problem set 4实验报告

16337183 孟衍璋

Problem 1

完成**SimpleBacteria**类与**Patient**类的实现。

SimpleBacteria

```
def __init__(self, birth_prob, death_prob):
    self.birth_prob = birth_prob
    self.death_prob = death_prob
```

```
def is_killed(self):
    return random.random() < self.death_prob
```

```
def reproduce(self, pop_density):
    reproduce_prob = self.birth_prob * (1 - pop_density)  # 计算复制的概率
    if random.random() < reproduce_prob:  # 如果复制了，则生成新的细菌并返回
        offspring = SimpleBacteria(self.birth_prob, self.death_prob)
        return offspring
    else:  # 如果没有复制，则raise a NoChildException
        raise NoChildException
```

Patient

```
def __init__(self, bacteria, max_pop):
    self.bacteria = bacteria
    self.max_pop = max_pop
```

```
def get_total_pop(self):
    return len(self.bacteria)
```

```
def update(self):
    surviving_bacteria = []
```

```
    # 判断每个细菌是否死去
    for bact in self.bacteria:
        if not bact.is_killed():
            surviving_bacteria.append(bact)
    # 计算存活下来的细菌密度
    new_pop_density = len(surviving_bacteria) / self.max_pop
    # 判断存活下来的细菌是否复制
    offspring_bacteria = []
    for bact in surviving_bacteria:
        try:
            offspring_bacteria.append(bact.reproduce(new_pop_density))
        except NoChildException:
            continue
    # 合并存活下来和生成的细菌
    self.bacteria = surviving_bacteria + offspring_bacteria
    return self.get_total_pop()
```

Problem 2

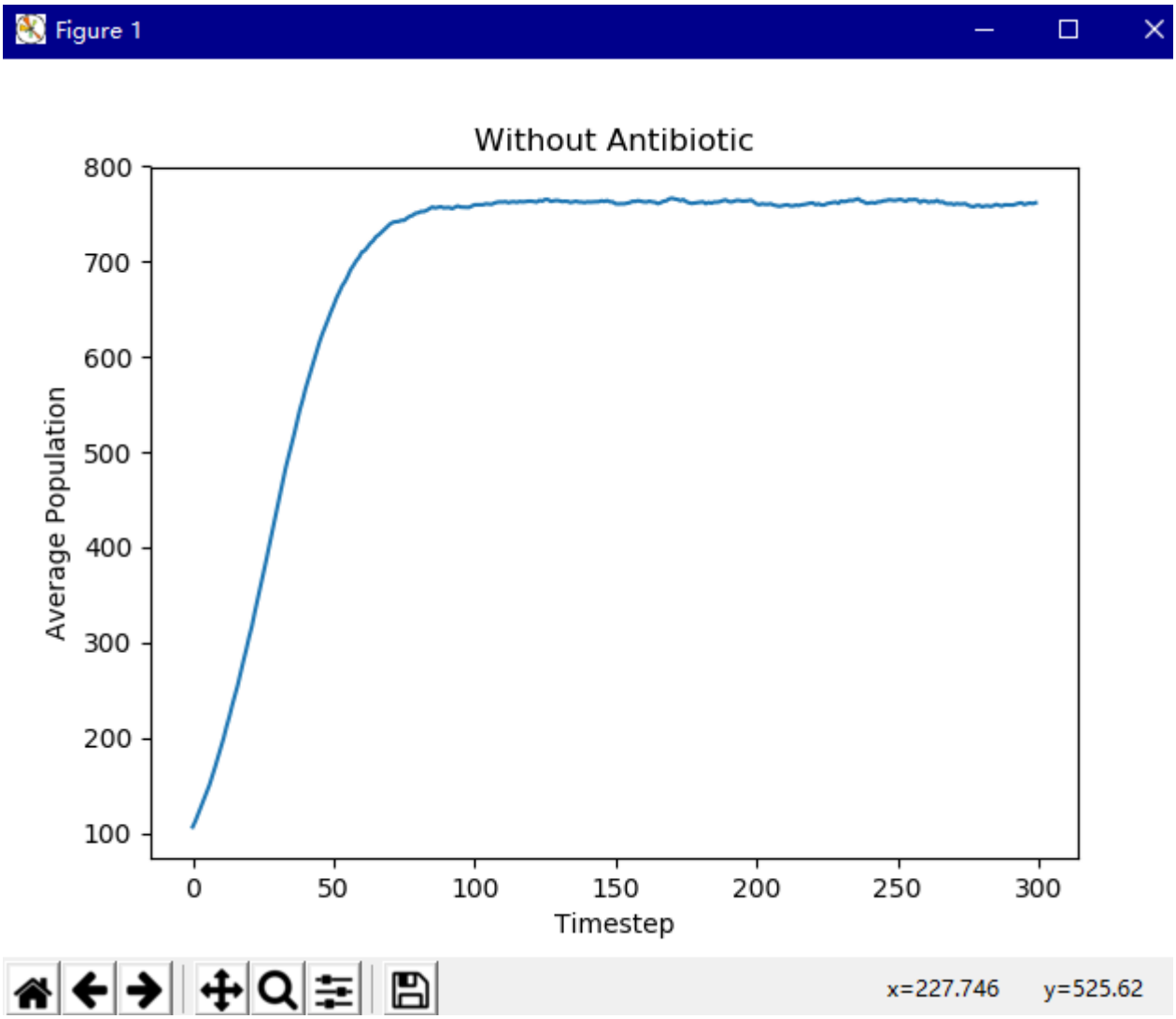
首先实现计算平均值的函数和模拟细菌在没有抗生素的情况下在人体内生长情况的函数。

```
def calc_pop_avg(populations, n):
    count = 0  # 某个时间所有trials的细菌数量总和
    for trial in populations:
        count += trial[n]
    return count / len(populations)
```

```
def simulation_without_antibiotic(num_bacteria,
                                  max_pop,
                                  birth_prob,
                                  death_prob,
                                  num_trials):
    populations = []
    for _ in range(num_trials):
        trial = []  # 存储每一次尝试的数据
        bacteria = []  # 病人体内最开始存在的细菌
        for _ in range(num_bacteria):
            bacteria.append(SimpleBacteria(birth_prob, death_prob))
        patient = Patient(bacteria, max_pop)  # 初始化病人
```

```
        for step in range(300):
            trial.append(patient.update())  # 存储每隔一个步骤的细菌数量
        populations.append(trial)
    return populations
```

