**Simglucose manual**

**Version 1**

**2021-10-11**

1. **동작 구조**
   1. Controller 생성

PID, BB, DQN등의 Controller를 생성. “Simglucose” 수행 중에 필요한 함수 policy를 꼭 포함해야 한다.

**Path**:

*from* simglucose.controller.basal\_bolus\_ctrller *import* BBController

*from* simglucose.controller.pid\_ctrller *import* PIDController

*from* simglucose.controller.DQN\_ctrller *import* DQNController, DqnPredController

**Use**: 아래 소스 코드와 같이 Controller를 선택하여 생성.

controller = BBController(useAnotherParameter= False)

controller = PIDController(useAnotherParameter= True)

controller = DQNController(state\_size=1, action\_size=3, episode=1, previous\_time=4, model='c')

* 1. Simulator 생성

“Simglucose” 의 설정과 혈당 계산을 수행. 매개변수로 controller가 필요하다.

**Path**:

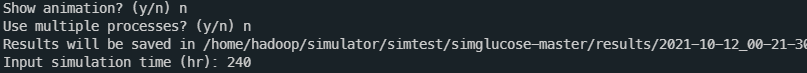
*from* simglucose.simulation.user\_interface *import* simulate

**Use**:

**옵션 명시 버전:** 사전 입력

 s = simulate(controller=controller, sim\_time=240, animate=False, parallel=False, name='', selection=1, seed=0, start\_time=random.randint(0, 23), cgm\_selection=2, pump\_selection=2, patients=[11])

**옵션 선택 버전:** 수행 중 입력

 s = simulate(controller=controller)

1. **Controller 구성**
   1. PID

PID방식으로 policy 함수를 통해 insulin 반환.

*from* .base *import* Controller

*from* .base *import* Action

*import* logging

logger = logging.getLogger(\_\_name\_\_)

class PIDController(Controller):

    def \_\_init\_\_(self, P=0.001, I=0.00001, D=0.001, target=140, useAnotherParameter=False):

        self.P = P

        self.I = I

        self.D = D

        self.target = target

        self.integrated\_state = 0

        self.prev\_state = 0

        self.asd = Action

        self.useAnotherParameter = useAnotherParameter

        self.name = 'pid'

        self.total\_pid = {"child#001": {"p": 3.49E-05

\_\_init\_\_ 함수의 매개변수를 통해 P, I, D, target을 생성시 설정이 가능하며, useAnotherParameter 값을 통해 각 가상환자별 맞는 PID값을 사용할 수 있다.

**Functions:**

**set\_parameter:** PID값을 환자에 맞게 설정

    def set\_parameter(self, patient\_name):

*if* self.useAnotherParameter:

            self.P = self.total\_pid[patient\_name]['p']

            self.I = self.total\_pid[patient\_name]['i']

            self.D = self.total\_pid[patient\_name]['d']

        self.integrated\_state = 0

        self.prev\_state = 0

        print("start patient name : ", patient\_name, "p: ", self.P, "i: ", self.I, "d: ", self.D)

**policy:** PID방식으로 계산된 insulin값 반환

  def policy(self, observation, reward, done, \*\*kwargs):

            sample\_time = kwargs.get('sample\_time')

            bg = observation.CGM

*if* self.useAnotherParameter:

                control\_input = self.P \* max([0, bg - self.target]) + \

                    self.I \* self.integrated\_state + \

                    self.D \* abs(bg - self.prev\_state) / sample\_time

*else*:

                control\_input = self.P \* (bg - self.target) + \

                                self.I \* self.integrated\_state + \

                                self.D \* (bg - self.prev\_state) / sample\_time

            self.prev\_state = bg

            self.integrated\_state += (bg - self.target) \* sample\_time

            action = Action(basal=control\_input, bolus=0)

*return* action

**reset:** parameter reset

 def reset(self):

        self.integrated\_state = 0

        self.prev\_state = 0

* 1. BB

BB방식으로 policy 함수를 통해 insulin 반환.

*from* .base *import* Controller

*from* .base *import* Action

*import* numpy *as* np

*import* pandas *as* pd

*import* pkg\_resources

*import* logging

logger = logging.getLogger(\_\_name\_\_)

CONTROL\_QUEST = pkg\_resources.resource\_filename(

    'simglucose', 'params/Quest.csv')

PATIENT\_PARA\_FILE = pkg\_resources.resource\_filename(

    'simglucose', 'params/vpatient\_params.csv')

class BBController(Controller):

    def \_\_init\_\_(self, target=140, useAnotherParameter=False):

        self.quest = pd.read\_csv(CONTROL\_QUEST)

        self.patient\_params = pd.read\_csv(

            PATIENT\_PARA\_FILE)

        self.target = target

        self.useAnotherParameter = useAnotherParameter

        self.name = 'pid'

        self.CR = 1 / 15

        self.CF = 1 / 50

        self.Age = 30

        self.TDI = 50

        self.u2ss = 1.43  *# unit: pmol/(L\*kg)*

        self.BW = 57.0  *# unit: kg*

        self.basal = self.u2ss \* self.BW / 6000  *# unit: U/min*

        self.patient\_info = {"child#001": {"CR": 28.

