Ruminant Farm System Model (RuFaS)

Scientific Documentation

v.1; Updated March 15, 2026

Contents

Part I

Preface: A Day in the Life on a RuFaS Farm

This document provides a high-level overview of the daily operations of a ruminant farm, as modeled by RuFaS. The overview will be through the eyes of Dr. Rue Fas. One day, Dr. Rue woke up and decided the life of a computer scientist wasn't for her. So she bought a dairy farm, and this is how she operates it every day.

Feeding the Animals Second thing in the morning, after coffee, Dr. Rue checks the calendar to see if it's time to formulate a new ration for her animals.

If a new ration needs to be formulated, Dr. Rue heads over to the barn where she keeps all her forages and feed. While there, she checks how well all the forages are holding up. After considering the feeds available, the quality of each one, and how much each one costs, she decides on how productive she wants her animals to be. It is a difficult balancing act, but eventually a ration recipe is found that will keep the animals happy while keeping costs low and sustainability high until the calendar says the ration recipe needs to be reformulated.

When Dr. Rue doesn't need to formulate a new ration, she takes the last ration recipe she made and grabs all the feed and forage the animals need for the day out of the feed barn. If there isn't enough for the animals, then a new ration is formulated using the steps above.

Taking Care of the Animals After making sure all the animals' feed is taken care of, Dr. Rue moves on to taking care of all the other animal tasks. This includes milking the animals, making sure they're healthy, getting them pregnant, and numerous other tasks.

As dawn breaks on the RuFaS farm, the animals awaken after a restful night. Joe, the farm master, prepares a large whiteboard to track HerdReproductionStatistics, ensuring he has all the data he needs for the day's activities.
Daily Routine - After leaving their pens, each animal lines up in front of the barn. Julie, the diligent farm veterinarian, examines them one by one. This daily check includes updates on the following areas:

- Nutrients: Each animal begins its day with a quick phosphorus requirement assessment. Julie records these details and updates the animal's phosphorus-related metrics.
- Digestive System: With nutrient information in hand, Julie shifts focus to each animal's digestive system, estimating manure output and enteric methane production for that day.
- Milk Production: All milking cows are assessed to gauge daily milk yield and milk component content (fat and protein). Any necessary adjustments are noted.
- Animal Growth: Around midday, Julie returns to each animal to evaluate daily weight changes. She updates each animal's body weight accordingly.
- Reproduction: HeiferII and Cow stage animals undergo a reproduction check. Julie closely monitors their reproductive progress and updates the HerdReproductionStatistics details on Joe's whiteboard.
- Life Stage Evaluation: Once the daily routines are complete, Julie determines if any animal is ready to progress to the next life stage. Graduating animals are marked as "graduated" to be moved to their new pens. Animals that must leave the herd—due to health, productivity, or other issues—are labeled as "sold" or "dead" to be removed from the farm population.
- Herd Size Maintenance: After every animal has been evaluated, Joe counts the herd to ensure it remains at a healthy size. If it is too large, he arranges the sale of some animals. If the herd is too small, he may opt to purchase additional heiferIIIs to keep numbers at the ideal level.

- Herd Structure Management: Joe also manages pen assignments. Newly graduated or newborn animals are guided to suitable pens, while animals marked for removal (sold or deceased) are taken out of the herd and removed from the farm records.

Output Reporting At day's end, Joe and Julie review the updated HerdReproductionStatistics on the whiteboard. Joe then compiles a concise report summarizing the day's herd stats. He forwards this information to the OutputManager before wrapping up another successful day on the RuFaS farm.

Dealing with the Manure Once all the animal chores are finished, Dr. Rue moves on to handling the manure. There are a lot of different ways that manure is collected from the animals: it is flushed out of the milking parlor with water, some of the pens need to have it shoveled out along with the bedding in the pen, and some pens have grates that allow the manure to go straight into a pit beneath the barn.

There are a lot of different places where manure is processed and stored on the farm. Dr. Rue goes from place to place on the farm, moving the manure from one place to the next. The manure chores are finished after moving all of the manure into her lagoons, compost piles, and other storages. As this final manure chore is completed, she writes down all the information about how much manure is in all these storages, and some information about what that manure is made of. How much water, nitrogen, phosphorus, other solids, etc. are in each storage.

Managing the Fields Now that Dr. Rue is done making sure all the manure has been handled and stored properly, she can take care of all the field tasks. She has a lot of fields on her farm, each with its own unique schedule. Her field management routine starts with checking each field's schedule to see if any of them are due to have manure applied, and if so, how much they should have applied. After noting all the manure that is supposed to go onto her fields, these amounts are checked against the amounts of manure that are in her manure storage. Dr. Rue then applies manure to each field on the schedule that day, trying to match the amount applied to the amount on the schedule as best she can.

After making sure every manure item on each farm's schedule has been taken care of for the day, the schedule for each field is rechecked, one field at a time. In each schedule she checks to see if that field needs to be tilled, if any chemical fertilizer needs to be applied, and if any crops need to be planted or harvested. After all these activities are taken care of, Mother Nature goes to work. The sun shines and rain falls, providing the field with water and energy that grows crops and soaks the soil. After taking care of each field one by one, Dr. Rue drives back to the main farm with all of the crops that she harvested on that day.

Storing the Feeds Now that she is back at the barns, all the crops that were just harvested can be stored. Fortunately she is very organized, so all her harvested crops are separate from one another and each has a label, indicating where it will be stored inside the feed barn. As each feed is stored, it is labeled with the current date. Dr. Rue is exhausted from another long day. She goes back to the farmhouse and rests up, preparing to do the whole thing over again tomorrow. Her daily routine might seem repetitive, but each day is a crucial part of the larger, continuous process that keeps her farm running efficiently and sustainably.

Part II
Animal Module

Contents

List of Figures

Contents

List of Tables

Contents

1 Module Introduction

The Animal Module simulates individual animals on a daily basis throughout each stage of life from birth until removal from the herd. In addition to the dynamic Monte-Carlo models that determine the occurrence of events throughout each animal's life, the Animal Module uses process-based, dynamic models to simulate the following processes:

Animal

- Individual animal growth
- Gestation and milk production
- Estimation of nutrient requirements and feed intake
- Enteric emission and manure excretion

Animals in the RuFaS model are divided into 5 life stages and a culling stage that were established based on the different management and physiological processes that need to be simulated for each class. The 5 classes and culling stage are described in more detail later in this document, but include: calves, heifers I, heifers II, heifers III, cow, and culled.
The attributes that define the animal characteristics and management are provided by the user and are comprehensively described in the inputs section of this module. When a simulation begins, the herd is populated by a semi-random draw of animals from a large, existing set of animal objects. This process and the process for regenerating the population of animals used to initialize the herd are described in detail in the The initially selected animal objects are then grouped into pens based on their life stage and the user-provided pen information. Once the herd has been set up, a regular progression of functions is called each day to simulate and report outputs of the processes required to manage the herd and the processes that define each animal's development and production. These daily updates are grouped into two classes in the RuFaS model: herd level daily routines (HerdManager) and animal level daily routines (AnimalManager). At the start of each day, the Herd-level daily update cycles through all the animals to call their daily update functions. Then based on the changes for each animal that resulted from the animal level daily updates, the HerdManager changes the animal life-stages, moves them to new pens, and removes them from the herd. If the herd size has decreased too much, the HerdManager will purchase new Heifer III animals to maintain the adult herd size. The HerdManager then checks if it is time to formulate a new diet (at a user-specified interval), summarizes and reports the herd statistics, and gathers and reports all of the manure excreted by the animals to pass to the Manure Module.
Daily outputs from the Animal module are provided at a variety of scales and time steps that range from reports at the individual animal level (e.g. animal event histories), animal class level (e.g. average dry-matter intake for Heifer II's), at the pen level (e.g. average net energy requirement or intake for Pen 3), and at the herd level (i.e. number of animals sold each day). A complete list of potential outputs from the Animal Module is provided in this document.

1.a Animal Life Cycle

The animal life cycle is driven by Monte Carlo stochastic processes that simulate the growth, production, reproduction, and culling of individual dairy cattle animals and shares aggregated outcomes with the herd management sub-module to manage herd dynamics on a daily basis.
The life cycle module represents the variation in animal performance through the use of Monte Carlo Simulation methods (Reuven and Kroese, 2017). Monte Carlo simulation relies on random draws from probability distributions rather than assigning fixed values. In this model, two main strategies are used:

- Event-based, in which a random draw from a uniform distribution (U(0,1)) is compared to the probability of an event occurring to simulate if that event occurs, and
- Population-based, in which a random draw from a known distribution of attribute values is assigned to an animal at instantiation. Probability distributions used for that purpose are Normal Gaussian or Empirical.

Each animal within the RuFaS herd progresses through five main life stages (calf, Heifer I, Heifer II, Heifer III, and Cow) as illustrated in the figure below.

- Calves progress to Heifer I when their age reaches the user-defined input for weaning day
- Heifer I progresses to Heifer II when their age reaches the user-defined input for breeding start day
- Heifer II animals that successfully conceive become Heifer III's when the number of days left in their pregnancy is equal to the user-defined input for prefresh days (default 21)
- When a Heifer III calves, she becomes a Cow for the rest of her life on the RuFaS farm
- Cows cycle through lactating and non-lactating phases until they are removed from the herd due to a failure to conceive or a user-provided probability for sale or death within each parity (lactation).

When a HeiferIII or Cow reaches full gestation length, they start lactating and a new calf object is created. The calf is assigned a birthweight according to a user-informed random distribution. All male calves are recorded and removed from the farm on day 1 of their life. Female calves either remain on the farm or are sold on day 1, determined by an event-based Monte Carlo method informed by the user-defined percent of female calves that are kept.

2 Herd-level Processes and Management

2.a Herd Initialization

Important Note

In RuFaS code, the term "herd initialization" is used to refer to both the process of generating an animal population file and of selecting from that population to create a specific herd at the start of a simulation. In this doc, those two processes will be differentiated by calling them "Generation" and "Selection", respectively.

"Animal population" can refer to both groups of animals (the pool of animals generated, and the herd of animals selected). In the code, the population is sometimes referred to as "pre_animal_-population" and "post_animal_population" to differentiate whether the specified group of animals exists before or after the random selection of individuals to create a herd.

Introduction Every simulation begins with a herd consisting of calves, heifers, and dry and lactating cows. This herd comes to be through a two-step process:

1. A population of animals is generated through simulation or loaded from data (pre-animal-population). The animals in that herd should reflect the attributes of animals in the herd to be modeled.
2. Animals are randomly selected with replacement from the pre-animal-population to be in the initial herd, based on group numbers and characteristics that reflect the distribution of animals within the herd being modeled. Note that the animals in the herd on day one will have attributes of the pre-animal-population and may deviate from the desired herd to be modeled.

For the rest of the simulation, animals enter the herd through birth as newborn calves or purchased replacements (heiferIII's).

Table 1: Overview of the distinct phases by which animals are created and introduced to the herd.

Before the simulation	**At the initiation of simulation**	**During the simulation**
Generate (simulate) or load an Animal Population file (pre-animal-population). The animals present at the start of the simulation, and those available in the replacement market, are drawn from this file.	Herd is established by random selection of animals from the Animal Population to match the user-specified group sizes and age/stage distribution (post-animal-population). Management attributes—including lactation-curve parameters—are assigned as each animal enters the herd.	New animals enter via **(a)** birth or **(b)** purchase of heifers from the replacement market. Purchased animals inherit two attributes from the population file: • Breed • Mature body weight

Required User Inputs

Table 2: User inputs in the animal input file that influence the generation of the Animal Population (pre-animal-population). The inputs of this table are found in the Herd Initialization Section

Variable	Definition	Default	Min	Max
Initial_animal_num	The initial number of animals that serve as a starting point for the population generation simulation	10000	0	-
Simulation_days	The number of days to simulate for the population generation simulation	5000	0	-

Table 3: User inputs in the animal input file that influence the generation of the Animal Population (pre-animal-population). The inputs of this table are found in the Herd Initialization Section.

