PhotovoltaicCell_BuckConversor

March 9, 2021

1 Expirence 2

2 Introduction

2.1 Differential Equations

$$\frac{di_1(t)}{dt} = D(t)\frac{v_1(t)}{L} - \frac{v_0(t)}{L} \tag{1}$$

$$\frac{dv_0(t)}{dt} = D(t)\frac{i_1(t)}{C} - \frac{v_0(t)}{RC} \tag{2}$$

2.1.1 First equation in discreet format

$$\frac{i_1[t+step] - i_1[t]}{step} = D[t]\frac{v_1[t]}{L} - \frac{v_0[t]}{L} \tag{3}$$

$$i_1[t+step] = i_1[t] + step \cdot \left(D[t]\frac{v_1[t]}{L} - \frac{v_0[t]}{L}\right) \tag{4}$$

2.1.2 Second equation in discreet format

$$\frac{v_0[t+step] - v_0[t]}{step} = \frac{i_1(t)}{C} - \frac{v_0(t)}{RC} \tag{5}$$

$$v_0[t+step] = v_0[t] + step\left(\frac{i_1(t)}{C} - \frac{v_0(t)}{RC}\right) \tag{6}$$

2.2 Differential Equations Discreet

$$i_1[t+step] = i_1[t] + step \cdot \left(D[t]\frac{v_1[t]}{L} - \frac{v_0[t]}{L}\right) \tag{7}$$

$$v_0[t+step] = v_0[t] + step\left(\frac{i_1(t)}{C} - \frac{v_0(t)}{RC}\right) \tag{8}$$

3 1a

3.1 Dependencies

[1]:
```
import math
import numpy as np
from scipy.optimize import fsolve

# Analysis and plotting modules
import pandas as pd
import plotly


# cadCAD configuration modules
from cadCAD.configuration.utils import config_sim
from cadCAD.configuration import Experiment
from cadCAD import configs

# cadCAD simulation engine modules
from cadCAD.engine import ExecutionMode, ExecutionContext
from cadCAD.engine import Executor



pd.options.plotting.backend = "plotly"
```

3.2 State Variables

[2]:
```
initial_state = {
    'i_1': 0, #A
    'v_o': 0, #V
    'D': 0.5  #abs
}

initial_state
```

[2]:
```
{'i_1': 0, 'v_o': 0, 'D': 0.5}
```

[3]:
```
system_params = {
    'step': [0.000001],             #1 s
    'R': [1],                       #ohms
    'L': [math.pow(10,-3)],         #H
    'C': [800*math.pow(10,-6)],     #F
    'vi': [1],                      #V
    'fs': [5 * math.pow(10,3)],      #5kHz
    'Rsh': [38.17],                 #ohms
    'Rs': [61.3*math.pow(10,-3)],   #ohms
```

```
    'beta': [86.14 * math.pow(10,-6)], #V/K
    'n': [1.7536],                     #abs
    'Is': [5.68*math.pow(10,-6)],      #A
    'Iphn': [3.1656],                  #A
    'Gn': [math.pow(10,3)],            #W/m2
    'T': [298]                         #°K
}


system_params
```

[3]:
```
{'step': [1e-06],
 'R': [1],
 'L': [0.001],
 'C': [0.0007999999999999999],
 'vi': [1],
 'fs': [5000.0],
 'Rsh': [38.17],
 'Rs': [0.0613],
 'beta': [8.614e-05],
 'n': [1.7536],
 'Is': [5.68e-06],
 'Iphn': [3.1656],
 'Gn': [1000.0],
 'T': [298]}
```

3.3 State Update Functions

[4]:
```
def s_duty_cycle_a(params, substep, state_history, previous_state,␣
 ↪policy_input):
    t = previous_state['timestep']*params['step']
    freq = params['fs']
    period = 1/freq
    dutycicle = 0.5
    if (t%period) < (period*dutycicle):
        pwm = 1
    else:
        pwm = 0
    return 'D', pwm


def s_i1(params, substep, state_history, previous_state, policy_input):
    L = params['L']
    vi = params['vi']
    step = params['step']
    i1 = previous_state['i_1'] + step*((previous_state['D'] * vi / L) -␣
 ↪(previous_state['v_o']/L))
```

```
    return 'i_1', i1

def s_vo(params, substep, state_history, previous_state, policy_input):
    C = params['C']
    R = params['R']
    step = params['step']
    vo = previous_state['v_o'] + step*((previous_state['i_1']/C) -␣
 ↪(previous_state['v_o']/(R*C)))
    return 'v_o', vo
```

3.4 Partial State Update Blocks

[5]:
```
partial_state_update_blocks = [
    {
        'policies': {},
        'variables': {
            'D': s_duty_cycle_a,
            'i_1': s_i1,
            'v_o': s_vo
        }
    }
]
```

3.5 Configuration

[6]:
```
sim_config = config_sim({
    "N": 1,
    "T": range(20000),
    "M": system_params
})

del configs[:] # Clear any prior configs
experiment = Experiment()
experiment.append_configs(
    initial_state = initial_state,
    partial_state_update_blocks = partial_state_update_blocks,
    sim_configs = sim_config
)
configs[-1].__dict__
```

[6]:
```
{'sim_config': {'N': 1,
  'T': range(0, 20000),
  'M': {'step': 1e-06,
   'R': 1,
   'L': 0.001,
   'C': 0.0007999999999999999,
```

```
    'vi': 1,
    'fs': 5000.0,
    'Rsh': 38.17,
    'Rs': 0.0613,
    'beta': 8.614e-05,
    'n': 1.7536,
    'Is': 5.68e-06,
    'Iphn': 3.1656,
    'Gn': 1000.0,
    'T': 298},
   'subset_id': 0,
   'subset_window': deque([0, None]),
   'simulation_id': 0,
   'run_id': 0},
  'initial_state': {'i_1': 0, 'v_o': 0, 'D': 0.5},
  'seeds': {},
  'env_processes': {},
  'exogenous_states': {},
  'partial_state_updates': [{'policies': {},
    'variables': {'D': <function __main__.s_duty_cycle_a(params, substep,
state_history, previous_state, policy_input)>,
     'i_1': <function __main__.s_i1(params, substep, state_history,
previous_state, policy_input)>,
     'v_o': <function __main__.s_vo(params, substep, state_history,
previous_state, policy_input)>}}],
  'policy_ops': [<function cadCAD.configuration.Experiment.<lambda>(a, b)>],
  'kwargs': {},
  'user_id': 'cadCAD_user',
  'session_id': 'cadCAD_user=0_0',
  'simulation_id': 0,
  'run_id': 0,
  'experiment_id': 0,
  'exp_window': deque([1, 0]),
  'subset_id': 0,
  'subset_window': deque([0, None])}
```

