PNet：用于 Petri 网建模与仿真的 Python 库

Zhu En Chay

Colossus Technologies LLP，新加坡共和国
chayzhuen@colossus-tech.com

Bing Feng Goh

新加坡理工大学，新加坡共和国 gohbingfeng@gmail.com

Maurice HT Ling

Colossus Technologies LLP，新加坡共和国墨尔本大学生物科学学院，澳大利亚 mauriceling@colossus-tech.com

摘要

Petri网是一种形式化方法，用于描述离散时间内两个或多个状态之间的变化，已被用于对许多系统进行建模。本文介绍 PNet——一个纯 Python 库，可使用 Python 编程语言进行 Petri网建模与仿真。PNet 的设计着重于降低定义 Petri 网时的学习门槛：它采用基于文本的语言来定义变迁规则，而非使用编程结构。复杂的变迁规则可以通过常规 Python 函数进一步细化。为展示 PNet 的简便性，本文介绍两个示例——面包烘焙和流行病学模型。

关键词：网络建模、时间步长仿真、Petri网、常微分方程、Python。

1. 引言

Petri网是 C. A. Petri 设计的工具，用于对并发系统进行建模，可作为事件系统的图形化表示和数学建模工具 [1-3]。Petri网可用于对所描述的模型或过程进行形式化分析 [1-3]。Petri网由三种对象构成——库所、变迁和弧 [1-3]。库所是一个输入位置，通过一条弧与另一个变迁相连。库所通常用圆圈表示，变迁则用横条表示 [1-3]。

Petri网常用于软件工程 [4],系统建模 [5] ，甚至用于生物化学领域，例如生化反应[6], 、信号转导网络[7]和基因调控网络[8]。Liu 和 Heiner [9] 使用统一 Petri 网框架对生化反应网络进行建模和分析，便是 Petri 网应用的一个例子 [4]。这是因为可以研究这些过程的性质：终止性、可达性、安全性、有界性、活性、可逆性和归宿状态、可覆盖性、持久性以及公平性 [1]。这些性质可以 [1] 通过可覆盖树方法、可达图、关联矩阵和状态方程进行研究。

已有一些用于建模Petri网的库，例如集成在OpenModelica中的SimForge GUI [10] ，以及用于MATLAB的Petri网Simulink模块（PNSB [11]）。Pommereau [12] 开发了SNAKES，将其作为一种用于在Python 编程语言中实现Petri网的库。SNAKES [12] 采用高级面向对象编程，将令牌和所有变迁规则都实现为Python对象。虽然这提供了灵活性，但也可能使学习曲线更加陡峭。将基于文本的Petri网规范转换为SNAKES中的模型可能会困难得多。同时，SNAKES [12]不支持只能通过函数实现的复杂变迁规则。不过，SNAKES [12] 的一大优势在于集成了插件和Petri网分析工具，并且能够将已实现的Petri网转换为C语言。

在这项工作中，我们介绍了 PNet，这是一个用于 Petri 网建模的纯 Python 替代库。该库将面向对象编程的开销降至最低，并加入 Python 函数作为一种替代的变迁规则类型。因此，PNet 可能有助于降低初学者开始接触 Petri网的学习曲线，之后再过渡到功能更丰富的库（如 SNAKES）[12]。PNet 已被整合到 COPADS（https://github.com/mauriceling/copads）中。COPADS 是一个完全使用 Python 编程语言开发的算法和数据结构库，不依赖任何第三方

依赖项。未来的工作旨在实现一个图形用户界面，以提升可用性，并开发用于分析 Petri网的工具。

2. PNet 描述

本节将通过编写仿真所需的步骤来介绍 PNet。使用 PNet 编写 Petri 网仿真分为 5 个步骤：建立 Petri 网、添加库所或状态、添加变迁规则、仿真 Petri 网，以及生成结果文件。

建立 Petri 网时，需要将 PNet 作为模块导入，并在导入的模块中实例化 PNet 类。接下来，使用 add places 方法向 Petri 网中添加库所或状态。该方法接受 2 个参数：库所名称，以及一个表示初始令牌的字典，其中字典的键和值分别表示令牌名称和对应名称的令牌数量。这样，一个库所/状态便可以包含多种类型的令牌。例如，一个混合碗中有 100 颗红豆和 100 颗绿豆，可以表示为

```
net.add_places('bowl',
               {'red_beans': 100,
                'green_beans': 100})
```