模拟得到的结果如下：



Problem 3

实现计算方差和95%的置信区间的函数。

```
def calc_pop_std(populations, t):
    mean = calc_pop_avg(populations, t)  # 计算第t个步骤时的平均数
    temp = 0
    for trial in populations:
        temp += (trial[t] - mean) ** 2
    return math.sqrt(temp / len(populations))
```

```
def calc_95_ci(populations, t):
    mean = calc_pop_avg(populations, t)
    std = calc_pop_std(populations, t)
    SEM = std / math.sqrt(len(populations))
    return (mean, 1.96 * SEM)
```

此时运行测试文件，得到正确的结果：

```
D:\Study\assignment\大三下\高级编程技术\assignment\6.0002\ps4>python3 ps4_tests.py
test_calc_95_ci (__main__.ps4_calc) ... 6.653904117133038
ok
test_calc_pop_avg (__main__.ps4_calc) ... 762.5
ok
test_calc_pop_std (__main__.ps4_calc) ... 10.735455276791944
ok

----------------------------------------------------------------------
Ran 3 tests in 0.002s

OK
```

使用上面模拟得到的数据，计算的结果如下：

```
D:\Study\assignment\大三下\高级编程技术\assignment\6.0002\ps4>python3 ps4.py
the 95% confidence interval: (766.04, 4.080928986003062)
```

Problem 4

实现**ResistantBacteria**类和**TreatedPatient**类。

ResistantBacteria

```
def __init__(self, birth_prob, death_prob, resistant, mut_prob):
    self.birth_prob = birth_prob
    self.death_prob = death_prob
    self.resistant = resistant
    self.mut_prob = mut_prob
```

```
def get_resistant(self):
    return self.resistant
```

```
def is_killed(self):
    if self.get_resistant():  # 如果有耐药性
        return random.random() < self.death_prob
    else:  # 如果没有耐药性
        return random.random() < self.death_prob / 4
```

```
def reproduce(self, pop_density):
    if random.random() < self.birth_prob * (1 - pop_density):  # 如果细菌
reproduce
        if self.get_resistant():  # 如果细菌本身具有抗药性，后代也具有抗药性
            return ResistantBacteria(self.birth_prob, self.death_prob,
True, self.mut_prob)
        else:  # 如果细菌不具有抗药性，判断后代是否会变异
            resi = random.random() < self.mut_prob * (1 - pop_density)
            return ResistantBacteria(self.birth_prob, self.death_prob,
resi, self.mut_prob)
    else:  # 如果细菌不reproduce
        raise NoChildException
```