3.6 Execution

[7]:
```python
exec_context = ExecutionContext()
simulation = Executor(exec_context=exec_context, configs=configs)
raw_result, tensor_field, sessions = simulation.execute()
```

```
                  ___________    ____
  ________ __ ___/ / ____/   |  / __ \
 / ___/ __` / __  / /   / /| | / / / /
/ /__/ /_/ / /_/ / /___/ ___ |/ /_/ /
```

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

Execution Mode: local_proc
Configuration Count: 1
Dimensions of the first simulation: (Timesteps, Params, Runs, Vars) = (20000,
14, 1, 3)
Execution Method: local_simulations
SimIDs   : [0]
SubsetIDs: [0]
Ns       : [0]
ExpIDs   : [0]
Execution Mode: single_threaded
Total execution time: 0.65s
```

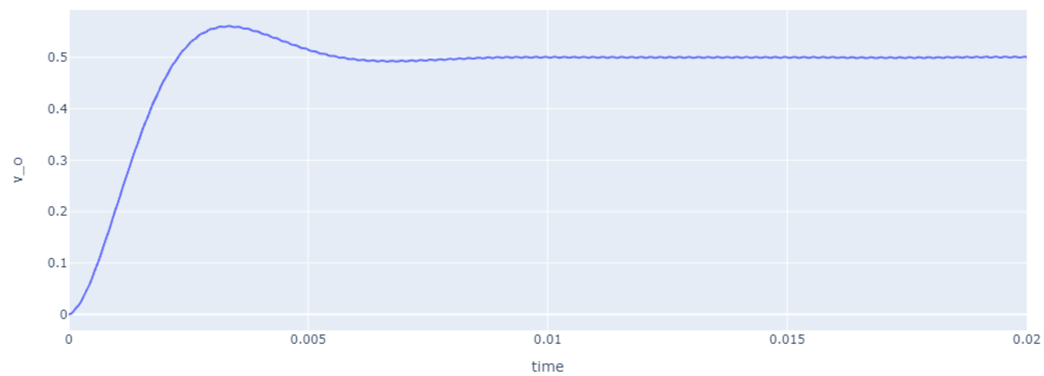
3.7 Simulation Output Preparation

[8]:
```
simulation_result = pd.DataFrame(raw_result)
simulation_result['time'] = simulation_result['timestep'] *␣
↪system_params['step'][0]
simulation_result.head()
```

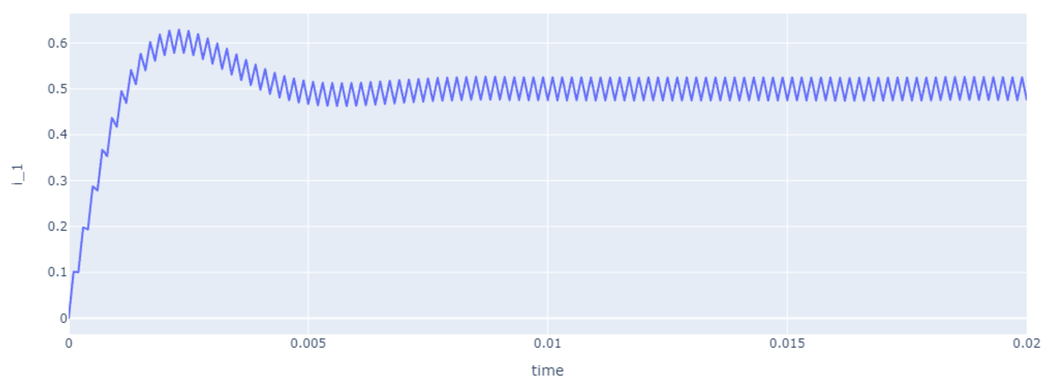
[8]:
```
      i_1           v_o    D  simulation  subset  run  substep  timestep  \
0  0.0000  0.000000e+00  0.5           0       0    1        0         0
1  0.0005  0.000000e+00  1.0           0       0    1        1         1
2  0.0015  6.250000e-07  1.0           0       0    1        1         2
3  0.0025  2.499219e-06  1.0           0       0    1        1         3
4  0.0035  5.621094e-06  1.0           0       0    1        1         4

       time
0  0.000000
1  0.000001
2  0.000002
3  0.000003
4  0.000004
```

3.8 Simulation Analysis

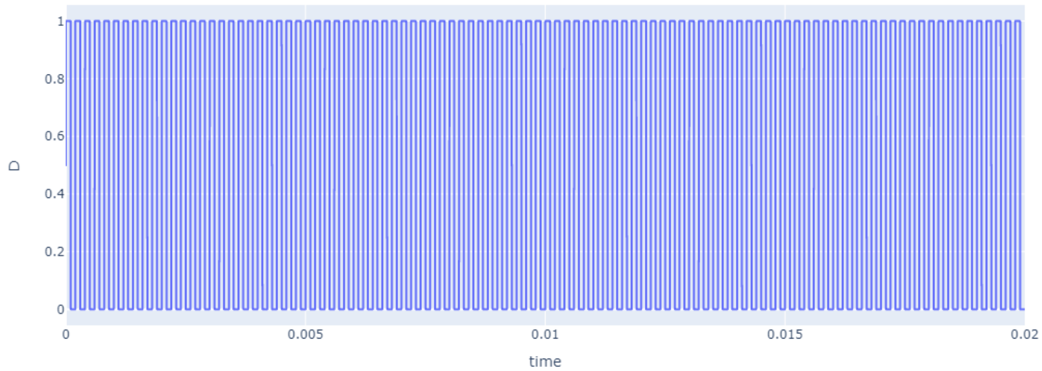
[9]:
```
simulation_result.plot(kind='line', x='time', y='v_o')
```



[10]:
```
simulation_result.plot(kind='line', x='time', y='i_1')
```



[11]:
```
simulation_result.plot(kind='line', x='time', y='D')
```



4 7

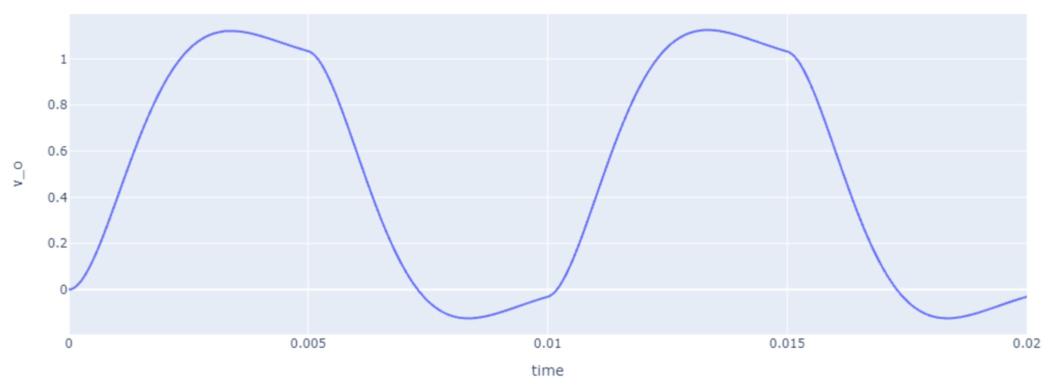
[12]:
```
system_params['fs'] = [100]
```

[13]:
```
sim_config = config_sim({
    "N": 1,
    "T": range(20000),
    "M": system_params
})

