代码文档

如何阅读

API 文档描述了 PCSE 源代码分发中可用的所有 SimulationObjects、AncillaryObjects 和实用程序的接口和内部结构。所有 SimulationObjects 和 AncillaryObjects 都使用相同的结构进行描述：

1. 对象的简短描述
2. 接口中指定的位置参数和关键字。
3. 指定模拟所需的模拟参数的表格
4. 指定 SimulationObject 的状态变量的表
5. 指定 SimulationObject 的速率变量的表
6. SimulationObject 发送或接收的信号
7. 对其他 SimulationObjects 的状态/速率变量的外部依赖性。
8. 在什么条件下引发的异常。

当这些部分中的一个或多个不适合所描述的 SimulationObject 时，可以将它们排除在外。

指定模拟参数的表包含以下列：

1. 参数的名称。
2. 参数的描述。
3. 参数的类型。这是作为具有以下解释的三字符代码提供的。参数的第一个字符表示是标量**(S)**或表**(T)**参数。第二个和第三个字符表示此参数是否应出现在定时器数据“ **Ti** ”、作物数据“ **Cr** ”、土壤数据“ **So** ”或站点数据“ **Si** ”字典中。
4. 参数的物理单位。

指定状态/速率变量的表具有以下列：

1. 变量的名称。
2. 变量的描述。
3. 变量是否在 kiosk 中发布：Y|N
4. 变量的物理单位。

最后，还描述了所有对象的所有公共方法。

发动机和模型

PCSE 引擎提供 SimulationObjects 的“生存”环境。该引擎负责读取模型配置、初始化模型组件（例如模拟对象组）、通过调用模拟对象推动模拟、调用农业管理单元、跟踪时间并提供所需的天气数据。

模型与引擎一起处理，因为模型只是预先配置的引擎。任何模型都可以通过使用适当的配置文件启动引擎来启动。唯一的区别是模型可以有处理模型特定特征的方法。这种功能无法在引擎中实现，因为事先不知道模型细节。

班级 **pcse.engine.CGMSEngine**(*parameterprovider* , *weatherdataprovider* , *agromanagement* , *config=None*) [来源]

模拟 CGMS 行为的引擎。

原始的 CGMS 并没有在作物周期结束时终止，而是继续其模拟周期，但不改变作物和土壤成分。这样做的效果是，在作物周期结束后，所有状态变量都保持在相同的值，而天计数器增加。此行为很有用，原因有二：

1. CGMS 通常产生 dekadal 输出，当成熟日或收获日与 dekad 边界不一致时，最终模拟值仍然可用并存储在下一个 dekad。
2. 当聚合空间模拟与成熟日或收获日的可变性时，它确保记录在数据库表中可用。因此，在计算空间平均值时，SQL 查询中的 GroupBy 子句会产生正确的结果。

与 Engine 的区别在于：

1. 不支持作物轮作
2. 在 CROP_FINISH 信号之后，引擎将继续，更新计时器，但土壤、作物和农业管理将不会执行它们的模拟周期。因此，所有状态变量都将保留其值。
3. TERMINATE 信号无效。
4. CROP_FINISH 信号永远不会删除 CROP SimulationObject。
5. 不支持 run() 和 run_till_terminate()，仅支持 run_till()。

run（天=*1* ） [来源]

以给定的天数推进系统状态

run_till(日) [来源]

运行系统直到到达 rday。

run_till_terminate( ) [来源]

运行系统直到发送终止信号。

班级 **pcse.engine.Engine**(*parameterprovider* , *weatherdataprovider* , *agromanagement* , *config=None*) [来源]

用于模拟组合土壤/作物系统的模拟引擎。

参数：
- **parameterprovider** –以键/值对形式提供模型参数的*ParameterProvider*对象。parameterprovider 封装了作物、土壤和场地参数的不同参数集。
- **weatherdataprovider** – WeatherDataProvider 的一个实例，它可以在 WeatherDataContainer 中返回给定日期的天气数据。
- 农业管理——农业管理数据。数据格式在农艺管理部分描述。
- **config** – 描述要使用的模型配置文件的字符串。通过仅提供文件名，PCSE 假定它位于主 PCSE 文件夹的“conf/”文件夹中。如果您想提供自己的配置文件，请将其指定为绝对路径或相对路径（例如，以“.”开头）

引擎处理组合土壤作物系统的实际模拟。引擎的核心部分 是在整个运行过程中不断模拟的土壤水分平衡。相比之下，*CropSimulation*对象仅在从 AgroManagement 单元接收到“CROP_START”信号后才被初始化。从那时起，模拟土壤-作物组合，包括土壤和作物之间的相互作用，例如根系生长和蒸腾作用。

类似地，当接收到“CROP_FINISH”信号时，裁剪模拟完成。在那一刻，作物模拟的*finalize()*部分被执行。此外，“CROP_FINISH”信号可以指定应该从层级中删除裁剪模拟对象。后者对于进

一步扩展 PCSE 运行作物轮作非常有用。

最后，当收到“TERMINATE”信号时，整个模拟终止。那时，水平衡的*finalize()*部分被执行，模拟停止。

引擎处理的信号：

引擎处理以下信号：

- CROP_START：启动用于模拟作物生长的*CropSimulation*实例。有关详细信息，请参阅*_on_CROP_START*处理程序。
- CROP_FINISH：运行*finalize()*部分一个 *CropSimulation*实例，并可选择删除 cropsimulation 实例。有关详细信息，请参阅*_on_CROP_FINISH*处理程序。
- TERMINATE：运行 waterbalance 模块的*finalize()*部分并终止整个模拟。有关详细信息，请参阅*_on_TERMINATE*处理程序。
- 输出：在模拟期间保留所选状态/速率变量值的副本以备后用。有关详细信息，请参阅*_on_OUTPUT*处理程序。
- SUMMARY_OUTPUT：保留所选状态/速率变量值的副本供以后使用。通常仅在作物模拟结束时才需要汇总输出。有关详细信息，请参阅*_on_SUMMARY_OUTPUT*处理程序。

get_output( ) [来源]

返回模拟期间已存储的变量。

如果没有存储输出，则返回一个空列表。否则，输出将作为按时间顺序排列的字典列表返回。每个字典都是特定日期的一组存储模型变量。

get_summary_output( ) [来源]

返回在模拟期间已存储的摘要变量。

get_terminal_output( ) [来源]

返回终端输出变量在模拟过程中已经存储。

run（天=*1* ） [来源]

以给定的天数推进系统状态

run_till(日) [来源]

运行系统直到到达 rday。

run_till_terminate( ) [来源]

运行系统直到发送终止信号。

set_variable（变量名，值） [来源]

设置指定状态或速率变量的值。

参数：
- **varname** – 要更新的变量的名称（字符串）。
- **value** – 应该更新到的值（float）

退货：包含已更新变量增量的字典（新 - 旧）。如果调用未成功找到类方法（见下文），它将返回一个空字典。

请注意，“设置”变量（例如更新模型状态）比仅仅获取变量复杂得多，因为通常还必须更新一些其他内部变量（校验和、相关状态变量）。由于没有通用规则来“设置”变量，因此模型设计者需要实施适当的代码来进行更新。

*set_variable()*的实现如下。首先，它将递归地在模拟对象上搜索名为*_set_variable_<varname>*（区分大小写）的类方法。如果找到该方法，将通过提供值作为输入来调用它。

因此，为了将作物叶面积指数 (varname 'LAI') 更新为值 '5.0'，调用将是：*set_variable('LAI', 5.0)*。在内部，此调用将搜索类方法*_set_variable_LAI*，该方法将以值“5.0”作为输入执行。

班级 **pcse.models.ALCEPAS**（参数提供者、天气数据提供者、农业管理） [来源]

ALCEPAS 洋葱生长模型。

班级 **pcse.models.FAO_WRSI**（参数提供者、天气数据提供者、农业管理） [来源]

使用用水需求满意度指数和（修改后的）粮农组织 WRSI 方法计算实际作物用水量的便利类。

参数：
- **parameterprovider** – 提供所有参数值的 ParameterProvider 实例
- **weatherdataprovider** – 一个 WeatherDataProvider 对象
- 农业管理——农业管理数据

班级 **pcse.models.LINGRA_NWLP_FD**（参数提供者、天气数据提供者、农业管理） [来源]

班级 **pcse.models.LINGRA_PP**（参数提供者、天气数据提供者、农业管理） [来源]

班级 **pcse.models.LINGRA_WLP_FD**（参数提供者、天气数据提供者、农业管理） [来源]

班级 **pcse.models.LINTUL3**（参数提供者、天气数据提供者、农业管理） [来源]

LINTUL 模型（Light INTerception and UtiLisation）是一个简单的通用作物模型，它模拟干物质生产作为光拦截和利用的结果，具有恒定的光利用效率。

LINTUL3 模拟限水限氮条件下的作物生长

参数：
- **parameterprovider** –以键/值对形式提供模型参数的*ParameterProvider*对象。parameterprovider 封装了作物、土壤和场地参数的不同参数集。
- **weatherdataprovider** – WeatherDataProvider 的一个实例，它可以在 WeatherDataContainer 中返回给定日期的天气数据。
- 农业管理——农业管理数据。数据格式在农艺管理部分描述。

pcse.models.Wofost71_PP

的别名 `pcse.models.Wofost72_PP`

pcse.models.Wofost71_WLP_FD

的别名 `pcse.models.Wofost72_WLP_FD`

班级 **pcse.models.Wofost72_PP**（参数提供者、天气数据提供者、农业管理） [来源]

运行 WOFOST7.2 Potential Production 的便捷类。

参数:
- **parameterprovider** – 提供所有参数值的 ParameterProvider 实例
- **weatherdataprovider** – 一个 WeatherDataProvider 对象
- 农业管理——农业管理数据

班级 **pcse.models.Wofost72_Phenology**（参数提供者、天气数据提供者、农业管理） [来源]

仅用于运行 WOFOST7.2 物候的便捷类。

参数:
- **parameterprovider** – 提供所有参数值的 ParameterProvider 实例
- **weatherdataprovider** – 一个 WeatherDataProvider 对象
- 农业管理——农业管理数据

班级 **pcse.models.Wofost72_WLP_FD**（参数提供者、天气数据提供者、农业管理） [来源]

运行WOFOST7.2限水制作的便利类。

参数:
- **parameterprovider** – 提供所有参数值的 ParameterProvider 实例
- **weatherdataprovider** – 一个 WeatherDataProvider 对象
- 农业管理——农业管理数据

班级 **pcse.models.Wofost80_NWLP_FD_beta**（参数提供者、天气数据提供者、农业管理） [来源]

运行WOFOST8.0限养限水生产的便捷类

参数:
- **parameterprovider** – 提供所有参数值的 ParameterProvider 实例
- **weatherdataprovider** – 一个 WeatherDataProvider 对象
- 农业管理——农业管理数据

班级 **pcse.models.Wofost80_PP_beta**（参数提供者、天气数据提供者、农业管理） [来源]

运行WOFOST8.0势产的便捷类（包括NPK动力学）

参数:
- **parameterprovider** – 提供所有参数值的 ParameterProvider 实例
- **weatherdataprovider** – 一个 WeatherDataProvider 对象
- 农业管理——农业管理数据

班级 **pcse.models.Wofost80_WLP_FD_beta**（参数提供者、天气数据提供者、农业管理） [来源]

运行WOFOST8.0限水制作的便捷类（包含NPK动态）

参数:
- **parameterprovider** – 提供所有参数值的 ParameterProvider 实例
- **weatherdataprovider** – 一个 WeatherDataProvider 对象
- 农业管理——农业管理数据

农业管理模块

下面的例程在 PCSE 中实现农业管理系统，包括作物日历、轮作、状态和定时事件。要从文件或数据库结构中读取农业管理数据，请参阅有关读取文件输入和数据库工具的部分。

班级 **pcse.agromanager.AgroManager**（信息亭，***args***，****kwargs***） [来源]

连续农业管理行动的类别，包括作物轮作和事件。

另请参阅*CropCalendar*、*TimedEventDispatcher*和*StateEventDispatcher*类的文档。

AgroManager 负责执行通常发生在农田上的农业代理操作，包括农作物的种植和收获，以及管理操作，例如施肥、灌溉、割草和喷洒。

模拟期间的农业管理是作为一系列活动实施的。活动在规定的日历日期开始，并在下一次活动开始时结束。模拟通过提供一个尾随的空活动或通过从作物日历和最后一个活动中的定时事件推导出结束日期来明确结束。另请参阅下面有关*end_date*属性的部分。

每个活动的特点是零个或一个作物日历、零个或多个定时事件以及零个或多个状态事件。通过下面的示例（在 YAML 中）最容易理解作为 AgroManager 输入所需的数据结构。该定义由三个活动组成，第一个活动从 1999-08-01 开始，第二个活动从 2000-09-01 开始，最后一个活动从 2001-03-01 开始。第一个活动包括从给定的 crop_start_date 开始播种的冬小麦作物日历。在活动期间，在 2000-05-25 和 2000-06-30 有灌溉的定时事件。此外，在 DVS 0.3、0.6 和 1.12 处，还有发育阶段 (DVS) 给出的施肥状态事件 (event_signal: apply_npk)。

第二个活动没有作物日历、定时事件或状态事件。这意味着这是一个只有水平衡运行的裸土时期。第三个活动是针对 2001 年 4 月 15 日播种的饲料玉米，有两个系列的定时事件（一个用于灌溉，一个用于 N/P/K 应用），没有状态事件。在这种情况下，模拟的结束日期为 2001-11-01（2001-04-15 + 200 天）。

农业管理定义文件的示例：

```
AgroManagement:
- 1999-08-01:
    CropCalendar:
        crop_name: wheat
        variety_name: winter-wheat
        crop_start_date: 1999-09-15
        crop_start_type: sowing
        crop_end_date:
        crop_end_type: maturity
        max_duration: 300
    TimedEvents:
    -   event_signal: irrigate
        name:  Timed irrigation events
        comment: All irrigation amounts in cm
        events_table:
        - 2000-05-25: {irrigation_amount: 3.0}
        - 2000-06-30: {irrigation_amount: 2.5}
    StateEvents:
    -   event_signal: apply_npk
        event_state: DVS
        zero_condition: rising
        name: DVS-based N/P/K application table
        comment: all fertilizer amounts in kg/ha
        events_table:
        - 0.3: {N_amount : 1, P_amount: 3, K_amount: 4}
        - 0.6: {N_amount: 11, P_amount: 13, K_amount: 14}
        - 1.12: {N_amount: 21, P_amount: 23, K_amount: 24}
- 2000-09-01:
    CropCalendar:
    TimedEvents:
    StateEvents
- 2001-03-01:
    CropCalendar:
        crop_name: maize
        variety_name: fodder-maize
        crop_start_date: 2001-04-15
```

```
        crop_start_type: sowing
        crop_end_date:
        crop_end_type: maturity
        max_duration: 200
    TimedEvents:
    -   event_signal: irrigate
        name:  Timed irrigation events
        comment: All irrigation amounts in cm
        events_table:
        - 2001-06-01: {irrigation_amount: 2.0}
        - 2001-07-21: {irrigation_amount: 5.0}
        - 2001-08-18: {irrigation_amount: 3.0}
        - 2001-09-19: {irrigation_amount: 2.5}
    -   event_signal: apply_npk
        name:  Timed N/P/K application table
        comment: All fertilizer amounts in kg/ha
        events_table:
        - 2001-05-25: {N_amount : 50, P_amount: 25, K_amount: 22}
        - 2001-07-05: {N_amount : 70, P_amount: 35, K_amount: 32}
    StateEvents:
```

end_date

检索农业管理序列的结束日期，例如最后的模拟日期。

退货： 日期对象

获取上次模拟日期比较复杂，因为有两个选项。

1. 添加一个明确的尾随空活动

第一个选项是通过在农业管理定义中添加“尾随空活动”来明确定义模拟的结束日期。下面给出了一个带有“尾随空活动”（YAML 格式）的农业管理定义示例。此示例将运行模拟直到 2001 年 1 月 1 日：

```
Version: 1.0
AgroManagement:
- 1999-08-01:
    CropCalendar:
        crop_name: winter-wheat
        variety_name: winter-wheat
        crop_start_date: 1999-09-15
        crop_start_type: sowing
        crop_end_date:
        crop_end_type: maturity
        max_duration: 300
    TimedEvents:
    StateEvents:
- 2001-01-01:
```

请注意，在最后一个活动包含状态事件定义的配置中，必须提供尾随的空活动，因为无法确定结束日期。因此，以下活动定义将导致错误：

```
Version: 1.0
AgroManagement:
- 2001-01-01:
    CropCalendar:
        crop_name: maize
        variety_name: fodder-maize
        crop_start_date: 2001-04-15
        crop_start_type: sowing
        crop_end_date:
        crop_end_type: maturity
        max_duration: 200
    TimedEvents:
    StateEvents:
    -   event_signal: apply_npk
```

```
        event_state: DVS
        zero_condition: rising
        name: DVS-based N/P/K application table
        comment: all fertilizer amounts in kg/ha
        events_table:
        - 0.3: {N_amount : 1, P_amount: 3, K_amount: 4}
        - 0.6: {N_amount: 11, P_amount: 13, K_amount: 14}
        - 1.12: {N_amount: 21, P_amount: 23, K_amount: 24}
```

2. 没有明确的尾随活动

第二个选项是没有尾随的空活动，在这种情况下，模拟的结束日期是从作物日历和/或计划的定时事件中检索的。在下面的示例中，结束日期将为 2000 年 8 月 5 日，因为这是收获日期，并且在此日期之后没有安排任何定时事件：

```
Version: 1.0
AgroManagement:
- 1999-09-01:
    CropCalendar:
        crop_name: wheat
        variety_name: winter-wheat
        crop_start_date: 1999-10-01
        crop_start_type: sowing
        crop_end_date: 2000-08-05
        crop_end_type: harvest
        max_duration: 330
    TimedEvents:
    -   event_signal: irrigate
        name:  Timed irrigation events
        comment: All irrigation amounts in cm
        events_table:
        - 2000-05-01: {irrigation_amount: 2, efficiency: 0.7}
        - 2000-06-21: {irrigation_amount: 5, efficiency: 0.7}
        - 2000-07-18: {irrigation_amount: 3, efficiency: 0.7}
    StateEvents:
```

在没有提供收获日期且作物一直到成熟的情况下，作物日历的结束日期将估计为 crop_start_date 加上 max_duration。

initialize（售货亭，农业管理） [来源]

