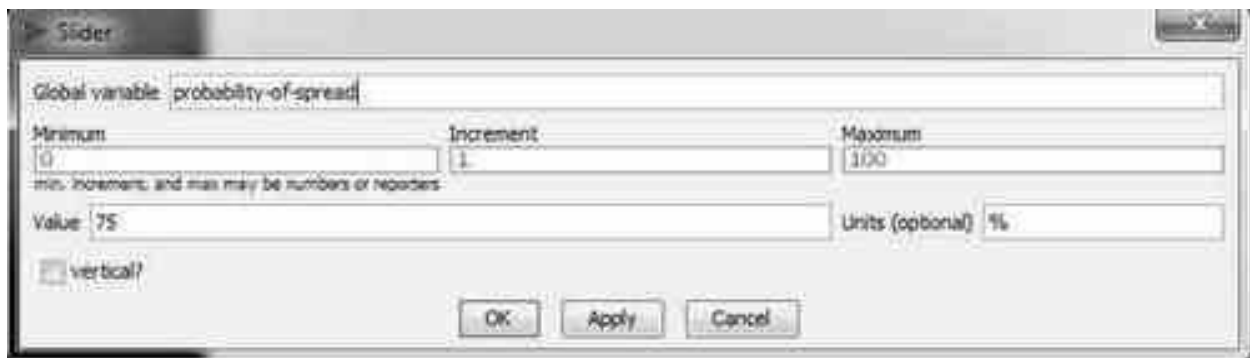
Overview

- Make changes in the original model to make it more realistic

Fire in the Real World

- Fire is not simple at all
- In the real world, it rather it jumps to another tree based on a variety of factors such as wind
- E.g. the type of wood, even how close the branches are to each other.
- See more complex extended models in the work given here:
<https://tinyurl.com/muazniazi>

Fire Model Extension



The image shows a 'Slider' dialog box from the NetLogo software. It is used to set the 'probability-of-spread' global variable. The 'Global variable' field contains 'probability-of-spread'. The 'Minimum' is set to 0, 'Increment' is 1, and 'Maximum' is 100. A note below these fields states: 'min, increment, and max may be numbers or reporters'. The 'Value' is currently set to 75, and the 'Units (optional)' are set to '%'. There is a checkbox for 'vertical?' which is currently unchecked. At the bottom are 'OK', 'Apply', and 'Cancel' buttons.

Model Code

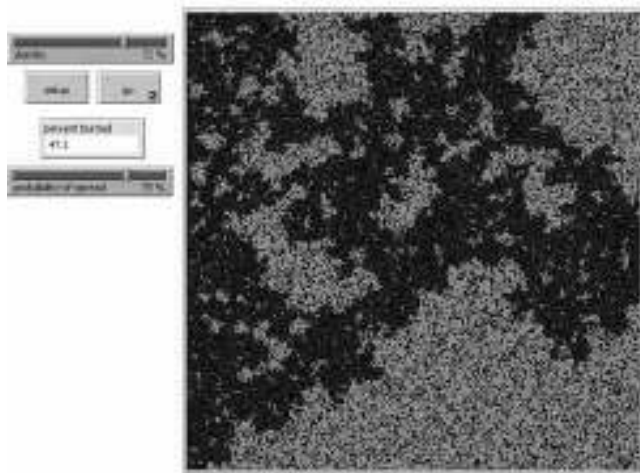
```
to go
  ;; stop the model when done
  if all? patches [ pcolor != red ] [ stop ]

  ;; ask the burning trees to set fire to any neighboring non-burning trees
  ask patches with [ pcolor = red ] [ ;; ask the burning trees
    ask neighbors4 with [ pcolor = green ] [ ;; ask their non-burning neighbor trees
      ;; only burn if a random draw is greater than the probability of spread
      if random 100 < probability-of-spread [ set pcolor red ] ;; to catch on fire
    ]
    set pcolor red - 3.5 ;; once the tree is burned, darken its color
  ]
  tick ;; advance the clock by one "tick"
end
```

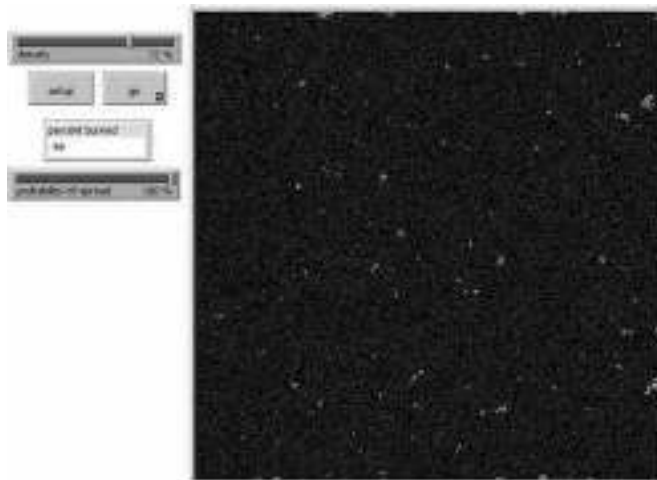
Results

- Try setting both the DENSITY and the PROBABILITY-OF-SPREAD parameters to a variety of values.
- We can reproduce the results of the original model by setting the PROBABILITY-OF-SPREAD slider to 100 percent.

New Model



Replication of Old Model



Summary

- We have examined how minor changes in a model can help make complex changes in the outcome
- Also, we have seen how to make models more realistic

Second Extension: Adding Wind

Overview

- Make changes to add new parameter of Wind

Commands

- In NetLogo, commands and reporters tell agents what to do.
- A command is an action for an agent to carry out, resulting in some effect.
- A reporter is a set of instructions for computing a value, which the agent then "reports" to whoever asked it.
- Typically, a command name begins with a verb, such as "create," "die," "jump," "inspect," or "clear."
- Most reporter names are nouns or noun phrases.
- For example, "forward" is a command, it tells a turtle agent to carry out a movement
- But "heading" is a reporter, it tells the agent to report the angle it is facing.

Built-in

- Many commands and reporters are built into NetLogo,
- We can create new ones in the Code tab.
- Commands and reporters built into NetLogo are called primitives.
- The NetLogo Dictionary has a complete list of built-in commands and reporters.

User Generated

- We can create your own commands and reporters by defining them in procedures.
- Each procedure has a name, preceded by the keyword to or to-report, depending on whether it is a command procedure or a reporter procedure.
- The keyword end marks the end of the procedure.
- It can be useful to notice that reporters are colored purple and commands are colored blue in NetLogo's automatic syntax highlighting.
- Sometimes an input to a primitive is a command block (zero or more commands inside square brackets, e.g., [forward 10 set color red])

- Or a reporter block (a single reporter expression inside square brackets, e.g., the Boolean reporter [color = red]).

Second Extension

- Sometimes we can take an ABM for which we used a probability to model a suite of mechanisms and refine it so that some of those mechanisms are modeled in a more physical way.
- E.g. Our first extension hides a host of possible mechanisms in the PROBABILITY-OF-SPREAD.
- Wind is a good example of a process that we can model more specifically.
- We can think of the effect of wind on a fire as increasing the chance of fire spread in the direction it is blowing, decreasing the chance of fire spread in the direction it is not blowing,
- Also, wind will have little effect on fire spread that occurs perpendicular to the direction of its movement
- Of course, this too is an oversimplification — there are often local effects of wind, such as turbulence
- However, all models are simplifications anyways

Second Extension Code

```

to go
  ;; stop the model when done
  if all? patches [ pcolor != red ] [ stop ]
  ;; each burning tree (red patch) checks its 4 neighbors.
  ;; If any are unburned trees (green patches), change their probability
  ;; of igniting based on the wind direction
  ask patches with [ pcolor = red ] [
    ;; ask the unburned trees neighboring the burning tree
    ask neighbors4 with [ pcolor = green ] [
      let probability probability-of-spread
      ;; compute the direction from you (the green tree) to the burning tree
      ;; (NOTE: "myself" is the burning tree (the red patch) that asked you
      ;; to execute commands)
      let direction towards myself
      ;; the burning tree is north of you
      ;; so the south wind impedes the fire spreading to you
      ;; so reduce the probability of spread
    ]
  ]

```

```

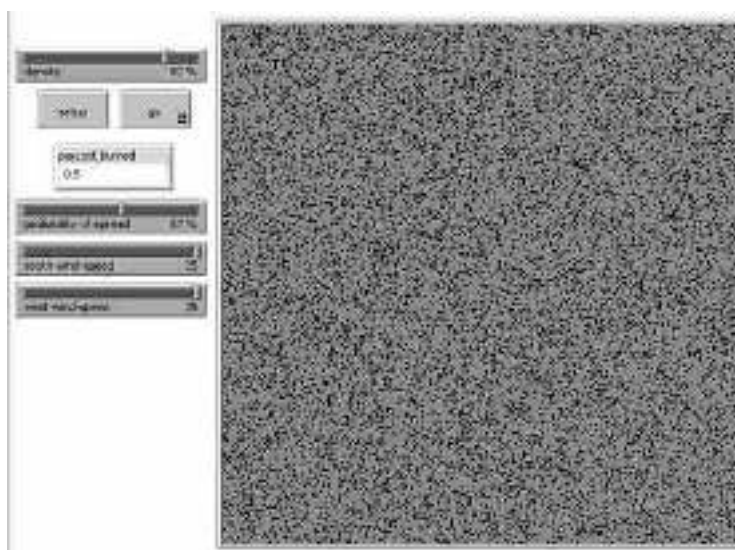
if direction = 90 [
    set probability probability - west-wind-speed
]
;; the burning tree is south of you
;; so the south wind aids the fire spreading to you
;; so increase the probability of spread
if direction = 180 [
    set probability probability + south-wind-speed
]
;; the burning tree is west of you
;; so the west wind aids the fire spreading to you
;; so increase the probability of spread
if direction = 270 [
    set probability probability + west-wind-speed
]
if random 100 < probability [
    set pcolor red ;; to catch on fire
]
]
set pcolor red - 3.5 ;; once the tree is burned, darken its color

```

Understanding the Code

- Essentially, what it does is modify the probability of spread, increasing it in the direction of the wind and decreasing it in the opposite direction

New UI



Understanding the Code

The 'Slider' dialog box for 'south-wind-speed' has the following fields: 'Global variable' is 'south-wind-speed'; 'Minimum' is '-25'; 'Increment' is '1'; 'Maximum' is '25'; 'Value' is '25'; 'Units (optional)' is empty; and 'vertical?' is unchecked. The 'OK', 'Apply', and 'Cancel' buttons are at the bottom.

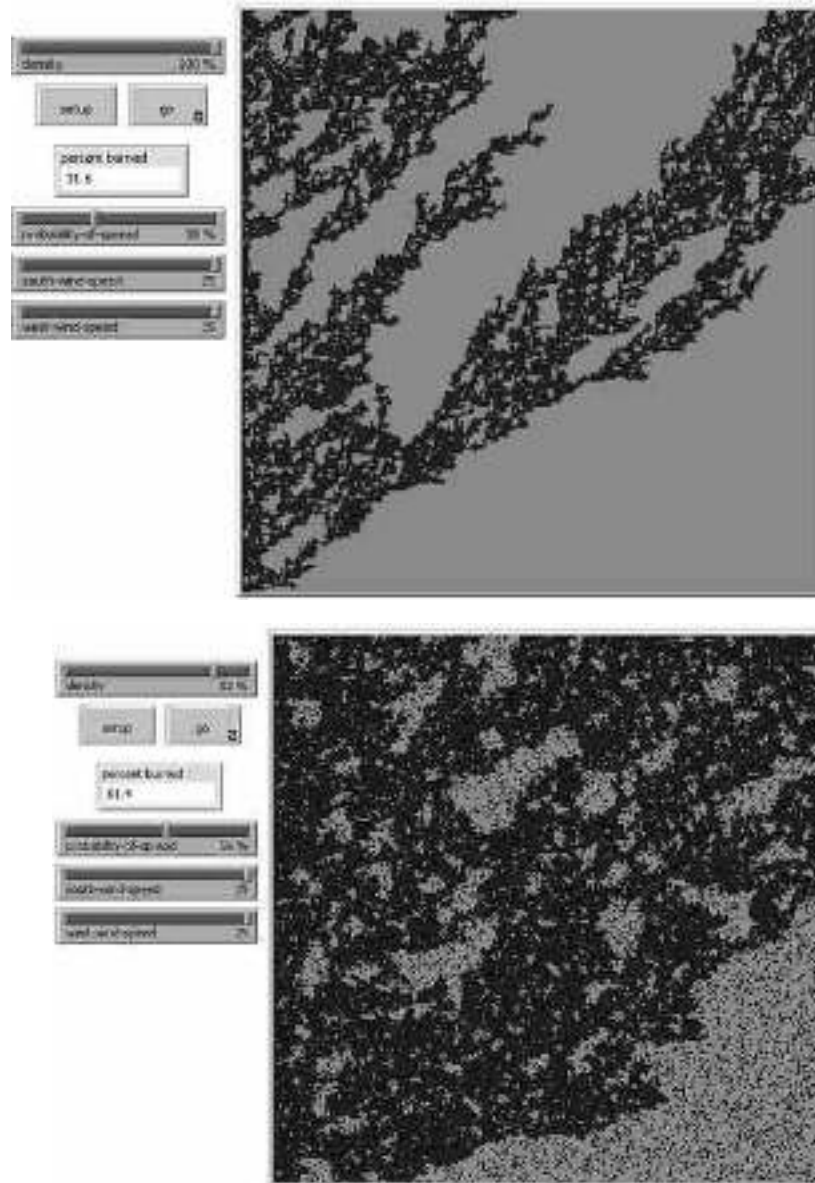
The 'Slider' dialog box for 'west-wind-speed' has the following fields: 'Global variable' is 'west-wind-speed'; 'Minimum' is '-25'; 'Increment' is '1'; 'Maximum' is '25'; 'Value' is '25'; 'Units (optional)' is empty; and 'vertical?' is unchecked. The 'OK', 'Apply', and 'Cancel' buttons are at the bottom.

Second Extension

- Also, wind will have little effect on fire spread that occurs perpendicular to the direction of its movement
- Of course, this too is an oversimplification — there are often local effects of wind, such as turbulence
- However, all models are simplifications anyways

Interesting Results

- This modification can lead to quite interesting patterns of spread.
- For example, set the density at 100 percent, and make the wind blow strong from the south and west.
- At the same time, set the PROBABILITY-OF-SPREAD fairly low, say around 38 percent.
- This creates a triangular spread, and, all else being equal, the fire should spread to the northeast.



Summary

- We have examined how we can make intelligent changes in a model
- This helps us predict and replicate complex phenomena in the real world

Third Extension: *Allow Long-Distance Transmission*

Overview

- Make changes to add support for long jumps
- Understand how Exploratory Agent-based Modeling (EABM) works

Exploratory Agent-based Modeling

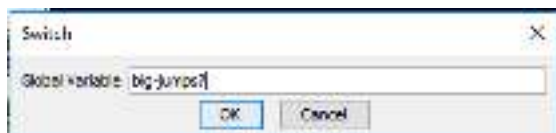
- Cognitive Agent-based Computing Framework by Niazi and Hussain presents detailed instructions on how to build Agent-based Models
- One of the key techniques in developing standard models is to build models and gradually improve them over time

Third Extension

- . We add a Boolean (TRUE or FALSE) switch labeled BIG-JUMPS?
- Switches in NetLogo provide a way to have direct control over variables that can only be true or false.
- One way to model fire jumping due to wind would be to ignite a new fire at some distance in the direction of the wind.



Third Extension Code



Understanding the Code

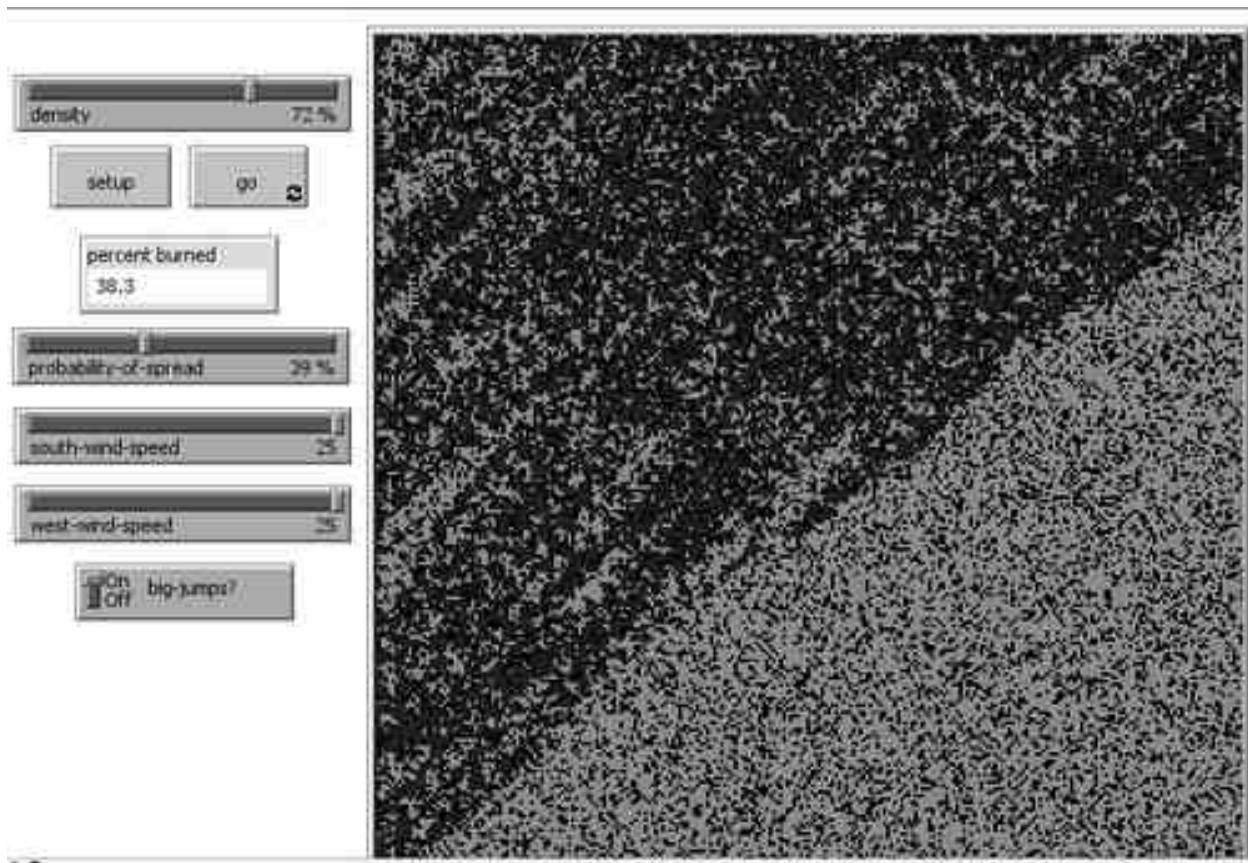
- Here we see the major changes in the code

```

if random 100 < probability [
  set pcolor red ;; to catch on fire
  ;; if big jumps is on, then sparks can fly farther
  if big-jumps? [
    let target patch-at (west-wind-speed / 5) (south-wind-speed / 5)
    if target != nobody and [ pcolor ] of target = green [
      ask target [
        set pcolor red ;; to ignite the target patch
      ]
    ]
  ]
]

```

Effects of jumps



Fire Model: Summary

- We have seen how EABM can be used to develop and extend a model
- There are three different changes
- Each adds something interesting

- The final change to the model points out that the measure we were observing (whether or not the fire made it to the right edge of the world) might not even be the measure we want to observe.
- As we change the fire spread mechanisms, other
- measures such as “ percent burned ”
- These may become more important. may or may not have tipping points.
- This illustrates an important feature of tipping points.
- They are defined by an input and an output measure.
- A model does not have a tipping point in and of itself — the tipping point is relative to the choice of inputs and output
- measures.

Summary

- We have learnt about another extension of the fire model
- We have seen how EABM can be used to extend models
- We have developed a basic understanding of modeling complex phenomena in the real world

Module – 93

The Diffusion-Limited Aggregation (DLA) Model

Overview

- Understand the **The Diffusion-Limited Aggregation (DLA) Model**

DLA Model

- As demonstrated with the Fire Simple model, it is possible to think about inanimate objects (such as trees) as agents.
- We can reconceptualize atoms and molecules as agents, and then developing procedural rules to describe how these agents interact.
- In many physical processes from the creation of clouds, to snowflakes, to soot, smoke, and dust, particles aggregate in interesting ways



- Diffusion-limited aggregation (DLA) is an idealization of this process,
- It was first examined as a computational model in the early eighties (Witten & Sander, 1981 , 1983).
- DLA models were able to generate patterns that resemble many found in nature

- These include crystals, coral, fungi, lightning, and growth of human lungs, as well as social patterns such as growth of cities (Garcia-Ruiz et al., 1993; Bentley & Humphreys, 1962; Batty & Longley, 1994).
- This model starts with a large number of red particle agents and one green patch in the center.
- The green patch at the center of the screen is stationary
- When we press GO, all of the red particles move randomly by "wiggling" left and right (resulting in a small random turn)
- Each also moves forward one step.
- After a particle moves, if any of its neighbors (i.e., nearby patches) are green, the particle dies, turning the patch it is on green
- The particle can do so because all turtles have direct access to the variables of the patch they are on.
- If you let the model run long enough, you will get an interesting fractal-like pattern forming from the green patches

NetLogo Model Simulation

- [Simulation](#)

NetLogo Model

```

to setup
  clear-all
  ;; start with one green "seed" patch at the center of the world
  ask patch 0 0 [
    set pcolor green
  ]
  create-turtles num-particles [
    set color red
    set size 1.5 ;; easier to see
    setxy random-xcor random-ycor
  ]
  reset-ticks
end

to go
  ask turtles [
    ;; turn a random amount right and left
    right random wiggle-angle
    left random wiggle-angle
    forward 1
    ;; if you are touching a green patch
    if any? neighbors with [ pcolor = green ] [
      set pcolor green ;; turn your own patch green
      die ;; and then die
    ]
  ]
  tick
end

```

Summary

- We have learnt about the NetLogo DLA Model
- We have also developed further understanding of modeling complex phenomena in the real world

Module – 94

First Extension: Probabilistic Sticking

Overview

- We apply Exploratory Agent-based Modeling to extend the DLA model

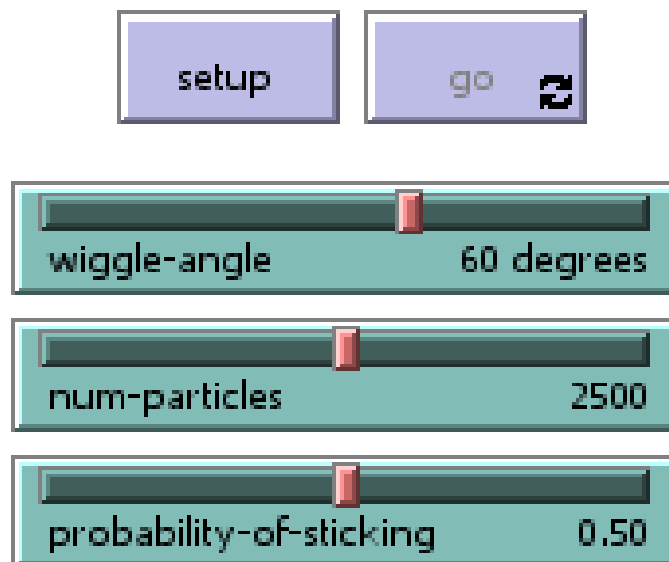
DLA Model

- The DLA Simple model has very simple rules.

- The code consists of only two procedures
- Both of them are very small, yet these two procedures can generate many interesting results.
- After you run the model for a while, you will realize that it always produces thin and wispy types of structures.
- Often the " stems " and " trunks " of the structures are only a single patch wide.
- This is because as soon as a particle touches anything green it will stop moving.
- It is much more likely that it will touch something toward the edge of the structure rather than near the interior of the structure.
- However, we can change this.
- One way to think of this system is that if a particle comes into contact with a stationary object, then it becomes stationary itself with 100 percent probability.

DLA Extension

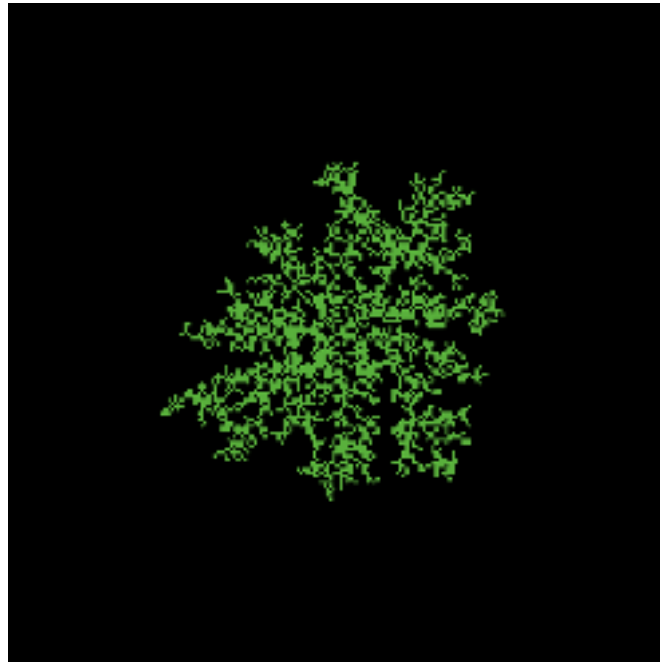
- What if we decrease this probability?
- That is exactly what we will do in this extension.
- We will allow the user to control the probability of a particle becoming stationary.



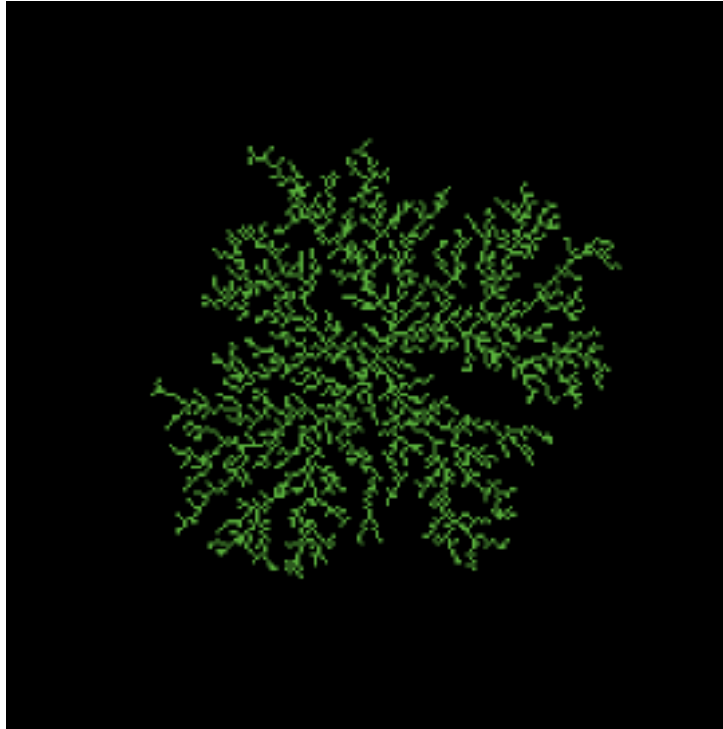

```

to go
  ask turtles [
    ;; turn a random amount right and left
    right random wiggle-angle
    left random wiggle-angle
    forward 1
    if any? neighbors with [ pcolor = green ]
      and pcolor = black
      and random-float 1.0 < probability-of-sticking [
        set pcolor green
        die
      ]
  ]
  tick
end

```



DLA Extension: Original Replication



Summary

- We have seen how EABM can be used to extend the original DLA model
- Credits: Uri Wilensky book

Module – 95

Second Extension: Neighbor Influence

Overview

- We apply Exploratory Agent-based Modeling to extend the DLA model further by adding the influence of neighbors

DLA Extension

- One commonly explored extension to the DLA model is to explore how the probability of sticking is related to the number of neighbors that are already stationary (the green patches) (Witten & Sander, 1983).
- If a particle is moving, it is more likely that it will stick somewhere if two or three of its neighbors are at rest than if just one of its neighbors is at rest.
- This can be done by first adding a switch NEIGHBOR-INFLUENCE?

```

to go
  ask turtles [
    right random wiggle-angle
    left random wiggle-angle
    forward 1
    ;; if neighbor-influence is TRUE then make the probability proportionate
    ;; to the number of green neighbors, otherwise use the slider as before
    let local-prob probability-of-sticking
    if neighbor-influence? [
      ;; increase the probability of sticking the more green neighbors there are
      set local-prob probability-of-sticking * (count neighbors with [ pcolor = green ] / 5)
    ]
    if any? neighbors with [ pcolor = green ]
      and pcolor = black
      and random-float 1.0 < local-prob [
        set pcolor green
        die
      ]
  ]
  tick
end

```



Simulation Results

- When we run the model with this change in the code, we notice
- (1) that it takes much longer to form a structure because there is a much lower probability of a mobile particle stopping,
- (2) the structures that emerge are very thick and almost bloblike.
- In addition there are many fewer one patch-wide branches, because the probability of stopping when in contact with only one other stationary patch is very small.

Summary

- We have seen how EABM can be used to further extend the original DLA model
- Credits: Uri Wilensky book

Module – 96

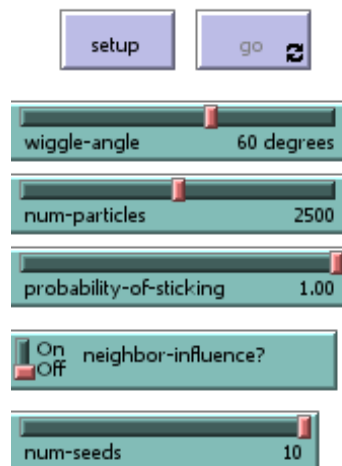
Third Extension: Different Aggregates

Overview

- We now apply Exploratory Agent-based Modeling to extend the DLA model further by adding the capability of different aggregates

DLA Extension 3

- The previous two extensions have concentrated on deciding how a particle should stop when it is moving
- There is another way that we can control how a particle stops.
- We can simply give it more places to stop at the beginning.

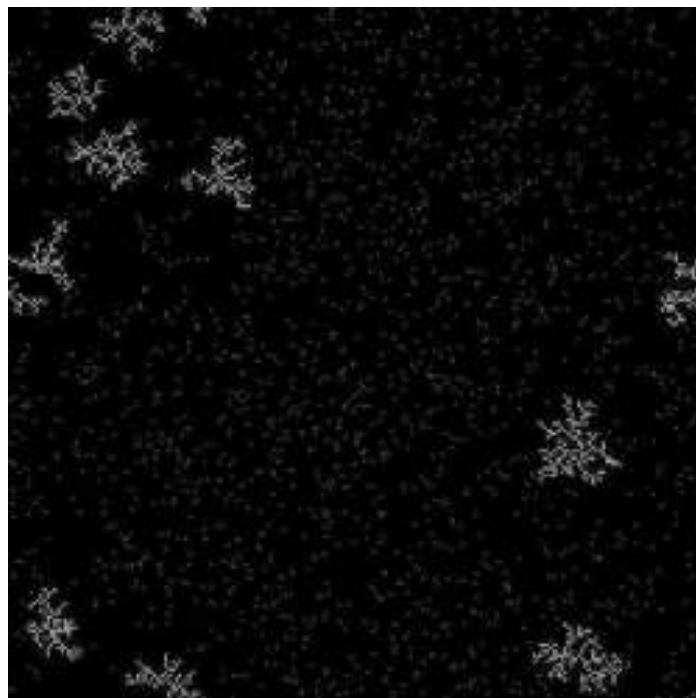
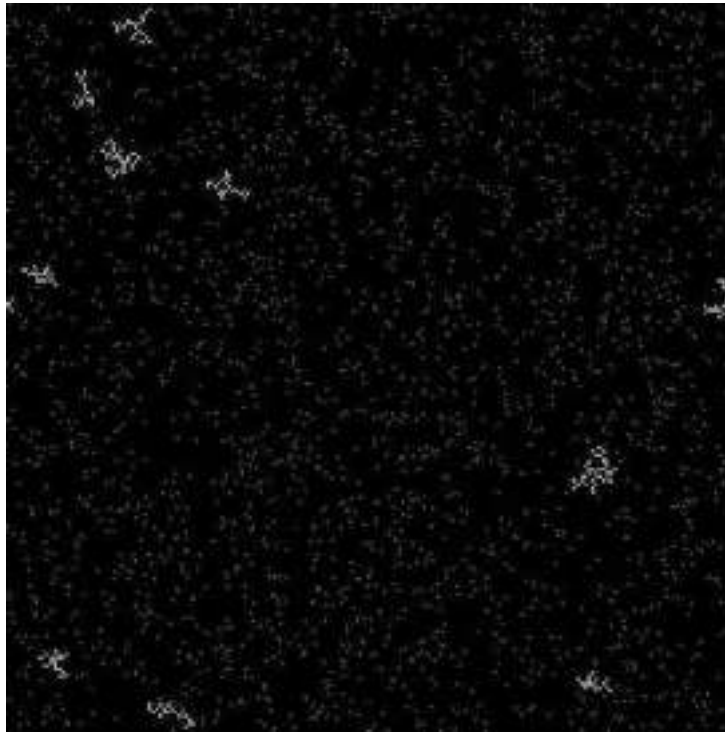


```
to setup
  clear-all
  ;; start with NUM-SEEDS green patches as "seeds"
  ask n-of num-seeds patches [
    set pcolor green
  ]
  create-turtles num-particles [
    set color red
    set size 1.5 ;; easier to see
    setxy random-xcor random-ycor
  ]
  reset-ticks
end
```

Simulation Results

- When we run the model with this change in the code, we notice multiple seeds

- However, it does not replicate the original results
- The seed may not be at the center





Summary

- We have seen how EABM can be used to further extend the original DLA model by adding the option of multiple seeds
- Credits: Uri Wilensky book

Module – 97

Segregation Model

Overview

- Here, we understand the basics of the so-called “Segregation” model

Fundamental Approaches to ABM

- There are two fundamental approaches to starting to build an agent-based model.
- The first approach, more common in science contexts, is to start with a known phenomenon to be modeled.
- This approach is called phenomena-based modeling.
- Typically phenomena-based modeling is linked with an aggregate pattern, termed a reference
- Another way to begin is to start with some simple rules and play them out to see what patterns develop.

- This approach is sometimes referred to as exploratory modeling
- Both ways can be classified under Exploratory agent-based modeling (EABM).
- EABM with Standard Reference
- EABM with Pure Exploration
- Often times a combination of these is used

Schelling Model

- Thomas Schelling took a more exploratory modeling approach with his Neighborhood Tipping model (1971).
- Schelling wondered what would happen if you assume that everyone in the world wanted to live in a place where at least a reasonable fraction of their neighbors were like themselves,
- i.e., they have an aversion to being an
- extreme minority (which he called weakly prejudiced; see Anas, 2002).
- He explored this model using a checkerboard as a grid,
- Pennies to represent one race and dimes to represent another.
- He counted by hand the number of pennies surrounding each dime and divide by the number of neighbors surrounding that dime.
- If this value exceeded a certain threshold value, he would move the dime to a random empty location on the board.
- He repeated this process hundreds of times and observed the outcome.

Summary

- We have seen how EABM can be used to develop Agent-based Models
- We have looked at the basis of Schelling's work
- Credits: Uri Wilensky book

Module – 98

Description of the Segregation Model-I

Overview

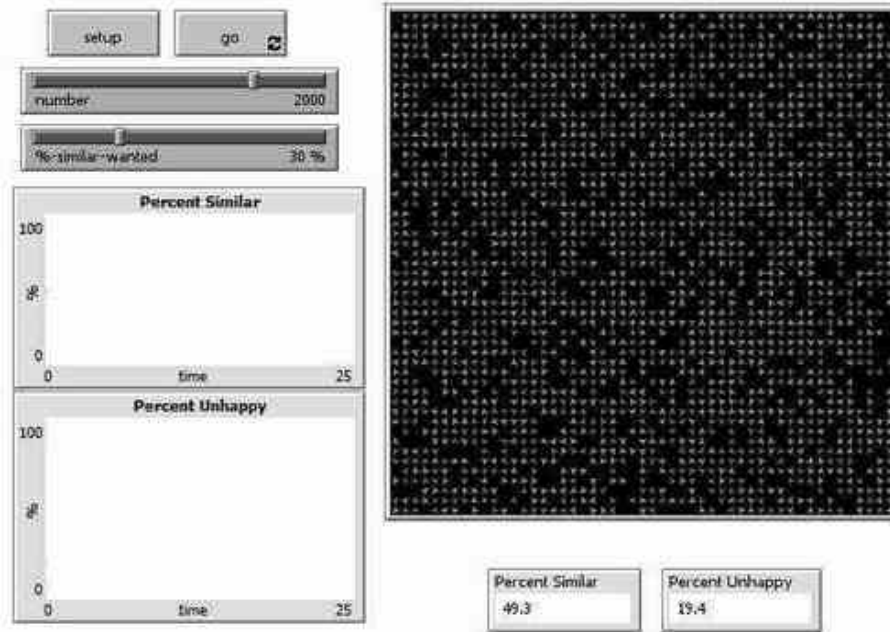
- Here, we understand the details of the so-called "Segregation" model

Schelling Model

- What surprised Schelling about this was that even at low levels of the threshold, he would still see groups of pennies and groups of dimes together in dense clusters.
- This global segregation occurred despite the relative tolerance of each individual.
- There was no single individual in the Schelling model who wanted a segregated neighborhood, but the group as a whole moved toward segregation.
- Schelling described this as "macrobehavior" derived from "micromotives" (1978).
- At the time when Schelling introduced this model, it was very controversial for several reasons.
- First, it was widely believed at the time that housing segregation was caused by individuals being prejudiced.
- Schelling's work seemed to be "excusing" people from prejudice, saying that prejudice itself is not the cause of segregation.
- The cause of segregation is an emergent effect of the aggregation of people's weak prejudice.
- But Schelling's point was not to excuse people for their prejudice, his model demonstrates that prejudice is not a "leverage point" of the housing system.
- Another controversial aspect of Schelling's model is that it modeled the people as behaving with very simple rules.
- Critics argued that "people are not ants";
- A map of Chicago with each dot representing twenty-five people.
- By representing the data at this finer level than traditional maps that aggregate demographics are mapped at a high level, Rankin illustrates nuances of segregation much like ABM enables us to explore nuances of individual-level behavior.



- Schelling was awarded the Nobel Prize in economics in 2005.
- To this day, it remains controversial how much of human behavior can be modeled with simple rules.
- Schelling was awarded the Nobel Prize in economics in 2005.
- To this day, it remains controversial how much of human behavior can be modeled with simple rules.



Summary

- We have seen the details of Schelling Model
- Credits: Uri Wilensky book

Module – 99

Description of the Segregation Model-II

Overview

- Here, we understand the details of the so-called "Segregation" model

Schelling Model

```
globals [
  percent-similar ;; on the average, what percent of a turtle's neighbors
                  ;; are the same color as that turtle?
  percent-unhappy ;; what percent of the turtles are unhappy?
]
```

```

turtles-own [
  happy?           ;; for each turtle, indicates whether at least %similar-wanted percent of
                   ;; that turtle's neighbors are the same color as the turtle
  similar-nearby   ;; how many neighboring patches have a turtle with my color?
  total-nearby     ;; sum of previous two variables
]

```

```

to setup
  clear-all
  ;; create a turtle on NUMBER randomly selected patches.
  ;; note that slider's maximum value is 2500 which is a
  ;; little less than the total number of patches
  ask n-of number patches [
    sprout 1
  ]
  ask turtles [
    ;; make approximately half the turtles red and the
    ;; other half green
    set color one-of [ red green ]
  ]
  update-turtles
  update-globals
  reset-ticks
end

```

Summary

- We have seen the details of Schelling Model
- Credits: Uri Wilensky book

Module – 100

Simulation of the Segregation Model

Overview

- Here, we understand the simulation of the so-called “Segregation” model

Schelling Model

```
to go
  if all? turtles [ happy? ] [ stop ]
  move-unhappy-turtles
  update-turtles
  update-globals
  tick
end

to move-unhappy-turtles
  ask turtles with [ not happy? ]
    [ find-new-spot ]
end

to find-new-spot
  rt random-float 360
  fd random-float 10
  if any? other turtles-here
    [ find-new-spot ] ;; keep going until we find an unoccupied patch
  setxy pxcor pycor ;; move to center of patch
end
```

```

to setup
  clear-all
  ;; create a turtle on NUMBER randomly selected patches.
  ;; note that slider's maximum value is 2500 which is a
  ;; little less than the total number of patches
  ask n-of number patches [
    sprout 1
  ]
  ask turtles [
    ;; make approximately half the turtles red and the
    ;; other half green
    set color one-of [ red green ]
  ]
  update-turtles
  update-globals
  reset-ticks
end

```

```

to update-turtles
  ask turtles [
    ;; in next two lines, we use "neighbors" to test the eight patches
    ;; surrounding the current patch

    ;; count the number of my neighbors that are the same color as me
    set similar-nearby count (turtles-on neighbors)
      with [color = [color] of myself]

    ;; count the total number of neighbors
    set total-nearby count (turtles-on neighbors)

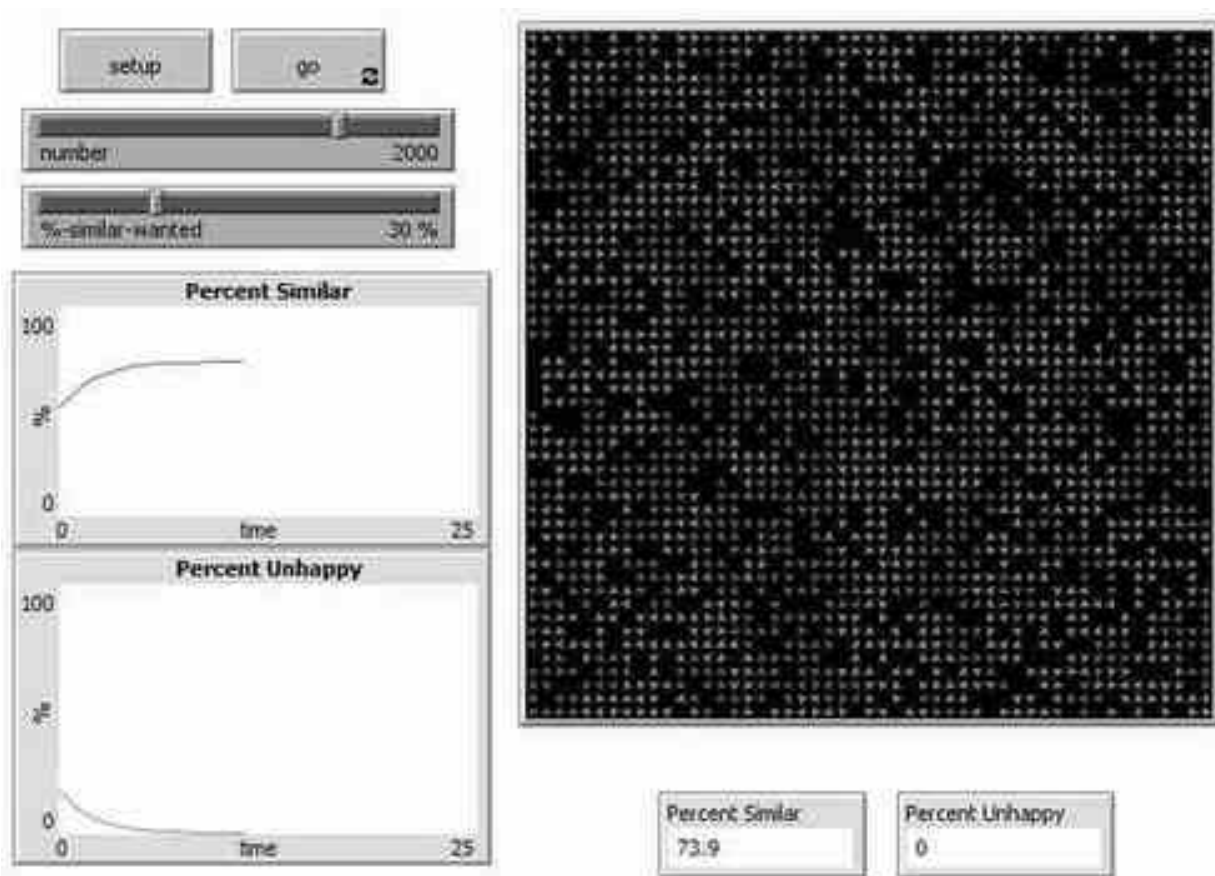
    ;; I'm happy if there are at least the minimal number of same-colored neighbors
    set happy? similar-nearby >= ( %-similar-wanted * total-nearby / 100 )
  ]
end

```

```

to update-globals
  let similar-neighbors sum [similar-nearby] of turtles
  let total-neighbors sum [total-nearby] of turtles
  set percent-similar (similar-neighbors / total-neighbors) * 100
  set percent-unhappy (count turtles with [not happy?]) / (count turtles) * 100
end

```



Summary

- We have seen the details of Schelling Model
- Credits: Uri Wilensky book

First Extension: Adding Multiple Ethnicities

Overview

- Here, we understand the extension to the “Segregation” model

Schelling Model

- One simple extension to this model is to add a third, fourth, and even fifth ethnicity.
- This can be done by modifying the SETUP code.

