Diskrétní simulace za použití knihovny SimPy

Panovský Tomáš

21. října 2025

1 Úvod

V této bakalářské práci se zabývám diskrétní simulací za použití knihovny SimPy v Pythonu. Knihovna Simpy umožňuje modelovat procesy, jenž probíhají souběžně, a mohou být zastaveny, nebo pozastaveny na určitou dobu. Praktická část bakalářské práce obsahuje aplikaci simulující průběh hudebního festivalu. Cílem simulace je zjistit, jak se návštěvníci pohybují a kde vznikají fronty, což může pomoci při organizaci reálného festivalu.

2 Systém

str17 V rámci této bakalářské práce budeme reálné jevy a procesy, které chceme zkoumat a analyzovat, označovat jako „systém“. Může jít například o zkoumání pohybu a chování lidí na hudebním festivalu, kde „systémem“ rozumíme souhrn všech prvků a pravidel, které tento proces ovlivňují. Systémy lze rozdělit na diskrétní a spojité.

2.1 Diskrétní systém

Diskrétní systém je takový, u kterého se stavová proměnná mění pouze v diskrétních časových okamžicích. Příkladem diskrétního systému je banka. Stavová proměnná, tedy počet zákazníků v bance, se mění pouze tehdy, když zákazník přijde, nebo když je dokončena obsluha zákazníka.

2.2 Spojitý systém

Spojitý systém je naopak takový, u kterého se stavová proměnná mění plynule v čase. Příkladem je hladina vody za přehradou. Během a krátce po dešti do nádrže za přehradou přitéká voda. Voda je také odebírána pro protipovodňovou ochranu a výrobu elektřiny. Odpařování dále snižuje hladinu vody.

3 Úvod do simulace

str6 Simulace je napodobení činnosti reálného jevu (systému) v čase. Cílem je vytvořit umělou historii daného systému, a následně pozorovat tuto uměle vytvořenou historii za účelem vyvození závěrů o provozních vlastnostech reálného systému. Chování systému vyvíjejícího se v čase se zkoumá pomocí simulačního modelu. V této práci je simulační model realizován formou aplikace vytvořené v knihovně SimPy. Simulační model poté může být použit k prozkoumání široké škály otázek typu „co by se stalo, kdyby“ týkajících se reálného systému. Například můžeme zkoumat, zda je daný počet stánků s občerstvením dostatečný pro obsloužení všech návštěvníků festivalu bez dlouhých front. Možné změny systému pak lze nejprve nasimulovat, aby bylo možné předpovědět jejich dopad na výkonnost systému. Simulace může být také použita ke studiu systémů ve fázi návrhu, ještě před jejich samotnou realizací.

3.1 Kdy je simulace vhodná

str7 Simulace je vhodná zejména v případech, kdy je potřeba detailně zkoumat vnitřní interakce složitých systémů nebo jejich částí. Díky možnosti měnit vstupní parametry a pozorovat výstupy se dají identifikovat klíčové proměnné a vztahy mezi nimi. Simulace také slouží jako efektivní nástroj pro výuku a demonstraci analytických metod. Simulace také umožňuje získávat data ze simulovaného systému v bezpečném prostředí bez nákladů a rizik spojených s reálným provozem. Pomocí animací lze také lépe vizualizovat fungování systému. U velmi složitých moderních systémů je simulace často jediným praktickým nástrojem, jak porozumět jejich vnitřním interakcím.

3.2 Kdy simulaci nepoužívat

str8 Existují situace, kdy simulace není vhodná. Pokud je problém řešitelný jednoduchým rozumovým odhadem nebo základním výpočtem. V některých případech je navíc levnější provést přímý experiment než složitou simulaci. Pokud náklady na provedení simulace převyšují možné úspory nebo přínosy, simulace není ekonomicky výhodná. Důležitým faktorem je také schopnost model validovat a ověřit, protože bez dostatečné validace nemůžeme výsledkům simulace plně důvěřovat.

4 Diskrétní simulace

str19 Diskrétní simulace systémů je založená na událostech, což znamená modelování takových systémů, ve kterých se stavové proměnné mění pouze v určitých, diskrétních časových okamžicích – tedy v momentech, kdy nastane konkrétní událost. Typickým příkladem může být příchod návštěvníka na festival nebo jeho odbavení u vstupu. Mezi těmito událostmi zůstává systém beze změny. Na rozdíl od analytických metod, které využívají matematické rovnice k nalezení řešení, jsou diskrétní

simulační modely obvykle řešeny numerickými metodami – tedy „spuštěním“ modelu na počítači. Simulace generuje umělou historii systému na základě předpokládaných pravidel a vstupů.

5 Simulace v SimPy

Simpy je Python knihovna pro diskrétní simulaci událostí a procesů. Procesy jsou v SimPy definovány pomocí Pythonových generátorových funkcí, ty lze využít pro simulaci aktivních entit, jako jsou například zákazníci, vozidla nebo třeba zvířata. SimPy také poskytuje různé typy sdílených zdrojů pro modelování bodů s omezenou kapacitou. Simulace lze provádět „co nejrychleji“, v reálném čase, nebo ručně krokovat jednotlivé události.

6 Příklad metody

Pro ilustraci uvádím metodu `go_to_festival_area()`, která simuluje průchod návštěvníka vstupním turniketem do areálu festivalu.

```
def go_to_festival_area(self, entrances):

    yield self.festival.timeout(random.expovariate(1/5))
    entrance_id = occupied.index(min(occupied))
    entrance = entrances[entrance_id]
    occupied[entrance_id] += 1

    with entrance.request() as req:
        queue_start = self.festival.now
        yield req

        queue_waiting_time = self.festival.now - queue_start
        entry_time = random.uniform(1, 3)
        yield self.festival.timeout(entry_time)

    self.state["location"] = resources.Location.FESTIVAL_AREA
    occupied[entrance_id] -= 1
```

Metoda `go_to_festival_area()` simuluje vstup návštěvníka jedním z dostupných vstupů. Metoda přijímá dva argumenty, a to instanci návštěvníka - self, a SimPy resource entrances. SimPy resource jsou objekty, ke kterým návštěvníci v simulaci přistupují, entrances je tedy seznam vstupů, mezi kterými si návštěvník může vybrat.

Každý návštěvník dorazí po náhodném zpoždění, které je generováno pomocí `random.expovariate(1/5)`, což modeluje příchody návštěvníků s průměrem 5 časových jednotek. Metoda využívá k simulování zastavení času pro návštěvníka funkci `yield self.festival.timeout()`, která na počet zadaných časových jednotek danou instanci uspí.

Návštěvník si vybere vstup, u kterého je aktuálně nejméně lidí čekajících ve frontě, což je sledováno pomocí globálního seznamu `occupied`. Následně požádá o přístup ke vstupu pomocí `entrance.request()`. Proměnná `queue_waiting_time` uchovává dobu, po kterou návštěvník čekal ve frontě, než se dostal „na řadu“. Po získání přístupu stráví návštěvník náhodnou dobu kontolou během vstupu, simulovanou pomocí `random.uniform(1, 3)` a `yield self.festival.timeout(entry_time)`. Nakonec se počet lidí u daného vstupu sníží, což znamená, že návštěvník dokončil průchod vstupem, a nastaví se mu atribut „location“ na FESTIVAL_AREA.

Tento přístup umožňuje sledovat délku front a čekací doby u jednotlivých vstupů, což poskytuje užitečné informace pro analýzu průchodnosti a vytíženosti vstupů během festivalu.