初始化农业管理器。

参数：
- **kiosk** – PCSE 变量 Kiosk
- **agromanagement** – agromanagement 定义，请参阅上面 YAML 中的示例。

ndays_in_crop_cycle

返回当前裁剪周期的天数。

如果没有作物周期处于活动状态，则返回零。

start_date

检索农业管理序列的开始日期，例如第一个模拟日期

退货： 日期对象

班级
pcse.agromanager.CropCalendar（*kiosk*，*crop_name=None*，*variety_name=None*，*crop_start_date=None*，*crop_start_type=None*，*crop_end_date=None*，*crop_end_type=None*，*max_duration=None*） [来源]

用于管理作物周期的作物日历。

CropCalendar对象负责存储、检查、开始和结束作物周期。通过提供定义作物周期所需的参数来初始化作物日历。在每个时间步，调用*CropCalendar*的实例，并在其参数定义的日期启动适当的操作：

- 播种/出现：发送*crop_start*信号，包括启动新作物模拟对象所需的参数
- 成熟/收获：通过发送带有适当参数的*crop_finish*信号结束作物周期。

参数：
- **kiosk** – PCSE VariableKiosk 实例
- **crop_name** – 识别作物的字符串
- **variety_name** – 标识品种的字符串
- **crop_start_date** – 作物模拟的开始日期
- **crop_start_type** – 作物模拟的开始类型（“播种”、“出苗”）
- **crop_end_date** – 作物模拟的结束日期
- **crop_end_type** – 作物模拟的结束类型（'harvest'、'maturity'、'earliest'）
- **max_duration** – 描述作物周期最长持续时间的整数

退货： CropCalendar 实例

get_end_date( ) [来源]

返回作物周期的结束日期。

这可以作为收获日期给出或计算为 crop_start_date + max_duration

退货： 日期对象

get_start_date( ) [来源]

返回循环的开始日期。这始终是 self.crop_start_date

退货： 开始日期

validate(***campaign_start_date*** **,** ***next_campaign_start_date*** **)** [来源]

根据农业活动的间隔在内部验证作物日历。

参数：
- **campaign_start_date** – 此活动的开始日期
- **next_campaign_start_date** – 下一个活动的开始日期

班级 **pcse.agromanager.TimedEventsDispatcher**（信息亭、事件信号、名称、评论、事件表） [来源]

小心处理与日期相关的事件。

事件通过发送信号（取自信号模块）并为信号提供相关参数来处理。查看 agromanagement 文件中的定义时，最容易理解 TimedEvents。以下部分（在 YAML 中）提供了 TimedEventsDispatchers 的两个实例的定义：

```
TimedEvents:
-   event_signal: irrigate
    name:  Timed irrigation events
    comment: All irrigation amounts in mm
    events_table:
```

```
    - 2000-01-01: {irrigation_amount: 20}
    - 2000-01-21: {irrigation_amount: 50}
    - 2000-03-18: {irrigation_amount: 30}
    - 2000-03-19: {irrigation_amount: 25}
-   event_signal: apply_npk
    name:  Timed N/P/K application table
    comment: All fertilizer amounts in kg/ha
    events_table:
    - 2000-01-10: {N_amount : 10, P_amount: 5, K_amount: 2}
    - 2000-01-31: {N_amount : 30, P_amount: 15, K_amount: 12}
    - 2000-03-25: {N_amount : 50, P_amount: 25, K_amount: 22}
    - 2000-04-05: {N_amount : 70, P_amount: 35, K_amount: 32}
```

每个 TimedEventDispatcher 都由一个*event_signal*、一个可选名称、一个可选注释和 events_table 定义。events_table 是一个列表，它为每个日期提供了应该用给定的 event_signal 发送的参数。

get_end_date( ) [来源]

返回给出定时事件的最后日期

validate(*campaign_start_date* , *next_campaign_start_date*) [来源]

在给定活动窗口的情况下验证定时事件

参数：
- **campaign_start_date** – 活动的开始日期
- **next_campaign_start_date** – 下一个活动的开始日期，可以是 None

班级 **pcse.agromanager.StateEventsDispatcher**(*kiosk* , *event_signal* , *event_state* , *zero_condition* , *name* , *comment* , *events_table*) [来源]

小心处理连接到模型状态变量的事件。

事件通过发送信号（取自信号模块）并为信号提供相关参数来处理。查看 agromanagement 文件中的定义时，StateEvents 最容易理解。以下部分（在 YAML 中）提供了 StateEventsDispatchers 的两个实例的定义：

```
StateEvents:
-   event_signal: apply_npk
    event_state: DVS
    zero_condition: rising
    name: DVS-based N/P/K application table
    comment: all fertilizer amounts in kg/ha
    events_table:
    - 0.3: {N_amount : 1, P_amount: 3, K_amount: 4}
    - 0.6: {N_amount: 11, P_amount: 13, K_amount: 14}
    - 1.12: {N_amount: 21, P_amount: 23, K_amount: 24}
-   event_signal: irrigate
    event_state: SM
    zero_condition: falling
    name: Soil moisture driven irrigation scheduling
    comment: all irrigation amounts in cm of water
    events_table:
    - 0.15: {irrigation_amount: 20}
```

每个 StateEventDispatcher 都由一个*event_signal*、一个*event_state*（例如触发事件的模型状态）和一个零条件定义。此外，可以提供可选的名称和可选的注释。最后，events_table 指定事件发生时的模型状态值。events_table 是一个列表，它为每个状态提供了应该用给定的 event_signal 调度的参数。

为了找到状态事件发生的时间步长，PCSE 使用过零的概念。这意味着当 (*model_state* - *event_state*) zero_condition

等于或过零时触发状态事件。 定义了这种交叉应该如何发生。*zero_condition*的值 可以是：

- ***rising***：当（***model_state - event_state***）从负值变为负值时触发事件

  零或正值。

- ***falling***：当（***model_state - event_state***）从正值变为正值时触发事件

  零或负值。

- 任一个：当（***model_state - event_state***）从任意值越过或达到零时触发事件

  方向。

可以使用上面的示例定义来说明 zero_condition 的影响。作物的发育阶段 (DVS) 仅从出苗时的 0 增加到成熟时的 2。因此，在 DVS 上设置的 StateEvent（第一个示例）逻辑上将具有 zero_condition“上升”，尽管也可以使用“任一”。zero_condition 设置为“下降”时不会发生基于 DVS 的事件，因为 DVS 的值不会减少。

然而，土壤水分 (SM) 既可以增加也可以减少。因此，用于应用灌溉（第二个示例）的 StateEvent 将使用 zero_condition 'falling' 指定，因为当土壤湿度水平达到或超过 events_table 指定的最低水平时必须触发该事件。请注意，如果我们将 zero_condition 设置为“任一个”，则事件可能会在下一个时间步再次发生，因为灌溉量增加了土壤水分并且（*model_state - event_state*）再次从另一个方向越过零。

计时器

班级 **pcse.timer.Timer**（信息亭，***args**，****kwargs** ） [来源]

此类实现了一个用于 WOFOST 裁剪模型的基本计时器。

该对象实现了一个简单的计时器，该计时器在每次调用时以一天的固定时间步长递增当前时间并返回其值。此外，它以每日、十次或每月的时间步长生成输出信号，可以捕获这些信号以存储模拟状态以备后用。

初始化定时器：

```
timer = Timer(start_date, kiosk, final_date, mconf)
CurrentDate = timer()
```

发送或处理的信号：

- “OUTPUT”：当生成输出的条件为真时发送，这取决于输出类型和间隔。

initialize(*kiosk* , *start_date* , *end_date* , *mconf*) [来源]

参数：

- 天——模拟的开始日期
- **kiosk** – PCSE 实例的可变信息亭
- **end_date** – 模拟的最终日期。例如，此日期表示单个种植季节的 (START_DATE + MAX_DURATION)。该日期不是收获日期，因为收获信号由*AgroManagement*模块处理。

mconf – 一个 ConfigurationLoader 对象，计时器需要访问配置属性 mconf.OUTPUT_INTERVAL、mconf.OUTPUT_VARS 和 mconf.OUTPUT_INTERVAL_DAYS

水平衡

PCSE 发行版提供了几个水平衡模块：

1. 非限水生产模拟WaterbalancePP
2. WaterbalanceFD，用于模拟自由排水土壤条件下的限水生产
3. 用于模拟积雪覆盖和融化的SnowMAUS 。
4. 对潜在条件、限水自由排水条件和限水地下水条件（在浅层地下水位的情况下）实施模拟的多层水平衡。此水平衡处于原型阶段，尚未可用，但源代码可在 PCSE 中获得。

班级 **pcse.soil.WaterbalancePP**（天，信息亭，***args***，****kwargs***） [来源]

模拟潜在生产条件下的假水平衡。

将土壤水分含量保持在田间持水量，并且仅通过模拟过程累积作物蒸腾和土壤蒸发率

班级 **pcse.soil.WaterbalanceFD**（天，信息亭，***args***，****kwargs***） [来源]

限水生产下自由排水土壤的水平衡。

土壤水分平衡计算的目的是估算土壤水分含量的日值。土壤含水量影响土壤水分吸收和作物蒸腾。

动态计算分两部分进行，一部分用于计算每个时间步长（= 1 天）的变化率，另一部分用于计算求和变量和状态变量。水平衡由降雨驱动，可能作为地表蓄水和蒸发蒸腾进行缓冲。考虑的过程是渗透、土壤保水、渗滤（这里被认为是水从根区向下流到第二层），以及最大根区以外的水分流失。

土壤的结构剖面被认为是均匀的。最初，土壤剖面由两层组成，实际生根的土壤和紧邻生根区下方的土壤，直到根部达到最大生根深度（取决于土壤和作物）。Root_Dynamics 类中描述了根区从初始生根深度到最大生根深度的扩展。从达到最大根深的那一刻起，土壤剖面可以被描述为单层系统，这取决于根是否能够穿透整个剖面。如果不是，则非根部分保留在配置文件的底部。

WaterbalanceFD 类派生自 WOFOST7.1 中的 WATFD.FOR，但土壤深度现在完全由最大土壤深度 (RDMSOL) 决定，而不是由最小土壤深度和作物最大根深 (RDMCR) 决定.

仿真参数：

姓名	描述	类型	单元
SMFCF	土壤的田间持水量	SSO	•
SM0	土壤的孔隙率	SSO	•
SMW	土壤枯萎点	SSO	•
CRAIRC	土壤临界含气量（水涝）	SSO	•
索普	最大渗透率根区	SSO	$cmday^{-1}$

KSUB	最大渗透率底土	SSO	$cmday^{-1}$
RMSOL	土壤可生根深度	SSO	厘米
干扰函数	指示雨的非渗透部分是风暴大小 (1) 还是 (0) 的函数	不锈钢	•
SSMAX	最大表面存储	不锈钢	厘米
SSI	初始表面存储	不锈钢	厘米
WAV格式	总土壤剖面中的初始水量	不锈钢	厘米
通知	雨水未渗入土壤的最大比例	不锈钢	•
SMLIM	初始生根深度区的初始最大含水量。	不锈钢	•

状态变量：

姓名	描述
SM	根区体积水分含量
党卫军	表面储存（表面水层）
SSI	初始表面存储
W	根区水量
无线网	根区初始水量
WLOW	底土中的水量（当前生根深度和最大可生根深度之间）
WLOI	底土中的初始水量
万维网	土壤剖面中的总水量 WWLOW = WLOW + W
WTRAT	根据水平衡计算的蒸腾损失的总水量。这可能不同于 CTRAT 变量，后者仅计算作物周期
EVST	土壤表面的总蒸发量
EVWT	水面的总蒸发量
总回报率	总地表径流
雨天	总降雨量（有效 + 无效）
WDRT	通过增加根系生长而添加到根区的水量
TOTINF	渗透总量
托蒂尔	有效灌溉总量

PERCT	从根区渗透到底土的水总量
丢失	流失到深层土壤的水总量
数字操作系统	氧应激后的天数，累积氧应激的连续天数
WBALRT	根区水平衡校验和。将在*finalize()*内计算，abs(WBALRT) > 0.0001 将引发 WaterBalanc
WBALTT	总水平衡的校验和。将在*finalize()*内计算，abs(WBALTT) > 0.0001 将引发 WaterBalanc

速率变量：

外部依赖：

姓名	描述	由...提供	单元
运输署	作物蒸腾速率	蒸散量	cmday^{-1}
EVSMX	作物冠层以下土壤表面的最大蒸发率	蒸散量	cmday^{-1}
EVWMX	作物冠层以下水面的最大蒸发率	蒸散量	cmday^{-1}
研发	生根深度	Root_dynamics	厘米

引发的异常：

当水平衡在模拟循环结束时没有关闭时（例如，水已经“漏”掉），会引发 WaterbalanceError。

班级 **pcse.soil.SnowMAUS**（天，信息亭，***args***，****kwargs***） [来源]

用于农业气象应用的简单积雪模型。

这是 SnowMAUS 模型的一个实现，该模型描述了降水、融雪和升华导致的雪的积累和融化。SnowMAUS 模型旨在跟踪表面上以雪形式存在的水层厚度，例如雪水等效深度（状态变量 SWEDEPTH [cm]）。将 SWEDEPTH 转换为实际雪深（状态变量 SNOWDEPTH [cm]）是通过将 SWEDEPTH 除以 [cm_water/cm_snow] 中的雪密度来完成的。

尽管已知雪密度会随着降雪类型、温度和积雪的年龄而变化，但雪密度被视为固定值。然而，更复杂的雪密度算法与 SnowMAUS 的简单性不一致。

当前实施的一个缺点是还没有与水平衡的联系。

参考资料：M. Trnka、E. Kocmánková、J. Balek、J. Eitzinger、F. Ruget、H. Formayer、P. Hlavinka、A. Schaumberger、V. Horáková、M. Možný、Z. Žalud，简单积雪模型用于农业气象应用，农业和森林气象学，第 150 卷，第 7-8 期，2010 年 7 月 15 日，第 1115-1127 页，ISSN 0168-1923

http://dx.doi.org/10.1016/j.agrformet.2010.04.012

模拟参数：（在crop、soil和sitedata字典中提供）

姓名	描述	类型	单元

TMINACCU1	积雪的上临界最低温度。	不锈钢	°C
TMINACCU2	积雪临界最低温度下限	不锈钢	°C
TMIN暴击	雪融化的临界最低温度	不锈钢	°C
极限暴击	雪融化的临界最高温度	不锈钢	°C
熔化	高于临界最低温度的每天每摄氏度的熔化速率。	不锈钢	$cm\,°C^{-1}\,day^{-1}$
阈值	考虑升华的雪水当量。	不锈钢	厘米
雪密度	雪的密度	不锈钢	厘米/厘米
瑞典人	土壤表面以雪形式存在的水层的初始深度	不锈钢	厘米

状态变量：

姓名	描述	铅	单元
扫一扫	表面上以雪形式存在的水层深度	否	厘米
雪深	表面积雪深度。	是	厘米

速率变量：

姓名	描述	铅	单元
RSNOWACCUM	积雪率	否	$cmday^{-1}$
RSNOWSUBLIM	雪升华率	否	$cmday^{-1}$
雪融	融雪速度	否	$cmday^{-1}$

WOFOST 的作物模拟过程

物候学

班级 **pcse.crop.phenology.DVS_Phenology**（天，信息亭，***args***，****kwargs***） [来源]

在 WOFOST 中实现物候发育算法。

WOFOST 中的物候发育使用无单位标度表示，出苗时取值为 0，开花时取值为 1，成熟时取值为 2。这种物候发育主要以谷类作物为代表。使用 WOFOST 模拟的所有其他作物也被迫采用此方案，尽管这可能并不适合所有作物。例如，对于马铃薯，发育阶段 1 代表块茎形成的开始而不是开花。

物候发育主要受温度控制，可以通过开花前的日照长度和春化作用进行修改。开花后，只有温度影响发育速度。

仿真参数

姓名	描述	
苏门	播种到出苗温度总和	
巴塞尔	出苗基准温度	
TEFFMX	出苗最高有效温度	
TSUM1	从出苗到开花的温度总和	
TSUM2	开花到成熟的温度总和	
国际用户线	仅物候发展选项温度 (IDSL=0) 的开关，包括日长 (IDSL=1) 和春化 (IDSL>=2)	
DLO	物候发育的最佳日长	
可下载内容	物候发育的临界日长	
DVSI	出现时的初始发育阶段。通常这是零，但对于移植的作物（例如水稻），它可能更高	
DVS发送	最终开发阶段	
DTSMTB	温度总和的每日增加量作为每日平均温度的函数。	

状态变量

姓名	描述	铅	
数字视频服务器	发展阶段	是	
TSUM	温度总和	否	
TSUME	出苗温度总和	否	
操作系统	播种日	否	
美国能源部	出现日	否	
DOA	开花日	否	
DOM	成熟日	否	
卫生部	收获之日	否	
阶段	当前物候阶段，可以取以下值： *emerging*|*vegetative*|*reproductive*|*mature*	否	

速率变量

姓名	描述	铅	单元
大同	出现温度总和增加	否	°C
DTSUM	开花期或成熟期温度总和的增加	否	°C
硬盘录像机	发展速度	是	第$^{-1}$天

外部依赖：

没有任何

发送或处理的信号

*DVS_Phenology*在达到成熟度时发送*crop_finish*信号并且*end_type*为“成熟”或“最早”。

班级 **pcse.crop.phenology.Vernalisation**（天，信息亭，***args**，****kwargs**） [来源]

由于春化作用改变了物候发育。

这里的春化方法基于 Lenny van Bussel (2011) 的工作，后者又基于 Wang 和 Engel (1998)。基本原理是冬小麦需要一定天数，温度在最适温度范围内才能完成春化要求。在满足春化要求之前，作物发育会延迟。

春化率 (VERNR) 由温度响应函数 VERNRTB 定义。在最佳温度范围内，春化状态 (VERN) 增加 1 天。物候发育的减少是根据基础和饱和春化要求（VERNBASE 和 VERNSAT）计算的。缩减系数 (VERNFAC) 在 VERNBASE 和 VERNSAT 之间线性缩放。

当达到此 DVS 时，使用关键发育阶段 (VERNDVS) 来停止春化作用。这样做是为了提高模型稳定性，以避免由于 VERNSAT 太高而永远无法达到开花期。尽管如此，如果发生这种情况，则会将警告写入日志文件。

- Van Bussel，2011 年。从田间到全球：作物生长模型的升级。瓦赫宁根博士论文。http://edepot.wur.nl/180295
- Wang 和 Engel，1998。小麦作物物候发育的模拟。农业。系统 58:1 第 1-24 页

模拟参数（在 cropdata 字典中提供）

姓名	描述	类型	单元
卫星	饱和春化要求	氯化铬	天
文库	基础春化要求	氯化铬	天
VERNRTB	作为日平均温度函数的春化率。	总铬	•
维纳斯	春化效果停止后的关键发育阶段	氯化铬	•

状态变量

姓名	描述	铅	单元
维尔恩	春化状态	否	天
DOV	满足春化要求的日子。	否	•
国际化	标志表示已达到春化要求	是	•

速率变量

姓名	描述	铅	单元
VERNR	春化率	否	•
维纳克	春化作用导致的发育率降低因子。	是	•

外部依赖：

姓名	描述	由…提供
数字视频服务器	发育阶段 仅用于确定是否达到春化作用的关键发育阶段 (VERNDVS)。	物候学

分区

班级 `pcse.crop.partitioning.DVS_Partitioning`（天，信息亭，***args***，****kwargs***） [来源]

基于开发阶段的同化分区类 (*DVS*)。

DVS_partioning calculates the partitioning of the assimilates to roots, stems, leaves and storage organs using fixed partitioning tables as a function of crop development stage. The available assimilates are first split into below-ground and abovegrond using the values in FRTB. In a second stage they are split into leaves (*FLTB*), stems (*FSTB*) and storage organs (*FOTB*).