del configs[:] # Clear any prior configs
experiment = Experiment()
experiment.append_configs(
    initial_state = initial_state,
    partial_state_update_blocks = partial_state_update_blocks,
    sim_configs = sim_config
)

exec_context = ExecutionContext()
simulation = Executor(exec_context=exec_context, configs=configs)
raw_result, tensor_field, sessions = simulation.execute()
```

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

Execution Mode: local_proc
```

```
Configuration Count: 1
Dimensions of the first simulation: (Timesteps, Params, Runs, Vars) = (20000,
14, 1, 3)
Execution Method: local_simulations
SimIDs    : [0]
SubsetIDs: [0]
Ns        : [0]
ExpIDs    : [0]
Execution Mode: single_threaded
Total execution time: 0.59s
```

[14]:
```
simulation_result = pd.DataFrame(raw_result)
simulation_result['time'] = simulation_result['timestep'] *␣
 ↪system_params['step'][0]
simulation_result.plot(kind='line', x='time', y='v_o')
```



5 1.b

[15]:
```
def s_duty_cycle_b(params, substep, state_history, previous_state,␣
 ↪policy_input):
    return 'D', 0.5
```

[16]:
```
partial_state_update_blocks = [
    {
        'policies': {},
        'variables': {
            'D': s_duty_cycle_b,
            'i_1': s_i1,
            'v_o': s_vo
        }
```

```
    }
]

del configs[:] # Clear any prior configs
experiment = Experiment()
experiment.append_configs(
    initial_state = initial_state,
    partial_state_update_blocks = partial_state_update_blocks,
    sim_configs = sim_config
)
configs[-1].__dict__
```

[16]:
```
{'sim_config': {'N': 1,
  'T': range(0, 20000),
  'M': {'step': 1e-06,
   'R': 1,
   'L': 0.001,
   'C': 0.0007999999999999999,
   'vi': 1,
   'fs': 100,
   'Rsh': 38.17,
   'Rs': 0.0613,
   'beta': 8.614e-05,
   'n': 1.7536,
   'Is': 5.68e-06,
   'Iphn': 3.1656,
   'Gn': 1000.0,
   'T': 298},
  'subset_id': 0,
  'subset_window': deque([0, None]),
  'simulation_id': 0,
  'run_id': 0},
 'initial_state': {'i_1': 0, 'v_o': 0, 'D': 0.5},
 'seeds': {},
 'env_processes': {},
 'exogenous_states': {},
 'partial_state_updates': [{'policies': {},
   'variables': {'D': <function __main__.s_duty_cycle_b(params, substep,
state_history, previous_state, policy_input)>,
    'i_1': <function __main__.s_i1(params, substep, state_history,
previous_state, policy_input)>,
    'v_o': <function __main__.s_vo(params, substep, state_history,
previous_state, policy_input)>}}],
 'policy_ops': [<function cadCAD.configuration.Experiment.<lambda>(a, b)>],
 'kwargs': {},
 'user_id': 'cadCAD_user',
 'session_id': 'cadCAD_user=0_0',
```

```
 'simulation_id': 0,
 'run_id': 0,
 'experiment_id': 0,
 'exp_window': deque([1, 0]),
 'subset_id': 0,
 'subset_window': deque([0, None])}
```

[17]:
```
exec_context = ExecutionContext()
simulation = Executor(exec_context=exec_context, configs=configs)
raw_result, tensor_field, sessions = simulation.execute()
```

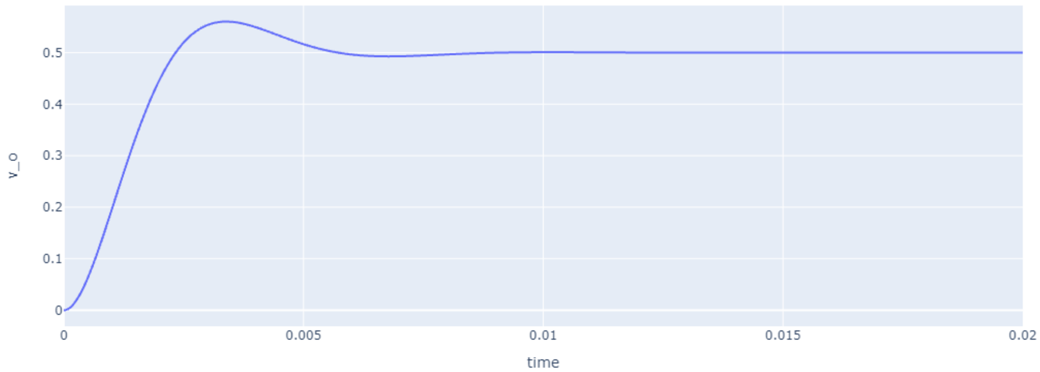
```
                  ___________    ____
  ________ __ ___/ / ____/   |  / __ \
 / ___/ __` / __  / /   / /| | / / / /
/ /__/ /_/ / /_/ / /___/ ___ |/ /_/ /
\___/\__,_/\__,_/\____/_/  |_/_____/
by cadCAD

Execution Mode: local_proc
Configuration Count: 1
Dimensions of the first simulation: (Timesteps, Params, Runs, Vars) = (20000,
14, 1, 3)
Execution Method: local_simulations
SimIDs   : [0]
SubsetIDs: [0]
Ns       : [0]
ExpIDs   : [0]
Execution Mode: single_threaded
Total execution time: 0.61s
```

[18]:
```
simulation_result = pd.DataFrame(raw_result)
simulation_result['time'] = simulation_result['timestep'] *␣
 ↪system_params['step'][0]
simulation_result.head()
```