Variable	Definition	Default	Min	Max
Calf_num	Number of Calves (head) – The initial number of pre-weaned calves	8	2	–
HeiferI_num	Number of HeiferI animals (head) – The initial number of heifers that are weaned but not yet bred	44	2	–
HeiferII_num	Number of HeiferII animals (head) – The initial number of heifers that are either eligible for breeding (based on user-inputted Breeding Start Day for heifers), or in early pregnancy	38	2	–
HeiferIII_num_springers	Number of HeiferIII animals (head) – The initial number of close-up heifers (within the user-defined close-up period, e.g., Prefresh Day)	5	2	–

Continued on next page

Table continued from previous page

Variable	**Definition**	**Default**	**Min**	**Max**
Cow_num	Number of Cows (head) – The initial number of dry and lactating cows	100	6	–
Replace_num	Replacements (head) – Number of replacement animals available in the replacement market	5000	50	–
Lactating_fraction	Fraction of cows that are lactating. Used during herd initialization	0.8356	0	1
Parity_fractions: 1, 2, 3, 4, 5	Fractions of the milking animal population that are parity 1, 2, 3, 4, and 5+. The sum must equal 1.0	0.36, 0.26, 0.18, 0.11, 0.09	0	1

Relevant Outputs
After the animals have been selected from the animal population to create the initial herd, a summary of the Population (pre-selection pool, aka pre-animal-population) and of the Initial herd (selected animals, aka post-animal-population) is provided to the output manager.

Class and function: AnimalModuleReporter.report_animal_population_statistics Output variables:

- .population_...
- .initial_...

... includes: breed, number of animals of each class (calf, heiferI, heiferII, heiferIII, cow), number of cows by parity and lactation status, average age of animals within each class, distribution of age of animals within each class, average body weight of animals within each class, average cow days in milk, days in preg, parity, and calving interval.

Methodology When a simulation begins, the HerdFactory class generates or loads the pre-animal-population, from which it selects and compiles the post-animal-population with an appropriate distribution of animal ages and stages to serve as the starting herd for simulation.

Designation of an Animal Population

Option 1: Creating a new Animal Population An animal population can be generated as a standalone simulation task that creates and saves a pre-animal-population file as the output based on the user Animal Module inputs. Or, the animal population can be generated as the first step of a regular simulation task which then uses that generated pre-animal-population to select from and create the post-animal-population to use for the subsequent herd simulation. See the documentation about Task Manager for more information about how to generate a new pre-animal-population.

The _generate_animals() method creates the pre_animal_population, with the option to save those animals to file at the end of the generation process.

Calves are created according to the number specified by the user (initial_animal_num), some are sold based on user inputs, and those remaining proceed through select daily updates (growth, reproduction, milk, and transition through the appropriate life stages) for the user-specified number of days (simulation_days). The simulation occurs similarly to the regular life cycle simulation process, but with an extended period and large number of animals. There are a few notable distinctions:

- After 3000 days of simulation (specified in animal_constants), a copy of each heiferIII is saved to the replacement herd. The copy is made on the day when she is ready to transition to a lactating cow (ready to give birth), so that when the copy enters the herd as a purchased heiferIII during the actual simulation, she will give birth the very next day.
- If a cow has made it to parity six without being culled, she is removed from the herd. (The maximum parity of cows kept in the generated herd is five).

After the simulation_days number of days, the full herd contains a mixture of calves, heifers, cows, and replacements. The core status information and life history of each animal are either saved to file or passed for random sampling to create the post-animal-population.

Option 2: Point to an existing Animal Population Alternatively, the pre-animal-population will be loaded from a specified Animal Population input file consisting of calf, heifer, cow, and replacement objects.

Herd Selection The _random_sample_with_replacement() method selects animals from each of the following groups: calf, heiferI, heiferII, heiferIII, replacement, lactating cows by parity for parities 1-5, and dry cows by parity for parities 1-5. Animals are sampled randomly with replacement from the pre-animal-population according to the numbers, parity fractions, and milking cow fraction specified in the animal input file. The resulting list of animals is returned as the post-animal-population and summaries of the pre- and post-animal-populations are reported to the output manager.

2.b Pens and Animal Grouping

Introduction All animals in RuFaS are assigned to a pen which is used to summarize nutrient requirements for diet formulation and feeding, aggregate enteric methane and manure excretions for reporting and passage to the Manure Module. Each pen tracks and updates daily a list of animals in the pen, the stocking density, the average nutrient requirement of the animals in the pen, the ration being fed to that pen, the manure excreted, and the methane emitted. In addition, constant attributes of each pen include the type of pen it is (freestall, tiestall, open lot, compost bedded pack barn), the type of animals that can be in that pen, the number stalls, the manure management practices associated with that pen, and the distance to the milking parlor for lactating cows. Each day after the HerdManager has updated the status of each animal and determined which animals need to be bought or sold, the animals are added and removed from pens as needed.

Required User Inputs

Table 4: The user must define at least 3 pens and for each pen, key inputs are:

Variable	Definition	Default	Min	Max
id	The index of the pen	NA	0	NA
pen_name	The name or identifier of a given pen	NA		
animal_combination	The valid combinations of animal types that can be in a pen. CALF = Calves; GROWING = HeiferI's and HeiferII's; CLOSE_UP = HeiferIII's and Dry Cows; LAC_COWS = Lactating Cows		CALF, GROWING CLOSE_UP, LAC_COW	
vertical_dist_to_milking_-parlor	The elevation gain (in meters) between the pen and the milking parlor	0.1	0	NA
horizontal_dist_to_milking_-parlor	The distance (in meters) between the pen and the milking parlor	10	0	NA
number_of_stalls	The number of stalls in the pen	1000	0	NA
pen_type	The type of pen (freestall, tiestall, open lot, compost bedded pack barn)	freestall	freestall, tiestall, open lot, compost bedded pack barn	
max_stocking_density	The maximum ratio of the number of animals in the pen to the number of stalls in the pen	1.2	0.5	5
manure_management_scenario_id	The ID number of the manure management practices that are used for this pen (scenarios are defined in manure management schema)	0	0	NA

Relevant Outputs

- **number_of_animals_in_pen_[id]_[AnimalCombination]** reports the number of animals in each pen each day.
- **ration_per_animals_for_pen_[id]_[AnimalCombination]** - reports average dry matter intake of an animal in the pen at the time of the ration formulation.
- **ration_per_animals_for_pen_[id]_[AnimalCombination].[RUFAS_feed_id]** - reports the amount of each feed that is delivered per animal in the pen
- **ration_nutrient_amount_for_pen__[AnimalCombination].[nutrient]** - is reporting the amount of each of the following nutrients that is provided by the ration per animal in the pen (all reported in kilograms unless otherwise indicated):

Methodology ***Animal Grouping*** To translate from the AnimalTypes needed to track an animal's life stages and the combinations of animals that are allowed to be grouped into a single pen, a new variable called AnimalCombination is created that combines some AnimalTypes to facilitate sorting into pens. The AnimalCombination class is defined in enums.py and includes the following mapping from AnimalTypes to AnimalCombinations:

Table 6: A summary of animal types

Animal Combination	Animal Types
CALF	Calves
GROWING	Heifer I and Heifer II
CLOSE_UP	Heifer III and Cows (non-lactating)
LAC_COW	Cows (lactating)

Currently there are only two options for assigning AnimalCombinations to pens.

- *Option 1* - The first and most commonly used option for grouping animals is to assign one AnimalCombination to each pen such that there is at least one pen for the 4 AnimalCombinations: CALF, GROWING, CLOSE_UP, and LAC_COW.
- *Option 2* - In some cases, and especially in very small farms, a user might wish to represent a scenario where all growing animals, pregnant heifers, and non-lactating cows are housed together. In this case, there is an option to have a pen that houses both GROWING and CLOSE_UP animals such that the there is at least one pen with each of the 3 AnimalCombination inputs: CALF, GROWING_AND_CLOSE_UP, and LAC_COW.

Pen Sizing Due to oscillations in the exact number of animals in each animal type, it is strongly recommended that the stocking density is at least 120% of the expected number of animals in each pen.

Class

Pen Pens are initiated from the Pen class by the HerdManager at the beginning of the simulation and, if needed, an additional pen is created to accommodate animals if the animal numbers have exceeded the max stocking density.
Noteworthy Functions

Remove_animals_by_ids This function is called when the HerdManager calls for: remove_animal_from_pen_and_id_map. It is pen specific and removes the animal from the pen and the animal ID from the animal ID map that tracks the animals in the pen.

_add_new_animals This function is called by Pen.update_animals which is in turn called by HerdManager._add_animal_to_pen_and_id_map. Thus, when HerdManager initiates addition of animals to each pen, this function executes the addition of those animals.

Set_animal_nutritional_requirements

2.c Ration Formulation

The diet recipes in RuFaS are either formulated through least-cost formulation (Li, Rosa, et al., 2022) or provided by the user for each of 4 animal categories (calves, growing heifers, dry and close-up cows, and lactating cows). The composition of the available feeds are provided by a feed library that was adapted from the National Academies of Sciences, Engineering, and Medicine, 2021. The amount of feed required by the farm is tracked by pen and multiple pens of each animal category can be simulated; however, at this time, the RuFaS model can only accept 1 user-defined diet per animal category even if there are multiple simulated pens for that animal category.
In practice, animal feeding on dairy farms is a constantly evolving process that responds to fluctuations in feed availability, animal requirements and responses, and management goals. Although the feed delivery algorithms will make small modifications to the diet recipe in response to animal nutrient requirements and adjust the total amount of feed delivered based on the number of animals and their requirements, the diversity and degree of fluctuation in the diets is less than what is expected on a commercial dairy. The RuFaS feed library can also be modified or expanded on by changing the compositions of the existing feeds or by adding new feeds.
Regardless of whether or not a user chooses to provide their own ration or have RuFaS build and feed a ration optimized for least-cost, RuFaS needs a set of ingredients to work from. The RuFaS feed library should be consulted for the full list of ingredients and ingredients that are in both the NASEM and NRC libraries can be used. Users interested in feeding a ration that they define, need to provide a list of ingredients and their relative % dry matter intake for each animal class (pre-weaned calves, growing heifers, close-up animals, and lactating cows).

If a user would like RuFaS to use the list of ingredients to formulate a least-cost ration, it is imperative that proper costs are inputted for each ingredient listed, but the % dry matter intake in the user defined ration percentages does not need to be included. Feed costs need to be entered in $ / kg DM. RuFaS will then use all the ingredients available (based on costs and inventory available if the feed is homegrown) to create a ration that meets the nutritional requirements of the animal class and is cost-optimized.

Important Note

For the time being, only one set of feeds and ration formula can be provided for each animal class.

2.c.i Energy and Nutrition Requirements

NRC: Calculating Nutrient Requirements of Dairy Cattle Below are the calculations utilized based on the energy and nutrient requirements of dairy cattle as reported by the NRC and more recently the NASEM (National Research Council, 2001; National Academies of Sciences, Engineering, and Medicine, 2021).

Calculations are included for the following requirement categories:

- Energy (Maintenance, Activity, Growth, Pregnancy, Lactation)
- Protein (Maintenance, Growth, Pregnancy, Lactation)
- Minerals (Maintenance, Growth, Pregnancy, Lactation)
- Dry Matter Intake
- Amino Acids (NASEM only)

Where appropriate, requirement calculations are differentiated by animal age and reproductive status.

2.c.ii Nutrient Requirements of Dairy Cattle (NRC)

Energy Requirements For details about the variables included in each calculation, please refer to the energy requirements key.

- ***Maintenance*** - The maintenance requirement is calculated based on metabolic body weight (body weight$^{0.75}$).

  Lactating and Dry Cows

$$CBW = MW * 0.06275$$

[AN.NRC.1]

$$CW = (18 + (DOP - 190) * 0.665) * (\frac{CBW}{45}), if DOP > 190.0$$

[AN.NRC.2]

Otherwise

$$NEmaint = 0.08 * (BW - CW)^{0.75}$$

[AN.NRC.3]

Heifers -The maintenance requirement is calculated based on metabolic body weight (body weight$^{0.75}$), body condition score, and the previous month's temperature.

$$CBW = MW * 0.06275$$

[AN.NRC.1]

$$CW = (18 + (DOP - 190) * 0.665) * (\frac{CBW}{45}), if DOP > 190.0$$

[AN.NRC.2]

Otherwise

$$BCS9 = (BCS5 - 1) * 2 + 1$$

[AN.NRC.6]

$$NEmaint = (BW - CW)^{0.75} * (0.086 * (0.8 + (BCS9 - 5) * 0.05) + 0.0007 * (20 - PrevTemp))$$

[AN.NRC.7]

- ***Activity*** - Activity requirement is proportional to body weight and daily walking distance. A grazing system and hilly topography will cost additional energy.