Since the partitioning fractions are derived from the state variable *DVS* they are regarded state variables as well.

Simulation parameters (To be provided in cropdata dictionary):

Name	**Description**	**Type**	**Unit**
FRTB	Partitioning to roots as a function of development stage.	TCr	•

FSTB	Partitioning to stems as a function of development stage.	TCr	•
FLTB	根据发育阶段划分叶子。	总铬	•
FOTB	根据发育阶段划分贮藏器官。	总铬	•

状态变量

姓名	描述	铅	单元
FR	分数划分为根。	是	•
FS	分数分配到茎。	是	•
佛罗里达州	部分分配给叶子。	是	•
FO	分数划分为存储器官	是	•

速率变量

没有任何

发送或处理的信号

没有任何

外部依赖：

姓名	描述	由...提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•

引发异常

如果给定日期的叶、茎和储存器官的分配系数加起来不等于“1”，则会引发 PartitioningError。

CO_2同化

班级 **pcse.crop.assimilation.WOFOST_Assimilation**（天，信息亭，***args***，****kwargs***） [来源]

实现 WOFOST/SUCROS 样式同化例程的类。

WOFOST 根据吸收的辐射和单片叶子的光合作用-光响应曲线计算作物每天的总 CO_2同化率。这种反应取决于温度和叶龄。吸收的辐射是根据总入射辐射和叶面积计算的。日总 CO_2同化是通过整合叶层和全天的同化率获得的。

仿真参数

姓名	描述	类型	单元
AMAXTB	最大限度。叶 CO 2吸收。速率作为 DVS 的函数	总铭	公斤 公顷$^{-1}$小时$^{-1}$
EFTB	光使用效果。单叶作为日平均温度的函数	总铭	kg ha $^{-1}$ hr $^{-1}$ /(J m $^{-2}$ sec $^{-1}$)
KDIFTB	作为 DVS 函数的漫反射可见光消光系数	总铭	•
TMPFTB	AMAX 的折减系数作为日平均温度的函数。	总铭	•
TMNFTB	AMAX 的减少因子作为日最低温度的函数。	总铭	•

状态和速率变量

*WOFOST_Assimilation*直接从*__call__()*方法返回潜在总同化率“PGASS”，但也将其作为速率变量包含在内。

速率变量：

姓名	描述	铅	单元
PGASS	潜在同化率	否	kg CH 2 O ha $^{-1}$天$^{-1}$

发送或处理的信号

没有任何

外部依赖：

姓名	描述	由...提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•
赖	叶面积指数	叶动力学	•

维持呼吸

班级 **pcse.crop.respiration.WOFOST_Maintenance_Respiration**（天，信息亭，***args***，****kwargs***）
[来源]

WOFOST 中的维持呼吸

WOFOST calculates the maintenance respiration as proportional to the dry weights of the plant organs to be maintained, where each plant organ can be assigned a different maintenance coefficient. Multiplying organ weight with the maintenance coeffients yields the relative maintenance respiration (*RMRES*) which is than corrected for senescence (parameter

RFSETB). Finally, the actual maintenance respiration rate is calculated using the daily mean temperature, assuming a relative increase for each 10 degrees increase in temperature as defined by *Q10*.

Simulation parameters: (To be provided in cropdata dictionary):

Name	Description
Q10	Relative increase in maintenance repiration rate with each 10 degrees increase in temperature
RMR	Relative maintenance respiration rate for roots
RMS	Relative maintenance respiration rate for stems
RML	Relative maintenance respiration rate for leaves
RMO	Relative maintenance respiration rate for storage organs

State and rate variables:

> ***WOFOSTMaintenanceRespiration* returns the potential maintenance respiration PMRES**

directly from the *__call__()* method, but also includes it as a rate variable within the object.

Rate variables:

Name	Description	Pbl	Unit
PMRES	Potential maintenance respiration rate	N	kg CH_2O $ha^{-1}day^{-1}$

Signals send or handled

None

External dependencies:

Name	Description	Provided by	Unit
DVS	Crop development stage	DVS_Phenology	-
WRT	Dry weight of living roots	WOFOST_Root_Dynamics	kg ha^{-1}
WST	Dry weight of living stems	WOFOST_Stem_Dynamics	kg ha^{-1}
WLV	Dry weight of living leaves	WOFOST_Leaf_Dynamics	kg ha^{-1}
WSO	Dry weight of living storage organs	WOFOST_Storage_Organ_Dynamics	kg ha^{-1}

Evapotranspiration

class **pcse.crop.evapotranspiration.Evapotranspiration**(*day*, *kiosk*, **args*, ***kwargs*) [source]

Calculation of potential evaporation (water and soil) rates and actual crop transpiration rate.

Simulation parameters:

Name	Description	Type	Unit
CFET	Correction factor for potential transpiration rate.	SCr	•
DEPNR	Dependency number for crop sensitivity to soil moisture stress.	SCr	•
KDIFTB	Extinction coefficient for diffuse visible as function of DVS.	TCr	•
IOX	Switch oxygen stress on (1) or off (0)	SCr	•
IAIRDU	Switch airducts on (1) or off (0)	SCr	•
CRAIRC	Critical air content for root aeration	SSo	•
SM0	Soil porosity	SSo	•
SMW	Volumetric soil moisture content at wilting point	SSo	•
SMCFC	Volumetric soil moisture content at field capacity	SSo	•
SM0	Soil porosity	SSo	•

State variables

Note that these state variables are only assigned after finalize() has been run.

Name	Description	Pbl	Unit
IDWST	Nr of days with water stress.	N	•
IDOST	Nr of days with oxygen stress.	N	•

Rate variables

Name	Description	Pbl	Unit
EVWMX	Maximum evaporation rate from an open water surface.	Y	cm day^{-1}
EVSMX	Maximum evaporation rate from a wet soil surface.	Y	cm day^{-1}
			$^{-1}$

TRAMX	Maximum transpiration rate from the plant canopy	Y	cm day
TRA	Actual transpiration rate from the plant canopy	Y	cm day^{-1}
IDOS	Indicates oxygen stress on this day (True|False)	N	•
IDWS	Indicates water stress on this day (True|False)	N	•
RFWS	Reduction factor for water stress	N	•
RFOS	Reduction factor for oxygen stress	N	•
RFTRA	Reduction factor for transpiration (wat & ox)	Y	•

Signals send or handled

None

External dependencies:

Name	Description	Provided by	Unit
DVS	Crop development stage	DVS_Phenology	•
LAI	叶面积指数	叶动力学	•
SM	体积土壤含水量	水平衡	•

pcse.crop.evapotranspiration.SWEAF（*ET0*，*DEPNR*） [来源]

计算土壤水易利用分数 (SWEAF)。

参数：
- **ET0** – 参考作物的蒸散量。
- **DEPNR** – 作物依存度数。

田间持水量和枯萎点之间容易获得的土壤水分的比例是潜在蒸散率（对于封闭的冠层）的函数，单位为厘米/天，ET0 和作物组数，DEPNR（从 1（=干旱敏感）至 5（=抗旱））。函数 SWEAF 描述了由 Doorenbos & Kassam (1979) 和 Van Keulen & Wolf (1986; p.108, table 20) http://edepot.wur.nl/168025以表格形式给出的这种关系 。

叶片动力学

班级 **pcse.crop.leaf_dynamics.WOFOST_Leaf_Dynamics**（天，信息亭，**args*，***kwargs*） [来源]

WOFOST 作物模型的叶动力学。

对叶片进行生物量分配，叶片的生长和衰老。WOFOST 跟踪每天分配给叶子的生物量（变量*LV*，称为叶类）。对于每个叶类，还记录了叶龄（变量“LVAGE”）和特定叶面积（变

量*SLA*）。总活叶生物量是通过对所有叶类的生物量值求和来计算的。同样，叶面积的计算方法是将叶生物量乘以比叶面积 (*LV* * *SLA*)。

叶片衰老可能是生理年龄、干旱胁迫或自我遮荫的结果。

模拟参数（在 cropdata 字典中提供）

姓名	描述	类型	单元
RGRLAI	LAI 的最大相对增加。	氯化铬	哈哈-1 d-1
跨度	树叶在35摄氏度生长的寿命	氯化铬	天
TBASE	下限温度 用于叶子老化	氯化铬	°C
PERDL	最大限度。水分胁迫引起的叶片相对死亡率	氯化铬	
TDWI	初始作物总干重	氯化铬	公斤公顷$^{-1}$
KDIFTB	漫射可见光的消光系数作为 DVS 的函数	总铬	
SLATB	作为 DVS 函数的比叶面积	总铬	公顷千克$^{-1}$

状态变量

姓名	描述	铅	单元
低压	每叶类的叶生物量	否	公斤公顷$^{-1}$
服务水平协议	每类叶的比叶面积	否	公顷千克$^{-1}$
领航	每个叶级的叶龄	否	天
LVSUM	LV总和	否	公斤公顷$^{-1}$
莱姆	出现时的 LAI	否	•
拉苏姆	总叶面积作为 LV*SLA 的总和，不包括茎和豆荚面积	否	•
来世	理论指数增长下的 LAI 值	否	•
莱美思	生长周期内达到的最大 LAI	否	•
赖	叶面积指数，包括茎和荚面积	是	•
WLV	活叶干重	是	公斤公顷$^{-1}$
DWLV	枯叶干重	否	公斤公顷$^{-1}$
TWLV	总叶干重（活+死）	是	公斤公顷$^{-1}$

速率变量

姓名	描述	铅	单元
GRLV	叶生长率	否	公斤 公顷$^{-1}$天
DSLV1	水分压力导致死亡率下降	否	公斤 公顷$^{-1}$天
DSLV2	死亡率因自遮蔽而离开	否	公斤 公顷$^{-1}$天
DSLV3	霜冻导致死亡率下降	否	公斤 公顷$^{-1}$天
DSLV	DLSV1、DSLV2、DSLV3 的最大值	否	公斤 公顷$^{-1}$天
达维	死亡率因衰老而离开。	否	公斤 公顷$^{-1}$天
DRLV	死亡率作为 DSLV 和 DALV 的组合离开	否	公斤 公顷$^{-1}$天
SLAT	当前时间步长的特定叶面积，根据源/汇限制的叶扩展率进行调整。	否	公顷千克$^{-1}$
飞世奇	生理叶龄增加	否	•
格莱克斯	库限叶展开率（指数曲线）	否	哈哈$^{-1}$天$^{-1}$
格拉索尔	源限叶片展开率（生物量增加）	否	哈哈$^{-1}$天$^{-1}$

外部依赖：

姓名	描述	由…提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•
佛罗里达州	叶生物量分数	DVS_分区	•
FR	部分生物量到根	DVS_分区	•
SAI	茎面积指数	WOFOST_Stem_Dynamics	•
PAI	吊舱面积指数	WOFOST_Storage_Organ_Dynamics	•
运输署	蒸腾速率	蒸散量	厘米天$^{-1}$
电车	最大蒸腾速率	蒸散量	厘米天$^{-1}$
管理员权限	地上干物质增加	作物模拟	公斤 公顷$^{-1}$天$^{-1}$
RF_霜	减少因子霜杀	弗罗斯托尔	•

根动力学

班级 **pcse.crop.root_dynamics.WOFOST_Root_Dynamics**（天，信息亭，***args***，****kwargs***） [来源]

根生物量动态和生根深度。

WOFOST 中的根系生长和根系生物量动态是独立的过程，唯一的例外是当不再有生物量发送到根系时根系生长停止。

根系生物量的增加是由分配到根系的同化物引起的。根死亡定义为当前根生物量乘以相对死亡率（*RDRRTB*）。后者作为开发阶段（*DVS*）的函数。

根深的增加是随时间的简单线性扩展，直到达到最大生根深度（*RDM*）。

仿真参数

姓名	描述	类型	单元
RDI	初始生根深度	氯化铬	厘米
再生资源研究所	每天增加生根深度	氯化铬	厘米
RDMCR	Maximum rooting depth of the crop	SCR	cm
RDMSOL	Maximum rooting depth of the soil	SSo	cm
TDWI	Initial total crop dry weight	SCr	kg ha
IAIRDU	Presence of air ducts in the root (1) or not (0)	SCr	•
RDRRTB	Relative death rate of roots as a function of development stage	TCr	•

State variables

Name	Description
RD	Current rooting depth
RDM	Maximum attainable rooting depth at the minimum of the soil and crop maximum rooting depth
WRT	Weight of living roots
DWRT	Weight of dead roots
TWRT	Total weight of roots

Rate variables

Name	Description	Pbl	Unit

RR	Growth rate root depth	N	cm
GRRT	Growth rate root biomass	N	kg $ha^{-1}day^{-1}$
DRRT	Death rate root biomass	N	kg $ha^{-1}day^{-1}$
GWRT	Net change in root biomass	N	kg $ha^{-1}day^{-1}$

Signals send or handled

None

External dependencies:

Name	Description	Provided by	Unit
DVS	Crop development stage	DVS_Phenology	•
DMI	Total dry matter increase	CropSimulation	kg $ha^{-1}day^{-1}$
FR	Fraction biomass to roots	DVS_Partitioning	•

Stem dynamics

class `pcse.crop.stem_dynamics.WOFOST_Stem_Dynamics`(*day*, *kiosk*, **args*, ***kwargs*) [source]

Implementation of stem biomass dynamics.

Stem biomass increase results from the assimilates partitioned to the stem system. Stem death is defined as the current stem biomass multiplied by a relative death rate (*RDRSTB*). The latter as a function of the development stage (*DVS*).

茎是植物冠层的绿色元素，因此可以贡献总光合活性面积。这表示为茎面积指数，该指数是通过将茎生物量乘以比茎面积 (SSATB) 而获得的，它是 DVS 的函数。

仿真参数：

姓名	描述	类型	单元
TDWI	初始作物总干重	氯化铬	公斤公顷$^{-1}$
RDRSTB	茎的相对死亡率随发育阶段的变化	总铬	•
科学技术委员会	特定茎面积作为发育阶段的函数	总铬	公顷千克$^{-1}$

状态变量

姓名	描述	铅	单元

SAI	茎面积指数	是	•
WST	活茎重量	是	公斤公顷$^{-1}$
DWST	死茎重量	否	公斤公顷$^{-1}$
TWST	茎杆总重量	是	公斤公顷$^{-1}$

速率变量

姓名	描述	铅	单元
GRST	生长速率茎生物量	否	公斤 公顷$^{-1}$天$^{-1}$
DRST	死亡率茎生物量	否	公斤 公顷$^{-1}$天$^{-1}$
全球气象局	茎生物量的净变化	否	公斤 公顷$^{-1}$天$^{-1}$

发送或处理的信号

没有任何

外部依赖：

姓名	描述	由…提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•
管理员权限	地上干物质增加	作物模拟	公斤 公顷$^{-1}$天$^{-1}$
FR	部分生物量到根	DVS_分区	•
FS	部分生物量到茎	DVS_分区	•

贮藏器官动力学

班级 **pcse.crop.storage_organ_dynamics.WOFOST_Storage_Organ_Dynamics**（天，信息亭，***args***，****kwargs***） [来源]

实施储存器官动力学。

储存器官是 WOFOST 中植物最简单的组成部分，由静态的生物质池组成。贮藏器官的生长是同化分裂的结果。存储器官死亡未实现，相应的速率变量（DRSO）始终设置为零。

豆荚是植物冠层的绿色元素，因此可以贡献总光合活性面积。这表示为 Pod 面积指数，它是通过将 Pod 生物量与固定的特定 Pod 面积 (SPA) 相乘获得的。

仿真参数

姓名	描述	类型	单元
TDWI	初始作物总干重	氯化铬	公斤公顷$^{-1}$
温泉	特定吊舱面积	氯化铬	公顷千克$^{-1}$

状态变量

姓名	描述	铅	单元
PAI	豆荚面积指数	是	•
世界安全组织	活体贮藏器官重量	是	公斤公顷$^{-1}$
DWSO	死贮藏器官重量	否	公斤公顷$^{-1}$
TWSO	贮藏机构总重量	是	公斤公顷$^{-1}$

速率变量

姓名	描述	铅	单元
GRSO	生长速度储存器官	否	公斤 公顷$^{-1}$天$^{-1}$
DRSO	死亡率储存器官	否	公斤 公顷$^{-1}$天$^{-1}$
全球气象局	储存器官生物量的净变化	否	公斤 公顷$^{-1}$天$^{-1}$

发送或处理的信号

没有任何

外部依赖

姓名	描述	由...提供	单元
管理员权限	地上干物质增加	作物模拟	公斤 公顷$^{-1}$天$^{-1}$
FO	部分生物量到储存器官	DVS_分区	•
FR	部分生物量到根	DVS_分区	•

N/P/K 动力学

班级 **pcse.crop.npk_dynamics.NPK_Crop_Dynamics**（天，信息亭，***args***，****kwargs***） [来源]

实施整体 NPK 作物动态。

NPK_Crop_Dynamics 实现作物内 N/P/K 簿记的整体逻辑。

仿真参数

姓名	描述	单元
NMAXLV_TB	叶子中的最大 N 浓度作为 dvs 的函数	kg N kg-1 干生物量
PMAXLV_TB	至于普	kg P kg-1 干生物量
KMAXLV_TB	至于克	kg K kg-1 干生物量
NMAXRT_FR	根中的最大 N 浓度占叶子中最大 N 浓度的分数	•
PMAXRT_FR	至于普	•
KMAXRT_FR	至于克	•
NMAXST_FR	茎中的最大 N 浓度占叶子中最大 N 浓度的分数	•
KMAXST_FR	至于克	•
PMAXST_FR	至于普	•
NRESIDLV	叶子中的残留 N 分数	kg N kg-1 干生物量
主席团	叶子中的残留磷分数	kg P kg-1 干生物量
克雷西德夫	叶子中的残留钾分数	kg K kg-1 干生物量
NRESIDRT	根中的残留氮分数	kg N kg-1 干生物量
主席团	根中的残留磷分数	kg P kg-1 干生物量
克雷西德特	根部残留钾分数	kg K kg-1 干生物量
NRESIDST	茎中的残留氮分数	kg N kg-1 干生物量
总裁	茎中的残留磷分数	kg P kg-1 干生物量
克雷斯迪斯特	茎中的残留钾分数	kg K kg-1 干生物量