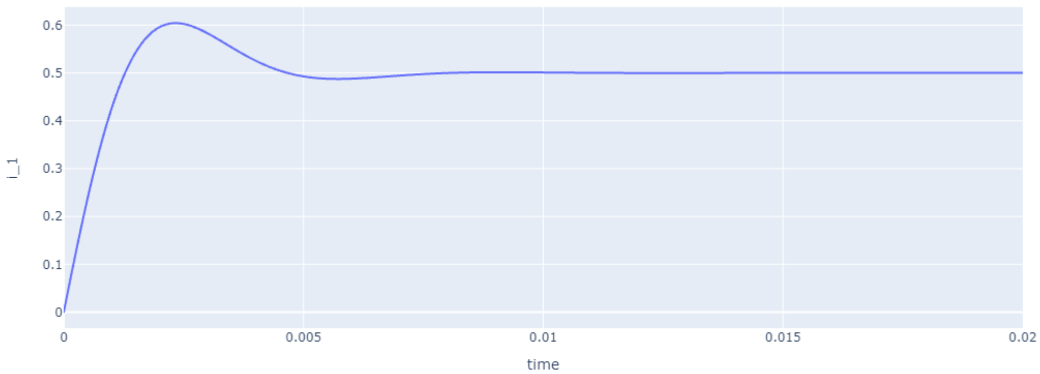
[18]:
```
      i_1           v_o    D  simulation  subset  run  substep  timestep  \
0  0.0000  0.000000e+00  0.5           0       0    1        0         0
1  0.0005  0.000000e+00  0.5           0       0    1        1         1
2  0.0010  6.250000e-07  0.5           0       0    1        1         2
3  0.0015  1.874219e-06  0.5           0       0    1        1         3
4  0.0020  3.746875e-06  0.5           0       0    1        1         4

       time
0  0.000000
1  0.000001
2  0.000002
```

```
3   0.000003
4   0.000004
```

[19]:
```python
simulation_result.plot(kind='line', x='time', y='v_o')
```



[20]:
```python
simulation_result.plot(kind='line', x='time', y='i_1')
```



6 2

Com a frequencia os valores tem uma oscilação

7 3

We know, that:

$$\frac{di_1(t)}{dt} = D(t)\frac{v_1(t)}{L} - \frac{v_0(t)}{L} \tag{9}$$

$$\frac{dv_0(t)}{dt} = D(t)\frac{i_1(t)}{C} - \frac{v_0(t)}{RC} \tag{10}$$

aplicando a transferência nas duas EDOs, temos que:

$$sI(s) = D(s)\frac{v_1}{L} - \frac{v_0(s)}{L} \tag{11}$$

ou ainda:

$$I(s) = \frac{D(s)v_1 - v_0(s)}{Ls} \tag{12}$$

na segunda EDO:

$$sV_o(s) = \frac{I(s)}{C} - \frac{v_0(s)}{RC} \tag{13}$$

Substituindo, temos que:

$$sV_o(s) = \frac{v_1 D(s) - V_o(s)}{CLs} - \frac{V_o(s)}{RC} \tag{14}$$

$$sV_o(s) + \frac{V_o(s)}{CLs} + \frac{V_o(s)}{RC} = \frac{v_1 D(s)}{CLs} \tag{15}$$

$$V_o(s)(s + \frac{V_o(s)}{CLs} + \frac{V_o(s)}{RC}) = \frac{v_1 D(s)}{CLs} \tag{16}$$

$$V_o(s)(\frac{R + Ls + LCs^2R}{CLsR}) = \frac{v_1 D(s)}{CLs} \tag{17}$$

$$V_o(s)(\frac{R + Ls + LCs^2R}{R}) = v_1 D(s) \tag{18}$$

$$G(s) = \frac{V_o(s)}{D(s)} = \frac{Rv_1}{RLCs^2 + sL + R} \tag{19}$$

8 4

Podemos rescrever a função acima no formato padrão da função de transferência de segunda ordem:

$$G(s) = \frac{Rv_1}{LCs^2 + \frac{sL}{R} + 1} \tag{20}$$

Dessa forma, temos $\tau = \sqrt{LC}$ e para o fator de amortecimento, temos:

$$2\tau\xi = \frac{L}{R} \tag{21}$$

$$\xi = \frac{L}{2R\tau} \tag{22}$$

$$\xi = \frac{\sqrt{L}}{2R\sqrt{C}} \tag{23}$$

Sendo assim, sabemos que a resposta é considerada criticamente amortecida quando $\xi = 1$. Logo:

$$\xi = \frac{\sqrt{L}}{2R\sqrt{C}} = 1 \tag{24}$$

$$R = \frac{\sqrt{L}}{2\xi\sqrt{C}} \tag{25}$$

$$R = \frac{\sqrt{L}}{2\sqrt{C}} \tag{26}$$

9 5

[21]:
```
system_params['R'] = [0.4, 0.559, 0.7]
system_params
```

[21]:
```
{'step': [1e-06],
 'R': [0.4, 0.559, 0.7],
 'L': [0.001],
 'C': [0.0007999999999999999],
 'vi': [1],
 'fs': [100],
 'Rsh': [38.17],
 'Rs': [0.0613],
 'beta': [8.614e-05],
 'n': [1.7536],
 'Is': [5.68e-06],
 'Iphn': [3.1656],
 'Gn': [1000.0],
 'T': [298]}
```

[22]:

```python
sim_config = config_sim({
    "N": 1,
    "T": range(20000),
    "M": system_params
})

partial_state_update_blocks = [
    {
        'policies': {},
        'variables': {
            'D': s_duty_cycle_b,
            'i_1': s_i1,
            'v_o': s_vo
        }
    }
]

del configs[:] # Clear any prior configs
experiment = Experiment()
experiment.append_configs(
    initial_state = initial_state,
    partial_state_update_blocks = partial_state_update_blocks,
    sim_configs = sim_config
)
configs[-1].__dict__
```

[22]:

```
{'sim_config': {'N': 3,
  'T': range(0, 20000),
  'M': {'step': 1e-06,
   'R': 0.7,
   'L': 0.001,
   'C': 0.0007999999999999999,
   'vi': 1,
   'fs': 100,
   'Rsh': 38.17,
   'Rs': 0.0613,
   'beta': 8.614e-05,
   'n': 1.7536,
   'Is': 5.68e-06,
   'Iphn': 3.1656,
   'Gn': 1000.0,
   'T': 298},
  'subset_id': 2,
  'subset_window': deque([0, None]),
  'simulation_id': 0,
  'run_id': 2},
 'initial_state': {'i_1': 0, 'v_o': 0, 'D': 0.5},
```

```
 'seeds': {},
 'env_processes': {},
 'exogenous_states': {},
 'partial_state_updates': [{'policies': {},
   'variables': {'D': <function __main__.s_duty_cycle_b(params, substep,
state_history, previous_state, policy_input)>,
    'i_1': <function __main__.s_i1(params, substep, state_history,
previous_state, policy_input)>,
    'v_o': <function __main__.s_vo(params, substep, state_history,
previous_state, policy_input)>}}],
 'policy_ops': [<function cadCAD.configuration.Experiment.<lambda>(a, b)>],
 'kwargs': {},
 'user_id': 'cadCAD_user',
 'session_id': 'cadCAD_user=0_2',
 'simulation_id': 0,
 'run_id': 2,
 'experiment_id': 0,
 'exp_window': deque([1, 0]),
 'subset_id': 2,
 'subset_window': deque([0, None])}
```