  Lactating and Dry Cows

$$NEa1 = 0.0012 * BW \text{ if Housing = Grazing } 0$$

[AN.NRC.8]

Otherwise

$$NEa2 = 0.006 * BW \text{ if Topography = Hilly } 0$$

[AN.NRC.9]

Otherwise

$$NEa2 = 0.Distance * 0.00045 * BW + NEa1 + NEa2$$

[AN.NRC.10]

  Heifers

$$NEa1 = 0.0009 * BW + 0.0016 * BW \text{ if Housing = Grazing } 0$$

[AN.NRC.11]

Otherwise

$$NEa2 = 0.006 * BW \text{ if Topography = Hilly } 0$$

[AN.NRC.12]

Otherwise

$$NEa2 = 0.Distance * 0.00045 * BW + NEa1 + NEa2$$

[AN.NRC.10]

- ***Growth***

  Lactating and Dry Cows Cows will continue to grow toward their mature body weight until the end of their second lactation.

$$MSBW = 0.96 * MW$$

[AN.NRC.13]

$$SBW = 0.96 * BW$$

[AN.NRC.14]

$$EBW = 0.96 * SBW$$

[AN.NRC.15]

$$EQSBW = (SBW - CW) * \frac{478}{MSBW} SBW$$

[AN.NRC.16]

$$ADG = \frac{(0.92-0.82)*MSBW}{CI}, if Parity = 1 \frac{(1-0.92)*MSBW}{CI}, if Parity = 2.0, if Parity > 2$$

[AN.NRC.17]

$$EQEBG = 0.956 * ADG$$

[AN.NRC.18]

$$EQEBW = 0.891 * EQSBW$$

[AN.NRC.19]

$$NEg = 0.0635 * EQEBW^{0.75} * EQEBG^{1.097}$$

[AN.NRC.20]

Heifers

$$ADG = \frac{0.55*MSBW-SBW}{(Age1stBred-Age)*30.4},\ before\ breeding\ \frac{0.82*MSBW-SBW}{(Age1st-Age)*30.4},\ after\ breeding$$

[AN.NRC.21]

$$EQEBG = 0.956 * ADG$$

[AN.NRC.22]

$$EQEBW = 0.891 * EQSBW$$

[AN.NRC.23]

$$NEg = 0.0635 * EQEBW^{0.75} * EQEBG^{1.097}$$

[AN.NRC.24]

- ***Pregnancy***

$$MEpreg = (2 * 0.00159 * DOP - 0.0352) * \frac{CBW}{45} * 0.14, if DOP > 190.0$$

$Otherwise$

[AN.NRC.25]

$$NEpreg = MEpreg * 0.664$$

[AN.NRC.26]

- ***Lactation***

$$Milk_{\text{Energy}} = 0.0929 * Fat\_Milk + \frac{0.0563}{0.93} * TP\_Milk + 0.0395 * Lactose\_Milk$$

[AN.NRC.27]

$$NElact = Milken * Milk$$

[AN.NRC.28]

Protein Requirements Reporting of protein requirements is divided into 4 components: maintenance, growth, pregnancy, and lactation (all in metabolizable protein). For details about the variables included in each calculation, please refer to the protein requirements key.

- ***Total Protein***

$$MPreq = MPm + MPg + MPpreg + MPlact$$

[AN.NRC.36]

- ***Maintenance***

  Lactating and Dry Cows

$$MPm = 0.3 * (BW - CW)^{0.6} + 4.1 * (BW - CW)^{0.5} + (DMI * 1000 * 0.03 - 0.5 * (\frac{MPbact}{0.8} - MPbact)) + 0.4 * 11.8 * \frac{DMI}{0.67}$$

[AN.NRC.29]

- ***Growth***

  Lactating and Dry Cows

$$NPg = 0, if ADG = 0, ADG * (268 - 29.4 * \frac{NEg}{ADG})$$

[AN.NRC.30]

$$EffMP\_NPg = \frac{(83.4-0.114*EQSBW)}{100}, if EQSBW \leq 478, 0.28908$$

[AN.NRC.31]

Otherwise

$$MPg = \frac{NPg}{EffMP\_NPg}$$

[AN.NRC.32]

Heifers

$$Npg = 0, if ADG = 0ADG * (268 - 29.4 * \frac{NEg}{ADG})$$

[AN.NRC.33]

$$EffMP\_NPg = \frac{(83.4-0.114*EQSBW)}{100}, if EQSBW \leq 478, 0.28908$$

[AN.NRC.34]

Otherwise

$$MPg = \frac{NPg}{EffMP\_NPg}$$

[AN.NRC.35]

$$MPreq = MPm + MPg + MPpreg + MPlact$$

[AN.NRC.36]

- ***Pregnancy***

$$MPpreg = (0.69 * DOP - 69.2) * \frac{CBW}{45*0.33}, if DOP > 190.0$$

[AN.NRC.37]

- ***Lactation***

$$MPlact = Milk * \frac{TP_{\text{Milk}}}{100} * \frac{1000}{0.67}$$

[AN.NRC.38]

Table 7: Key variables used in protein requirement calculations.

Variable	**Description**	**Units**
EffMP_NPg	Efficiency of converting metabolizable protein or net protein	
MPg	Metabolizable protein requirement for growth	g
MPlact	Metabolizable protein requirement for lactation	g
MPpreg	Metabolizable protein requirement for pregnancy	g
NPg	Net protein requirement for growth	g

Mineral Requirements

Important Note

Only calcium (Ca) and phosphorus (P) requirements are considered currently. For details about the variables included in each calculation, please refer to the mineral requirements key.

- ***Maintenance***

  Lactating and Dry Cows

  Calcium Equations

$$Ca_{\text{main}} = 0.031 * BW + 0.08 * \frac{BW}{100}, \text{ for lactating cows}$$
$$Ca_{\text{main}} = 0.0.0154 * BW + 0.08 * \frac{BW}{100}, \text{ for dry cows}$$

[AN.NRC.39]

Phosphorus Equations

$$P_{\text{main}} = 1 * DMI + 0.002 * BW, \text{ for lactating cows}$$
$$P_{\text{main}} = 0.8 * DMI + 0.002 * BW, \text{ for dry cows}$$

[AN.NRC.40]

<u>Heifers</u>

Calcium Equation

$$Ca_{\text{main}} = 0.0.0154 * BW + 0.08 * \frac{BW}{100},$$

[AN.NRC.41]

Phosphorus Equation

$$P_{\text{main}} = 0.8 * DMI + 0.002 * BW,$$

[AN.NRC.42]

- ***Growth***

  Calcium Equation

$$Ca_{\text{growth}} = 9.83 * MW^{0.22} * BW^{-0.22} * \frac{ADG}{0.96}$$

[AN.NRC.43]

Phosphorus Equation

$$P_{\text{growth}} = (1.2 + 4.635 * MW^{0.22} * BW^{-0.22}) * \frac{ADG}{0.96}$$

[AN.NRC.44]

- ***Pregnancy***

  Calcium Equation

$$Ca_{\text{preg}} = 0.02456 * exp0.05581 - 0.00007 * DOP * DOP - 0.02456 * exp0.05581 - 0.00007 * -1 * DOP - 1, if DOP > 190.0, else$$

[AN.NRC.45]

Phosphorus Equation

$$P_{\text{preg}} = 0.02743*exp0.05527-0.000075*DOP*DOP-0.02743*exp0.05527-0.000075*DOP-1*(DOP-1),$$

If: $DOP > 190.0$ $else$

[AN.NRC.46]

- ***Lactation***

  Calcium Equation

$$Ca_{\text{lact}} = 1.22 * Milk$$

[AN.NRC.47]

Phosphorus Equation

$$P_{\text{lact}} = 0.9 * Milk$$

[AN.NRC.48]

- ***Total Minerals***

  Calcium Equation

$$Ca_{\text{req}} = Ca_{\text{main}} + Ca_{\text{growth}} + Ca_{\text{preg}} + Ca_{\text{lact}}$$

[AN.NRC.49]

Phosphorus Equation

$$P_{\text{req}} = P_{\text{main}} + P_{\text{growth}} + P_{\text{preg}} + P_{\text{lact}}$$

[AN.NRC.50]

Table 8: Key variables used in mineral requirement calculations.

Variable	Description	Units
$Ca_{\mathbf{main}}$	Calcium maintenance requirement	g
$Ca_{\mathbf{growth}}$	Calcium growth requirement	g
Ca_{lact}	Calcium lactation requirement	g
Ca_{preg}	Calcium pregnancy requirement	g
$Ca_{\mathbf{req}}$	Total calcium requirement	g
DOP	Days of pregnancy	d
P_{growth}	Phosphorus growth requirement	g
P_{lact}	Phosphorus lactation requirement	g
P_{main}	Phosphorus maintenance requirement	g
P_{preg}	Phosphorus pregnancy requirement	g
P_{req}	Total phosphorus requirement	g

Dry Matter Intake (DMI) Estimation

Estimations are different for lactating cows and dry cows. For details about the variables included in each calculation, please refer to the DMI requirements key.

The sum of DMI of each feed is assumed to be less than the estimated DMI:

$$\sum_i Feed_i < DMIest$$

- ***Lactating Cows***

$$FCM = (0.4 * Milk) + (15 * Fat\_Milk * \frac{Milk}{100})$$

[AN.NRC.51]

$$DMIest = (0.372 * FCM + 0.0968 * BW^{0.75}) * (1 - exp - 0.192 * WOL + 3.67)$$

[AN.NRC.52]

- ***Dry Cows***

$$DMIest = (\frac{(1.97 - 0.75 * exp(0.16 * (DOP - 280)))}{100}) * BW$$

[AN.NRC.53]

- ***Heifers***

If the heifer is **less** than 365 days old:

$$DMIest = (BW^{0.75} * \frac{(0.2435 * NEadj - 0.0466 * NEadj^2 - 0.0869)}{NEadj}) - PREGadj$$

[AN.NRC.54]

If the heifer is **more** than 365 days old:

$$DMIest = (BW^{0.75} * \frac{(0.2435 * NEadj - 0.0466 * NEadj^2 - 0.1128)}{NEadj}) - PREGadj$$

[AN.NRC.55]

Table 9: Key variables used in DMI requirement calculations.

Variable	Description	Units
DMIest	Dry matter intake estimation	kilogram (kg)
DOP	Days of pregnancy	days
NEadj	the adjustment for the net energy (NE) concentration of the diet	if the NE concentration $<$ 1 mcal/kg DM, then = 0.95 if the concentration $>$ 1 mcal/kg DM, then is = to the NE concentration
PREGadj	the adjustment for pregnancy status	if DOP $>$ 210 1 + ((210 - DOP) * 0.0025)
WOL	Week of lactation	Units are the integer part of days $\frac{DIM}{7}$

NASEM: Calculating Nutrient Requirements of Dairy Cattle: Eighth Revised Edition Energy Requirements The maintenance requirement is calculated based on metabolic body weight (MBW, body weight$^{0.75}$) adjusted for the tissues associated with pregnancy. For details about the variables included in each calculation, please refer to the energy requirements key.

