PROJECT TITLE: SIMPY AND GPSS

COIS 4470H WINTER 2024

MODELLING AND SIMULATION

INTRUCTOR NAME: WENYING FENG

SAMAKSH MONGA - 0721348

KRUTARTH GHUGE - 0746015

General Discussion

Specialized simulation languages are software programs created for the purpose of modeling and executing simulations of intricate systems. These languages have specific features designed for simulation modeling requirements, including support for modeling entities, events, and system components (Gordon, 1961). General Purpose Simulation System (GPSS) and SimPy are two well-known examples of special purpose simulation languages. GPSS was created in the late 1960s by Geoffrey Gordon at IBM (Gordon, 1961). Its main goal was to offer a specific instrument for emulating discrete event systems, especially within the realm of operations research and industrial engineering. GPSS was created to simplify the process for users to model and analyze intricate systems that include queues, resources, and stochastic processes. The language became popular in both academic and industrial settings because of its simplicity and effectiveness in representing various systems.

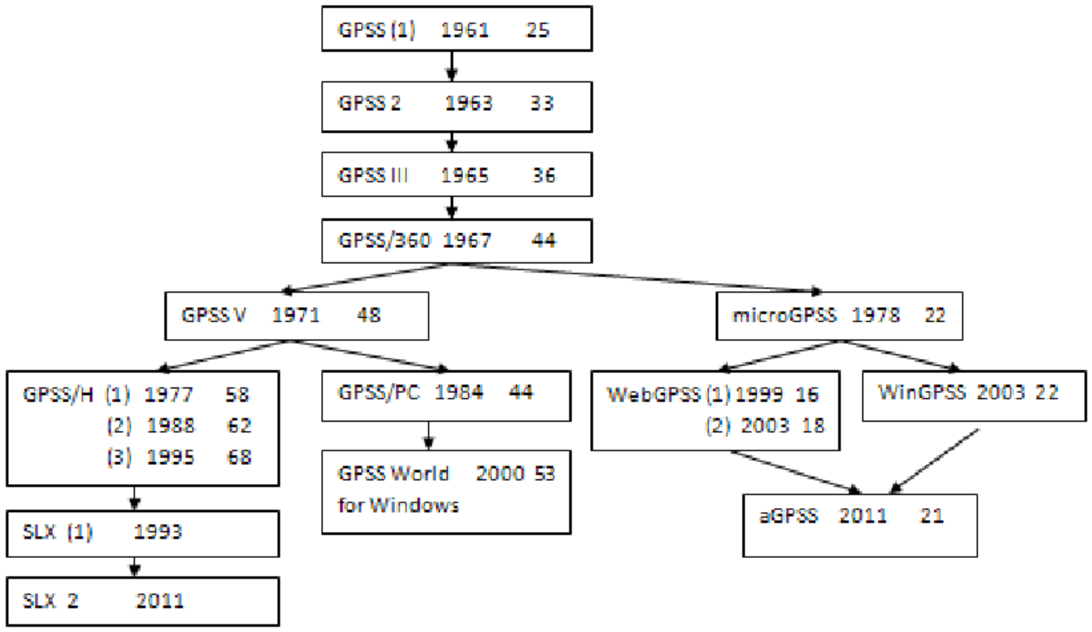
Meanwhile, SimPy is a newer inclusion in the realm of specialized simulation languages (Matloff, 2008). It is a Python-written open-source framework that was developed by Klaus Muller and initially launched in 2002. The increasing use of Python for scientific computing and simulation inspired the creation of SimPy. SimPy strives to offer a versatile and customizable framework for modeling discrete event systems, focusing on simplicity, modularity, and user-friendliness. Unlike GPSS, which exists as a language on its own, SimPy works together with Python, enabling users to make use of the various libraries and tools present in the Python ecosystem. The original purposes of GPSS and SimPy vary slightly because of their historical background and design principles. GPSS is mainly utilized for emulating systems that involve discrete occurrences, like production procedures, transportation systems, and computer networks. Its emphasis on simplicity and effectiveness makes it ideal for quickly creating models and evaluating discrete event systems. In contrast, SimPy is created as a versatile simulation platform capable of managing various simulation assignments. Its ability to adapt and grow makes it appropriate for representing intricate systems with complex interactions among parts.