[23]:
```python
exec_context = ExecutionContext()
simulation = Executor(exec_context=exec_context, configs=configs)
raw_result, tensor_field, sessions = simulation.execute()
```

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

Execution Mode: local_proc
Configuration Count: 1
Dimensions of the first simulation: (Timesteps, Params, Runs, Vars) = (20000,
14, 3, 3)
Execution Method: local_simulations
SimIDs   : [0, 0, 0]
SubsetIDs: [0, 1, 2]
Ns       : [0, 1, 2]
ExpIDs   : [0, 0, 0]
Execution Mode: parallelized
Total execution time: 1.68s
```

[24]:
```python
simulation_result = pd.DataFrame(raw_result)
```

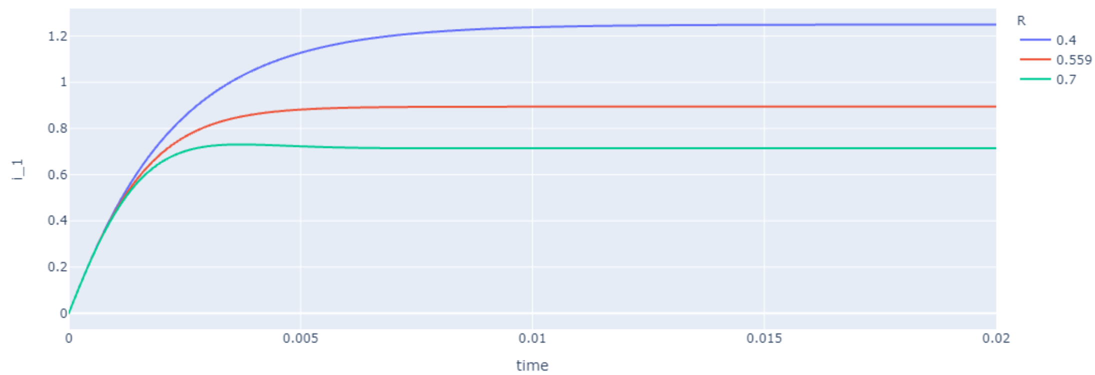
```
simulation_result['time'] = simulation_result['timestep'] *␣
 ↪system_params['step'][0]
simulation_result.head()
```

[24]:

```
        i_1           v_o    D  simulation  subset  run  substep  timestep  \
0  0.0000  0.000000e+00  0.5           0       0    1        0         0
1  0.0005  0.000000e+00  0.5           0       0    1        1         1
2  0.0010  6.250000e-07  0.5           0       0    1        1         2
3  0.0015  1.873047e-06  0.5           0       0    1        1         3
4  0.0020  3.742193e-06  0.5           0       0    1        1         4

       time
0  0.000000
1  0.000001
2  0.000002
3  0.000003
4  0.000004
```

[25]:

```
simulation_result['R'] = simulation_result['subset']
simulation_result['R'].replace({0: 0.4, 1: 0.559, 2:0.7}, inplace=True)
simulation_result.plot(kind='line', x='time', y='i_1', color='R')
```



10 9

10.1 a e b

[26]:

```
system_params['kp'] = [0.1]   #1/V
system_params['ki'] = [200]   #1/Vs
system_params['kd'] = [0.02] #s/V
system_params['N'] = [10**5] #1/V
system_params['Vref'] = [0.15]   #V
```

```
system_params['R'] = [1]

system_params
```

[26]:
```
{'step': [1e-06],
 'R': [1],
 'L': [0.001],
 'C': [0.0007999999999999999],
 'vi': [1],
 'fs': [100],
 'Rsh': [38.17],
 'Rs': [0.0613],
 'beta': [8.614e-05],
 'n': [1.7536],
 'Is': [5.68e-06],
 'Iphn': [3.1656],
 'Gn': [1000.0],
 'T': [298],
 'kp': [0.1],
 'ki': [200],
 'kd': [0.02],
 'N': [100000],
 'Vref': [0.15]}
```

[27]:
```
initial_state = {
    'i_1': 3.15660026, #A
    'v_o': 0.15, #V
    'D': 0.15, #abs
    'I': 3.15660026,    #A
    'e': 0,
    'ui': 0.15,
    'ud': 0,
    'vref': 0.15
}

initial_state
```

[27]:
```
{'i_1': 3.15660026,
 'v_o': 0.15,
 'D': 0.15,
 'I': 3.15660026,
 'e': 0,
 'ui': 0.15,
 'ud': 0,
 'vref': 0.15}
```

[28]:
```
def calculate_e(params, previous_state):
    C = params['C']
    R = params['R']
    step = params['step']
    R = previous_state['v_o'] / previous_state['I']
    vo = previous_state['v_o'] + step*((previous_state['i_1']/C) -␣
↪(previous_state['I']/C))
    e = previous_state['vref'] - vo
    return e

def s_vref(params, substep, state_history, previous_state, policy_input):
    return 'vref', 0.15

def s_I(params, substep, state_history, previous_state, policy_input):
    Iph = params['Iphn']
    Is = params['Is']
    Vt = params['beta'] * params['T']
    Rs = params['Rs']
    n = params['n']
    Rsh = params['Rsh']
    v_o = previous_state['v_o']

    def equation(vars):
        I = vars
        eq = I - Iph + Is*(np.exp((v_o + I*Rs)/(n*Vt)) - 1) + (v_o + (I*Rs))/Rsh
        return eq

    return 'I', fsolve(equation, previous_state['I'], xtol=1e-6)[0]

def s_vo(params, substep, state_history, previous_state, policy_input):
    C = params['C']
    R = params['R']
    step = params['step']
    R = previous_state['v_o'] / previous_state['I']
    vo = previous_state['v_o'] + step*((previous_state['i_1']/C) -␣
↪(previous_state['I']/C))
    return 'v_o', vo

def s_e(params, substep, state_history, previous_state, policy_input):
    e = calculate_e(params, previous_state)
    return 'e', e

def s_control_ui(params, substep, state_history, previous_state, policy_input):
    ui = previous_state['ui']+(params['step']*params['ki']*previous_state['e'])

    if ui>1:
        ui=1
```

```
    elif ui < 0:
        ui=0
    return 'ui', ui

def s_control_ud(params, substep, state_history, previous_state, policy_input):
    e = calculate_e(params, previous_state)
    kd = params['kd']
    N = params['N']
    step = params['step']

    return 'ud', (previous_state['ud']*(1-(N*step))) +␣
 ↪(kd*N*(e-previous_state['e']))


def s_control_D(params, substep, state_history, previous_state, policy_input):
    kd = params['kd']
    N = params['N']
    step = params['step']
    e = calculate_e(params, previous_state)

    up = params['kp'] * e

    ui = previous_state['ui']+(params['step']*params['ki']*previous_state['e'])

    ud = (previous_state['ud']*(1-(N*step))) + (kd*N*(e-previous_state['e']))

    d = up+ui+ud
    if d > 1:
        d = 1

    elif d < 0:
        d = 0
    return 'D', d
```

[29]:
```
partial_state_update_blocks = [
    {
        'policies': {},
        'variables': {
            'D': s_control_D,
            'i_1': s_i1,
            'v_o': s_vo,
            'I': s_I,
            'e': s_e,
            'ui': s_control_ui,
            'ud': s_control_ud,
```