- ***Maintenance***

  Lactating and Dry Cows For Lactating cows, the body weight is adjusted both for the gravid uterine tissues and the uterine tissues for normal activity in confinement conditions, otherwise include adjustments for activity requirements under grazing conditions.

$$NEmaint = 0.10 * (MBW - GrUterW - UterW)^{0.75}$$

[AN.NSM.1]

$$GrUterW = (CBW * 1.825) * exp^{-(0.0243-(0.0000245*DOP))*(280-DOP)}$$

[AN.NSM.2]

$$UterW = ((CBW * 0.2288 - 0.204) * \exp^{-0.2*DyLact} + 0.204)$$

[AN.NSM.3]

$$CW = inputvariable,$$

Otherwise

$$CBW = MW * 0.06275$$

[AN.NSM.4]

<u>Heifers</u> For heifers, the maintenance energy requirement is only adjusted for the gravid uterine weight:

$$NEmain = 0.10 * (BW - GrUterW)^{0.75}$$

[AN.NSM.5]

- ***Activity*** - Activity requirement is proportional to body weight and daily walking distance. A grazing system and hilly topography will cost additional energy.

$$NEa1 = \frac{(Dt\_PastIn)}{DtDMIn} \text{ if } < 0.005\text{, then } 0$$

[AN.NSM.6]

Otherwise

$$NEa1 = 0.0075 * BW^{0.75} * (\frac{(600-12*DtPastSupplIn)}{600}) \text{ if } \geq 0.005\text{, then } 0$$

[AN.NSM.7]

Energy requirements for walking to and from the milking parlor are estimated as follows:

$$NEa2 = 0.00035 * \frac{(DistParlor)}{1000} * TripParlor * BW$$

[AN.NSM.8]

$$NEa3 = 0.0067 * \frac{(EnvTopoParlor)}{1000} * BW$$

[AN.NSM.9]

$$NEa = NEa1 + NEa2 + NEa3$$

[AN.NSM.10]

- ***Growth*** In National Academies of Sciences, Engineering, and Medicine (2021), body frame gain (fat + protein) corresponds to the true growth and it is part of the calculation which is further partitioned into body reserves or condition gain (or loss), and pregnancy-associated gain (considered a pregnancy requirement).

  The value shown below was calculated but was never utilized in any further calculations in the model's current version. Thus, it has been omitted from the model and remains solely as a reference to past versions of the model.

$$EBW = 0.85 * BW$$

[AN.NSM.11]

$$EBG = 0.85 * ADG$$

[AN.NSM.12]

$$FatADG = (0.067 + 0.375 * (\frac{BW}{MW})) * \frac{EBG}{ADG}$$

[AN.NSM.13]

$$ProtADG = (0.201 + 0.081 * (\frac{BW}{MW})) * \frac{EBG}{ADG}$$

[AN.NSM.14]

$$REFADG = 9.4 * FatADG + 5.55 * ProtADG$$

[AN.NSM.15]

$$MEFrameADG = \frac{REFADG}{0.4}$$

[AN.NSM.16]

$$NElFrameADG = \frac{REFADG}{0.61}$$

[AN.NSM.17]

- ***Pregnancy*** - Daily rates of wet tissue deposition (kg/d) are derived from equations **[AN.NSM.2]** and **[AN.NSM.3]** as defined above.

  During Gestation

$$GrUterWGain = (0.0243 - (0.0000245 * DayGest)) * GrUterW$$

[AN.NSM.18]

During Involution

$$MEpreg = (2 * 0.00159 * DOP - 0.0352) * \frac{CBW}{45} * 0.14, if DOP > 190.0$$

[AN.NSM.19]

Otherwise

$$NEpreg = MEpreg * 0.64$$

[AN.NSM.20]

- ***Lactation***

$$MilkEn = (0.0929 * Fat\_Milk + 0.0547 * Prot\_Milk + 0.0395 * Lact\_Milk)$$

[AN.NSM.21]

If true protein, (TP_Milk) is measured, its energy content is adjusted to account for the energy of Non-Protein-Nitrogen (NPN), then use the following equation.

$$MilkEn = (0.0929 * Fat\_Milk + 0.0585 * TP\_Milk + 0.0395 * Lact\_Milk)$$

[AN.NSM.22]

Protein Requirements are divided into 4 components: maintenance, growth, pregnancy, and lactation (all in MP). The last is defined as the sum of rumen undegraded protein (RUP + microbial protein (MICP)). Requirements for EEA are also included in National Academies of Sciences, Engineering, and Medicine (2021) but have not yet been implemented.

Each physiological function is quantified in terms of TP (true protein), instead of CP (crude protein).

To convert both net protein and net amino acid values to a metabolizable basis, net values have to be divided into the calculated efficiencies for each physiological function. Efficiencies can be either fixed or combined for lactating cows. For simplicity, requirements were included on a fixed basis. For details about the variables included in each calculation, please refer to the protein requirements key.

- ***Maintenance*** - The protein requirements for non-productive functions (the quantification of protein secretion and accretion), include: scurf + endogenous urinary loss + metabolic fecal protein.

  1. Scurf Protein

$$NPscurf = 0.20 * BW^{0.60} * 0.85$$

[AN.NSM.23]

$$NetAAscurf = NPscurf * \frac{AACorrScurf}{100}$$

[AN.NSM.24]

  2. Endogenous Protein

$$NPEndUrin = 53 * 6.25 * BW * 0.001$$

[AN.NSM.25]

$$NetAAEndUrin = 0.010 * 6.25 * BW * \frac{AACorrWhEmpBody}{100}$$

[AN.NSM.26]

  3. Metabolic Fecal Protein The daily loss of CP as metabolic fecal protein (MFP) is estimated using the following equation.

$$CPMFP = (11.62 + 0.134 * NDF) * DMI$$

[AN.NSM.27]

$$NPMFP = CPMFP * 0.73$$

[AN.NSM.28]

$$NPAAMFP = NPMFP * \frac{AACorrMFP}{100}$$

[AN.NSM.29]

- ***Growth*** There are two important concepts: frame growth and BW changes related to the mobilization of body reserves.

  The following are the protein requirements associated with growth associated with increased frame size:

  $$NPGrowth_{\text{Frame}} = FrameWG * 0.11 * 0.86$$

  [AN.NSM.30]

  If the change in BW is not frame growth but rather a change in body reserves, then the following equation is used because the protein content is assumed to be 8.0 percent protein.

  $$NPGrowth_{\text{Tissue}} = TissueWG * 0.08 * 0.86$$

  [AN.NSM.31]

  The following equation is used to translate the protein requirement into the requirement for AA:

  $$NetAAGrowth = NPGrowth * \frac{AACorrWEBW}{100}$$

  [AN.NSM.32]

- ***Pregnancy***

  $$NPGest = GainGrUter * 125$$

  [AN.NSM.33]

  $$NetAAGest = NPGest * \frac{AACorrWEBW}{100}$$

  [AN.NSM.34]

- ***Lactation***

  $$NetAAMilk = NPMilk * \frac{AAcalcMilk}{100}$$

[AN.NSM.35]

- ***Total Requirements*** -Total MP requirements are expressed in grams per day. This corresponds to the sum of all maintenance and other physiological functions.

Lactating Cows

$$RecomMPSupply1 = [\frac{(NetAAScurf + NetAAMFP + NetAAMilk + NetAAGrowth)}{TargetEffMP}] + \frac{NPGest}{0.33} + NPEndUrin$$

[AN.NSM.36]

$$RecomMPAASupply1 = \frac{NPScurf + NPMFP + NPMilk + NPGrowth}{TargetEffAA} + \frac{NetAAGest}{0.33} + NetAAEndUrin$$

[AN.NSM.37]

Late Gestation Heifers and Cows

$$RecomMPSupply2 = \frac{(NPScurf + NPMFP)}{TargetEffMP} + \frac{NPGest}{0.33} + \frac{NPGrowth}{0.4} + NPEndUrin$$

[AN.NSM.38]

$$RecomMPAASupply2 = \frac{(NetAAScurf + NetAAMFP)}{NetAAMFP} + \frac{NetAAGest}{0.33} + \frac{NetAAGrowth}{0.4} + NPEndUrin$$

[AN.NSM.39]

Table 10: Key variables used in protein requirement calculations.

Variable	Description	Units

Continued on next page

125 (coefficient)	Daily gain in mass of the gravid uterus	estimated constant 125 grams (g) of protein per kilogram (kg) of wet weight.
AAcalcMilk	Estimated AA composition of the true protein (TP) fraction involved in the estimation of AA Supply and recommendations	g of AA per 100 g of TP.
AACorrMFP	AA corrected for MFP	Fixed Value*
AACorrScurf	Each AA excreted as scurf	Fixed Value*
AACorrWEBW	Each AA excreted as assuming whole empty body weight	Fixed Value*
AACorrWEBW	Each AA excreted assuming whole empty body composition	Fixed Value*
BW	body weight	kg
CPMFP	Crude protein (CP) in MFP	grams per day (g/d)
DMI	Dry matter intake	kg/d
FrameWG	Daily increase in body weight due to increased frame size	kg/d
GainGrUter	Daily gain in mass of the gravid uterus	kg/d
NetAA-Milk	Net AA yield in milk	g/d
NetAAGest	AA composition of protein accretion associated with pregnancy	g/d
NetAAGrowth	Net AA requirement for body frame weight gain	g/d
NetAAMFP	Net AA requirement for metabolic fecal protein	g/d
NetAAEndUrin	Net AA requirement for endogenous urinary excretion	g/d
NetAAscurf	Net AA demand for scurf excretion	g/d
NDF	Neutral detergent fiber in the diet	% DM
NPGest	Net protein requirement for pregnancy	g/d
NPGrowth	Net protein requirement for body frame weight gain	g/d
NPMFP	Net protein requirement for metabolic fecal protein i	g/d; *assuming 73 % of TP in MFP*
NPMilk	Milk's TP yield	g/d
NPscurf	Net protein requirement for scurf	g/d
PEndUrin	Net protein requirement for endogenous urinary excretion	g/d
TargetEffMP and TargetEffAA	Proposed target efficiencies of converting metabolizable protein (MP) and essential amino acids (EAA) to export proteins and body gain	Fixed Value*

Note: AA = Amino Acids; MFP= Metabolic Fecal Protein; *Fixed values are establish by National Academies of

Sciences, Engineering, and Medicine (2021)

Mineral Requirements Only calcium and phosphorus requirements are included in the code at present. For details about the variables included in each calculation, please refer to the mineral requirements key.

- ***Maintenance***

  Calcium Equations

  $$Ca_{\text{main}} = 0.90 * DMI$$

  [AN.NSM.40]

  Phosphorus Equations

  $$P_{\text{main}} = 1.0 * DMI + 0.0006 * BW$$

  [AN.NSM.41]

- ***Growth***

  Calcium Equations

  $$Ca_{\text{growth}} = 9.83 * MW^{-0.22} * BW^{-0.22} * ADG$$

  [AN.NSM.42]

  Phosphorus Equations

  $$P_{\text{growth}} = (1.2 + 4.635 * MW^{0.22} * BW^{-0.22} + ADG)$$

  [AN.NSM.43]

- ***Pregnancy***

  Calcium Equations

  $$Ca_{\text{preg}} = 0.02456 * exp((0.05581 - 0.00007 * DOP) * DOP) - 0.02456 * exp((0.05581 - 0.00007 * (DOP - 1)) * (DOP - 1)) * (\frac{BW}{715}), \text{ if DOP} > 190; 0$$

  [AN.NSM.44]

  Phosphorus Equations

  $$P_{\text{preg}} = 0.02743 * exp((0.05527 - 0.000075 * DOP) * DOP) - 0.02743 * exp((0.05527 - 0.000075 * (DOP - 1)) * (DOP - 1)) * (\frac{BW}{715}), \text{ if DOP} > 190; 0$$

[AN.NSM.45]

The denominator, 715 kg, is the average cow weight in the study by House and Bell (1993); therefore, the pregnancy (gestation) requirement is scaled to that BW. For details see Page 107 in the Minerals chapter of National Academies of Sciences, Engineering, and Medicine (2021). IF DOP < 190; then it is assumed that Ca_Preg is equal to zero.

– ***Lactation***

Calcium Equations

$$Ca_{\text{lact}} = (0.295 + 0.239 * MilkTP) * Milk$$

[AN.NSM.46]

Phosphorus Equations

$$P_{\text{lact}} = Milk * (0.496 + 0.13 * TPMilk); Milk * 0.90$$

[AN.NSM.47]

– ***Total***

Calcium Equations

$$Ca_{\text{req}} = Ca_{\text{main}} + Ca_{\text{growth}} + Ca_{\text{preg}} + Ca_{\text{lact}}$$

[AN.NSM.48]

Phosphorus Equations

$$P_{\text{lact}} = P_{\text{main}} + P_{\text{growth}} + P_{\text{preg}} + P_{\text{lact}}$$

[AN.NSM.49]

Table 11: Key variables used in mineral requirement calculations.