TreatedPatient

```
def __init__(self, bacteria, max_pop):
    Patient.__init__(self, bacteria, max_pop)
    self.bacteria = bacteria
    self.max_pop = max_pop
    self.on_antibiotic = False
```

```
def set_on_antibiotic(self):
    self.on_antibiotic = True
```

```
def get_resist_pop(self):
    count = 0
    for bact in self.bacteria:
        if bact.get_resistant():  # 判断细菌是否具有抗药性
            count += 1
    return count
```

```
def update(self):
    surviving_bacteria = []  # 存储存活下来的细菌
    for bact in self.bacteria:
        if not bact.is_killed():
            surviving_bacteria.append(bact)
    surviving_bacteria_after_antibiotic = []  # 存储有耐药性的细菌
    if self.on_antibiotic:  # 如果患者注射了抗生素
        for bact in surviving_bacteria:
            if bact.get_resistant():  # 如果细菌有耐药性
                surviving_bacteria_after_antibiotic.append(bact)
    else:  # 如果患者没有注射抗生素
        surviving_bacteria_after_antibiotic = surviving_bacteria
    # 计算存活下来的细菌密度
    new_pop_density = new_pop_density =
len(surviving_bacteria_after_antibiotic) / self.max_pop
    offspring_bacteria = []  # 存储后代细菌
    for bact in surviving_bacteria_after_antibiotic:
        try:
            offspring_bacteria.append(bact.reproduce(new_pop_density))
        except NoChildException:
            continue
    # 合并存活下来和生成的细菌
    self.bacteria = surviving_bacteria_after_antibiotic +
offspring_bacteria
    return self.get_total_pop()
```

Problem 5

模拟有抗药性的细菌在打了抗生素的患者体内的生长情况。

```
def simulation_with_antibiotic(num_bacteria,
                               max_pop,
                               birth_prob,
                               death_prob,
                               resistant,
                               mut_prob,
                               num_trials):
    populations = []
    resistant_pop = []
    for _ in range(num_trials):
```

```
        bacteria = []  # 存储生成的细菌
        for _ in range(num_bacteria):
            bacteria.append(ResistantBacteria(birth_prob, death_prob,
resistant, mut_prob))
        patient = TreatedPatient(bacteria, max_pop)  # 初始化受治疗的病人
        total_trial = []  # 存储每次尝试各个步骤的总细菌数
        resi_trial = []  # 存储每次尝试各个步骤的具有抗药性的细菌数
        for _ in range(150):  # 经过150个time step
            total_trial.append(patient.update())
            resi_trial.append(patient.get_resist_pop())
        patient.set_on_antibiotic()  # 注射抗生素
        for _ in range(250):  # 再经过250个time step
            total_trial.append(patient.update())
            resi_trial.append(patient.get_resist_pop())
        populations.append(total_trial)
        resistant_pop.append(resi_trial)
    return (populations, resistant_pop)
```