GPSS and SimPy are two special-purpose simulation languages with distinct histories, design philosophies, and intended purposes. While GPSS is a standalone language focused on simulating discrete event systems, SimPy is an open-source framework integrated with Python, offering flexibility and extensibility for a wide range of simulation tasks.

Overview of GPSS and SimPy

General Purpose Simulation System (GPSS) and SimPy, two specialized simulation tools, are utilized across different fields to simulate and analyze intricate systems. It is crucial to comprehend their individual application domains and common use in simulation projects in order to efficiently employ these tools.

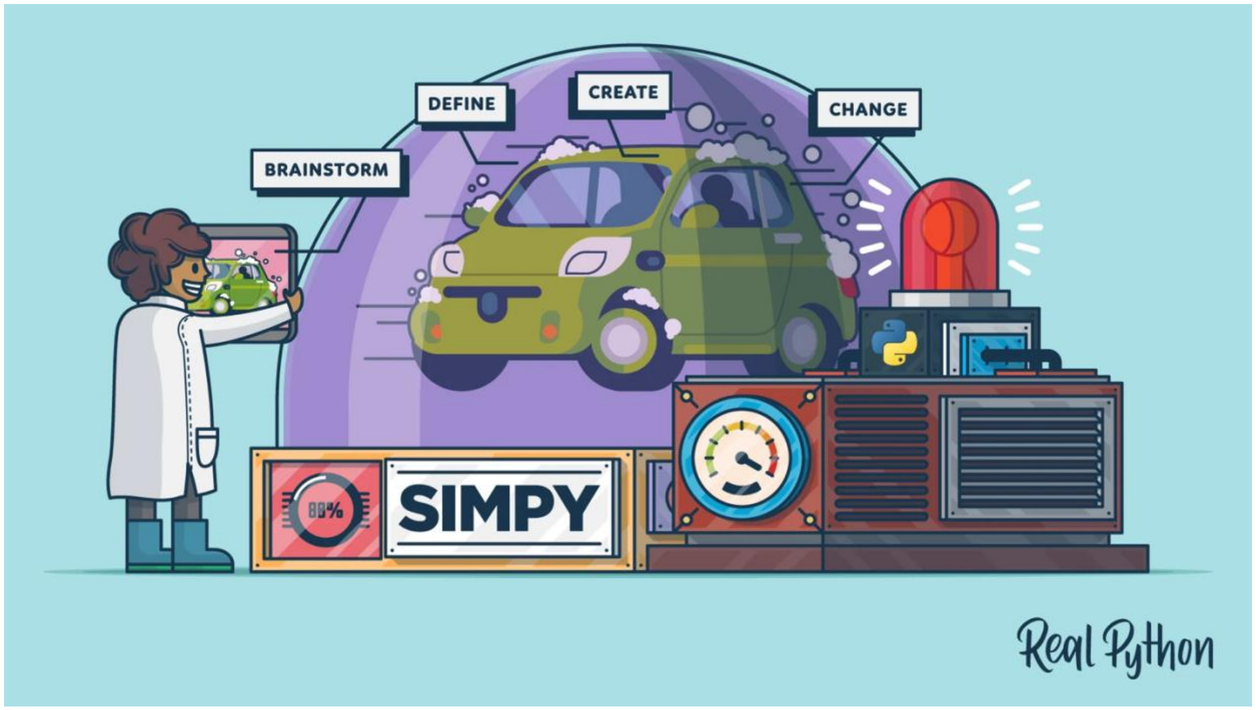
GPSS is widely utilized in industries like manufacturing, transportation, telecommunications, and computer systems. The main use of this technology is in modeling discrete event systems, which involve events happening at specific times and impacting the system's state. GPSS is utilized to simulate manufacturing processes like assembly lines and production facilities in order to optimize resource allocation, minimize bottlenecks, and enhance overall efficiency (Gordon, 1961). GPSS is utilized in transportation systems to replicate traffic flow, airport activities, and logistics networks for the purpose of evaluating performance, detecting possible problems, and suggesting remedies (Gordon, 1961). Moreover, GPSS is used in telecommunications to model call centers, network protocols, and data transmission systems for evaluating performance metrics like call waiting times, packet loss rates, and throughput (Gordon, 1961).