Variable	**Description**	**Units**
ADG	average daily BW gain	grams/day (g/d)
ADG	average daily gain	g/d
BW	Body weight	kilograms (kg)
BW	Body weight	kg
Ca_Req	Total calcium requirement	g/d
Ca_growth	Calcium requirement for growth	g/d

Continued on next page

Variable	Description	Units
Ca_lact	Calcium requirement for lactation	g/d
Ca_main	Calcium requirement for maintenance	g/d
Ca_preg	Calcium pregnancy requirement	g/d
DMI	dry matter intake	kg/d
DOP	days of pregnancy	d
Milk	Milk yield	kg/d
MilkTP	Milk true protein	%
MW	mature body weight	kg
P_lact	Phosphorus requirement for lactation	g/d
P_Req	Total phosphorus requirement	g/d
P_growth	Phosphorus requirement for growth	g/d
P_main	Phosphorus requirement for maintenance	g/d
P_preg	Phosphorus requirement for pregnancy	g

Dry Matter Intake Estimation Dry matter intake estimation is different for lactating cows and dry cows. The sum of the dry matter intake of each feed is assumed to be less than the dry matter intake estimation For details about the variables included in each calculation, please refer to the DMI estimation key.

$$\sum_i Feed_i < DMIest$$

Lactation Cows

$$DMIest = ((3.7 + Parity * 5.7) + 0.305 * MilkE + 0.022 * BW + (-0.689 - 1.87 * Parity) * BCS) * (1 - (0.212 + Parity * 0.136 * exp(-0.053 * DIM)))$$

[AN.NSM.50]

Dry Cows and Growing Heifers

$$DMIest = (0.0226 * MatBW * (1 - exp(*1.47 * (\frac{BW}{MatBW})))) - (0.082 * (NDF - (23.1 + 56 * (\frac{BW}{MatBW}) - 30.6 * (\frac{BW}{MatBW})^2)))$$

[AN.NSM.51]

Table 12: Key variables used in DMI estimation calculations.

Variable	Description	Units
BCS	Body condition score	Scale of 1 to 5
BW	Body weight	kilograms (kg)
DIM	Days in milk	days (d)
DMIest	Dry matter intake estimation	kg/d
MatBW	Mature body weight	kg
MilkE	Milk energy	Mcal/d
NDF	Neutral detergent fiber in diet	%
Parity	adjustment factor	primiparous (0) or multiparous (1)

NASEM: Calculating Amino Acid Requirements The methods for estimating the amino acids requirements for lactating cows based on the National Academies of Sciences, Engineering, and Medicine (2021), into RuFaS are described below.

- Amino acids are divided into essential (EAA) and non-essential (NEAA).
- This section will focus on calculating the requirements for the essential amino acids. The EAA and NEAA are listed in the next table.
- Find the equations in RuFaS in amino_acid.py

Table 13: Essential and non-essential amino acids

Essential (EAA)	Non-essential (NEAA)
Arginine	Alanine
Histidine	Aspartic acid
Isoleucine	Asparagine
Leucine	Cisteine
Lysine	Glutamic acid
Methionine	Glutamine
Phenylalanine	Glycine
Theonine	Proline
Tryptophan	Serine
Valine	Tyrosine

EAA Calculations - The AA requirements for lactating cows are categorized into maintenance, growth, pregnancy, and lactation. Notice that the first variable from each equation comes from the outputs of the metabolizable protein equations that are already in the RuFaS model. The second variable corresponds to the composition of AA used or secreted for each metabolic state (maintenance, growth, pregnancy, and lactation) of the cow, we can find those values in Table 6-2 (page 79) of National Academies of Sciences, Engineering, and Medicine (2021). The subscript "i" in the equations represents each specific amino acid being calculated; for example, "TotalAArequirementsi_ii" refers to the total requirements for individual amino acids such as methionine, lysine, arginine, and others.

- ***Maintenance***

  1. Scurf Protein

  $$NetAAscurf_{\text{i}} = NPscurf * \frac{AACorrScurf_{\text{i}}}{100}$$

  2. Endogenous Urinary Protein

  $$NetAAEndUrine_{\text{i}} = 0.010 * 6.2.5 * BW * \frac{AACorrWholeEmptyBody_{\text{i}}}{100}$$

  3. Metabolic Fecal Protein

  $$NetAAMFP_{\text{i}} = NPMFP * \frac{AACorrMFP_{\text{i}}}{100}$$

- ***Growth***

  $$NetAAGrowth_{\text{i}} = NPGrowth * \frac{AACorrWholeEmptyBody_{\text{i}}}{100}$$

- ***Pregnancy***

  $$NetAAGest_{\text{i}} = NPGest * \frac{AACorrWholeEmptyBody_{\text{i}}}{100}$$

- ***Lactation***

  $$NetAAMilk_{\text{i}} = NPMilk * \frac{AAcalcMilk_{\text{i}}}{100}$$

- ***Total***

  Here are the equations for calculating the total AA_{i} requirements for lactating cows, late-gestation cows, and heifers.

  The variables needed to calculate these equations come from the outputs generated in the AA requirements equations for scurf, endogenous urine, metabolic fecal protein, growth, gestation, and milk production (see formulas above).

  The "$TargetEfficiencyAA_{\text{i}}$" variable, refers to the target efficiency required to convert the EAA to build proteins and support body gain. It is a fixed value, and it can be found in Table 6-4 (page 88) of National Academies of Sciences, Engineering, and Medicine (2021).

  1. Lactating Cows

  $$TotalAArequirements_{\text{i}} = (\frac{NetAAscurf_{\text{i}} + NetAAMFP_{\text{i}} + NetAAGrowth_{\text{i}} + NetAAMilk_{\text{i}}}{TargetEfficiencyAA_{\text{i}}}) + (\frac{NetAAGest_{\text{i}}}{TargetEfficiencyGest}) + NetAAEndUrine_{\text{i}}$$

2. Late Gestation Cows and Heifers

$$TotalAArequirements_i = (\frac{NetAAscurf_i + NetAAMFP_i}{TargetEfficiencyAA_i}) + (\frac{NetAAGrowth_i}{TargetEfficiencyGrowth}) + (\frac{NetAAGest_i}{TargetEfficiencyGest}) + NetAAEndUrine_i$$

Important Note

If there are any questions regarding calculations regarding calculations of maintenance AA requirements for Lactating Cows and/or Total AA requirement of Lactating or Late Gestation Cows and Heifers, please refer to Tables 6-2 and 6-4 in National Academies of Sciences, Engineering, and Medicine, 2021.

Table 14: Key variables used in calculation of AA.

Variable	Description	Units
NetAAscurfi	Net demand for AAi in scurf excretion	g/d
NPscurf	Net total protein requirement for scurf	g/d
AACorrScurfi[1]	For each amino acid, a fixed value for the proportion of the scurf in each AA.	-
NetAAEndUrinei	Net AAi requirement for endogenous urinary excretion g/d	
BW: Body weight	kg	
AACorrWholeEmptyBodyi[1]	For each amino acid, a fixed value for each AA within the tissues of the animal's whole empty body	g/100g TP
NetAAMFPi	Net AAi requirement for metabolic fecal protein	(MFP) g/d.
NPMFP	Net protein requirement for metabolic fecal protein	g/d. *assumes 73% of TP in MFP.*
AACorrMFPi[1]	Amino acids corrected for MFP. For each amino acid, a fixed value for each AA is excreted as metabolic fecal protein	g/100g TP
NetAAGrowthi	Net AAi requirement for body frame weight gain	g/d.
NPGrowth: Net protein requirement for body frame weight gain	g/d.	
AACorrWholeEmptyBodyi[1]	For each amino acid, a fixed value for each AA within the tissues of the animal's whole empty body	g/100g TP

Continued on next page

Variable	Description	Units
NetAAGesti	Amino acids composition of protein accretion associated with gestation	g/d.
NPGest	Net protein requirement for gestation	g/d.
AACorrWholeEmptyBodyi[1]	For each amino acid, a fixed value for each AA within the tissues of the animal's whole empty body	g/100g TP
NetAAMilki	Net amino acids yield in milk	g/d.
NPMilk	Net protein in milk. This refers to milk TP yield	g/d.
AAcalcMilki[1]	Estimated AA composition of the true protein (TP) fraction involved in the estimation of AA Supply and recommendations. Fixed values for each individual amino acid in milk	-
TotalAArequirementsi	Total AA requirement g/d.	
NetAAscurfi	Net demand for AAi in scurf excretion	g/d
NetAAMFPi	Net AAi requirement for metabolic fecal protein	g/d
NetAAGrowthi	Net AAi requirement for body frame weight gain	g/d
NetAAMilki	Net amino acids yield in milk	g/d
NetAAGesti	Amino acids composition of protein accretion associated with pregnancy	g/d
NetAAEndUrinei	Net AAi requirement for endogenous urinary excretion	g/d
TargetEfficiencyGest	Proposed target efficiency of using MP for gestation.	Value = 0.33
TargetEfficiencyAAi[1]	Proposed target efficiencies of essential amino acids (EAA) to export proteins and body gain	g/100g TP
TotalAArequirementsi	Total AA requirement	g/d
NetAAscurfi	Net demand for AAi in scurf excretion	g/d
NetAAMFPi	Net AAi requirement for metabolic fecal protein	g/d
NetAAGrowthi	Net AAi requirement for body frame weight gain	g/d
NetAAGesti	Amino acids composition of protein accretion associated with pregnancy	g/d
NetAAEndUrinei	Net AAi requirement for endogenous urinary excretion	g/d

Continued on next page

Variable	Description	Units
TargetEfficiencyGest	Proposed target efficiency of using MP for gestation.	Value = 0.33
TargetEfficiencyGrowth	Proposed target efficiency of using MP for growth.	Value = 0.40
TargetEfficiencyAAi[1]	Proposed target efficiencies of essential amino acids (EAA) to export proteins and body gain	-

[1] National Academies of Sciences, Engineering, and Medicine, 2021

2.c.iii Energy and Nutrition Supply

Nutrient Supply

Energy Supply - NRC 2001 Energy supply from the feed includes NEm, NEl, and NEg. NEm provides energy for maintenance and activity requirements, NEl provides energy for lactation and pregnancy, and NEg provides energy for growth. Feed actual energy contents are corrected from standard values provided by National Research Council (2001). Minerals do not provide any energy.

- ***Total Digestible Nutrients***

$$TDNconc = \frac{TotalTDN}{DMI} * 100$$

[.1]

$$DMI\_to\_maint = 1, if TotalTDN < 0.035 * BW^{0.75}, \frac{TotalTDN}{0.035 * SBW^{0.75}}$$

[AN.SUP.2.1]

Otherwise:

$$Discount = 1, if TDNconc < 60 \frac{(TDNconc - (0.18 * TDNconc - 10.3) * (DMI_{\mathrm{to,maint}} - 1))}{TDNconc}$$

[AN.SUP.3.1]

$$TDNact_{\mathrm{i}} = TDN_{\mathrm{i}} * Discount$$

[AN.SUP.4.1]

- ***Digestible Energy***

$$DEact_{\mathrm{i}} = DE_{\mathrm{i}} * Discount$$

[AN.SUP.4]

- ***Metabolizable Energy***

$$MEacti = 1.01 * DEacti - 0.45 + 0.0046 * 0.0046 * EEi - 3, if\ feed\ type\ is\ not\ fat\ and\ fat\ content\ is \geq 3\%$$

$$MEacti = 1.01 * DEacti - 0.45, if\ feed\ type\ is\ not\ fat\ and\ fat\ content\ is\ \leq 3\%$$

$$MEacti = 1.01 * DEi,\ if\ feed\ type\ is\ fat$$

[AN.SUP.5.1]

- ***Net Energy***

1. Maintenance

$$NEmacti = 1.37*MEacti - 0.138*MEacti^2 + 0.0105*MEacti^3 - 1.12,\ if\ feed\ type\ is\ not\ fat$$

$$NEmacti = MEacti * 0.8,\ if\ feed\ type\ is\ fat$$

[AN.SUP.7.1]

2. Lactation

$$NElacti = 0.703 * MEacti - 0.19 + 0.097 * MEacti + 0.1997 * EEi - 3,\ if\ feed\ type\ is\ not\ fat\ and\ fat\ content\ is \geq 3\%$$

$$NElacti = 0.703 * MEacti - 0.19,\ if\ feed\ type\ is\ not\ fat\ and\ fat\ content\ is \leq 3\%$$

$$NElacti = 0DEacti * 0.8,\ if\ feed\ type\ is\ fat$$

[AN.SUP.6.1]

3. Activity

$$NEgacti = 1.42*MEacti - 0.174*MEacti^2 + 0.0122*MEacti^3 - 1.65,\ if\ feed\ type\ is\ not\ fat$$

$$NEgacti = MEacti * 0.55, \ if \ feed \ type \ is \ fat$$

[AN.SUP.8.1]

Table 15: Key variables used in calculation of net energy in NRC 2001.