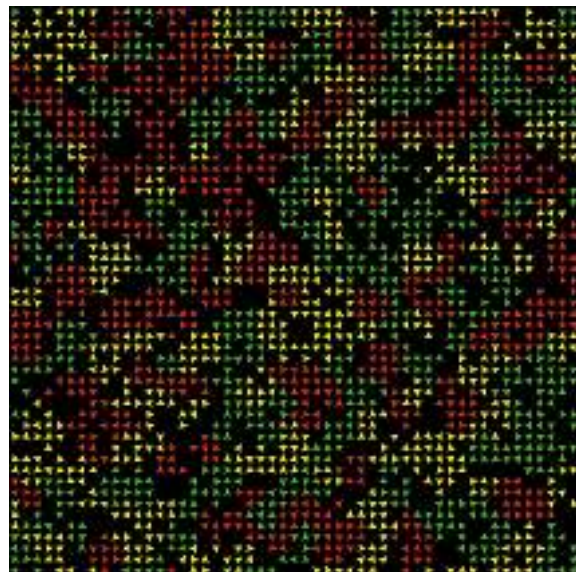
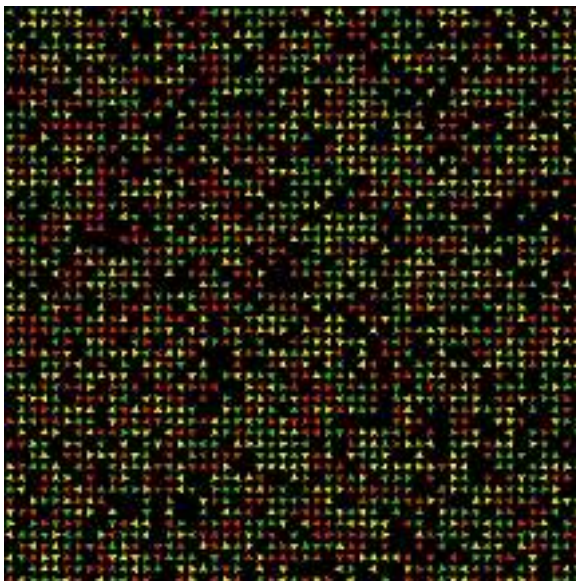
```
globals [
  percent-similar    ;; on average, what percent of a turtle's neighbors
                      ;; are the same color as that turtle?
  percent-unhappy    ;; the percent of the turtles that are unhappy
  colors             ;; a list of colors we use to color the turtles
]
```

```
to setup
  clear-all

  set colors [ red green yellow blue orange ]

  ;; create a turtle on EVERY randomly selected patch.
  ;; note that slider's maximum value is 2500 which is a little less than the total number
  ask n-of number-patches [
    sprout 1
  ]
  ;; assign a color to each turtle from our list of colors
  ask turtles [
    set color (item (random number-of-ethnicities) colors)
  ]

  update-turtles
  update-globals
  reset-ticks
end
```



Summary

- We have seen the details of Schelling Model
- Credits: Uri Wilensky book

Module – 102

Second Extension: Allowing Diverse Thresholds

Overview

- Here, we understand a further extension to the “Segregation” model

Schelling Model

- In the original version of Schelling ’ s model, every agent in the world had exactly the same similarity threshold.
- This meant that every agent had the same level of tolerance for having other ethnicities in its neighborhood.
- Although relaxing this assumption might have been difficult to enact with a checkerboard, it is straightforward to do using an ABM language.

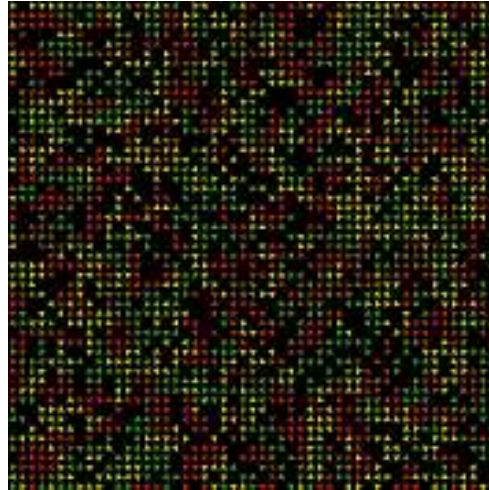
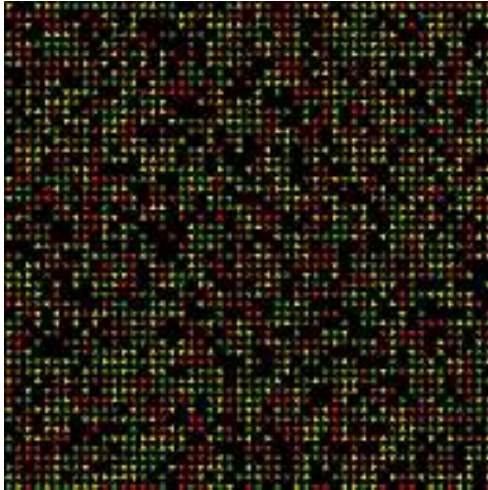
```

] turtles-own [
  happy?           ;; for each turtle, indicates whether at least
  ;; %-similar-wanted percent of
                    ;; that turtle's neighbors are the same color as the turtle
  similar-nearby    ;; how many neighboring patches have a turtle with my color?
  total-nearby      ;; sum of previous two variables
  my-%-similar-wanted ;; the threshold for this particular turtle
]

;; assign a color to each turtle from the list of our colors
;; and assign an individual level of %-similar-wanted
ask turtles [
  set color (item (random number-of-ethnicities) colors)
  set my-%-similar-wanted random %-similar-wanted
]

;; I'm happy if there are at least the minimal number of same-colored neighbors
set happy? similar-nearby >= ( my-%-similar-wanted * total-nearby / 100 )

```

Summary

- We have seen another extension of Schelling Model
- Credits: Uri Wilensky book

Module – 103

Third Extension: Adding Diversity-Seeking Individuals

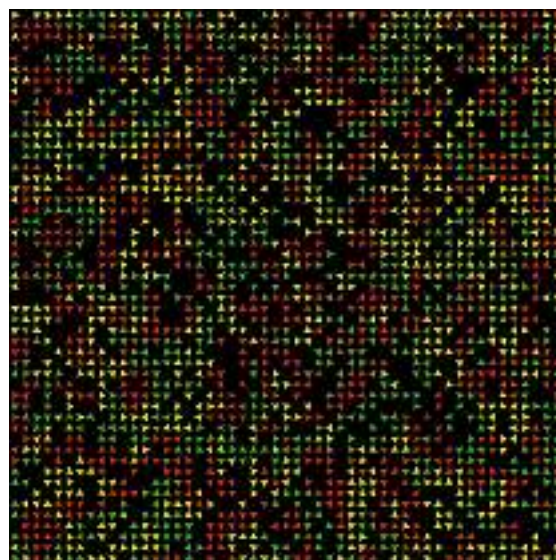
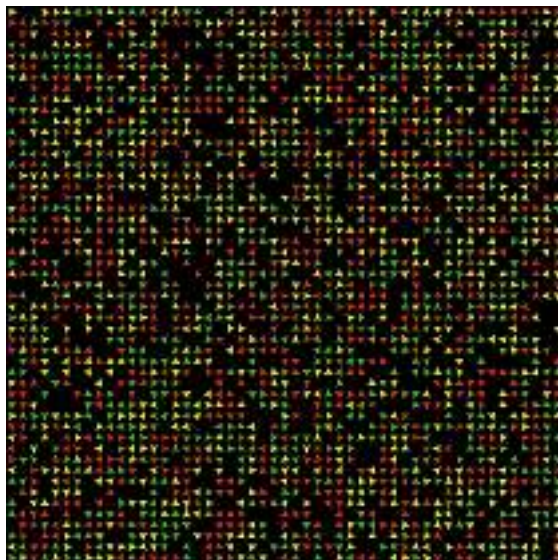
Overview

- Here, we understand an extension to allow for diversity-seeking individuals in the “Segregation” model
- Another simplification that the original segregation model makes about the world is that individuals are concerned about too much diversity
- However no agents actively seek diversity in their neighborhoods
- Although relaxing this assumption might have been difficult to enact with a checkerboard, it is straightforward to do using an ABM language.

```
] turtles-own [  
  happy?           ;; for each turtle, indicates whether at least  
  ;|%-similar-wanted percent of  
                    ;; that turtle's neighbors are the same color as the turtle  
  similar-nearby    ;; how many neighboring patches have a turtle with my color?  
  total-nearby      ;; sum of previous two variables  
  my-%-similar-wanted ;; the threshold for this particular turtle  
]
```

```
;; assign a color to each turtle from the list of our colors
;; and assign an individual level of %-similar-wanted
ask turtles [
  set color (item (random number-of-ethnicities) colors)
  set my-%-similar-wanted random %-similar-wanted
]

;; I'm happy if there are at least the minimal number of same-colored neighbors
set happy? similar-nearby >= ( my-%-similar-wanted * total-nearby / 100 )
```



Summary

- We have seen another extension of Schelling Model
- Credits: Uri Wilensky book

Module – 104

Advanced Urban Modeling Applications

Overview

- Here, we understand a Advanced Urban Modeling Applications

Modeling Of Urban Landscapes.

- Agent-based modeling has frequently been used for the modeling of urban landscapes.

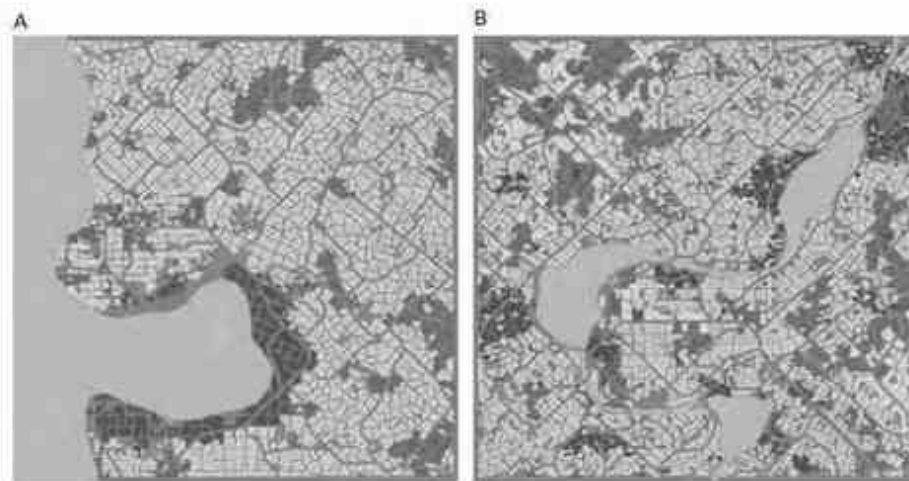
“ Residential Preference ”

- The Segregation model described earlier can be seen as one particular example of what are generally called “Residential Preference” models, which are a major component of urban modeling systems

CITIES project

- These models can be combined with commercial, industrial, and governmental models of urban policy to create an integrated model of a city.
- One such example is the CITIES
- CITIES project carried out by the Center for Connected Learning and Computer-Based Modeling at Northwestern University
- The goal of the CITIES project was to procedurally develop realistic cities that could be used as tools in education and exploration

CITIES model in Net Logo (Lechner et al., 2006).



Goal Of Urban Modeling

- Another goal of urban modeling is to explore the ecological impact of cities and their footprint on the environment.

SLUCE Project

- The SLUCE project at the University of Michigan built several models of residential and tract development to explore land-use policies and their effect on the surrounding environs in southeastern Michigan (Brown et al., 2005 ; Rand et al., 2003).

Summary

- We have seen how Agent Based Modeling can be used for Advanced Urban Modeling Applications
- Credits: Uri Wilensky book

Module – 105

The El Farol Model &

Description of the El Farol Model

Overview

- Here, we understand *the El Farol model*, and we will add additional reporters to this model.
- In this way we can gain *new information from the model* that was not previously available.

The El Farol Model

- The El Farol model uses “ lists ” and mathematical regression, but the modifications we will make to this model do not require knowledge of how these particular methods work

Description of the El Farol Model

- W. Brian Arthur was an Irish economist who, was a pioneer of using complexity methods in economics
- Arthur posed the El Farol Bar problem as an exploration of bounded rationality and inductive reasoning

" Optimal ” Equilibrium

- Traditional neoclassical economics presupposes that humans are completely rational:
 - that is, they have *access to perfect information* and *maximize their utility in every situation*
- When *each agent behaves* this way the aggregate can achieve an *optimal equilibrium*

El Farol Bar

- On Canyon Road in Santa Fe, New Mexico, there is a bar called the El Farol.
- This bar was popular with researchers from the Santa Fe Institute, including Arthur
- One night each week, the El Farol had live Irish music, and Arthur enjoyed going on these nights, but occasionally it got crowded, and he did not enjoy it on those nights.
 - *Arthur wondered, how do people decide if they should go to the bar or not?*

Arthur’s hypothesis

- Arthur hypothesized that one way to model this situation would be to give each agent a bag of strategies.

- Each of these strategies would be some rule of thumb about what the attendance was that week, e.g.

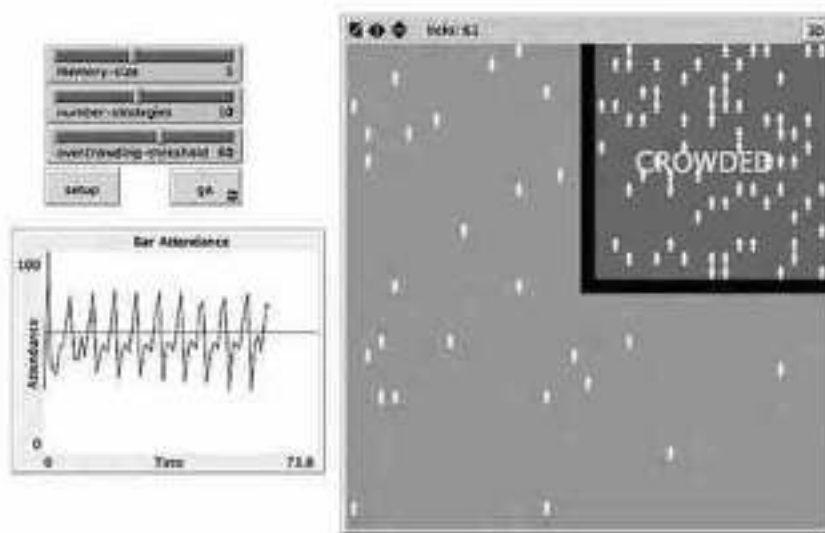
Strategies

- “*twice last week ’ s attendance,*” “*half the attendance of two weeks ago,*”
- or
- “*an average of the last three weeks attendance.*”
- The agents would use whichever strategy would have worked the best to predict this coming week ’ s attendance

NetLogo Model

- This model is available in the NetLogo models library under Sample Models > IABM Textbook > Chapter Three > El Farol
- This model has a few controls. You can adjust the MEMORY-SIZE of the agents, which controls how many weeks of attendance they remember.

The El Farol model



Summary

- In this Module we have seen The El Farol model
- This model was a good example of how you do not need to understand every part of an agent-based model in order to work with it
- Credits:
 - Uri Wilensky book

Module – 106

First Extension:

Color Agents That Are More Successful Predictors

Overview

- Here, we understand about First Extension of El Farol model and will see how to further modify it in order to understand :
 - Are some agents better at deciding when the bar will not be crowded than others?

Color Agents That Are More Successful Predictors

- The original model tells you how many agents are attending the bar, and you can get the visual reference of them moving to and from the bar,
- But there is nothing in this model to indicate which agents are better at choosing good times to attend the bar.

Add a Property to Each Agent

- In order to find this out, Are some agents better at deciding when the bar will not be crowded than others?
- We first need each agent to keep track of how often they go to the bar when it is not crowded; to do this, we add a property to each agent.

REWARD

- We will modify the turtles ' properties to add one additional item, REWARD:
- We are calling this new property REWARD, since the agents get rewarded if they attend the bar when it is not crowded.
- Now we have to initialize this property, since each agent will start with reward of 0.

Initialize The REWARD To 0

```
turtles-own [  
  strategies          ;; list of strategies  
  best-strategy ;; index of the current best strategy  
  attend?            ;; true if the agent currently plans to attend the bar  
  prediction          ;; current prediction of the bar attendance  
  reward              ;; the amount that each agent has been rewarded  
]  
  
;; Create the agents and give them random strategies  
create-turtles 100 [  
  set color white  
  move-to-empty-one-of home-patches  
  set strategies n-values number-strategies [random-strategy]  
  set best-strategy first strategies  
  set reward 0  
  update-strategies  
]
```

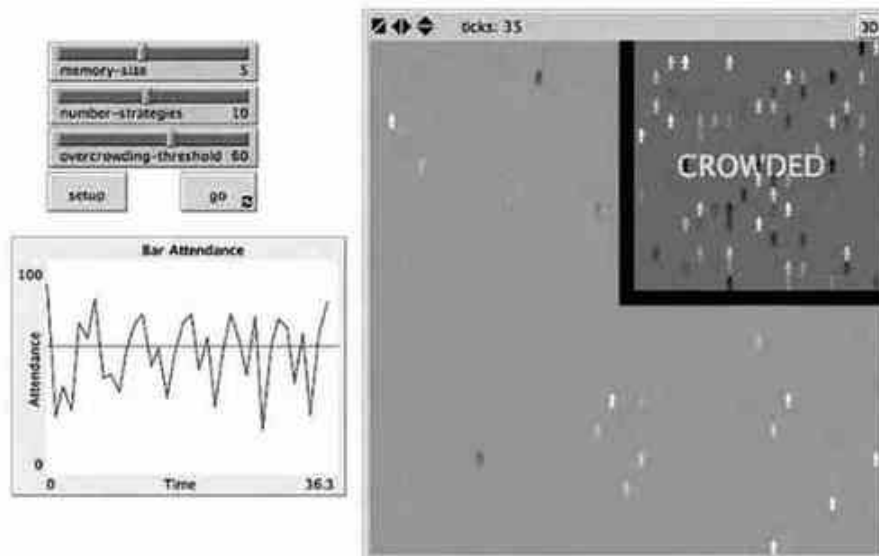
Update The Reward

- We also need to update the reward whenever the agent goes to the bar and it is not crowded.
- The word CROWDED is displayed on a patch in the bar
- We want to update the reward of turtles who attend the bar when this condition is not true, so we can make this IF statement

```
ifelse attendance > overcrowding-threshold [
  ask crowded-patch [ set plabel "CROWDED" ]
]
[ ;; if the bar is not overcrowded, reward the turtles that are attending
  ask turtles with [ attend? ] [
    set reward reward + 1
  ]
]
```

SCALE-COLOR

- NetLogo has a command called SCALE-COLOR that allow you to do this, we can set each agent 's color to a shade of a color based on their reward.
- In this case, the darkest agents will be the one with the lowest reward, and the lightest agents will be the ones with the most reward



Summary

- The first extension enables us to visually see which agents are doing better at going to the bar when it is not crowded
- Credits: Uri Wilensky book

Module – 107

Lesson 107 : Second Extension: Average, Min and Max Rewards

Second Extension: Average, Min, and Max Rewards

The first extension enables us to visually see which agents are doing better at going to the bar when it is not crowded. However, we do not have any hard numbers for the agents.

We could simply inspect each agent by right clicking on it and selecting inspect agent.

This would enable us to see the reward for each agent, but a better way might be to constantly display some information about the agents. We can do this by using monitors, which display values about the agent-based models.

To begin with we add a monitor to the Interface tab. In the first monitor, we can calculate the maximum reward collected by any agent. In the reporter area of the monitor editing window, we can put the following code snippet:

```
max [ reward ] of turtles
```

We can also give this monitor the name “ Max Reward. ” After this we can add another monitor for the minimum reward with the following code:

```
min [ reward ] of turtles
```

And give this monitor the name “ Min Reward. ” Finally, we can create a third monitor and calculate the average or mean reward using the following code:

```
mean [ reward ] of turtles
```

We can give this monitor the name “ Avg. Reward. ” The model interface window now should look like figure 3.24 . When we run this model we now get data on the average,

maximum, and minimum rewards over time. It quickly becomes apparent that though both the average and maximum reward increase over time, the max reward grows faster than the average reward meaning that some agents do better and better as time goes on. Though this extension did not take much coding, it gives you information and insight into what is going on in the model that was not available before.

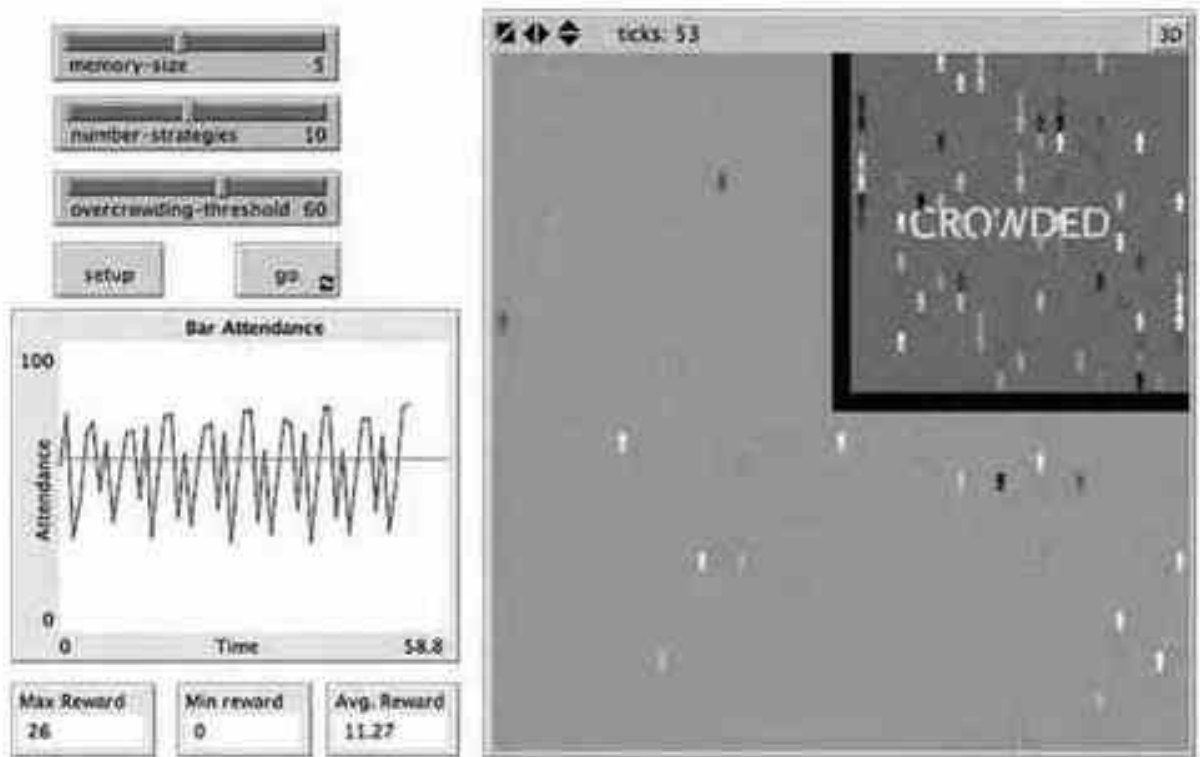


Figure 3.24
El Farol model: second extension.

Module – 108

Third Extension: Histogram Reward Values

Overview

- Here, we understand are the agents distributed uniformly across reward levels or are many agents at the minimum value and many agents at the maximum value with few agents in between?
- To find out an answer to this question, we are going to add another plot to the El Farol model.

Distribution Of Rewards

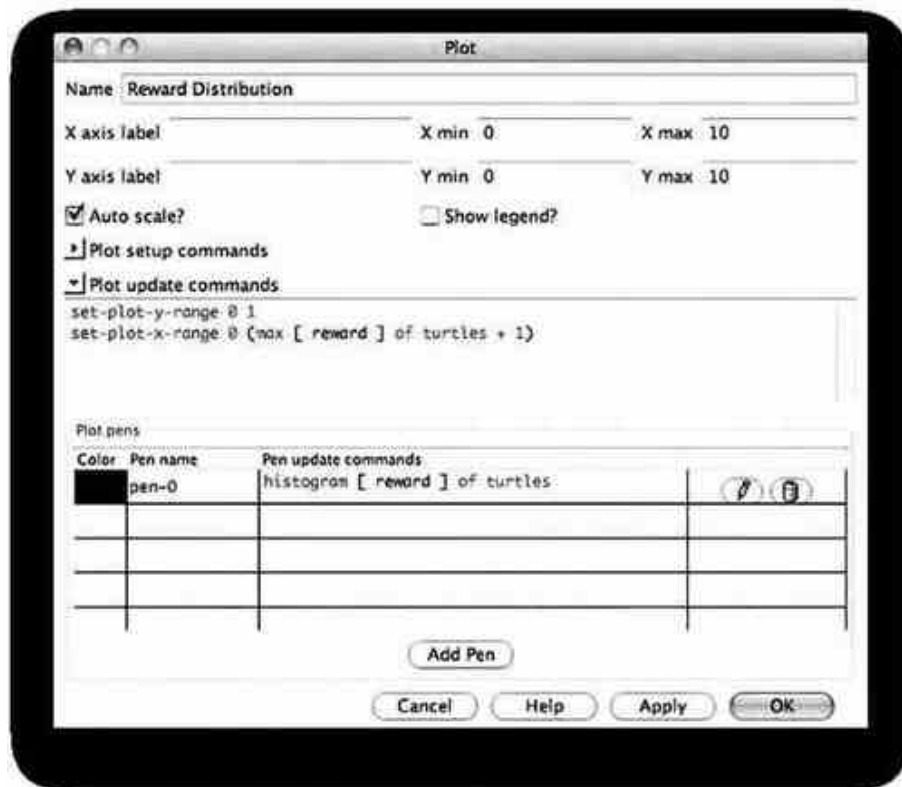
- In this model we will need to use the plot widget 's "Plot update commands " feature.
- To do this we first need to create a new plot in the Interface tab.
- In the configuration for this plot, we can give the plot the name " Reward Distribution, "

Histogram of the REWARDS

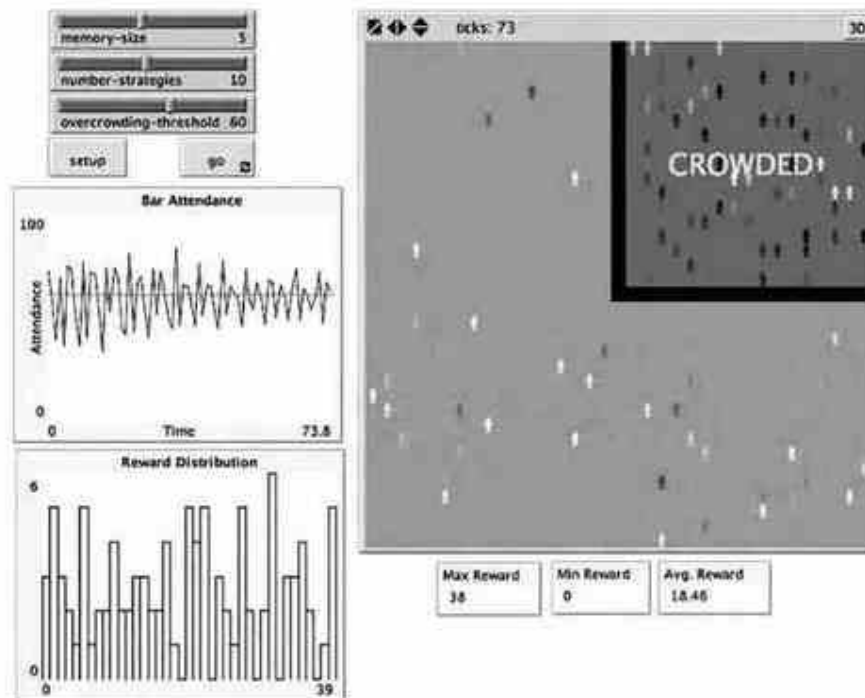
- Change the Mode of the pen to “Bar” by clicking on the pencil in the first row of the Plot pens table.
- We tell the pen to display a histogram of the distribution by putting the following code in “Pen update commands.”
 - *histogram [reward] of turtles*

Histogram

- If we ran the model in this state, we would see a histogram drawn. However, the axes of the plot would not update correctly.
- Thus, we must enter the following under “Plot setup commands.”
 - *set-plot-y-range 0 1*
 - *set-plot-x-range 0 (max [reward] of turtles + 1)*



El Farol model: third extension



Summary

- We have seen when we run the model over time, you will see it start out similar to a normal distribution, but the *distribution will quickly change with a few groups of individuals maintaining large rewards* and many more having a lower reward.
- Credits: Uri Wilensky book

Module – 109

Advanced Modeling Applications

Overview

- Here we understand the Advanced Modeling Applications

Machine learning

- Machine learning enables the creation of powerful agent-based models where the agents not only change their actions over time, but also their strategies, i.e.,
 - the agents change the way they decide to take actions

Minority Game

- The *El Farol model* has also been idealized into the *Minority Game* which has been studied by *physicists and economists* because it provides insights into *complex systems*.
- For instance, *both El Farol and the Minority Game* can be seen as a crude approximation for financial markets

Economic Model

- The El Farol model is also one example of an economic model.
- The Artificial Stock Market is an early agent-based economic model
- This model attempted to simulate the stock market

Economic Models In NetLogo

- There are several other economic models in the NetLogo models library, such as the Oil Cartel model and the Root Beer Supply Chain
- Both of these models are interesting because they make use of the HubNet facility that is also provided with NetLogo.

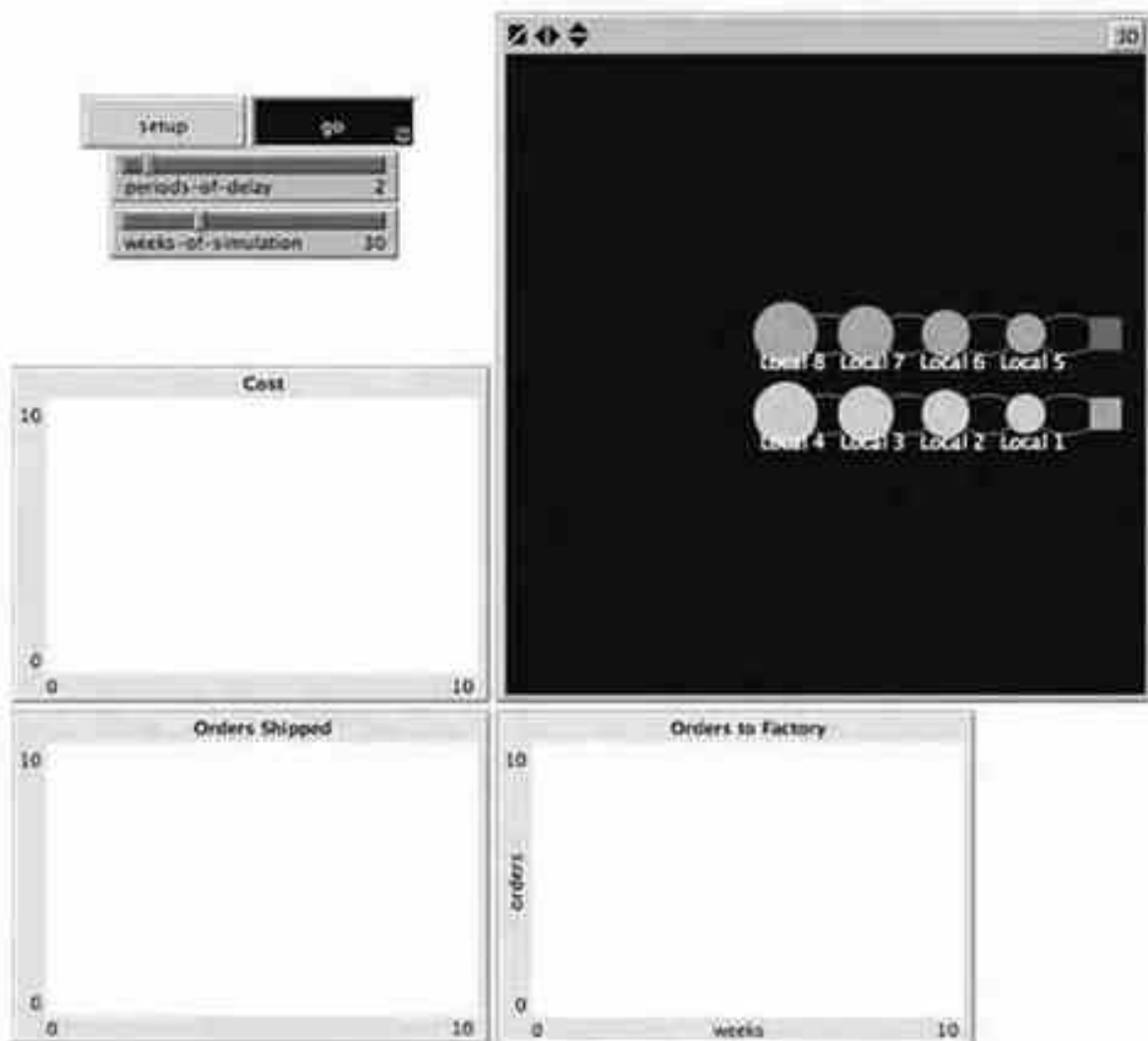
HubNet

- HubNet (Wilensky & Stroup, 1999c) is a NetLogo feature that enables Participatory Simulations, which is a simulation method where humans control agents in an agent-based model

Tools To Extend NetLogo Models

- In the Fire model extensions and in the DLA extensions, we have seen that it is often useful to extend a model by replacing deterministic rules by probabilistic rules.
- Conversely, as we saw in modeling wind and long distance transmission in the Fire model
- It is sometimes useful to replace a probabilistic rule with an agent-based mechanism that generates the probabilities.
- And with each change to the model, we need to consider whether we also need to design a new metric for measuring model performance

NetLogo HubNet Root Beer Game model.



Summary

- We have illustrated how to extend four different models, each in three different ways, and we have explored how these models relate to some of the key concepts of ABM.
- Credits: Uri Wilensky book

Module – 110

Creating Agent-Based Models

Overview

- Here we will understand about how to design and build our own model from the ground up.

First Step Of Devising A Question

- This Module is intended to take you from the first step of devising a question or area you want to explore, all the way through designing, and building your model, to refining your question and revising your model, to analyzing your results and answering your question.

Sequence of Steps

- The sequence of steps are presented here in linear order, but in reality these steps fold back on each other and are part of an iterative exploration and refinement of the model and motivating question.

Three Main Sections

- To facilitate the process, this module and other modules are broken into three main sections:

1. Designing your model will take you through the process of determining what elements to include in your model,
2. Building your model,
3. Building your model will demonstrate how to take a conceptual model and create a computational object
4. (3) Examining your model will address how to run your model, create some results, and analyze those results to provide a useful answer to your motivating question.

Model Of An Ecological System

- Throughout this chapter we will be designing, building, and examining a simple model of an ecological system.
- The basic question that we will be addressing is: “ How do the population levels of two habitat-sharing animal species change over time? ”

Wolf Sheep Simple model

- For our purposes in this chapter, we will call this model the Wolf Sheep Simple model.

Summary

- In this chapter More important, the components that we will be developing in this model are basic components utilized in most ABMs.
- Credits: Uri Wilensky book

Module – 111

Designing Your Model

Overview

- Here in this Module we will understand about many ways of designing an agent-based model.
- Which you choose will depend on many factors including the type of phenomenon to be modeled, your level of knowledge of the content domain, your comfort with NetLogo coding

Phenomena-based Modeling

- We consider two major categories of modeling:
 - phenomena-based modeling and exploratory modeling
- In phenomena-based modeling , you begin with a known target phenomenon
- Typically, that phenomenon has a characteristic pattern,

Reference Patterns

- A characteristic pattern, known as a reference pattern .
- Examples of reference patterns might include common housing segregation patterns in cities, spiral-shaped galaxies in space, leaf arrangement patterns on plants, or oscillating population levels in interacting species.

Goal Of Phenomena-based Modeling

- The goal of phenomena-based modeling is to create a model that will somehow capture the reference pattern
- In ABM, this translates to finding a set of agents, and rules for those agents, that will generate the known reference pattern.

Exploratory Modeling

- The second core modeling form is exploratory modeling.
- This form is perhaps less common in equational contexts than it is in ABM
- In exploratory modeling with ABM, you create a set of agents, define their behavior, and explore the patterns that emerge.

Formulate Research Question

- Another distinction in modeling methodology is to what degree we specify a question to be answered by a model.
- At one end of the spectrum, we formulate a specific research question (or set of questions) such as “ How does a colony of ants forage for food? ”

Designing The Conceptual Model

- A third dimension is the degree to which the process of designing the conceptual model is combined with coding your model.
- In some cases, it is advisable to work out the entire conceptual model design in advance of any coding of the model. This is referred to as top-down design.

Top-down design

- In a top-down design, the model designer will have worked out the types of agents in the model, the environment they reside in, and their rules of interaction before writing a single line of code.

Bottom-up Design

- In bottom-up design, you choose a domain or phenomenon of interest with or without specifying a formal question.
- Using this approach, you would then start writing code relevant to that domain, building the conceptual model

Design Principle

- We will emphasize the top-down approach.
- Throughout the design process there is one major principle that we will use
- We call this the ABM design principle :
 - Start simple and build toward the question you want to answer

Components Of Design Principle

- There are two main components of this principle
- The first is to begin with the simplest set of agents and rules of behavior that can be used to explore the system you want to model.

1st Components

- “ Everything should be made as simple as possible, but not simpler. ”
- In the case of ABM, this means making your model as simple as possible given that it must provide you with a stepping-stone toward your final destination

2nd Component

- Second, always have your question in mind, which means not adding anything to your model that does not help you in answering your question
- This core ABM design principle is useful in several ways

Verification

- Another virtue of the ABM design principle is that, by keeping the model simple, you make it both more understandable and easier to verify.
- Verification is the process of ensuring that a computational model faithfully implements its target conceptual model.

Simpler Conceptual Model

- A simpler conceptual model leads to a simpler model implementation, which makes it easier to verify the model.
- Starting simple and building up also facilitates the process of just-in-time results.

Identifying a Question

- We will start by identifying a question we want to explore, which is required by the ABM design principle (top-down version).
- After that we will discuss what the agents in our models are and how they can act. Then we move on to the environment and its characteristics.

Summary

- In this Module we have seen what happens in an individual time step of the model.
- Finally, we discuss what measures we will be using to answer our question
- Credits: Uri Wilensky book

Module – 112

Choosing Your Questions

Overview

- Here, we understand why Choosing a question may seem to be a separate issue from model design?

Natural Progression

- The natural progression seems to be:
 - *First* choose a question,
 - *Second* build a model to answer that question.

- In many instances we will need to refine our questions when we start to think about it in an agent based way.

Wolf Sheep Simple model

- Our original question for the Wolf Sheep Simple model was:
 - “ How do the population levels of two species change over time when they coexist in a shared habitat? ”
- We will now evaluate whether this question is one that is amenable to ABM

ABM For Making Sense Of Systems

- Agent-based *modeling is particularly useful for making sense of systems* that have a number of interacting entities, and therefore have *unpredictable results*.
- As we discussed in chapter 1, there are certain problems and questions that are more amenable to ABM solutions

Consider A Different Modeling Method

- If our primary question of interest violates our guidelines, it may be an indication that we should consider a different modeling method.
- For example, we might be interested in examining the dynamics of two very large populations under the assumption

Assumption

- The *assumption* that the species are homogenous and well-mixed (no spatial component or heterogeneous properties) and that the population level of each species is simply dependent on the population level of the other species.

EBM vs. ABM

- If that is the case then we could have used an *equation-based model instead of an agent-based model* since EBM's work well for large homogeneous groups, and as we mentioned in previous chapters there is a classic EBM for this situation known as the *Lotka-Volterra differential*

Conceptualize The Agents

- *ABM will be more useful* to us if we are thinking of the agents as heterogeneous with spatial locations. This affects how we conceptualize the agents.
- One aspect of the animals that is likely to be relevant to our question is how they make use of their resources.

Agent's Interaction With Environment

- A third guideline is to consider whether the aggregate results are dependent on the *interactions of the agents* and on the *interaction of the agents with their environment*.

- For example, if one species is consuming another then the results will be dependent on agent interaction.