模拟的结果如下：

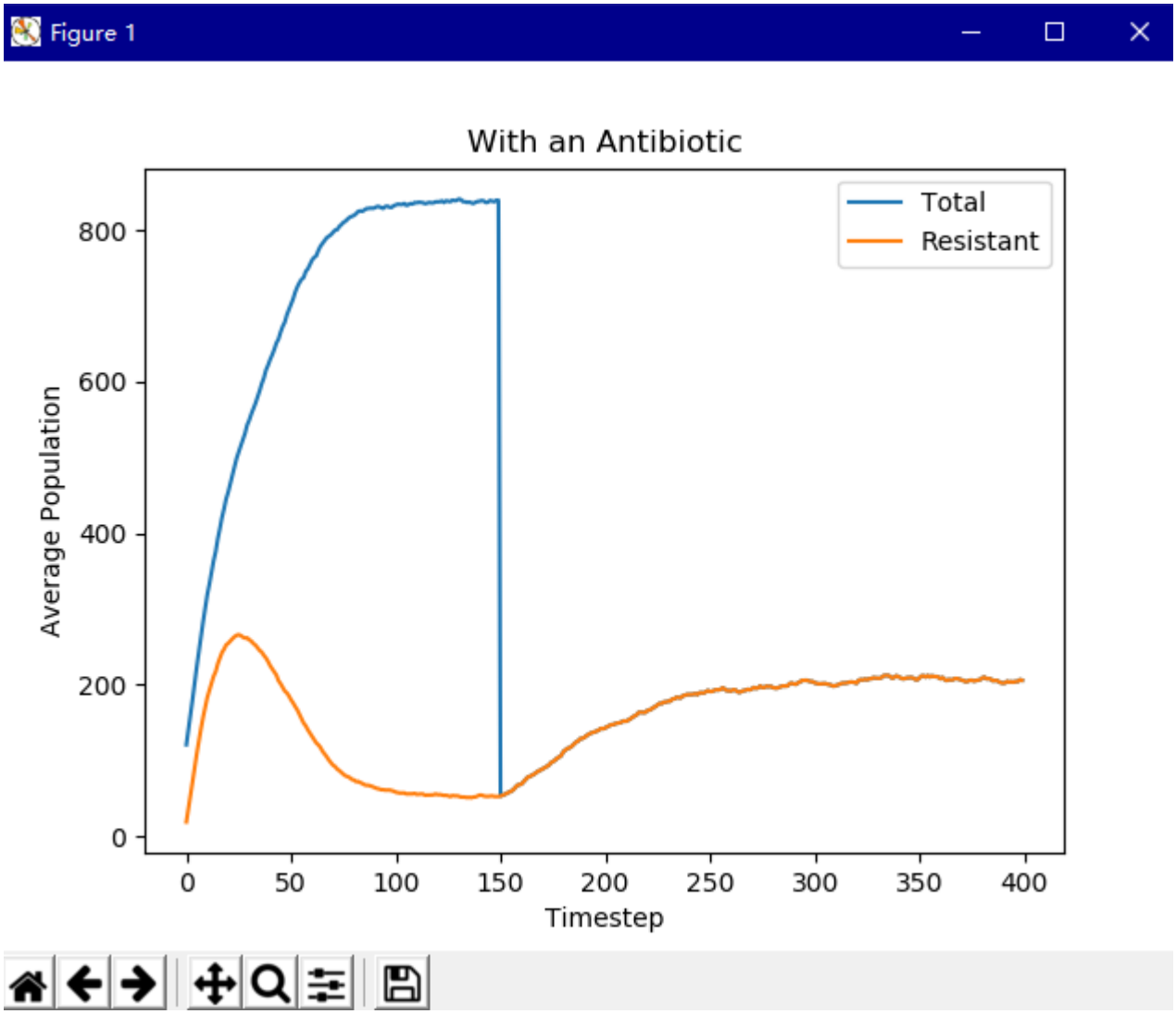
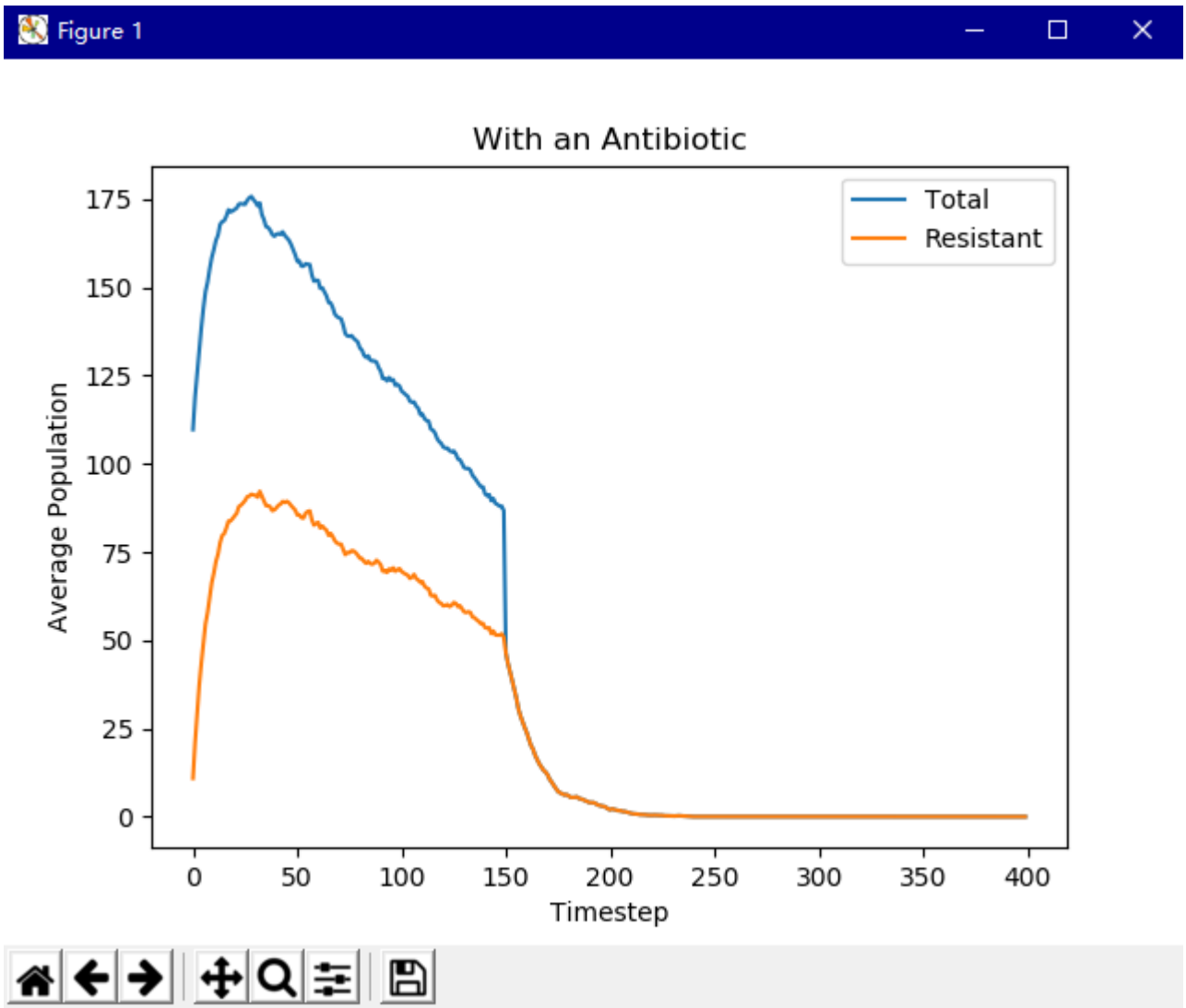