Variable	Description	Units
NEgacti	Actual net energy for growth of feed i	Mcal/kg
NEmacti	Actual net energy for maintenance of feed i	Mcal/kg
NElacti	Actual net energy for lactation of feed i	Mcal/kg
MEacti	Actual metabolizable energy of feed i	Mcal/kg
EEi	Feed fat (ether extract) content	%
DEacti	Actual digestible energy of feed i	Mcal/kg
DEi	Standard DE feed i,	Mcal/kg
TDNacti	Actual TDN content of feed i	%
TDNi	Standard TDN content of feed i	%
Discount	TDN digestibility decrease caused by DMI and TDNconc	
DMI_to_maint	The amount of intake needed to meet the maintenance requirement	dimensionless
TDNconc	TDN concentration	%
TotalTDN[1]	Dietary TDN content	

[1] National Research Council, 2001

Energy Supply - NASEM 2021 Energy supply in the updated 2021 Nutrient Requirement model National Academies of Sciences, Engineering, and Medicine (2021) is estimated based only on the NEl and follows a simplified estimation method when compared to the 2001 model National Research Council (2001). First, the digestible energy of each feed is estimated based on the nutrient content and the estimated digestibility of that nutrient.

- ***Digestible Energy (DE) of a Feed***

$$DE_i = 0.042 * NDF_i * dNDF_i + 0.0423 * Starch_i * dStarch_i + 0.0940 * FA_i * dFA_i + 0.0565 * (RDP_i + RUP_i * dRUP_i - NPN_{supp,i}) + 0.0089 * NPN_{supp,i} + 0.040 * ROM_i * dROM_i - 0.318$$

[AN.SUP.1.2]

Parameter definitions and units are provided in Table 1. The nutrient compositions of each feed $(NDF_i, Starch_i, EE_i, RUP_i)$ are provided in the feed library. The digestibilities of each feed are estimated to account for the effects of intake and diet composition:

– ***NDF Digestibility (dNDF)*** The digestibility of NDF is adjusted from the base NDF digestibility of the feed to account for the effect of total DMI and starch intake.

$$dNDF_{\text{i}} = [(dNDF_{\text{base},i} - 0.0059 * (Starch_{\text{Diet}} - 0.26) - 1.1 * (\frac{DMI}{BodyWeight} - 0.035)$$

And the $dNDF_{\text{base},i}$ is calculated based on the lignin and NDF content of the feed.

$$dNDF_{\text{base},i} = [0.75 * (NDF_{\text{i}} - Lignin_{\text{i}}) * (1 - (\frac{Lignin_{\text{i}}}{NDF_{\text{i}}})^{0.667})]/NDF_{\text{i}}$$

[AN.SUP.2.2]

– ***Starch Digestibility (dStarch)*** Similarly, the digestibility of starch is adjusted for the level of intake from a base digestibility:

$$dStarch_{\text{i}} = dStarch_{\text{base},i} - 1.0 * (\frac{DMI}{BodyWeight} - 0.035)$$

[AN.SUP.3.2]

And the base starch digestibility, $dStarch_{\text{base},i}$, is considered to be an innate property of the feed and is provided in the feed library.

– ***Fatty Acid Digestibility (dFA)*** Fatty acids are assumed to have a digestibility of 0.73 ($dFA_{\text{i}} = 0.73$) unless a digesibility specific to the fat supplement is provided by the manufacturer.

– ***Protein Digestibility (RUP and dRUP)*** The rumen degradable protein (RDP) content of a feed is calculated as the proportion of crude protein (CP) that is not considered to be rumen undegradable (RUP). Thus:

$$RDP_{\text{i}} = \frac{1 - RUP_{CP,i}}{CP_{\text{i}}}$$

$$RUP_{\text{i}} = \frac{RUP_{CP,i}}{CP_{\text{i}}}$$

[AN.SUP.4.2]

– ***Residual Organic Matter (ROM)*** The Residual Organic Matter (ROM) fraction of a feed is calculated by difference as the following:

$$ROM_{\text{i}} = 1 - \frac{FA_{\text{i}}}{1.06} - Ash_{\text{i}} - NDF_{\text{i}} - Starch_{\text{i}} - (CP_{\text{i}} - 0.64 * NPN_{supp,i})$$

The digestibility of ROM is assumed to be 0.96 ($dROM_{\text{i}} = 0.96$) and $NPN_{supp,i}$ is the supplemented non-protein nitrogen fraction as a proportion of the total CP content.

[AN.SUP.5.2]

- ***Digestible Energy of the diet***

$$DE_{\text{Diet}} = \sum_{i=1}^{n} DE_i * PropFeed_i$$

[AN.SUP.6.2]

- ***Metabolizable Energy*** The dietary ME is calculated by subtracting the energy found in gaseous losses (i.e. enteric methane) and the energy lost in urine from the total diet digestible energy.

$$ME_{\text{D}iet} = DE_{\text{D}iet} - GasEnergy - UrineEnergy$$

[AN.SUP.7.2]

  – ***Gaseous Energy***

$$GasEnergy = 13.28 * EntericMethane$$

[AN.SUP.8.2]

  – ***Urine Energy***

$$UrineEnergy = 0.0146 * UrinaryNitrogen$$

[AN.SUP.9.2]

where $GasEnergy$ and $UrineEnergy$ are in units of Mcal/d, $EntericMethane$ is in units of kg/d, and $UrinaryNitrogen$ is in units of g/d.

- ***Net Energy (NE)*** Net energy is simply estimated as metabolizable energy used at an average efficiency of 0.66

$$NEl = 0.66 * ME$$

[AN.SUP.10.2]

Table 16: Key variables used in calculation of net energy supply from NASEM 2021.

Variable	Description	Units
NElact	Actual net energy supply of the diet	Mcal

Continued on next page

Variable	Description	Units
ME	Actual metabolizable energy supply of the diet	Mcal
DE_i	Digestible energy content of feed i	Mcal/kg
DE_{Diet}	Digestible energy content of the diet	Mcal
$PropFeed_i$	Proportion of diet DM that is feed i	proportion
NDF_i	Neutral Detergent Fiber content of feed i	%
$dNDF_i$	Neutral Detergent Fiber digestibility of feed i	proportion
$dNDF_{base,i}$	The baseline Neutral Detergent Fiber digestibility of feed i	proportion
$Lignin_i$	Lignin content of feed i	%
$Starch_i$	Starch content of feed i	%
$dStarch_i$	Starch digestibility of feed i	proportion
$Starch_{Diet}$	Starch content of the diet	%
FA_i	Fatty Acid content of feed i	%
dFA_i	Fatty Acid digestibility of feed i	proportion
RDP_i	Rumen Degradable Protein content of feed i	%
$RDP_{CP,i}$	Percent of Crude Protein that is Rumen Degradable i	%
RUP_i	Rumen Undegradable Protein content of feed i	%
$RUP_{CP,i}$	Percent of Crude Protein that is Rumen Undegradable i	%
$dRUP_i$	Rumen Undegradable Protein digestibility of feed i	proportion
$NPN_{supp,i}$	Supplemental Non-Protein Nitrogen content	%
ROM_i	Residual Organic Matter content of feed i	%
$dROM_i$	Residual Organic Matter digestibility of feed i	proportion

[1] National Academies of Sciences, Engineering, and Medicine, 2021

Protein Supply - Protein supply from the feed includes 3 parts: digestible RUP, digestible MCP, which is produced through RDP, and endogenous protein supply from sloughed cells and enzyme secretion into the gut lumen. MCP production requires a nitrogen source and an energy source, so MCP is either nitrogen-limited (when energy is sufficient) or energy-limited (when nitrogen is sufficient). Minerals do not provide any protein.

$$MP_{\text{supply}} = MPbact + RUP_{\text{diet}} + EndP$$

[AN.SUP.14]

- ***Microbial Protein Supply***

$$Kpi = 2.904 + 1.375 * \frac{DMI}{BW} * 100 - 0.02 * PercentConc,\ if\ feed\ is\ concentrate$$

$$Kpi = 3.362 + 0.479 * \frac{DMI}{BW} * 100 - 0.17 * NDFi - 0.007 * PercentConc,\ if\ feed\ is\ forage$$

$$Kpi = 3.054 + 0.614 * \frac{DMI}{BW} * 100,\ if\ feed\ is\ wet\ forage$$

[AN.SUP.9]

$$RDPi = \frac{Kdi}{Kdi+Kpi} * \frac{NBi}{100} * CPi + \frac{NAi}{100} * CPi,\ if\ Kpi + Kdi > 0$$

$$RDPi = 0,\ if\ Kpi + Kdi \leq 0$$

[AN.SUP.10]

$$MPbact = 0.64 * min(1000 * 0.13 * TDNact_{\text{diet}}, 1000 * 0.85 * RDP_{\text{diet}})$$

[AN.SUP.11]

- ***Rumen Undegradable Protein Supply***

$$RUP_{\text{i}} = CP_{\text{i}} - RDP_{\text{i}}$$

[AN.SUP.12]

$$RUP_{\text{i}} = \sum_i Feed_i * RUP_{\text{i}} - dRUP_{\text{i}}$$

[AN.SUP.13]

- ***Endogenous Protein Supply***

$$EndP = 0.4 * 11.8DMI$$

Table 17: Key variables used in calculation of AA.

Variable	Description	Units
EndP	the endogenous protein supply in	g/d
RUPdiet	Dietary digestible RUP,	g
feedi	Dry matter intake of feed i, g	
dRUPi	RUP digestibility of feed i,	% of RUP
RUPi	Rumen undegradable protein of feed i,	% of DM
MPbact	Metabolizable bacterial protein production,	g
TDNactdiet	Dietary actual TDN,	kg
RDPdiet	Dietary RDP,	kg
RDPi	Rumen degradable protein of feed i,	% of DM
Kdi	Protein degradation rate of feed i,	%/h
NAi	Fraction A of protein of feed i,	% of CP
NBi	Fraction B of protein of feed i,	% of CP
Kpi	Protein passage rate of feed i,	%/h
PercentConc	Dietary concentrate percentage,	% of DM
NDFi	Neutral detergent fiber content of feed i,	%
MPsupply	Total metabolizable protein supply,	g/d
MPbact	Metabolizable bacterial protein production,	g/d
RUPdiet	Dietary digestible RUP,	g/d

Mineral Supply We will use calcium as an example, but phosphorus calculations will be similar.

$$Ca_{supply} = \sum_i Feed_i * Ca_i * \frac{dCa_i}{100}$$

[AN.SUP.15]

Table 18: Key variables used in calculation of AA.

Variable	Description
dCai	Ca digestibility of feed i

Continued on next page

Variable	Description
dCa	30% for forage, 60% for concentrate, 95% for mineral
dP	(P digestibility) = 64% for forage, 70% for concentrate, 80% for mineral

AA Nutrient Supply This section is divided into several subsections including Rumen Degraded and Undegraded Protein, Dry Matter Intake, Dietary Nutrient Intake, Dietary Nutrient Concentration, Digested Nutrients, Digested Rumen Undegradable Protein (RUP), Microbial Nitrogen Growth, Ruminal Nitrogen Outflow, and lastly Amino Acid Supply.