```
                'vref': s_vref
            }
        }
]
```

[30]:
```
sim_config = config_sim({
    "N": 1,
    "T": range(300000),
    "M": system_params
})

del configs[:] # Clear any prior configs
experiment = Experiment()
experiment.append_configs(
    initial_state = initial_state,
    partial_state_update_blocks = partial_state_update_blocks,
    sim_configs = sim_config
)
configs[-1].__dict__
```

[30]:
```
{'sim_config': {'N': 1,
  'T': range(0, 300000),
  'M': {'step': 1e-06,
   'R': 1,
   'L': 0.001,
   'C': 0.0007999999999999999,
   'vi': 1,
   'fs': 100,
   'Rsh': 38.17,
   'Rs': 0.0613,
   'beta': 8.614e-05,
   'n': 1.7536,
   'Is': 5.68e-06,
   'Iphn': 3.1656,
   'Gn': 1000.0,
   'T': 298,
   'kp': 0.1,
   'ki': 200,
   'kd': 0.02,
   'N': 100000,
   'Vref': 0.15},
  'subset_id': 0,
  'subset_window': deque([0, None]),
  'simulation_id': 0,
  'run_id': 0},
 'initial_state': {'i_1': 3.15660026,
  'v_o': 0.15,
```

```
  'D': 0.15,
  'I': 3.15660026,
  'e': 0,
  'ui': 0.15,
  'ud': 0,
  'vref': 0.15},
 'seeds': {},
 'env_processes': {},
 'exogenous_states': {},
 'partial_state_updates': [{'policies': {},
   'variables': {'D': <function __main__.s_control_D(params, substep,
state_history, previous_state, policy_input)>,
    'i_1': <function __main__.s_i1(params, substep, state_history,
previous_state, policy_input)>,
    'v_o': <function __main__.s_vo(params, substep, state_history,
previous_state, policy_input)>,
    'I': <function __main__.s_I(params, substep, state_history, previous_state,
policy_input)>,
    'e': <function __main__.s_e(params, substep, state_history, previous_state,
policy_input)>,
    'ui': <function __main__.s_control_ui(params, substep, state_history,
previous_state, policy_input)>,
    'ud': <function __main__.s_control_ud(params, substep, state_history,
previous_state, policy_input)>,
    'vref': <function __main__.s_vref(params, substep, state_history,
previous_state, policy_input)>}}],
 'policy_ops': [<function cadCAD.configuration.Experiment.<lambda>(a, b)>],
 'kwargs': {},
 'user_id': 'cadCAD_user',
 'session_id': 'cadCAD_user=0_0',
 'simulation_id': 0,
 'run_id': 0,
 'experiment_id': 0,
 'exp_window': deque([1, 0]),
 'subset_id': 0,
 'subset_window': deque([0, None])}
```

[31]:
```
exec_context = ExecutionContext()
simulation = Executor(exec_context=exec_context, configs=configs)
raw_result, tensor_field, sessions = simulation.execute()
```

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

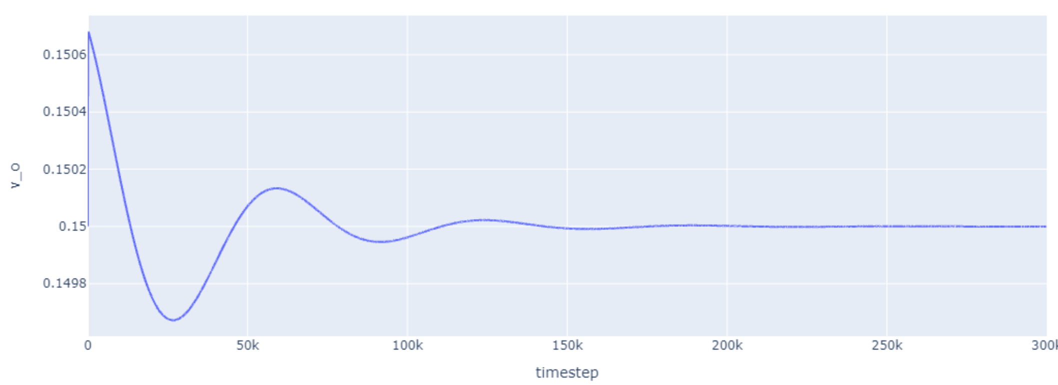
```
by cadCAD

Execution Mode: local_proc
Configuration Count: 1
Dimensions of the first simulation: (Timesteps, Params, Runs, Vars) = (300000,
19, 1, 8)
Execution Method: local_simulations
SimIDs   : [0]
SubsetIDs: [0]
Ns       : [0]
ExpIDs   : [0]
Execution Mode: single_threaded
Total execution time: 57.05s
```

[32]:
```python
simulation_result = pd.DataFrame(raw_result)
simulation_result.head()
```

[32]:

```
        i_1       v_o         D         I         e    ui        ud  vref  \
0  3.156600  0.150000  0.150000  3.156600  0.000000  0.15  0.000000  0.15
1  3.156600  0.150000  0.150000  3.145101  0.000000  0.15  0.000000  0.15
2  3.156600  0.150014  0.121251  3.145101 -0.000014  0.15 -0.028748  0.15
3  3.156571  0.150029  0.095376  3.145097 -0.000029  0.15 -0.054621  0.15
4  3.156517  0.150043  0.072151  3.145093 -0.000043  0.15 -0.077845  0.15

   simulation  subset  run  substep  timestep
0           0       0    1        0         0
1           0       0    1        1         1
2           0       0    1        1         2
3           0       0    1        1         3
4           0       0    1        1         4
```

[33]:
```python
simulation_result.plot(kind='line', x='timestep', y='v_o')
```



10.2 c

[34]:
```python
system_params['step'] = [0.000005]
system_params
```

[34]:
```
{'step': [5e-06],
 'R': [1],
 'L': [0.001],
 'C': [0.0007999999999999999],
 'vi': [1],
 'fs': [100],
 'Rsh': [38.17],
 'Rs': [0.0613],
 'beta': [8.614e-05],
 'n': [1.7536],
 'Is': [5.68e-06],
 'Iphn': [3.1656],
 'Gn': [1000.0],
 'T': [298],
 'kp': [0.1],
 'ki': [200],
 'kd': [0.02],
 'N': [100000],
 'Vref': [0.15]}
```

[35]:
```python
def s_vref(params, substep, state_history, previous_state, policy_input):
    steps = (previous_state['timestep']+1)*params['step'] // 0.3
    vref = 0.15 + steps*0.03
    if vref > 0.42:
        vref = 0.42
    return 'vref', vref
```

[36]:
```python
partial_state_update_blocks = [
    {
        'policies': {},
        'variables': {
            'D': s_control_D,
            'i_1': s_i1,
            'v_o': s_vo,
            'I': s_I,
            'e': s_e,
            'ui': s_control_ui,
            'ud': s_control_ud,
            'vref': s_vref
        }
```