不过，有时需要无限源或汇；例如，可能有数量近乎无限的人出生。在电子学中，地球被视为正电荷和负电荷的无限源。为满足这一需求，我们定义了一个名为 ouroboros 的特殊库所（ouroboros 是数学中“无穷”符号的名称），其中包含无限多个“U”令牌。

第三步是使用 add rules 方法添加变迁规则。每条变迁规则都有名称。变迁是令牌从一个库所移动到另一个库所的通道，而规则则决定移动如何发生。通常，变迁规则由源库所和目标库所（用于指引令牌）、源令牌类型（用于识别）以及目标令牌类型（用于获得精确结果）组成。随后，系统会使用逻辑运算符，根据令牌的查找值检查变迁规则的执行情况。检查所用的逻辑运算符由判据决定；判据也称为令牌经过变迁后预期得到的结果。

规则分为 5 种类型：步进、比率、时滞、孵育和函数规则。变迁规则的执行/触发取决于时间步。尽管严格来说，每条规则都应只定义一个变迁；但在实践中，一条规则可以触发一个或多个变迁，因为可以在一条规则中指定多个变迁。这可以看作是 PNet 提供的一种语法简写。

基于步进执行的步进规则会在每个时间步触发。必须指明起始库所和该库所中的令牌；同时，还必须指定目标库所以及目标库所中受影响的令牌。这表示单个变迁。例如，给定一碗红豆和绿豆，以下步进规则定义了在每个时间步交换一颗豆子：

```
网络。添加_规则('交换豆子，  '步进',        ['B1.红_豆子
```

比例规则也是一种步进执行。步进规则和比例规则具有相似的参数，区别在于使用令牌比例来触发执行。会指定变迁所需的比例，并通过逻辑运算符将其与所指示的限制值进行检查。这样可以定义移动令牌数量的增加或减少。例如，假设有 2 个碗（B1 和 B2），其中 B1 装有全部红豆，而 B2 是空碗，则可以将 B1 中剩余红豆的 10% 移动到 B2，定义如下：

```
net.add_rules('swap_ratio', 'ratio',
    ['B1.red_beans -> B2.red_beans; 0.1; \
     B1.red_beans < 1; 0'])
```

延迟规则是一种在每次令牌移动之间具有时间间隔的步进规则。实际上，延迟规则可用于产生规则脉冲移动。例如，每隔 5th 时间步长将 10 个豆子从碗 B1 移动到 B2，可以定义为

```
网。添加_规则('间隔转移', '时滞',        ['B1.豆子 -> B2.豆子;
```

孵化规则可以理解为“在执行特定操作之前，先静置一段时间”。它需要一个数值和一个计时器，计时器内部包含逻辑检查，以确保在发送到目标位置之前满足条件。例如，加入水后，将一碗豆子浸泡 60 个时间步长（例如 60 分钟），并在浸泡 60 个时间步长后将泡好的豆子转移到锅中，可以定义为

```
net.add_rules('soak', 'incubate',
    ['60; bowl.beans -> pot.beans; \
     bowl.water > 0'])
```

函数规则是一种用户定义条件。通常，当之前的 4 条规则都不符合用户需求时，会使用函数规则。不过，所有形式的变迁规则

都可以写成函数规则，因此函数规则是一个超集。函数规则的另一个常见用途是将令牌从一种类型转换为另一种类型。变迁规则之间的差异在于触发变迁规则的条件。计算时需要起始位置和目标位置，以及初始令牌和最终令牌。例如，上述比例规则

```
net.add_rules('swap_ratio', 'ratio',
    ['B1.red_beans -> B2.red_beans; 0.1; \
     B1.red_beans < 1; 0'])
```

可以写成如下函数规则，

```
def bean_swap(库所):
```

函数规则中使用的函数只接受一个参数 places，它是 PNet 中状态/库所的字典。可以使用状态/库所的名称作为 places 字典的键来访问每个状态/库所。与特定状态/库所关联的令牌以属性字典的形式实现，并可使用令牌的名称进行访问。

定义规则后，第四步是使用 simulate 或 simulate_yield 方法仿真 Petri网。随后将执行上述规则，并设置当前仿真的墙上时间和时间间隔。系统会根据规则检查墙上时间，以确保其符合指定的时间间隔。

simulate方法会将仿真结果存储在内存空间中；因此，报告间隔越短，生成的报告就越多，从而导致内存更快耗尽。另一方面，simulate_yield方法是一个生成器函数，不会预先将所有仿真结果存储在内存中。simulate方法的参数包括仿真时长、时间步长和报告频率。不过，simulate yield方法只需要仿真时长和时间步长。