状态变量

姓名	描述	单元
名门LV	活叶中的实际 N 量	公斤 N 公顷$^{-1}$
派蒙LV	活叶中的实际磷量	公斤 P 公顷$^{-1}$

卡蒙LV	活叶中的实际钾含量	公斤 K 公顷$^{-1}$
南山ST	活茎中的实际氮量	公斤 N 公顷$^{-1}$
派蒙ST	活茎中的实际磷量	公斤 P 公顷$^{-1}$
卡蒙ST	活茎中的实际钾含量	公斤 K 公顷$^{-1}$
NamountSO	活体储存器官中的实际氮量	公斤 N 公顷$^{-1}$
派蒙SO	活储存器官中的实际磷量	公斤 P 公顷$^{-1}$
卡蒙SO	活体储存器官中的实际钾量	公斤 K 公顷$^{-1}$
NamountRT	活根中的实际氮量	公斤 N 公顷$^{-1}$
派蒙RT	活根中的实际磷量	公斤 P 公顷$^{-1}$
卡蒙RT	活根中的实际钾量	公斤 K 公顷$^{-1}$
Nuptake_T	总吸收氮量	公斤 N 公顷$^{-1}$
Puptake_T	总磷吸收量	公斤 P 公顷$^{-1}$
库普塔克_T	总吸收钾量	公斤 K 公顷$^{-1}$
Nfix_T	生物固氮总量	公斤 N 公顷$^{-1}$

速率变量

姓名	描述	单元
RNamountLV	叶子的重量增加 (N)	公斤 N 公顷$^{-1}$天$^{-1}$
RPamountLV	叶子的重量增加 (P)	公斤 P 公顷$^{-1}$ d $^{-1}$
RKamountLV	叶子的重量增加 (K)	公斤 K 公顷$^{-1}$天$^{-1}$
RNamountST	茎的重量增加 (N)	公斤 N 公顷$^{-1}$天$^{-1}$
RPamountST	茎的重量增加 (P)	公斤 P 公顷$^{-1}$ d $^{-1}$
RKamountST	茎的重量增加 (K)	公斤 K 公顷$^{-1}$天$^{-1}$
RNamountRT	根的重量增加 (N)	公斤 N 公顷$^{-1}$天$^{-1}$
RPamountRT	根的重量增加 (P)	公斤 P 公顷$^{-1}$ d $^{-1}$
RKamountRT	根的重量增加 (K)	公斤 K 公顷$^{-1}$天$^{-1}$
RNa数量SO	贮藏器官重量增加（N）	公斤 N 公顷$^{-1}$天$^{-1}$
RPamountSO	储存器官的重量增加（P）	公斤 P 公顷$^{-1}$ d $^{-1}$

RKamountSO	储存器官的重量增加（K）	公斤 K 公顷$^{-1}$天$^{-1}$
RN死亡LV	叶片氮素损失率	公斤 N 公顷$^{-1}$天$^{-1}$
RP死亡LV	至于P	公斤 P 公顷$^{-1}$ d $^{-1}$
RK死亡LV	至于克	公斤 K 公顷$^{-1}$天$^{-1}$
RN死亡ST	根部氮流失率	公斤 N 公顷$^{-1}$天$^{-1}$
RP死亡ST	至于P	公斤 P 公顷$^{-1}$ d $^{-1}$
RK死亡ST	至于克	公斤 K 公顷$^{-1}$天$^{-1}$
RN死亡RT	茎中氮损失率	公斤 N 公顷$^{-1}$天$^{-1}$
RP死亡RT	至于P	公斤 P 公顷$^{-1}$ d $^{-1}$
RK死亡RT	至于克	公斤 K 公顷$^{-1}$天$^{-1}$
RNloss	衰老导致的 N 损失	公斤 N 公顷$^{-1}$天$^{-1}$
R损失	衰老导致的 P 损失	公斤 P 公顷$^{-1}$ d $^{-1}$
RKloss	衰老导致的 K 损失	公斤 K 公顷$^{-1}$天$^{-1}$

发送或处理的信号

没有任何

外部依赖

姓名	描述	由…提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•
WLV	活叶干重	WOFOST_Leaf_Dynamics	公斤公顷$^{-1}$
WRT	活根干重	WOFOST_Root_Dynamics	公斤公顷$^{-1}$
WST	活茎干重	WOFOST_Stem_Dynamics	公斤公顷$^{-1}$
DRLV	叶片死亡率	WOFOST_Leaf_Dynamics	公斤 公顷$^{-1}$天$^{-1}$
DRRT	根的死亡率	WOFOST_Root_Dynamics	公斤 公顷$^{-1}$天$^{-1}$
DRST	茎的死亡率	WOFOST_Stem_Dynamics	公斤 公顷$^{-1}$天$^{-1}$

班级 **pcse.crop.nutrients.NPK_Demand_Uptake**（天，信息亭，***args***，****kwargs***） [来源]

计算作物 N/P/K 需求及其从土壤中的吸收。

作物 N/P/K 需求计算为植物营养器官（叶、茎和根）中的实际 N/P/K 浓度（kg N/P/K/kg 生物量）与最大 N/P 之间的差值/K 每个器官的浓度。然后将 N/P/K 吸收估计为土壤供应和作物需求的最小值。

固氮（豆科植物）的计算方法是假设每日 N 需求的固定部分由固氮提供。其余部分必须由土壤提供。

储存器官的 N/P/K 需求量的计算方式有些不同，因为假设储存器官的需求量是通过从叶、茎和根转移 N/P/K 来满足的。因此，储存器官的吸收量计算为可转移的 N/P/K（供应）和储存器官的需求中的最小值。此外，还有易位时间系数，考虑到可易位 N/P/K 的可用性存在延迟

仿真参数

姓名	描述	单元
NMAXLV_TB	叶片中的最大 N 浓度作为 DVS 的函数	kg N kg-1 干生
PMAXLV_TB	至于普	kg P kg-1 干生
KMAXLV_TB	至于克	kg K kg-1 干生
NMAXRT_FR	根中的最大 N 浓度占叶子中最大 N 浓度的分数	•
PMAXRT_FR	至于普	•
KMAXRT_FR	至于克	•
NMAXST_FR	茎中的最大 N 浓度占叶子中最大 N 浓度的分数	•
PMAXST_FR	至于普	•
KMAXST_FR	至于克	•
氮氧化物	储存器官中的最大氮浓度	kg N kg-1 干生
PMAXSO	至于普	kg P kg-1 干生
凯美索	至于克	kg K kg-1 干生
NCRIT_FR	临界 N 浓度占整个植物营养器官（叶 + 茎）最大 N 浓度的分数	•
PCRIT_FR	至于普	•
KCRIT_FR	至于克	•
碳纳米管	N 转化为储存器官的时间系数	天
TCPT	至于普	天
TCKT	至于克	天
NFIX_FR	作物通过生物固定吸收氮的比例	kg N kg-1 干生

RNUPTAKEMAX	最大吸氮率	公斤 N 公顷$^{-1}$天
RPUPTAKEMAX	最大吸磷率	公斤 N 公顷$^{-1}$天
RKUPTAKEMAX	钾吸收的最大速率	公斤 N 公顷$^{-1}$天

状态变量

速率变量

姓名	描述	铅	单元
RN摄取LV	叶片吸氮率	是	公斤 N 公顷$^{-1}$天$^{-1}$
RN摄取ST	茎吸氮率	是	公斤 N 公顷$^{-1}$天$^{-1}$
RN摄取RT	根系吸氮率	是	公斤 N 公顷$^{-1}$天$^{-1}$
RN摄取SO	贮藏器官吸氮率	是	公斤 N 公顷$^{-1}$天$^{-1}$
RPuptakeLV	叶片吸磷率	是	公斤 P 公顷$^{-1}$ d $^{-1}$
R摄取ST	茎吸磷率	是	公斤 P 公顷$^{-1}$ d $^{-1}$
RPuptakeRT	根系吸磷率	是	公斤 P 公顷$^{-1}$ d $^{-1}$
R摄取SO	贮藏器官吸磷率	是	公斤 P 公顷$^{-1}$ d $^{-1}$
RKuptakeLV	叶片吸钾率	是	公斤 K 公顷$^{-1}$天$^{-1}$
RKuptakeST	茎中钾的吸收率	是	公斤 K 公顷$^{-1}$天$^{-1}$
RKuptakeRT	根部吸钾率	是	公斤 K 公顷$^{-1}$天$^{-1}$
RKuptakeSO	贮藏器官吸收钾的速率	是	公斤 K 公顷$^{-1}$天$^{-1}$
R摄取	总氮吸收率	是	公斤 N 公顷$^{-1}$天$^{-1}$
R摄取	总吸磷率	是	公斤 P 公顷$^{-1}$ d $^{-1}$
吸收	总钾摄取率	是	公斤 K 公顷$^{-1}$天$^{-1}$
RN固定	固氮率	是	公斤 N 公顷$^{-1}$天$^{-1}$
NdemandLV	N 对活叶的需求	否	公斤 N 公顷$^{-1}$
NdemandST	N 对活茎的需求	否	公斤 N 公顷$^{-1}$
NdemandRT	N 对活根的需求	否	公斤 N 公顷$^{-1}$
NdemandSO	N 对存储机构的需求	否	公斤 N 公顷$^{-1}$
请求LV	P 活叶需求	否	公斤 P 公顷$^{-1}$

PdemandST	P 生活需求	否	公斤 P 公顷$^{-1}$
PdemandRT	P 对活根的需求	否	公斤 P 公顷$^{-1}$
PdemandSO	P 储存器官的需求	否	公斤 P 公顷$^{-1}$
KdemandLV	K 活叶需求	否	公斤 K 公顷$^{-1}$
KdemandST	K 生活需求	否	公斤 K 公顷$^{-1}$
KdemandRT	K 对活根的需求	否	公斤 K 公顷$^{-1}$
KdemandSO	K 储存器官的需求	否	公斤 K 公顷$^{-1}$
点播	作物总氮需求	否	公斤 N 公顷$^{-1}$天$^{-1}$
需求	总作物磷需求	否	公斤 P 公顷$^{-1}$ d $^{-1}$
需求	作物总钾需求	否	公斤 K 公顷$^{-1}$天$^{-1}$

发送或处理的信号

没有任何

外部依赖

姓名	描述	由…提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•
运输署	作物蒸腾	蒸散量	|cm d-1|
电车	潜在的作物蒸腾	蒸散量	|cm d-1|
海军	土壤总有效氮	NPK_土壤_动力学	公斤公顷$^{-1}$
帕维尔	土壤总有效磷	NPK_土壤_动力学	公斤公顷$^{-1}$
可乐	土壤总有效钾	NPK_土壤_动力学	公斤公顷$^{-1}$
N易位	从茎、叶和根转移的氮量	NPK_易位	公斤公顷$^{-1}$
P易位	至于普	NPK_易位	公斤公顷$^{-1}$
K易位	至于克	NPK_易位	公斤公顷$^{-1}$

班级 **pcse.crop.nutrients.NPK_Stress**（天，信息亭，***args***，****kwargs***） [来源]

通过[NPK]营养指数实现NPK压力计算。

胁迫因子是根据植物叶和茎生物量中 N/P/K 的质量浓度计算的。对于每个养分池，根据叶和茎的生物量计算四种浓度： - 基于实际养分量的实际浓度

除以实际的叶和茎生物量。

- 最大浓度，是植物可以吸收到其叶子和茎中的最大值。
- 临界浓度，即维持不受 N/P/K 限制的生长速率所需的浓度。对于 P 和 K，临界浓度通常等于最大浓度。对于 N，临界浓度可能低于最大浓度。这种浓度有时被称为“最佳浓度”。
- 残留浓度是锁定在植物结构生物量中且不能再被调动的量。

压力指数 (SI) 是根据以下公式确定为这些浓度之间的简单比率：

$$SI = (C_a - C_r)/(C_c - C_r)$$

下标*a*、*r*和*c*是营养物的实际浓度、残留浓度和临界浓度。该方程式应用于 N、P 和 K 的模拟，并得出氮营养指数 (NNI)、磷营养指数 (PNI) 和钾营养指数 (KNI)。接下来，NPK 指数 (NPKI) 计算为 NNI、PNI、KNI 中的最小值。最后，使用光利用效率的缩减因子 (NLUE_NPK) 计算同化缩减因子 (NPKREF)。

仿真参数

姓名	描述	单元
NMAXLV_TB	叶片中的最大 N 浓度作为 DVS 的函数	kg N kg-1 干生物量
PMAXLV_TB	至于普	kg P kg-1 干生物量
KMAXLV_TB	至于克	kg K kg-1 干生物量
NMAXRT_FR	根中的最大 N 浓度占叶子中最大 N 浓度的分数	•
PMAXRT_FR	至于普	•
KMAXRT_FR	至于克	•
NMAXST_FR	茎中的最大 N 浓度占叶子中最大 N 浓度的分数	•
PMAXST_FR	至于普	•
KMAXST_FR	至于克	•
NCRIT_FR	临界 N 浓度占整个植物营养器官（叶 + 茎）最大 N 浓度的分数	•
PCRIT_FR	至于普	•
KCRIT_FR	至于克	•
NRESIDLV	叶子中的残留 N 分数	kg N kg-1 干生物量
主席团	叶子中的残留磷分数	kg P kg-1 干生物量
克雷西德夫	叶子中的残留钾分数	kg K kg-1 干生物量
NRESIDST	茎中的残留氮分数	kg N kg-1 干生物量

总裁	茎中的残留磷分数	kg P kg-1 干生物量
克雷斯迪斯特	茎中的残留钾分数	kg K kg-1 干生物量
NLUE_NPK	由于营养 (NPK) 胁迫导致 RUE 减少的系数	•

速率变量

这里的利率变量不是实际利率变量，因为它们是派生的状态变量，不代表利率。然而，由于它们直接用于利率变量计算，因此将它们放在这里是合乎逻辑的。

姓名	描述	铅	单元
神经网络接口	氮素营养指标	是	•
PNI	氮素营养指标	否	•
国民党	氮素营养指标	否	•
NPKI	最低 NNI、PNI、KNI	是	•
RFNPK	基于NPKI和参数NLUE_NPK的CO 2同化还原因子	否	•

外部依赖：

姓名	描述	由...提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•
WST	活茎干重	WOFOST_Stem_Dynamics	公斤公顷$^{-1}$
WLV	活叶干重	WOFOST_Leaf_Dynamics	公斤公顷$^{-1}$
名门LV	叶子中的 N 量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
南山ST	茎中的氮含量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
派蒙LV	叶子中的磷量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
派蒙ST	茎中的磷含量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
卡蒙LV	叶子中的钾含量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
卡蒙ST	茎中的钾含量	NPK_Crop_Dynamics	公斤公顷$^{-1}$

班级 **pcse.crop.nutrients.NPK_Translocation**（天，信息亭，***args***，****kwargs***） [来源]

记录 N/P/K 从根、叶和茎向作物储存器官的转移。

首先，例程计算 N/P/K 的可转移量的状态。该可转移量定义为高于剩余 N/P/K 量的 N/P/K 量，计算为剩余浓度乘以活生物量。剩余量被锁定在植物结构生物量中，不能再动员起来。计算茎、根和叶的易位量，并作为状态变量 Ntranslocatable、Ptranslocatable 和 Ktranslocatable 发布。

总易位率计算为存储机构的供应（可易位量）和需求的最小值，如 Demand_Uptake 组件中计算的那样。假设吸收率分布在根、茎和叶上，与每个器官的可转移量成比例，计算不同植物器官的实际 N/P/K 转移率。

仿真参数

姓名	描述	单元
NRESIDLV	叶子中的残留 N 分数	kg N
主席团	叶子中的残留磷分数	kg P
克雷西德夫	叶子中的残留钾分数	kg K
NRESIDST	茎中的残留氮分数	kg N
总裁	茎中的残留磷分数	kg P
克雷斯迪斯特	茎中的残留钾分数	kg K
NPK_TRANSLRT_FR	来自根的 NPK 易位作为 resp 的一小部分。从叶和茎转移的 NPK 总量	•

状态变量

姓名	描述	铅	单元
N易位LV	活叶中可转移的 N 量	否	公斤 N 公顷$^{-1}$
P易位LV	活叶中可转移的磷量	否	公斤 P 公顷$^{-1}$
K易位LV	活叶中可转运的钾量	否	公斤 K 公顷$^{-1}$
N易位ST	活茎中可转移的 N 量	否	公斤 N 公顷$^{-1}$
P易位ST	活茎中可转移的磷量	否	公斤 P 公顷$^{-1}$
K易位ST	活茎中可转运的钾量	否	公斤 K 公顷$^{-1}$
N易位RT	活根中可转移的 N 量	否	公斤 N 公顷$^{-1}$
P易位RT	活根中可转移的磷量	否	公斤 P 公顷$^{-1}$
K易位RT	活根中可转运的钾量	否	公斤 K 公顷$^{-1}$
N易位	可以转移到储存器官的总氮量	是	[kg N ha-1]

P易位	可以转移到储存器官的总磷量	是	[kg P ha-1]
K易位	可转运至储存器官的总钾量	是	[kg K ha-1]

速率变量

姓名	描述	铅	单元
RN易位LV	叶子的重量增加 (N)	是	公斤 公顷$^{-1}$天$^{-1}$
RP易位LV	叶子的重量增加 (P)	是	公斤 公顷$^{-1}$天$^{-1}$
RK易位LV	叶子的重量增加 (K)	是	公斤 公顷$^{-1}$天$^{-1}$
RN易位ST	茎的重量增加 (N)	是	公斤 公顷$^{-1}$天$^{-1}$
RP易位ST	茎的重量增加 (P)	是	公斤 公顷$^{-1}$天$^{-1}$
RK易位ST	茎的重量增加 (K)	是	公斤 公顷$^{-1}$天$^{-1}$
RN易位RT	根的重量增加 (N)	是	公斤 公顷$^{-1}$天$^{-1}$
RP易位RT	根的重量增加 (P)	是	公斤 公顷$^{-1}$天$^{-1}$
RK易位RT	根的重量增加 (K)	是	公斤 公顷$^{-1}$天$^{-1}$