```
    }
]
```

[37]:
```
sim_config = config_sim({
    "N": 1,
    "T": range(600000),
    "M": system_params
})

del configs[:] # Clear any prior configs
experiment = Experiment()
experiment.append_configs(
    initial_state = initial_state,
    partial_state_update_blocks = partial_state_update_blocks,
    sim_configs = sim_config
)
configs[-1].__dict__
```

[37]:

```
{'sim_config': {'N': 1,
  'T': range(0, 600000),
  'M': {'step': 5e-06,
   'R': 1,
   'L': 0.001,
   'C': 0.0007999999999999999,
   'vi': 1,
   'fs': 100,
   'Rsh': 38.17,
   'Rs': 0.0613,
   'beta': 8.614e-05,
   'n': 1.7536,
   'Is': 5.68e-06,
   'Iphn': 3.1656,
   'Gn': 1000.0,
   'T': 298,
   'kp': 0.1,
   'ki': 200,
   'kd': 0.02,
   'N': 100000,
   'Vref': 0.15},
  'subset_id': 0,
  'subset_window': deque([0, None]),
  'simulation_id': 0,
  'run_id': 0},
 'initial_state': {'i_1': 3.15660026,
  'v_o': 0.15,
  'D': 0.15,
  'I': 3.15660026,
```

```
  'e': 0,
  'ui': 0.15,
  'ud': 0,
  'vref': 0.15},
 'seeds': {},
 'env_processes': {},
 'exogenous_states': {},
 'partial_state_updates': [{'policies': {},
   'variables': {'D': <function __main__.s_control_D(params, substep,
state_history, previous_state, policy_input)>,
    'i_1': <function __main__.s_i1(params, substep, state_history,
previous_state, policy_input)>,
    'v_o': <function __main__.s_vo(params, substep, state_history,
previous_state, policy_input)>,
    'I': <function __main__.s_I(params, substep, state_history, previous_state,
policy_input)>,
    'e': <function __main__.s_e(params, substep, state_history, previous_state,
policy_input)>,
    'ui': <function __main__.s_control_ui(params, substep, state_history,
previous_state, policy_input)>,
    'ud': <function __main__.s_control_ud(params, substep, state_history,
previous_state, policy_input)>,
    'vref': <function __main__.s_vref(params, substep, state_history,
previous_state, policy_input)>}}],
 'policy_ops': [<function cadCAD.configuration.Experiment.<lambda>(a, b)>],
 'kwargs': {},
 'user_id': 'cadCAD_user',
 'session_id': 'cadCAD_user=0_0',
 'simulation_id': 0,
 'run_id': 0,
 'experiment_id': 0,
 'exp_window': deque([1, 0]),
 'subset_id': 0,
 'subset_window': deque([0, None])}
```

[38]:

```
exec_context = ExecutionContext()
simulation = Executor(exec_context=exec_context, configs=configs)
raw_result, tensor_field, sessions = simulation.execute()
```

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