Genealogy of the GPSS systems

SimPy is a simulation framework that works flexibly with a wide range of uses because of its adaptability and ability to be extended. It is generally used in fields of operations research, healthcare, social sciences, and computer science - with applicability increasing constantly. In operations research, SimPy can be used in processes like queuing systems, inventory management, and supply chain logistics (Matloff, 2008). In the field of healthcare, SimPy is used to model patient movement within hospitals, the transmission of diseases among populations, and the effectiveness of treatment strategies to aid in decision-making and resource distribution (Matloff, 2008). Social scientists use SimPy for simulating agent-based systems, population dynamics, and social networks to analyze phenomena like opinion formation, diffusion of innovations, and collective behavior (Matloff, 2008). Furthermore, SimPy is utilized in computer science to model computer systems, network protocols, and distributed algorithms in order to assess performance, scalability, and reliability (Matloff, 2008).

In simulation endeavors, GPSS and SimPy are commonly utilized to simulate intricate systems, test varying scenarios, and evaluate system performance in different conditions. Users use the appropriate syntax and elements of GPSS or SimPy to determine the system's framework, such as entities, resources, events, and interactions. They define factors like arrival rates, service times, and resource capacities for setting up the simulation model. After defining the model, users conduct simulations to produce data and assess performance metrics like throughput, latency, utilization, and queue lengths to understand system behavior and make well-informed decisions (Fishman, 2013; Matloff, 2008).



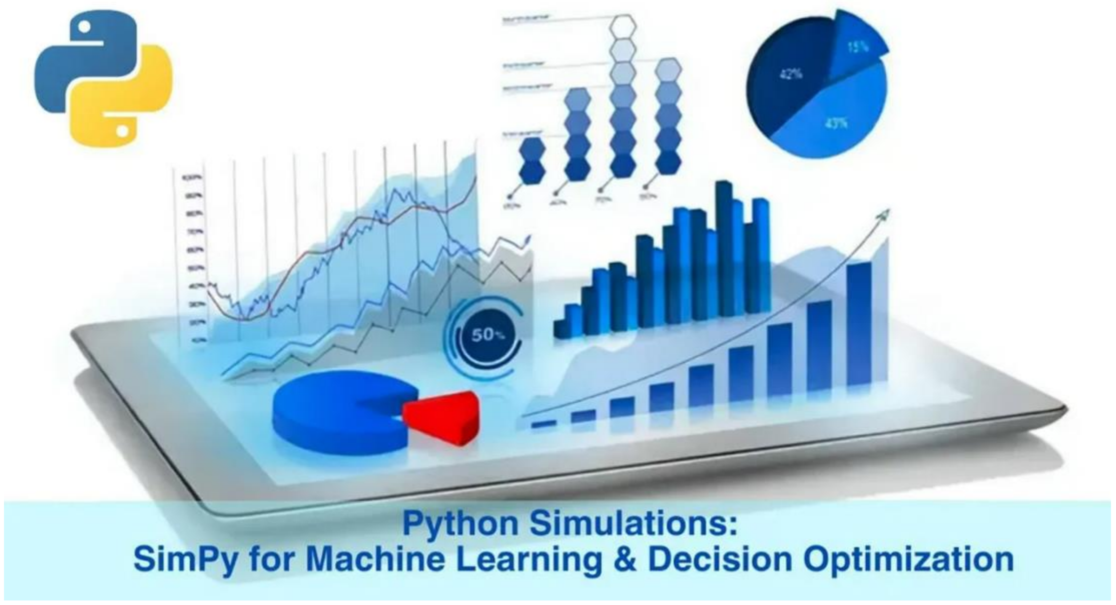
Built-in Features and Implementation

GPSS and SimPy both offer a range of built-in features for modeling and simulating systems, each having its own set of capabilities and methodologies.