发送或处理的信号

没有任何

外部依赖：

姓名	描述	由...提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•
WST	活茎干重	WOFOST_Stem_Dynamics	公斤公顷$^{-1}$
WLV	活叶干重	WOFOST_Leaf_Dynamics	公斤公顷$^{-1}$
WRT	活根干重	WOFOST_Root_Dynamics	公斤公顷$^{-1}$
名门LV	叶子中的 N 量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
南山ST	茎中的氮含量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
NamountRT	根中的 N 量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
派蒙LV	叶子中的磷量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
派蒙ST	茎中的磷含量	NPK_Crop_Dynamics	公斤公顷$^{-1}$

派蒙RT	根中的磷量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
卡蒙LV	叶子中的钾含量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
卡蒙ST	茎中的钾含量	NPK_Crop_Dynamics	公斤公顷$^{-1}$
卡蒙RT	根中的钾量	NPK_Crop_Dynamics	公斤公顷$^{-1}$

非生物损害

班级 **pcse.crop.abioticdamage.FROSTOL**（天，信息亭，***args**，****kwargs**） [来源]

冬小麦霜害 FROSTOL 模型的实施。

参数：
- 天——模拟的开始日期
- **kiosk** – 此 PCSE 实例的可变信息亭
- **parvalues** – *ParameterProvider*对象以键/值对形式提供参数

仿真参数

姓名	描述
国际用户线	仅物候发展选项温度 (IDSL=0) 的开关，包括日长 (IDSL=1) 和春化 (IDSL
LT50C	临界 LT50 定义为小麦品种可以获得的最低 LT50 值
冰霜_H	硬化系数
冰霜_D	脱硬系数
冰霜_S	低温应力系数
冰霜_R	呼吸应力系数
FROSTOL_SDBASE 数据库	呼吸压力下的最小积雪深度
FROSTOL_SDMAX	最大呼吸压力下的雪深。更大的雪深不再增加压力。
FROSTOL_KILLCF	logistic kill 函数的陡度系数。
ISNOWSRC	使用来自驱动变量 (0) 的规定雪深或通过售货亭模拟的雪深 (1)

状态变量

姓名	描述	铅	单元
LT50T	当前 LT50 值	否	°C
			°

LT50I	未硬化作物的初始 LT50 值	否	C
IDFST	霜冻胁迫总天数	否	•

速率变量

姓名	描述
相对湿度	硬化率
RDH_TEMP	温度脱硬率
RDH_RESP	呼吸应力脱硬率
RDH_TSTR	因温度应力引起的脱硬率
国际文件系统	霜冻胁迫，是 (1) 或否 (0)。霜冻应力定义为：RF_FROST > 0
RF_霜	叶生物量的减少因子作为最小值的函数。冠温和 LT50T：范围从 0（无损坏）到 1（完全
RF_FROST_T	整个生长季节的总霜冻死亡数计算为每日霜冻死亡事件的乘积，0 表示没有损坏，1 表示

外部依赖：

姓名	描述	由...提供
临时皇冠	通过调用 crown_temperature 模块得出的每日平均冠温。	皇冠温度
TMIN_CROWN	通过调用 crown_temperature 模块得出的每日最低冠温。	皇冠温度
国际化	反映作物春化状态的布尔值。	带 DVS_Phenology 模块

参考资料：**Anne Kari Bergjord**、**Helge Bonesmo**、**Arne Oddvar Skjelvag**，**2008**。

模拟冬小麦的抗冻过程：I. 模型开发，欧洲农学杂志，第 28 卷，第 3 期，2008 年 4 月，第 321-330 页。

http://dx.doi.org/10.1016/j.eja.2007.10.002

班级 **pcse.crop.abioticdamage.CrownTemperature**（天，信息亭，***args***，****kwargs***） [来源]

实施一个简单的算法来估算雪下的树冠温度（土壤表面下 2 厘米）。

Is 基于一个简单的经验方程，该方程估计每日最低、最高和平均树冠温度作为每日最低或最高温度和相对积雪深度 (RSD) 的函数：

$$RSD = \min(15, SD)/15$$

和

$$T_{min}^{crown} = T_{min} * (A + B(1 - RSD)^2)$$

和

$$T_{max}^{crown} = T_{max} * (A + B(1 - RSD)^2)$$

和

$$T_{avg}^{crown} = (T_{max}^{crown} + T_{min}^{crown})/2$$

在零积雪深度下，树冠温度估计接近空气温度。增加的积雪深度起到缓冲作用，抑制低气温对冠部温度的影响。雪深的最大值限制在 15cm。A 和 B 的典型值为 0.2 和 0.5

请注意，仅当 drv.TMIN<0 时才估计冠温，否则返回 TMIN、TMAX 和日平均温度 (TEMP)。

参数:
- **day** – 模型初始化的日期
- **kiosk** – 此实例的 VariableKiosk
- **parvalues** – *ParameterProvider*对象以键/值对形式提供参数

退货: 包含最低、最高和每日平均冠温的元组。

仿真参数

姓名	描述	类型	单元
ISNOWSRC	使用来自驱动变量 (0) 的规定雪深或通过售货亭模拟的雪深 (1)	不锈钢	•
皇冠	冠温方程中的一个参数	不锈钢	•
皇冠MPB	冠温方程中的 B 参数	不锈钢	•

速率变量

姓名	描述	铅	单元
临时皇冠	日平均树冠温度	否	°C
TMIN_CROWN	日最低冠温	否	°C
TMAX_皇冠	日最高冠温	否	°C

请注意，计算出的冠层温度不是实际速率变量，因为它们与变化率无关。事实上它们是一个派生的驱动变量。尽管如此，为了计算霜冻损坏，它们应该在速率计算步骤中变得可用，并将它们视为速率变量，它们可以通过***get_variable()***调用找到，因此在配置文件的 OUTPUT_VARS 列表中定义

外部依赖:

姓名	描述	由...提供	单元
雪深	积雪深度。	由驱动变量规定或由积雪模块模拟并从信息亭获取	cm

LINGRA 和 LINGRA-N 的作物模拟过程

LINGRA草地模拟模型的实现

该模块提供了 Schapendonk 等人描述的草地 LINGRA (LINtul GRAssland) 模拟模型的实现。1998 (https://doi.org/10.1016/S1161-0301(98)00027-6) 在 Python 作物模拟环境中使用。

整体草地模型

班级 **pcse.crop.lingra.LINGRA**（天，信息亭，***args***，****kwargs***） [来源]

LINGRA 的顶级实施，集成所有组件

此类集成了 LINGRA 模型的所有组件，包括与不同生物量池的权重、叶面积、分蘖数和叶长相关的主要状态变量。集成组件包括源/汇限制生长、土壤温度、蒸发蒸腾和根系动态的实施。为了避免代码重复，后两个取自WOFOST。

与 Schapendonk 等人的原始代码相比。(1998) 进行了多项改进：

- 代码的整体重组，删除不需要的变量并重命名剩余的变量以具有更易读的名称。
- 更清晰地实施汇/源有限增长，包括使用储备
- 由库限制生长模块计算的潜在叶片伸长率现在针对实际生长进行了校正。从而避免在水分胁迫条件下导致不切实际的结果的无限叶片生长。

仿真参数：

姓名	描述	单元
初始化	初始叶面积指数	•
TillerNumberinit	初始分蘖数	分蘖数/m2
权重重新初始化	储备初始权重	公斤/公顷
重量RTinit	根的初始权重	公斤/公顷
暴击	由于自遮蔽导致死亡的临界 LAI	•
RDR库	根的背景相对死亡率	d-1
RDR阴影	自遮荫叶片最大相对死亡率	d-1
RDR干旱	干旱胁迫下叶片的最大相对死亡率	d-1
服务水平协议	比叶面积	公顷/公斤
临时数据库	光合作用和发育的基准温度	C

分区根TB	根的分配分数作为蒸腾减少因子 (RFTRA) 的函数	-, -
TSUM最大值	最高发育阶段的温度总和	光盘

速率变量

姓名	描述	单元
dTSUM	开发温度总和的变化	C
dLAI	叶面积指数的净变化	d-1
d天后收获	收获后天数的变化	•
d切削数	插条数量的变化（收获）	•
d权重LV	叶重净变化	公斤/公顷/天
d权重RE	储备池净变动	公斤/公顷/天
d叶长法案	实际叶长的变化	厘米/天
LV死亡	叶片死亡率	公斤/公顷/天
增长	叶片生长速度	公斤/公顷/天
d重量HARV	收获干物质的变化	公斤/公顷/天
d权重RT	根重的净变化	公斤/公顷/天
LV分数	分配给叶子的分数	•
RT分数	分数划分为根	•

状态变量

姓名	描述	单元
TSUM	温度总和	Cd
赖	叶面积指数	•
收获后天数	收获后的天数	d
切削数	插条数量（收获）	•
分蘖数	分蘖数	分蘖数/m2
重量LV绿色	绿叶重量	公斤/公顷

体重LVdead	枯叶重量	公斤/公顷
重量HARV	收获的干物质重量	公斤/公顷
权重RE	储备权重	公斤/公顷
体重RT	根的重量	公斤/公顷
叶长	叶长	公斤/公顷
体重ABG	地上总重量（收获+当前）	公斤/公顷
斯莱特	赛季期间的综合 SLA	公顷/公斤
数字视频服务器	发展阶段	•

发送或处理的信号

广播*pcse.signals.mowing*事件时将进行割草。这将减少活叶重量，假设一定量的生物量将保留在田间（这是 MOWING 事件的一个参数）。

外部依赖：

姓名	描述	由...提供
RFTRA	蒸腾减少因子	pcse.crop.蒸发量
dLeafLength花盆	叶长的潜在增长	pcse.crop.lingra.SinkLimitedGrowth
dTillerNumber	舵柄数变化	pcse.crop.lingra.SinkLimitedGrowth

源/汇增长有限

班级 **pcse.crop.lingra.SourceLimitedGrowth**（天，信息亭，***args***，****kwargs***） [来源]

根据辐射和温度作为驱动变量并可能受土壤水分或叶片氮含量限制，计算草地的源限生长速率。后者基于当前和最大 N 浓度的静态值，主要用于连接 N 模块将来。

此例程使用光利用效率 (LUE) 方法，其中根据温度和辐射水平的影响调整 LUE。前者是必需的，因为光合作用具有明确的温度响应。后者是必需的，因为光合作用速率在较高辐射水平下趋于平缓，从而导致较低的“表观”光利用效率。参数*LUEreductionRadiationTB*是对此效应的粗略经验校正。

请注意，由于土壤水分导致的生长速度下降是通过蒸腾减少因子 (RFTRA) 获得的。

该模块不提供任何真实的速率变量，而是通过 __call__() 将计算出的增长率直接返回给调用例程。

仿真参数：

姓名	描述	单元

KDIFTB	作为 DVS 函数的漫反射可见光的消光系数。	•
二氧化碳	大气中的二氧化碳浓度	ppm
LUE还原土温TB	光利用效率的降低函数作为土壤温度的函数。	C， -
LUE减少辐射TB	光利用效率的降低函数作为辐射水平的函数。	兆焦耳, -
LUE最大值	最大的光利用效率。	

速率变量

姓名	描述	单元
RF_RadiationLevel	辐射水平引起的光利用效率降低系数	•
RF_RadiationLevel	辐射水平引起的光利用效率降低系数	•
LUEact	实际光利用效率	g /(MJ PAR)

发送或处理的信号

没有任何

外部依赖：

姓名	描述	由...提供
数字视频服务器	作物发育阶段	pylingra.LINGRA
温度土壤	土壤温度	pylingra.SoilTemperature
RFTRA	蒸腾减少因子	pcse.crop.蒸发量

班级 **pcse.crop.lingra.SinkLimitedGrowth**（天，信息亭，***args***，****kwargs***） [来源]

假设温度驱动的最大叶片伸长率乘以分蘖数，计算草地的汇限制生长率。通过除以比叶面积 (SLA) 来转换为以 kg/ha 干物质表示的生长。

除了库限制生长率外，该类还计算了分蘖数的变化，同时考虑了生长率、死亡率和收获落叶后的天数。

仿真参数：

姓名	描述	单元
临时数据库	叶片发育和草物候的基础温度	C
		•

LAIC暴击	临界叶面积，超过该面积会因自遮而导致叶片死亡	
站点填充最大值	新芽的最大位置填充	分蘖/叶-1
服务水平协议	比叶面积	公顷/公斤
TSUM最大值	最高发育阶段的温度总和	光盘
TillerFormRateA0	直到收获后 7 天有效的分蘖形成率方程中的一个参数	
TillerFormRateB0	分蘖形成率公式中的 B 参数在收获后 7 天内有效	
TillerFormRateA8	从收获后 8 天开始的分蘖形成率方程中的一个参数	
TillerFormRateB8	从收获后 8 天开始的分蘖形成率方程中的 B 参数	

速率变量

姓名	描述	单元
dTillerNumber	辐射水平导致的分蘖数变化	分蘖数
dLeafLength花盆	叶长的潜在变化。稍后将在考虑源限制的情况下计算叶长度的实际变化。	厘米/天
LAIGrowthSink	基于汇限制增长率的 LAI 增长。	d-1

发送或处理的信号

没有任何

外部依赖：

姓名	描述	由...提供
数字视频服务器	作物发育阶段	pylingra.LINGRA
赖	叶面积指数	pylingra.LINGRA
温度土壤	土壤温度	pylingra.SoilTemperature
RF_温度	基于温度的 LUE 降低系数	pylingra.SourceLimitedGrowth
分蘖数	实际分蘖数	pylingra.LINGRA
LV分数	进入叶子的同化物的分数	pylingra.LINGRA
d重量HARV	收获重量的变化（表示今天收获）	pylingra.LINGRA

氮动力学

班级 **pcse.crop.lingra_ndynamics.N_Demand_Uptake**（天，信息亭，***args***，****kwargs***） [来源]

计算作物对氮的需求及其从土壤中的吸收。

作物对氮的需求计算为植物营养器官（叶、茎和根）中的实际氮浓度（千克氮/千克生物量）与每个器官的最大氮浓度之间的差值。然后将 N 吸收量估计为土壤供应量和作物需求量的最小值。

仿真参数

速率变量

姓名	描述	铅	单元
RN摄取LV	叶片吸氮率	是	公斤 N 公顷$^{-1}$天
RN摄取RT	根系吸氮率	是	公斤 N 公顷$^{-1}$天
R摄取	总氮吸收率	是	公斤 N 公顷$^{-1}$天
NdemandLV	Ndemand of leaves 基于当前的增长率和之前时间步长的不足	否	公斤 N 公顷$^{-1}$
NdemandRT	N 求根，求叶	否	公斤 N 公顷$^{-1}$
点播	总 N 需求量（叶 + 根）	否	公斤 N 公顷$^{-1}$

发送或处理的信号

没有任何

外部依赖

姓名	描述	由...提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•
海军	土壤总有效氮	NPK_土壤_动力学	公斤公顷$^{-1}$

班级 **pcse.crop.lingra_ndynamics.N_Stress**（天，信息亭，***args***，****kwargs***） [来源]

通过氮营养指数实现氮胁迫计算。

胁迫因子是根据植物营养生物量中 N 的质量浓度计算的。对于每个养分池，根据叶和茎的生物量计算四种浓度： - 基于实际养分量的实际浓度

除以植物生物量。

- 最大浓度，是植物可以吸收到其叶子和茎中的最大值。
- 临界浓度，即维持不受 N 限制的增长率所需的浓度（由 NCRIT_FR 调节）。对于 N，临界浓度可能低于最大浓度。这种浓度有时被称为“最佳浓度”。
- 残留浓度是锁定在植物结构生物量中且不能再被调动的量。

压力指数 (SI) 是根据以下公式确定为这些浓度之间的简单比率：

\(SI = (C_{a) - C_{r})/(C_{c} - C_{r})\)

下标*a*、*r*和*c*是营养物的实际浓度、残留浓度和临界浓度。这导致了氮营养指数 (NNI)。最后，使用光利用效率降低因子 (NLUE) 计算同化降低因子 (RFNUTR)。

仿真参数

速率变量

这里的利率变量不是实际利率变量，因为它们是派生的状态变量，不代表利率。然而，由于它们直接用于利率变量计算，因此将它们放在这里是合乎逻辑的。

姓名	描述	铅	单元
神经网络接口	氮素营养指标	是	•
RFNUTR	光利用效率折减系数	是	•

外部依赖：

姓名	描述	由...提供	单元
数字视频服务器	作物发育阶段	DVS_物候学	•
WST	活茎干重	WOFOST_Stem_Dynamics	公斤公顷$^{-1}$
重量LV绿色	活叶干重	WOFOST_Leaf_Dynamics	公斤公顷$^{-1}$
名门LV	叶子中的 N 量	N_Crop_Dynamics	公斤公顷$^{-1}$

班级 **pcse.crop.lingra_ndynamics.N_Crop_Dynamics**（天，信息亭，***args***，****kwargs***） [来源]

实施整体 N 作物动态。

NPK_Crop_Dynamics 实现了作物内 N 簿记的整体逻辑。

仿真参数

状态变量

姓名	描述	铅	单元
名门LV	活叶中的实际 N 量	是	公斤 N 公顷$^{-1}$
NamountRT	活根中的实际氮量	是	公斤 N 公顷$^{-1}$
Nuptake_T	总吸收氮量	否	公斤 N 公顷$^{-1}$
Nlosses_T	由于衰老损失的总氮量	否	公斤 N 公顷$^{-1}$

速率变量

姓名	描述	铅	单元
RNamountLV	叶子的重量增加 (N)	否	公斤 公顷$^{-1}$天$^{-1}$
RNamountRT	根的重量增加 (N)	否	公斤 公顷$^{-1}$天$^{-1}$
RN死亡LV	叶片氮素损失率	否	公斤 公顷$^{-1}$天$^{-1}$
RN死亡RT	根部氮流失率	否	公斤 公顷$^{-1}$天$^{-1}$
RNloss	衰老导致的 N 损失	否	公斤 公顷$^{-1}$天$^{-1}$

发送或处理的信号

没有任何

外部依赖

基类

基类定义了 PCSE 中“幕后”使用的大部分功能。除了*VariableKiosk*和*WeatherDataContainer*之外，所有类都不能直接调用，而应该子类化。

可变亭

班级 **pcse.base.VariableKiosk** [来源]

VariableKiosk 用于在 PCSE 中注册和发布状态变量。

实例化 VariableKiosk 不需要任何参数。在 PCSE 中定义的所有变量都将在 VariableKiosk 中注册，而通常只有一小部分变量会通过 kiosk 发布。已发布变量的值可以使用括号表示法检索，因为变量 Kiosk 本质上是一个（有点花哨的）字典。

注册/注销速率和状态变量通过 *self.register_variable()*和*self.deregister_variable()*方法进行，而 *set_variable()*方法用于更新已发布变量的值。一般来说，这些方法都不需要由用户直接调用，因为*StatesTemplate*和*RatesTemplate*中的逻辑会处理这个问题。