```
Execution Mode: local_proc
Configuration Count: 1
Dimensions of the first simulation: (Timesteps, Params, Runs, Vars) = (600000,
19, 1, 8)
Execution Method: local_simulations
SimIDs   : [0]
SubsetIDs: [0]
Ns       : [0]
ExpIDs   : [0]
Execution Mode: single_threaded
Total execution time: 108.63s
```

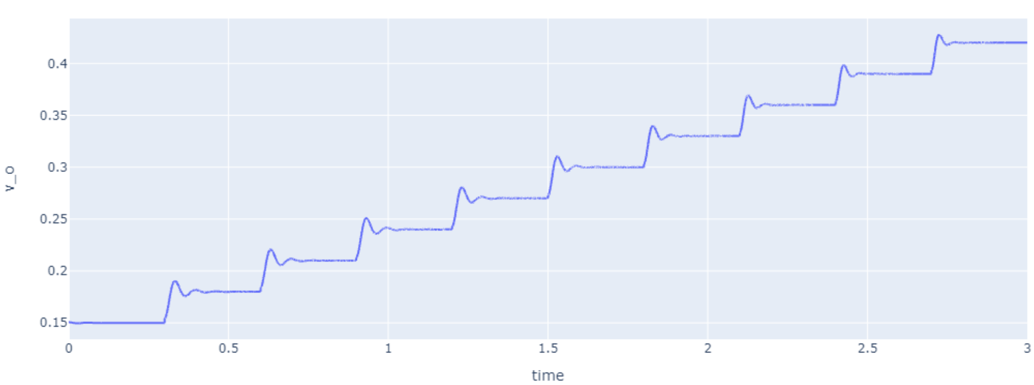
[39]:
```python
simulation_result = pd.DataFrame(raw_result)
simulation_result['time'] = simulation_result['timestep'] *␣
 ↪system_params['step'][0]
simulation_result['R'] = simulation_result['v_o'] / simulation_result['I']
simulation_result['P'] = simulation_result['v_o'] * simulation_result['I']
simulation_result.head()
```

[39]:

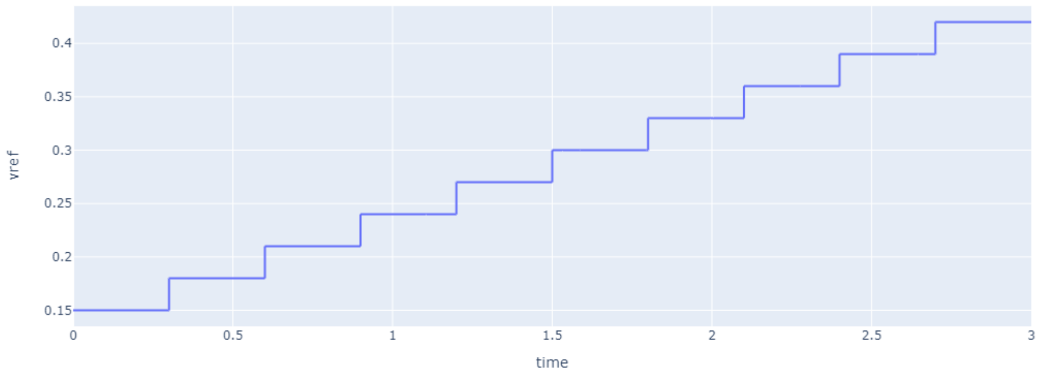
	i_1	v_o	D	I	e	ui	ud	vref \
0	3.156600	0.150000	0.150000	3.156600	0.000000	0.15	0.000000	0.15
1	3.156600	0.150000	0.150000	3.145101	0.000000	0.15	0.000000	0.15
2	3.156600	0.150072	0.006254	3.145101	-0.000072	0.15	-0.143739	0.15
3	3.155881	0.150144	0.000000	3.145081	-0.000144	0.15	-0.215609	0.15
4	3.155130	0.150211	0.000000	3.145061	-0.000211	0.15	-0.242804	0.15

	simulation	subset	run	substep	timestep	time	R	P
0	0	0	1	0	0	0.000000	0.047519	0.473490
1	0	0	1	1	1	0.000005	0.047693	0.471765
2	0	0	1	1	2	0.000010	0.047716	0.471991
3	0	0	1	1	3	0.000015	0.047739	0.472214
4	0	0	1	1	4	0.000020	0.047761	0.472424

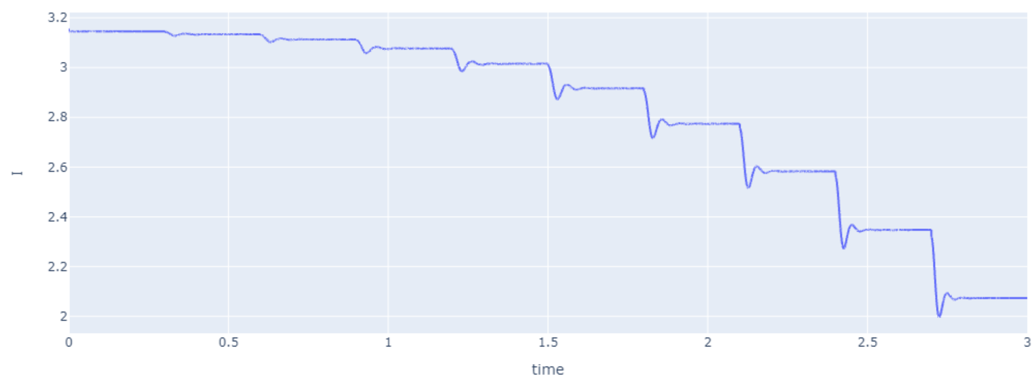
[40]:
```python
simulation_result.plot(kind='line', x='time', y='v_o')
```



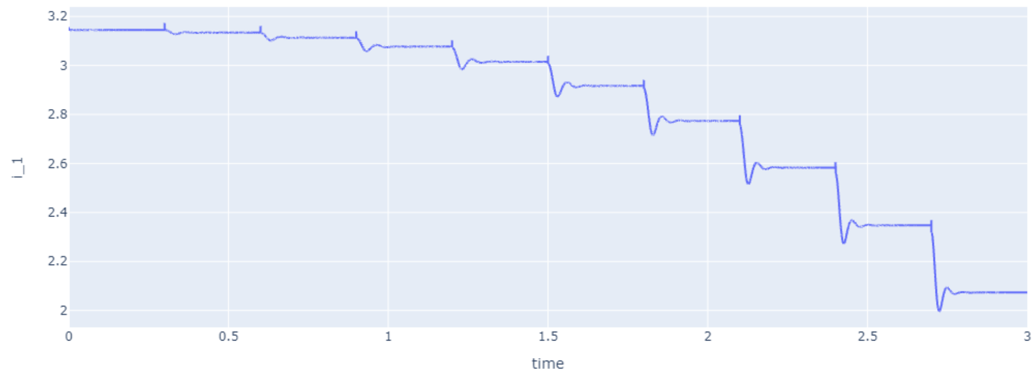
[41]:
```python
simulation_result.plot(kind='line', x='time', y='vref')
```



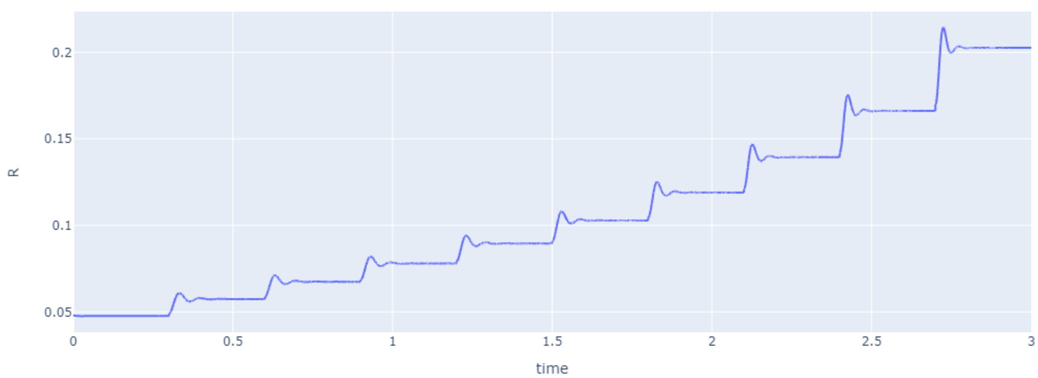
[42]:
```python
simulation_result.plot(kind='line', x='time', y='I')
```



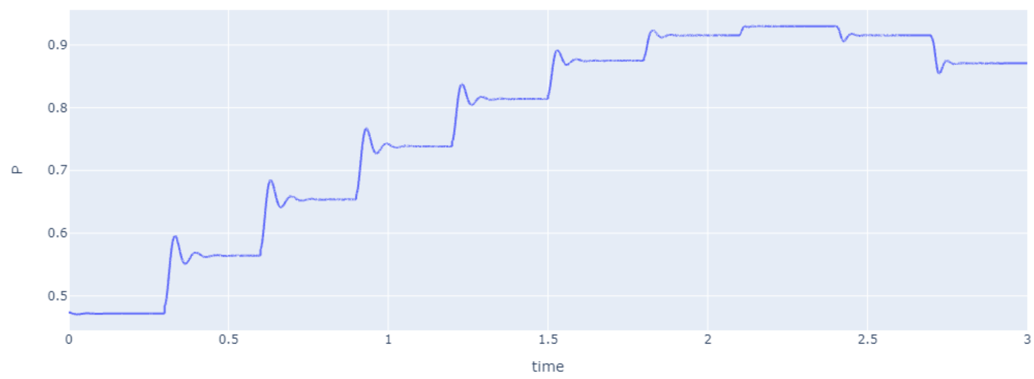
[43]:
```python
simulation_result.plot(kind='line', x='time', y='i_1')
```



[44]:
```
simulation_result.plot(kind='line', x='time', y='R')
```



[45]:
```
simulation_result.plot(kind='line', x='time', y='P')
```



[46]:
```
simulation_result.plot(kind='line', x='time', y='D')
```