Time Dependent Processes

- Yet another guideline is that agent-based modeling is most useful for modeling *time dependent processes*.
- In the Wolf Sheep Simple model, our core interest lies in examining *how population levels change over time*.

Summary

- In This Module we understood question “ Can we find model parameters for two species that will sustain positive population levels in a limited geographic area when one species is a predator of the other
- Credits: Uri Wilensky book

Module – 113

A Concrete Example

Overview

- Now that we have identified our research question in detail here we will understand how it can be useful to consider a particular context for this research question.

Phenomena-based Agent-based Models

- Earlier, we discussed reference patterns as a source of phenomena-based agent-based models.
- Sometimes that reference pattern is the original inspiration for the model.

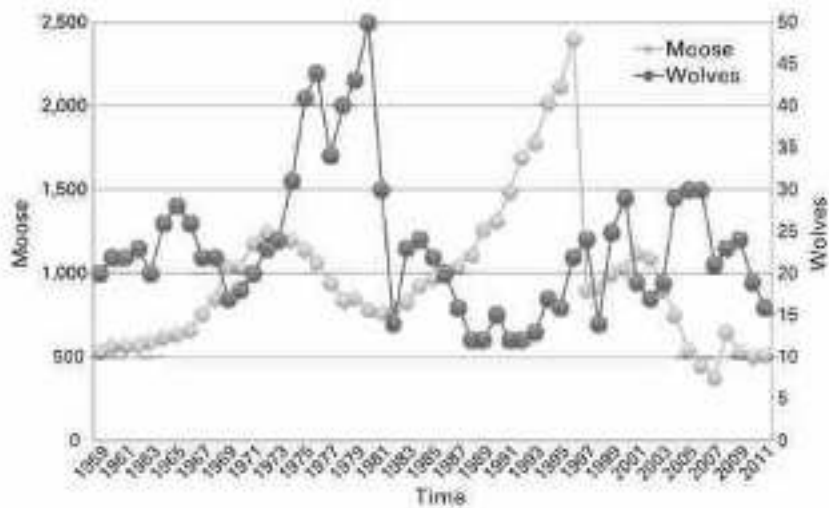
Refined Research Question

- we have refined our research question enough that we seek out a reference pattern that will help us test whether our model is a valid answer to the question.

Predator-prey Relations

- In the case of the predator-prey relations, there is a famous case of cohabiting small predator-prey populations in a small geographic area.
- This is the case of fluctuating wolf and moose populations in Isle Royale, Michigan.

Wolf And Moose Populations In Isle Royale



Five decades of fluctuating wolf and moose populations at Isle Royale. Note that when the wolf population peaks, the moose population is at a low point and, similarly, when the moose population peaks, the wolf population is at a low point.

Populations Exhibit a Rough Oscillation

- As we can see in the data from Isle Royale, the wolf and moose populations in Isle Royale have been sustaining themselves for more than fifty years without either species going extinct.
- The populations also exhibit a rough oscillation, with moose at a low when wolves peak and vice versa.

Reference Pattern

- We can refine our research question to:
 - “Can we find model parameters for two species that will sustain oscillating positive population levels in a limited geographic area when one species is a predator of the other and the second species consumes resources from the environment?”

Summary

- In this Module we have introduced you to classic examples of predator-prey modeling and to try to reproduce the oscillating sustained pattern of population levels.
- Credits: Uri Wilensky book

Module – 114

Choosing Your Agents

Overview

- Now that we have identified and contextualized our driving research question, Here we will understand that we can begin to design the components that will help us answer it.

What Are The Agents In The Model?

- The first question we should ask ourselves is:
- What are the agents in the model? When designing our agents, we want to choose those components of our model that are autonomous and have properties, states, and behaviors that could possibly have bearing on our question.

Agent Overload

- we must be careful to avoid agent overload
- Depending on the perspective one takes, almost any model component could be considered an agent.
- However, a model that is designed with an excess of agent types can quickly become unmanageable.

Entities Relevant to Research Question

- When choosing what are to be the agents in a model, it is important to concentrate on those autonomous entities which are most relevant to our research question.
- A related issue is the “granularity” of the agent. Every entity is composed of multiple smaller entities.

What is The Right Level of Entity to Choose?

- Should our agent be molecules or atoms? Body organs or cells? Some agents can be treated as mass properties.
- If we want to model a field of grass, we might not want to model every blade of grass, but instead choose “clumps” of grass as our agents.

Granularity Of Each Agent in temporal scale

- It is important that the granularity of each agent be at roughly the same level.
- For instance in the temporal scale, if you are modeling the sheep actions at the level of days of activity, but the grass minute by minute, that can be difficult to reconcile.

Granularity of Each Agent In Physical Scale

- In the physical scale, if there is more grass than the sheep can consume then you will not see many interesting behaviors since grass will not serve as a limiting condition.

Proto-agents

- By the term proto-agent, we mean agents that do not have individual properties, states, or behaviors but instead inherit some or all of their characteristics from a global agent type.
- For example, in the Wolf Sheep Simple model we might desire to have a human hunter who interacts with the other two species.

Create a Simpler Proto-agent

- There is no need to build the hunter as a full agent at the start; instead we can create a simpler proto-agent that has the ability to kill off a random percentage of the population.

Three Agent Types

- We model the predators, which we will call wolves, and the prey, which we will call sheep, and the resources the sheep consume, grass.
- We could have added many other agents to this model.
- However, by choosing just wolves, sheep, and grass, we stick to the ABM design principle.

Summary

- Here we have seen We have the two simple mobile agent types, and one stationary agent type to model the environment, and those are the minimal set of agents necessary to answer our question of what parameters will allow two populations to coexist in a limited geographic area.
- Credits: Uri Wilensky book

Module – 115

Choosing Agent Properties

Overview

- Here we will understand that Agents have properties that distinguish them from other agents.
- It is important to determine these properties in advance so we can conceptualize the agent and design the agents' interaction with each other and with the environment.

Wolf Sheep Simple model

- In the Wolf Sheep Simple model we give the sheep and wolves three properties each:
 - An energy level, which tracks the energy level of the agent,
 - A location, which is where in the geographic area the agent is,

Three Properties

- A heading, which indicates the direction the agent is currently moving or would be moving

- The energy property is not merely describing temporary energy (such as whether an animal is fresh or fatigued).

Energy Property

- Rather, “energy” incorporates some notion of the amount of “vitality” in a creature, abstracting away the messy details of metabolism, calorie storage, or starvation, and condensing it all into a single measure.

Adding Additional Properties

- We could add additional properties and some of them might be useful for future extensions.
- For instance, we could add a movement speed and allow different agents to move at different speeds, or an offense/defense capability that affects the ability of the individual.

Summary

- Here in this Module we have seen the behavior that each of these two agent types exhibit in wolf sheep model
- Further we will see what makes agents different from each other?
- Credits : Uri Wilensky

Module – 116

Choosing Agent Behavior

Overview

- Here we will understand how it is important to determine what kind of behavior the agents can exhibit.
- These behaviors are necessary to describe how agents interact with each other and the environment.

Wolf Sheep Simple Model

- In the Wolf Sheep Simple model, sheep and wolves share many common behaviors.
 - They both have the ability to turn randomly, move forward, reproduce and die.
- However, sheep and wolves differ

Differentiates The Two Agents

- Sheep have the ability to consume grass and wolves have the ability to consume sheep.
- This differentiates the two species/agent types from each other.
- Of course, once again there are many other behaviors we could prescribe for these agents.

- For instance, we could give sheep the ability to huddle in herds to defend against wolf attacks, or the ability to fight back.
- The wolves could have the ability to move at different speeds from a walk to a run or to chase sheep.

Differentiates The Two Agents

- The behaviors we have described (moving, reproducing, eating, and dying) are reasonable choices for a simple model that can address our research question.
- For the grass clump agents, we give one simple behavior, the ability to grow.

Summary

- Here we have seen how agents have different behaviors from each other so we can easily propose our research question on basis of their differentiated properties
- Credits : Uri Welinsky

Module – 117

Choosing Parameters of the Model

Overview

- Here we will understand about what is the next step to define what attributes of the model we will be able to control through parameters.

Several Possible Parameters Of Interest

- There are several possible parameters of interest in the Wolf Sheep Simple model.
 - For instance, we will want to be able to control the number of initial sheep and wolves.
- This will enable us to see how different values of the initial population levels affect the final population levels.

Possible Parameters

- Another factor to control is how much energy it costs an agent to move.
- Using this parameter, we can make the landscape more or less difficult to traverse, and thus simulate different types of terrain.

Parameters For Controlling The Energy

- Related to the cost of movement is the energy that each species gains from food.
- Thus, we choose to have parameters for controlling the energy gained from grass and the energy gained from sheep.

Homogenous Parameters

- we will need a parameter for the grass re-growth rate.
- There are many other parameters that we could have included in this model.
- The parameters we chose are homogenous across the model.
- In other words, one sheep will gain the same energy from grass as any other sheep.

Heterogeneous Parameters

- However, we could make this model more heterogeneous by drawing the energy gain for each sheep from a normal distribution, and have two model parameters that control the mean and variance of the energy gain.

Add Parameters to Control Aspects

- We could also add parameters to control aspects that we are currently planning to specify as constant values in the code.
- For example, we did not create a parameter to control the speed of the agents.

Summary

- Here we have seen how to create some parameters that enable us to control the model, so that we can easily examine different conditions.
- Credits : Uri Welinsky

Module – 118

Summary of the Wolf Sheep Simple Model Design

Overview

- Now that we have gone through the major design steps, we can create a summary document that describes our model design.

Wolf Sheep Simple model Summary

- The Wolf Sheep Simple model can be described in the following way:
 - Driving Question *Under what conditions do two species sustain oscillating positive population levels in a limited geographic area when one species is a predator of the other and the second species consumes limited but regenerating resources from the environment?*

Agent

- **Agent Types:**
 - Sheep, Wolves, Grass

- **Agent Properties:**
 - Energy, Location, Heading (wolf and sheep), Grass-amount (grass)
- **Agent Behaviors:**
 - Move, Die, Reproduce (wolf and sheep), Eat-sheep (wolf only), Eat-grass (sheep only), Re-grow (grass)

Parameters

- **Parameters:**
 - Number of Sheep, Number of Wolves, Move Cost, Energy Gain From Grass, Energy Gain From Sheep, Grass Re-growth Rate
- **Time Step :**
 - 1. Sheep and Wolves Move
 - 2. Sheep and Wolves Die
 - 3. Sheep and Wolves Eat
 - 4. Sheep and Wolves Reproduce
 - 5. Grass Re-grows

Measures

- **Measures:**
 - Sheep Population versus Time, Wolf Population versus Time
- For instance, if you are using *the top-down design process* then you might look over the following set of questions and write down answers to them as a provisional guide

Set Of Questions

- What part of your phenomenon would you like to build a model of?
- What are the principal types of agents involved in this phenomenon?
- In what kind of environment do these agents operate? Are there environmental agents?
- What properties do these agents have (describe by agent type)?
- What part of your phenomenon would you like to build a model of?
- What are the principal types of agents involved in this phenomenon?
- In what kind of environment do these agents operate? Are there environmental agents?
- What properties do these agents have (describe by agent type)?

Bottom-up Approach

- A more bottom-up approach would not start with these questions.
- Instead, you might know only that you want to build some kind of ecological model and could begin by creating some sheep and have them spread out in the world.

Implementing Some Behaviors

- In order for the sheep to eat, you might decide to add grass to the model.
- Having sheep eat grass provokes the question, “ What eats sheep? ” Therefore, you modify the model to include wolves, as predators for the sheep.
- This process allows the wolf sheep model to be designed

Summary

- In this Module we summarize Wolf Sheep Model in few steps
- Credits : Uri Welinsky

Module – 119

Examining a Model

Overview

- Here we will examine the model in detail We did not calibrate our model from real-world data, so the parameter values themselves are not important.
- But, discovering that there exist model parameters that exhibit our reference pattern gives us insight into the natural phenomenon we were trying to model

Parameters of two Species

- We have partially answered our research question:
 - *What parameters of two species will sustain oscillating positive population levels in a limited geographic area when one species is a predator of the other and the second species consumes limited but re-growing resources from the environment?*

Observing the Behavior Of Model Once

- Just observing the behavior of this model once does not provide us with a robust answer.
 - First, because many components of our model are stochastic in nature, there is no guarantee that the model when run again with the exact same parameters will exhibit the same behavior.
 - Second, we have found one set of parameters that allow wolves and sheep to exist, but are there other parameter settings?

- A more general answer would allow us to make statements about ranges of parameters, and relationships between parameters, that allow both the wolves and the sheep to survive.

Variety of Parameter Settings

- If we run the model many times with a variety of parameter settings, this will create a lot of data.
- Thus, to give us even a simple answer to our question, we really need to examine multiple runs across multiple different parameter settings and summarize this data in a useful way

Summary

- Here we have seen how to examine model.
- We will explore data analysis in greater detail in next modules, but before we leave the Wolf Sheep Simple model, we will perform a basic analysis, in order to develop a preliminary answer to our question.
- Credits : Uri Wilensky

Module – 120

Multiple Runs

Overview

- Here we will understand Whenever you have a model that has stochastic components, it is important to run the model several times so that you can be certain you have correctly characterized the behavior of the model.

Anomalous Behavior

- If the model is run only once, you might happen to see anomalous behavior that is not what the model usually produces.
- For instance, in the Wolf Sheep Simple model it might be possible to run the model and arrive at a state i.e.
 - In which there is simply one wolf and one sheep, and the sheep produces a second sheep often enough to keep the wolf fed, but not often enough to produce three sheep.

Run the Model Multiple Times

- It is important to run the model multiple times so that you can characterize the normal/average behavior of the model and not the aberrant behavior of the model

BehaviorSpace Tool

- NetLogo provides you with the BehaviorSpace tool (*Wilensky & Shargel, 2002*) that enables you to run a model for several iterations with the same (or different) parameter settings and collect the results.

Number of Time Steps

- One additional consideration when performing multiple runs is how many time steps to run the model for.
- Since we want to be able to compare the results across different random number seeds, it is useful to hold the number of time steps we run the model constant.

Summary

- Here we have seen how it is important to run the model several times to ensure that you have correctly characterized the behavior of the model.
- Credits : Uri Wilensky

Module – 121

Predator-Prey Models: Additional Context

Overview

- Here we will understand how parameter settings effects the population what is relationship between pray and predators depending upon parameter settings

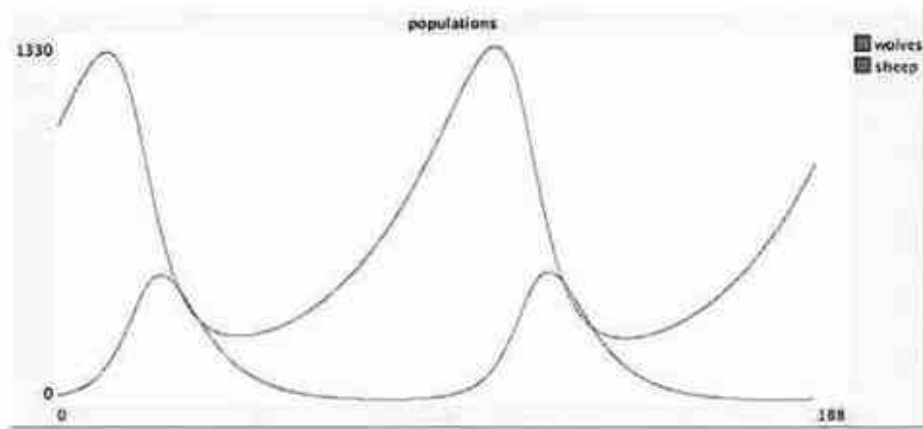
Predator-prey Interaction

- Lotka and Volterra has designed way to study ecologies and population dynamics in a concrete way
- They developed a system of equations to describe a two species predator-prey interaction.

Lotka-Volterra equations

- These simple equations showed that you could meaningfully model ecological systems with just a few parameters.
- And once you have a model of a system you can start to explore options for perturbing the system into favorable states.
- The general result of the Lotka-Volterra equations is that predators and prey populations move in cycles.
- Depending on the parameter settings, if the two populations are equal at the start, then the predators will begin consuming the prey; this will result in a decrease of the prey population.

Lotka-Volterra relationship



Lotka-Volterra relationship (Wolf Sheep Predation System Dynamics model)

Prey and Predator Population

- Eventually, the prey will die off to the point where predators cannot find any more prey to eat.
- This will in turn cause predators to die off because they have no prey to feed on
- As the prey population increases it will become easier for the predator population to find them and the predator population will increase as well.
- This, in turn, results in the prey population's decreasing trend, and the whole process begins again.

Lotka-Volterra Equations

- The Lotka-Volterra equations have been a standard way of describing the fluctuations in predator and prey populations
 - They are differential equations
 - They represent a continuous model of population change

“ Nano-sheep Problem ”

- The phenomenon is so common in differential equation-based modeling that it is sometimes referred to as the “ nano-sheep problem ” in specific reference to the wolf-sheep/predator-prey model
- The problem is that nano-sheep (i.e., a millionth of a sheep) do not exist: Sheep either exist or do not exist, and therefore modeling them with a continuous distribution can be problematic
- To rectify the nano-sheep problem, biologists have created agent-based models of predator-prey relationships

Classic Lotka Volterra differential equations

$$\frac{dx}{dt} = \alpha x - \beta xy$$

$$\frac{dy}{dt} = \delta xy - \gamma y$$

- x is the number of prey, y is the number of predators, and α , β , γ , and δ are parameters describing the interactions of the two species.

Gause Work

- In 1934, Gause showed that indeed for isolated predator and prey (with no other competing species) the agent-based model was more accurate in its predictions — indeed, extinctions happen much more frequently than the Lotka-Volterra equations predict.

Summary

- Here we have seen parameter settings are very important factors to control the population
- Credits : Uri Wilensky

Module – 122

Advanced Modeling Applications

Overview

- Here we will understand some Advanced Modeling Applications of ABM

Modeling of Food Webs

- One use of agent-based models in ecological systems is the modeling of food webs
- This combines ABM with another new complex systems methodology, network analysis

Structure of Ecological Interactions

- By understanding both the structure of ecological interactions via network analysis and the process of animal interactions via ABM, researchers gain a deeper insight into overall ecological systems

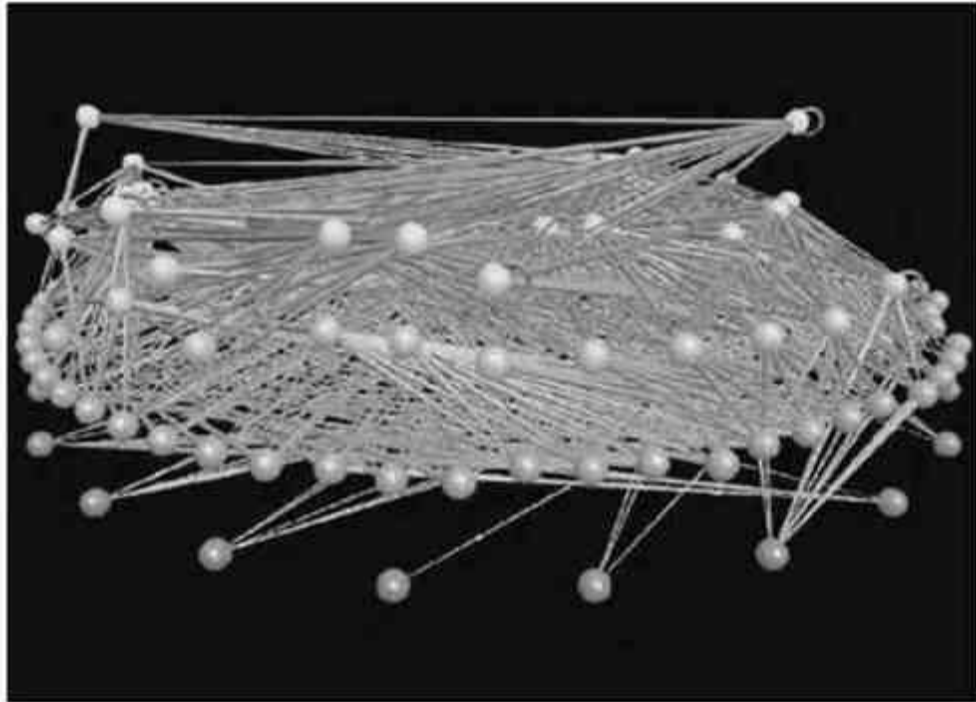
ABM and Ecological Modeling

- Another particularly interesting application of ABM and ecological modeling has been in doing prescriptive design of engineered systems to ameliorate human interventions in the environment.

Modeling Of Fish Populations

- Weber and his colleagues have done significant modeling of fish populations, and then examined the effect of various fish ladders near dams on salmon populations
- Agent-based modeling has been frequently used to model evolution of organisms

Little Rock Lake food web



ABM Approach as Natural Selection

- Evolution lends itself well to the ABM approach as natural selection and other evolutionary mechanisms can be thought of as computational algorithms.
- Models can be used to try to understand adaptation and speciation in the historical record

Summary

- In this module we have seen some Advance modeling application Moreover, as we will see in next modules mechanisms inspired by evolution can be employed as methods of machine learning
- Credits : Uri Wilensky

Module – 123

The Components of Agent-Based Modeling

Overview

- Here we will understand about Components of ABM

Components of ABM

- The main components of any ABM are
 - Agents ,
 - Environment ,
 - And Interactions.
- Agents are the basic ontological units of the model, while the environment is the world in which the agent lives
- Distinction Between Agents and Environment The distinction between agents and environment can be fluid, as the environment can some-times be modeled as agents.
- Interactions can occur either between agents or between agents and the environment.

Agent Actions

- Agent actions can also occur internally, directly affecting only the agent's internal state
- The environment is not merely passive; it can also act autonomously
 - For instance, in the models we discussed, the grass re-grows on its own.

Observer/User Interface

- The observer is an agent itself, but one that has access to all of the other agents and the environment
- The observer asks agents to carry out specific tasks.

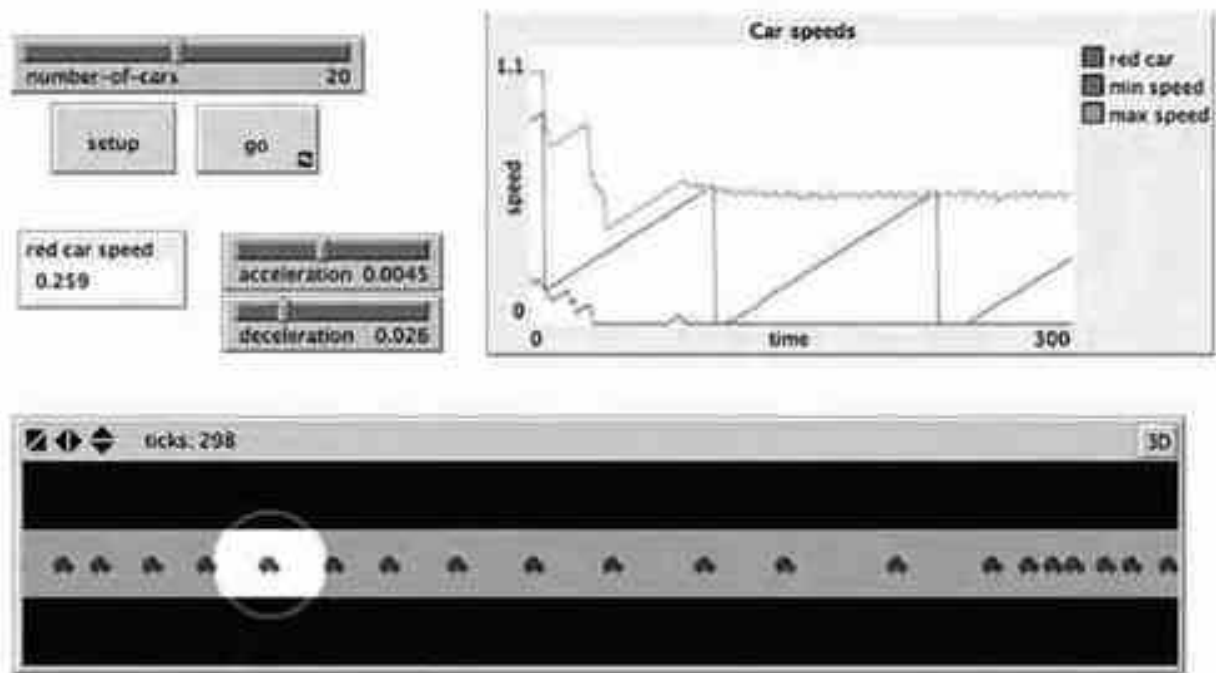
User Interface

- Users of agent-based models can interact with the agents by way of the User Interface, which enables the user to tell the observer what the model should do

Schedule

- The Schedule, is what the observer uses to tell the agents when to act.
- The Schedule often involves user interaction as well.
- In NetLogo models, the interface typically includes SETUP and GO buttons.
- The user presses SETUP and then GO to schedule these events to occur.

Traffic Basic model



Summary

- Here we have introduced basic components of ABM ,
- We will now describe each of these five components in further depth
- Credits : Uri Wilensky

Module – 124

Agents

Overview

- Agents are the basic units of agent-based modeling
- Here we will understand about, how it is important to choose the design of your agents carefully ?

Properties and Actions

- The two main aspects that define agents are
 - The properties that they have

- And the actions (sometimes called behaviors or methods) that they can execute
- Agent properties are the internal and external state of the agents — their data and description
- Agent behaviors or actions are what the agents can do
- Besides these two main agent attributes, there are also several issues that are related to agent design

“ Grain-size ”

- First is the issue of agent “ grain-size ”:
 - which is most effective for the chosen model? For example, if you are modeling a political system, do you want to model the individual actors or, instead, the political institutions or even each nation’s government as a single entity?

Agent Cognition

- A second factor to consider is agent cognition.
- How much capability do the agents have to observe the world around them and make a decision? Do they act in stimulus-response fashion? Or do they plan out their actions?

Proto-agents and Meta-agents

- Some special types of agents:
 - Proto-agents, which are not fully specified agents;
 - Meta-agents, composed of other agents.

Summary

- Here we have introduced about what is an Agent? Next we will see in detail about its properties and behavior
- Credits : Uri Wilensky

Module – 125

Properties & Behaviors

Overview

- Agent properties describe an agent’s current state, the items that you see when you inspect an agent
- Besides items that define the state of an agent (properties), it is also necessary to define how the agent can behave (the actions it can take)

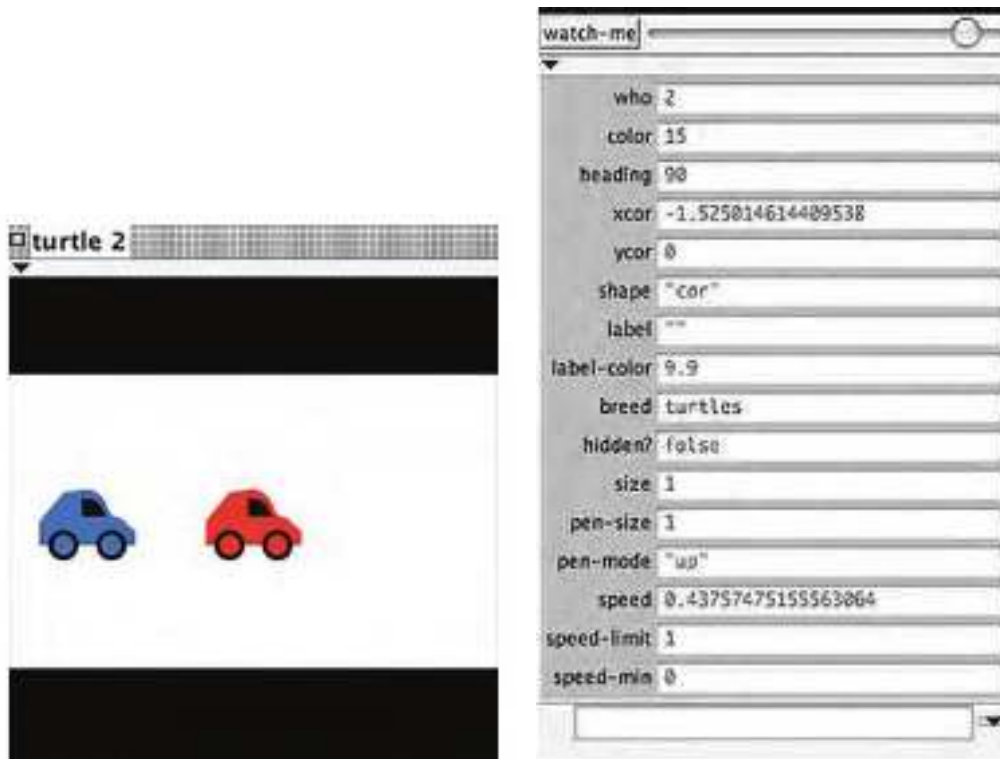
NetLogo

- In NetLogo, turtles and links have COLOR as a property, while patches have PCOLOR as a property.
- Likewise, turtles have XCOR and YCOR as properties, while patches have PXCOR and PYCOR
- The turtle can just execute SET COLOR PCOLOR to set its color

Link Properties

- Patches cannot directly access turtle or link properties, because a patch could have 0, 1 or many links or turtles on it
- Therefore, you must specify which turtle or link you are referring to when accessing their properties

Agent Properties for the Traffic Basic Model



Agent Properties for the Traffic Basic Model

- For this model, SPEED describes the current speed of the car, SPEED-LIMIT is the maximum speed of the car and SPEED-MIN, its minimum speed.
- When the Traffic Basic model starts up, all of these properties are set in the SETUP-CARS procedure:

```
set speed 0.1 + random-float .9
set speed-limit 1
set speed-min 0
```

Initialized Agent Properties

- We have initialized agent properties by setting them to constant values or to statistical distributions.
- Other ways to initialize agent properties are to set the values from a list or from a data file.
- Properties define an agent's current state, but they can also change to reflect how the state changes

Behaviors (Actions)

- An agent's actions or behaviors are the ways in which the agent can change the state of the environment, other agents, or itself
- In NetLogo, there are many behaviors that are predefined for the agents
 - It includes actions like FORWARD, RIGHT, LEFT, HATCH, DIE, and MOVE-TO.

Traffic Basic model

- In the Traffic Basic model, the agents have two additional behaviors that they can carry out —
- SPEED-UP-CAR and SLOW-DOWN-CAR — both of which modify the agents' speed:

```
;; turtle procedure
to slow-down-car
  set speed speed - deceleration
end

;; turtle procedure
to speed-up-car
  set speed speed + acceleration

end
```

Summary

- Behaviors are the basic way for agents to interact with the world while Action define an agent's current state
- Here we have seen it is also necessary to define how the agent can behave (the actions it can take).
- Credits : Uri Wilensky

Collections of Agents

Types of Agents

Agents are typically divided into three main types:

mobile agents that can move about the landscape;

stationary agents that cannot move at all; and

connecting agents that link two or more other agents.

NetLogo

In NetLogo, turtles are mobile agents, patches are stationary agents, and links are connecting agents. From a geometric standpoint, turtles are generally treated as shapeless area-less points, although they may be visualized with various shapes and sizes). Thus, even if a turtle appears to be large enough to be on several patches at the same time, it is only contained by the patch at the turtle's center (given by XCOR and YCOR). Patches are sometimes used to represent a passive environment and are acted upon by the mobile agents; other times, they can take actions and perform operations.

Patches and Turtles

A primary difference between patches and turtles is that patches cannot move. Patches also take up a defined space/area in the world; thus, a single patch can contain multiple turtle agents on it.

Links

Links are also unable to move themselves. They connect two turtles that are the “end nodes” of the link, but the visual representations of the links do move when their end nodes move.

Links are often used to represent the relationship between turtles. They can also represent the environment — e.g., by defining transportation routes that agents can move along, or by representing friendship/communication channels. Despite these differences, all three share the ability to behave, that is, to take actions and perform operations on themselves.

Breeds of Agents

Besides these three predefined agent-types, modelers can also create their own types of agent called breeds. The breed of an agent designates the category or class to which the agent belongs. In the Traffic Basic model, all of the agents are of the same type, so it is not necessary to distinguish between different agent breeds. Instead, we used agents of the “turtles” breed, the default breed in NetLogo. Different breeds of agents are required if different agents have different properties or actions.

Wolf and sheep

we had two breeds of agents, the “ wolf ” the “ sheep ” breeds. Even though these two breeds had the same set of properties, it was useful to create two breeds because

each had different characteristic actions — wolves eat sheep, while sheep eat grass. We defined the breeds at the beginning of the model:

```
breed [sheep a-sheep]
breed [wolves wolf]
```

At the same time, we can define the properties that breeds have:

```
turtles-own [energy]
```

In this case, both the wolves and the sheep have the same property of “ energy, ” but we could also give them separate properties. For instance, we could give the wolves a “ fangstrength ” property and the sheep a “ wooliness ” property. The fang-strength could be used to determine how successful a wolf is at killing sheep. Wooliness, by contrast, could be an indication of how well the sheep was able to fend off the wolves ’ attack, if we assume that the wolves sometimes get a mouthful of wool instead of flesh. If this were the case, we would add two additional lines:

```
sheep-own [ wooliness ]
wolves-own [ fang-strength ]
```

These properties are in addition to the properties that all agents have, so that the sheep would have “ wooliness ” and “ energy ” just as the wolves have “ fang-strength ” and “ energy. ”

Sets of Agents Breeds are particular collections of agents where the collections are defined by the kinds of properties and actions of their agents. We can also define collections of agents in other ways. NetLogo uses the term *agentset* to designate an unordered collection of agents, and we will also use this term/definition throughout this textbook. Usually we construct agentsets either by collecting agents that have something in common (e.g., agent location or other properties) or by randomly selecting a subset of another agentset.

In the Traffic Basic model, we could create a set of all the agents that have a speed over 0.5. In NetLogo, this is usually done with the **WITH** primitive. Here is an example of how we could create such an agentset that we could insert into the GO procedure in this model:

```
let fast-cars turtles with [speed > 0.5]
```

However, we usually want to ask these turtles to do something specific. For instance,

we can ask all of the turtles that have a high speed to set their size bigger so they are easier to see.

```
let fast-cars turtles with [speed > 0.5]
ask fast-cars [
  set size 2.0
]
```

The “let” statement above collects the fast cars into an agentset. If we don’t plan to “talk to” this agentset again, we can do without the LET and instead construct the agentset “on the fly” and ask the turtles directly:

```
ask turtles with [speed > 0.5] [
  set size 2.0
]
```

If you insert this code into the model, you will soon see that all of the cars are big. This is because we never told the turtles to set their size to small again, once their speed has dropped below 0.5. To do that we would need to add some more code. One way to accomplish this would be to use another ask.

```
ask turtles with [speed > 0.5] [
  set size 2.0
]
ask turtles with [speed <= 0.5] [
  set size 1.0
]
```

Another way to accomplish the same goal, without creating agentsets, is to ask all the turtles to do something, but choose what actions they take based on their properties. In the previous example, all of the faster turtles took their actions before any of the slower turtles took their actions. In the following example, all of the agents are asked with the same ASK command, so the order in which the turtles take actions will be random. In this case the result of running the code will be the same, but depending on what actions the agents take (for instance, if they were to change their speed, instead of their size), the results could be different.

```
ask turtles [
  ifelse speed > 0.5 [
    set size 2.0
  ]
  set speed 1.0
]
```

Another method for creating agentsets is on the basis of their location. The Traffic Basic model does this in the GO procedure:

```
let car-ahead one-of turtles-on patch-ahead 1
ifelse car-ahead != nobody
[ slow-down-car ]
```

TURTLES-ON PATCH-AHEAD 1 creates an agentset of all of the cars in the patch in front of the current car. If there is at least one such car, it then selects one of these cars at random, using the ONE-OF primitive, and sets the speed of the current car to a little less than the speed of that car ahead. There are many other ways to access a collection of agents based on their location, such as using NEIGHBORS, TURTLES-AT, TURTLES-HERE, and IN-RADIUS.

Agentsets and Lists Throughout our examples, we have used both agentsets and lists, sometimes explicitly and sometimes implicitly. This may be a good time to stop and clarify what

they are, how they work, how they differ, and under what circumstances we would use one over the other. Agentsets and lists are both variables that can contain one or many other variables. We can specify that a variable is an agentset or a list when we create them, by using their respective constructor reporters. If we want to create an empty list, we can write

```
let a-list []
```

If we want, we can then add numbers, strings, and even turtles to the list.

```
;; we put items at the start of the list
set a-list fput 1 a-list
set a-list fput "and" a-list
set alist fput turtle 0 a-list
show a-list
;; prints [(turtle 0) "and" 1]
```

In contrast to lists, which can hold any type of item, an agentset can hold only agents.

Moreover, it can hold only agents of the same type, such as turtles, patches, or links.

NetLogo has special reporters for empty agentsets, no-turtles, no-patches, and no-links.

```
;; this creates an empty agentset of turtles
let an-agentset no-turtles
```

no-turtles is an empty turtle agentset (i.e., an agentset containing no turtles), so by setting an-variable to no-turtles, we specify that this variable is of the type agentset.

Now that we have an empty agentset, we can then add turtles to it by using the turtle-set reporter:

```
;; this adds turtle 0 to the agentset
set an-agentset (turtle-set turtle 0)
;; this adds turtle 1 and turtle 2 to the agentset
set an-agentset (turtle-set an-agentset turtle 1 turtle 2)
show an-agentset
;; prints (agentset, 3 turtles)
```

We can only put agents (turtle-breeds, patches and links) in agentsets, but we can put

anything we want, including agents, in lists. There are two important properties that set agentsets and lists apart from each other.

First, we can “ask” agentsets to do things, but we cannot ask lists to do things. So for instance, when we use

```
ask turtles [] ;; do stuff
```

what we are really doing is first creating an agentset of all turtles, and then asking all those turtles, in a random order, to do whatever we put in the brackets. If we try this with lists, we will simply get an error.

Second, agentsets are unordered. This means that whenever we invoke them, they will output a list of agents in a random order. Notice how showing a list shows both what is inside the list and the order in which it appears, but showing an agentset shows only what is inside the agentset.

So when we want to interact with turtles, when would we use one or the other? Unless

we have a very good reason to, we always use agentsets. The primary reason is that we usually want our turtles to do things in a random order. That is because there could be threats to model validity if the agents always execute in the same order. For instance, in the Wolf Sheep Predation model, some sheep would have an unfair advantage if they are always first to check if there is grass on their patches.

But sometimes we do want to determine the order in which turtles do things. For instance, it could be that we want some turtles to have an advantage. In that case we would need to create an ordered list, and order them by whatever parameter we think determines their advantage. If turtles have different information, we might, for example, want the turtles with more information to take their turns later.

```
turtles-own [information] ;; information determines how late they get their turn
;; (later is better because then they will know what everyone else did
;; before making their decision)

;; create an empty list
let a-list []
;; sort-on reports a list containing turtles sorted by the parameter specified
;; in the brackets
set a-list sort-on [information] turtles
```

a-list now contains all turtles sorted by information in ascending order. But, as we just discussed, we cannot ask lists of agents to do things. Rather, we must iterate through the lists, and ask each agent in the list to do what we ask. For this we can use the foreach command:

```
foreach a-list [
  ask ? [ ] ;; do stuff here
]
```

The “ ? ” is a special variable that takes on the value of each element of the list. So this code will iterate through each turtle in the list, and ask each in that particular order, to do what we specify in the brackets.

Similarly we can create lists of patches. Suppose we want to label the patches with numbers in left-to-right, top-to-bottom order. We can create a sorted list of patches and then label them using the code below:

```
;; patches are labeled with numbers in left-to-right,  
;; top-to-bottom order  
let n 0  
foreach sort patches [  
  ask ? [  
    set plabel n  
    set n n + 1  
  ]  
]
```

Agentsets and Computation Before we end our discussion of agentsets, it should be noted that once you ask an agentset to perform an action, all the agents collected at that moment (and only those agents) will perform the action. If one agent's (agent A) action causes another agent (agent B) in the agentset to no longer satisfy the collection criteria, B will still perform the action. Likewise, if A's action causes agent C, which is not in the agentset, to meet the collection criteria, C will still not perform the action.

Agent-set Ordering model

As an illustration, let's consider this code snippet from the Agentset Ordering model in the chapter 5 subfolder of the IABM Textbook folder in the models library:

```

to setup
  clear-all
  create-turtles 100 [
    set size random-float 2.0
    forward 10
  ]
end

to go
  ask turtles [
    set color blue
  ]
  ask turtles with [ size < 1.0 ] [
    ask one-of turtles with [size > 1.0] [
      set size size - 0.5
    ]
    set color red
  ]
  print count turtles with [ color = red ]
  print count turtles with [ size < 1.0 ]
end

```

The bolded code has the potential to change the set of agents with size < 1.0 by making some agents smaller, thereby changing the agentset defined in the first ASK.

Agent-set Efficiency model

```

;; GO-1 sets red patch labels to a small random number and
;; green patch labels to a larger random number
to go-1
  if any? patches with [ pcolor = red ] [
    ask patches with [ pcolor = red ] [
      set plabel random 5 ;; red patches are labeled 0-4
    ]
  ]
  if any? patches with [ pcolor = green ] [
    ask patches with [ pcolor = green ] [
      set plabel 5 + random 5 ;; green patches are labeled 5-9
    ]
  ]
  tick
end

```

This code computes each of the two agentsets PATCHES WITH [PCOLOR = RED] and PATCHES WITH [PCOLOR = GREEN] twice.

```
;; GO-2 has the same behavior as go-1 above. But it is more efficient as it
computes each of the patch agentsets only once.
to go-2
  let red-patches patches with [ pcolor = red ]
  let green-patches patches with [ pcolor = green ]
  if any? red-patches [
    ask red-patches [
      set plabel random 5
    ]
  ]
  if any? green-patches [
    ask green-patches [
      set plabel 5 + random 5
    ]
  ]
  tick
end
```

The procedure GO-2 below has the same behavior as GO-1, but it is more efficient in that it computes each of those agentsets only once.

A more pernicious issue arises if we change just one line of GO-1 and GO-2:

```
;; GO-3 shows what happens if patch colors are changed "on the fly"
to go-3
  if any? patches with [ pcolor = red] [
    ask patches with [ pcolor = red] [
      set pcolor green
    ]
  ]
  if any? patches with [ pcolor = green] [
    ask patches with [ pcolor = green] [
      set pcolor red
    ]
  ]
  tick
end
```

If you run this code, you might expect to get a picture that looks like figure 5.4; that is, you expect red patches to turn green and vice versa.