Figure 1
With an Antibiotic
Total
Resistant
Average Population
0
200
400
600
800
0
50
100
150
200
250
300
350
400
Timestep



分别计算两种模拟情况的95%置信区间如下：

```
D:\Study\assignment\大三下\高级编程技术\assignment\6.0002\ps4>python3 ps4.py
the 95% confidence interval for the total population estimate: (203.0, 7.593071368030198)
the 95% confidence interval for the resistant bacteria estimate: (203.0, 7.593071368030198)
```

```
D:\Study\assignment\大三下\高级编程技术\assignment\6.0002\ps4>python3 ps4.py
the 95% confidence interval for the total population estimate: (0.0, 0.0)
the 95% confidence interval for the resistant bacteria estimate: (0.0, 0.0)
```

Problem 6

Trends of Simulation A and Simulation B

1. What happens to the total population before introducing the antibiotic?

   在A模拟中，细菌数量逐渐上升，最终达到饱和；在B模拟中，数量先逐渐上升，后又逐渐下降。

2. What happens to the resistant bacteria population before introducing the antibiotic?

   在A模拟中，有抗药性的细菌数量先上升后下降，再趋于平稳；在B模拟中，数量先逐渐上升，后又逐渐下降。

3. What happens to the total population after introducing the antibiotic?

   在A模拟中，细菌数量突然骤减，后缓慢上升；在B模拟中，数量先骤减，再缓慢下降到0。

4. What happens to the resistant bacteria population after introducing the antibiotic?

   在A模拟中，有抗药性的细菌数量缓慢上升；在B模拟中，数量先下降，最后下降到0。