GPSS includes inherent features for representing discrete event systems with elements like queues, transactions, and facilities. Law (2000) states that the syntax of the system is intuitive and concise, making it easy for users to define complex structures with little effort. GPSS has a diverse range of already defined functions and commands to handle entities, resources, and events, simplifying the process of modeling different system elements and interactions (Law, 2000). GPSS has a notable capability in supporting hierarchical modeling, enabling users to arrange simulation components into hierarchical structures for enhanced modularity and abstraction (Law, 2000). This capability allows for the representation of intricate systems with various levels of intricacy and abstraction, making it easier to examine how the system behaves at different levels of detail (Law, 2000). GPSS also provides pre-installed statistical functions that can be used to produce random numbers from popular probability distributions like exponential, uniform, and normal distributions (Law, 2000). These features allow users to imitate random processes and represent unknowns in system parameters, enhancing the authenticity of simulation outcomes (Law, 2000).

When it comes to implementation, GPSS simulations usually follow a sequential process-based approach, with entities progressing through the simulation model in specific time increments and engaging with resources and events based on predetermined rules (Law, 2000). This method makes it easier to create simulation models and simplifies the process of debugging and verifying simulation outcomes (Law, 2000).

SimPy provides a modeling approach based on processes, where system parts are portrayed as separate processes that communicate with each other via shared resources and events (Matloff, 2008). This enables more adaptable and descriptive modeling of intricate systems, since processes can be created, carried out, and ended dynamically according to system dynamics and user-defined logic (Matloff, 2008). One important aspect of SimPy is its capability to perform event-driven simulation, in which system events prompt the initiation of processes and the arrangement of forthcoming events (Matloff, 2008). This method of asynchronous modeling permits dynamic systems with unpredictable behavior to be simulated, including real-time systems and distributed systems (Matloff, 2008).



SimPy also provides built-in assistance for coordinating and synchronizing processes through synchronization tools like locks, semaphores, and events (Matloff, 2008). These characteristics allow users to create intricate synchronization patterns and scenarios involving sharing resources, like producer-consumer relationships and mutual exclusion (Matloff, 2008).

When implementing SimPy simulations, they are usually run with a discrete event simulation engine that schedules and processes events according to their timestamps (Matloff, 2008). This enables the effective simulation of extensive systems with millions of events by only handling active events at each simulation step (Matloff, 2008).

Effectiveness in Modeling Different Systems

The efficiency of modeling various systems with GPSS and SimPy varies based on the unique characteristics and needs of each system being modeled. GPSS is suitable for modeling systems that have discrete event dynamics and straightforward resource interactions, like manufacturing processes and queuing systems (Law, 2000). Law (2000) states that its hierarchical modeling strategy and pre-installed backing for typical probability distributions simplify the modeling and analysis of a diverse array of system behaviors.

In contrast, SimPy is highly skilled in simulating systems that have intricate relationships and changing characteristics, like real-time systems, distributed systems, and agent-based systems (Matloff, 2008). The modeling method is based on processes and the simulation engine is event-driven, allowing users to accurately depict complicated system dynamics and emergent occurrences (Matloff, 2008).

GPSS offers simplicity and efficiency for modeling discrete event systems, SimPy provides flexibility and expressiveness for modeling diverse systems with complex interactions and dynamics.

Input Parameters and Output Statistics

In GPSS, predefined commands and functions within the simulation model are commonly used to define input parameters. These factors consist of the frequency of arrivals, duration of service, number of resources available, and the likelihood of random events occurring. In a queuing system model, parameters like the average arrival rate of customers and the average service time of servers are defined through GPSS commands like GENERATE and ADVANCE (Law, 2000). According to Law (2000), GPSS includes pre-installed functions that can produce random values from popular probability distributions like exponential, uniform, and normal distributions. These features enable users to represent unknowns in system parameters and replicate random processes with authentic variations.
In the same way, SimPy offers output data for analyzing simulation outcomes, but users have greater freedom in specifying and customizing the metrics they wish to gather. SimPy users usually create their own simulation code to produce statistics, which can cover various measures like mean response times, usage of resources, and system capacity.

