PCTMC

April 27, 2022

1 Population CTMC (PCTMC)

Consider a CTMC model of a population in which each of N individuals can be in a state of state space S. Firing rate may depends on the density of individuals in certain states.

Use the classes developed in the notebook `PCTMC_methods` to define an instance of a PCTMC model. In order to correctly define a PCTMC model, one should define: - state variables, - rate parameters, - initial state x_0, - system size, - transitions: update vectors and propensity functions.

[1]:
```
# Terminal:
# pip install ipynb
# pip install sympy

from ipynb.fs.full.PCTMC_methods import *
from time import perf_counter as timer
```

pctmc = Model() #### Syntax to add variables: pctmc.add_variable("A", initial_value) #### Syntax to add parameters: pctmc.add_parameter("k", param_value) #### Set the dimension of the population: pctmc.set_system_size("N",population_size) #### Syntax to add transitions: pctmc.add_transition({"A": -1, "B": +1}, "rate*A") –> use symbolic expressions for propensity functions

Remember to **finalize** the initialization: pctmc.finalize_initialization()

Below you can see an example for the SIR epidemic model.

[2]:
```
def epidemic_model(population_size):
    sir = Model()
    # variables are generated in the following order
    sir.add_variable("S", 0.99*population_size)
    sir.add_variable("I", 0.01*population_size)
    sir.add_variable("R", 0.0*population_size)
    # Adding parameters
    sir.add_parameter("ki", 2)
    sir.add_parameter("kr", 1)
    sir.add_parameter("ke", 1)
    sir.add_parameter("ks", 3)
    #setting the system size N
    sir.set_system_size("N",population_size)
    # Adding transitions, using a dictionary to represent the update vector
    sir.add_transition({"S":-1, "I":1},  "ke*S + ki/N*I*S")
    sir.add_transition({"I":-1, "R":1},  "kr*I")
    sir.add_transition({"R":-1, "S":1},  "ks*R")
    #finalize initialization
    sir.finalize_initialization()
    return sir
```

1.0.1 Predator-Prey model

Lotka-Volterra model. Species are two: Predators and Preys.

Possible events are: preys reproduce, predators eat preys and reproduce, predators die off.

[3]:
```
def lotka_volterra():
    lv = Model()
    # Adding variables

    # Adding parameters

    # setting the system size N

    # Adding transitions, using a dictionary to represent the update vector

    # adding observables

    # remember to finalize
    lv.finalize_initialization()

    return lv
```

1.0.2 Genetic Toggle Switch

Model the following system as a PCTMC. There are two genes G_1 and G_2, that can be in two states: either on or off. When gene G_i is on, it produces a protein P_i that can inhibit the expression of the other gene.

Species are $\{G_1^{on}, G_2^{on}, G_1^{off}, G_2^{off}, P_1, P_2\}$, such that $G_i^{on} + G_i^{off}$ is constant. The inhibition event is modeled as the binding of protein P_1 (P_2) with gene G_2^{off} (G_1^{off}). The proteins also unbind and degradate according to a given rate.

[ ]:
```

```

1.1 Stochastic Simulation - SSA algorithm

Look into the class Simulator() and complete function `_SSA_single_simulation` with the ingredients needed to perform SSA simulation of stochastic trajectories.

Plot the stochastic trajectories.

```
[7]:
```

1.2 Stochastic Approximation

Consider the PCTMC models defined above and implement the **Mean Field** (MF) and the **Linear Noise approximation** (LNA) of the stochastic evolution of the system.

Add the methods *MF_simulation()* and *LN_simulation* in the Simulator class. The overall structure of the solution is already in place, fill the gaps. Some methods in class Model() have to be completed as well.

Symbolic representation is a key ingredient for the solution, so look at the Simpy documentation if needed.

Plot the deterministic trajectories of the MF approximation and the confidence interval given by the LNS.

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```