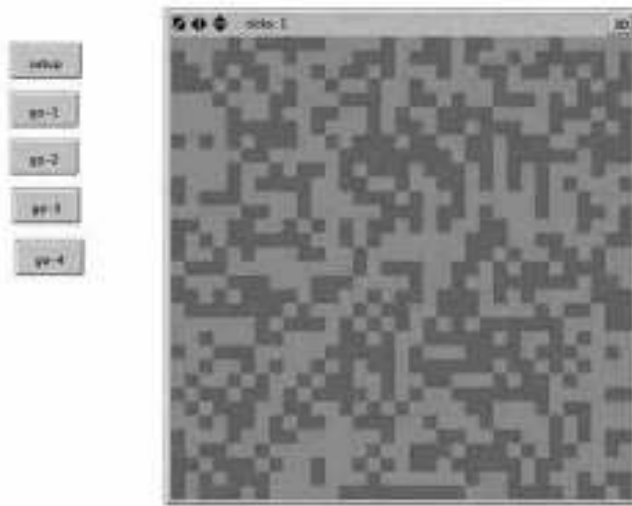


Figure 5.4
Constructing agents at the outset results in expected behavior

Module – 127

The Granularity of an Agent

One of the first considerations when designing an agent-based model is at what granularity should you create your agent — at what level of complexity is the agent you are modeling.

Level of Complexity for the Agents

So how do you choose the level of complexity for the agents in your model? The guideline is to choose agents such that they represent the fundamental level of interaction that pertains to your question about the phenomenon.

Tumor Model

For instance, if you look at the Tumor model (shown in figure 5.6) in the Biology section of the NetLogo models library, the agents are cells within the human body, since the research question the model examines is how tumor cells spread.

NetLogo Tumor model

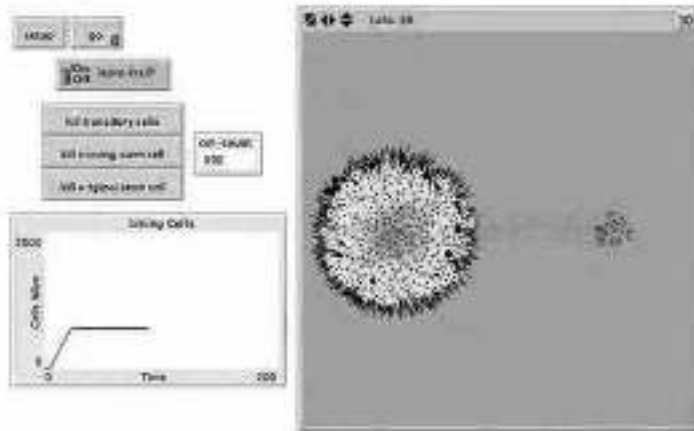


Figure 5.6
NetLogo Tumor model, <http://ccl.netmeister.org/models/tumor/> (Wilensky, 1995).

However, even though the AIDS model in the Biology section of the NetLogo models library (see figure 5.7) is also concerned with disease, the basic agents are humans instead of cells.

NetLogo AIDS model

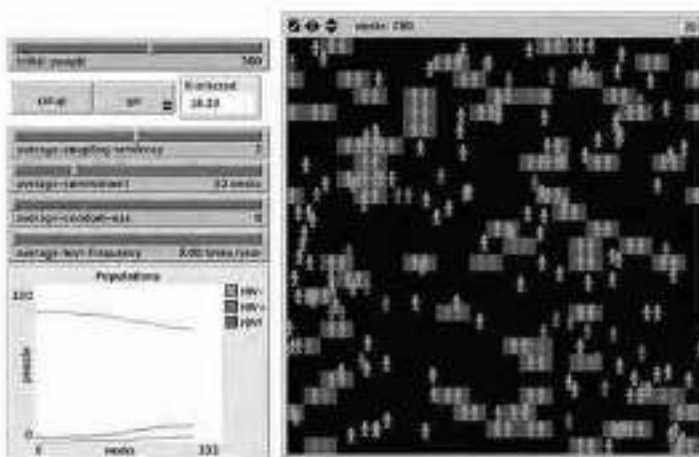


Figure 5.7
NetLogo AIDS model, <http://ccl.netmeister.org/models/aids/> (Wilensky, 1997).

Question Of Concern

This is because the AIDS model is more concerned with how the disease spreads between humans instead of how it spreads within the human body. In the Tumor model, the question of concern is how cells interact to create cancer.

Thus, the most important interactions happen at the cell level, and this makes cells a good choice for agents. In the AIDS model, the question of concern is how humans interact to spread disease, with humans as the agents being the most appropriate choice.

Choices Of Granularity

These two models highlight the relationship between the question being investigated, the interactions being modeled, and the resulting choices of granularity for the agents.

Level Of Complexity

Both individual cells and groups of cells can be valid choices for the level of complexity of these agents, and deciding between them will depend on details of the question you are asking as well as considerations of computational complexity.

Module – 128

Agent Cognition

As we have described, agents have different properties and behaviors. Still, how do agents examine their properties and the world around them to decide what actions to take? This question is resolved by a decision-making process called agent cognition.

Types Of Agent Cognition

We will discuss several types of agent cognition:

- *reflexive agents*,
- *utility-based agents*,
- *goal-based agents*, and
- *adaptive agents* (Russell & Norvig, 1995).

Often, these types of cognition are thought of in increasing order of complexity, with reflexive agents being the simplest, and adaptive agents the most complex. In reality, though this is not a strict hierarchy of complexity.

Instead, these are descriptive terms that help us talk about agent cognition and can be mixed and matched; for instance, it is possible to have a utility-based adaptive agent.

Reflexive

Reflexive agents are built around very simple rules. They use if-then rules to react to inputs and take actions (Russell & Norvig, 1995). For instance, the cars in the Traffic Basic model are reflexive agents. If you look at the code that controls their actions, it looks like this:

```
let car-ahead one-of turtles-on patch-ahead 1 ;;choose a car on the patch ahead
ifelse car-ahead != nobody [ ;; if there is a car ahead
  slow-down-car car-ahead ;; set car speed to be slower than car ahead
]
[ ;; otherwise, speed up
  speed-up-car ;; increase speed variable by the acceleration
]
;; --
fd speed
```

Put in words, this code says that if there are cars ahead, then slow down below the speed of the car in front; if there are not any cars ahead, then speed up. This is a reflexive action based on the state of the program, hence the name *reflexive agent*. This is the most basic form of agent cognition (and often a good starting point), but it is possible to make the agent cognition more sophisticated.

Utility-based Form of Agent Cognition

For instance, we could elaborate this model by giving the cars gas tanks and fuel efficiencies based on the speed they are going. We could then have the cars change their speeds in order to improve their fuel efficiencies. As a result, the agents might have to speed up and slow down at different times than they currently do to minimize their gas usage while still not causing accidents. Giving the agents this type of decision-making process would give them a *utility-based* form of agent cognition in which they attempt to maximize a utility function — namely, their fuel efficiency (Russell & Norvig, 1995).

Implementation Of Model Of Agent Cognition

To implement this model of agent cognition, we need to start by replacing our SPEED-UP-CAR procedure with a new procedure that accounts for the car's fuel efficiency.

```
;; choose a car on the patch ahead
let car-ahead one-of turtles-on patch-ahead 1
ifelse car-ahead != nobody [ ;; if there is a car ahead
  slow-down-car car-ahead ;; set car speed to be slower than car ahead
]
[ ;; otherwise, adjust speed to find ideal fuel efficiency
  adjust-speed-for-efficiency ]
```

We want the ADJUST-SPEED-FOR-EFFICIENCY procedure to maintain the same speed if the car is at the maximally fuel efficient speed. If it is not at its most efficient speed, the logic in this procedure should have the car speed up if it is moving too slow and slow down if the car is moving too fast. Additionally, the car would still need to slow down if it was about to crash into the car in front of it. As such, the true utility function includes an exception that gives a utility of 0 to any action that results in a crash. As a result, we can leave the first part of the code that slows the car before it crashes and just add ADJUST-SPEED-FOR-EFFICIENCY when there is no car directly ahead, as illustrated in the code below from the Traffic Basic Utility model, of the NetLogo models library.

```
;; car procedure
to adjust-speed-for-efficiency
  if (speed != efficient-speed) [ ;; if car is at efficient speed, do nothing
    if (speed + acceleration < efficient-speed) [
      ;; if accelerating will still put you below the efficient speed then accelerate
      set speed speed + acceleration
    ]
    ;; if decelerating will still put you above the efficient speed then decelerate
    if (speed - deceleration > efficient-speed) [
      set speed speed - deceleration
    ]
  ]
end
```

Utility Functions

In the language of utility functions, each car agent is minimizing a function f ,

defined by:

$$f(v) = |v - v^*|$$

where v is the current velocity of the car and v^* is the most efficient velocity. The constraint that is described in the code is that the v cannot be adjusted arbitrarily, but instead, can only be changed by a limited increment every time step. By trying different values for EFFICIENT-SPEED, you can obtain quite different traffic patterns from the original model. For low values of EFFICIENT-SPEED, the system can achieve a free-flow state (with no jams) on a consistent basis.

Goal-based Agents

Goal-based agents are intelligent agents that select actions based on achieving specific goals.

The third type of agent cognition we will discuss is goal-based cognition. Imagine that each car in the Traffic Grid model (see figure 5.8) from the social sciences section of the NetLogo models library has a home and a place of work, with the goal of moving from home to work in a reasonable amount of time. Now, not only do the agents have to be able to speed up and slow down, but they also have to be able to turn left and right. In this version of the model, the cars are **goal-based agents**, since they have a goal (getting to and from work) that they are using to dictate their actions.

Module – 129

Other Kinds of Agents

We have already discussed breeds and various types of agents, but there are two other special kinds of agents that deserve at least a brief mention.

- The first type are **meta-agents**, *agents made up of other agents*;
- The second is **proto-agents**, *placeholder agents that allow you to define interactions for your fully defined agents with other entities that have not been fully developed.*

Meta-Agents Many things that we would consider agents in reality are actually composed of other agents. In reality, all “agents” are composed of other agents; to cite the oft-told parable, “It’s turtles all the way down.” In other words, there is always a lower level of detail that you can use to describe an agent in your model, at least until you reach the most basic level so far described by physics. For instance, if we have selected a human to be the agent, he or she is actually composed of many subagents, with these subagents in turn being different depending on how you view the human. You could view the subagents of a human as the systems of the body — e.g., the immune and the respiratory systems — or you could view the subagents of a human as psychological aspects like the intellect and emotion. Moreover, these subagents are not the most basic level; in our example, the systems of the body are made up of organs, tissues, and cells.

Meta-agents and Sub-agents

At each of these levels, we are describing the relationship between **meta-agents** (*agents composed of other agents*) and **subagents** (*agents which compose other agents*). Still, agents can also be both meta-agents and subagents at the same time. For instance, organs are composed of cells and are meta-agents. However, they also play a compositional role in the systems of the human body and are thus subagents.

We can use meta-agents in our ABMs by defining the subagents that make up our agents and providing them with their own actions and properties.

Meta-agent Appears to be a Single Agent

From the perspective of another meta-agent, a meta-agent appears to be a single agent. If a person meets another person, they do not (usually) directly interface with the heart or the lungs (unless the meeting takes place at the surgical table). Instead, they interface with the person as a whole. Modeling agents and their interactions involves defining agent behaviors at a chosen level of detail, and sometimes using meta-agents "groups of agents acting as one" to simplify complex systems.

Modeling Agents and Their Interactions

When we think of modeling agents and their interactions, we have to determine what level of granularity we want to describe the agent behaviors. However, we always have the option of refining our model by converting agents into meta-agents that describe the agents that constitute other agents. Sometimes, it can be useful to represent the agents in our models not as autonomous individuals, but instead, as meta-agents composed of other agents. NetLogo doesn't include explicit language support for these meta-agents, though there are commands for locking together the movement of several agents (e.g., the TIE command) that may be useful in some circumstances. However, there is nothing to prevent you from designing models that have groups of agents representing single agents.

Proto-Agents

Proto-agents are placeholder entities without their own behaviors or properties, used to plan or simulate interactions with future, fully developed agents.

To truly be an agent within the agent-based modeling framework, an entity must have its own properties or actions. Sometimes, though, it can be desirable to create agents that, rather than having their own properties or behaviors, are instead placeholders for future agents. We call these agents **proto-agents**, and their primary purpose is to enable us to specify how other agents would interact with them if they

were fleshed out into full agents.

Example of Proto-agent

For instance, if you were creating a model of residential location decision making, you would have residents as agents; however, since where a resident lives is greatly influenced by places of employment and services (e.g., grocery stores and restaurants) you might also want to include these "service centers" as agents as well. The same level of detail necessary for the residents,

who are the focal agents of concern, may not be necessary for the service centers. Instead, they might be rendered as placeholders to represent where residents might potentially find jobs and transact business. However, as you continue to refine the model, you might give the service centers additional decision-making abilities. For instance, they might have a more elaborate model of market demand and decide where to locate using their own properties and beliefs about the future growth of the world. Keeping these agents as proto-agents early on means that when you add in the more richly detailed versions to the model, you do not have to go back and revise your resident agents. Instead, you can use the interaction events that they had already been using to interface with the service center proto-agents.

Module – 130

Environments

Another early and critical decision is how to design the *environment* of the agent-based model. The environment consists of the conditions and habitats surrounding the agents as they act and interact within the model. The environment can affect agent decisions, and, in turn, can be affected by agent decisions.

Environment Can Affect Agent

For instance, in the Ants model from chapter 1, the ants leave pheromone in the environment that changes the environment, and in turn changes the behavior of the ants. There are many different kinds of environments that are common in ABMs. In this section, we will discuss a few of the most common types of environments.

Implementation of an Environment

Before we discuss the types of environments, it is important to mention that the environment itself can be implemented in a variety of ways. First, the environment can be composed of agents such that each individual piece of the environment can have a full set of properties and actions.

NetLogo

In NetLogo, this is the default view of the environment — the environment is represented by the agentset of patches. This allows different parts of the environment to have different properties and act differently based on their local interactions.

Module – 131

Spatial Environments

Spatial environments in agent-based models generally have two variants:

- discrete spaces and
- Continuous spaces.

Continuous Spaces: In a mathematical representation, in continuous spaces between any pair of points, there exists another point,

Discrete Spaces: While in discrete spaces, though each point has a neighboring point, there do exist pairs of points without other points between them, so that each point is separated from every other point.

ABM Implementation

However, when implemented in an ABM, all continuous spaces must be implemented as approximations, so that continuous spaces are represented as discrete spaces where the spaces between the points are very small. It should be noted that both discrete and continuous spaces can be either finite or infinite.

Discrete Spaces: The most common discrete spaces used in ABM are *lattice graphs* (also sometimes referred to as *mesh graphs* or *grid graphs*), which are environments where every location in the environment is connected to other locations in a regular grid.

Toroidal Square Lattice

For instance, every location in a toroidal square lattice has a neighboring location up, down, to the left, and to the right. As mentioned, the most common representation of the environment in NetLogo is patches, which are located on a 2D lattice underlying the world of the ABM (see figure 5.9 for a colorful pattern of patches whose code is simply `ASK PATCHES [ SET PCOLOR PXCOR * PYCOR ]`). This uniform connectivity makes them different from network-based environments.

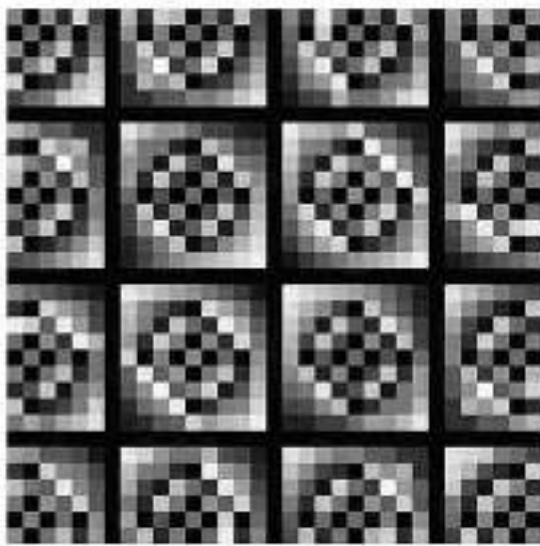


Figure 5.9
Patches displaying a range of colors.

The two most common types of lattices are:

- Square Lattices
- Hexagonal Lattices

Square Lattices The square lattice is the most common type of ABM environment. A square lattice is one composed of many little squares, akin to the grid paper used in mathematics classrooms.

There are two classical types of neighborhoods on a square lattice:

- The *von Neumann neighborhood*, consisting of four neighbors located in the cardinal directions (see figure 5.10a); and
- The *Moore neighborhood*, comprising the 8 adjacent cells (See figure 5.10b).

Module – 132

Network-Based Environments

Link: A *link* is defined by the two ends it connects, which are frequently referred to as *nodes*. However, we will use the network/node/link vocabulary throughout this module.

In NetLogo, links are their own agent-type.

Lattice networks

In fact, lattice graphs can also be called *lattice networks*, with the property that each position in the network looks exactly like every other position in the network. However, ABM environments usually do not implement lattice environments as networks for both conceptual and efficiency reasons. Additionally, using patches as the default topology allows for either discrete or continuous representations of the space, whereas a network is always discrete.

Random Networks

Network-based environments have been found useful in studying a wide variety of phenomena, such as the spread of disease or rumors, the formation of social groups, the structure of organizations, or even the structure of proteins. There are several network topologies that are commonly used in ABM. Besides the regular networks described earlier, the three most common network topologies are *random*, *scale-free*, and *small-world*.

In *random networks*, each individual is randomly connected to other individuals. These networks are created by randomly adding links between agents in the system. For example, if you had a model of agents moving around a large room and connected every agent in the room to another

agent based on which agent had the next largest last two digits of their social security number, you would probably create a random network. We show one simple methods for creating a random network. This code is also in the Random Network model in the chapter 5 subfolder of the IABM Textbook folder of the NetLogo models library

Code for creating a random network



Scale-free networks have the property that any subnetwork of the global network has the same properties as the global network. A common way to create this type of network is by adding new nodes and links to the system so that extant nodes with a large number of links are more likely to receive new connections (Barabási, 2002). This technique is sometimes called **preferential attachment**, since nodes with more connections are attached to preferentially. This method for network creation tends to produce networks with central nodes that have many radiating links; because of the resemblance to a bicycle wheel, this network structure is sometimes also called hub-and-spoke. Many real-world networks such as the Internet, electricity grids, and airline routes have similar properties to a scale-free network.

Module – 133

Special Environments

The two methods we have demonstrated for defining the environment, two-dimensional (2D) grid-based (i.e., lattice) and network-based, are both instances of “interaction topologies.” **Interaction topologies describe the paths along which agents can communicate and interact in a model.** Besides the two interaction topologies we have looked at so far, there are several other standard topologies

to consider. Two of the most interesting topologies involve the use of 3D worlds and Geographic Information Systems (GIS). 3D worlds allow agents to move in a third dimension as well as the two dimensions in traditional ABMs. GIS formats enable the importation of layers of real-world geographical data into ABMs. We will discuss each of these in turn.

3D Worlds

3D environments enable model developers to explore complex systems which are irreducibly bound up with a third dimension as well as sometimes increasing the apparent physical realism to their models.

There is a version of NetLogo called NetLogo 3D (it is a separate application in the NetLogo folder) that allows modelers to explore ABMs in three dimensions. There are many implementations of classic ABMs that have been developed for this environment in the NetLogo 3D models library (Wilensky, 2000). For instance, there is a three-dimensional version of the Percolation model (see figure 5.16).

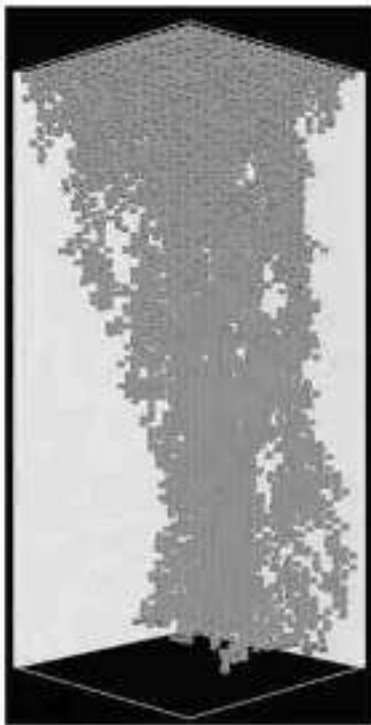
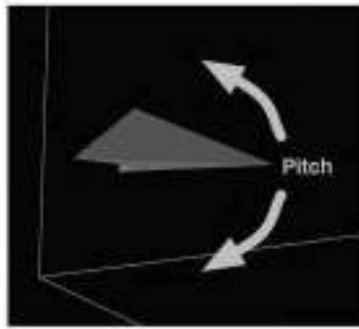


Figure 5.16
Percolation 3D. <http://ccl.netmeister.org/data/netlogo/models/Percolation3D> (Wilensky & Rand, 2006).

Working With 3D Worlds

Working with 3D worlds is not much different from working with 2D worlds. Although, using a 3D world does require additional data and commands. For instance, agents now have a Z-coordinate, and new commands are needed to manipulate this new degree of freedom.

In 3D, the orientation of an agent can no longer be described by its heading alone, we must also use *pitch* and *roll* to describe the orientation of the agent. If you think about an agent as an airplane, then the *pitch* of the agent is how far from horizontal the nose of the airplane is pointing. For instance, if the nose of the airplane is pointing straight up then the pitch is 90 degrees (see figure 5.17).



Using the airplane metaphor the *roll* of an agent is how far the wings are from horizontal. For instance, if the wings are pointing up and down then the roll of the agent is 90 degrees (see figure 5.18). In many 3D systems, heading has a new name, which is *yaw*, but in NetLogo we still use `HEADING` regardless of whether you are in 2D or 3D NetLogo to keep things consistent.

Module – 134

Interactions

Now that we have discussed both agents and the environments in which they exist, we will look at how agents and environments interact.

There are five basic classes of interactions that exist in ABMs:

- *agent-self*
- *environment-self*
- *agent-agent*
- *environment-environment*
- *agent-environment*

We will discuss each in turn along with some examples of these common interactions.

Agent-Self Interactions Agents do not always need to interact with other agents or the environment.

In fact, a lot of agent interaction is done within the agent.

For instance, most of the examples of advanced cognition that we discussed in the Agent section involve the agent interacting with itself.

The agent considers its current state and decides what to do. One classic type of self-agent interaction is birth.

Birth events are a typical event in ABMs

In these, one agent creates another agent. Below is the birth routine:

```
;; check to see if this agent has enough energy to reproduce
to reproduce
  if energy > 200 [
    set energy energy - 100 ;; reproduction costs energy to the parent
    hatch 1 [ set energy 100 ] ;; which is transferred to the offspring
  ]
end
```

As you can see, the agent considers its own state and, based on this state, decides whether or not to give birth to a new agent. It then manipulates its state, lowering its energy and creating the new agent. This is the typical way of having agents reproduce.

Environment-Self Interactions Environment-self interactions are when areas of the environment alter or change themselves. For instance, they could change their internal state variables as a result of calculations. The classic example of an environment-self interaction is when the grass regrows:

```
;; regrow the grass
to regrow-grass
  ask patches [
    set grass-amount grass-amount + grass-regrowth-rate
    if grass > 10 [
      set grass 10
    ]
  ]
  recolor-grass
end
```

Each patch is asked to examine its own state and increment the amount of grass it has, but if it has too much grass then it is set back to the maximum value it can contain.

Agent-Agent Interactions Interactions between two or more agents are usually the most important type of action within agent-based models. We saw a canonical example of agent-agent interactions in the Wolf Sheep Predation model when the wolves consume the sheep:

```

;; wolves eat sheep
to eat-sheep
  if any? sheep-here [ ;; if there are sheep here then eat one
    let target one-of sheep-here
    ask target [
      die
    ]
    ;; increase the energy by the parameter setting
    set energy energy + energy-gain-from-sheep
  ]
end

```

In this case, one agent is consuming another agent and taking its resources, whereby the wolf always eats the sheep. However, it is also possible to add competition or flight

to this model, where the wolf gets a chance of eating the sheep and the sheep gets a chance to flee. Competition is another example of agent-agent interaction.

Communication

A final example of agent-agent interaction is communication. Agents can share information about their own state as well as that of the world around them. This type of interaction allows agents to gain information to which they might not have direct access.

Environment-Environment Interactions: Interactions between different parts of the environment are probably the least commonly used type of interaction in agent-based models. However, there are some common uses of environment-environment interactions: one of these is diffusion. In the Ants model discussed in chapter 1, the ants place a pheromone in the environment, which is then diffused throughout the world via an environment-environment interaction. This interaction is contained in the following piece of code in the main GO procedure:

```

diffuse chemical (diffusion-rate / 100)
ask patches
  [ set chemical chemical * (100 - evaporation-rate) / 100
    ;; slowly evaporate chemical
    recolor-patch ]

```

Agent-Environment Interactions: Agent-environment interactions occur when the agent manipulates or examines part of the world in which it exists, or when the environment in some way alters or observes the agent. A common type of agent-environment interaction involves agents observing the environment. The Ants model demonstrates this kind of interaction when the ants examine the environment to look for food and sense pheromone:

```

to look-for-food ;; turtle procedure
  if food > 0
  [ set color orange + 1      ;; pick up food
    set food food - 1        ;; and reduce the food source
    rt 180                   ;; and turn around
  stop ]
  ;; face in the direction where the chemical smell is strongest
  if (chemical >= 0.05) and (chemical < 2)
  [ uphill-chemical ]
end

```

In the Ants model, the patches contain food and chemical, so the first part of this code checks to see if there is any food in the current patch. If there is food, the ant then picks up the food and turns around back to the nest, and the procedure stops.

Otherwise, the ant checks to see if there is chemical, wherein it follows the chemical in that direction.

Agent Movement

Another common type of agent-environment interaction is agent movement. In some ways, movement is simply an agent-self interaction, since it only alters the current agent's state.

In the Ants model the ants move around by “wiggling”.

Module – 135

Observer/User Interface

Now that we have talked about the agents, the environments, and the interactions that occur between agents and environmental attributes, we may discuss who controls the running of the model.

The *observer* is a high-level agent that is responsible for ensuring that the model runs and proceeds according to the steps developed by the model author.

The observer issues commands to agents and the environment, telling them to manipulate their data or to take certain actions. Most of the control that model developers have with an ABM is mediated through the observer. However, the observer is a special agent. It does not have many properties, though it can access global properties like any agent or patch can.

Properties Specific to the observer

The only properties that one could consider to be specific to the *observer* are those relating to the perspective from which the modeled world is viewed. For instance, in NetLogo the view may be centered on a specific agent, or focusing a highlight on a certain agent, using the FOLLOW, WATCH, or RIDE commands (See figure 5.21).

Schedule

The *schedule* is a description of the order in which the model operates. Different ABM toolkits can have more or less explicit representations of the schedule. In NetLogo, there is no single identifiable object that can be identified as “the schedule.” Rather, the schedule is the order of events that occur within the model, which depends on the sequence of buttons that the user pushes, and the code/procedures that those buttons run. We will first discuss the common SETUP/GO idiom which is employed in almost all agent-based models, and then move on to discuss some of the subtler issues concerning scheduling in ABMs.

SETUP and GO First of all, there is usually an initialization procedure that creates the agents, initializes the environment, and readies the user interface. In NetLogo this procedure is usually called SETUP and it executes whenever a user presses the SETUP button on a NetLogo model. The SETUP routine usually starts by clearing away all the agents and data related to the previous run of the model. Then it examines how the user has manipulated the various variables controlled by the user interface creating new agents and data to reflect the new run of the model. For instance, in Traffic Basic the SETUP procedure looks like this:

```
to setup
  clear-all
  ask patches [ setup-road ]
  setup-cars
  watch sample-car
end
```

The other main part of the schedule is what is often called the main loop, or in NetLogo, the GO procedure. The GO procedure describes what happens in one time unit (or tick) of the model. Usually that involves the agents being told what to do, the environment changing if necessary, and the user interface updating to reflect what has happened. In Traffic Basic the GO procedure looks like this:

```
to go
  ;; if there is a car right ahead of you, slow down to a speed below its speed
  ask turtles [
    let car-ahead one-of turtles-on patch-ahead 1
    ifelse car-ahead != nobody
    [ slow-down-car car-ahead ]
    ;; otherwise, speed up
    [ speed-up-car ]
  ]
  ;; don't slow down below speed minimum or speed up beyond speed limit
  if speed < speed-min [ set speed speed-min ]
  if speed > speed-limit [ set speed speed-limit ]
  fd speed ]
  tick
end
```

Types of Measurements

Heretofore, we have examined ABMs, modified them, built them from scratch, and analyzed their behavior. In this module, we will learn to employ ABM to produce new and interesting results about the domain that we are investigating. What kinds of results can ABMs produce? There are many different ways of examining and analyzing ABM data.

Choosing just one of these techniques can be limiting; therefore, it is important to know the advantages and disadvantages of a variety of tools and techniques. It is often useful to consider your analysis methods *before* building the ABM, to enable you to design output that is conducive to your analysis.

Modeling the Spread of Disease

If someone catches a cold and is coughing up a storm, he might infect others. Those that he comes into contact with — his friends, co-workers, and even strangers — may catch the cold. If a cold virus infects someone, that person might spread that disease to five other people (six now infected) before they recover. In turn, those five other people might spread the cold to five more people each (thirty-one are now infected), and those twenty-five people might spread the cold to five additional people (a hundred and fifty-six people are now infected). In fact, the rate of infection initially rises exponentially.

However, since this infection count grows so quickly, any population will eventually reach the limit of the number of people who can be infected. For instance, imagine that the 156 people mentioned above all work for the same company of 200 individuals. It is impossible for the remaining 125 people to each infect five new people, and thus the number of infected people will tail off because there is no one left to infect. As we have described it so far, this simple model assumes that each person infects the same number of people, which is manifestly not the case in real contexts. As a person moves through

their workspace, it might be the case that, they happen to not see many people in one day, whereas another individual might see many people. Also, our initial description assumes that if one person infects five people and another person infects five, there will not be any overlap. In reality, there is likely to be substantial overlap. Thus, the spread of disease in a workplace is not as straightforward as our initial description suggests. Suppose that we are interested in understanding the spread of disease, and we want to build an ABM of such a spread. How should we go about doing it?

First, we need some agents that keep track of whether they are infected with a cold or not. Additionally, these agents need a location in space and the ability to move. Finally, we need the ability to initialize the model by infecting a group of individuals. That is exactly how the NetLogo model we will be discussing in this module behaves (See figure 6.1). Individuals move around randomly on a landscape and infect other individuals whenever they come into contact with them.

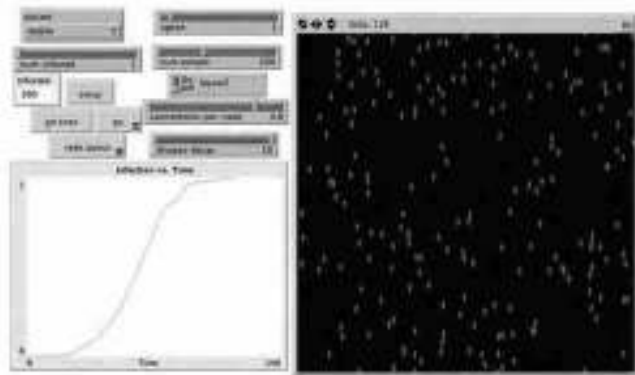


Figure 6.1
Spread of Disease model

Though this model is simple, it exhibits interesting and complex behavior. For instance, what happens if we increase the number of people in the model? Does the disease spread quicker throughout the population, or does it take a longer time because there are more people? Let us run the model at population sizes of 50, 100, 150, and 200, and examine the results. We will keep the size of the world constant, so that, as we increase the number of individuals, we are also increasing the population density. Along the way we will write down at what time the entire population becomes infected (see table 6.1).

Table 6.1
Infection Data

Population	50	100	150	200
Time to 100% Infection	419	188	169	127

Based on these results, we conclude that as the population density increases, the time to full infection dramatically decreases.

Let us suppose that we show this data to a friend of ours and she does not believe it.

She believes that the time to 100 percent infection should grow linearly with the population. She looks over the code and determines that it seems to match the description (a process called verification, which we will discuss in the next chapter). After that, she runs the model and collects the same data that we did (a form of replication). Her data is in table 6.2.

Table 6.2
Your Friend's Data

Population	50	100	150	200
Time to 100% Infection	305	263	118	126

These results do not support our friend's prediction that the time to 100 percent infection will grow as the population increases, but, on the other hand, they are quite different from the results that

were originally collected. In fact, the time to 100 percent infection for 150 and 200 increases in our friend's data, seemingly contradicting our original results. Moreover, if we run the model several more times you might again get different results. We need some methods for determining if there are trends in the data.

The data is inconsistent because most ABM models employ randomness in their algorithms — i.e., the code makes use of a random number generator. How the agents move around on the landscape is not specifically determined, but instead is the result of several calls to the random number generator at each time step that determine the actions of any particular agent. Moreover, these random decisions occur at least once for each agent in the population for each time step. Clearly, then, one set of runs is not enough to characterize the behavior of this model. Suppose, then, that we collect additional data for a set of population densities for ten different model runs as in table 6.3. Though most of these runs look more like our original results than our friend's results, it might be difficult to see clear trends, and to analyze the results overall. Thus, to describe these patterns of behavior it makes sense to turn to some statistics.

Table 6.3
Raw Data

Population	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
50	419	365	305	318	323	337	432	380	430	359
100	188	263	256	205	206	205	201	181	202	231
150	169	118	163	146	143	167	137	121	140	140
200	127	126	113	111	133	129	109	101	105	133

Module – 138

Here, we will understand about the different types of measurements in an ABM. Statistical Analysis of ABM:

- Moving beyond Raw Data
- Statistical results are the most common way of looking at any kind of scientific data
- The general methodology behind descriptive statistics is to provide numerical measures These measures summarize a large data set
- They also describe the data set in such a way that it is not necessary to examine every single value.

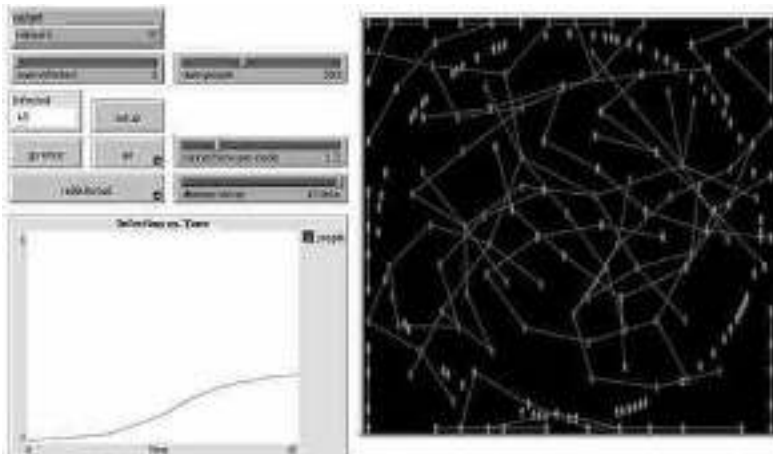
Summary Statistics

Population	Mean	Std. Dev.
50	366.8	47.38315802
100	213.8	27.40154894
150	144.4	17.65219535
200	118.7	12.12939497

Module – 139

Here we will Model the spread of disease in detail.

- If a cold virus infects someone, that person might spread that disease to five other people (six now infected) before they recover.
- In fact, the rate of infection initially rises exponentially.
- Suppose that we are interested in understanding the spread of disease, and we want to build an ABM of such a spread. How should we go about doing it?
- First, we need some agents that keep track of whether they are infected with a cold or not
- We need the ability to initialize the model by infecting a group of individuals
- Individuals move around randomly on a landscape and infect other individuals whenever they come into contact with them.



- We conclude that as the population density increases, the time to full infection dramatically decreases
- In the beginning, when the first person becomes infected, if there are not many other people around, the person has no one to infect, and thus the infection rate increases slowly.
- However, if there are many people around then there will be plenty of infection opportunities
- Moreover, at the end of the run, when there are only one or two uninfected agents, they will be more likely to run into someone with an infection if the population count is high.
- This is true despite the fact that the total number of people that need to be infected increases.
- To describe these patterns of behavior it makes sense to turn to some statistics

- Credits : Uri Wilensky

Module – 140

Here we will learn about the general methodology behind descriptive statistics that is to provide numerical measures that summarize a large data set and describe the data set in such a way that it is not necessary to examine every single value.

- Suppose we are interested in determining whether a coin is fair. (i.e., it is as likely to turn up heads when flipped as it is to turn up tails) then we can conduct a series of experiments where we flip the coin and observe the results.
- It is much easier to look at means and standard deviations than it is to examine large series of data
- (e.g., for HHHHTTHTTT, the observed probability is 0.5, and the expected outcome for ten trials is to observe five heads with a standard deviation of 1.58).
- To apply this technique to our Spread of Disease model in more depth, we can create summary statistics

Module – 141

- When you are trying to collect statistical results from an ABM you should run the model multiple times and collect different results at different points.
- Here we will understand how ABM toolkits will provide you with a way to collect the data from these runs automatically.
- In NetLogo there is a tool called BehaviorSpace. ABM toolkits are often full-featured programming languages, allowing you to write your own tools for creating experiments to produce the data sets you want to analyze.
- These tools will automatically run the model multiple times with multiple different settings and collect the results in some easy to use format like the CSV files mentioned earlier.
- Let us call this experiment “ population density”

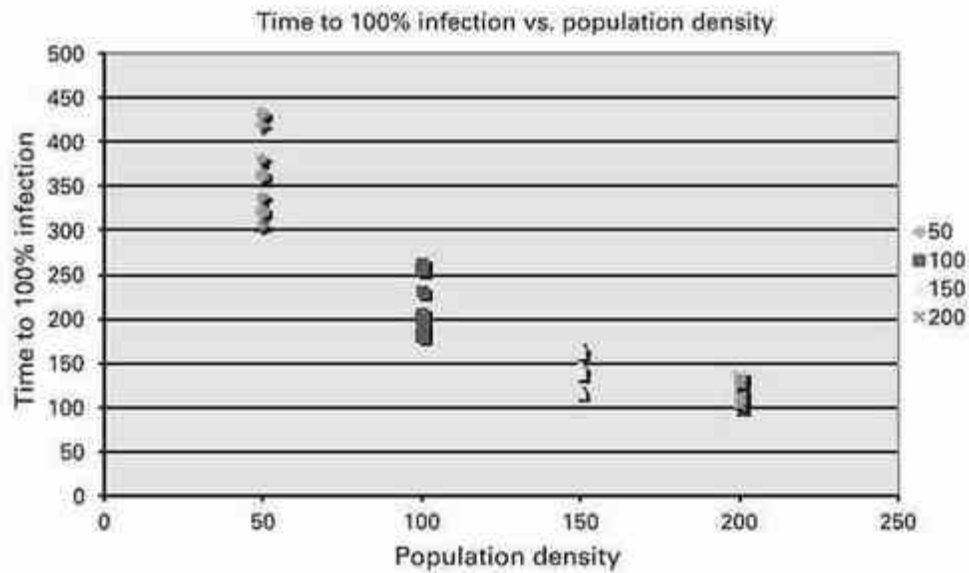


- The SETUP and GO commands allow you to specify any additional NetLogo code that you need to make the model start and go.
- “ Stop condition ” allows you to specify special stop conditions for each run, and
- “ Final commands ” allows you to insert any commands that you want executed between runs of the model
- In general in ABM, it is important to carry out multiple runs of your experiments so that you can determine if some result is truly a pattern or just a one-time occurrence.
- One common way is to start by manually running your model multiple times, but to get a better sense of the results it is usually much easier to use a batch experiment tool.
- We have illustrated the BehaviorSpace tool, which is the batch experiment tool for NetLogo.
- Credits : Uri Wilensky

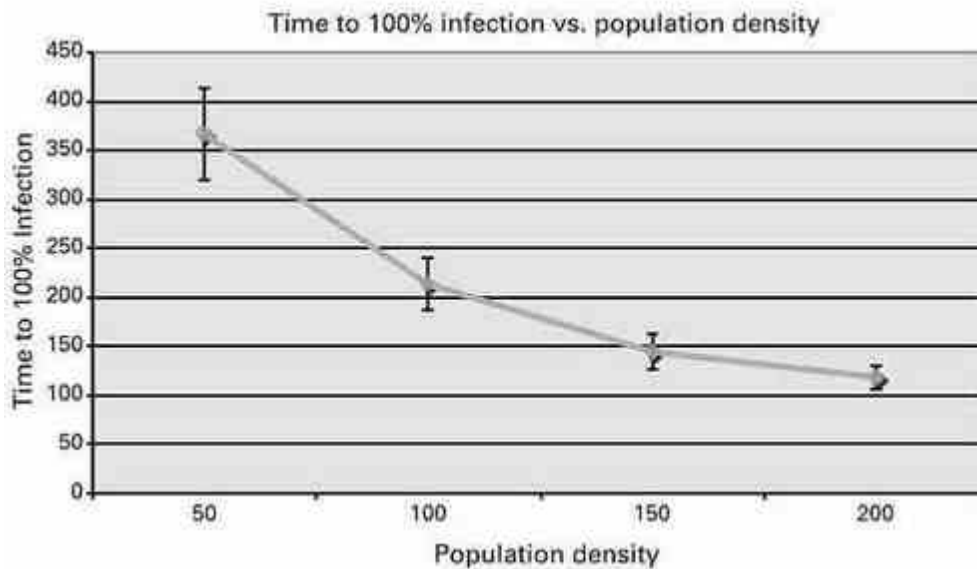
Module – 142

In NetLogo creating simple graphs is easy to do and will often suffice for simple data analysis.

But with complex data sets, designing a useful and immediately informative graph can be challenging and is the subject of an extensive body of literature.