SimPy enables individuals to gather and combine simulation information by utilizing Python data structures like lists, dictionaries, and arrays. After gathering data, users are able to examine it by using common Python libraries and tools like NumPy, SciPy, and Matplotlib for statistical analysis and visualization (Matloff, 2008).

Relevance in Analyzing Simulation Results

The output data from GPSS and SimPy are important for analyzing simulation results and assessing the efficiency of simulated systems. These figures assist users in pinpointing bottlenecks, evaluating resource usage, and enhancing system efficiency and performance by tweaking system parameters.
To sum up, GPSS and SimPy provide input parameters and output statistics for simulating and analyzing results. Although GPSS comes with a pre-set simulation environment for defining

input parameters and producing output statistics, SimPy allows for greater customization and expansion by being compatible with Python and accommodating user-created simulation code.

Below is an example table showing the output statistics generated from a SimPy simulation of a queuing system:

Output Statistics

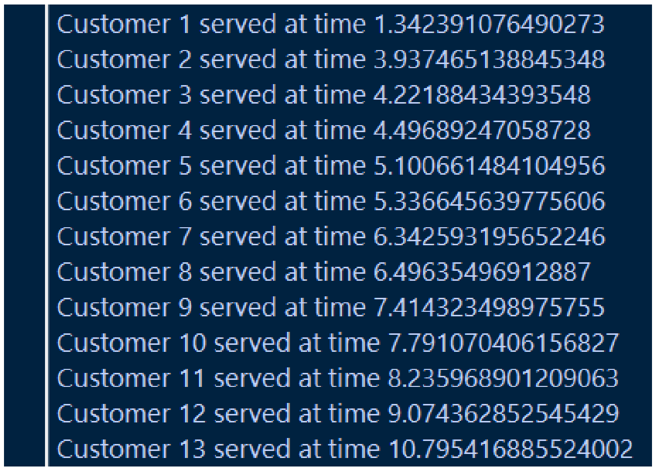
GPSS offers multiple output statistics like mean queue lengths, server utilization, and wait times. Typically, this data is produced during the simulation process and is displayed in tables or graphs within the GPSS simulation interface. For example, you have the option to create tables that display the mean waiting time for every customer in a simulation of a queuing system or graphs that illustrate the usage of servers throughout time (Kelton, Sadowski, & Swets, 2007).

In SimPy, output statistics are customizable based on user-defined simulation code. Users can collect and aggregate simulation data using Python data structures and libraries. For example, you can create tables or plots using Matplotlib to visualize simulation results, such as the distribution of customer waiting times in a SimPy-based queuing system simulation. This flexibility allows users to tailor output statistics to their specific needs and analyze simulation results in detail (Matloff, 2008).

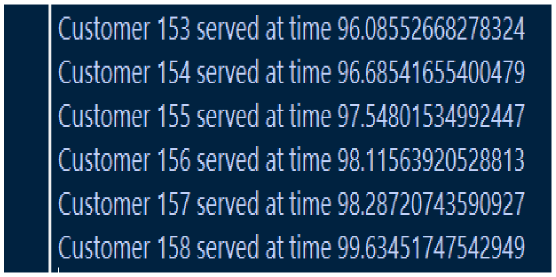
In conclusion, although GPSS and SimPy have differences in how they define input parameters and generate output statistics, they both offer necessary resources for modeling and analyzing simulation results across various application fields.

Demo (with Screenshots)

The given SimPy code replicates a queuing system, a typical situation found in different practical settings like customer service centers, manufacturing lines, and transportation systems. The simulation aims to examine how the queuing system's performance varies with changes in arrival and service rates.