最后，PNet 提供了一种方法，可将仿真结果处理为适合输出到 CSV文件 的格式。报告将在当前仿真的每个步进计数时生成，并记录内存中的每个令牌状态。令牌状态也可以通过生成列表来报告令牌的状态，该列表表示某一步进或整个仿真期间的状态。

仿真和报告通常彼此相关。例如，以下代码片段展示了如何使用simulate或simulate_yield方法进行仿真和报告生成：

```
仿真时长_时间步长 报告_频率 = 1  # 使用simulate方法
```

3. 示例

下面通过两个示例来说明 PNet 的使用方法。第一个示例是一个轻松有趣的面包烘焙示例，第二个示例则是一个更严肃但简单的流行病学模型。

示例 1：面包烘焙。在此示例中，我们对一个面包烘焙配方进行了建模（实现方式请参见附录 A），并模拟了 90 个时间步长。值得注意的是，该配方使用了 PNet 的所有功能，但不包括使用来自 Ouroboros 的无限令牌。该配方需要 1000 克面粉、500 克水、20 克糖和 1 克酵母，并按照以下步骤进行：

1. 启动搅拌机，并在每个时间步长加入 100 g 面粉、50 g 水和 2 g 糖。每隔 5 个时间步长向搅拌机中加入 0.5 g 酵母。2. 每个时间步长，搅拌机都会将 80 g 面粉、40 g 水、1.5 g 糖和 1 g 酵母制成面团。3. 搅拌完成后，将面团留在搅拌机中发酵 10 个时间步长。4. 将面团转移到烤盘中，再发酵 10 个时间步长。5. 在 400°C 下烘焙。每个时间步长，剩余面团的 30% 会被烘焙成面包。剩余面团少于 1 g 时，烘焙过程完成。6. 将面包转移到桌上冷却。每个时间步长，热量会散失 10%，直至达到 30°C 的室温。7. 享用面包。

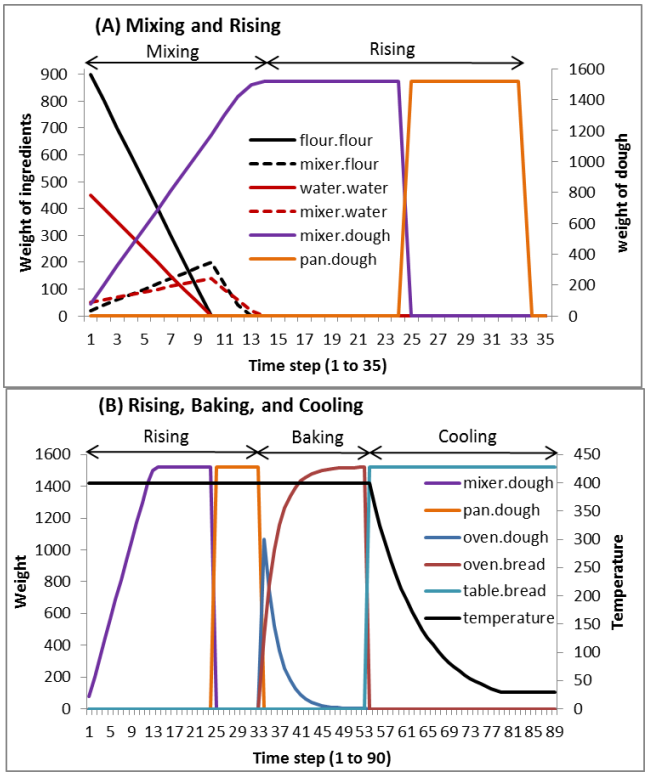

图 1 面包烘焙模拟中的令牌值。图 A 显示了配料混合成面团以及面团发酵的过程。图 B 再次展示了面团发酵，并继续呈现烘焙和冷却过程。

步骤 1 和 2 是添加配料并将其搅拌成面团的过程。面团的形成速度慢于配料的添加速度；例如，每个时间步长向搅拌机中加入 100 g 面粉和 50 g 水，但只有 80 g 面粉和 40 g 水会转化为面团。因此，在所有面粉和水都添加完之前，搅拌机中会积累 20 g 面粉和 10 g 水（如图 1A 的搅拌阶段所示）；之后，搅拌机中积累的面粉会转化为面团。所有配料都搅拌成面团后，面团会经过两个阶段的发酵（步骤 3 和 4）。随后，面团会在 400°C 下烘焙（步骤 5）。烘焙过程以每个时间步长将 30% 的面团令牌转化为面包标记来表示。当面包烘焙完成（即面团残余量可忽略不计，少于 1 g）后，会将面包转移到桌上冷却（步骤 6）。从 400° C 冷却至 30°C 室温的过程由另一个函数表示。

