**Experiment No. 1**

**Aim:** To study of Simulation Software

**Theory: CASE STUDY ON GPSS**

1. **INTRODUCTION**

General Purpose Simulation System (GPSS) after creator Geoffrey Gordon is a discrete time simulation general-purpose programming language, where a simulation clock advances in discrete steps. A system is modeled as transactions enter the system and are passed from one service (represented by blocs) to another. This is particularly well-suited for problems such as a factory. GPSS is less flexible than simulation languages such as Simula and SIMSCRIPT II.5 but it is easier to use and more popular.

1. **HISTORY**

General Purpose Simulation System is a language designed for discrete event modeling. The language as it exists today is the result of more than twenty years of evolution. While GPSS has its roots in the early days of mainframe computing, its basic ideas have proven suitable for application to today’s problems using modern computing environments. The popularity of GPSS is due, in part, to its power of expression. A short, easily understood GPSS model would require many pages of FORTRAN coding to accomplish a similar goal. The GPSS user is free to concentrate on the important issues in the model being developed since the language itself collects statistics, produces tabulated results and performs a host of mundane tasks one would prefer not to deal with. The purpose of this brief article is to present some basic terminology and concepts which will enable the person meeting GPSS for the first time to understand the broad range of applications,

power and ease of use that this language brings to the simulationist.

1. **COMPONENTS**

GPSS provides a set of abstract components of various types and a set of operators called blocks which perform certain actions on the individual components. The transaction is the component which moves through a sequence of blocks that has been designed to model the system being studied. The state of the components of the model determines the details of how a block of a given type will operate.

1. **GPSS BLOCKS**
2. GENERATE BLOCK:

This block will produce a flow of transactions with inter-arrival times determined by the attribute values. The label is optional. The distribution of inter-arrival times follows a uniform probability distribution.

SYNTAX:

*line number label* GENERATE A,B,C,D,E

ATTRIBUTES:

A = average value of uniform distribution

B = half-width of uniform distribution

C = time delay before first transaction is generated

D = maximum number of transactions generated

E = priority allocated to transactions

1. QUEUE BLOCK:

This block will instruct GPSS to start gathering queuing statistics on the queue named in its attribute value.

The label is optional but may be necessary if you have to refer to this line from somewhere else in the program

SYNTAX:

*line number label* QUEUE A

ATTRIBUTES:

A = name of queue (for example: garage)

If a transaction arriving at the queue block cannot proceed because it is blocked by the next stage, then it will stay in the queue block until it can gain entry to the next stage.

1. DEPART BLOCK:

This block instructs GPSS that a transaction is leaving the queue named in its attribute value. This is necessary in order to compile the statistics on the queue. The label is optional.

SYNTAX:

*line number label* DEPART A

ATTRIBUTES:

A = name of the queue (for example: checkout)

1. SEIZE BLOCK:

This block allows the transaction to seize a facility if it is free. Thus it may be a car “seizing” a “facility” such as a petrol pump or a customer in a supermarket “seizing” a “facility” such as the checkout assistant. When the car or customer is being serviced by the facility, then it is said to “own the facility”. The label is optional.

SYNTAX:

*line number label* SIZE A

ATTRIBUTES:

A = name of facility (for example: pump)

1. RELEASE BLOCK:

A transaction entering this block informs GPSS that it is giving up ownership of the facility named in its attribute value. The label is optional.

SYNTAX:

*line number label* RELEASE A

ATTRIBUTES:

A = name of facility (for example: runaway)

1. ENTER BLOCK:

This Block instructs GPSS that a transaction has entered STORAGE. The name of storage is given by the first attribute value. The second attribute value gives the amount the storage will be incremented by, when the transaction enters the ENTER block. A STORAGE must be declared at the beginning of a program.

SYNTAX:

*line number label* ENTER A,B

ATTRIBUTES:

A = name of the storage (for example: warehouse)

B = increment storage by this value

1. LEAVE BLOCK:

This block instructs GPSS that a transaction is leaving a STORAGE. The first attribute gives the name of the STORAGE and the second attribute decrements the storage by the value of the attribute.

SYNTAX:

*line number label* LEAVE A,B

ATTRIBUTES:

A = name of the storage (for example: warehouse)

B = Decrement storage by the value

1. ADVANCE BLOCK:

This block represents the servicing of a transaction. The servicing times follow a uniform probability distribution. The label is optional.

SYNTAX:

*line number label* ADVANCE A,B

ATTRIBUTES:

A = average value of uniform distribution

B = half-width of uniform distribution

\* A transaction entering this block will be delayed by a time interval chosen at random from the specified probability distribution.

1. TERMINATE BLOCK:

This block destroys any transaction entering it and removes it from computer memory. Each time a transaction enters this block it decrements a counter by an amount equal to its attribute value. The counter is set by the user upon starting the simulation.