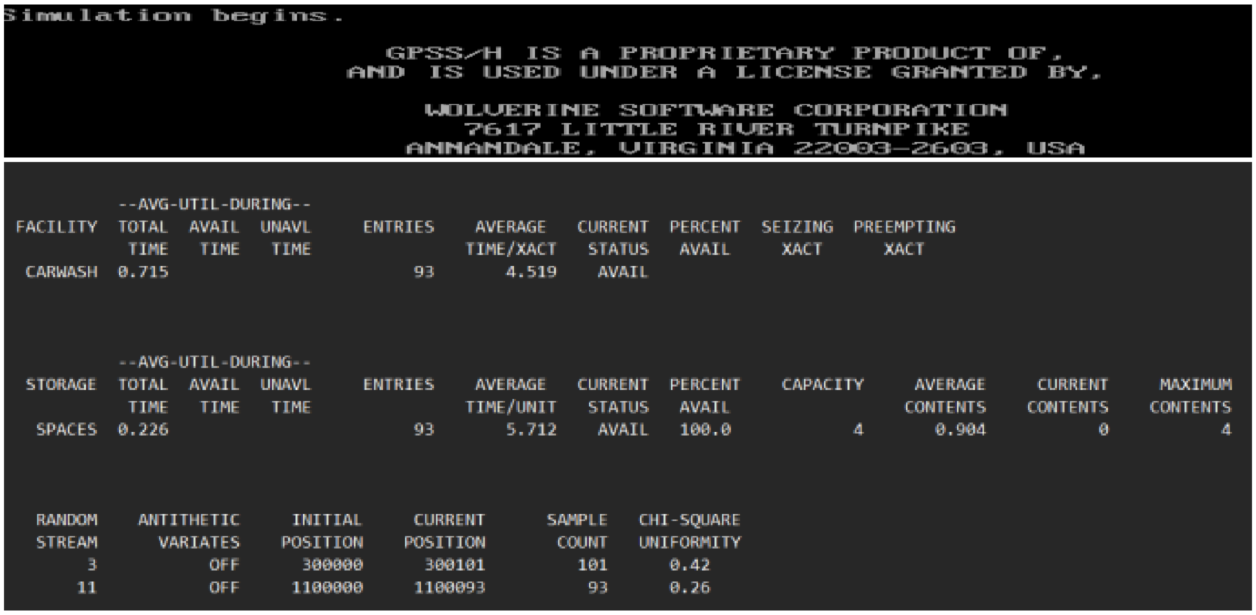


```
Customer 1 served at time 1.342391076490273
Customer 2 served at time 3.937465138845348
Customer 3 served at time 4.22188434393548
Customer 4 served at time 4.49689247058728
Customer 5 served at time 5.100661484104956
Customer 6 served at time 5.336645639775606
Customer 7 served at time 6.342593195652246
Customer 8 served at time 6.49635496912887
Customer 9 served at time 7.414323498975755
Customer 10 served at time 7.791070406156827
Customer 11 served at time 8.235968901209063
Customer 12 served at time 9.074362852545429
Customer 13 served at time 10.795416885524002
```



```
Customer 153 served at time 96.08552668278324
Customer 154 served at time 96.68541655400479
Customer 155 served at time 97.54801534992447
Customer 156 served at time 98.11563920528813
Customer 157 served at time 98.28720743590927
Customer 158 served at time 99.63451747542949
```

GPSS:
Carwash, and the testing-and-adjustment system and answer the questions indicated in the slides as follows: Carwash: Assume there are 4 waiting spaces.



```
Simulation begins.
                        GPSS/H IS A PROPRIETARY PRODUCT OF,
                      AND IS USED UNDER A LICENSE GRANTED BY,

                         WOLVERINE SOFTWARE CORPORATION
                           7617 LITTLE RIVER TURNPIKE
                        ANNANDALE, VIRGINIA 22003-2603, USA
```

```
          --AVG-UTIL-DURING--
FACILITY  TOTAL  AVAIL  UNAVL     ENTRIES    AVERAGE   CURRENT  PERCENT  SEIZING  PREEMPTING
           TIME   TIME   TIME               TIME/XACT   STATUS   AVAIL     XACT      XACT
 CARWASH  0.715                        93      4.519     AVAIL


          --AVG-UTIL-DURING--
 STORAGE  TOTAL  AVAIL  UNAVL     ENTRIES    AVERAGE   CURRENT  PERCENT   CAPACITY     AVERAGE     CURRENT     MAXIMUM
           TIME   TIME   TIME               TIME/UNIT   STATUS   AVAIL                CONTENTS    CONTENTS    CONTENTS
  SPACES  0.226                        93      5.712     AVAIL   100.0           4       0.904           0           4


  RANDOM    ANTITHETIC     INITIAL      CURRENT      SAMPLE   CHI-SQUARE
  STREAM      VARIATES    POSITION     POSITION       COUNT   UNIFORMITY
       3           OFF      300000       300101         101       0.42
      11           OFF     1100000      1100093          93       0.26
```