示例 2：流行病学模型。流行病学模型是描述生态学和流行病学现象的框架，常用于研究宿主与病原体之间的相互作用 [13]。流行病学模型已被证明有助于研究进化动力学的进化动力学，并预测病原体传播的特性，例如患病率和持续时间 [13]。字母模型是描述种群的框架，其中易感个体被视为会受到传染性病原体的侵袭 [13]。种群分为三个流行病学亚类：S 表示易感疾病者，I 表示感染者人数，R 表示在某一时刻不再参与疾病传播的人数 [13]。

易感-感染-易感（SIS）模型基于这样一种假设：病原体会感染易感人群，导致其感染；感染者康复后又会重新回到易感类 [14]。感染宿主以恒定的人均速率康复，γ 而β 是易感类的感染率 [14]。SIS 模型适用于快速演化的病毒以及不会产生免疫力的感染 [14]。易感-感染-康复（SIR）模型与 SIS 模型相似，但病原体会使感染者获得终身免疫 [15]。感染后康复的个体对再次感染具有免疫力，拥有终身免疫 [15]。该模型用于麻疹、腮腺炎和风疹等病毒性疾病 [14]。易感-感染-康复-易感（SIRS）模型与 SIR 模型相似，但获得的免疫力是暂时的 [16]。感染后康复的个体对再次感染不具有免疫力[14]。结核病就是一种采用 SIRS 模型模拟的感染疾病 [16]。

然而，大多数流行病学模型通常采用常微分方程（ODE）系统来实现 [17-20]。常微分方程和Petri网是数学建模中最常见的两种数学构造 [21]。因此，需要一种将常微分方程表示为Petri网符号形式的方法，而Soliman和Heiner提供了常微分方程与状态转移网络之间的对应关系 [22]。简而言之，常微分方程描述状态随时间的变化，而Petri网描述导致状态随时间变化的运动（图2）。从状态（节点）和变迁（弧）的角度来看，这意味着常微分方程描述节点，而Petri网描述变迁；如果两种表示中的时间单位相同，就可以轻松地在常微分方程表示与Petri网表示之间进行转换。

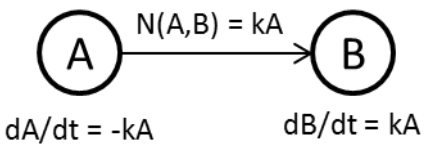

图 2 常微分方程与佩特里网转换规则之间的对应关系。

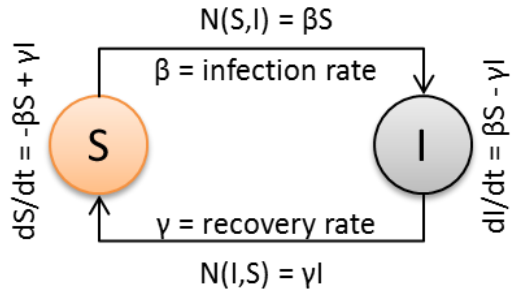

图 3 SIS流行病学模型。

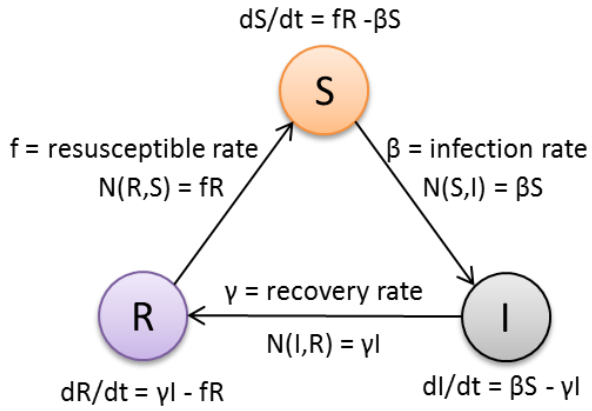

图 4 SIR/SIRS流行病学模型。

因此，SIRS [17, 18] 的标准常微分方程组可以轻松转换为Petri网表示（图 3 和图 4）。附录 B 给出了使用 PNet 实现 SIRS模型的方法。