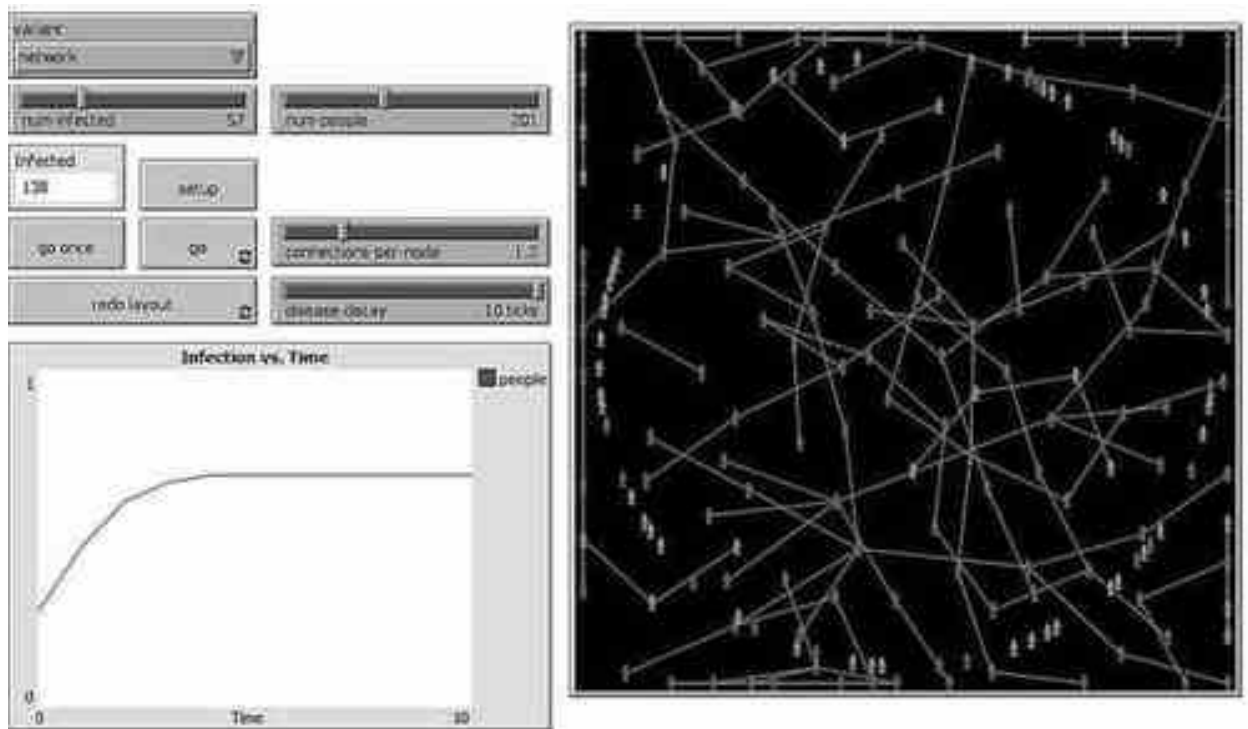


- We can quickly see how the data is distributed and how the data changes with population density



- This new figure might not be easier to understand than previous figure, but if there were one hundred data points in previous figure rather than ten data points, then a figure like this one might be very helpful.
- Many ABM toolkits include capabilities for continually updating graphs and charts during the running of a model and thus enable you to see the progress of the model temporally

- For instance, in the Spread of Disease model there is a graph that illustrates the change in the fraction of infected agents as time proceeds

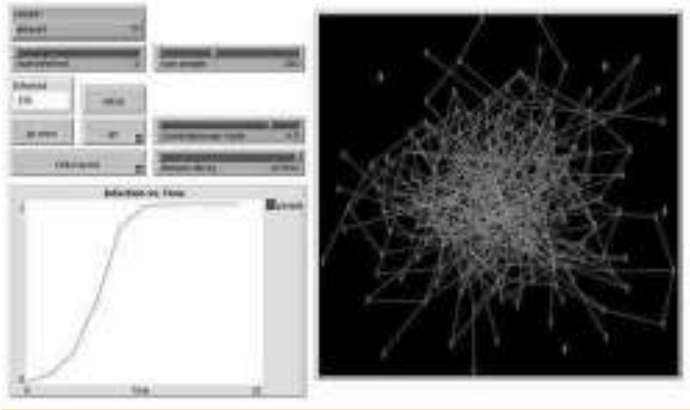


- This is one example of how we can use time series to help understand the behavior of a model.
- Summary: We have seen the use of graphs (plots) in simulation software
- Credits: Uri Wilensky

Module – 143

Here, we understand the analysis of “networks within ABM.”

- Interactions do not occur in physical space. But rather they can also occur across social networks.
- E.g. some diseases spread only through certain kinds of social networks. If we set the chooser to “network” we can explore this further.
- The property which determines how many connections.
- Each individual has to other individual in the network.
- Known as “connections-per-node”.



- A well known property of random graph is the average number of individuals infected.
- Grows substantially
- The connections-per-node exceeds 1.0
- Forms a giant component in the network.
- The property of average path length, which measure the distance between any two nodes in the network.
- Affect the spread of disease in the network.
- Another property is clustering co-efficient.
- Different properties and tools are there to measure and analyze a wide variety of metrics
- Associated with social network analysis (SNA).
- To summarize, each of these network properties can be analyzed as to their effect on the spread of disease.
- Reports and toolkits like UCInet can be used for further examination.
- Credits: Uri Wilensky book.

Module – 144

In this section we are going to see agent-to-environment and environment-to-agent transmission.

- The key areas to understand are:
 - Spread of disease model for examining.
 - Environmental interaction effect.

Module – 145

- Here, we understand the about the “correctness of a model”. Output relating the concerned issue must be accurate.
- Model accuracy is evaluated through validation, verification and replication.
- Implemented model corresponds to, and explains, some phenomenon in the real world.

- **Model verification: determining whether an implemented model corresponds to the target conceptual model.**
- Make sure that the model has been implemented correctly.
- Model replication is the implementation by one researcher or group of researchers of a conceptual model previously implemented by someone else.
- Set of results from a model that corresponds to the real world is not sufficient.
- Multiple runs are often needed to confirm that a model is accurate.
- Verification, validation, and replication collectively underpin the correctness, and thus utility, of a model.
- Credits: Uri Wilensky book.

Module – 146

- Here, we understand the role of “Verification” in Correctness of a Model.
- In larger models, the code can be difficult to understand as it evolves over time.
- Verification ensures the elimination of “bugs” from the code.
- The process of debugging becomes more difficult for complex models.
- Our goal, however, is to keep the process easy and simple.
- Build the model simply to begin with – it will be easier to verify. Expand the complexity of the model as necessary. This incremental approach makes it easy to verify the additional components.
- Even if all the components are verified, it is still possible that the system is not. Complications may arise from the interaction between model components.
- In this section we have examined the issue of verification of a model in the context of a simple ABM.
- Credits: Uri Wilensky book

Module – 147

Here, we understand need of “communication” for the correctness of a model.

- Sometimes a team of people builds a model. Other team members actually implement the model. Therefore, verification becomes critical.
- Communication is critical to ensure that the implemented model is correct.
- It is essential.
- E.g. In the voting model.
- Political scientists differentiate between Moore and Van Neumann neighborhoods.
- Small world network and hexagonal versus a rectangular grid.
- In an ideal situation, the model author and the implementer is the same person which averts the sort of communication errors.
- But when it is not, then there is often room for human error and misunderstanding.
- In the past it was difficult to be expert model implementer and model author.

- However, low-threshold ABM languages, such as NetLogo is narrowing the gap between author and implementer.
- We have seen how communication plays an important role in the correctness of a model.
- How communication can fill the gap between the implementer and the author of the model.
- Credits: Uri Wilensky book.

Module – 148

Here, we will understand how “Describing Conceptual Models” plays a role in analyzing correctness of a model.

- Implementing the voting model, we may realize subsequently that we never understand the idea completely. This could happen if we talked with a political scientist.
- Describing how we plan to implement the model is a specific type of document.
- We and the political scientist should have the same “conceptual model” in our mind.
- Describe the model in more formal terms. This includes a pictorial description of the model using flowcharts.
- Subsequently, we can convert the flowcharts into pseudo-code
- The goal of pseudo-code is to serve as a midway point between natural language and programming language.
- Pseudocode:

```
Voters have votes = {0, 1}

For each voter:
  Set vote either 0 or 1, chosen with equal probability
Loop until election
  For each voter
    If majority of neighbors' votes = 1 and vote = 0 then set vote 1
    Else If majority of neighbors' votes = 0 and vote = 1 then set vote 0
    If vote = 1: set color blue
    Else: color = green
  Display count of voters with vote = 1
  Display count of voters with vote = 0
End loop
```

- Other methods are UML, choosing a language similar to pseudo-code and NetLogo.
- We have learnt about processes involved in describing conceptual model which include flowcharts, pseudo-code, UML, and NetLogo.
- Credits: Uri Wilensky book

Module – 149

Here, We will discuss and understand how we can use “Verification Testing” for finding out about the correctness of a model.

- When implementing the design into code we need to follow ABM core design principles.

```
patches-own
[
  vote ;; my vote (0 or 1)
  total ;; sum of votes around me
]

to setup
  clear-all
  ask patches [
    if (random 2 = 0) ;; half a chance of this
    [ set vote 1 ]
  ]
  ask patches [
    if (random 2 = 0) ;; half a chance of this
    [ set vote 0 ]
  ]
  ask patches [
    recolor-patch
  ]
end

to recolor-patch ;; patch procedure
  ifelse vote = 0
  [ set pcolor green ]
  [ set pcolor blue ]
end
```

- This verification technique is a form of unit testing. We can modify the code without disrupting previous code.

Module – 150

Here, we will learn about going beyond verification using agent-based modeling.

- Sometimes results are not produced to what the implementers and authors hypothesized.
- Results of modeling can therefore end up causing further confusions for the scientists.
- Jagged edges can occur due to ties in votes.
- Design the model that neighbors do not change their votes if tied.
- Switch called CHANGE VOTE IF TIED.



```

patches-own [
  vote    ;; my vote (0 or 1)
  total   ;; sum of votes around me
]

to setup
  clear-all
  ask patches [
    set vote random 2 ;; set vote to either 0 or 1
    recolor-patch
  ]
  reset-ticks
  check-setup
end

to go
  ;; keep track of whether any patch has changed their vote
  let any-votes-changed? false
  ask patches [
    set total (sum [ vote ] of neighbors)
  ]
  ;; use two ask patches blocks so all patches compute "total"
  ;; before any patches change their votes
  ask patches [
    let previous-vote vote
    if total < 3 [ set vote 0 ] ;; if majority of your neighbors vote 0, set your vote to 0
    if total = 3 [
      ifelse award-close-calls-to-loser?

```

- Switch AWARD CLOSE CALLS TO LOSER?
- With both switches on, we have a different outcome.

Module – 151

Here, we will discuss sensitivity analysis and robustness.

- Creating the parameter to test the hypothesis of initial balance in one direction.
- Using BehaviorSpace, we can run the experiment varying from 25 to 75 percent increment.
- Two conditions:
- If no voter switch vote in the last step the model will stop.
- The model will stop after one hundred times the steps have executed.

```
patches-own
[
  vote ;; my vote (0 or 1)
  total ;; sum of votes around me
]

to setup
  clear-all
  ask patches [
    ifelse random 100 < initial-green-pct
    [ set vote 0 ]
    [ set vote 1 ]
  ]
  recolor-patch
  reset-ticks
  check-setup
end

to go
  ;; keep track of whether any patch has changed their vote
  let any-votes-changed? false
  ask patches [
    set total (sum [ vote ] of neighbors)
  ]
  ;; use two ask patches blocks so all patches compute "total"
  ;; before any patches change their votes
  ]
  ;; use two ask patches blocks so all patches compute "total"
  ;; before any patches change their votes
  ask patches [
    let previous-vote vote
    if total < 3 [ set vote 0 ] ;; if majority of your neighbors vote 0, set your vote to 0
    if total = 3 [
      ifelse award-close-calls-to-loser?
      [ set vote 1 ]
      [ set vote 0 ]
    ]
    if total = 4 and change-vote-if-tied? [
      set vote (1 - vote) ;; invert the vote
    ]
    if total = 5 [
      ifelse award-close-calls-to-loser?
      [ set vote 0 ]
      [ set vote 1 ]
    ]
    if total > 5 [ set vote 1 ] ;; if majority of your neighbors vote 1, set your vote to 1
    if vote != previous-vote [ set any-votes-changed? true ]
    recolor-patch
  ]
  if any-votes-changed? [
    ask patches [
      ifelse random 100 < initial-green-pct
      [ set vote 0 ]
      [ set vote 1 ]
    ]
    recolor-patch
  ]
  reset-ticks
  if not any-votes-changed? [
    stop
  ]
end
```

```

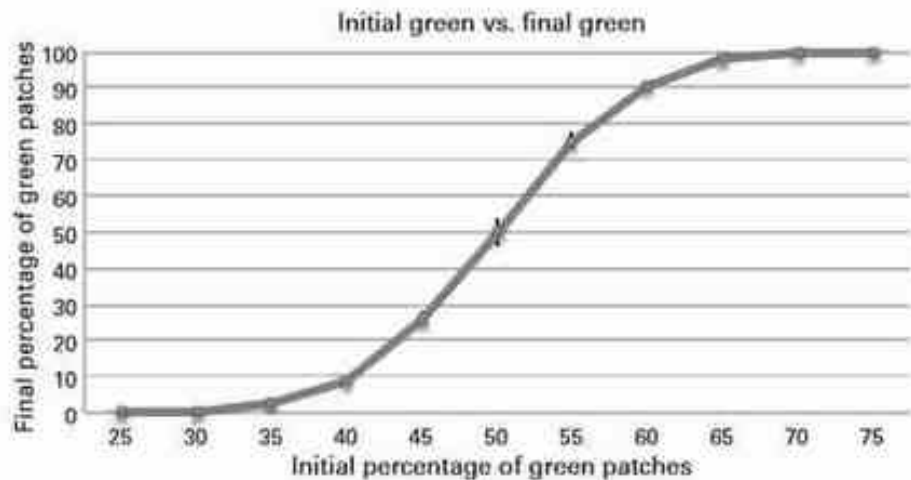
;; If the voters have stabilized, we stop the simulation
if not any-votes-changed? [ stop ]
tick
end

to recolor-patch ;; patch procedure
ifelse vote = 0
[ set pcolor green ]
[ set pcolor blue ]
end

;; This procedure checks to see if the SETUP procedure sets up the model with
;; roughly expected numbers, given the value of the initial-green-pct slider
to check-setup
let expected-green (count patches * initial-green-pct / 100)
let diff-green (count patches with [ vote = 0 ]) - expected-green
if diff-green > (.1 * expected-green) {
  print "Initial number of green voters is more than expected."
}
if diff-green < (- .1 * expected-green) {
  print "Initial number of green voters is less than expected."
}
end

; Copyright 2000 Uri Wilensky.
; See Info tab for full copyright and license.

```



- Past research methodologies for sensitivity analysis include Active Nonlinear Testing.
- We have seen the details of Sensitivity Analysis and Robustness
- Credits : Uri Wilensky book

Module – 152

Here, we are going to examine benefits and issues related to verification.

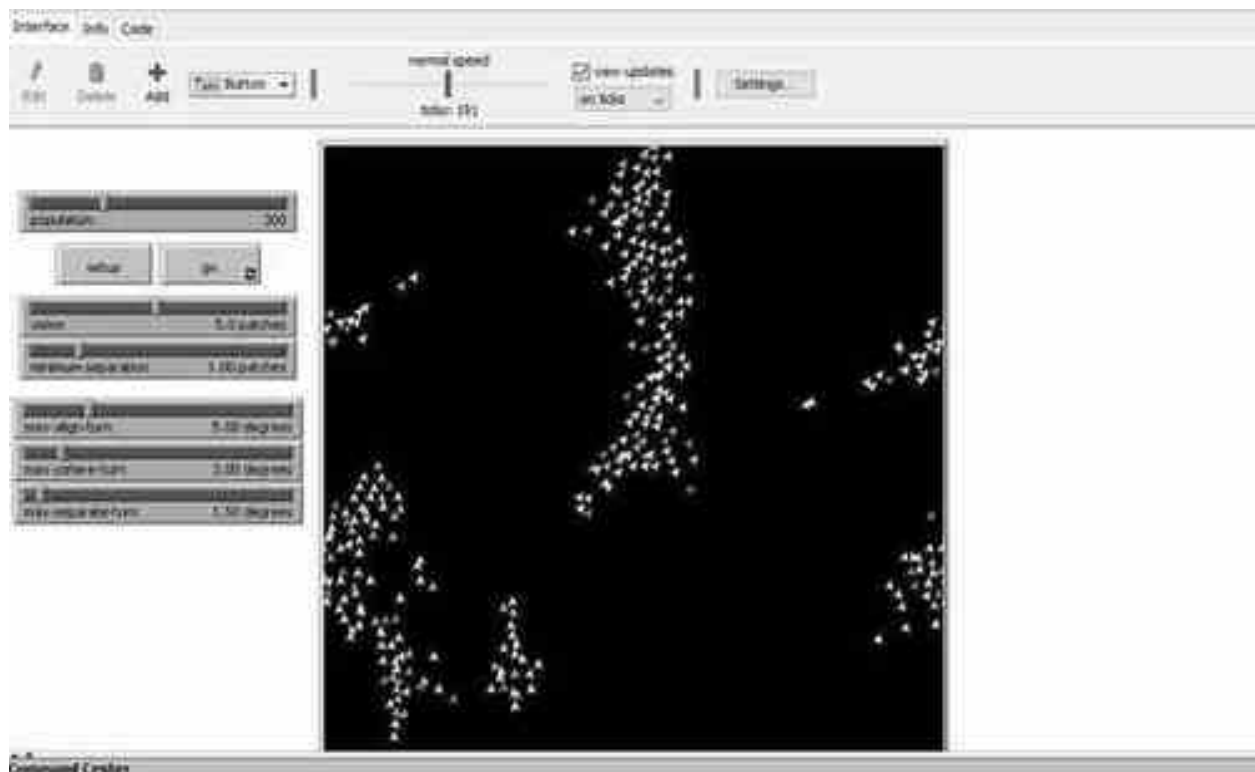
- We need to develop an understanding of the cause of unexpected outcomes and the impact of small changes.
- Implemented model needs to correspond well with the conceptual model.

- This all depends on the model's low level rules as well as an understanding of the mechanisms involved. A bug in the code can produce surprising results.
- Understanding the operation of the model is therefore quite essential. It is also important to note that the verification process is not binary.
- We have discussed verification benefits and issues.
- Credits: Uri Wilensky book

Module – 153

We will discuss the validation of agent based modeling.

- Validation involves corresponding between implemented model and reality.
- The various topics related to validation include:
- Two axis.
- Macrovalidation.
- Face validation.
- Flocking model.
- A classical agent based model.



```

to turtles-on [
  flockmates      ;; agentset of nearby turtles
  nearest-neighbor ;; closest one of our flockmates
]

to setup
  clear-all
  create-turtles population
  [ set color yellow - 2 + random 7 ;; random shades look nice
    set size 1.5 ;; easier to see
    setxy random-xcor random-ycor
    set flockmates no-turtles ]
  reset-ticks
end

to go
  ask turtles [ flock ]
  ;; the following line is used to make the turtles
  ;; animate more smoothly.
  repeat 5 [ ask turtles [ fd 0.2 ] display ]
  ;; for greater efficiency, at the expense of smooth
  ;; animation, substitute the following line instead:
  ;; ask turtles [ fd 1 ]
  tick
end

```

```

to flock ;; turtle procedure
  find-flockmates
  if any? flockmates
  [ find-nearest-neighbor
    ifelse distance nearest-neighbor < minimum-separation
    [ separate ]
    [ align
      cohere ] ]
end

to find-flockmates ;; turtle procedure
  set flockmates other turtles in-radius vision
end

to find-nearest-neighbor ;; turtle procedure
  set nearest-neighbor min-one-of flockmates (distance myself)
end

;;; SEPARATE

to separate ;; turtle procedure
  turn-away ([heading] of nearest-neighbor) max-separate-turn
end

;;; ALIGN

to align ;; turtle procedure
  turn-towards average-flockmate-heading max-align-turn
end

```

```

;; ALIGN
;; to align ;; turtle procedure
  turn-towards average-flockmate-heading max-align-turn
end

;; to-report average-flockmate-heading ;; turtle procedure
  ;; We can't just average the heading variables here.
  ;; For example, the average of 1 and 180 should be 9,
  ;; not 180. So we have to use trigonometry.
  let x-component sum [dx] of flockmates
  let y-component sum [dy] of flockmates
  ifelse x-component = 0 and y-component = 0
    [ report heading ]
    [ report atan x-component y-component ]
end

;; COHERE
;; to cohere ;; turtle procedure
  turn-towards average-heading-towards-flockmates max-cohere-turn
end

;; to-report average-heading-towards-flockmates ;; turtle procedure
  ;; "towards myself" gives us the heading from the other turtle
  ;; to me, but we want the heading from me to the other turtle,
  ;; so we add 180
  let x-component mean [sin (towards myself + 180)] of flockmates
  let y-component mean [cos (towards myself + 180)] of flockmates
  ifelse x-component = 0 and y-component = 0
    [ report heading ]
    [ report atan x-component y-component ]
end

```

```

;; HELPER PROCEDURES

;; to turn-towards [new-heading max-turn] ;; turtle procedure
  turn-at-most (subtract-headings new-heading heading) max-turn
end

;; to turn-away [new-heading max-turn] ;; turtle procedure
  turn-at-most (subtract-headings heading new-heading) max-turn
end

;; turn right by "turn" degrees (or left if "turn" is negative),
;; but never turn more than "max-turn" degrees
;; to turn-at-most [turn max-turn] ;; turtle procedure
  ifelse abs turn > max-turn
    [ ifelse turn > 0
      [ rt max-turn ]
      [ lt max-turn ] ]
    [ rt turn ]
  end

;; Copyright 1998 Uri Wilensky.
;; See Info tab for full copyright and license.

```

- Here, we have discussed the validation of agent based models.

Module – 154

Here, we will understand the difference between “macrovalidation and microvalidation” and examine their role in the validation of a model.

Microvalidation checks if individual agent behaviors match real-world counterparts, while macrovalidation evaluates if the overall model outcomes align with real-world system-level patterns.

- ABMs are built from agents hence we can directly compare them with those that exist in the real world.
- For instance, we can ask whether the agents have properties similar to real birds in flocking model.
- There are some limitation e.g. real birds can fly in three dimensions but our bird agents move in two dimension only. However, these limitations does not make our model invalid.
- We can also build the flocking model in three dimensions and examine the resultant flocks to see if they are relevantly different from the 2D flocks.
- The other major avenue of validation is to investigate the relationship between the global properties of the model and the flocking patterns of real birds, a process called macrovalidation.
- By showing that our model corresponds to the macro-level phenomenon, we further validate that our model is descriptive of real world systems.
- Macrovalidation tells you if you have captured the important parts of the system, whereas microvalidation tells you if you ’ve captured the important parts of the agent’s individual behavior.

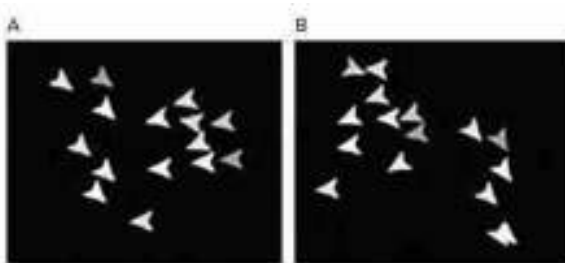


Figure 7.9
Two mini-flocks (A) before and (B) after a collision.

- We now understand how macrovalidation and microvalidation compares to each other.

Module – 155

Here we will discuss how “face validation” is different from “empirical validation”.

- Face validation is the process of showing that the mechanisms and properties of the model look like mechanisms and properties of the real world.
- Empirical validation is making sure that the model generates data. It corresponds to similar patterns of data in the real world.
- Face validity can exist at both micro and macro levels.

- Determining whether the birds in the model correspond to real birds is face microvalidity.
- While determining if the flocks correspond to the appearance of real flocks is face macrovalidity.
- Empirical validation sets a higher bar. Data produced by the model must correspond to empirical data derived from the real-world.
- Measures and numerical data generated both by the model and the actual phenomenon.
- Inputs and outputs in “ the real world ” are often poorly defined. It is hard to isolate and measure parameters from a real-world phenomenon.
- The process of finding the parameters and initial conditions that cause the model to match with real-world data is called calibration.
- We are now able to understand how these four types of validation (micro-face, macro-face, micro-empirical, and macro-empirical) characterize the majority of validation efforts carried out.

Module – 156

Here we will understand why it is important to validate a model.

- A valid model can be useful for extracting general principles about the world.
- Changing the mechanisms and parameters can often help predict what might occur in the real world.
- A model is not either valid or invalid:
- A model can be said to be more or less valid based upon how closely it has been compared to the real process it is modeling.
- A model is never inherently valid.
- Its validity comes from the context of what it is being used for.
- Something in the model corresponds to something in reality.
- The user compare his observation's with real world and use it as the basis of his validation.
- Here we have understood the benefits of validation and question it arises.

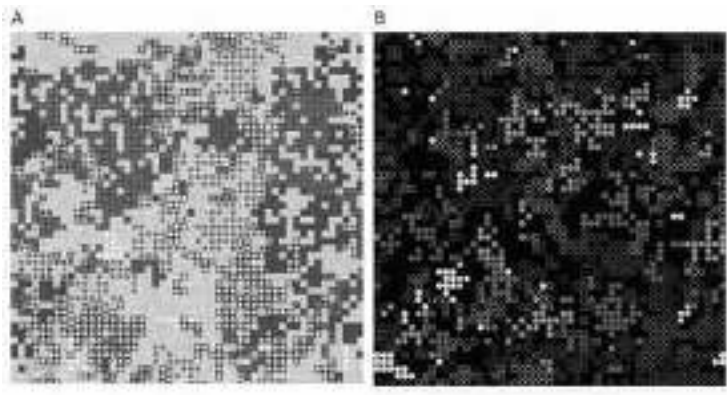
Module – 157

Here, we will understand about the “Replication”.

Replication

- Model replication is the implementation by one researcher or group of researchers of a conceptual model previously implemented by someone else.
- Scientists must publish the details of how the experiment was conducted.
- Then subsequent teams of scientists carry out the experiment themselves to ascertain.
- Replicating a physical experiment strengthens original results.
- Comparing both the experimental setup and ensuing results.

- Replicating a computational model serves this same purpose.
- Replicating a computational model increases our confidence.
- New implementation of the model has yielded the same results as the original.



- This model includes a mechanism for inheritance of strategies.
- The model suggests that “ethnocentric” behavior can evolve under a wide variety of conditions.
- Even when there are no native “ethnocentrics”.
- Here we understand about the basic concepts of replication and how it is useful for the correctness of a model.

Module – 158

Here, we will understand about the “Replication of Computational Models: Dimensions and Standards”.

Replication

- Replication refers to the creation of a new implementation of a conceptual model based on the previous implementation.

Dimensions

- An original model and replicated model can differ across at least six dimensions.
- Time
- Hardware
- Languages
- Toolkits
- Algorithms
- Authors

Successful Replication

- A successful replication is one in which the replicators are able to establish that the replicated model creates outputs sufficiently similar to the outputs of the original.

Replication Standard

- The criterion by which the replication's success is judged is called the replication standard.
- Different replication standards exist for the level of similarity between model outputs.

Categories of Replication Standard

- There are three categories of replication standard.
- Numerical identity
- Distributional equivalence.
- Relational alignment.

Data

- ABMs usually produce large amounts of data.
- much of which is usually irrelevant.
- Data that are central to the conceptual model should be measured and tested during replication.

Summary

- Here, we understand about dimensions of conceptual model like time, hardware, toolkit, language, algorithms and author.
- Then we discussed about replication standard and its categories.

Module – 159

Overview

- We are going to see the benefits of replication in this section.

Benefits of Replication

- Advances scientific knowledge.
- Model verification.
- Model validation.
- Shared understanding.
- Shared understanding.

- Creation of sets of terms, idioms and best practices.
- Communicate about model.
- Model verification.
- Distinct implementations. producing same results.
- Conceptual model grows by capturing confidence.
- Correction made if there is any difference found in implemented model and replication.
- Model validation.
- Correspondence between the outputs.
- Validation of a model on the basis of output closer to real-world.
- Reevaluating the original mapping.
- Researchers investing and getting interested in it through replication
- Describing the modeling process through developing a language.
- Culture of replication fosters.
- "mean" and "standard deviation" applying them to tests, overtime, replication of ABM experiment to understand "time step" and "shuffled lists".

Summary

- Here, we understood the benefits of replication in modeling.

Module – 160

Overview

- Here, we will learn about recommendations for model replicators.

Recommendations for model replicators

- RS is to produce the level of precision to establish hypothesis regularity.
- Examples : “numerical identity”, “distributional equivalence” and “relational alignment”.
- How detailed the description of the conceptual model is in the original paper or to communicate with authors or original model.
- Beneficial to delay the contact with original author until after first attempt to recreate the original model.
- Differences between public conceptual model and an implementation can be interesting and resulting in new discoveries.
- Becoming familiar with the toolkit in which original model was written.
- Better understanding of original model.
- Replicator understands the subtler working.
- Deliberately implementing a strategy.
- Obtain the source code of original model.
- Effective for illuminating discrepancies in the two model implementations.
- Groupthink
- Unconsciously adopting some of the practices of the original model developer.

Table 7.1

Details to be included in published replications. For each category, sample options to choose from are listed.

Categories of Replication Standards:

Numerical Identity, Distributional Equivalence, Relational Alignment

Focal Measures:

Identify particular measures used to meet goal

Level of Communication:

None, Brief email contact, Rich discussion and personal meetings

Familiarity with Language/Toolkit of Original Model:

None, Surface understanding, Have built other models in this language/toolkit

Examination of Source Code:

None, Referred to for particular questions, Studied in-depth

Exposure to Original Implemented Model:

None, Reran original experiments, Ran experiments other than original ones

Exploration of Parameter Space:

Examined results from original paper, Examined other areas of the parameter space

Module – 161

Overview

- Here, we will understand the recommendations for the model authors.

Recommendations for the model authors

- Research paper should be well specified.
- Complete source code for the model may not sufficient.
- Model authors must make their source code publicly available or at least the “pseudo code”.
- To what extent the model developer presents a sensitivity analysis.
- Small modification to the original model can effect results drastically.
- These sensitive differences should be published by the authors.
- Original author may not be the original implementer.
- Implementation will not be veridical.
- Model authors implement their own models using “ low-threshold ” and toolkits.
- Authors to examine their conceptual models through the lens of a potential model replicator.

Module – 162

Overview

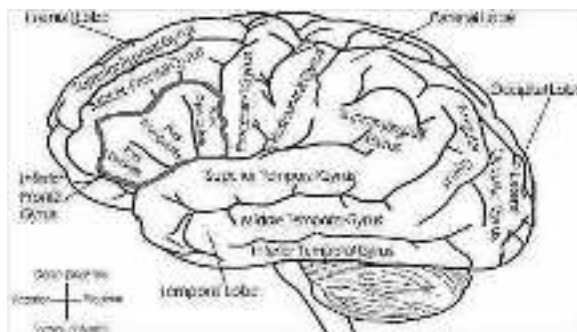
- Here, we will understand the basics about complex adaptive systems (CAS).

Complex Adaptive Systems

- Complexity
- Adaptation
- Systems



- The systems approach is an old concept.
- The approach stands on the assumption that breaking down of a complex concept into simple easy to understand units helps in better understanding of the complexity.
- Ludwig von Bertalanffy first proposed it as 'General System Theory'
- A system is a delineated part of the universe which is distinguished from the rest by an imaginary boundary. (NECSI)
- Properties of the system, the properties of the universe excluding the system which affect the system and the interactions / relationships.
- An adaptive system (or a complex adaptive system, CAS) is a system that changes its behavior in response to its environment.
- The adaptive change that occurs is often relevant to achieving a goal or objective. (NECSI)
- Complexity is:
 - ...the (minimal) length of a description of the system.
 - ...the (minimal) amount of time it takes to create the system.
 - Here the length of a description is measured in units of information.



Module – 163

Overview

- Here, we will understand about the historical perspective of CAS

History

- Started with the concepts of General Systems Theory
- Biologists focused primarily on individual cells
- Social Scientists focused on connections
- John Holland and cas/CAS

Summary

- We have learnt about the historical perspective of CAS.
- Credits: Niazi & Hussain book.

Module – 164

Overview

- Here, we will understand about the basic ideas of complexity

Complexity

- Disambiguation from Computational Complexity
- Nonlinearity
- Large number of variables
- Chaos theory
- The term "chaos" is popularly used to refer to disorder or confusion.
- In science, chaos is an important conceptual paradox that has a precise mathematical meaning:
- A chaotic system is a deterministic system that is difficult to predict.
- A deterministic system is defined as one whose state at one time completely determines its state for all future times.
- So what does it mean for a chaotic system to be difficult to predict?
- Butterfly effect

Summary

- We have learnt about the basics of Complexity in CAS.
- Credits: Niazi and Hussain book.

Module – 165

Overview

- Here, we will understand about adaptation in CAS.

Adaptation

- Adaptive changes in interactions
- Who changes
- What changes
- When changes
- How changes

Summary

- We have learnt about the adaptation in CAS.
- Credits: Niazi and Hussain book.

Module – 166

Overview

- Here, we will understand about the Systems approach.

Systems Approach

- Concept of a System - boundaries
- Many systems can be correlated
- Example: Life/Humans/Society
- Whole vs. parts
- Reductionist vs. Wholistic approach

Summary

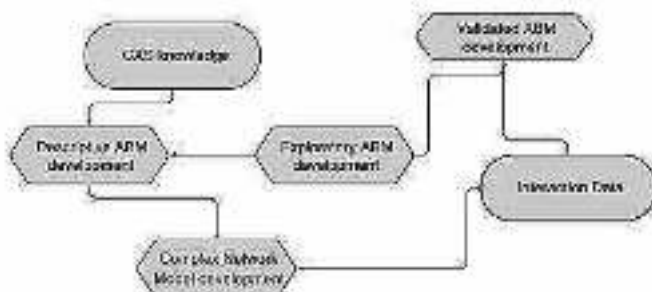
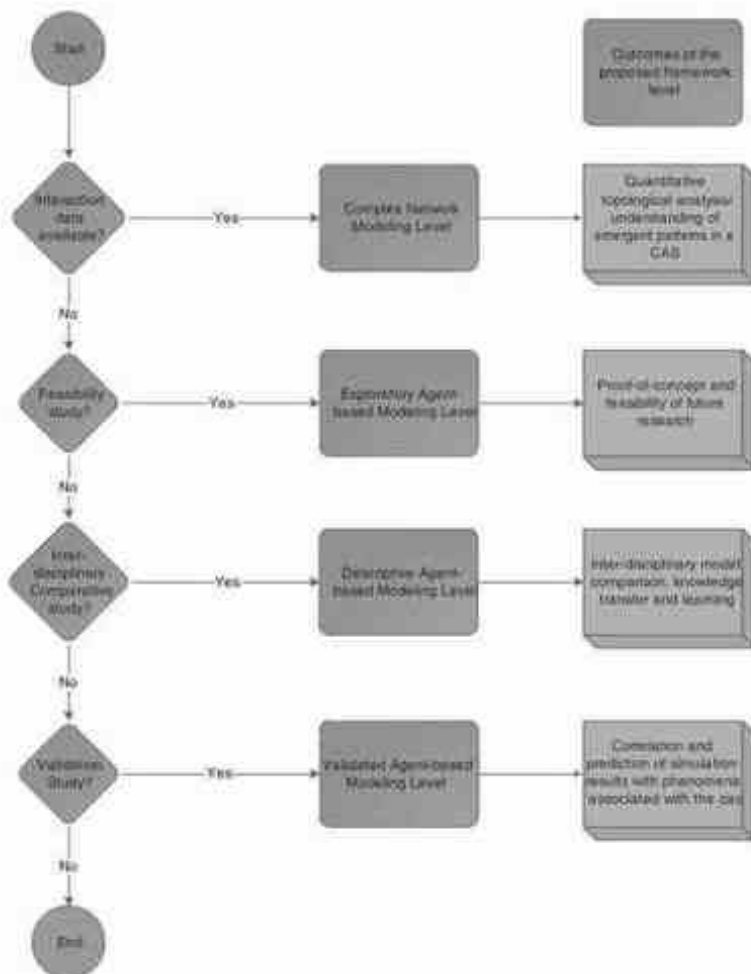
- We have learnt about the systems perspective of things.
- Credits: Niazi and Hussain book.

Overview

- Here, we will understand about the problems encountered in modeling of CAS.

Modeling of CAS

- Cognitive Agent-based (CABC) Framework



Module – 168

Overview

- Here, we will understand about the agent-based approach to modeling

Agent-based Approach

- Agent – something that acts
- There is a concept of agent in domains as distinct as:
 - AI/CS/Robotics
 - Sociology (Individual)
 - Biology
 - Ecology
 - Etc.
- CABC framework has 3 levels for Agent-based modeling
- Exploratory Agent-based modeling (EABM)
- Descriptive Agent-based Modeling (DREAM)
- Validated Agent-based Modeling (VOMAS-based Modeling or VABM)
- Exploratory ABM
- DREAM
- Virtual Overlay Multiagent System (VOMAS)
- Validated Agent-based Modeling

Summary

- We have learnt about various approaches for modeling using Agents and the CABC framework.
- Credits: Niazi and Hussain book.

Module – 169

Overview

- Here, we will understand about complex networks/graphs and modeling using the CABC Level 1

Complex Network-based Modeling

- What is a graph?
- Graph/Network
- Same entity different terms in different domains
- Network = weighted graph
- Different aspects of modeling using networks
- Labels

- arcs/lines
- Directed/Undirected
- Centrality-based
- Ego
- Sentiment graphs
- Community Detection
- Key benefits
- Very well-defined
- Lot of free tools
- Lot of community support
- Well-defined mathematical models
- Cons
- Difficult to figure out
- Difficulty in graph mining – lack of expertise in the community
- Missing data can be very difficult
- Requires complete data
- Complexity in deciding data boundaries

Summary

- We have learnt about complex-network modeling using the CABC framework
- Credits: Niazi and Hussain book.

Module – 170

Overview

- Here, we will understand how to put modeling and simulation to use employing the CABC framework.

Cognitive Agent-based Computing Framework

- Data and Models
- How much data is needed
- Why choose Agent-based Modeling?
- Why choose Complex network modeling?
- When to choose EABM?
- When to choose DREAM?
- When to choose VOMAS?

Summary

- We have learnt about putting the CABC framework to work.
- Credits: Niazi and Hussain book.

Overview

- Here, we will learn about the useful statistical models.

Useful Statistical Model

- The models that are useful in the case of limited data are:
- Queueing systems.
- Inventory and supply-chain systems.
- Reliability and maintainability.
- Limited data.
- Other distributions.

Queueing Systems

- Simulation solved the waiting line problems.
- Interval and service time patterns were given in queueing examples.
- In these examples service and arrival time was probabilistic.
- But it is possible to have constant arrival time and constant service time.
- “Arrivals” can occur due to many reasons like machine break downs.
- Exponential distribution used to simulate the random exponential distribution of services.
- For special cases truncated normal distribution can be utilized.

Inventory and supply-chain systems

- Three random variables
- Number of units demanded per order or per time period.
- Time between demands.
- Lead time.
- Simple inventory systems have constant demand and lead time constant or zero.
- Mathematical tractability based demand and lead time in inventory theory text could be invalid.
- Variety of demand patterns are satisfied by geometric, Poisson and negative binomial distributions.

Reliability and Maintainability

- Failure time is modeled with many distributions.
- When failure occurs randomly model distribution is considered as exponential.
- Modeling standby redundancy required for gamma distribution.

Limited Data

- Three distributions.

- Uniform distribution
- Triangular distribution
- Beta distribution
- Uniform distribution is just a special case of beta distribution

Summary

- Here, we discussed about useful statistical model, queuing systems, Inventory and supply-chain systems, Limited data, Reliability, and maintainability.
- Credits: Jerry Banks book.

Module – 172

Overview

- Here, we will learn about the Bernoulli distribution.

Bernoulli Distribution

- Bernoulli trail is one of the simplest experiment you can do in statistic and probability which has one out of two possible outcome.
- Bernoulli distribution is a discrete probability distribution for Bernoulli trail which has one out of two possible outcomes success or fail.
- Let us consider an example which consists of n trails and each of which can be a success or a failure.
- $X_j = 1$ if the jth experiment is success
- $X_j = 0$ if its failure.
- N trails and each trail has only two out comes success or failure.

$$p(X_1, X_2, \dots, X_n) = p_1(X_1) \cdot p_2(X_2) \cdot \dots \cdot p_n(X_n)$$

$$p_j(x_j) = p(x_j) = \begin{cases} p, & x_j = 1, j = 1, 2, \dots, n \\ 1 - p = q, & x_j = 0, j = 1, 2, \dots, n \\ 0, & \text{otherwise} \end{cases}$$

- Success here happens with probability (p) and failure at (p – 1).
- X is a discrete random variable. It takes value 1 in case of success and 0 in case of failure.

$$E(X_j) = 0 \cdot q + 1 \cdot p = p$$

- Below is the expected value of random variable X is calculated a
- Below is the variance of the random variable X is calculated as:

$$V(X_j) = \left[(0^2 \cdot q) + (1^2 \cdot p) \right] - p^2 = p(1 - p)$$

Module – 173

Overview

- Here, we will learn about the binomial distribution.

Binomial Distribution

- Binomial distribution is the discrete probability distribution with parameters n and p of the number of successes in the sequence of n independent experiments.
- Each n and p with its own boolean-value.
- Random variable.
- Bernoulli trial's random variable X has binomial distribution:

$$p(x) = \begin{cases} \binom{n}{x} p^x q^{n-x}, & x = 0, 1, 2, \dots, n \\ 0, & \text{otherwise} \end{cases}$$

- This equation is for calculating the probability of a specific outcome with all successes.

$$P(\overbrace{SSS \dots SS}^{x \text{ of these}} \overbrace{FF \dots FF}^{n-x \text{ of these}}) = p^x q^{n-x}$$

- S = success.
- In first X trials.
- $n-X = f$ (failures).

$$\binom{n}{x} = \frac{n!}{x!(n-x)!}$$

- $q = 1 - p$
- This equation shows the required number of successes and failures.

$$p(x) = \begin{cases} \binom{n}{x} p^x q^{n-x}, & x = 0, 1, 2, \dots, n \\ 0, & \text{otherwise} \end{cases}$$

- Again this equation can be used to calculate mean and variance of binomial distribution.
- X = sum of independent Bernoulli random variables. Then:

$$X = X_1 + X_2 + \dots + X_n$$

- Expected value $E(X)$

$$E(X) = p + p + \dots + p = np$$

- n = total number of experiments.
- p = probability of each experiment getting successful result.
- X = binomially random variable.

- Variance $V(X)$

$$V(X) = pq + pq + \dots + pq = npq$$

- n = total number of experiments.
- p = probability of each experiment getting successful result.
- X = binomially random variable.
- q = probability of failure in each experiment.

Module – 174

Overview

- Here, we will learn about the Poisson distribution.

Poisson Distribution

- A discrete frequency distribution that gives the probability of independent events occurring in a fixed interval of time is known as the Poisson distribution.

$$p(x) = \begin{cases} \frac{e^{-\alpha} \alpha^x}{x!}, & x = 0, 1, \dots \\ 0, & \text{otherwise} \end{cases}$$

- $X = 0, 1, 2, 3, \dots$
- $e = e = 2.71828$
- $\alpha = \alpha =$ mean number of successes in the given time interval or region of space.
- $\alpha > 0$

$$E(X) = \alpha = V(X)$$

- This is the important property of Poisson distribution that is mean and variance is equal to α .

$$F(x) = \sum_{i=0}^x \frac{e^{-\alpha} \alpha^i}{i!}$$

- This is the cumulative distribution function.

Module – 175

Overview

- Here, We will understand “Uniform Distribution” also known as “rectangular distribution”.