Reviews

GPSS

ADVANTAGES	**DISADVANTAGES**
GPSS is renowned for its ease of use and effectiveness in simulating discrete event systems. The software provides a friendly interface and built-in tools for modeling queues, resources, and events. Moreover, GPSS's method of hierarchical modeling enables the simple portrayal of intricate systems that contain various levels of detail (Law, 2000).	GPSS might not have the necessary adaptability and expandability for simulating intricate systems with changing dynamics. Law (2000) states that its syntax and functionality are customized for discrete event simulation, restricting its usefulness for other simulation types.

SimPy

ADVANTAGES	**DISADVANTAGES**
Its integration with Python provides users with flexibility and extensibility, enabling them to utilize Python's wide range of libraries and tools. It is capable of process-based modeling and event-driven simulation, making it appropriate for various simulation tasks such as discrete event simulation, agent-based modeling, and system dynamics (Matloff, 2008).	Users who are not experienced in Python programming may find that SimPy has a more challenging learning curve. Beginners may struggle with the requirements of programming experience and a basic grasp of simulation concepts, as stated by Matloff in 2008.

Similarities:

- GPSS and SimPy are simulation languages specifically created for modeling intricate systems.
- They offer functions for creating simulation models, generating random variables, and gathering output data.
- Both languages strive to make modeling and simulating systems easier in order to aid in analysis and decision-making.

Differences:

- GPSS operates independently with a visual interface, whereas SimPy is fused with Python and utilizes Python's coding abilities.
- GPSS mainly concentrates on discrete event simulation, while SimPy covers a wider variety of simulation approaches such as process-based modeling and event-driven simulation.

- GPSS offers a hierarchical modeling approach, whereas SimPy offers flexibility with its object-oriented design and modular architecture.

GPSS and SimPy offer further understanding of their advantages and disadvantages. Some users appreciate GPSS for its simplicity and user-friendliness, while others complain about its lack of advanced modeling techniques and limited functionality (Kelton, Sadowski, & Swets, 2007). Likewise, SimPy has been praised for its adaptability and ability to be expanded, however, there are worries about its efficiency and ability to handle large simulations (Matloff, 2008).

References

- Gordon, G. (1961). GPSS—A new tool for systems simulation. Proceedings of the 1961 16th ACM National Meeting, 175–185 https://dl.acm.org/doi/10.5555/1513605.1513781'
- Matloff, N. (2008). SimPy: A Tutorial Introduction. https://simpy.readthedocs.io/en/latest/simpy_intro/index.html
- Fishman, G. S. (2013). Discrete-event simulation: modeling, programming, and analysis. Springer Science & Business Media https://www.scirp.org/reference/referencespapers?referenceid=2105619
- Matloff, N. (2008). SimPy: A Tutorial Introduction https://simpy.readthedocs.io/en/latest/simpy_intro/index.html
- A.M. Law (2014). Simulation Modelling and Analysis https://www.researchgate.net/publication/31639535_Simulation_Modeling_and_Analysis_AM_Law_WD_Kelton
- Kumar, G. (2023). Introduction to SimPy (simulation in python)-01, Medium https://gaurav-adarshi.medium.com/introduction-to-simpy-simulation-in-python-01-97b05d1376b9
- (n.d.) Figure 1: Genealogy of the GPSS systems https://www.researchgate.net/figure/Genealogy-of-the-GPSS-systems_fig2_261349030
- J. Zhane (n.d.). SimPy: Simulating Real-World Processes with Python https://realpython.com/simpy-simulating-with-python/