psub_map_label

March 10, 2021

1 Referencing Partial State Update Blocks labels to substeps in cadCAD

Vitor Marthendal Nunes

This notebook shows how to use label metadata on PSUBs to do post-processing on the simulation. We use the key `label` as metadata to the partial state update blocks, so it's possible to map the substeps order to the PSUBs label. The used prey and predator model was taken from `minimal_prey_predator.ipynb`

1.1 Dependences

[1]:
```
%%capture
!pip install cadcad
```

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from cadCAD.configuration import Experiment
from cadCAD.configuration.utils import config_sim
from cadCAD.engine import ExecutionMode, ExecutionContext, Executor
```

1.2 Definitions

1.2.1 Initial conditions and parameters

[3]:
```
initial_conditions = {
    'prey_population': 100,
    'predator_population': 15
    }

params = {
    "prey_birth_rate": [1.0],
    "predator_birth_rate": [0.01],
    "predator_death_const": [1.0],
    "prey_death_const": [0.03],
```

```
    "dt": [0.1] # Precision of the simulation. Lower is more accurate / slower
}

simulation_parameters = {
    'N': 1,
    'T': range(200),
    'M': params
}
```

1.2.2 Policies

[4]:
```
def p_predator_births(params, step, sL, s):
  dt = params['dt']
  predator_population = s['predator_population']
  prey_population = s['prey_population']
  birth_fraction = params['predator_birth_rate'] + np.random.random() * 0.0002
  births =  birth_fraction * prey_population * predator_population * dt
  return {'add_to_predator_population': births}


def p_prey_births(params, step, sL, s):
  dt = params['dt']
  population = s['prey_population']
  birth_fraction = params['prey_birth_rate'] + np.random.random() * 0.1
  births =  birth_fraction * population * dt
  return {'add_to_prey_population': births}


def p_predator_deaths(params, step, sL, s):
  dt = params['dt']
  population = s['predator_population']
  death_rate = params['predator_death_const'] + np.random.random() * 0.005
  deaths = death_rate * population * dt
  return {'add_to_predator_population': -1.0 * deaths}


def p_prey_deaths(params, step, sL, s):
  dt = params['dt']
  death_rate = params['prey_death_const'] + np.random.random() * 0.1
  prey_population = s['prey_population']
  predator_population = s['predator_population']
  deaths = death_rate * prey_population * predator_population * dt
  return {'add_to_prey_population': -1.0 * deaths}
```

1.2.3 State update functions

[5]:
```
def s_prey_population(params, step, sL, s, _input):
    y = 'prey_population'
    x = s['prey_population'] + _input['add_to_prey_population']
    return (y, x)


def s_predator_population(params, step, sL, s, _input):
    y = 'predator_population'
    x = s['predator_population'] + _input['add_to_predator_population']
    return (y, x)
```

1.2.4 State update blocks

[6]:
```
partial_state_update_blocks = [
    {
        'label': 'Predator dynamics',
        'policies': {
            'predator_births': p_predator_births,
            'predator_deaths': p_predator_deaths
        },
        'variables': {
            'predator_population': s_predator_population
        }
    },
    {
        'label': 'Prey dynamics',
        'policies': {
            'prey_births': p_prey_births,
            'prey_deaths': p_prey_deaths
        },
        'variables': {
            'prey_population': s_prey_population
        }
    }
]
```

1.2.5 Configuration and Execution

[7]:
```
sim_config = config_sim(simulation_parameters)

exp = Experiment()
exp.append_configs(sim_configs=sim_config,
                   initial_state=initial_conditions,
                   partial_state_update_blocks=partial_state_update_blocks)
```

```
from cadCAD import configs
exec_mode = ExecutionMode()
exec_context = ExecutionContext(exec_mode.local_mode)
executor = Executor(exec_context=exec_context, configs=configs)
(records, tensor_field, _) = executor.execute()
```

```
                  ___________    ____
  ________ __ ___/ / ____/   |  / __ \
 / ___/ __` / __  / /   / /| | / / / /
/ /__/ /_/ / /_/ / /___/ ___ |/ /_/ /
\___/\__,_/\__,_/\____/_/  |_/_____/
by cadCAD