Uniform Distribution

- The uniform distribution is a continuous distribution.
- It takes values within a specified range.
- Probability density function for a uniform distribution always take a random number on $[0,1]$.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

- Consider a random variable X on the interval $[a, b]$ distributed uniformly. The, equation is given by:
- $X = a + b - aR$
- For the derivative of function.
- First, compute the cdf of variable X .
- Second, set $F_X = R$ on the range of X .
- Third, solve the equation F_X in terms of R for X .
- Step 1:**
 - The cdf for the equation is :

$$F(x) = \begin{cases} 0 & : x < 0 \\ x & : 0 \leq x < 1 \\ 1 & : x \geq 1. \end{cases}$$

- If X takes only discrete values 0 and 1 then:

$$F(x) = \begin{cases} 0 & : x < 0 \\ 1/2 & : 0 \leq x < 1 \\ 1 & : x \geq 1. \end{cases}$$

- Step 2:**
- Set

$$F(x) = (X - a)(X - b)R$$

- Step 3:**
- Solve for X in terms of R which proves.

$$X = a + b - aR = a + (b - a)R$$

Module – 176

Overview

- Here, we will understand About “Exponential Distribution”.

Exponential Distribution

- The Exponential Distribution is a continuous valued probability distribution.
- Takes positive real values and λ which controls the shape of the distribution.
- The probability density function

$$f(x; \lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

- This can also be defined as right-continuous Heaviside step function.

$$f(x; \lambda) = \lambda e^{-\lambda x} H(x)$$

- Here λ is called the rate parameter.
- The cumulative distribution function

$$F(x; \lambda) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

- This can also be defined as Heaviside step function

$$F(x; \lambda) = (1 - e^{-\lambda x}) H(x)$$

- Here λ is the number of occurrences per time unit.

The inverse transform technique is used when the form of cdf is so simple that its F^{-1} can be computed easily.

This technique for exponential distribution consists of four steps defined next.

- **Step 1:**
- Compute the cdf of random variable X given

$$F(x) = 1 - e^{-\lambda x}, x \geq 0.$$

- **Step 2**
- Set $F_X = R^F(x) = R$ on the range of X which becomes as

$$1 - e^{-\lambda x} = R \text{ on the range } x \geq 0.$$

- Step 3
- Solve the equation $F_X = F^F(X) = R$ for X for X in terms of R as follows:

$$1 - e^{-\lambda X} = R$$

$$e^{-\lambda X} = 1 - R$$

$$-\lambda X = \ln(1 - R)$$

$$X = -\frac{1}{\lambda} \ln(1 - R)$$

- Here, X is called random-variate generator.
- Step 4
- Generate uniform random numbers to compute desired random variates.

$$X_i = F^{-1}(R_i)$$

- For exponential case.

$$X_i = -\frac{1}{\lambda} \ln(1 - R_i)$$

- If $i=1, 2, 3, \dots$ $i = 1, 2, 3, \dots$ replace $1 - R_i$ with R_i as follows:

$$X_i = -\frac{1}{\lambda} \ln R_i$$

- Which proves that both R_i and $1 - R_i$ are uniformly distributed on $[0, 1]$.

Module – 177

Overview

- Here, we will discuss about “Triangular Distribution”.

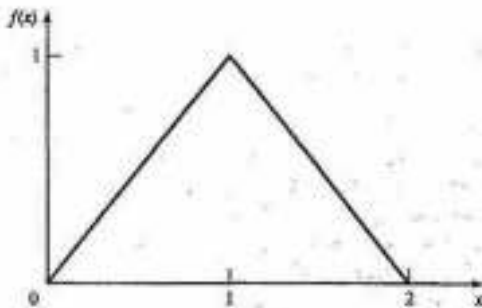
Triangular Distribution

- A triangular distribution is a continuous probability distribution with a probability density function shaped like a triangle defined by
- The minimum value a

- The maximum value b
- The peak value c
- Where $a < b$ and $a \leq c \leq b$
- Probability density function with variable X

$$f(x; \lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

- Where endpoints are $0, 2^{(0,2)}$ and mode at 1^1 .



- **Triangular Distribution**
- Its cumulative distribution function is given as :

$$F(x) = \begin{cases} 0, & x \leq 0 \\ \frac{x^2}{2}, & 0 < x \leq 1 \\ 1 - \frac{(2-x)^2}{2}, & 1 < x \leq 2 \\ 1, & x > 2 \end{cases}$$

- For $0 \leq X \leq 1$: $0 \leq X \leq 1$:

$$R = \frac{x^2}{2}$$

- Which implies that $x=2R$ $x = \sqrt{2R}$
- For $1 \leq X \leq 2$: $1 \leq X \leq 2$

$$R=1-2XZ \quad R = 1 - \frac{(2 - X^2)}{2}$$

- Which implies that $x=2-2(1-R)$ $x = 2 - \sqrt{2(1-R)}$
- Thus A^X is generated as:

$$X = \begin{cases} \sqrt{2R}, & 0 \leq R \leq \frac{1}{2} \\ 2 - \sqrt{2(1-R)}, & \frac{1}{2} < R \leq 1 \end{cases}$$

Module – 178

Overview

- Here, we will learn about characteristics of queueing systems.

CHARACTERISTICS OF QUEUEING SYSTEMS

- Customers and servers are the key elements of a queueing system.
- A Customer is anything that visits a facility and requires some service.
- Servers the service provider.
- Not every time the customer visits the service facility.

Table 6.1 Examples of Queueing Systems

System	Customers	Server(s)
Reception desk	People	Receptionist
Repair facility	Machines	Repairperson
Garage	Trucks	Mechanic
Tool crib	Mechanics	Tool-crib clerk
Hospital	Patients	Nurses
Warehouse	Pallets	Fork-lift Truck
Airport	Airplanes	Runway
Production line	Cases	Case-packer
Warehouse	Orders	Order-picker
Road network	Cars	Traffic light
Grocery	Shoppers	Checkout station
Laundry	Dirty linen	Washing machines/dryers
Job shop	Jobs	Machines/workers
Lumberyard	Trucks	Overhead crane
Sawmill	Logs	Saws
Computer	Jobs	CPU, disk, CDs
Telephone	Calls	Exchange
Ticket office	Football fans	Clerk
Mass transit	Riders	Buses, trains

Module – 179

Overview

- Here, we will learn about the calling population.

The Calling Population

- Calling population is the population of potential customers.
- Finite or infinite.
- Example of the bank having five machines and each machine is a customer and worker is its server.
- Infinite calling population when the system has a large population of potential.
- Examples are banks and restaurants.
- Arrival rate is the main difference between the finite and infinite population models.
- Infinite population system is not affected by the arrival rate.
- The Finite system depends on the no of customers waiting and being served.

Module – 180

Overview

- Here, we will learn about the system capacity.

System Capacity

- Every queueing system has some capacity.
- They have different number of customers entering the system.
- There is a limit for every queueing system to hold the number of customers.
- If the entering system finds the capacity full it will return to the calling population.
- Some systems have unlimited capacity
- Like tickets sold for a concert to students and many students can wait to purchase them.
- In limited systems, there is a distinction in arrival time and effective arrival time.

Module – 181

Overview

- Here, we will learn about the arrival process.

Arrival Process

- For infinite population models the arrival process characterized in terms of interarrival times of successive customers.
- Probability distribution is used to characterize the interarrival times when arrival is occurred in random time.
- Poisson arrival process is the most important for random arrivals.
- If A_n = inter-arrival
- Between $n-1$ (customer) and n (customer) then
- A_n is exponentially distributed with mean $1/\lambda$ time units.
- The arrival rate is λ customers per time unit.
- So the number of arrivals in a time interval of length t has Poisson distribution with mean λt customers.
- In restaurants and banks the Poisson arrival process has been employed successfully.
- Scheduled arrival is the second arrival class.
- Inter-arrival could be either constant or constant plus or minus a small random amount to represent early or late arrivals.
- Third situation occurs when one customer is always present in the queue.
- So we do not have an idle server.
- For finite population the arrival process is different.
- We define the customer as pending when he is out of the queue system and a member of potential calling population.
- **Arrival Process**

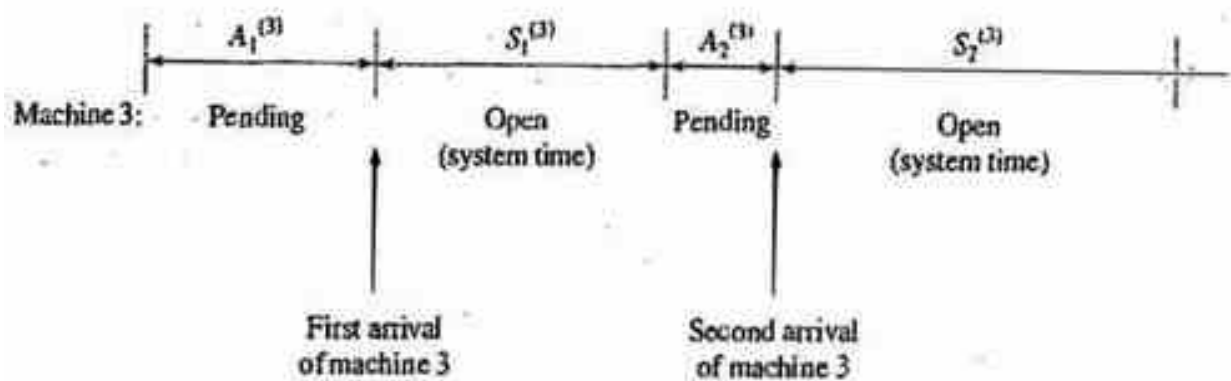


Figure 6.2 Arrival process for a finite-population model.

- Important application of finite population model is machine repair problem.

Module – 182

Overview

- Here, we will learn about the Queue behavior and Queue discipline.

Queue Behavior

- The actions of the customer while waiting in the queue system for his service is known as the queue behaviour.
- The customers can balk, renege or jockey.

Queue Discipline

- The logical ordering of the customers in a queue and determining that which customer should come for service when the server is free.
- FIFO, LIFO, SIRO, SPT and PR are some common queue disciplines.

Module – 183

Overview

- Here, we will learn about the Service Times and the Service Mechanism.

Service Times and the Service Mechanism

- Service times of successive arrivals are denoted by $S_1 S_2 \dots$
- Their duration can be random or constant.
- This latter case is characterized as a sequence of random or identically distributed variables.
- The exponential, Weibull, gamma, lognormal and truncated normal distributions are used for service times.
- Sometimes the services are identically distributed for all same customers. Same type customers can have different service time.
- Number of service centers and interconnecting queues are present in a queue system.
- Parallel working servers for each service centre.
- Server = c
- Single server ($c = 1$)

- Multiple servers ($1 < c < \infty$)
- Unlimited servers ($c = \infty$)

Service Times and the Service Mechanism

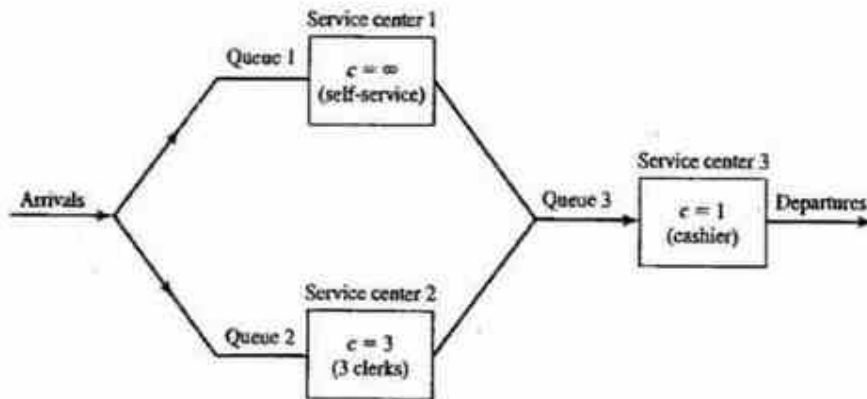


Figure 6.3 Discount warehouse with three service centers.

Service Times and the Service Mechanism

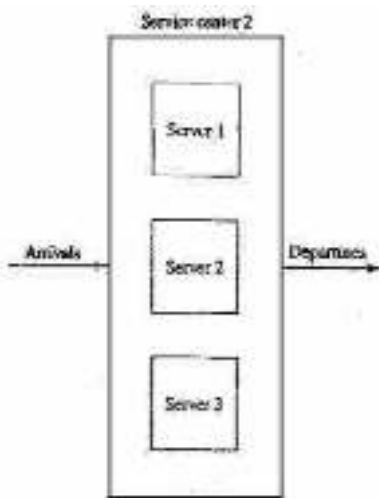


Figure 6.4 Service center 2, with $c = 3$ parallel servers.

Service Times and the Service Mechanism

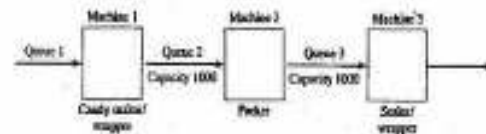


Figure 6.5 Candy-production line.

Module – 184

Overview

- Here, we will learn about the Queueing notation.

Queueing Notation

- Proposed by Kendall [1953]

- For parallel server systems.
- A/B/c/N/K based on this format.
- “A” represents the inter-arrival-time distribution.
- “B” represents the service-time distribution.
- “C” represents the number of parallel servers.
- “N” represents the system capacity.
- “K” represents the size of the calling population.
- Common symbols for A and B include M (exponential or Markov)
- D (constant or deterministic).
- Ek (Erlang of order k).
- PH (phase-type).
- H (hyperexponential).
- G (arbitrary or general)
- GI (general independent).

Queueing Notation

Table 6.2 Queueing Notation for Parallel Server Systems

P_n	Steady-state probability of having n customers in system
$P_n(t)$	Probability of n customers in system at time t
λ	Arrival rate
λ_e	Effective arrival rate
μ	Service rate of one server
ρ	Server utilization
A_n	Interarrival time between customers $n - 1$ and n
S_n	Service time of the n th arriving customer
W_n	Total time spent in system by the n th arriving customer
W_n^q	Total time spent in the waiting line by customer n
$L(t)$	The number of customers in system at time t
$L_q(t)$	The number of customers in queue at time t
L	Long-run time-average number of customers in system
L_q	Long-run time-average number of customers in queue
w	Long-run average time spent in system per customer
w_q	Long-run average time spent in queue per customer

Module – 185

Overview

- Here, we will learn about the long-run measures of performance of queueing systems.

LONG-RUN MEASURES OF PERFORMANCE

- The primary long-run measures of performance of queueing systems are:
- Long-run time-average number of customers in the system (L).
- In the queue (L_q).
- Long-run average time spent in the system (w).
- In the queue (w_q) per customer.

- Server utilization.
- A Proportion of time that a server is busy (p).
- Waiting line and mechanism is a system.
- System can also be sub-system of queue system.
- “queue” means waiting line alone.
- Other measurements include long-run proportion of customer who is delayed longer than t_0 time units.
- Customers turned away due to capacity constraints.
- Waiting line containing more than k_0 customers.
- G/G/c/N/K is defined.
- Their performance relationships and simulation run.
- The two types of estimators are:
 - An ordinary sample average.
 - A time-integrated sample average.

Module – 186

Overview

- Here, we are going to learn about properties of random numbers.

Properties of Random Numbers

- Uniformity and independence are two important statistical properties.
- Each sample must be drawn from continuous uniform distribution between 0 and 1.

$$f(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

- The density function and the expected value of $R_i^{(R)}$ is:

$$E(R) = \int_0^1 x dx = \frac{x^2}{2} \Big|_0^1 = \frac{1}{2}$$

- And the variance is:

$$V(R) = \int_0^1 x^2 dx - [E(R)]^2 = \frac{x^3}{3} \Big|_0^1 - \left(\frac{1}{2}\right)^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}$$

- Generated by a digital computer.
- Individual computation is expensive so computationally efficient method should be used.
- Routine should be portable for different computers and programming languages.
- Routine having a long cycle.
- Replicable random numbers.

- Random numbers should closely approximate the ideal statistical properties.
- Techniques to generate random is easy to develop.
- Techniques that produce sequences that are independent and uniformly distributed are difficult

Module – 187

Overview

- Here, we will learn about the linear congruential method.

Linear Congruential Method

- Widely used technique.
- Yields sequences with longer periods.
- Produces the sequence of integers.
- Between zero and $m - 1$.

$$X_{i+1} = (aX_i + c) \bmod m, \quad i = 0, 1, 2, \dots$$

- $x_0 = \text{seed}$,
- $a = \alpha$ = multiplier.
- $c = \beta$ = increment ($c \neq 0$) ($c \neq 0$).
- $m = \mu$ = modulus.
- This is called mixed congruential method.
- When $c = 0$ then known as multiplicative congruential.
- Interval $[0,1]$ is divided into n classes than expected observation for each interval is Nn/N .
- N is the total number of observations.
- Observing a value in a particular interval is independent.

Module – 188

Overview

- Here, we will learn about the generation of pseudo-random numbers.

Generation Of Pseudo-random Numbers

- False number generation.
- Removes the potential of true randomness.
- Set of random numbers can be replicated.
- Producing numbers between 0 and 1 that simulates the properties of uniform distribution and independence.
- Some errors or departures are:
- Not uniformly distributed random numbers.
- Not too high or too low variance.

- Generated numbers might not be continuous valued but are discrete valued.
- **Dependence:** Autocorrelation between numbers.
- Numbers successively higher or lower than adjacent numbers.
- Several numbers above the mean followed by the several numbers below the mean.
- Departures from uniformity and independence for a particular generation scheme often can be detected by tests.
- If detected then dropped in the favour of an acceptable generator.

Module – 189

Overview

- Here, we will understand about the Tests for Random numbers.

Tests for Random numbers

- Two categories:
- 1. Uniformity.

$$H_0: R_i \sim U[0, 1]$$

$$H_1: R_i \neq U[0, 1]$$

- H_0 denotes numbers distributed uniformly on the interval $[0, 1]$.
- 2. Independence.

$$H_0: R_i \sim \text{independently}$$

$$H_1: R_i \neq \text{independently}$$

- H_0 denotes the independent numbers.
- A level of significance α must be stated for each test.
- Where α is probability of rejecting the null hypothesis.

$$\alpha = P(\text{reject } H_0 | H_0 \text{ true})$$

- Software not specifically developed for simulation comes with random number generator pre-installed

Additional Tests:

- Good's serial test, the median-spectrum test, the run's test and a variance heterogeneity test.

Module – 190

Overview

- Here, we will understand about the “Random Variate Generation”.

Random Variate

- A random variate is a generated variable using uniformly distributed pseudorandom numbers.
- A random variate can be uniformly or nonuniformly distributed.
- Modeling activities is unpredictable or indeterminate.
- Random Variate Generations explain and illustrate some techniques for generating random
- Sometimes there is no availability of random-variate-generation libraries.
- It is up to the modeller to create an acceptable routine.
- This topic includes:
 - inverse-transform technique
 - Acceptance-rejection technique
- Special properties
- The composition method

Module – 191

Overview

- Here, we will understand about the “Inverse Transform Technique”.

Inverse Transform Technique

- The inverse-transform technique can be used to sample from the exponential.
- The uniform.
- The Weibull.
- The triangular and empirical distributions.
- The basic principle is to find the inverse function of F , F^{-1} such that $FF^{-1}=I$
- Where F^{-1} denotes the solution of the equation $r=F(x)$ in terms of y .

Inverse Transform Technique



Inverse Transform Technique

- This technique explains:
 - Exponential Distribution
 - Uniform Distribution
 - Weibull Distribution
 - Triangular Distribution
 - Empirical Continuous Distribution
 - Continuous Distribution
 - Discrete Distribution

Module – 192

Overview

- Here, We will understand "Acceptance Rejection Technique".

Acceptance Rejection Technique

- Acceptance Rejection Technique is a method used to generate random variate using some steps explained next.
- Step 1. Generate a Random Number R^R .
- Step2a. If $R \geq 1/4$ Accept R^R .
- Step2b. If $R \leq 1/4$ Reject R^R .
- If another uniform random variate on $[1/4, 1]$ is needed, repeat the procedure.
- The random variate generated using above methods is indeed uniformly distributed over $[1/4, 1]$.
- Take any $1/4 \leq a < b \leq 1$ then

$$P(a < R \leq b | 1/4 \leq R \leq 1) = \frac{P(a < R \leq b)}{P(1/4 \leq R \leq 1)} = \frac{b - a}{3/4}$$

- Proves that probability for uniform distribution is on $[1/4, 1]$.
- The efficiency of an acceptance-rejection technique rest on reducing the total of rejections.
- Which depends on computer being used.
- The skills of programmer.
- Rejected numbers.
- The Distributions which operates on acceptance rejection technique are:
 - Poisson Distribution
 - Non-stationary Poisson process
 - Gamma Distribution.

Module – 193

- **Overview**
- Here, we will discuss about “Other topics in Randomness”.

Special Properties

- Special Properties are variate generation techniques based on a particular feature of probability distributions.

Direct Transformation

- Direct transformation is used for the normal and lognormal distributions.
- The cdf is given by:

$$\Phi(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt, \quad -\infty < x < \infty$$

- Box and Muller made this method and is easy to program in scientific programming languages.

Convolution Method

- The probability distribution of a sum of two or more independent random variables is called convolution method.
- This method uses two random variables.
- Sum them and generate a new random variable.
- This technique can be applied on Erlang and Binomial variates.

More Special Properties

- The other relationship among distributions include relation between gamma and beta distribution.
- We can generate beta variates, using two gamma variates with acceptance-rejection technique.

Module – 194

Overview

- Here, we will discuss about “Data Collection”.

Data Collection

- The relation between construction of the model and the collection of needed input data is constant.
- The required data elements changes with complexity of the model.
- Time taking nature of data collection.
- Which type of data is to be collected is dictated by the objectives of study.

Module – 195

Overview

- Here, we will understand About “Identifying the Distribution with Data”.

Identifying the Distribution with Data

- Data distributions are graphical methods.
- These methods are used for selecting input distribution families.
- The specific distribution is often specified by estimating its parameters.

Module – 196

Overview

- Here, we will understand about the “Histograms”.

Histograms

- Histograms is a powerful technique for identifying how a distribution is shaped.
- Histogram is also known as frequency distribution.
- There are some methods for constructing a histogram.

1. Divide the data ranges into intervals of equal width.

2. Label the horizontal axis to follow the nominated intervals.

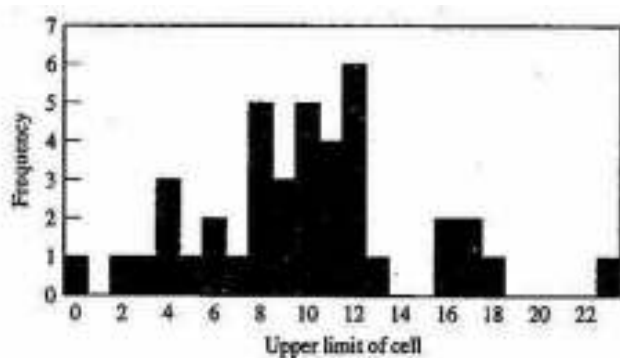
3. Find the frequency of occurrences inside each intervals.

4. Label the vertical axis to plot total occurrences for every interval.

5. Plot the frequencies on the vertical axis.

- The number of class intervals is decided by the number of observations.
- Amount of scatter in the data.

Histograms



Module – 197

Overview

- Here, we will understand about the “Selecting the Family of Distributions”.

Selecting the Family of Distributions

- The selection of a family of distributions is based on the context for what it is being investigated.
- The shape of the histogram.
- The Distributions which are frequently encountered and easy to analyze includes exponential, normal and poisson distribution.
- Whereas difficult to analyze includes the beta, gamma and Weibull distributions.
- Out of many, Here are some examples of distributions which were created on the physical basis.
 -
 - Binomial
 - Negative Binomial
 - Poisson
 - Normal
 - Lognormal
- An input Model is an estimate of reality.
- There is no “true” distribution for any stochastic input process.

Module – 198

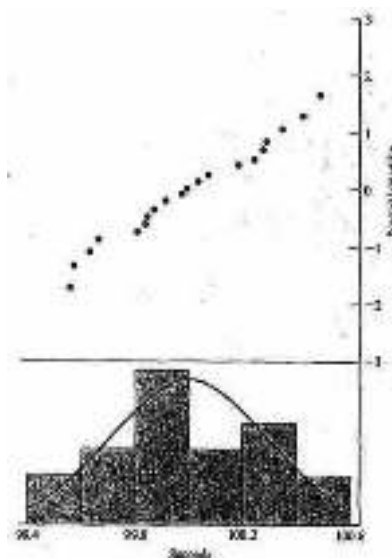
Overview

- Here, we will understand about the "Quantile-Quantile Plot".

Quantile-Quantile Plot

- A Quantile-Quantile is a graphical method for comparison of two probability distributions by plotting their quantiles against each other.
- If $F_Y = P(X \leq y) = q^F(y) \approx P(X \leq y) = q$ for $0 < q < 1$ and $0 < q < 1$.
- Here x is random variable with cdf F^F .
- Q -quantile of X^F is value of y .
- We write $y = F^{-1}(q)$ when F^F has an inverse.
- If $x_i, i = 1, 2, 3, \dots, n$ and $y_j, j = 1, 2, 3, \dots, n$
- Where $y_1 \leq y_2 \leq \dots \leq y_n$
- Let j^F denote the ranking of order
- $j^F = 1$ for smallest
- $j^F = n$ for largest
- Then

$$y_{j^F} \approx F^{-1}\left(\frac{j-1}{n}\right)$$



Module – 199

Overview

- Here, we will learn about the parameter estimation.

Parameter Estimation

- After selecting the family of distribution, the next step is to estimate the parameters of the distribution.
- Many useful distribution estimators are described here.
- Software packages integrated into simulation languages are available.

Module – 200

Overview

- Here, we will learn about the preliminary statistics: sample mean and sample variance.

Preliminary Statistics

- To estimate the parameters of a hypothesized distribution.
- When discrete and continuous raw data is available then the following equation is used:

$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n}$$

- $\bar{X}$ = sample mean.
- $X_1, X_2, \dots, X_n$ = observation in a sample of size n.
- The sample variance s^2 is defined as:

$$s^2 = \frac{\sum_{i=1}^n X_i^2 - n\bar{X}^2}{n-1}$$

- When discrete or continuous raw data are available following modified equations are used:

$$\bar{X} = \frac{\sum_{j=1}^k f_j X_j}{n}$$

- Where $\bar{X}$ is the sample mean.
- And the sample variance is:

$$s^2 = \frac{\sum_{j=1}^k f_j X_j^2 - n\bar{X}^2}{n-1}$$

- Here k is the number of distinct values of X.
- f_j is the observed frequency of the value X_j of X.
- When data is discrete or continuous and have been placed in class intervals then the sample mean and the sample variance are approximated from the following equation:

$$\bar{X} = \frac{\sum_{j=1}^k f_j m_j}{n}$$

- For the sample variance the following equation is used.

$$S^2 = \frac{\sum_{j=1}^c f_j m_j^2 - n \bar{X}^2}{n-1}$$

- f_j observed frequency of j^{th} class interval.
- m_j midpoint of j^{th} interval.
- c = number of intervals.

Module – 201

Overview

- Here, we will learn about the evolution of computer systems architectures

Evolution of Computer Systems Architectures

- ENIAC was the first computer system with no operating system and was large in size. They used vacuum tubes.
- Early computer doesn't have a good operating system, databases.
- They doesn't have any high-level programming language for simplifying program.

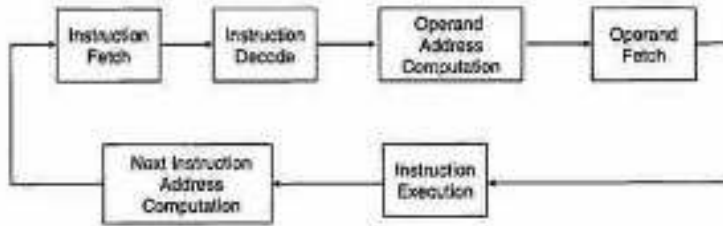
Systems Architecture

- Central Processing Unit.
- Graphics Processing Unit.
- Temporary memory.
- Permanent Memory.
- Input unit.
- Output unit.
- Control element.
- Disk Drives.

CPU Architectures

- Central processing unit.
- CU fetch functions.
- ALU performs functions.
- Registers to store the operands.
- L1, L2, L3 Cache.
- CPU Execution cycle.

Instruction architectures



Instructions and Memory Architectures

- Reduced instruction set computer (RISC).
- Highly optimized instructions.
- Complex instruction set computer (CISC).
- Memory address register.
- Memory data register.

Secondary Storage and Peripheral Device Architecture

- I/O devices control secondary storage.
- Magnetic tape drives.
- Magnetic disk drives.
- Compact optical disk drives.
- Disk jukeboxes.
- High density data storage.

Network Architecture

- Share information and processing capacity in real life.
- Another path to send or receive data.
- Central switching element. Central storage repository.
- Connected using intelligent interface units.

Module – 202

Overview

- Here, we will learn about the evolution of database systems.

Evolution of Database Systems

- Powerful tools to manage information.
- Early computers lag in online data storage.
- System architects have to rely on external storage.
- Initial storage was constructed using simple direct addressing.
- File system used coarse semantic means.

- Information Centralization.
- Security.
- Integrity.
- File systems added finer-grained definitions of stored information.
- Relational database system was introduced.
- Database must be entered at a specific entry point.
- Must traverse the path leading to the specific item.
- An issue for network is its legacy.
- Database system's demise began with development.
- Publication of Codd's relational database model.
- Information can be formed into tables called relations.
- Have to use calculus and algebra to operate.
- Concurrent access was not present in early network-based databases.
- The serializability theory and concurrency control led to further improvements in database technology.
- "ACID" properties.
- Object-oriented database was developed.
- For application developers to store data and data types efficiently.
- But did not possess some fundamental concepts.
- Object relational databases.
- Embraced by the U.S and international vendors.
- Next change will be in the area of transactions and transaction processing.
- ACID properties may be altered.

Module – 203

Overview

- **Here, we will learn about the evolution of operating systems.**

Evolution of Operating Systems

- Operating system is computer's software, firmware, and hardware.
- Interacts at low-level.
- Manage sharing of computer's resources.
- The most privileged of software elements on the system.
- Requires hardware.
- Early coders load the software in a location then make hardware to start processing.
- No automated means to switch between different jobs.
- Monitor or batch operating system provided the solution to transition between processing jobs.

Services of Operating System

- Process and Hardware management.
- Memory management.
- I/O management.
- Inter-process synchronization and communications.
- Resource allocation.
- Protection of system and user resources.

Evolution of Operating Systems

- interrupt service routine checks everything and determines what to perform next.
- Efficient use of computing facilities.
- Convenient interface to user by hiding the details of machine.
- Transparent use of computer resources.
- Improved architectures and instruction execution schemes.
- Improve the performance of the processor.
- Not to depend on a single processor
- Add entire new processors.
- Distributed processors.
- Processors operating system is the single global operating system.
- In one case the entire operating system can be replicated on each site.
- Each processor has a specific function.
- New concepts are still being examined.
- Client/server processing.
- Multiprocessing operating system forum.

Module – 204

Overview

- Here, we will learn about the evolution of computer networks

Evolution of Computer Networks

- The term network can mean many things.
- In computer it refers to combination of interconnected resources.
- ARPANET was an early computer network.
- A loose coalition of devices that share information.
- All this evolved into internet.
- ARPANET provided early research into communication protocols.
- Greater availability and access to a wider range of devices.
- As more applications were added.
- Information access and manipulation took on greater emphasis.
- As the networks improve new applications arose.
- Remote job execution was added with information exchange.
- But still real-time was not provided.

- They had to develop more advance protocols and network operating systems.
- It was time consuming.
- Local area network was introduced in 1970s.
- LAN became more widely available.
- LAN designs have been produced to fit an extremely wide spectrum of user requirements.
- LAN is widely used in all aspects of society.
- LAN had rapid growth in 1990s and early 2000s.
- LAN can connect a device from anywhere.
- LAN is used to link factory robots.
- LAN provide sensor data, control data and feedback.
- Walt Disney World used LAN for different services.
- Large banks also use LAN for interconnection of their local sites.
- Wireless technology has improved.
- Began in 1970s with Aloha net as foundation.
- Wireless phone network development has opened the door for computer networks.
- More development will be made to provide more applications.

Module – 205

Overview

- Here, we will learn about the need for performance evaluation

Need for Performance Evaluation

- Selecting a specific computer architecture for an application.
- An operating system.
- A database system.
- A wide area or local area network system.
- If not used properly then decrease productivity.
- Operating system can increase the concurrency access.
- Or can block access.
- A database can provide efficiently sharing information among many applications.
- Or blocking of information by dropping data availably.
- LAN provides means for streamline information processing and eliminate redundancies.
- But it may also deter users from logging on because of link or protocol problems.
- We need to select the computer system mapped to the needs.
- A computer system can be very simple with simple processor etc.
- A computer can be highly elaborate with multiple processors etc.
- A purchaser must decide according to motherboard its using.
- While purchasing the system all this must be kept in mind.

Module – 206

Overview

- Here, we will learn about role of performance evaluation in computer engineering.

Role of Performance Evaluation in Computer Engineering

- Client/server architectures.
- These systems are from Government, industry and academia.
- There is constant demand for improved reliability and availability.
- Present systems provide more features.
- Sharing of resources on global scale.
- Ability to bring computing power to the user.
- The optimal design or selection of computer system is therefore very important.

Module – 207

Overview

- Here, we will learn about the overview of performance evaluation methods.

Overview of Performance Evaluation Methods

- Models provide a tool to define a system.
- Define its problems in a concise fashion.
- This provide an environment to study a system before its existence.
- Developed on the base of theoretical laws and principles.
- Modeling is better if:
- Physical laws are available.
- Pictorial representation is possible.
- Input/output elements are manageable.
- But we don't have clear physical laws.
- Faithful model of a system must be constructed.
- Model a slice of the real-world system.
- Which element is important to understand in faithful system.
- We can say its intuition.

Module – 208

Overview

- Here, we will learn about performance metrics and evaluation criteria.

Performance Metrics and Evaluation Criteria

- Performance metrics and evaluation criteria are required for systems.
- User must follow a methodology.
- The motivations.
- The environment.
- Technological boundaries.
- Must understand the usage of the system.
- Needs and uses must be defined.
- Must compile a wish list of all potential uses.
- User must know processing requirements.
- Communication transfer.
- Management requirements.
- Define or know why we need something at first priority.
- Checking in which environment the system fits.
- Checking in which technological aspect it fits.
- Connecting to diverse set of company assets.
- Link planned new resources.
- Need large volume of data for system requirements.
- How this data assist in the selection?
- Collection of data can be divided in quantitative and qualitative.
- Specific performance measures can be derived from one set of data.
- Subjective measures can be derived from another.

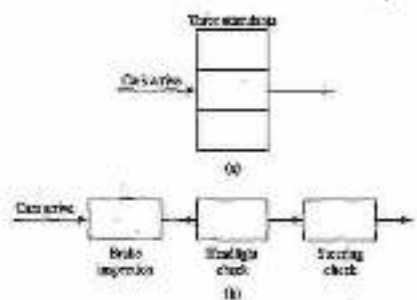
Module – 209

Overview

- Here, we will understand about the “Comparison of Two System Designs”.

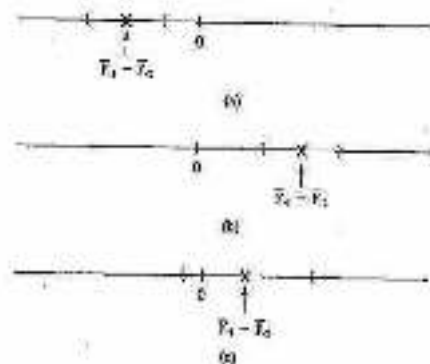
Comparison of Two System Designs

- A simulation analyst can compare two possible configurations.
- Using supply chain inventory system there ordering policies will be compared.
- Replications is used to analyze the output data.
- The mean performance measure for system ‘I’ will be denoted by $\theta_i, i=1,2$.
- The goal of the simulation experiments is to obtain point.
- Interval estimates of the difference in mean performance.
- Namely $\theta_1 - \theta_2$.
- Inverse Transform Technique
- Replications is used to analyze the output data.
- The mean performance measure for system ‘I’ will be denoted by $\theta_i, i=1,2$.
- Vehicle Safety Inspection:



- In comparing two systems, number of replications $R_i^{(R)}$ to be made and run length $TE_i^{(R)}$ for each model $i \in I_2$ ($i = 1, 2$) is decided.

System	Replication				Sample Mean	Sample Variance
	1	2	...	R_i		
1	Y_{11}	Y_{12}	...	Y_{1R_1}	$\bar{Y}_1$	S_1^2
2	Y_{21}	Y_{22}	...	Y_{2R_2}	$\bar{Y}_2$	S_2^2



Module – 210

Overview

- Here, We will understand "Independent Sampling with Equal Variances".

Independent Sampling with Equal Variances

- Independent Sampling refers to different and independent random numbers stream to simulate the two systems.
- Acceptance-Rejection**
- The observations of simulated system 1, $Y_1, Y_2, \dots, Y_{R_1}$ are all statistically independent of simulated system 2, $Y_1, Y_2, \dots, Y_{R_2}$.
- Y_i is given by:

$$V(\bar{Y}_i) = \frac{V(Y_i)}{R_i} = \frac{\sigma_i^2}{R_i}, \quad i = 1, 2$$

- When Y_1 and Y_2 are statistically different as:

$$V(\bar{Y}_1 - \bar{Y}_2) = V(\bar{Y}_1) + V(\bar{Y}_2) \\ = \frac{\sigma_1^2}{R_1} + \frac{\sigma_2^2}{R_2}$$

- It implies two variances are equal (but unknown in value) $\sigma_1^2 = \sigma_2^2 = \sigma^2$.
- If two-sample-t confidence interval approach is used. The point estimate of the mean difference is: $\bar{Y}_1 - \bar{Y}_2$
- Where $\bar{Y}_i$ is given as:

$$S_i^2 = \frac{1}{R_i - 1} \sum_{j=1}^{R_i} (Y_{ij} - \bar{Y}_i)^2 \\ = \frac{1}{R_i - 1} \left(\sum_{j=1}^{R_i} Y_{ij}^2 - R_i \bar{Y}_i^2 \right)$$

- Here, S_i^2 is an estimator of variance σ^2 . Assuming $\sigma_1^2 = \sigma_2^2 = \sigma^2$ a pool estimate of σ^2 is obtained as:

$$S_p^2 = \frac{(R_1 - 1)S_1^2 + (R_2 - 1)S_2^2}{R_1 + R_2 - 2}$$

- Where $v = R_1 + R_2 - 2$
- The c.i. $\theta_1 - \theta_2 = \mu_1 - \mu_2$ for with the standard error is computed by:

$$s.e.(\bar{Y}_1 - \bar{Y}_2) = S_p \sqrt{\frac{1}{R_1} + \frac{1}{R_2}}$$

- The standard error is an estimate of the standard deviation of the point given as: $\sigma \sqrt{1/R_1 + 1/R_2}$.

Module – 211

Overview

- Here, we will discuss about "Independent Sampling with Unequal Variances".

Independent Sampling with Unequal Variances

- If the assumption of equal variances cannot safely be made, an approximate $100(1 - \alpha)\%$ for $\theta_1 - \theta_2 = \mu_1 - \mu_2$ can be computed as:

$$s.e.(\bar{Y}_1 - \bar{Y}_2) = \sqrt{\frac{S_1^2}{R_1} + \frac{S_2^2}{R_2}}$$

- Degrees of freedom, v can be as:

$$v = \frac{(S_1^2 / R_1 + S_2^2 / R_2)^2}{[(S_1^2 / R_1)^2 / (R_1 - 1)] + [(S_2^2 / R_2)^2 / (R_2 - 1)]}$$

- A minimum number of replications $R1 \geq 6^{R1} \geq 6$ and $R2 \geq 6^{R2} \geq 6$ is recommended for this procedure.

Module – 212

Overview

- Here, we will understand About “Common Random Numbers”.

Common Random Numbers

- CRN means, for each replication, the same random numbers are used to simulate both systems.
- Therefore, R1 and R2 must be equal, say $R1 = R2 = R$.
- The purpose of CRN is to induce positive correlation between $y1$ and $y2$.
- A variance reduction in point estimator of mean difference is given by:

$$\begin{aligned} V(\bar{Y}_1 - \bar{Y}_2) &= V(\bar{Y}_1) + V(\bar{Y}_2) - 2\text{cov}(\bar{Y}_1, \bar{Y}_2) \\ &= \frac{\sigma_1^2}{R} + \frac{\sigma_2^2}{R} - \frac{2\rho_{12}\sigma_1\sigma_2}{R} \end{aligned}$$

- Compare variance of $\bar{Y}_1 - \bar{Y}_2$, call it V_{cm} given as :

$$V_{\text{CRN}} = V_{\text{IND}} - \frac{2\rho_{12}\sigma_1\sigma_2}{R}$$

- Which implies:

$$V_{\text{CRN}} < V_{\text{IND}}$$

- To compute a $100(1 - \alpha)\%$ c.i compute the differences:

$$D_r = Y_{r1} - Y_{r2}$$

- Compute Sample mean difference:

$$\bar{D} = \frac{1}{R} \sum_{r=1}^R D_r$$

- The sample variance of the difference $\{D_r\}$ is computed as:

$$S_D^2 = \frac{1}{R-1} \sum_{r=1}^R (D_r - \bar{D})^2$$

$$= \frac{1}{R-1} \left(\sum_{r=1}^R D_r^2 - R\bar{D}^2 \right)$$

- Where standard error of $Y_1 - Y_2$ $\bar{Y}_1 - \bar{Y}_2$ is estimated by:

$$s.e.(\bar{D}) = s.e.(\bar{Y}_1 - \bar{Y}_2) = \frac{S_D}{\sqrt{R}}$$

- Implementation of common random number:

1. Dedicate a random-number stream to a specific purpose.

1. Dedicate a random-number stream to a specific purpose.
2. For system with external arrivals, dedicate one random stream to one of arrival and their attributes.
3. For systems having an entity performing given activities in a cyclic fashion, assign a random-number stream to this entity.
4. If synchronization is not possible use independent streams of random numbers for this subset.

Module – 213

Overview

- Here, we will understand about the “Confidence Intervals with Specified Precision”.

Confidence Intervals with Specified Precision

- Confidence intervals for the difference between two systems’ performance can be achieved in an parallel way.
- If the goal is to obtain error in our estimate $\theta_1 - \theta_2$ $\bar{\theta}_1 - \bar{\theta}_2$ to be less than $\pm \epsilon$ then number of replications R should be:

$$H = t_{\alpha/2, \nu} s.e.(\bar{Y}_1 - \bar{Y}_2) \leq \epsilon$$

- Using $RO \geq R_0 \geq 2$ we obtain an initial estimate.
- We solve total number of replication $R \geq RO^R \geq R_0$ to achieve half-length criterion.
- We make replications $R \geq RO^R \geq R_0$ for each system to compute confidence-interval and half-length criterion.