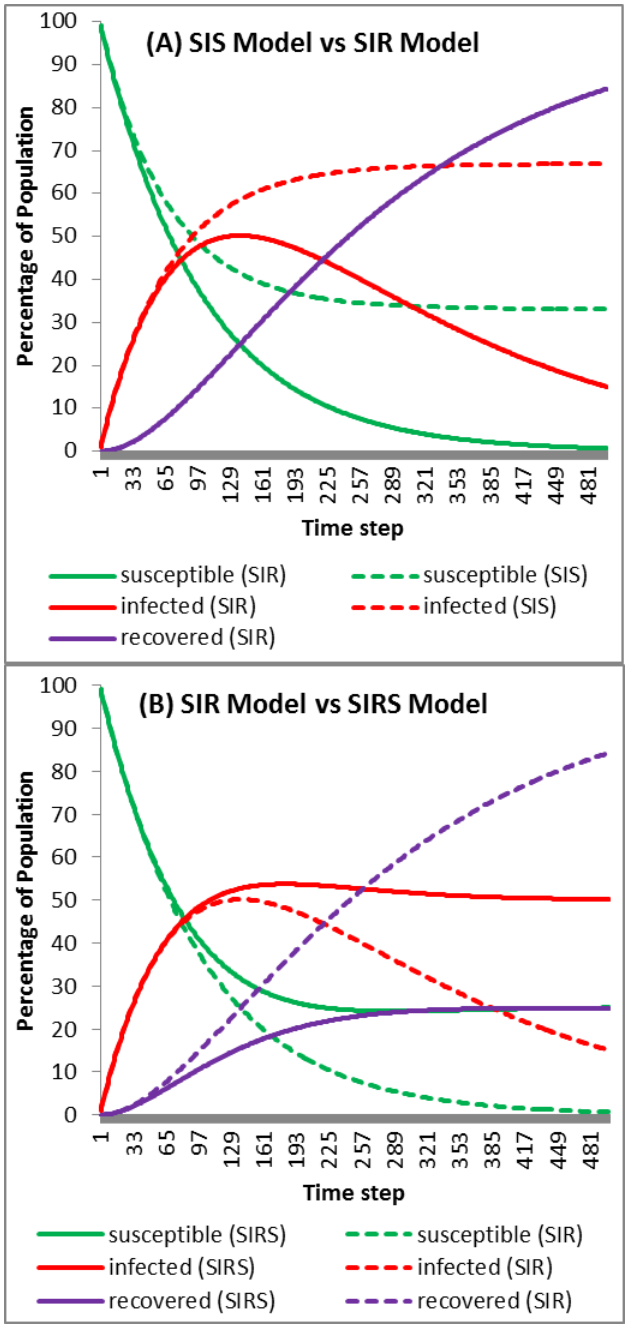

图 5 数据框示意图。

我们的仿真结果表明，感染者群体和易感者群体的比例会随时间达到平衡（图5a）。由于康复后也不会获得免疫力，预计感染者群体会保持恒定 [23], ，这也称为地方性流行群体。这里假设在整个期间没有出生和死亡，且该疾病不会导致死亡。如果康复后能够获得免疫力，SIS模型就会变为SIR模型，其中

如果没有新增出生人口，种群会逐渐变得完全免疫（图5b）。这类似于水痘的情况[24]，，水痘会使大多数康复患者获得终身免疫，因此儿童最易感，而大多数成年人具有免疫力。然而，如果获得的免疫力是暂时的，就可能再次感染，从而使模型由SIR模型转变为SIRS模型（图5b）。我们的结果表明，与SIS模型相比，SIRS模型的表现方式相似：其中存在一个数量恒定的感染者（地方性流行）群体[25]。尽管如此，其中还存在一个数量恒定的免疫个体群体，他们是近期从疾病中康复的人。当感染因子能够再次感染康复者时，这种情况是预期的。