\_\_init\_\_ 함수 매개변수를 통해 target을 설정이 가능하며, useAnotherParameter 값을 통해 각 가상환자별 맞는 CR, CF, TDI, AGE 값을 사용할 수 있다. False로 설정 시 default 값으로 수행

**Functions:**

**set\_parameter:** 파라미터 값을 환자에 맞게 설정

def set\_parameter(self, patient\_name):

        quest = self.quest[self.quest.Name.str.match(patient\_name)]

        params = self.patient\_params[self.patient\_params.Name.str.match(patient\_name)]

        self.u2ss = params.u2ss.values.item()  *# unit: pmol/(L\*kg)*

        self.BW = params.BW.values.item()  *# unit: kg*

        self.basal = self.u2ss \* self.BW / 6000

*if* not self.useAnotherParameter:

            print("start patient: ", patient\_name)

            print(quest.CR.values)

            self.CR = int(quest.CR.values)

            self.CF = int(quest.CF.values)

*else*:

            print("start patient(anotherParameter): ", patient\_name)

            self.CR = self.patient\_info[patient\_name]['CR']

            self.CF = self.patient\_info[patient\_name]['CF']

        self.Age = self.patient\_info[patient\_name]['Age']

        self.TDI = self.patient\_info[patient\_name]['TDI']

        print("self.basal ", self.basal)

**policy:** basal bolus 방식으로 계산된 insulin값 반환

    def policy(self, observation, reward, done, \*\*kwargs):

        sample\_time = kwargs.get('sample\_time', 1)

        meal = kwargs.get('meal')

        action = self.\_bb\_policy(

            meal,

            observation.CGM,

            sample\_time)

*return* action

**\_bb\_policy:** policy 함수에서 호출되며, basal, bolus값을 계산 및 반환

    def \_bb\_policy(self, meal, glucose, env\_sample\_time):

*if* meal > 0:

            bolus = (meal / self.CR + (glucose > 150)

                     \* (glucose - self.target) / self.CF).item()  *# unit: U*

*else*:

            bolus = 0  *# unit: U*

        bolus = bolus / env\_sample\_time  *# unit: U/min*

*return* Action(basal=self.basal, bolus=bolus)

* 1. DQN(Deep Q-function Network)

DQN방식으로 policy 함수를 통해 insulin 반환

* 1. PPO(Proximal Policy Optimization)

DQN방식으로 policy 함수를 통해 insulin 반환

* 1. SAC(Soft Actor Critic)

DQN방식으로 policy 함수를 통해 insulin 반환

1. **Simulator**
2. **Supplementary**

{

"child#001": {"p": 3.49E-05, "i": 1.00E-07, "d": 1.00E-03},

'child#002': {"p": 3.98E-05, "i": 2.87E-08, "d": 3.98E-03},

'child#003': {"p": 6.31E-05, "i": 1.74E-08, "d": 1.00E-03},

'child#004': {"p": 6.31E-05, "i": 1.00E-07, "d": 1.00E-03},

'child#005': {"p": 1.00E-04, "i": 2.87E-08, "d": 6.31E-03},

'child#006': {"p": 3.49E-05, "i": 1.00E-07, "d": 1.00E-03},

'child#007': {"p": 3.98E-05, "i": 6.07E-08, "d": 2.51E-03},

'child#008': {"p": 3.49E-05, "i": 3.68E-08, "d": 1.00E-03},

'child#009': {"p": 3.49E-05, "i": 1.00E-07, "d": 1.00E-03},

'child#010': {"p": 4.54E-06, "i": 3.68E-08, "d": 2.51E-03},

'adolescent#001': {"p": 1.74E-04, "i": 1.00E-07, "d": 1.00E-02},

'adolescent#002': {"p": 1.00E-04, "i": 1.00E-07, "d": 6.31E-03},

'adolescent#003': {"p": 1.00E-04, "i": 1.00E-07, "d": 3.98E-03},

'adolescent#004': {"p": 1.00E-04, "i": 1.00E-07, "d": 4.79E-03},

'adolescent#005': {"p": 6.31E-05, "i": 1.00E-07, "d": 6.31E-03},

'adolescent#006': {"p": 4.54E-10, "i": 1.58E-11, "d": 1.00E-02},

'adolescent#007': {"p": 1.07E-07, "i": 6.07E-08, "d": 6.31E-03},

'adolescent#008': {"p": 4.54E-10, "i": 4.54E-12, "d": 1.00E-02},

'adolescent#009': {"p": 6.31E-05, "i": 1.00E-07, "d": 3.98E-03},

'adolescent#010': {"p": 4.54E-10, "i": 4.54E-12, "d": 1.00E-02},

'adult#001': {"p": 1.58E-04, "i": 1.00E-07, "d": 1.00E-02},

'adult#002': {"p": 3.98E-04, "i": 1.00E-07, "d": 1.00E-02},

'adult#003': {"p": 4.54E-10, "i": 1.00E-07, "d": 1.00E-02},

'adult#004': {"p": 1.00E-04, "i": 1.00E-07, "d": 3.98E-03},

'adult#005': {"p": 3.02E-04, "i": 1.00E-07, "d": 1.00E-02},

'adult#006': {"p": 2.51E-04, "i": 2.51E-07, "d": 1.00E-02},

'adult#007': {"p": 1.22E-04, "i": 3.49E-07, "d": 2.87E-03},

'adult#008': {"p": 1.00E-04, "i": 1.00E-07, "d": 1.00E-02},

'adult#009': {"p": 1.00E-04, "i": 1.00E-07, "d": 1.00E-02},

'adult#010': {"p": 1.00E-04, "i": 1.00E-07, "d": 1.00E-02}

}