Overview

- Here, we will understand about the “COMPARISON OF SEVERAL SYSTEM DESIGNS”.

COMPARISON OF SEVERAL SYSTEM DESIGNS

- To compare “K” alternative system designs.
- Many different statistical procedures have been developed.
- These are used to analyze simulation data and to draw statistically sound inferences about parameters θ_i .
- These procedures are classified as:
 - Fixed-sample size.
 - Sequential sampling.
- **Fixed Sample Size**

A predetermined sample size is used to draw inferences via hypothesis tests or confidence intervals.

Example:

- - The interval estimation of a mean performance measure.

Advantages:

- - An easily estimated or known cost in terms of computer time before running the experiments.
 - Can be used when :
 - Computer time is limited.
 - A pilot study is being conducted.

Disadvantages

- - A strong Conclusion could be impossible.
 - A hypothesis test may lead to a failure to reject the null hypothesis.
- **Sequential Sampling**
 - In sequential sampling more and more data are collected until an estimator with a pre-specified precision is achieved.
 - In two stage procedure, an initial sample estimates additional observations needed to draw conclusions with a specified precision.

- Goal of simulation analyst
- Estimation of each parameter, θ_i .
- Comparison of each performance measure.
- All pair wise comparisons.
- selection of the best θ_i .

Module – 215

Overview

- Here, we will understand about the “Bonferroni Approach to Multiple Comparisons”.

Bonferroni Approach to Multiple Comparisons

- If C confidence intervals are computed and that the i th interval has confidence coefficient $1 - \alpha_i$.
- The Bonferroni inequality states that:

$$P(\text{all statements } S_i \text{ are true, } i = 1, \dots, C) \geq 1 - \sum_{j=1}^C \alpha_j = 1 - \alpha_E$$

- Where

$$\alpha_E = \sum_{j=1}^C \alpha_j \text{ is called overall error probability.}$$

- This is equivalent to:

$P(\text{one or more of the } C \text{ confidence intervals does not contain the parameter being estimated}) \leq \alpha_E$

- Here, α_E provides the upper bound.

- A major advantage of Bonferroni approach is that runs with independent sampling and or common random numbers.
- The major disadvantage of the Bonferroni approach in making a large number of comparisons is the increased width of each individual interval.
- The width of a c.i. is a measure of the precision of the estimate.
- 3 possible ways of using the Bonferroni inequality when comparing K alternative system designs:

1. Construct a $100(1 - \alpha_i)\%$ c.i. for parameter θ_i , where number of intervals is $C = K$.

2. Compare all designs to one specific design that is:

Construct a $100(1 - \alpha_i)\%$ c.i. for $\theta_i - \theta_1$.

Here, the number of intervals is $C = K - 1$.

3. Compare all designs to each other. That is:

• Construct a $100(1 - \alpha_i)\%$ c.i. for $\theta_i - \theta_j$.

• With K designs, the number of intervals is computed as $C = K(K - 1) / 2$.

Module – 216

Overview

- Here, We will understand “Bonferroni Approach to Selecting the Best”.

Bonferroni Approach to Selecting the Best

- Gross Level
- If there are K system designs, and the i^{th} design has expected performance θ_i .
- Then, we're interested in best system.
- Where “best” means maximum expected performance.
- Refined
- At a refined level, we are also interesting in knowing how much better the best is relative to alternatives.
- It helps us in choosing a inferior system not deficient by much.
- **Difference**
- If system design i^* is the best, then $\theta_{i^*} - \max_{j \neq i^*} \theta_j$ is equal to the difference in performance between the best and the second best.
- If system design i^* is not the best then, $\theta_{i^*} - \max_{j \neq i^*} \theta_j$ is equal to the difference between system i^* and the best.
- i^* denote index of the best system.
- Smaller the difference $\theta_{i^*} - \max_{j \neq i^*} \theta_j$ is, the more certain we find the best system.
- If the difference between system i^* and the other is significant, then we try to achieve i^* with high probability.
- The best system probability should be at least: $1 - \alpha$ whenever $\theta_{i^*} - \max_{j \neq i^*} \theta_j \geq \epsilon$
- We select best of one of near best if multiple system are within ϵ of the best.
- **Achievement**
- This procedure achieves the desired probability and also forms $100(1-\alpha)\%$ confidence intervals for $\theta_{i^*} - \max_{j \neq i^*} \theta_j$.

Module – 217

Overview

- Here, we will discuss about “Bonferroni Approach to Screening”.

Bonferroni Approach to Screening

- When there exist multiple systems and two stage procedure is not possible.
- Then, divide set of systems into the best.
- This is achieved by using screening or subset selection procedure.
- Independent sampling with unequal variances
- This promises that stayed subset contains best system.
- With probability $\geq 1 - \alpha$.
- When the data is distributed normally, by independent sampling or CRN is used.
- This subset may contain all K systems, only one or some.
- It depends on the number of replications and sample variances.
- Procedure
- Specify correct probability selection and sample size.
- Make R replications of the system.
- Calculate the sample and sample variance of the difference.
- If bigger is better then retain selected subset system.

- If smaller is better then retain selected subset system.
- All design not retained can be eliminated from further considerations.

Module – 218

Overview

- Here, we will understand About “Metamodeling”.

Metamodeling

- Metamodeling is the process of generating metamodels.
- Where metamodel refers to model of a model.
- There is a simulation output variable Y.
- It is related to k independent variables.
- Dependent variable Y is random variable.
- Independent variable is design variable.
- Relation between independent and dependent variable is represented by simulation model.
- Metamodel simplifies this relationship by simpler mathematical function.
- This relationship is represented by as function $Y=f(x)$ $Y = f(x)$.
- Most of the time relationship is unknown.
- Function is created using unknown parameter.
- Regression analysis estimates this parameter.

Module – 219

Overview

- Here, we will understand about the “Simple Linear Regression”.

Simple Linear Regression

- The relationship is between independent variable x and a dependent variable y.
- Linear relationship.
- Value of y for given x is:

$$E(Y | x) = \beta_0 + \beta_1 x$$

- β_0 intercept on y axis.
- β_1 is the slope.
- Y can be described as:

$$Y = \beta_0 + \beta_1 x + \epsilon$$

- ϵ^i is random error with mean zero.
- This model is commonly called a simple linear regression model.
- n^{th} pair of observations: $y1, x1, y2, x2, \dots, yn, xn$ (y_1, x_1), (y_2, x_2) ... (y_n, x_n)
- β_0 and β_1 estimates the observation.
- Sum of squares of the deviation and regression line is minimized.
- The individual observations:

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i, i = 1, 2, \dots, n$$

- $x1, x2, \dots, x_n, \epsilon_1, \epsilon_2, \dots$ are uncorrelated random variables.
- ϵ^i is given by:

$$\epsilon_i = Y_i - \beta_0 - \beta_1 x_i$$

- It shows how ϵ^i is related to x_i, Y_i and EY_i .
- The sum of squares of the deviation is :

$$L = \sum_{i=1}^n \epsilon_i^2 = \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 x_i)^2$$

- Where L is called least-squares function.
- Rewriting Y_i as:

$$Y_i = \beta'_0 + \beta_1(x_i - \bar{x}) + \epsilon_i$$

- Where:

$$\beta'_0 = \beta_0 + \beta_1 \bar{x} \text{ and } \bar{x} = \sum_{i=1}^n x_i / n.$$

- It is often called as Transformed linear regression model.

$$L = \sum_{i=1}^n [Y_i - \beta'_0 - \beta_1(x_i - \bar{x})]^2$$

- Taking partial derivatives and setting each to zero.

$$n\hat{\beta}'_0 = \sum_{i=1}^n Y_i$$

$$\hat{\beta}_1 \sum_{i=1}^n (x_i - \bar{x})^2 = \sum_{i=1}^n Y_i(x_i - \bar{x})$$

- These are called normal equations.

Overview

- Here, we will understand about the "Testing for Significance of Regression".

Testing for Significance of Regression

- Testing for the significance of regression provides another means for assessing the adequacy of the model.
- Error component ϵ_i is normally distributed.
- Testing for Significance of Regression
- The adequacy of the model is checked by residual analysis.
- Developed by variance properties of β_0 and β_1 .
- **Alternative hypothesis:**

$$H_0 : \beta_1 = 0$$

$$H_1 : \beta_1 \neq 0$$

- H_0 indicates no relationship between x and Y.
- x is of little value in explaining the variability in Y.

- Illustration Best Estimator:

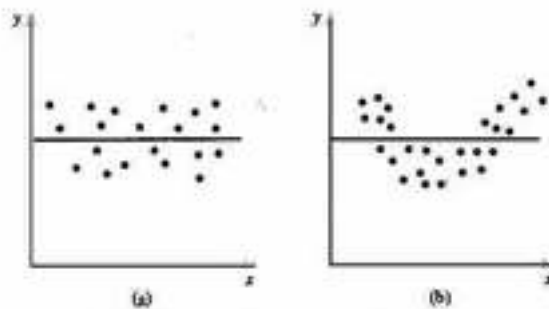


Figure 12.5 Failure to reject $H_0 : \beta_1 = 0$.

- Illustration Non-linear relationship:

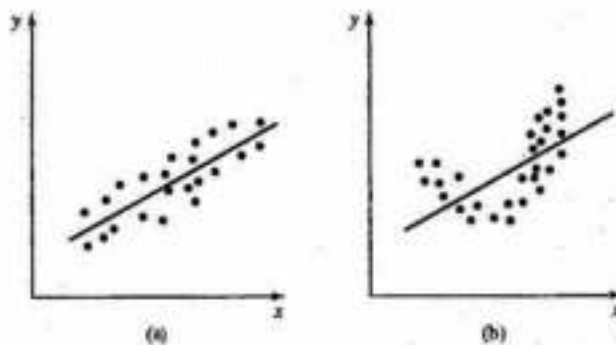


Figure 12.6 $H_0 : \beta_1 = 0$ is rejected.

- Statistic for significance of regression:

$$t_0 = \frac{\hat{\beta}_1}{\sqrt{MS_e / S_{xx}}}$$

- MS_e, MS_e is mean squared error given by:

$$MS_e = \sum_{i=1}^n \frac{e_i^2}{n-2}$$

- Testing for Significance of Regression
- The direct method calculates $\sum_{i=1}^n e_i^2$

$$\sum_{i=1}^n e_i^2 = S_{yy} - \hat{\beta}_1 S_{xy}$$

- Where $S_{yy}^{Corrected}$ is corrected sum of squares Y given by:

$$S_{yy} = \sum_{i=1}^n y_i^2 - \frac{\left(\sum_{i=1}^n y_i\right)^2}{n}$$

Module – 221

Overview

- Here, we will understand about the “Optimization via Simulation”.

Optimization via Simulation

- Optimization is a key tool.
- Well developed algorithms for classes of problems.
- Linear Programming is Most famous.
- Much of optimization deals with performance of design.
- In discrete event simulation, result is a random variable.
- Let $x_1, x_2 \dots x_n$ be m controllable design variables.
- Let $Y(x_1, x_2 \dots x_n)$ be observed simulation output performance.
- $x_1, x_2 \dots x_n$ might denote number of Automated Guided Vehicles.
- $Y(x_1, x_2 \dots x_n)$ could be total Material Handling System acquisition and operation cost.
- $Y(x_1, x_2 \dots x_n)$ with respect to $x_1, x_2 \dots x_n$.
- Y is random variable, so cannot optimize actual value of Y .
- Definition of optimization.

maximize or minimize $E(Y(x_1, x_2, \dots, x_m))$

- $E(Y(x_1, x_2, \dots, x_m))$ is long-run average cost of operating the MHS.
- $x_1^{x_1}$ AVGs, $x_2^{x_2}$ load per average, and routing algorithm $x_3^{x_3}$.
- Formulation of performance measure.

$$Y'(x_1, x_2, x_3) = \begin{cases} 1, & \text{if } Y(x_1, x_2, x_3) \leq D \\ 0, & \text{otherwise} \end{cases}$$

1. Combine all of the performance measures into a single measure.
 - The most common being cost.
 2. Optimize with respect to one key performance measure.
 - Evaluate with secondary performance measures.
 3. Optimize with respect to one key performance measure.
 - Consider those meeting certain constraints on performance.
- Expected cost and cycle time.

Module – 222

Overview

- Here, We will understand “Why is Optimization via Simulation Difficult?”

Difficulties in Optimization via Simulation

- If design variables are numerous optimization can be difficult.
- A diverse collection of design variable types is known as the structure of performance function
- The performance can not be evaluated, but can be estimated.
- No conclusion with estimates.
- It can frustrate optimization algorithms that try to move in improving directions.
- It can be eliminated using many replications.
- Or using long runs.
- At each design point that the performance estimate has essentially no variance.
- It causes other designs to be explored.
- Standard sampling variability force optimizations are as following.
- Guarantee a prespecified probability of correct selection.
- Guarantee asymptotic convergence.
- Optimal for deterministic counterpart.

- Robust heuristics.
- Robust heuristics are the most common algorithms.
- It is found in commercial optimization via simulation software.

Module – 223

Overview

- Here, we will discuss about “Using Robust Heuristics”.

Using Robust Heuristics

- It means a procedure that does not depend on strong problem structure.
- Can be applied on decision variables and sampling variability.
- Generic algorithms and tabu search are examples.
- There are k possible solutions to the optimization via simulation problem.
- On each iteration a GA operates on a "population" of p solutions.
- Basic Genetic Algo

$$p(0) = \{x_1(0) \dots x_p(0)\}$$

- Set the iteration counter $j = 0$
- Run simulation experiments.
- Select population p from $P(j)$.
- Recombine Solutions via $P(j + 1)$.
- Set $j = j + 1$ and repeat.
- The G.A can be terminated after some number of iterations.
- At termination X^* with smallest $Y(x)$ is chosen as best.
- **Basic Tabu Search**
- Set the iteration counter $j = 0$. select initial x^* in X
- Find solution x' which minimizes $Y(x)$.
- If $Y(x') < Y(x^*)$ then $x^* = x'$.
- Update the list of Tabu and repeat from step 2.
- Tabu search can be terminated when some number of iterations has passed without changing x^* .
- T.S is fundamentally a discrete-decision-variable optimizer.

Module – 224

Overview

- Here, we will understand About “An Illustration: Random Search”.

Random Search: An Illustration

- Random search provides guaranteed asymptotic convergence in certain conditions.
- Convergence can be slow and memory consumptive.
- Random search is not a Robust Heuristic.
- Random search algorithm requires finite number of possible designs.
- Rule out problem with continuous decision variables.
- Algorithms designed for continuous variable problem is there.
- On each iteration, we compare a current solution to a randomly chosen competitor.
- If competitor is better it becomes current good solution.

- **Step 1.**
 - Initialize random variables $C_i = C(i) = 0$.
 - Initial solution $f^{(0)}$ and set $C(f^{(0)}) = 1$.
 - $C(f^{(i)})$ count number of times $f^{(i)}$ is visited.
- **Step 2.**
 - Chose another solution except $f^{(i)}$.
 - Such as each solution has equal chance of being selected.
- **Step 3.**
 - Run two simultaneous experiments.
 - Use $f^{(0)}$ and $f^{(i)}$ to obtain $Y(f^{(0)})^{(i^0)}$ and $Y(f^{(i)})^{(i)}$
 - Set $f^{(i^0)} = f^{(i)}$
- **Step 4.**
 - Set $C(f^{(i^0)}) = C(f^{(i)}) + 1$
 - If not done go to step 2.
 - If done, Then select $x^{(i^*)}$ such that $C(f^{(i^*)})$ is largest count.

Module – 225

Overview

- Here, we will see the introduction to performance evaluation.

Introduction to performance evaluation

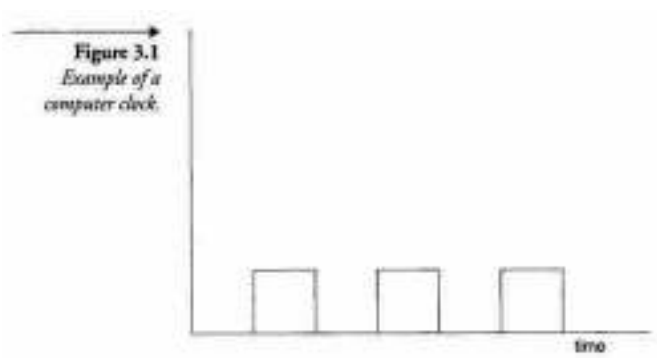
- Understanding all elements of the computer system.
- All aspects are important while understanding the issues.
- Realizing the requirements.
- Computer system must function correctly.
- Performing the intended function efficiently.
- Describing the computer system's design.
- Construction.
- Fielding.
- Life-cycle maintenance.

Overview

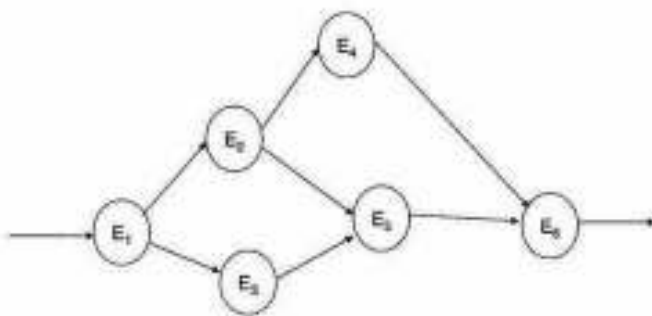
- Here, we will understand about the “Events”.

Events

- Time is an important measure.
- Useful if we can measure under computer evaluation.
- An event describes an entity of interest in our system.
- E.g. Clock cycle.



- All events must be controlled.
- Each action is designed to synchronize with other actions.
- Small actions makes larger events.
- System analyst must have understanding of the events.
- And the relationship among different events.



Overview

- Here, we will learn about the measurements (sampling).

Measurements (sampling)

- Events represent all of the real actions that occur in the computer system.
- Must know all the possible conditions.
- Measurements (sampling)
- State is an important concept in any computer system.

$$S = \{E_1(\text{value}), E_2(\text{value}), E_3(\text{value}), \dots, E_n(\text{value})\}$$

- Each event must have valid components.
- Primary types of measures are:
 - Count a number of items over a given time period.
 - Measures all state variables.
 - Measures the fraction of time the system is within a state.
 - Valid state must reach.
- Many ways are there to measure system events and decision depends on many factors.
- In hardware monitoring system analyst has ability to add instrumentation.
- Can also use integral test hardware.
- The hardware monitoring be designed as an integral component of the system.
- Must determine and define all aspects.
- Software monitoring requires designers to system hardware elements.
- As well as low level software elements.
- Need to have access to low level operating system calls.
- Hybrid monitoring uses the concepts of both software and hardware monitoring.
- Synchronization is required.

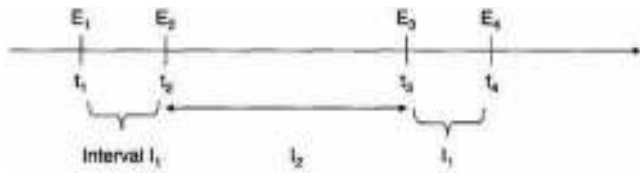
Module – 228

Overview

- Here, we will learn about the intervals.

Intervals

- An interval represents a period of time.
- System environment is system clocks.
- Instruction execution cycle.
- Intervals are same if they represent the same sequence of events.
- Separate sequence are represented by two intervals then they are unrelated.
- **Intervals**



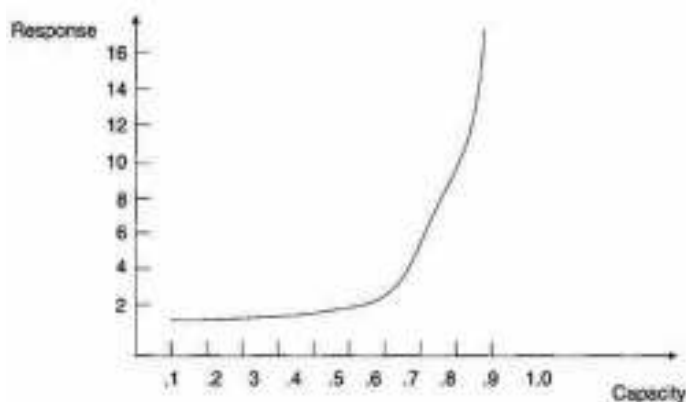
Module – 229

Overview

- Here, we will learn about the response.

Response

- Represent a measure of period of time a user or application to wait before task assignment.
- Typical measure may pit the response time.
- **Response**



- Load increases above specific capacity the response increases.
- Study the measurable sequences against system load.
- How these loads impact each other.

Module – 230

Overview

- Here, we will learn about the Independence.

Independence

- Occurrence if one does not influence the outcome of other.
- Important concept while evaluating systems.
- Two programs that cannot run concurrently must be considered independent.
- Running on the same operating system and hardware.
- Cannot interfere each other then they are separate and unrelated.
- Important in modeling to define all elements and their relationship.

Module – 231

Overview

- Here, we will learn about randomness.

Randomness

- Randomness is a property of an event and its reoccurrence.
- Have no pattern.
- Realization of the property of independence.
- It's a mathematical concept.
- External sources can be viewed as random events in systems.
- Important to analyze systems using mathematical concepts.

Module – 232

Overview

- Here, we will learn about the workloads.

Workloads

- Simply we can say load.
- Event or event sequences presented to system to model.
- Represent the number of events offered to system for execution.
- Arranging the requests and perform them by the system.
- The duration could be endless.
- The load is re-entered after prescribed period of time.
- The database community has developed a set of transactional workloads.

Module – 233

Overview

- Here, we will understand about the problems encountered in model development and use.

Problems encountered in model development and use

- Must start with having concept that what we are doing.
- Determining what to do.
- Two main classes performance assessment.
- First take an existing system and design some experiments.
- Second is either analytical modeling or simultaneous.
- Analyst uses following measures:
- Responsiveness and use levels.
- Mission-ability and dependability.
- Productivity and predictability.
- Analyst must avoid the following mistakes:
- Having no goals.
- Setting biased goals.
- Unsystematic approach to development.
- Choosing of incorrect performance metrics.
- Choosing non-stressful workload.
- To solve these problems the analyst must ask some questions to himself/herself.

Module – 234

Overview

- Here, we will understand about how to conduct a case study

A Case Study

- First Step: Define the System that we wish to study
- E.g. Remote pipes vs. Remote Procedures
- Focus on Services – small and large data transfers
- Select Metrics
- Rate of access
- Time for Performance
- Resource requirements per service
- Evaluate the developed model and start over again
- Trial and error
- Conducting our own: Modeling a ? ? ?
- A Bus system
- A railway terminal

- A hospital
- A human cell
- Human brain
- Ecosystem

Module – 235

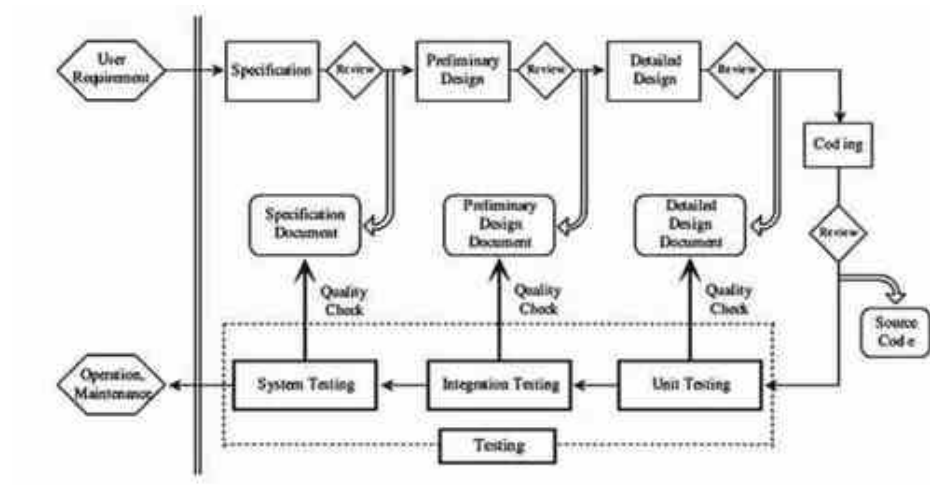
Overview

- Here, we will understand about software reliability

Introduction to Software Reliability

- Software fails on a regular basis
- This is a serious issue
- Especially in case of critical systems – nuclear reactors, patient care etc.
- Problems with X-Ray machine
- Problems with SCUD Missiles
- Problems with rocket launch
- Etc.

Introduction to Software Reliability



Module – 236

Overview

- Here, we will understand about definitions about reliability

Definitions and Software Reliability Model

- Generally, a software failure caused by software faults latent in the system cannot occur except on a specific occasion when a set of specific data is put into the system under a specific condition, i.e. the program path including software faults is executed.
- A software system is a product which consists of the programs and documents produced through a software development process
- Software failure is defined as an unacceptable departure of program operation from the program requirements.
- The cause of software failure is called a software fault.
- Then, software fault is defined as a defect in the program which causes a software failure.
- The software fault is usually called a software bug.
- Software error is defined as human action that results in the software system containing a software fault
- **Definitions and Software Reliability Model**

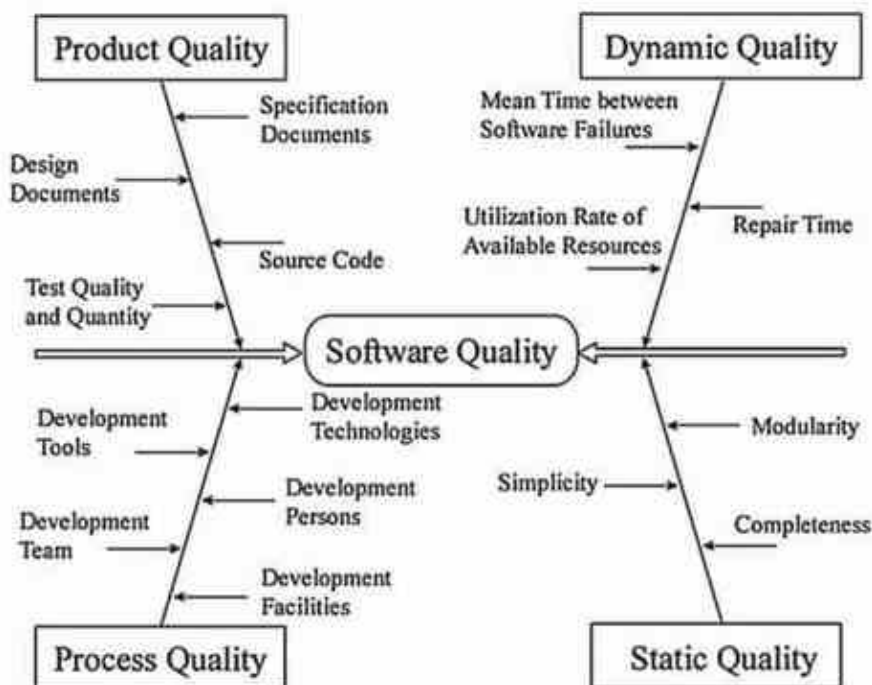
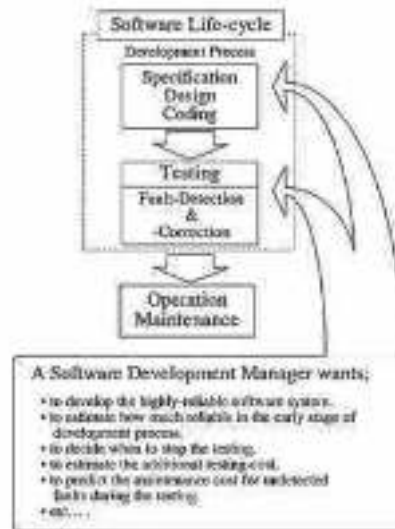


Fig. 1.2 Elements of software quality based on a cause-and-effect diagram

- **Definitions and Software Reliability Model**

Fig. 1.3 Aim of software quality/reliability measurement and assessment.



• Definitions and Software Reliability Model

Terminologies

- **Software Failure**
— an unacceptable departure of program operation from the program requirements
- **Software Fault**
— a defect in the program which causes a software failure (software bug)
- **Software Reliability**
— the attribute that a software system will perform without causing software failures over a given time period under specified conditions
— measured by its probability

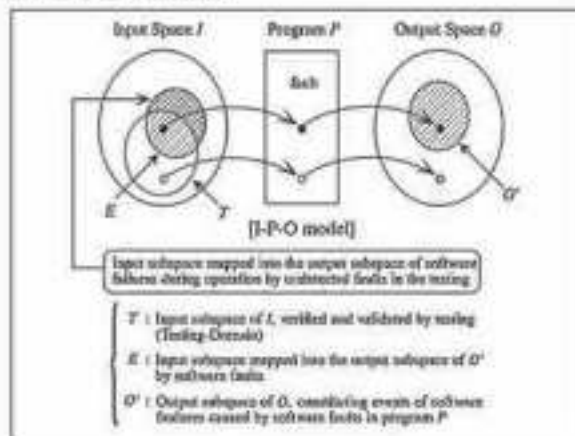


Fig. 1.4 An input-program-output model for software behavior

• Definitions and Software Reliability Model

Software Reliability	Hardware Reliability
(1) Software failures can be due to a myriad of phenomena.	(1) Hardware failures can be due to wear.
(2) Software reliability is, inherently, determined during the earlier phase of the development process, i.e. specification, analysis and design phases.	(2) Hardware reliability is affected by deficiencies injected during all phases of the development, operation, and maintenance.
(3) Software reliability can not be improved by redundancy with identical versions.	(3) Hardware reliability can be improved by redundancy with identical units.
(4) A verification method of software reliability has not been established.	(4) A testing method of hardware reliability is established and standardized.
(5) A maintenance technology is not established since the market of software products is rather recent.	(5) A maintenance technology is advanced since the market of hardware products is established and the user environment is seized.

Fig. 1.5. Comparison between the characteristics of software reliability and hardware reliability.

Module – 237

Overview

- Here, we will understand about software reliability growth modeling

Software Reliability Growth Modeling

- Generally, a mathematical model based on stochastic and statistical theories is useful to describe the software fault-detection phenomena or the software failure-occurrence phenomena and estimate the software reliability quantitatively.
- **Software Reliability Growth Modeling**

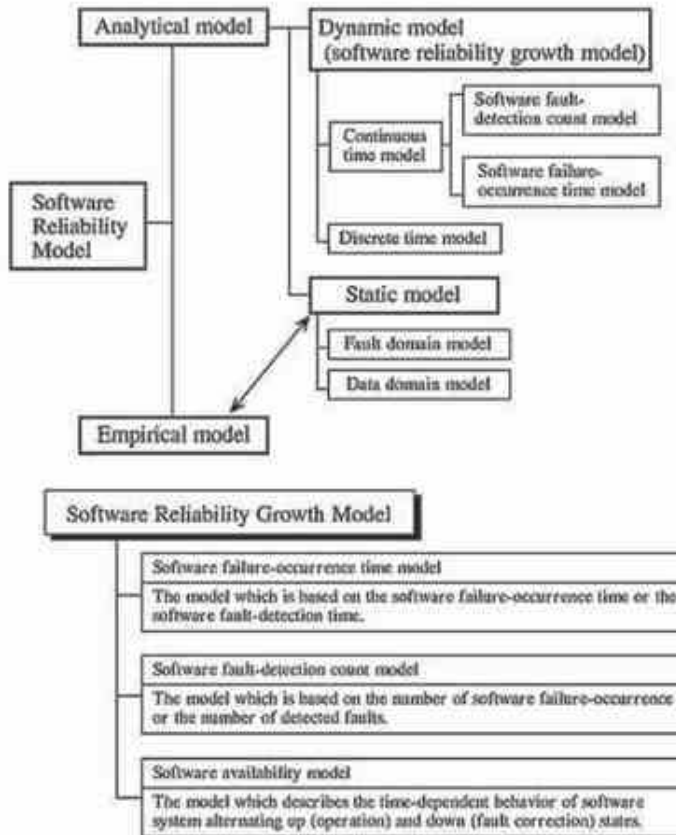


Fig. 1.6 Hierarchical classification of software reliability models

• Software Reliability Growth Modeling

- $N(t)$ the cumulative number of software faults (or the cumulative number of observed software failures) detected up to time t .
- S_i the i th software-failure occurrence time ($i = 1, 2, \dots; S_0 = 0$).
- X_i the time-interval between $(i - 1)$ -st and i th software failures ($i = 1, 2, \dots; X_0 = 0$).

- **Software Reliability Growth Modeling**

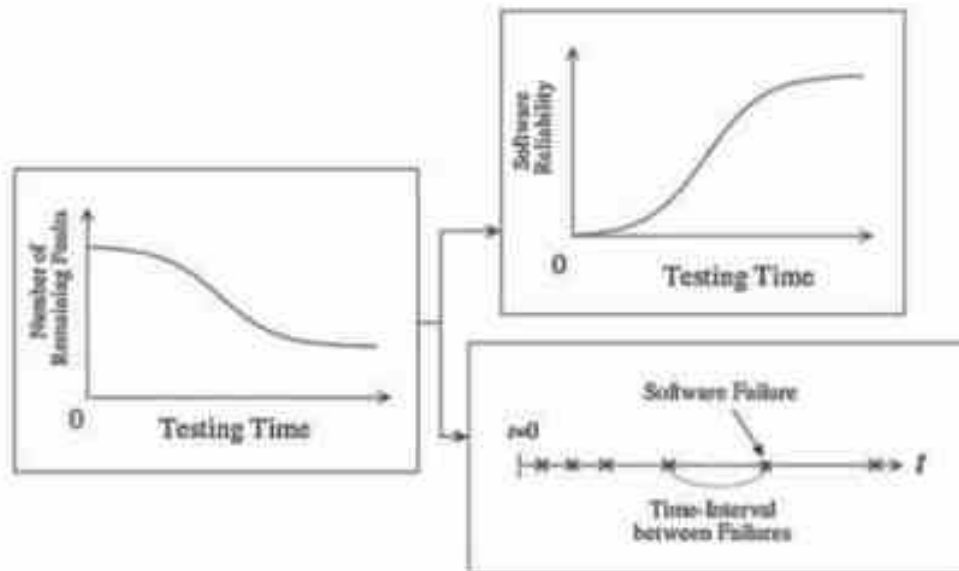


Fig. 1.7 Software reliability growth

Fig. 1.7 Software reliability growth

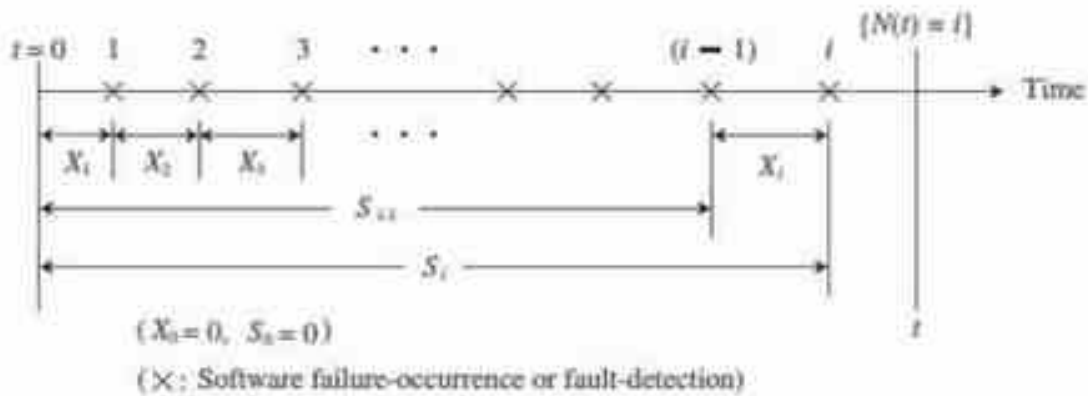


Fig. 1.8 The stochastic quantities related to a software fault-detection phenomenon or a software failure-occurrence phenomenon

Module – 238

Overview

- Here, we will understand about imperfect debugging modeling

Imperfect Debugging Modeling

- Most software reliability growth models proposed so far are based on the assumption of perfect debugging
- i.e. All faults detected during the testing and operation phases are corrected and removed perfectly.
- However, debugging actions in real testing and operation environment are not always performed perfectly.
- For example, Typing errors invalidate the fault-correction activity or fault-removal is not carried out precisely due to incorrect analysis of test results
- As a result it is preferable to develop an incorrect debugging model.
- This will result in a better estimation of reliability

Module – 239

Overview

- Here, we will understand about imperfect debugging model with perfect correction rate

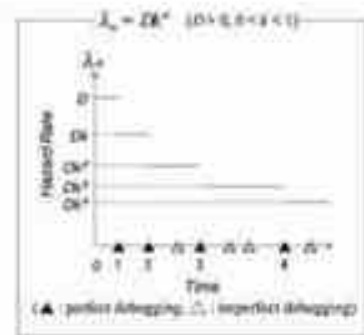
Imperfect Debugging Model with Perfect Correction

- To model an imperfect debugging environment, the following assumptions are made:

1. Each fault which causes a software failure is corrected perfectly with probability p ($0 \leq p \leq 1$). It is not corrected with probability $q (= 1 - p)$. We call p the perfect debugging rate or the perfect correction rate.
2. The hazard rate is given by (1.5) and decreases geometrically each time a detected fault is corrected (see Fig. 1.14).
3. The probability that two or more software failures occur simultaneously is negligible.
4. No new faults are introduced during the debugging. At most one fault is removed when it is corrected, and the correction time is not considered.

- Imperfect Debugging Model with Perfect Correction

Fig. 1.14 Behavior of hazard rate



- Imperfect Debugging Model with Perfect Correction

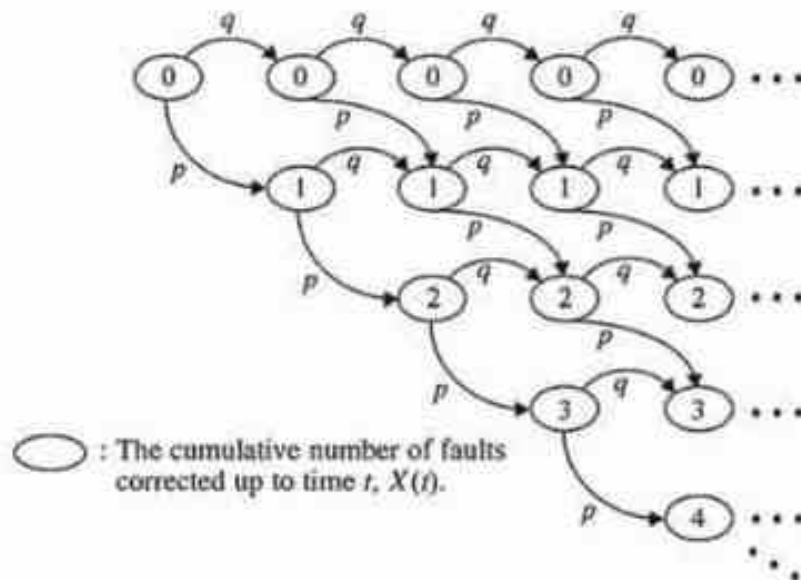


Fig. 1.15 A diagrammatic representation of transitions between states of $X(t)$

- Let $X(t)$ be a random variable representing the cumulative number of faults corrected up to the testing time t . Then, $X(t)$ forms a Markov process.
- That is, from assumption 1, when i faults have been corrected by arbitrary testing time t ,

$$X(t) = \begin{cases} i, & \text{with probability } q, \\ i+1, & \text{with probability } p. \end{cases} \quad (1.14)$$

Overview

- Here, we will understand about imperfect debugging modeling with introduced faults

Imperfect Debugging with Introduced Faults

- Besides the imperfect debugging factor above in fault-correction activities, we consider the possibility of introducing new faults in the debugging process.
- It is assumed that two kinds of software failures exist in the dynamic environment
- (F1) software failures caused by faults originally latent in the software system prior to the testing (which are called inherent faults)
- (F2) software failures caused by faults introduced during the software operation owing to imperfect debugging.
- In addition, it is assumed that one software failure is caused by one fault
- And that it is impossible to discriminate whether the fault that caused the software failure that has occurred is F1 or F2.
- As to the software failure-occurrence rate due to F1, the inherent faults are detected with the progress of the operation time.
- In order to consider two kinds of time dependencies on the decreases of F1, let
- $\lambda_i(t) = (1,2)$ denote software failure occurrence rate for F1
- For F2, it is constant ($\lambda > 0$)

Overview

- **Here, we will understand about software availability modeling**

Software Availability Modeling

- Recently, software performance measures such as the possible utilization factors have begun to be interesting for metrics as well as the hardware products.
- That is, it is very important to measure and assess software availability,
- It is defined as the probability that the software system is performing successfully, according to the specification, at a specified time point
- The actual operational environment needs to be more clearly reflected in software availability modeling, since software availability is a customer-oriented metrics.
- **Software Availability Modeling**

• Software Reliability : — the attribute that software systems can continue to perform and do not cause any software failures <i>for a given time period</i>, under a specified environment.
■ <u>software failure</u> [an unacceptable departure from program operation caused by a software fault remaining in the software system.]
• Software Availability : — the attribute that software systems are available and performing <i>at a given time point</i>, under a specified environment.

- Software fails on a regular basis
- This is a serious issue
- Especially in case of critical systems – nuclear reactors, patient care etc.