最后，*variable_exists()*可用于检查变量是否已注册，而*flush_states()*和*flush_rates()*用于删除（刷新）任何已发布状态和速率变量的值。

例子：

```
>>> import pcse
>>> from pcse.base import VariableKiosk
>>>
>>> v = VariableKiosk()
>>> id0 = 0
>>> v.register_variable(id0, "VAR1", type="S", publish=True)
>>> v.register_variable(id0, "VAR2", type="S", publish=False)
>>>
>>> id1 = 1
>>> v.register_variable(id1, "VAR3", type="R", publish=True)
>>> v.register_variable(id1, "VAR4", type="R", publish=False)
>>>
```

```
>>> v.set_variable(id0, "VAR1", 1.35)
>>> v.set_variable(id1, "VAR3", 310.56)
>>>
>>> print v
Contents of VariableKiosk:
 * Registered state variables: 2
 * Published state variables: 1 with values:
  - variable VAR1, value: 1.35
 * Registered rate variables: 2
 * Published rate variables: 1 with values:
  - variable VAR3, value: 310.56

>>> print v["VAR3"]
310.56
>>> v.set_variable(id0, "VAR3", 750.12)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "pcse/base.py", line 148, in set_variable
    raise exc.VariableKioskError(msg % varname)
pcse.exceptions.VariableKioskError: Unregistered object tried to set the value of variable 'VAR3':
access denied.
>>>
>>> v.flush_rates()
>>> print v
Contents of VariableKiosk:
 * Registered state variables: 2
 * Published state variables: 1 with values:
  - variable VAR1, value: 1.35
 * Registered rate variables: 2
 * Published rate variables: 1 with values:
  - variable VAR3, value: undefined

>>> v.flush_states()
>>> print v
Contents of VariableKiosk:
 * Registered state variables: 2
 * Published state variables: 1 with values:
  - variable VAR1, value: undefined
 * Registered rate variables: 2
 * Published rate variables: 1 with values:
  - variable VAR3, value: undefined
```

deregister_variable（*oid*，变量名） [来源]

具有 id(object) 的对象要求从 kiosk 中注销 varname

参数：
- **oid** – 注册此变量的状态/速率对象的对象 ID（来自 python 内置 id() 函数）。
- **varname** – 要注册的变量的名称，例如“DVS”

flush_rates( ) [来源]

从 kiosk 刷新所有已发布的利率变量的值。

flush_states( ) [来源]

从信息亭刷新所有状态变量的值。

register_variable(*oid* , *varname* , *type* , *publish=False*) [来源]

从具有 id 和给定类型的对象注册一个 varname

参数：
- **oid** – 注册此变量的状态/速率对象的对象 ID（来自 python 内置 id() 函数）。
- **varname** – 要注册的变量的名称，例如“DVS”
- 类型——“R”（利率）或“S”（状态）变量，由状态/利率模板类自动处理。
- **publish** – 如果变量应该在信息亭发布，则为 True，默认为 False

set_variable(*id* ,变量名,值) [来源]

让带有id的对象，设置变量varname的值

参数:
- **id** – 注册此变量的状态/速率对象的对象 ID（来自 python 内置 id() 函数）。
- **varname** – 要更新的变量的名称
- **value** – 要分配给变量的值。

variable_exists（变量名） [来源]

如果状态/速率变量已在信息亭中注册，则返回 True。

参数: 变量名– 要检查注册的变量的名称。

参数、速率和状态的基类

班级 **pcse.base.StatesTemplate**(*kiosk=None* , *publish=None* , ***kwargs*) [来源]

负责为状态变量分配初始值、在信息亭中注册变量以及监控对已发布变量的分配。

参数:
- **kiosk** – VariableKiosk 类的实例。所有状态变量都将在信息亭中注册，以确保变量名称在整个模型中是唯一的。此外，发布的变量值将通过 VariableKiosk 提供。
- **publish** – 列出其值需要在 VariableKiosk 中发布的变量。如果不需要发布变量，则可以省略。

在实例化 States 类时，可以将状态变量的初始值指定为关键字。

例子:

```
>>> import pcse
>>> from pcse.base import VariableKiosk, StatesTemplate
>>> from pcse.traitlets import Float, Integer, Instance
>>> from datetime import date
>>>
>>> k = VariableKiosk()
>>> class StateVariables(StatesTemplate):
...     StateA = Float()
...     StateB = Integer()
...     StateC = Instance(date)
...
>>> s1 = StateVariables(k, StateA=0., StateB=78, StateC=date(2003,7,3),
...                     publish="StateC")
>>> print s1.StateA, s1.StateB, s1.StateC
0.0 78 2003-07-03
>>> print k
Contents of VariableKiosk:
 * Registered state variables: 3
 * Published state variables: 1 with values:
  - variable StateC, value: 2003-07-03
 * Registered rate variables: 0
 * Published rate variables: 0 with values:

>>>
>>> s2 = StateVariables(k, StateA=200., StateB=1240)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "pcse/base.py", line 396, in __init__
    raise exc.PCSEError(msg)
pcse.exceptions.PCSEError: Initial value for state StateC missing.
```

touch( ) [来源]

重新分配每个状态变量的值，从而在变量发布时更新其在变量信息亭中的值。

班级 **pcse.base.RatesTemplate**(*kiosk=None , publish=None*) [来源]

负责在信息亭中注册变量并监视对已发布变量的分配。

参数：
- **kiosk** – VariableKiosk 类的实例。所有速率变量都将在信息亭中注册，以确保变量名称在整个模型中是唯一的。此外，发布的变量值将通过 VariableKiosk 提供。
- **publish** – 列出其值需要在 VariableKiosk 中发布的变量。如果不需要发布变量，则可以省略。

有关示例，请参阅*StatesTemplate*。唯一的区别是不需要指定速率变量的初始值，因为该值将被设置为零（Int、Float 变量）或 False（布尔变量）。

zerofy( ) [来源]

将所有速率值的值设置为零（Int、Float）或 False（布尔）。

班级 **pcse.base.ParamTemplate**（面值） [来源]

用于存储参数值的模板。

这意味着由定义参数的实际类进行子类化。

例子：

```
>>> import pcse
>>> from pcse.base import ParamTemplate
>>> from pcse.traitlets import Float
>>>
>>>
>>> class Parameters(ParamTemplate):
...     A = Float()
...     B = Float()
...     C = Float()
...
>>> parvalues = {"A" :1., "B" :-99, "C":2.45}
>>> params = Parameters(parvalues)
>>> params.A
1.0
>>> params.A; params.B; params.C
1.0
-99.0
2.4500000000000002
>>> parvalues = {"A" :1., "B" :-99}
>>> params = Parameters(parvalues)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "pcse/base.py", line 205, in __init__
    raise exc.ParameterError(msg)
pcse.exceptions.ParameterError: Value for parameter C missing.
```

天气数据的基类和实用类

班级 **pcse.base.WeatherDataProvider** [来源]

所有天气数据提供者的基类。

必须通过设置类变量*supports_ensembles = True*来指示 WeatherDataProvider 中对天气集合的支持

例子：

```
class MyWeatherDataProviderWithEnsembles(WeatherDataProvider):
    supports_ensembles = True

    def __init__(self):
        WeatherDataProvider.__init__(self)

        # remaining initialization stuff goes here.
```

check_keydate（关键） [来源]

检查存储/检索天气数据的日期表示。

支持以下格式：

1. 日期对象
2. 日期时间对象
3. YYYYMMDD 格式的字符串
4. YYYYDDD 格式的字符串

Formats 2-4 都在内部转换为日期对象。

export() [来源]

将 WeatherDataProvider 的内容导出为字典列表。

导出的结果可以直接转换成pandas dataframe，方便绘图或分析。

班级 **pcse.base.WeatherDataContainer**(***args*** , ****kwargs***) [来源]

用于存储天气数据元素的类。

天气数据元素是通过关键字提供的，这些关键字也是可以在 WeatherDataContainer 中访问变量的属性名称。因此关键字 TMAX=15 将属性 TMAX 设置为 15。

以下关键字是必填项：

参数：
- **LAT** – 位置纬度（十进制度）
- **LON** – 位置经度（十进制度）
- **ELEV** – 位置海拔（米）
- **DAY** – 观察日 (python datetime.date)
- **IRRAD** – 传入的全球辐射（焦耳/平方米/天）
- **TMIN** – 每日最低温度（摄氏度）
- **TMAX** – 每日最高温度（摄氏度）
- **VAP** – 日平均蒸气压 (hPa)
- **RAIN** – 每日总降雨量（厘米/天）
- **WIND** – 2 米高度处的日平均风速（米/秒）
- **E0** – 开放水域的每日蒸发率（厘米/天）
- **ES0** – 裸土的日蒸发率（厘米/天）
- **ET0** – 参考作物的每日蒸散率（厘米/天）

有两个可选的关键字参数：

参数：
- **TEMP** – 每日平均温度（摄氏度），否则将由 (TMAX+TMIN)/2 得出。
- **SNOWDEPTH** – 积雪深度（厘米）

add_variable（变量名、值、单位） [来源]

添加具有 <value> 和给定 <unit> 的属性 <varname>

参数：
- **varname** – 要设置为属性名称的变量名称（字符串）
- **value** – 要添加的变量（属性）的值。
- **unit** – 变量单位的字符串表示。仅用于打印 WeatherDataContainer 的内容。

定义的信号

该模块定义和描述了 PCSE 使用的信号

信号被 PCSE 用于通知组件诸如播种、收获和终止等事件。任何 SimulationObject 都可以通过其*SimulationObject._send_signal()*方法发送事件。同样，任何 SimulationObject 都可以通过*SimulationObject._connect_signal()*方法注册处理程序来接收信号。变量可以通过位置参数或关键字参数传递给信号处理程序。但是，强烈建议不要在发送信号时使用位置参数，以避免位置参数和关键字参数之间的冲突。

一个示例可以帮助阐明信号在 PCSE 中的使用方式，但还可以查看PyDispatcher包的文档以获取更多信息：

```
import sys, os
import math
sys.path.append('/home/wit015/Sources/python/pcse/')
import datetime as dt

import pcse
from pcse.base import SimulationObject, VariableKiosk

mysignal = "My first signal"

class MySimObj(SimulationObject):

    def initialize(self, day, kiosk):
        self._connect_signal(self.handle_mysignal, mysignal)

    def handle_mysignal(self, arg1, arg2):
        print "Value of arg1,2: %s, %s" % (arg1, arg2)

    def send_signal_with_exact_arguments(self):
        self._send_signal(signal=mysignal, arg2=math.pi, arg1=None)

    def send_signal_with_more_arguments(self):
        self._send_signal(signal=mysignal, arg2=math.pi, arg1=None,
                          extra_arg="extra")

    def send_signal_with_missing_arguments(self):
        self._send_signal(signal=mysignal, arg2=math.pi, extra_arg="extra")


# Create an instance of MySimObj
day = dt.date(2000,1,1)
k = VariableKiosk()
mysimobj = MySimObj(day, k)

# This sends exactly the right amount of keyword arguments
mysimobj.send_signal_with_exact_arguments()

# this sends an additional keyword argument 'extra_arg' which is ignored.
mysimobj.send_signal_with_more_arguments()

# this sends the signal with a missing 'arg1' keyword argument which the handler
# expects and thus causes an error, raising a TypeError
```

```
try:
    mysimobj.send_signal_with_missing_arguments()
except TypeError, exc:
    print "TypeError occurred: %s" % exc
```

将此代码保存为文件*test_signals.py*并导入它会产生以下输出：

```
>>> import test_signals
Value of arg1,2: None, 3.14159265359
Value of arg1,2: None, 3.14159265359
TypeError occurred: handle_mysignal() takes exactly 3 non-keyword arguments (1 given)
```

目前，以下信号在 PCSE 中使用，并带有以下关键字。

CROP_START

表示新的作物周期将开始：

self._send_signal(signal=signals.crop_start, day=<date>,

crop_name=<string>, variety_name=<string>, crop_start_type=<string>, crop_end_type=<string>)

*signals.crop_start*关键字参数：

- 日：当前日期
- crop_name：标识作物的字符串
- variety_name：标识作物品种的字符串
- crop_start_type：“播种”或“出苗”
- crop_end_type：“成熟”、“收获”或“最早”

裁剪完成

表示当前作物周期结束：

```
self._send_signal(signal=signals.crop_finish, day=<date>,
                  finish_type=<string>, crop_delete=<True|False>)
```

*signals.crop_finish*关键字参数：

- 日：当前日期
- finish_type：描述完成模拟的原因的字符串，例如成熟度、收获、所有叶子死亡、达到的最大持续时间等。
- crop_delete：当必须从系统中删除 CropSimulation 对象时设置为 True，例如为了实施作物轮作。默认为假。

终止

表示整个系统应该终止（作物和土壤水分平衡）并且应该收集终端输出：

```
self._send_signal(signal=signals.terminate)
```

没有为此信号定义关键字参数

输出

指示应保存模型状态以备后用：

```
self._send_signal(signal=signals.output)
```

没有为此信号定义关键字参数

SUMMARY_OUTPUT

指示应保存模型状态以备后用，SUMMARY_OUTPUT 仅在收到指示作物模拟必须完成的 CROP_FINISH 信号时生成：

```
self._send_signal(signal=signals.output)
```

没有为此信号定义关键字参数

申请**NPK**

用于施用硝酸盐/磷酸盐/钾 (N/P/K) 肥料：

```
self._send_signal(signal=signals.apply_npk, N_amount=<float>, P_amount=<float>, K_amount=<float>,
                  N_recovery<float>, P_recovery=<float>, K_recovery=<float>)
```

*signals.apply_npk*的关键字参数：

- N/P/K_amount：当天施用的肥料量（kg/ha）。
- N/P/K_recovery：给定类型肥料的回收率

灌溉

用于发送灌溉事件：

```
self._send_signal(signal=signals.irrigate, amount=<float>, efficiency=<float>)
```

*signals.irrigate*的关键字参数：

- amount：当天的灌溉量，cm 水。
- 效率：灌溉效率，意味着添加到土壤水库的水总量等于数量 * 效率

割草

用于发送 LINGRA/LINGRA-N 型号使用的割草事件：

```
self._send_signal(signal=signals.mowing, biomass_remaining=<float>)
```

带有*signals.mowing*的关键字参数：

- biomass_remaining：割草后剩余的生物量，以 kg/ha 为单位。

公用事业

实用程序部分介绍用于从文件或数据库中读取天气数据和参数值的工具。

读取输入文件的工具

file_input 工具包含几个用于读取天气文件、参数文件和农业管理文件的类。

班级 **pcse.fileinput.CABOFileReader**（姓名） [来源]

读取带有模型参数定义的 CABO 文件。

Wageningen作物模型的参数定义一般都是用CABO格式写的。此类读取内容，解析参数名称/值并将它们作为字典返回。

参数： 名字– 要读取和解析的参数文件

退货： 带有参数键/值对的字典类对象。

请注意，此类尚未完全支持读取 CABO 文件的所有功能。例如，布尔值、日期/时间和表格参数的解析不受支持，会导致错误。

CABO 文件的标题（在第一行用 ** 标记）被读取，并且可以通过 get_header() 方法或仅通过在返回的字典上打印来检索。

例子

参数文件“parfile.cab”如下所示：

```
** CROP DATA FILE for use with WOFOST Version 5.4, June 1992
**
** WHEAT, WINTER 102
** Regions: Ireland, central en southern UK (R72-R79),
**          Netherlands (not R47), northern Germany (R11-R14)
CRPNAM='Winter wheat 102, Ireland, N-U.K., Netherlands, N-Germany'
CROP_NO=99
TBASEM   = -10.0    ! lower threshold temp. for emergence [cel]
DTSMTB   =   0.00,    0.00,     ! daily increase in temp. sum
           30.00,   30.00,     ! as function of av. temp. [cel; cel d]
           45.00,   30.00
** maximum and minimum concentrations of N, P, and K
** in storage organs        in vegetative organs [kg kg-1]
NMINSO   =   0.0110 ;       NMINVE   =   0.0030
```

可以用下面的语句来阅读：

```
>>>fileparameters = CABOFileReader('parfile.cab')
>>>print fileparameters['CROP_NO']
99
>>>print fileparameters
** CROP DATA FILE for use with WOFOST Version 5.4, June 1992
**
** WHEAT, WINTER 102
** Regions: Ireland, central en southern UK (R72-R79),
**          Netherlands (not R47), northern Germany (R11-R14)
```

```
-----------------------------------
TBASEM: -10.0 <type 'float'>
DTSMTB: [0.0, 0.0, 30.0, 30.0, 45.0, 30.0] <type 'list'>
NMINVE: 0.003 <type 'float'>
CROP_NO: 99 <type 'int'>
CRPNAM: Winter wheat 102, Ireland, N-U.K., Netherlands, N-Germany <type 'str'>
NMINSO: 0.011 <type 'float'>
```

班级 **pcse.fileinput.CABOWeatherDataProvider**(*fname* , *fpath=None* , *ETmodel='PM'* , *distance=1*) [来源]

CABO 天气文件阅读器。

参数：
- **fname** – 要读取的 CABO 天气文件的根名称
- **fpath** – 查找文件的路径，可以是绝对路径或相对路径。
- **ETmodel** – “PM”|“P” 用于选择 penman-monteith 或 Penman 方法作为参考蒸散。默认为“下午”。
- **distance** – 气象变量的最大插值距离，默认为 1 天。

退货： 可调用的对象，带有按日期键入的气象记录。

用 FORTRAN 或 FST 编写的 Wageningen 作物模型通常使用 CABO 天气系统 (http://edepot.wur.nl/43010) 来存储和读取天气数据。此类实现了 CABO 天气文件的读取器，还实现了附加功能，例如在缺少值的情况下对天气数据进行插值、将日照持续时间转换为全球辐射估计以及计算水、土壤和植物的参考蒸散值（E0、ES0 , ET0) 使用 Penman 方法。

与旧 CABOWE 系统的不同之处在于，python 实现将读取并存储特定站点可用的所有文件（例如年份），而不是在跨越年份边界时加载新文件。

□ 笔记

为了与 WOFOST 兼容，一些转换由 CABOWeaterDataProvider 从 CABO 天气文件中的单位完成：

- 蒸气压 从 kPa 到 hPa
- 辐射从 kJ/m2/day 到 J/m2/day
- 降雨从毫米/天到厘米/天。
- 所有蒸发/蒸腾速率也以厘米/天为单位返回。