Energy Supply - Energy supply from the feed includes NEm, NEl, and NEg. NEm provides energy for maintenance and activity requirements, NEl provides energy for lactation and pregnancy, and NEg provides energy for growth. Feed actual energy contents are corrected from standard values provided by National Research Council (2001). Minerals do not provide any energy.

Total Digestible Nutrients

$$TDNconc = \frac{TotalTDN}{DMI} * 100$$

[AN.SUP.16]

$$DMI\_to\_maint = 1, \text{ If TotalTDN} < \ 0.035 * BW^{0.75}, \frac{TotalTDN}{0.035 * SBW^{0.75}},$$

[AN.SUP.17]

Otherwise

$$Discount = 1, \text{ IF TDNconc} < \ 60 \frac{(TDNconc-(0.18*TDNconc-10.3*(DMI_{tomaint}-1))}{TDNconc}$$

[AN.SUP.18]

Otherwise

$$TDNact_i = TDN_i * Discount$$

[AN.SUP.19]

Digestible Energy

$$DEact_i = DE_i * Discount$$

[AN.SUP.20]

Metabolizable Energy

$$MEacti = 1.01 * DEacti - 0.45 + 0.0046 * EEi - 3, if\ feed\ type\ is\ not\ fat\ and\ fat\ content\ is\ \geq 3\%$$
$$MEacti = 1.01 * DEacti - 0.45, if\ feed\ type\ is\ not\ fat\ and\ fat\ content\ is\ \leq 3\%$$
$$MEacti = DEacti, if\ feed\ type\ is\ fat$$

[AN.SUP.21]

Net Energy

Maintenance

$$NEmacti = 1.37 * MEacti - 0.138 * MEacti^2 + 0.0105 * MEacti^3 - 1.12, if\ feed\ type\ is\ not\ fat$$
$$NEmacti = MEacti * 0.8, if\ feed\ type\ is\ fat$$

[AN.SUP.22]

Lactation

$$NElacti = 0.703*MEacti-0.19+0.097*MEacti+0.1997*EEi-3, if\ feed\ type\ is\ not\ fat\ and\ fat\ content\ is\ \geq 3\%$$
$$NElacti = 0.703 * MEacti - 0.19, if\ feed\ type\ is\ not\ fat\ and\ fat\ content\ is\ \leq 3\%$$
$$NElacti = DEacti * 0.8, if\ feed\ type\ is\ fat$$

[AN.SUP.23]

Growth

$$NEgacti = 1.42 * MEacti - 0.174 * MEacti^2 + 0.0122 * MEacti^3 - 1.65, if\ feed\ type\ is\ not\ fat$$
$$NEgacti = MEacti * 0.55, if\ feed\ type\ is\ fat$$

[AN.SUP.24]

Protein Supply - Protein supply from the feed includes 3 parts: digestible RUP, digestible MCP, which is produced through RDP, and endogenous protein supply from sloughed cells and enzyme secretion into the gut lumen. MCP production requires a nitrogen source and an energy source, so MCP is either nitrogen-limited (when energy is sufficient) or energy-limited (when nitrogen is sufficient). Minerals do not provide any protein.

$$MP_{supply} = MPbact + RUP_{diet} + EndP$$

[AN.SUP.25]

Microbial Protein Supply

$$Kpi = 2.904 + 1.375 * \frac{DMI}{BW} * 100 - 0.02 * PercentConc \text{ if feeding concentrate}$$
$$Kpi = 3.362 + 0.479 * \frac{DMI}{BW} * 100 - 0.017 * NDFi - 0.007 * PercentConc \text{ if feeding forage}$$
$$Kpi = 3.054 + 0.614 * \frac{DMI}{BW} * 100 \text{ if feeding wet forage}$$

[AN.SUP.26]

$$RDPi = \frac{Kdi}{Kdi+Kpi} * (\frac{NBi}{100}) * Cpi + (\frac{NAi}{100}) * CPi, if Kpi + Kdi > 0,\ RDPi = 0, if Kpi + Kdi \leq 0,$$

[AN.SUP.27]

$$MPbact = 0.64 * min(1000 * 0.13 * TDNact_{diet}, 1000 * 0.85 * RDP_{diet})$$

[AN.SUP.28]

Rumen Undegradable Protein Supply

$$RUP_i = CP_i - RDP_i$$

[AN.SUP.29]

$$RUP_{diet} == \sum_i feed_i * RUP_i * dRUP_i$$

[AN.SUP.30]

Endogenous Protein Supply

$$EndP = 0.4 * 11.8DMI$$

[AN.SUP.31]

Mineral Supply

$$Ca_{supply} = \sum_i feed_i * Ca_i * \frac{dCa_i}{100}$$

[AN.SUP.32]

2.c.iv Automated Ration Formulation

Introduction Automated ration formulation in RuFaS optimizes a ration for a single objective (least cost), utilizing sequential least squares quadratic programming, as described in Li, Rosa, et al., 2022. This process is carried out on an individual pen level, and uses the mean animal requirements and body weight during formulation. The ration formulation process includes information not only from the Animal Module's inputs (Table RF1), but the primary Feed input file (Table RF2), and Feed ingredient inclusion rates defined in the Feed Storage module's input files

Required User Inputs

Table 19: Key inputs found in the animal input file.

Variable	**Definition**	**Default**	**Min**	**Max**
ration.user_input	User decision to use the user-defined ration formulation or the automated ration formulation methods (true or false, respectively).	TRUE	-	-
ration.formulation_interval	Ration formulation interval (days).	30	1	-
ration.phosphorus_requirement_buffer	Increase in phosphorus nutrient requirement in calculation of animal requirements (percentage).	75	0	-

Table 20: Key inputs found in the feed input file.

Section	**Variable**	**Definition**	**Default**	**Min**	**Max**
purchased_feeds	calf_feeds or growing_feeds or close_up_feeds or lac_cow_feeds	Each section is comprised of a list of feeds (RuFaS Feed IDs).	-	-	-
	purchased_feed	RuFaS Feed ID	-	-	-
	purchased_feed_cost	Price per feed (dollars/kg DM).	-	-	-
User_defined_-ration_percent-ages	calves/growing/close_up/lac_-cow.ration_percentage	For each feed ID, amount of a ration that is a given feed (percentage).	-	0	100
	tolerance	Allowable +/- percentage variance in each of the defined ration inclusion percentage values (fraction).	0.1	0	1
	milk_reduction_maximum	Allowable amount of milk reduction (kg) when dietary nutrient supply cannot meet animal requirements.	15	0	50

Relevant Outputs

- Average nutrient requirements, on a per animal basis, for each pen.
  - *Class and function*: AnimalModuleReporter.report_ration_interval_data
  - *Output variable*: avg_rqmts_pen_pen.id_pen.animal_combination.name
- Each formulation interval's mean animal ration, on a per animal basis, for each pen.
  - *Class and function*: AnimalModuleReporter.report_ration_interval_data
  - *Output variable*: ration_per_animal_for_pen_pen.id_pen.animal_combination.name
- Nutrient and energetic composition of ration, on a per animal basis, for each pen.
  - *Class and function*: AnimalModuleReporter.report_ration_interval_data
  - *Output variable*: ration_nutrient_amount_pen_pen.id_pen.animal_combination.name
- The total amount of Feeds, per pen.
  - *Class and function*: AnimalModuleReporter.report_daily_ration
  - *Output variable*: ration_daily_feed_totals_for_pen_pen.id_pen.animal_combination.name
- In cases of ration formulation failure, a summary of the constraints that failed, and the attempted ration and its supply will be supplied in a report.
  - *Class and function*: RationOptimizer.handle_failed_constraints
  - *Output variable*: failed_constraint_summary_for_pen_pen_id

Methodology Ration formulation is carried out at the pen level on a user-defined formulation interval (Table RF1). The mean requirements for a given pen are calculated prior to first formulation, as described in [REF IV. Animal Nutrition and Feeding, A. Calculations of Animal Energy and Nutrient Requirement]. This process then proceeds to utilize a series of constraints methods (Table RF3) that compares the animal requirements against the supply of the optimizer's ration attempt, as described in [REF IV. Animal Nutrition and Feeding, B. Calculations of Animal Energy and Nutrient Supply], all while abiding feed ingredient-specific inclusion rates, as defined in [REFERENCE TO NRC/NASEM LIBRARY].
If a ration cannot be formulated that meets the constraints given the pen-level requirements and limits to inclusion rates, then one of the following things will happen.

For lactating cow pens

- Record failed constraints and formulation attempt to OutputManager.
- Milk production will be reduced.
  - If average milk production for the pen drops below the values defined in [REFERENCE TO 5. Milk Production and Reduction], then the simulation halts with an error.
- Recalculation of the nutritional requirements for all animals in the pen.
- Ration formulation reattempt (repeat).

For growing and close-up pens

- Record failed constraints and formulation attempt to OutputManager.
- IF a successful ration from a prior formulation is available, said ration will be used.
- IF no previously successful ration is available, the simulation will stop.

Note: A graphical overview of this process can be found in Figure 4 in II Animal Management, C. Milk Production

The generalized equation for sequential quadratic programming is as follows (Li, Rosa, et al., 2022) and gi(x) is constraint i, which must be twice continuously differentiable with respect to all xj in x.").

Table 21: Key inputs found in the feed input file specific to user defined ration methodology. All are found in the section user_defined_ration_percentages

Nutrient or Energy	Min	Max	Animal Groups
Dry matter intake	80% of requirement	120% of requirement	All except calves
Metabolizable protein	100% of requirement	150% of requirement	All except calves
Net energy: lactation	100% of requirement		Lactating pen only
Net energy: maintenance and activity	100% of requirement		All except calves
Net energy: growth	100% of requirement		All except calves
Total net energy	100% of requirement		Lactating pen only
Calcium	100% of requirement		All except calves
Phosphorus	100% of requirement		All except calves
Fat		7% of ration	All except calves
NDF	25% of ration	45% of ration	All except calves
Forage NDF	15% of ration		All except calves

2.c.v User-defined ration formulation

Introduction The user-defined ration methodology follows the same process as described in the Automated ration formulation methodology, with the addition of the inputs found in this section, as described in the Methodology below.

Required User Inputs

Table 22: Key inputs found in the feed input file specific to user defined ration methodology. All are found in the section user_defined_ration_-percentages.

Variable	Definition	Default	Min	Max
calves or growing or close_up or lac_-cow.ration_percent-age	For each feed ID, amount of a ration that is a given feed (percent-age).		0	100
tolerance	Allowable +/- percentage variance in each of the defined ration inclusion percentage values (fraction).	0.1	0	1
milk_reduction_max-imum	Allowable amount of milk reduction (kg) when dietary nutrient supply cannot meet animal require-ments.	15	0	50

Methodology The key difference in the methodology for the user-defined ration methodology is that the feed inclusion rates are bounded by the user_defined_ration_percentage values defined for each feed item (Table RF4). Those percentage values become the minimum and maximum inclusion value for each feed, with +/- the tolerance value (Table RF4) as a fraction of the percentage.
The ration formulation will proceed as described in the Automated Methodology above, but when formulation fails, different outcomes will occur.
For lactating cow pens

- Record failed constraints and formulation attempt to OutputManager.
- Milk production will be reduced.
  - IF average milk production for the pen drops below the values defined in [REFERENCE TO 5. Milk Production and Reduction], then the simulation halts with an error.
  - IF if the milk_reduction_maximum (Table RF4) is exceeded, then the simulation proceeds by using the user-defined ration percentages exactly.
- Recalculation of the nutritional requirements for all animals in the pen.
- Ration formulation reattempt (repeat).

For growing and close-up pens

- Record failed constraints and formulation attempt to OutputManager.
- Recording of which constraints were unable to be met.
- The simulation proceeds by using the user-defined ration percentages exactly.

2.d Herd Size Management

If the number of close-up heifers is more than the herd needs, heifers in the Heifer III group are sold. If the number of close-up is less than the herd needs, a replacement is bought from the market. This is evaluated daily by comparing the number of (cows + Heifer III) to the herd num specified by the user. Details of this method are described in the Culling section of Animal-level processes and management.