SYNTAX:

*line number label* TERMINATE A

ATTRIBUTES:

A = decrements simulation counter by this amount

1. TEST BLOCK:

This block can test the logical condition of a queue or storage according to a particular reference value. If a transaction enters the TEST block, the block will check this condition and if it is true, it will send the transaction to one destination in the program and if the condition is false, it will send it to another.

SYNTAX:

*line number label* TEST O A,B,C

ATTRIBUTES:

A = value and name of the block being referenced

B = reference value

C = destination for the transaction if the logical condition is not satisfied

1. TRANSFER BLOCK:

This block will take transactions entering it and transfer them to each of two different destinations according to laid down proportions.

SYNTAX:

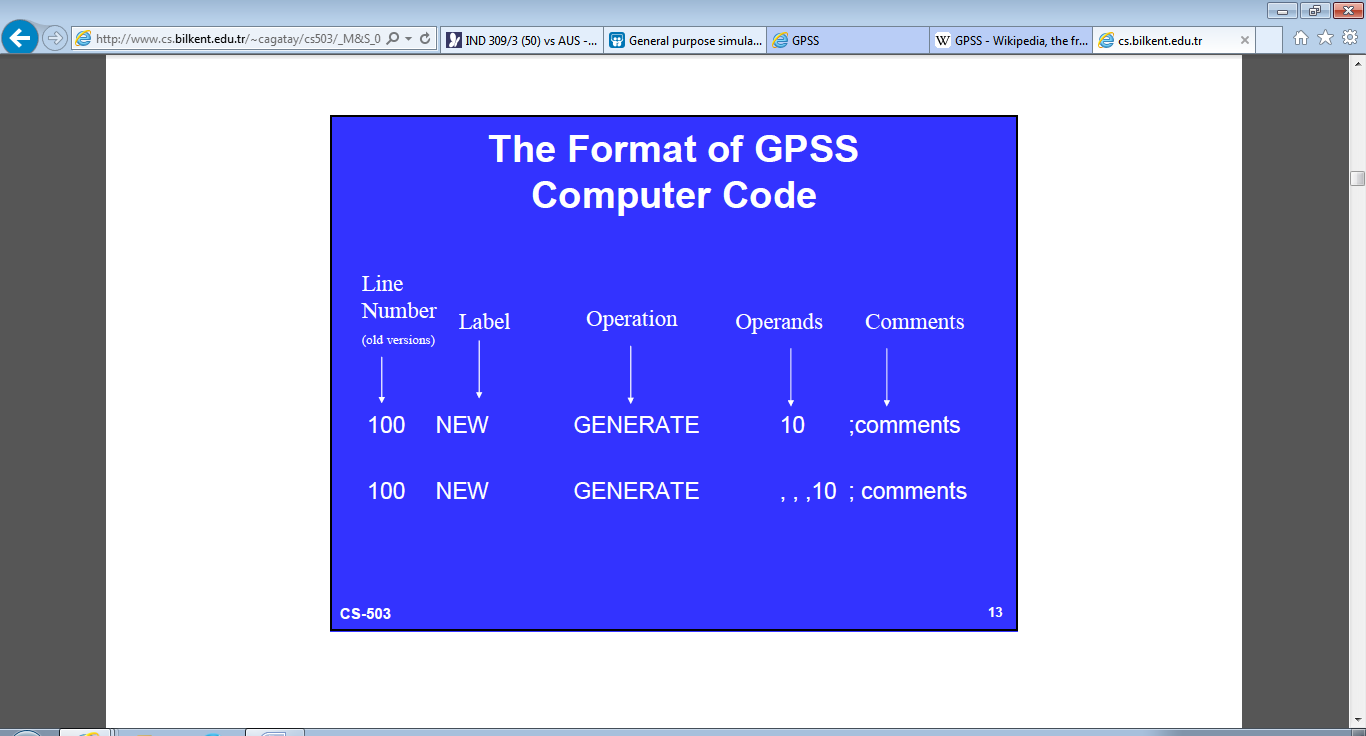
*line number label* TRANSFER A,B,C

ATTRIBUTES:

A = probability value (0 to 1)