Module – 242

Overview

- Here, we will understand about model description for software availability modeling

Model Description

- **Assumptions**

1. The software system is unavailable and starts to be restored as soon as a software failure occurs, and the system cannot operate until the restoration action is complete
2. The restoration action implies debugging activity, which is performed perfectly with probability a
 - $(0 < a \leq 1)$ and imperfectly with probability $b (= 1 - a)$.
 - a is the perfect debugging rate
 - One fault is corrected and removed from the software system when the debugging activity is perfect.
3. When n faults have been corrected, the time to the next software failure- occurrence and the restoration time follow exponential distributions
4. The probability that two or more software failures will occur simultaneously is negligible

Overview

- Here, we will understand about software availability measures

Model Description

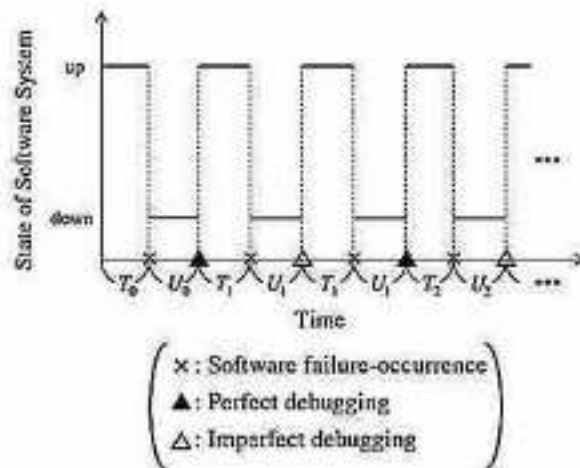


Fig. 1.17 Sample behavior of the software system alternating between up and down states

- Software Availability Measures

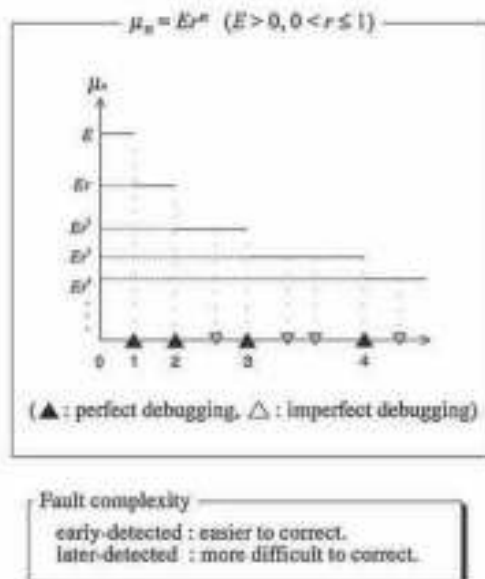
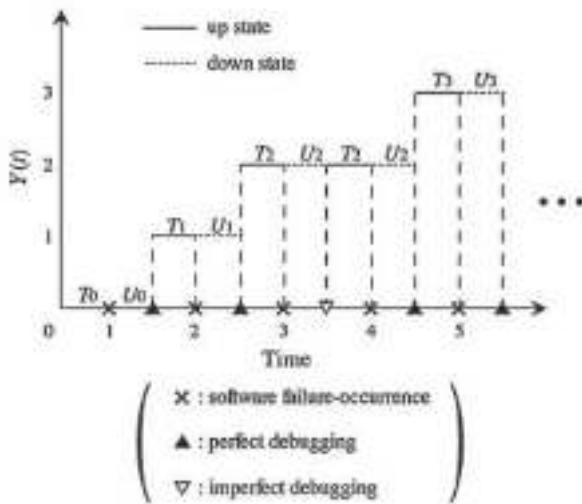


Fig. 1.18 Behavior of restoration rate



- Software Availability Measures

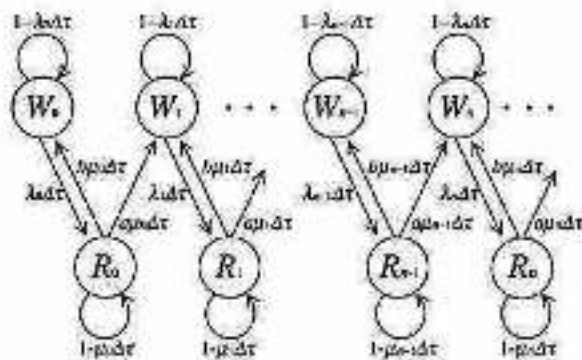


Fig. 1.20 A state transition diagram for software availability modeling

- Software Availability Measures

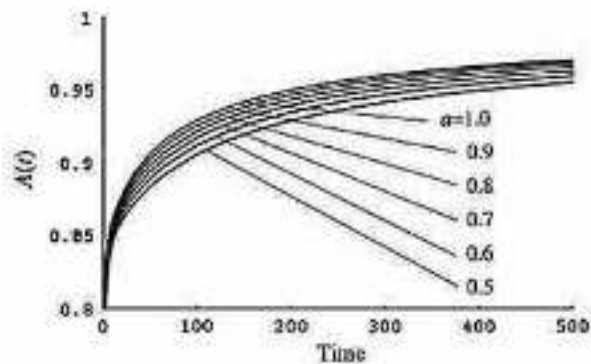


Fig. 1.21 Dependence of perfect debugging rate α on $A(t)$

- **Software Availability Measures**

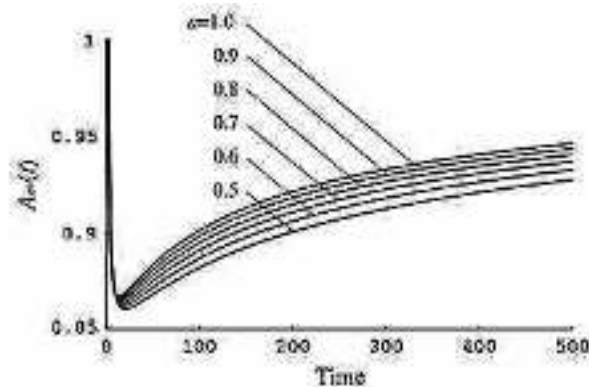


Fig. 1.22 Dependence of perfect debugging rate α on $A_{\text{perfect}}(t)$

Module – 244

Overview

- Here, we will understand about software reliability assessment

Application of Software Reliability Assessment

- It is very important to apply the results of software reliability assessment to management problems with software projects for attaining higher productivity and quality.
- We discuss three software management problems as application technologies of software reliability models.
- Optimal Software Release Problem
- Statistical Software Testing-Progress Control
- Optimal Testing-Effort Allocation Problem

Module – 245

Overview

- Here, we will understand about optimal software release issues

Optimal Software Release Problem

- Recently, it is becoming increasingly difficult for the developers to produce highly reliable software systems efficiently.
- Thus, it has been necessary to control a software development process in terms of quality, cost, and release time.

- In the last phase of the software development process, testing is carried out to detect and fix software faults, introduced by human work, prior to its release for the operational use.
- The software faults that cannot be detected and fixed remain in the released software system after the testing phase.
- Thus, if a software failure occurs during the operational phase, then a computer system stops working and it may cause serious damage.
- If the duration of software testing is long, we can remove many software faults in the system and its reliability increases. However, this increases the testing cost and delays software delivery.
- In contrast, if the length of software testing is short, a software system with low reliability is delivered and it includes many software faults which have not been removed in the testing phase.
- Thus, the maintenance cost during the operation phase increases.
- It is therefore very important in terms of software management that we find the optimal length of software testing, which is called an optimal software release problem
- These decision problems have been studied in the last decade by many researchers.
- In optimal software release problems we consider both a present value and a warranty period (in the operational phase)

Module – 246

Overview

- Here, we will understand about statistical software testing-progress control

Statistical Software Testing-Progress Control

- As well as quality/reliability assessment, software-testing managers should assess the degree of testing-progress.
- We can construct a statistical method for software testing-progress control based on a control chart method
- This method is based on several instantaneous fault-detection rates derived from software
- **Statistical Software Testing-Progress Control**

The procedure of testing-progress control is shown as follows:

- Step 1: An appropriate model is selected to apply and the model parameters are estimated by the method of least-squares.
- Step 2: To certify goodness-of-fit of the estimated regression equation for the observed data, we use the F -test.
- Step 3: Based on the result of the F -test, the central line and upper and lower control limits of the control chart are calculated. The control chart is drawn.
- Step 4: The observed data are plotted on the control chart and the stability of the testing-progress is judged.

- **Statistical Software Testing-Progress Control**

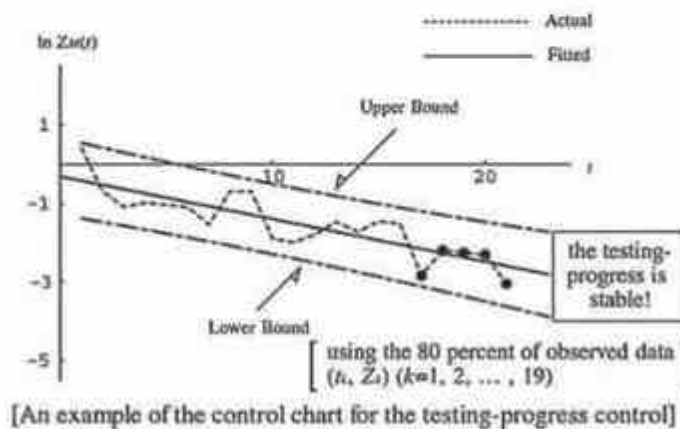
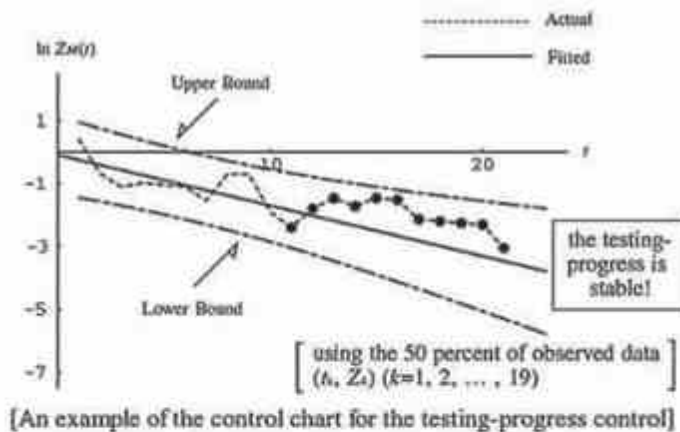


Fig. 1.25 Examples of the control chart for the testing-progress control

Module – 247

Overview

- Here, we will understand about Optimal Testing-Effort Allocation

Optimal Testing-Effort Allocation

- We discuss a management problem to achieve a reliable software system efficiently during module testing in the software development process by applying a testing-effort-dependent software reliability growth model
- We take account of the relationship between the testing-effort spent during the module testing and the detected software faults where the testing-effort is defined as resource expenditures spent on software testing, e.g. manpower, CPU hours, and executed test cases.

- The software development manager has to decide how to use the specified testing-effort effectively in order to maximize the software quality and reliability.
- That is, to develop a quality and reliable software system, it is very important for the manager to allocate the specified amount of testing-effort expenditure for each software module under some constraints.
- We can observe the software reliability growth in the module testing in terms of a time-dependent behavior of the cumulative number of faults detected during the testing stage.
- Based on the testing-effort dependent software reliability growth model, we consider the following testing-effort allocation problem:
- Based on the testing-effort dependent software reliability growth model, we consider the following testing-effort allocation problem:

1. The software system is composed of M independent modules. The number of software faults remaining in each module can be estimated by the model.
2. The total amount of testing-effort expenditure for module testing is specified.
3. The manager has to allocate the specified total testing-effort expenditure to each software module so that the number of software faults remaining in the system may be minimized.

Module – 248

Overview

- Here, we will understand about the “formal modeling”.

Formal modeling

- Petri nets PNs provide graphical tool and notation method.
- Include arrival rate and service rate.
- Individual state information.
- Contention and concurrency.
- Petri Nets have place in computer systems performance assessment.
- Between analytical queuing theory and computer simulation.
- System state more complete than analytical models.
- But not as simulations.
- Have fundamental theory but act like simulations.
- Single entities, movement and affect effect of the state of the entire system.
- Petri Nets was introduced in 1966.
- It describes concurrent systems.
- Followed by continuous improvements.
- Timing of transitions.
- Priority of transitions.
- Types to Token.
- Colors depicting.
- Analysis tools and modelling aid.

Overview

- Here, we will understand about the “Basic notation”?

Basic notation

- Petri Nets provides abstraction and information flow.
- Using four fundamental components.
- Places as Circles.
- Transition as bars.
- Arc as line segments.
- Token as dots.
- Places represents useful system components and their state e.g. disk drive.
- Transitions represents events.
- Action of reading an item from disk drive.
- Arcs represents relationship between transitions and places.
- Proved path for activation of transition.
- Tokens define state of Petri net.

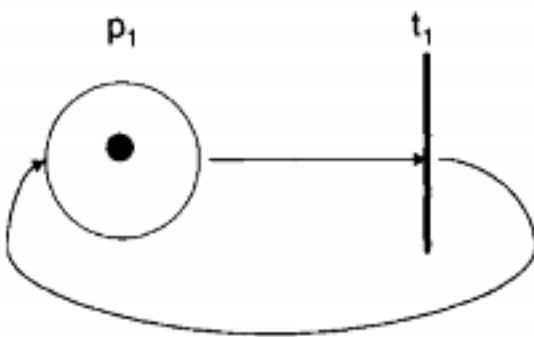
Forever Cycling Petri Net

The place is connected to the transition by an arc, and the transition is likewise connected to the place by a second arc.

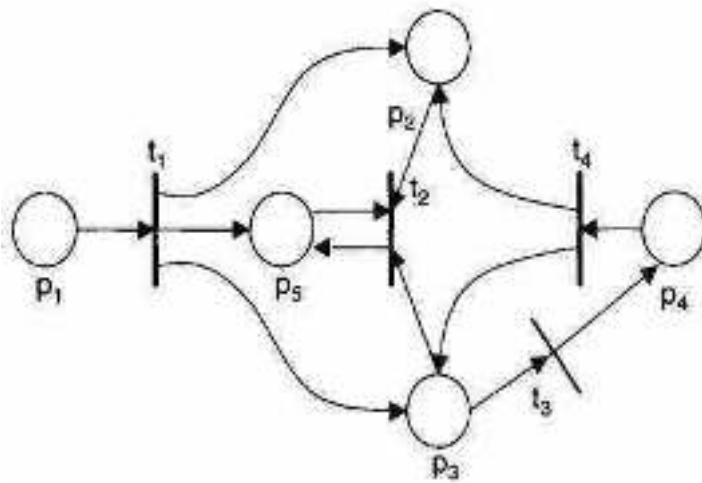
The former arc is an input arc, while the latter arc is an output arc.

The placement of a token represents the active marking of the Petri net state

The Petri net shown represents a net that will continue to cycle forever.



Petri Net Graphical Representation



Places(P), Transitions(T) Arc(input I, Output O), and Tokens for net (MP).

Five tuple $M=(P,T,I,O,MP)$.

P represent set of Places $P\{p_1, p_2, \dots p_n\}$.

P represent set of Transitions $P\{t_1, t_2, \dots t_n\}$.

I as input functions $I = \{I_{t1}, I_{t2} \dots I_{tn}\}$ and as output functions $O = \{O_{t1}, O_{t2} \dots O_{tn}\}$.

MP represents marking of places with tokens.

MP_O is represented as ordered tuple of magnitude where n represents number of places in petri net.

Module – 250

Overview

- Here, we will understand about the “Basic notation”.

Basic notation-II

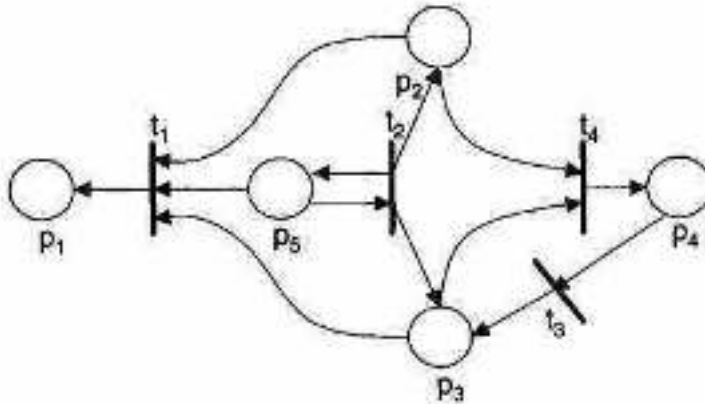
Petri Net Dual

- Petri net model also has dual.
- In dual of petri, transitions are changed to places and places to transitions.

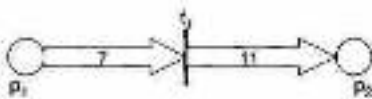
- The input becomes output function and output becomes output function.

Petri Net Inverse

- A petri net model is also defined for inverse.
- Places and transitions are kept same.
- Whereas, input functions switches to output functions.
- **Petri Net Inverse Representation**



- **Petri Net Multigraphs**
- A petri net model is also defined for Multigraphs.
- It implies there could be several arcs between a single place and a transition.
- Can be represented as:
 - Multiple arc as thick arc with numbers inside.
 - A bold arc with number attached to it.
- **Basic Petri Net Multigraphs Representation**



Petri Net State

- A petri net also has a state defined by the cardinality of tokens.
- Represented as function, μ^t .

$$\mu: P \rightarrow N$$

- μ^t can be defined as vectors.

$$\mu = (\mu_1, \mu_2, \mu_3, \dots, \mu_n),$$

Petri Net Multigraphs

- Where,

$$n = |P| \text{ and each } \mu_i \in N, i = 0, \dots, n$$

- Place marking function and vector relation:

$$\mu(p_i) = \mu_i$$

- Petri net structure:

$$M = (P, T, I, O, \mu_i)$$

- It represents state of petri net at time t.

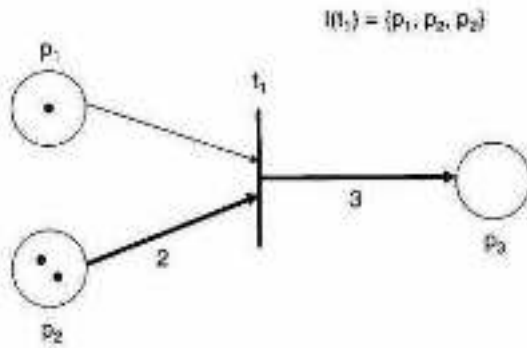
Module – 251

Overview

- Here, we will understand about the “Classical Petri Nets”?

Classical Petri Nets

- By using notations Petri nets can be used in modelling.
- A new petri net state transition notation is required.
- These transitions are helpful in movements.
- The exact firing moment can be captured as a clock signal.
- During Clock cycle, all gates start signal execution.
- In petri net all transitions fire once during this cycle.

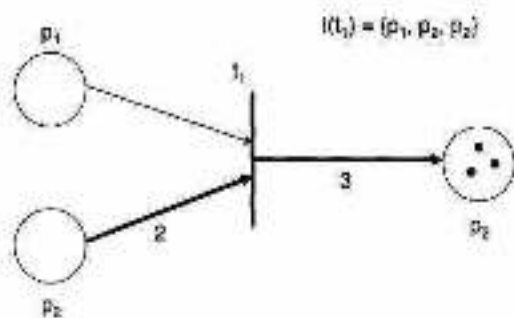


The Petri net shown has the marking $g_0 = (1, 2, 0)$, input function $I(t_1) = \{p_1, p_2, p_2\}$, and output function $O(t_1) = \{p_3, p_3, p_3\}$.

Given this initial marking and the defined input and output functions, transition t_1 is enabled, since it requires one token contributed by p_1 and two tokens contributed by p_2 to have all of its input functions satisfied with tokens available in the named places and in the required quantities.

transition is enabled at the beginning of a Petri net's firing cycle, it will fire the transition, causing the movement of the number of tokens from its input places to its output places. The result of this firing will be a new Petri net state, B_1 . For example, in given the initial state B_0 , transition t_1 will fire at the beginning of the firing cycle, removing one token from place p_1 and two tokens from place p_2 and placing these three tokens into place p_3 .

Forever Cycling Petri Net



The resulting new Petri net state is represented by the marking $B_1 = (0, 0, 3)$, and is depicted in the figure. The firing action is considered an atomic action, in that it appears as if all of the tokens are removed from the input places and deposited into the output places instantaneously. The firing of the Petri net provides the movement from one state to a new state. The new state is the only state reachable in a single total transition firing of all the enabled transitions. The collection of all possible states that can be represented by this Petri net and its initial markings is called state-space.

This implies that a Petri net can only fire enabled transitions, and, after they fire, they cannot fire any newly enabled transitions until the next cycle.

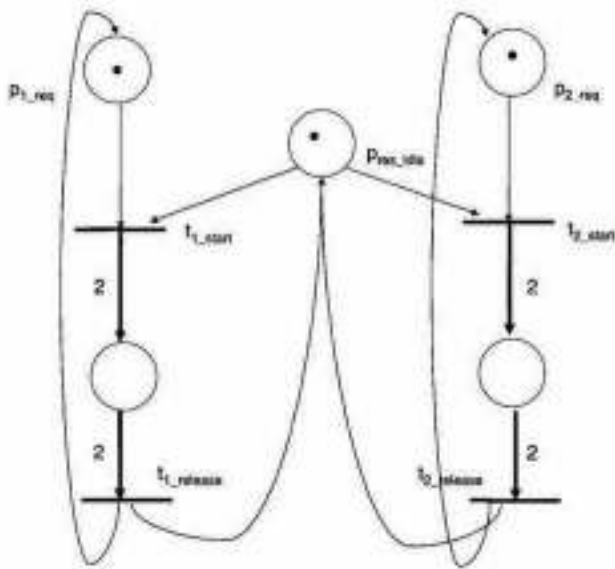
Petri Net Set Notation Representation

Petri nets enables modeling not available through queuing theory.

- Synchronization.
- Conflict.
- Concurrency.

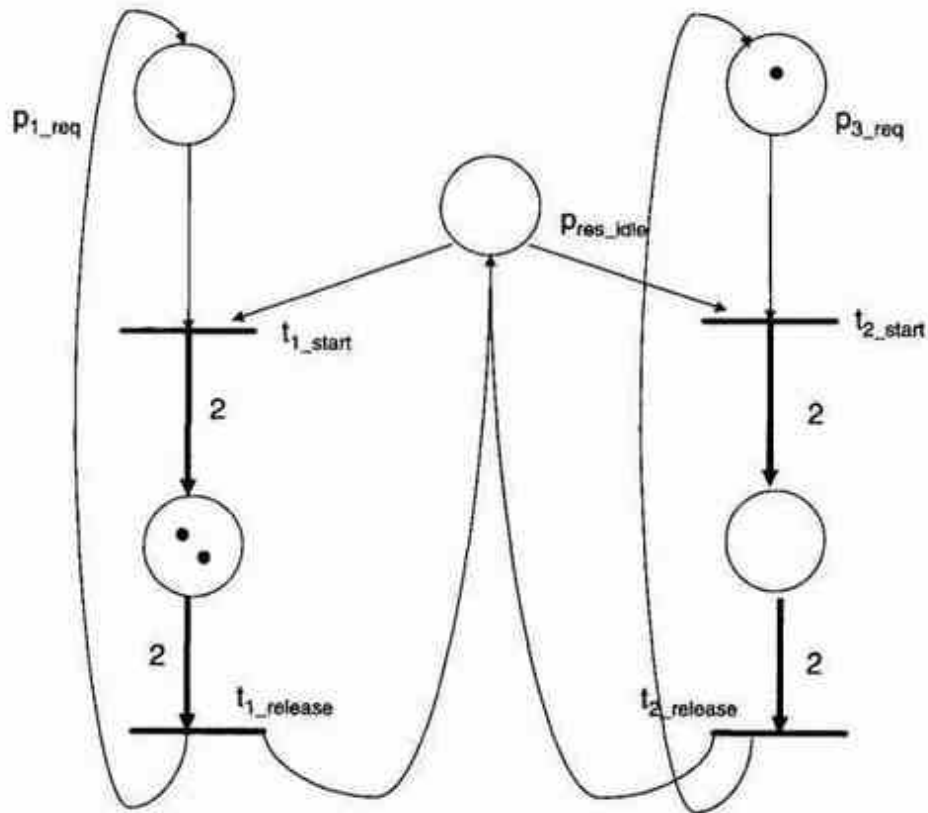
These are not easily defined and modeled by queuing theory.

Resource Sharing



We model two user processes requesting a specific resource (P_{res_idle}). If the resource is idle, there is a token present in the (P_{res_idle}) place. If a process wanting this resource has a token in its place (e.g., P_{p1_req}), and the resource is idle (a token in P_{res_idle}), then the transition (e.g., t_{1_start}) is enabled and can fire on the next cycle. When the transition fires, the resource now becomes unavailable (it is busy), and the second process must wait until the resource is released.

Allocated Resources



Module – 252

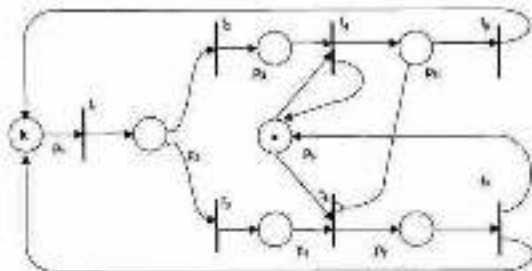
Overview

- Here, We will understand “Classical Petri Nets”.

Classical Petri Nets-II^{II}

Classical Petri Nets

- A Petri net's state, is said to be reachable from some other state, u^{II} .
- There exists some finite number of firings of the Petri net beginning at state u^{II} .



The reachability set for a Petri net with an initial marking, $\mu_0 P_n$, is denoted $RS(\mu_0 P_n)$ and is defined as the smallest set of markings, so that:

$$\mu_0 P_n \in RS(\mu_0 P_n) \text{ and}$$

$$\mu_0 P_n \in RS(\mu_0 P_n)$$

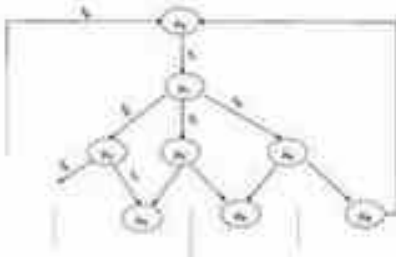
$$\mu_0 P_n \in RS(\mu_0 P_n) \in RS(\mu_0 P_n)$$

To determine the reachability set, we must begin from the initial state, $\mu_0 P_n$, and successively define each step possible stemming from this initial state and all states derivable from this state.

The reachability set contains no information about which transitions fired to reach the state markings. Such information can be found in a reachability graph shown next.

The reachability set contains no information about which transitions fired to reach the state markings. Such information can be found in a reachability graph shown next.

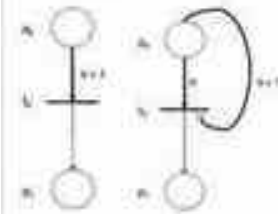
• Reachability Graph



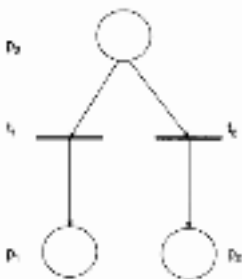
In this graph, each node represents a state of the Petri net and each arc represents a direct transition, which is possible from one end of the directed arc to the other and due to a single transition firing. For example, you can see that if we fire transition t_1 from $\mu_0 P_n$ we can get to a new state, $\mu_1 P_n$, where a token has moved from place 1 to place 2.

It is often desirable to model logical conditions. For example, to only fire a transition when there are more than n tokens in a place, we simply need to include an arc with weight $n + 1$. Since the basic condition on a transition firing is that its associated input places meet the conditions of the transition input function, by including a $n + 1$ redundant set items for the input place we can accomplish what we need shown in next figure.

• Component Test



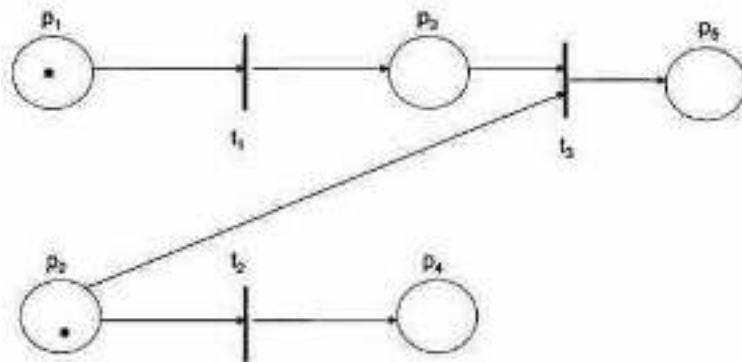
An important property required when modeling computer systems is that of conflict shown next.



In this example, when there is a token in place P_0 , both transitions t_1 and t_2 are enabled.

However, only one of them may fire, since there is only one token available. As soon as one fires, say t_1 , it removes the token from place P_0 and transfers it to place P_1 . As soon as transition

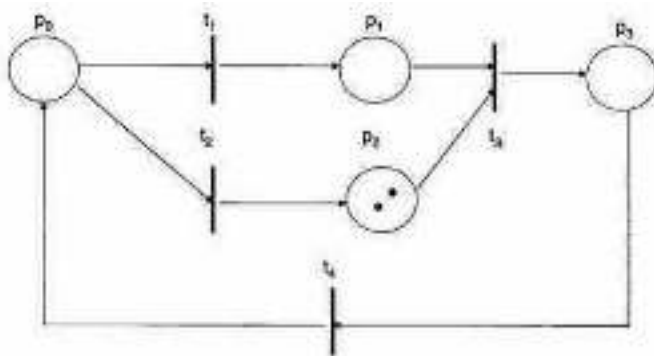
t_1 removes the token, transition t_2 is no longer enabled. If there were two tokens in place P_0 , then both of the transitions would be enabled and could fire during this firing cycle.



Very often a computer system can have both conflict and concurrency occur at the same time as shown.

we have an initial marking, $u = (1,1,0,0,0)$, which results in transitions t_1 and t_2 being enabled, the condition of concurrent transitions. If t_1 fires first, then we now have two transitions enabled, t_2 and t_3 . This then depicts conflict, since the token from P_2 can only satisfy one of the necessary conditions for the two transitions it is enabling at this point in time.

A Petri net can have a variety of other qualities as well.



A Petri net is deadlocked if there are no transitions in the net that are enabled. In the example shown, the net has an initial marking, $u_0 = (0,0,2,0)$. This marking results in no transitions being enabled and no hope after an infinite amount of time of becoming enabled. Conversely, a Petri net is considered live if there are any transitions enabled.

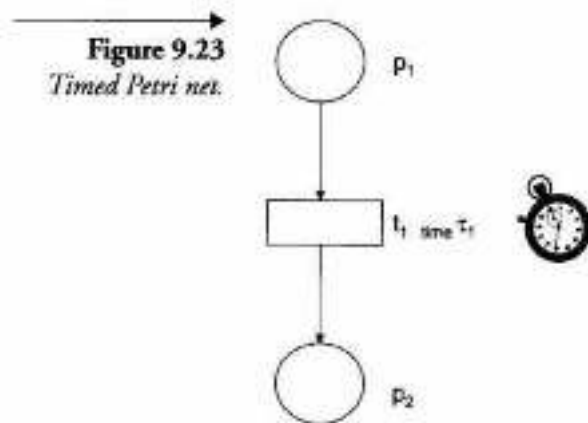
Module – 253

Overview

- Here, we will learn about the timed petri nets.

Timed Petri nets

- Every action takes some time to complete.
- Time addition provides Petri net modeler another powerful tool.
- Time in Petri net modeling is associated with transitions.



Transition $t_1^{f_1}$ = timed transition.

Semantic of the firing is different.

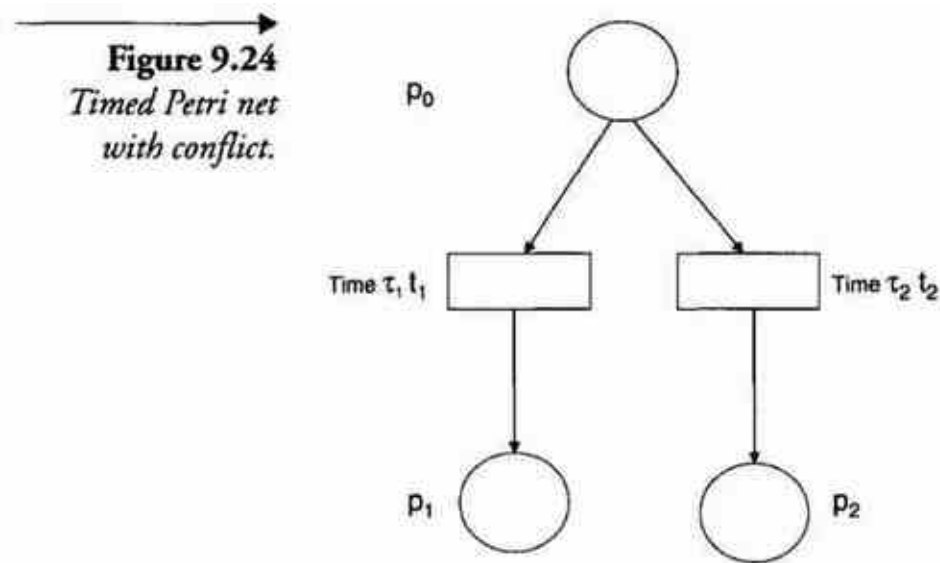
Once transition enabled, its time period clock timer is set.

Transition fires when timer reaches 0.

When token reaches $p_1^{f_1}$ the timer for transition $t_1^{f_1}$ is set to $\tau_1^{f_1}$.

Token moves to $p_2^{f_1}$ once $\tau_1^{f_1}$ time is passed.

The decrement of the timer must be constant speed.



If $t_1 < t_2, t_1 < t_2$.

Two timers would begin when $p_2^{t_2}$ receives a token.

When $t_1^{t_1}$ reached 0 then $t_2^{t_2}$ becomes disabled.

Reset the timer.

Allowing transition $t_2^{t_2}$'s timer to maintain the present clock.

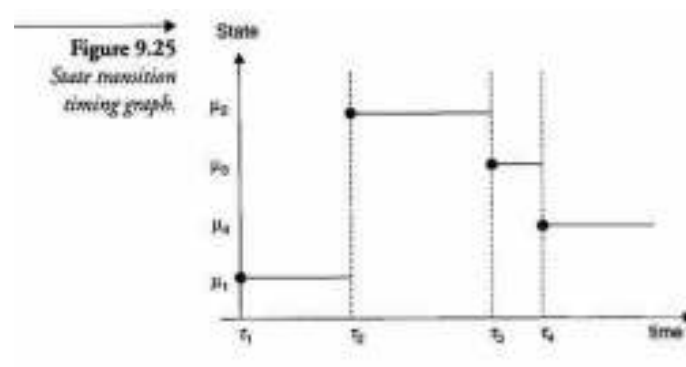
Module – 254

Overview

- Here, we will learn about the timed Petri nets.

Timed Petri nets

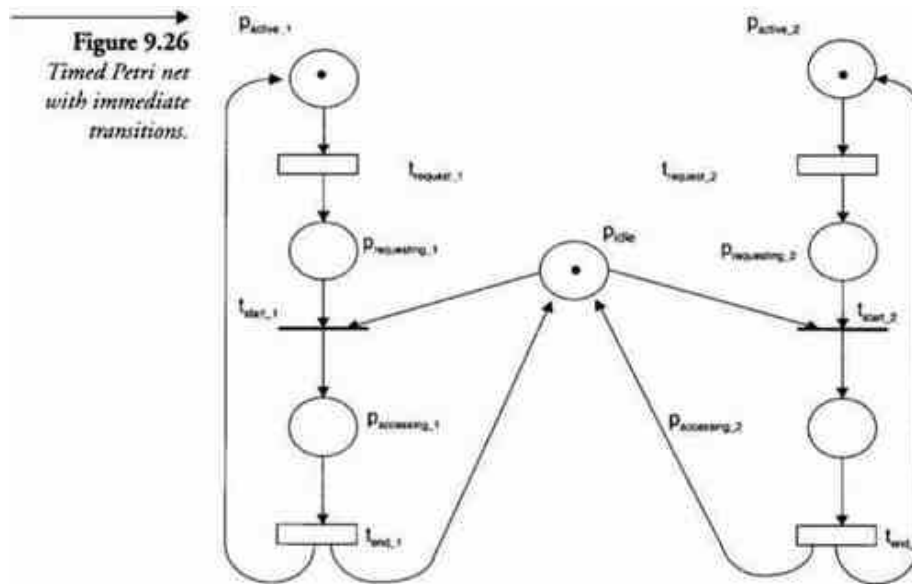
- Some Petri net models have proposed using state transition timing graphs.
- Each state μ is enumerated.
- Time period is set for each state in sequence.



- Time units can be represented using time transition sequences.
- $f_{1,2}[(t_1, t_2); (t_2, t_3); \dots; (t_{j-1}, t_j); \dots; (t_1, t_1); \dots]$

Time units are require to move from $\mu_1^{t_1}$ to $\mu_2^{t_2}$.

Transitions Can be also be represented as immediate transitions.



Immediate transitions are used to capture resources.

Acts like a semaphores.

Active process will win it and other processes have to wait until resources become free.

Module – 255

Overview

- Here, we will learn about the Priority-based Petri nets.

Priority-based Petri nets

- The Petri net is described by nine tuple.

$$M = (P, T, I, O, H, \Pi, PAR, Pred, \mu)$$

P represents the set of places.

T is the set of transitions.

I represents the set of input functions.

O represents the set of output functions.

H represents the inhibition functions.

PAR represents the parameter set for this Petri net.

$Pred$ represents the predicates defining how the parameter set can be distributed.

μ^H represents the set of rankings.

Π^H represents the priority function.

Petri nets are enabled with right amount of tokens.

No additional transitions are enabled.

$t_1^{c_1}$ has lowest priority at 1.

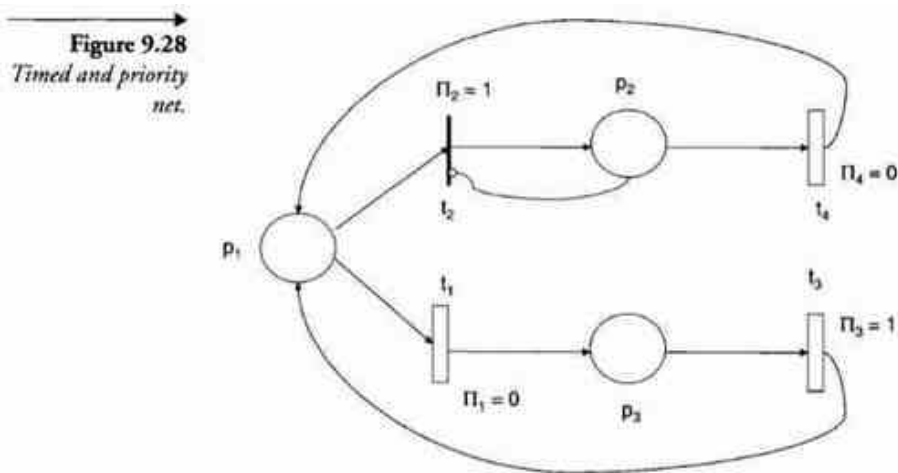
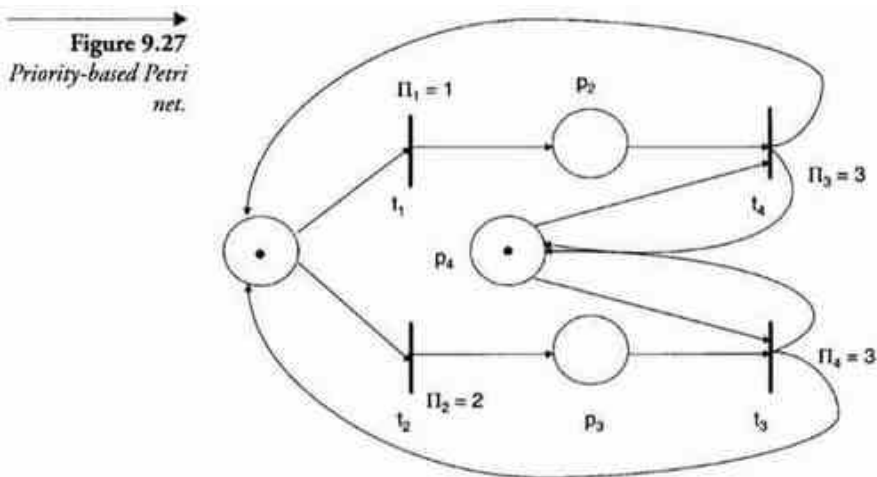
$t_2^{c_2}$ has lowest priority at 2.

t_3 and $t_4^{c_3}$ and t_4 have the highest priority at 3.

If we start initial marking $\mu = 1, 0, 0, 1^{\mu} = (1, 0, 0, 1)$.

Only $t_2^{c_2}$ will be enabled.

$t_1^{c_1}$ cannot be enabled.



Contention, confusion and concurrency condition can also be defined.

This picture has both time and priority features.

System can toggle between two events.

Inhibitor blocks the immediate transition at t_2^2 .

In this way it allows t_1^1 to get the service.

Module – 256

Overview

- Here, we will learn about the colored Petri nets.

Colored Petri nets

- Different token types.
- Colored tokens are used to represent graphically.

P represents the set of places.

T is the set of transitions.

I represents the set of input functions.

O represents the set of output functions.

H represents the inhibitor functions.

Par represents the parameter set for this Petri net.

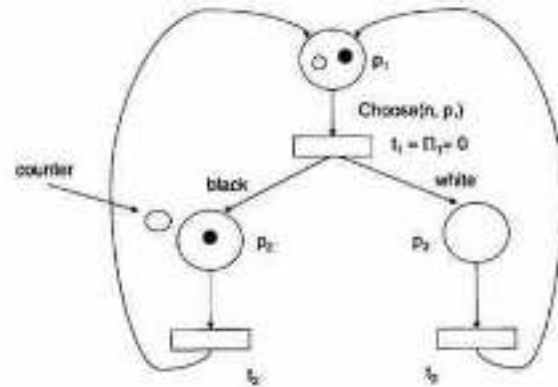
Pred represents the predicates defining how the parameter set can be distributed.

μ^M represents the set of markings.

Π^M represents the priority function

- Cardinality of the token is represented by all colors.
- All definitions of petri nets can be redefined now.
- And firing rules.

Figure 9.29
Generalized Petri
net.



Two token types.

Transitions can have priority and time associated with them.

$t_1^{t_1}$ has priority and time.

Arc from p_1 to $t_1^{t_1}$ has condition choose.

Other arcs are used to select specific token.

- We can model about any complex condition in systems.

Module – 257

Overview

- Here, we will learn about the Generalized Petri nets .

Generalized Petri nets

- Rate or weight transition function,
- $W_{tk}^{IV}(t_k, \mu)$ must be defined.
- $W_{tk}^{IV}(t_k)$ can refer to this when function needs no definition.
- This is rate of transition.

P represents the set of places.

T is the set of transitions.

I represents the set of input functions.

O represents the set of output functions.

H represents the inhibitor functions.

Par represents the parameter set for this Petri net.

Pred represents the predicates defining how the parameter set can be distributed.

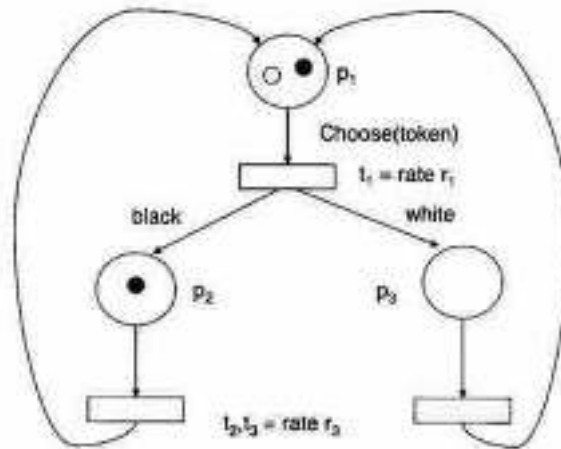
μ^k represents the set of markings.

Π^k represents the priority function

Generalized Petri nets

- To compute time to fire transition an exponential function is used.
- Rate must be define.
- Computed timer value is decremented until 0.
- System state changes from one state to another.
- **Generalized Petri nets**

Figure 9.30
Generalized
Petri Net.



initial marking is $\mu^0 = (2,0,0)$.

t_1 : is enabled then it will compute transition rate using mean rate $r(t_1)$.

Once timer value reaches 0 then we will use the selected token to choose the path.

- Each state will not be computed at the same time.