致谢作者谨此感谢

HJ Wang（南洋理工大学）在开发和测试阶段提供的宝贵讨论。

参考文献

[1] W Reisig，《Petri网：导论》，Springer–Verlag，1985年。[2] W. Reisig，《理解Petri网：建模技术、分析方法与案例研究》，Springer–Verlag，2013年。[3] T. Murata，《Petri网：性质、分析与应用》，IEEE论文集77，1989年，541-580。[4] A. Bobbio，《使用Petri网进行系统建模》。意大利都灵10135，Strada Delle Cacce 91，Instituto Elettrotecnico Nazionale Galileo Ferraris，1990年，14-15。[5] S. Hardy和R. Iyengar，《利用Petri网和动态图分析信号网络的动态模型》，《系统生物学建模》16，2011年，225-251。[6] D. Gilbert和M. Heiner，《从Petri网到微分方程——生化网络分析的集成方法》，《Petri网与其他并发模型：I CATPN 2006》，2006年，181-200。[7] C. Chaouiya，《生物网络的Petri网建模》，《生物信息学简报》8，2007年，210-219页。[8] H. Matsuno和A. Doi，《基因调控网络的混合Petri网表示》，《太平洋生物计算研讨会》5，2000年，338-349。[9] F. Liu和M. Heiner，《使用有色随机Petri网建模膜系统》，《自然计算》12，2013年，617-629。[10] S. Proß、B. Bachmann、R. Hofestädt、K. Niehaus、R. Ueckerdt、F.-J. Vorhölter、P. Lutter、A. Stadtholz，《模拟细菌生命：Modelica中的Petri网库》，Modelica 2009会议论文集，2009年。[11] M. Matcovschi、C. Popescu和O. Pastravanu，《混合系统仿真的新方法：用于Petri网模型的Simulink库开发》，《控制工程与应用信息学杂志》7，2005年，55-62。[12] F. Pommereau，《SNAKES：灵活的高级Petri网库（工具论文）》，第36届国际Petri网会议（PETRI NETS 2015）论文集，2015年，254-265。[13] F. Brauer，《流行病学舱室模型》，载于《数学讲义》1945卷，2008年，19-79。[14] F. Brauer和C. Castillo-Chavez，《流行病学基本模型》，载于《生态时间序列》1995年，410–447。[15] M. J. Keeling和K. T.D. Eames，《网络与流行病模型》，《英国皇家学会界面期刊》2，2005年，295 307-。[16] C. Ozcaglar、A. Shabbeer、S. L. Vandenberg、B. Yener、K. P. Bennett，《结核分枝杆菌复合群感染的流行病学模型》，《数学生物科学》236，2012年，77–96。[17] P. Munz、I. Hudea、J. Imad和R. J. Smith，《僵尸来袭！：僵尸感染暴发的数学建模》，《传染病建模研究进展》4，2009年，133-150。[18] M. Ling，《COPADS IV：以Python函数集实现的方程组固定时间步长常微分方程求解器》，《计算机科学进展》5，2016年，xx-xxx。[19] J.P. Aparicio和M. Pascual，《从基本再生数构建流行病学模型：对网络中传播的隐式处理》，《英国皇家学会学报B辑：生物科学》274，2007年，505-512。[20] L. Kong、J. Wang、W. Han和Z. Cao，《在舱室模型中对直接传染病传播的异质性进行建模》，《国际环境研究与公共卫生杂志》13，2016年，253。[21] M. Ling，《（生物）模型与仿真》，《MOJ蛋白质组学与生物信息学》3，2016年，00093。[22] S. Soliman和M. Heiner，《从常微分方程到反应网络的独特转换》，《公共科学图书馆·综合》5，2010年，文章编号e14284。[23] H.W. Hethcote和P. van den Driessche，《具有可变种群规模和时滞的易感者-感染者-易感者流行病模型》，《数学生物学杂志》34，1995年，177-194。[24] 《关于水痘的事实》，《儿科与儿童健康》10，2005年，413 414-。[25] Y. Chen、J. Yang和F. Zhang，《具有感染年龄的SIRS模型的全局稳定性》，《数学生物科学与工程》11，2014年，449-469。

附录 A：面包烘焙代码

```
导入 pnet

网络= pnet。PNet()

# 配料 net.add_库所('面粉', {'面粉': 1000}) net.add_库
```

net.add _ 0,

再易感 = 0.01

网络 = pnet.PNet() 网络.add¯库所('易感者', {'易感者

;

定义 感染者 = ¯ [ ] 康复者(库所) : 库所 库所'感染

_

者'返回 康复 + 感染者

定义 康复者 = ¯ [ ] 易感者

net.add¯rules('感染', '函数', ['易感者.易感者 -> \ 感染者.感

data[0][1]

net.模拟(500, 1, 1)

数据 = net.report¯令牌() 表头 = ['时间步'] + 数据

```
f = 打开('面包.csv', 'w')
f.写入(',',.连接(标头) + '\n')
对于 tdata 在 data: tdata = [tdata[0]] + \
```

附录 B：SIRS模型代码

```
导入 pnet
```

感染 = 0.01康复 = 0.