Process Simulation using Python

Dominique Theodore

April 2023

Agenda

- What are discrete event simulations (DES)?
- Use cases
- What is SimPy?
- Generators in Python
- Basic SymPy concepts
- Examples
- DES using simply_helpers

What is a simulation?

- a replication of a real-world process or event in an environment that is isolated or disconnected from its real-world counterpart.
- 3 broad types of simulations
  - discrete: state changes in response to events, typically using next-event time progression
  - continuous: physical systems with material objects moving in space
  - stochastic (Monte-Carlo): a system is simulated using variables that change stochastically with individual propbabilities

What is discrete event simulation?

- a method used to model real world systems that can be decomposed into a set of logically separate processes that autonomously progress through time
- DES models complex systems as things going though processes in discrete time
- DES assumes events happen at specified intervals

Use cases

- Queuing theory: making decisions about how many resources we need to provide a service
- Manufacturing (How many machines and workers will we need?)
- Healthcare (How many nurses will we need?)
- Call center services (how many phone lines will we need?)
- Military applications (how many tanks will we need?)
- Organising outpatient dialysis services during the COVID-19 pandemic https://www.researchgate.net/publication/343648345_A_simulation_modelling_toolkit_for_organising_outpatient_dialysis_services_during_the_COVID-19_pandemic

What is Simpy?

- a process-based discrete-event simulation framework based on standard Python
- uses Python generator functions under the hood to model active components such as customers, vehicles, messages etc
- describes the flow of entities through a complex system using:
- processes that generate entities: customers, messages, packets, patients
- resources e.g servers, nurses, emergency rooms to model limited capacity congestion points

Generators in Python

Normal function

```
def myFunction():
    return 20
```

- `return` returns a function and terminates execution

Generator function

```
def myGenerator():
    yield 10
    yield 20
```

- `yield` returns a value and pauses execution while maintaining internal states

Basic SimPy concepts

- use processes to model the behaviour of active components (vehicles, patients, widgets)
- processes are described using Python generators
- processes create events and yield them in order to wait for them to be triggered
- An important event type is the Timeout. Events of this type are triggered after a certain amount of (simulated) time has passed.

A simple process

```
>>> def car(env):
...     while True:
...         print('Start parking at %d' % env.now)
...         parking_duration = 5
...         yield env.timeout(parking_duration)
...
...         print('Start driving at %d' % env.now)
...         trip_duration = 2
...         yield env.timeout(trip_duration)
```

simpy_helpers

- Simpy is tough!
- `simpy_helpers` makes it easier to build simulations using Simpy
- Provides 4 main classes:
  - Entity
  - Resource
  - Source
  - Stats

Resources and recommended reading

- https://simpy.readthedocs.io/en/latest/
- https://www.youtube.com/watch?v=TALKZZV0TiU&t=1634s
- https://www.youtube.com/watch?v=jXDjrWKcu6w&t=5836s
- https://github.com/bambielli/simpy_helpers
- https://campus.datacamp.com/courses/discrete-event-simulation-in-python/introduction-to-dynamic-systems-and-discrete-event-simulation-models?ex=8