例子

文件“nl1.003”提供了瓦赫宁根站 2003 年的天气数据，可以在 WOFOST 模型分发的 cabowe/ 文件夹中找到。可以使用以下方式读取此文件：

```
>>> weather_data = CABOWeatherDataProvider('nl1', fpath="./meteo/cabowe")
>>> print weather_data(datetime.date(2003,7,26))
Weather data for 2003-07-26 (DAY)
IRRAD:  12701000.00  J/m2/day
 TMIN:        15.90   Celsius
 TMAX:        23.00   Celsius
  VAP:        16.50       hPa
 WIND:         3.00     m/sec
 RAIN:         0.12    cm/day
   E0:         0.36    cm/day
  ES0:         0.32    cm/day
  ET0:         0.31    cm/day
Latitude  (LAT):    51.97 degr.
Longitude (LON):     5.67 degr.
Elevation (ELEV):    7.0 m.
```

或者，可以将上面打印命令中的日期指定为格式为 YYYYMMDD 或 YYYYDDD 的字符串。

班级 **pcse.fileinput.PCSEFileReader**（姓名） [来源]

PCSE 格式参数文件的阅读器。

此类是*CABOFileReader*的替代品。后者可以用来读取CABO格式的参数文件，但是这种格式有比较严重的局限性：它只支持字符串、整数、浮点数和数组参数。例如，不支持使用日期指定参数（除了将它们指定为字符串）。

PCSEFileReader是一种用于创建参数文件的更加通用的工具，因为它利用 python 解释器的强大功能通过 python 中的*execfile*功能处理参数文件。这意味着任何可以在 python 脚本中完成的事情也可以在 PCSE 参数文件中完成。

参数： 名字– 要读取和解析的参数文件

退货： 带有参数键/值对的字典对象。

例子

下面是参数文件“parfile.pcse”的示例。参数可以定义为“CABO”方式，但也可以通过导入模块、将参数定义为日期或 numpy 数组，甚至在数组上应用函数（在本例中为 *np.sin*）来使用高级功能：

```
"""This is the header of my parameter file.

This file is derived from the following sources
* dummy file for demonstrating the PCSEFileReader
* contains examples how to leverage dates, arrays and functions, etc.
"""

import numpy as np
import datetime as dt

TSUM1 = 1100
TSUM2 = 900
DTSMTB = [ 0., 0.,
           5., 5.,
          20., 25.,
          30., 25.]
AMAXTB = np.sin(np.arange(12))
cropname = 'alfalfa'
CROP_START_DATE = dt.date(2010,5,14)
```

可以用下面的语句来阅读：

```
>>>fileparameters = PCSEFileReader('parfile.pcse')
>>>print fileparameters['TSUM1']
1100
>>>print fileparameters['CROP_START_DATE']
2010-05-14
>>>print fileparameters
PCSE parameter file contents loaded from:
D:\UserData\pcse_examples\parfile.pw

This is the header of my parameter file.

This file is derived from the following sources
* dummy file for demonstrating the PCSEFileReader
* contains examples how to leverage dates, arrays and functions, etc.
DTSMTB: [0.0, 0.0, 5.0, 5.0, 20.0, 25.0, 30.0, 25.0] (<type 'list'>)
CROP_START_DATE: 2010-05-14 (<type 'datetime.date'>)
TSUM2: 900 (<type 'int'>)
```

```
cropname: alfalfa (<type 'str'>)
AMAXTB: [ 0.          0.84147098  0.90929743  0.14112001 -0.7568025
  -0.95892427  -0.2794155   0.6569866   0.98935825  0.41211849
  -0.54402111 -0.99999021] (<type 'numpy.ndarray'>)
TSUM1: 1100 (<type 'int'>)
```

班级 **pcse.fileinput.ExcelWeatherDataProvider**(*xls_fname* , *missing_snow_depth=None* , *force_reload=False*) [来源]

从 Excel 文件（仅限 .xlsx）读取天气数据。

参数:
- **xls_fname** – 要读取的 Excel 文件的名称
- **mising_snow_depth** – 应该用于缺少 SNOW_DEPTH 值的值，默认值为*None*。
- **force_reload** – 绕过缓存文件，从 .xlsx 文件重新加载数据并写入一个新的缓存文件。缓存文件写在*$HOME/.pcse/meteo_cache* 下

对于从文件中读取天气数据，最初只有 CABOWeatherDataProvider 可用，它可以从 CABO Weather 格式的文本文件中读取数据。然而，构建 CABO 天气文件非常繁琐，因为每年都必须构建一个新文件。此外，它很容易出错，格式错误很容易导致错误。

为了简化向 PCSE 模型提供天气数据的过程，编写了一个新的数据提供程序，它可以从简单的 excel 文件中读取数据

ExcelWeatherDataProvider 假定记录是完整的，并且不会努力插入数据，因为这可以在 Excel 本身中轻松完成。只允许缺少 SNOW_DEPTH，因为冬季以外通常不提供此参数。

班级 **pcse.fileinput.CSVWeatherDataProvider**(*csv_fname* , *delimiter=','* , *dateformat='%Y%m%d'* , *ETmodel='PM'* , *force_reload=False*) [来源]

从 CSV 文件中读取天气数据。

参数:
- **csv_fname** – 要读取的 CSV 文件的名称
- 定界符——CSV 定界符
- **dateformat** – 要读取的日期格式。默认为“%Y%m%d”
- **ETmodel** – “PM”|“P” 用于选择 Penman-Monteith 或 Penman 方法作为参考蒸散。默认为“PM”。
- **force_reload** – 忽略缓存文件并从 CSV 文件重新加载

CSV 文件应具有以下结构（示例），缺失值应添加为“NaN”:

```
## Site Characteristics
Country     = 'Netherlands'
Station     = 'Wageningen, Haarweg'
Description = 'Observed data from Station Haarweg in Wageningen'
Source      = 'Meteorology and Air Quality Group, Wageningen University'
Contact     = 'Peter Uithol'
Longitude = 5.67; Latitude = 51.97; Elevation = 7; AngstromA = 0.18; AngstromB = 0.55; HasSunshine
= False
## Daily weather observations (missing values are NaN)
DAY,IRRAD,TMIN,TMAX,VAP,WIND,RAIN,SNOWDEPTH
20040101,NaN,-0.7,1.1,0.55,3.6,0.5,NaN
20040102,3888,-7.5,0.9,0.44,3.1,0,NaN
20040103,2074,-6.8,-0.5,0.45,1.8,0,NaN
20040104,1814,-3.6,5.9,0.66,3.2,2.5,NaN
20040105,1469,3,5.7,0.78,2.3,1.3,NaN
[...]

with
IRRAD in kJ/m2/day or hours
TMIN and TMAX in Celsius (°C)
```

```
VAP in kPa
WIND in m/sec
RAIN in mm
SNOWDEPTH in cm
```

为了从文件中读取天气数据，最初可以使用 CABOWeatherDataProvider 从 CABO 天气格式的文本中读取数据。然而，构建 CABO 天气文件非常繁琐，因为每年都必须构建一个新文件。此外，它很容易出错，格式错误很容易导致错误。

为了简化向 PCSE 模型提供天气数据的过程，从 ExcelWeatherDataProvider 派生了一个新的数据提供程序，它从简单的 CSV 文件中读取数据。

CSVWeatherDataProvider 假定记录是完整的，并且不会努力插入数据，因为这可以在文本编辑器中轻松完成。只允许缺少 SNOWDEPTH，因为在冬季以外通常不提供此参数。

班级 **pcse.fileinput.YAMLAgroManagementReader**（姓名） [来源]

读取 YAML 格式的 PCSE 农业管理文件。

参数： 名字– 农业管理文件的文件名。如果 fname 未作为绝对路径或相对路径提供，则假定文件位于当前工作目录中。

班级 **pcse.fileinput.YAMLCropDataProvider**(*fpath=None , repository=None , force_reload=False*) [来源]

用于读取以 YAML 格式存储的作物参数集的作物数据提供程序。

参数 **fpath:** 包含 YAML 文件的目录的完整路径

参数库：

包含 YAML 文件的存储库的 URL。此 url 应该是原始内容（例如以“ https://raw.githubusercontent.com ”开头）

参数 **force_reload**：

如果设置为 True，则忽略缓存文件并重新加载所有参数（默认为 False）。

该作物数据提供者可以读取和存储多种作物的参数集，这与大多数只能保存单一作物数据的其他作物数据提供者不同。因此，该作物数据提供者适合运行不同作物类型的作物轮作，因为数据提供者可以切换活动作物。

最基本的用法是调用不带参数的 YAMLCropDataProvider。它将从我的 github 存储库中提取裁剪参数， 网址为https://github.com/ajwdewit/WOFOST_crop_parameters：

```
>>> from pcse.fileinput import YAMLCropDataProvider
>>> p = YAMLCropDataProvider()
>>> print(p)
YAMLCropDataProvider - crop and variety not set: no activate crop parameter set!
```

所有作物和品种都已从 YAML 文件中加载，但尚未设置激活作物。因此，我们需要激活一个特定的作物和品种：

```
>>> p.set_active_crop('wheat', 'Winter_wheat_101')
>>> print(p)
YAMLCropDataProvider - current active crop 'wheat' with variety 'Winter_wheat_101'
```

```
Available crop parameters:
 {'DTSMTB': [0.0, 0.0, 30.0, 30.0, 45.0, 30.0], 'NLAI_NPK': 1.0, 'NRESIDLV': 0.004,
 'KCRIT_FR': 1.0, 'RDRLV_NPK': 0.05, 'TCPT': 10, 'DEPNR': 4.5, 'KMAXRT_FR': 0.5,
 ...
 ...
 'TSUM2': 1194, 'TSUM1': 543, 'TSUMEM': 120}
```

此外，还可以从本地文件系统加载 YAML 参数文件：

```
>>> p = YAMLCropDataProvider(fpath=r"D:\UserData\sources\WOFOST_crop_parameters")
>>> print(p)
YAMLCropDataProvider - crop and variety not set: no activate crop parameter set!
```

最后，通过指定该存储库的 URL，可以从我的 github 存储库的分支中提取数据：

```
>>> p =
YAMLCropDataProvider(repository="https://raw.githubusercontent.com/<your_account>/WOFOST_crop_parame
```

为了提高加载参数的性能，YAMLCropDataProvider 将创建一个缓存文件，与加载 YAML 文件相比，该文件可以更快地恢复。从本地文件系统读取 YAML 文件时，请注意确保在对本地 YAML 进行更新时重新创建缓存文件。但是，应该强调的是，当从 URL 检索参数时这是不可能的，并且存在从过时的缓存文件加载参数的风险。在这种情况下，使用*force_reload=True*强制从 URL 加载参数。

简单或虚拟数据提供者

在不需要或不实用单独的文件或数据库的情况下，此类数据提供者可用于提供参数值。一个例子是用于模拟潜在生产条件的土壤参数集，其中参数值无关紧要，但必须向模型提供一些值。

班级 **pcse.util.DummySoilDataProvider** [来源]

这个类是为潜在的生产模拟提供一些虚拟土壤参数。

潜在生产水平的模拟与土壤无关。尽管如此，该模型没有一些参数值。此数据提供程序针对这种情况提供了一些硬编码参数值。

班级 **pcse.util.WOFOST72SiteDataProvider**(***夸格斯**) [来源]

WOFOST 7.2 的站点数据提供程序。

WOFOST 7.2 的站点特定参数可以通过此数据提供程序以及普通的 Python 字典提供。实施此数据提供程序的唯一目的是记录和检查 WOFOST 的站点参数，并提供合理的默认值。

可以通过此数据提供程序设置以下站点特定参数值：

```
- IFUNRN    Indicates whether non-infiltrating fraction of rain is a function of storm size (1)
            or not (0). Default 0
- NOTINF    Maximum fraction of rain not-infiltrating into the soil [0-1], default 0.
- SSMAX     Maximum depth of water that can be stored on the soil surface [cm]
- SSI       Initial depth of water stored on the surface [cm]
- WAV       Initial amount of water in total soil profile [cm]
- SMLIM     Initial maximum moisture content in initial rooting depth zone [0-1], default 0.4
```

目前只有 WAV 的值是强制指定的。

班级 **pcse.util.WOFOST80SiteDataProvider(**夸格斯)** [来源]

WOFOST 8.0 的站点数据提供程序。

WOFOST 8.0 的站点特定参数可以通过此数据提供程序以及普通的 Python 字典提供。实施此数据提供程序的唯一目的是记录和检查 WOFOST 的站点参数，并提供合理的默认值。

可以通过此数据提供程序设置以下站点特定参数值：

```
- IFUNRN        Indicates whether non-infiltrating fraction of rain is a function of
                storm size (1) or not (0). Default 0
- NOTINF        Maximum fraction of rain not-infiltrating into the soil [0-1],
                default 0.
- SSMAX         Maximum depth of water that can be stored on the soil surface [cm]
- SSI           Initial depth of water stored on the surface [cm]
- WAV           Initial amount of water in total soil profile [cm]
- SMLIM         Initial maximum moisture content in initial rooting depth zone [0-1],
                default 0.4
- CO2           Atmospheric CO2 level (ppm), default 360.
- BG_N_SUPPLY   Background N supply through atmospheric deposition in kg/ha/day. Can be
                in the order of 25 kg/ha/year in areas with high N pollution. Default 0.0
- NSOILBASE     Base N amount available in the soil. This is often estimated as the nutrient
                left over from the previous growth cycle (surplus nutrients, crop residues
                or green manure).
- NSOILBASE_FR  Daily fraction of soil N coming available through mineralization
- BG_P_SUPPLY   Background P supply in kg/ha/day. Usually this is mainly through deposition
                of dust and an order of magnitude smaller then N deposition. Default 0.0
- PSOILBASE     Base P amount available in the soil.
- PSOILBASE_FR  Daily fraction of soil P coming available through mineralization
- BG_K_SUPPLY   Background P supply in kg/ha/day. Default 0.0
- KSOILBASE     Base K amount available in the soil
- KSOILBASE_FR  Daily fraction of soil K coming available through mineralization
- NAVAILI       Amount of N available in the pool at initialization of the system [kg/ha]
- PAVAILI       Amount of P available in the pool at initialization of the system [kg/ha]
- KAVAILI       Amount of K available in the pool at initialization of the system [kg/ha]
```

目前，必须指定初始可用水量 (WAV) 和初始可用养分参数（NAVAILI、PAVAILI、KAVAILI）。

数据库工具

数据库工具包含用于从为不同版本的欧洲作物生长监测系统实施的数据库结构中检索农业管理、参数值和天气变量的函数和类。

请注意，数据提供者仅提供读取数据的功能，此处没有将模拟结果写入*CGMS* 数据库的工具。这是有意为之的，因为写入数据可能是一件复杂的事情，根据我们的经验，使用专用的数据库加载器工具（例如用于 ORACLE 的SQLLoader或MySQL 的语法）可以更轻松地完成此操作`load data infile`

CGMS8数据库

CGMS8 工具用于从 CGMS 可执行版本 9 和 10 使用的数据库结构中读取数据。

班级 **pcse.db.cgms8.GridWeatherDataProvider**（引擎，*grid_no*，*start_date=None*，*end_date=None*，*recalc_ET=False*，*use_cache=True* ） [来源]

从 CGMS 数据库的 GRID_WEATHER 表中检索气象数据。

参数：
- 元数据——提供数据库访问的 SqlAlchemy 元数据对象
- **grid_no** – CGMS 网格 ID
- **startdate** – 检索以 startdate 开始的气象数据（datetime.date 对象）
- **enddate** – 检索截至并包括 enddate 的气象数据（datetime.date 对象）

recalc_ET – 设置为 True 以强制计算参考 ET 值。当未在 CGMS 数据库中计算值时最有用。
- **use_cache** – 设置为 False 以忽略读/写缓存文件。

请注意，所有气象数据首先从数据库中检索并存储在内部。因此，没有数据库连接存储在类实例中。这使得类实例可以被腌制。

班级 **pcse.db.cgms8.SoilDataIterator**（引擎，*grid_no*） [来源]

CGMS8 的土壤数据迭代器。

唯一的区别是，在 CGMS8 中，该表称为“ELEMENTARY_MAPPING_UNIT”，在 CGMS12 中，它称为“EMU”

班级 **pcse.db.cgms8.CropDataProvider**（引擎，*grid_no*，*crop_no*，*campaign_year*） [来源]

从表 CROP_CALENDAR、CROP_PARAMETER_VALUE 和 VARIETY_PARAMETER_VALUE 中检索给定 grid_no、crop_no 和年份的作物参数。

参数：
- 引擎——提供数据库访问的 SqlAlchemy 引擎对象
- **grid_no** – 整数网格 ID，映射到表中的 GRID_NO 列
- **crop_no** – 整数作物 ID，映射到表中的 CROP_NO 列
- **campaign_year** – 整数活动年份，映射到表中的 YEAR 列。运动年通常是指收获的年份。因此，对于跨日历年的作物，start_date 可以是前一年。

班级 **pcse.db.cgms8.STU_Suitability**（引擎，*crop_no*） [来源]

为给定的 crop_no 返回合适的 STU 的 set()。

参数：
- 引擎——提供数据库访问的 SqlAlchemy 引擎对象
- **crop_no** – 整数作物 ID，映射到表中的 CROP_NO 列

班级 **pcse.db.cgms8.SiteDataProvider**（引擎、*grid_no*、*crop_no*、*campaign_year*、*stu_no*） [来源]

提供来自表 INITIAL_SOIL_WATER 和 SITE 的站点数据。

参数：
- 引擎——提供数据库访问的 SqlAlchemy 引擎对象
- **grid_no** – 网格编号（整数）
- **crop_no** – 作物数量（整数）
- **campaign_year** – 活动年份（整数）
- **stu_no** – 土壤类型单元编号 (int)

请注意，参数 SSI（初始表面存储）设置为零

此外，水平衡的开始日期由 GIVEN_STARTDATE_WATBAL 列定义。该值可以作为属性*start_date_waterbalance*访问。

CGMS12 数据库

CGMS12 工具用于从 CGMS 可执行版本 11 和 BioMA 2014 使用的 CGMS12 数据库结构中读取数据。

用于从 CGMS12 兼容数据库读取天气数据和计时器、土壤和场地参数的工具。

班级 **pcse.db.cgms12.WeatherObsGridDataProvider**（引擎，*grid_no*，*start_date=None*，*end_date=None*，*recalc_ET=False*，*recalc_TEMP=False*，*use_cache=True*） [来源]

从 CGMS12 兼容数据库中的 WEATHER_OBS_GRID 表中检索气象数据。

参数：
- 引擎——提供数据库访问的 SqlAlchemy 引擎对象
- **grid_no** – 用于检索数据的网格编号 (int)
- **start_date** – 检索以 start_date 开头的气象数据（datetime.date 对象）
- **end_date** – 检索直到并包括 end_date（datetime.date 对象）的气象数据
- **recalc_ET** – 设置为 True 以强制计算参考 ET 值。当未在 CGMS 数据库中计算值时最有用。
- **recalc_TEMP** – 设置为 True 以强制根据 TMIN 和 TMAX 计算日平均温度 (TEMP)：TEMP = (TMIN+TMAX)/2。