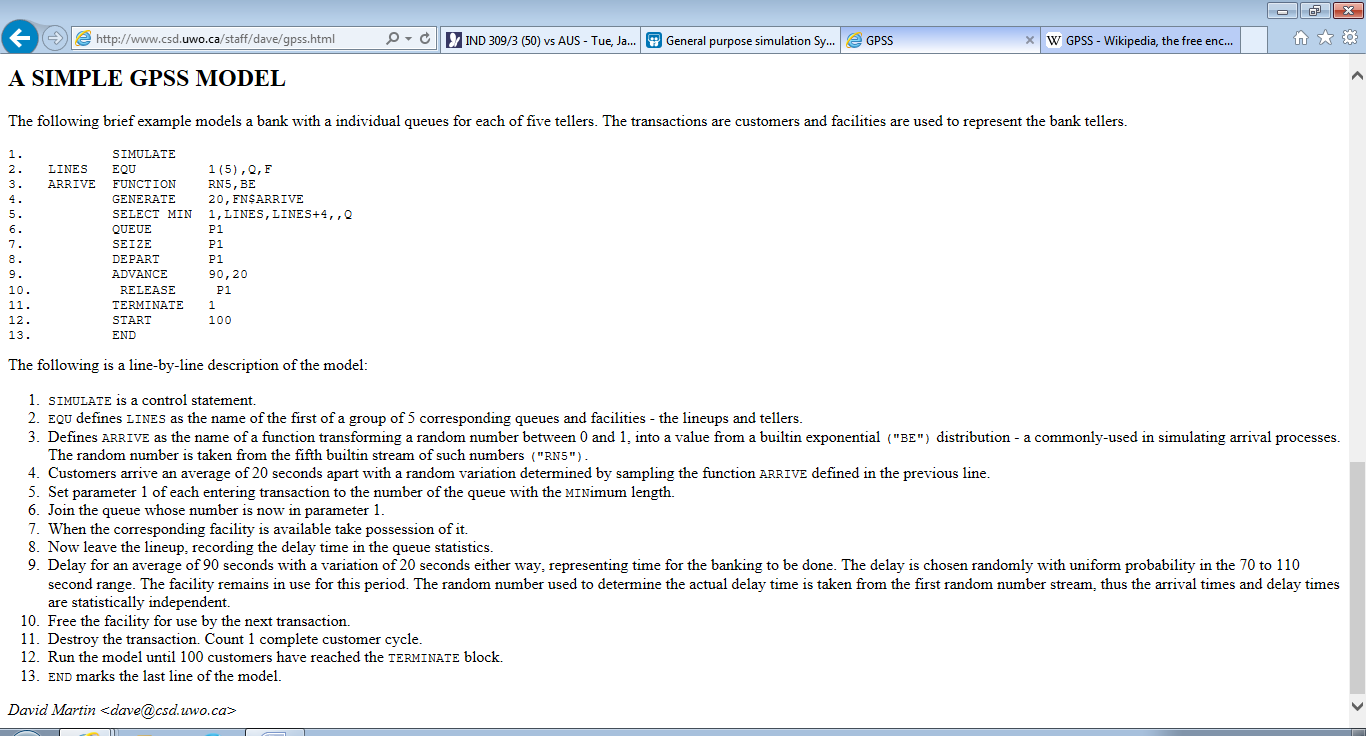
B = proportion of (1-A) transactions transferred to this labeled location

C = proportion A transactions transferred to this labeled location.



1. **SAMPLE CODE**

The following brief example models a bank with a individual queues for each of five tellers. The transactions are customers and facilities are used to represent the bank tellers.



The following is a line-by-line description of the model:

1. SIMULATE is a control statement.
2. EQU defines LINES as the name of the first of a group of 5 corresponding queues and facilities - the lineups and tellers.
3. Defines ARRIVE as the name of a function transforming a random number between 0 and 1, into a value from a built-in exponential ("BE") distribution - a commonly-used in simulating arrival processes. The random number is taken from the fifth built-in stream of such numbers ("RN5").
4. Customers arrive an average of 20 seconds apart with a random variation determined by sampling the function ARRIVES defined in the previous line.
5. Set parameter 1 of each entering transaction to the number of the queue with the Minimum length.
6. Join the queue whose number is now in parameter 1.
7. When the corresponding facility is available take possession of it.
8. Now leave the lineup, recording the delay time in the queue statistics.
9. Delay for an average of 90 seconds with a variation of 20 seconds either way, representing time for the banking to be done. The delay is chosen randomly with uniform probability in the 70 to 110 second range. The facility remains in use for this period. The random number used to determine the actual delay time is taken from the first random number stream, thus the arrival times and delay times are statistically independent.
10. Free the facility for use by the next transaction.
11. Destroy the transaction. Count 1 completes customer cycle.
12. Run the model until 100 customers have reached the TERMINATE block.
13. END marks the last line of the model.
14. **CONCLUSION**

GPSS is a programming system designed for the simulation of discrete systems. These are systems that can be modeled as a series of state changes over a period of time. The simulation technique uses numerical computation methods to follow the system elements through their changes of state.

1. **REFERENCES**
   1. *Greenberg, Stanley (1972). GPSS Primer. New York: Wiley-Interscience.*
   2. *Schriber, Thomas (1974). Simulation using GPSS. Wiley.*
   3. http://www.slideshare.net/drkredsight1/general-purpose-simulation-system-gpss
   4. http://www.ravi.kahlon.co/2013/02/explanation-of-some-gpss-blocks.html
   5. http://www.csd.uwo.ca/staff/dave/gpss.html