Execution Mode: local_proc
Configuration Count: 1
Dimensions of the first simulation: (Timesteps, Params, Runs, Vars) = (200, 5,
1, 2)
Execution Method: local_simulations
SimIDs   : [0]
SubsetIDs: [0]
Ns       : [0]
ExpIDs   : [0]
Execution Mode: single_threaded
Total execution time: 0.02s
```

1.2.6 Results

[8]:
```
df = pd.DataFrame(records)
df
```

[8]:

	prey_population	predator_population	simulation	subset	run	substep	timestep
0	100.000000	15.000000	0	0	1	0	0
1	100.000000	15.008646	0	0	1	1	0
2	98.836472	15.008646	0	0	1	2	0
3	98.836472	14.994031	0	0	1	1	0
4	98.116429	14.994031	0	0	1	2	0
..	…	…	…	…	…	…	…
396	83.174488	7.365290	0	0	1	2	0
397	83.174488	7.241514	0	0	1	1	0
398	90.051579	7.241514	0	0	1	2	0
399	90.051579	7.181079	0	0	1	1	0
400	97.158694	7.181079	0	0	1	2	0

```
1           1
2           1
3           2
4           2
..        …
396       198
397       199
398       199
399       200
400       200

[401 rows x 7 columns]
```

[9]:
```
# Mapping the substep order to the PSUB label
psubs = partial_state_update_blocks
psub_map = {order+1: psub['label'] for (order, psub) in enumerate(psubs)}
```

[10]:
```
df['psubs'] = df.substep.map(psub_map)
df
```

[10]:
```
     prey_population  predator_population  simulation  subset  run  substep  \
0         100.000000            15.000000           0       0    1        0
1         100.000000            15.008646           0       0    1        1
2          98.836472            15.008646           0       0    1        2
3          98.836472            14.994031           0       0    1        1
4          98.116429            14.994031           0       0    1        2
..               …                    …         …     …  …       …
396        83.174488             7.365290           0       0    1        2
397        83.174488             7.241514           0       0    1        1
398        90.051579             7.241514           0       0    1        2
399        90.051579             7.181079           0       0    1        1
400        97.158694             7.181079           0       0    1        2

     timestep              psubs
0           0                NaN
1           1  Predator dynamics
2           1      Prey dynamics
3           2  Predator dynamics
4           2      Prey dynamics
..        …                  …
396       198      Prey dynamics
397       199  Predator dynamics
398       199      Prey dynamics
399       200  Predator dynamics
400       200      Prey dynamics

[401 rows x 8 columns]
```

1.2.7 Filtering the results by the PSUB labels

[11]:
```
df.query("psubs=='Predator dynamics'")
```

[11]:

	prey_population	predator_population	simulation	subset	run	substep	timestep	psubs
1	100.000000	15.008646	0	0	1	1	1	Predator dynamics
3	98.836472	14.994031	0	0	1	1	2	Predator dynamics
5	98.116429	14.979513	0	0	1	1	3	Predator dynamics
7	100.014331	14.987678	0	0	1	1	4	Predator dynamics
9	97.323763	14.949892	0	0	1	1	5	Predator dynamics
..	…	…	…	…	…	…	…	…
391	72.048785	7.722693	0	0	1	1	196	Predator dynamics
393	73.389519	7.522422	0	0	1	1	197	Predator dynamics
395	79.218778	7.365290	0	0	1	1	198	Predator dynamics
397	83.174488	7.241514	0	0	1	1	199	Predator dynamics
399	90.051579	7.181079	0	0	1	1	200	Predator dynamics

[200 rows x 8 columns]

[12]:
```
df.query("psubs=='Prey dynamics'")
```

[12]:

	prey_population	predator_population	simulation	subset	run	substep	timestep	psubs
2	98.836472	15.008646	0	0	1	2	1	Prey dynamics
4	98.116429	14.994031	0	0	1	2	2	Prey dynamics
6	100.014331	14.979513	0	0	1	2	3	Prey dynamics
8	97.323763	14.987678	0	0	1	2	4	Prey dynamics
10	97.906393	14.949892	0	0	1	2	5	Prey dynamics
..	…	…	…	…	…	…	…	…
392	73.389519	7.722693	0	0	1	2	196	Prey dynamics
394	79.218778	7.522422	0	0	1	2	197	Prey dynamics
396	83.174488	7.365290	0	0	1	2	198	Prey dynamics
398	90.051579	7.241514	0	0	1	2	199	Prey dynamics
400	97.158694	7.181079	0	0	1	2	200	Prey dynamics

```
     timestep          psubs
2           1  Prey dynamics
4           2  Prey dynamics
6           3  Prey dynamics
8           4  Prey dynamics
10          5  Prey dynamics
..        …           …
392       196  Prey dynamics
394       197  Prey dynamics
396       198  Prey dynamics
398       199  Prey dynamics
400       200  Prey dynamics

[200 rows x 8 columns]
```