请注意，所有气象数据首先从数据库中检索并存储在内部。因此，没有数据库连接存储在类实例中。这使得类实例可以被腌制。

如果未提供 start_date 和 end_date，则检索网格的整个时间序列。

班级 **pcse.db.cgms12.AgroManagementDataProvider**（引擎、*grid_no*、*crop_no*、*campaign_year*、*campaign_start=None*） [来源]

用于从 CGMS12 数据库中的 CROP_CALENDAR 表提供农业管理数据的类。

参数：
- 引擎——提供数据库访问的 SqlAlchemy 引擎对象
- **grid_no** – 整数网格 ID，映射到表中的 grid_no 列
- **crop_no** – crop 的整数 id，映射到表中的 crop_no 列
- **campaign_year** – 整数活动年份，映射到表中的 YEAR 列。运动年通常是指收获的年份。因此，对于跨日历年的作物，start_date 可以是前一年。
- **campaign_start** –
  可用于定义活动开始的可选关键字。请注意，默认情况下，campaign_start_date 设置为等于 crop_start_date，这意味着模拟在裁剪开始时开始。可以使用此关键字更改此默认行为。它可以有多种含义：
  - 如果传递了日期对象，则假定活动从该日期开始。
  - 如果传递了 int/float，则 campaign_start_date 计算为 crop_start_date 减去 campaign_start 提供的天数。

要调整 campaign_start_Date，另请参阅*set_campaign_start_date(date)*方法以更新现有 AgroManagementDataProvider 上的 campaign_start_date。

set_campaign_start_date（开始日期） [来源]

更新 campaign_start_date 的值。

这仅在 CGMS12 中的 INITIAL_SOIL_WATER 表定义不同的 campaign_start 时有用

班级 **pcse.db.cgms12.SoilDataIterator**（引擎，*grid_no*） [来源]

用于迭代 CGMS 网格中不同土壤的类。

这个类的实例表现得像一个列表，允许迭代 CGMS 网格中的土壤。一个例子：

```
>>> soil_iterator = SoilDataIterator(engine, grid_no=15060)
>>> print(soildata)
Soil data for grid_no=15060 derived from oracle+cx_oracle://cgms12eu:***@eurdas.world
  smu_no=9050131, area=625000000, stu_no=9000282 covering 50% of smu.
    Soil parameters {'SMLIM': 0.312, 'SMFCF': 0.312, 'SMW': 0.152, 'CRAIRC': 0.06,
                     'KSUB': 10.0, 'RDMSOL': 10.0, 'K0': 10.0, 'SOPE': 10.0, 'SM0': 0.439}
  smu_no=9050131, area=625000000, stu_no=9000283 covering 50% of smu.
    Soil parameters {'SMLIM': 0.28325, 'SMFCF': 0.28325, 'SMW': 0.12325, 'CRAIRC': 0.06,
                     'KSUB': 10.0, 'RDMSOL': 40.0, 'K0': 10.0, 'SOPE': 10.0, 'SM0': 0.42075}
>>> for smu_no, area, stu_no, percentage, soil_par in soildata:
...     print(smu_no, area, stu_no, percentage)
...
(9050131, 625000000, 9000282, 50)
(9050131, 625000000, 9000283, 50)
```

班级 **pcse.db.cgms12.CropDataProvider**（引擎，*grid_no*，*crop_no*，*campaign_year*） [来源]

从表 CROP_CALENDAR、CROP_PARAMETER_VALUE 和 VARIETY_PARAMETER_VALUE 中检索给定 grid_no、crop_no 和年份的作物参数。

参数：
- 引擎——提供数据库访问的 SqlAlchemy 引擎对象
- **grid_no** – 整数网格 ID，映射到表中的 GRID_NO 列
- **crop_no** – 整数作物 ID，映射到表中的 CROP_NO 列
- **campaign_year** – 整数活动年份，映射到表中的 YEAR 列。运动年通常是指收获的年份。因此，对于跨日历年的作物，start_date 可以是前一年。

班级 **pcse.db.cgms12.STU_Suitability**（引擎，*crop_no*） [来源]

为给定的 crop_no 返回合适的 STU 的 set()。

参数：
- 引擎——提供数据库访问的 SqlAlchemy 引擎对象
- **crop_no** – 整数作物 ID，映射到表中的 CROP_NO 列

班级 **pcse.db.cgms12.SiteDataProvider**（引擎、*grid_no*、*crop_no*、*campaign_year*、*stu_no*） [来源]

提供来自表 INITIAL_SOIL_WATER 和 SITE 的站点数据。

参数：
- 引擎——提供数据库访问的 SqlAlchemy 引擎对象
- **grid_no** – 网格编号（整数）
- **crop_no** – 作物数量（整数）
- **campaign_year** – 活动年份（整数）
- **stu_no** – 土壤类型单元编号 (int)

请注意，参数 SSI（初始表面存储）设置为零

此外，水平衡的开始日期由 GIVEN_STARTDATE_WATBAL 列定义。该值可以作为属性*start_date_waterbalance*访问。

CGMS14 数据库

CGMS14数据库是兼容WOFOST 2015 BioMA实施的数据库结构。请注意，CGMS14 数据库结构与 CGMS8 和 CGMS12 有很大不同。

NASA POWER 数据库

班级 **pcse.db.NASAPowerWeatherDataProvider**（纬度、经度、*force_update=False*、*ETmodel='PM'*）[来源]

用于将 NASA POWER 数据库与 PCSE 结合使用的 WeatherDataProvider

参数：
- 纬度——请求天气数据的纬度
- **longitude** – 请求天气数据的经度
- **force_update** – 设置为 True 以强制从 POWER 网站请求新数据。
- **ETmodel** – “PM”|“P” 用于选择 penman-monteith 或 Penman 方法作为参考蒸散。默认为“下午”。

NASA POWER 数据库是专门为农业气象应用而设计的每日天气数据的全球数据库。数据库的空间分辨率为 0.5x0.5 度（截至 2018 年）。它来自气象站的观测结果，并结合辐射等参数的卫星数据。

天气数据更新有大约 3 个月的延迟，这使得数据库不适合实时监测，但是 POWER 数据库对许多其他研究很有用，与 WOFOST 使用的月度天气数据相比是一个重大改进在过去。

有关 NASA POWER 数据库的更多信息，请参阅文档： http://power.larc.nasa.gov/common/AgroclimatologyMethodology/Agro_Methodology_Content.html

NASAPowerWeatherDataProvider从 th NASA POWER *API*检索天气，并进行必要的转换以与 PCSE 兼容。检索并存储数据后，内容将转储到二进制缓存文件中。如果对同一位置发出另一个请求，则会加载缓存文件，而不是向 NASA Power 服务器发出完整请求。

使用缓存文件直到它们超过 90 天。90 天后，NASAPowerWeatherDataProvider 将发出新请求以从 NASA POWER 服务器获取更新的数据。如果此请求失败，它将回退到现有的缓存文件。可以通过设置*force_update=True*来强制更新缓存文件。

最后，注意 0.5x0.5 度网格框内的任何纬度/经度都将产生相同的天气数据，例如，lat/lon 5.3/52.1 和 lat/lon 5.1/52.4 之间没有区别。然而，由于白天长度的微小差异，PCSE 模拟可能会出现细微差异。

便民套路

这些例程用于方便地启动 WOFOST 模拟以进行演示和教程。它们可以作为构建您自己的脚本的示例，但没有进一步的相关性。

pcse.start_wofost.start_wofost(*grid=31031* , *crop=1* , *year=2000* , *mode='wlp'* , *dsn=None*) [来源]

提供启动WOFOST实例的便捷接口

如果不带任何参数启动，例程将连接到演示数据库并初始化 WOFOST 用于西班牙 (grid_no=31031) 的冬小麦 (cropno=1) 2000 年限水生产 (mode='wlp')

参数：
- **grid** – 网格数，默认为 31031
- **crop** – 作物数量，默认为 1（演示数据库中的 winter-wheat）
- **year** - 开始年份，默认为 2000
- 模式——生产模式（'pp'或'wlp'），默认为'wlp'
- **dsn**——作为 SQLAlchemy 数据源名称的 PCSE DB 默认为*None*，在这种情况下，将建立与演示数据库的连接。

例子：

```
>>> import pcse
>>> wofsim = pcse.start_wofost(grid=31031, crop=1, year=2000,
...   mode='wlp')
>>>
>>> wofsim
<pcse.models.Wofost71_WLP_FD at 0x35f2550>
>>> wofsim.run(days=300)
>>> wofsim.get_variable('tagp')
15261.752187075261
```

杂项公用事业

许多杂项用于各种目的，例如用于线性插值的任意函数发生器 (AfGen) 和用于计算 Penman Penman/Monteith 参考蒸散量的函数、埃方程和天文计算（例如日长）。

pcse.util.reference_ET（*DAY*、*LAT*、*ELEV*、*TMIN*、*TMAX*、*IRRAD*、*VAP*、*WIND*、*ANGSTA*、*ANGSTB*、*ETMODEL='PM'*、***kwargs*） [来源]

计算参考蒸散值 E0、ES0 和 ET0。

开放水域（E0）和裸土蒸散量（ES0）采用修正的Penman方法计算，而参考冠层蒸散量采用修正的Penman或Penman-Monteith方法计算，后者是默认方法。

输入变量：

```
DAY      -  Python datetime.date object                    -
LAT      -  Latitude of the site                        degrees
ELEV     -  Elevation above sea level                      m
TMIN     -  Minimum temperature                            C
TMAX     -  Maximum temperature                            C
IRRAD    -  Daily shortwave radiation                   J m-2 d-1
VAP      -  24 hour average vapour pressure               hPa
WIND     -  24 hour average windspeed at 2 meter          m/s
ANGSTA   -  Empirical constant in Angstrom formula         -
ANGSTB   -  Empirical constant in Angstrom formula         -
ETMODEL  -  Indicates if the canopy reference ET should    PM|P
            be calculated with the Penman-Monteith method
            (PM) or the modified Penman method (P)
```

输出是一个元组（E0、ES0、ET0）：

```
E0       -  Penman potential evaporation from a free
            water surface [mm/d]
ES0      -  Penman potential evaporation from a moist
            bare soil surface [mm/d]
ET0      -  Penman or Penman-Monteith potential evapotranspiration from a
            crop canopy [mm/d]
```

□ 笔记

Penman-Monteith 算法仅对参考冠层有效，因此不用于计算裸土和开阔水域 (ES0, E0) 的参考值。

背景是 Penman-Monteith 模型基本上是一个地表能量平衡，其中净太阳辐射被划分为潜热和感热通量（忽略土壤热通量）。为了估计这种划分，PM 方法在地表温度和空气温度之间建立了联系。然而，仅当发生这种分区的表面对于潜热通量和感热通量相同时，PM 模型的假设才有效。

对于作物冠层，这个假设是有效的，因为冠层的叶子形成了潜热通量（通过气孔）和感热通量（通过叶片温度）被分割的表面。对于土壤，这个原理不适用，因为当土壤干燥时，蒸发前沿会迅速消失在表面以下，因此分区表面相同的假设不再成立。

对于水面，PM 的假设不成立，因为水面温度和净入射辐射之间没有直接关系，因为辐射被水柱吸收，水面温度由其他因素共同决定因素（混合等）。只有非常浅的水层（1 厘米）才可以应用 PM 方法。

对于裸露的土壤和开放水域，Penman 模型是首选。尽管它部分地遇到了同样的问题，但根据其经验风函数，它在开阔水域和裸土上的校准稍好一些。

最后，在作物模拟模型中，开阔水域蒸发和裸土蒸发仅起次要作用（播前条件和早期淹水稻），不值得投入大量精力来改进 E0 和 ES0 的参考值估计。

pcse.util.penman_monteith（天，*LAT*，*ELEV*，*TMIN*，*TMAX*，*AVRAD*，*VAP*，*WIND2*） [来源]

根据 Penman-Monteith 模型计算参考 ET0。

此例程计算参考作物冠层 (ET0) 的潜在蒸散率，单位为 mm/d。对于这些计算，粮农组织的分析遵循粮农组织出版物 计算作物需水量指南 - 粮农组织灌溉和排水文件 56

输入变量：

```
DAY   -  Python datetime.date object                -
LAT   -  Latitude of the site                      degrees
ELEV  - Elevation above sea level                    m
TMIN  - Minimum temperature                          C
TMAX  - Maximum temperature                          C
AVRAD - Daily shortwave radiation                 J m-2 d-1
VAP   - 24 hour average vapour pressure              hPa
WIND2 - 24 hour average windspeed at 2 meter         m/s
```

输出是：

ET0 - Penman-Monteith 潜在蒸腾

作物冠层的速率 [mm/d]

pcse.util.penman（天，*LAT*，*ELEV*，*TMIN*，*TMAX*，*AVRAD*，*VAP*，*WIND2*，*ANGSTA*，*ANGSTB*） [来源]

基于Penman模型计算E0、ES0、ET0。

此例程计算自由水面 (E0)、裸露土壤表面 (ES0) 和作物冠层 (ET0) 的潜在蒸发（蒸腾）率，单位为 mm/d。对于这些计算，遵循 Penman 的分析（Frere 和 Popov，1979；Penman，1948、1956 和 1963）。调用的子程序和函数：ASTRO、LIMIT。

输入变量：

```
DAY     -  Python datetime.date object                          -
LAT     -  Latitude of the site                        degrees
ELEV    -  Elevation above sea level                     m
TMIN    -  Minimum temperature                           C
TMAX    -  Maximum temperature                           C
AVRAD   -  Daily shortwave radiation                  J m-2 d-1
VAP     -  24 hour average vapour pressure               hPa
WIND2   -  24 hour average windspeed at 2 meter          m/s
ANGSTA  -  Empirical constant in Angstrom formula         -
```

```
ANGSTB  -  Empirical constant in Angstrom formula         -
```

输出是一个元组 (E0,ES0,ET0):

```
E0      -  Penman potential evaporation from a free water surface [mm/d]
ES0     -  Penman potential evaporation from a moist bare soil surface [mm/d]
ET0     -  Penman potential transpiration from a crop canopy [mm/d]
```

pcse.util.check_angstromAB(*xA* , *xB*) [来源]

例程检查埃系数的有效性。

这是“weather.for”中 FORTRAN 例程“WSCAB”的 python 版本。

pcse.util.wind10to2(风 *10*) [来源]

使用对数将 10m 处的风速转换为 2m 处的风速。风廓线

pcse.util.angstrom（天，纬度，*ssd*，*cA*，*cB* ） [来源]

使用埃方程计算全球辐射。

全球辐射是使用埃方程从日照持续时间得出的：globrad = Angot * (cA + cB * (sunshine / daylength)

参数：
- 日——观察日（日期对象）
- 纬度——观察的纬度
- **ssd**——观察到的日照时数
- **cA** – 埃 A 参数
- **cB** – 埃 B 参数

退货： 以焦耳/平方米/天为单位的全球辐射

pcse.util.doy（日） [来源]

将日期或日期时间对象转换为一年中的某一天（1 月 1 日 = doy 1）

pcse.util.limit（最小值，最大值，*v* ） [来源]

将 v 的范围限制在 min 和 max 之间

pcse.util.daylength（天，纬度，角度*=-4*，*_cache={}* ） [来源]

计算给定日期、高度和基准的日长。

参数：
- 日——日期/日期时间对象
- 纬度——位置的纬度
- **angle** – 当太阳在地平线下成角度时，光周期日长开始/结束。默认为 -4 度。

源自 WOFOST 例程 ASTRO.FOR 并简化为仅包括日长计算。正在缓存结果以提高性能

pcse.util.astro（天，纬度，辐射，*_cache={}* ） [来源]

Daniel van Kraalingen 的 ASTRO 程序的 python 版本。

该子程序计算天文日长、昼夜辐射特性，如大气传输、漫辐射等。

参数：
- 日——日期/日期时间对象
- 纬度——位置的纬度
- 辐射——每日全球入射辐射（焦耳/平方米/天）

输出是一个具有以下顺序和标签的命名元组：

```
DAYL      Astronomical daylength (base = 0 degrees)     h
DAYLP     Astronomical daylength (base =-4 degrees)     h
SINLD     Seasonal offset of sine of solar height       -
COSLD     Amplitude of sine of solar height             -
DIFPP     Diffuse irradiation perpendicular to
          direction of light                        J m-2 s-1
ATMTR     Daily atmospheric transmission                -
DSINBE    Daily total of effective solar height         s
ANGOT     Angot radiation at top of atmosphere      J m-2 d-1
```

作者：Daniel van Kraalingen 日期：1991 年 4 月

Python 版本作者：Allard de Wit 日期：2011 年 1 月

pcse.util.merge_dict(*d1* , *d2* , *overwrite=False*) [来源]

合并 d1 和 d2 的内容并返回合并后的字典

笔记：

- 字典 d1 和 d2 没有改变。
- 如果*overwrite=False*（默认），当存在重复键时将引发*RuntimeError*，否则 *d1* 中的任何现有键都会被 *d2* 静默覆盖。

班级 **pcse.util.Afgen**（*tbl_xy*，单位=无） [来源]

模拟 WOFOST 中的 AFGEN 功能。

参数：
- **tbl_xy** – XY 值对的列表或数组，描述 X 值应单调递增的函数。
- **unit** – 内插值返回给定 单位分配，如果未使用 Unum，则默认为 None。

返回内插值，该内插值与应发生插值的 absicca 值一起提供。

例子：

```
>>> tbl_xy = [0,0,1,1,5,10]
>>> f =  Afgen(tbl_xy)
>>> f(0.5)
0.5
>>> f(1.5)
2.125
>>> f(5)
10.0
>>> f(6)
10.0
>>> f(-1)
0.0
```

班级 **pcse.util.ConfigurationLoader**（配置） [来源]

用于从 PCSE 配置文件加载模型配置的类

参数: 配置– 包含模型配置的给定文件名的字符串

pcse.util.is_a_month（日） [来源]

如果日期在一个月的最后一天，则返回 True。

pcse.util.is_a_dekad（日） [来源]

如果日期在 dekad 边界上，即每月的 10 日、20 日或最后一天，则返回 True

pcse.util.is_a_week（天，工作日*=0* ） [来源]

默认工作日是星期一。星期一是 0，星期日是 6

pcse.util.load_SQLite_dump_file（转储文件名，*SQLite_db_*名） [来源]

从转储文件 <dump_file_name> 构建 SQLite 数据库 <SQLite_db_name>。

以前的