Herd Size Management: Heifer III Sales

If the number of the heifers is more than the herd needs, heifers in the Heifer III group are sold. If the number of Heifer III is less than the herd needs, a replacement is bought from the market. This is evaluated daily by comparing the number of (cows + Heifer III) to the herd_num specified by the user.

User Inputs and Constants

- Herd_num: the target number of cows (dry + lactating) to maintain
- SELLING_THRESHOLD = 1.03 – hard-coded value (animal constant) used to determine threshold above herd_num at which heiferIII's should be sold.
- BUYING_THRESHOLD = 1.01 – hard-coded value used to determine threshold below herd_num at which heiferIII's should be bought

Parameters and Equations

- HerdManager._check_if_heifers_need_to_be_sold(). Evaluates if the current number of Heifer III and cows exceeds a specified threshold (defined as 3% of the herd statistics' target herd size). If the threshold is surpassed, Heifer IIIs are removed from the herd until herd size falls within acceptable range.
  - **Called by** HerdManager.daily_routines()
  - Returns a list of animals removed

  - Logic - While the number of (Heifer IIIs + cows) is greater than herd_num * selling_threshold:
    - The last heifer III is removed and is added to animals_removed
    - Information about that heifer III is added to sold_heiferIIIs_info
    - The number of heifer III's is decremented by 1
    - The number of sold_heiferIII_oversupply_num is incremented by 1
  - This cycle iterates until the number of (Heifer IIIs + cows) is equal to herd_num*selling_-threshold

- HerdManager._check_if_replacement_heifers_needed() Determines whether additional Heifer IIIs need to be added to the herd based on the current herd size, purchase thresholds, and the availability of heifers in the replacement market.
  - **Called by** HerdManager.daily_routines()
  - Returns a list of animals purchased
  - While the number of (Heifer IIIs + cows + bought_heifer_num) is less than herd_num * buying_threshold:
    - The first animal in the replacement_market enters the herd, assigned a phosphorus requirement and net merit value, and added to the animals_added list
    - Number of bought_heifer_num is incremented by 1
  - This cycle iterates until the number of (Heifer IIIs + cows + bought_heifer_num) is equal to herd_num*buying_threshold

Reproductive Culls: Heifer II Sales

If the heifer does not have a successful conception or pregnancy and age (days born) exceeds heifer repro cull time, this heifer is culled

User Inputs and Constants

- Heifer_repro_cull_time: days old when a heifer would be culled for failure to become pregnant (Default: 500 d)

Parameters and Equations

- Animal._evaluate_heiferII_for_culling()
  - **Called by** Animal.daily_routines() → Animal.animal_life_stage_update()
  - This method determines whether a heifer should be culled based on its pregnancy status and age, returning *True* if the heifer is not pregnant and has surpassed the specified culling age.

Calf Sales

All live male calves are sold immediately after birth. Female calves are sold/kept according to a roll of the dice evaluated against the user-defined rate of keeping female calves.

User Inputs and Constants

- Keep_female_calf_rate: the percentage of female calves kept and raised on-farm (Default: 1)

Parameters and Equations

- Animal._initialize_newborn_calf() to self.sold
  - **Called by** HerdManager._create_newborn_calf() in _perform_daily_routines_for_animals()
    - Creates a newborn calf instance

  - Contains logic that determines whether a calf should be sold, returning True if the calf is a male or if a random draw generates a number higher than the keep_female_calf_rate
  - If assigned to be sold, sold_at_day = simulation_day

Calf Losses (Deaths) A portion of calves are born dead according to the user-defined death rate.

User Inputs and Constants

- Still_birth_rate: rate of stillbirths (percent, 0-1) (Default: 0.065)

Parameters and Equations

- Animal._initialize_newborn_calf()
  - **Called by** HerdManager._create_newborn_calf() in _perform_daily_routines_for_animals()
    - Creates a newborn calf instance
  - If a random draw is less than the still_birth_rate, then self.sold = True and STILL_BIRTH is added to self.events
  - If assigned to be sold, sold_at_day = simulation_day

3 Animal-level Processes and Management

The methods in the RuFaS model that simulate individual animals' life events and the effects of herd management on individual animals are referred to as the Animal Life Cycle Sub-module. This part of the model is driven by a Monte Carlo stochastic process that simulates the growth, production, reproduction, and culling of individual dairy cattle animals and shares aggregated outcomes with the Herd Management Sub-Module to manage herd dynamics on a daily basis.
The life cycle module represents the variation in animal performance through the use of Monte Carlo Simulation methods (Reuven and Kroese, 2017). Monte Carlo simulation relies on random draws from probability distributions rather than assigning fixed values. In this model, two main strategies are used:

- **Monte Carlo Simulation (Event-Based):** Comparing a random draw from $U(0,1)\rho Uniform$ to the probability of an event occurring to simulate if that event occurs or did not (RandomUni in this document representing a random draw from $U(0,1)\rho Uniform$.
- **Random Population Simulation:** Selecting a random draw from a known distribution of animal attributes and assigning that value to the instantiation of an individual animal. (Random in this document representing a random draw from the following distribution) Probability distributions used are:
  - **Normal Gaussian:** $N(\mu, \sigma)$ where μ is the distribution mean, and σ is the standard deviation.
  - **Empirical:** $F(x) = \frac{1}{n}\sum_{i=1}^{n} 1_{\{x_i \leq x\}}$: where x is the observations from the sample and an indicator function equal to if and otherwise.

Flow of information For each animal object, the daily updates are called according to the order described in the n a RuFaS Farm. This is:

DigestiveSystem → NutrientInputs → MilkProduction → BodyweightChange → Reproduction → LifeStage Change → Parturition

The processes that determine significant events in each animal's life cycle occur in the

1. Reproduction,
2. LifeStage Change, and

3. Parturition updates.

Within the Reproduction update methods, there are separate methods to simulate breeding management for heifers and cows. All of the Reproduction update methods use an event-based Monte Carlo model to stochastically simulate the probability that each animal will conceive, the date they conceive, and the success of the pregnancy. In addition to recording success and failures to conceive, the Reproduction update records hormone deliveries and number and success of pregnancy checks. All methods are based on protocols described by the CDCB and are described in more detail in the Reproduction Section.

Each animal within the RuFaS herd will progress through the 5 main life stages from calf, to Heifer I, to Heifer II, to Heifer III, to Cow followed by a cull stage as illustrated in the figure below:

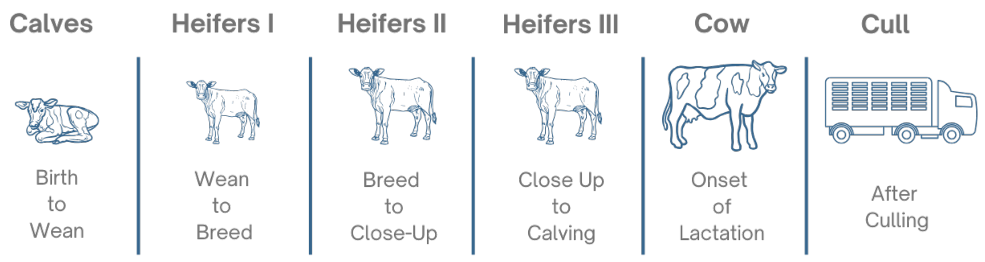


Figure 1: There are 5 life stages from calf to cow and a cull stage as outlined in this figure.

Progression Between Life Stages: Calves progress to Heifer I when their age reaches the user-defined input for weaning day. Heifer I animals progress to Heifer II animals when their age reaches the user-defined input for breeding start day. Heifer II animals that successfully conceive progress to the Heifer III stage when they are close to calving. Specifically, Heifers II become Heifers III when the number of days left in their pregnancy is equal to the user-defined input for the prefresh days. When a Heifer III animal calves for the first time, she becomes a Cow and will remain a cow for the rest of her life on the RuFaS farm.

When a Heifer III calves for the first time and becomes a Cow, then that animal will cycle through lactating and non-lactating phases until they are removed from the herd due to a failure to conceive or a user-provided probability for culling within each parity. The methods that determine when a cow will leave the herd are described in detail in the Herd Exits section.

Parturition and New Calves: When a Heifer III or Cow animal reaches the end of their pregnancy and starts lactating, a new calf animal object is created and assigned a birthweight according to a user-informed random distribution. All male calves are recorded and removed from the farm on day 1 of their life. Female calves remain on the farm determined by an event-based Monte Carlo method informed by the user-defined percent of female calves that are kept.

3.a RuFaS Bodyweight and Growth

Introduction An animal's bodyweight is initialized at birth according to a user defined normal distribution. Depending on her life stage, her bodyweight is then updated daily by adding their daily growth (commonly referred to as average daily gain), conceptus weight change, and tissue accretion and depletion associated with tissue mobilization to support lactation. The changes associated with each life stage are:

- Calves increase bodyweight by calf specific estimate for daily growth from birth to weaning.
- Non-pregnant heifers increase bodyweight by non-pregnant heifer specific estimate for daily growth

from weaning until conception.

- Pregnant heifers increase their bodyweight by the pregnant heifer specific estimate for daily growth and an estimate for conceptus growth.
- Cows in parity 1 and 2 increase their bodyweight according to a parity and pregnancy status specific estimate for daily growth, an estimate for conceptus growth, and an estimate for body tissue changes that occur in lactation.
- Cows in parities 3 and above change their bodyweight according to estimates for conceptus growth and body tissue changes that occur in lactation.

These methods are further described by Li, Reed, et al. (2023)

Required User Inputs

Table 23: Below is a summary of the user inputs that are available and their definitions.

Variable	Definition (units)	Default	Min	Max
birth_weight_avg_ho	Average Holstein Birth Weight (kg/head)	42.9	1	NA
birth_weight_std_ho	Holstein Birth Weight Standard Deviation (kg/head)	69	0	NA
birth_weight_avg_je	Average Jersey Birth Weight (kg/head)	25.2	1	NA
birth_weight_std_je	Jersey Birth Weight Standard Deviation (kg/head)	4.4	0	NA
mature_body_weight_-avg	The average mature body weight of cows (kg/head)	740.1	0	NA
mature_body_weight_-std	The standard deviation of mature body weight of cows (kg/head)	73.5	0	NA
wean_day	The days of age at which calves are fully weaned from milk or milk replacer (days)	60	0	NA
target_heifer_preg_day	The target age (in days) for heifers to become pregnant - for adjusting heifer body weight (days)	420	NA	NA

Expected Outputs

- herd_statistics.avg_cow_body_weight (kg): the daily average bodyweight of all cows in the herd
- Pen_[pen ID]_[pen animal combination].avg_BW (kg): the daily average bodyweight of all animals in each pen
- sold_weight (kg)

Methodology Most methods determining the bodyweight of animals in RuFaS are implemented in growth.py.

The function evaluate_body_weight_change in growth.py calls the appropriate bodyweight change functions for each animal depending on their life stage and sums the appropriate changes into a single variable called daily_growth.

After setting the daily bodyweight change in the daily_growth variable, the evaluate_body_weight_change function updates each animal's bodyweight by adding it to the current body_weight. Thus for all animals, the final updated body_weight is set as:

$$\text{body\_weight}_{\text{updated}} = \text{body\_weight}_{\text{current}} + \text{daily\_growth}$$

- **Calf Birth Weight** is assigned in reproduction.py using a random draw from a truncated normal distribution based on user inputs for the breed-specific average and standard deviation of the birthweight. The distribution is truncated at +/- 2 SD from the mean to prevent extremely small or large calves.

$$Birth\ Weight = Random\ N\ (average\_birth\_weight\_by\_breed,\ std\_std\_birth\_by\_breed)$$

*Note - truncated at 2 standard deviations

[AN.BWT.6]

- **Calf Growth** Calves are assumed to double their weight between birth and weaning based on recommendations in the National Research Council, 2001. Thus the target average daily gain is estimated as:

$$target\_average\_daily\_gain = \frac{birth\ weight}{wean\ day}$$

[AN.BWT.7]

- **Non Pregnant Heifer Growth** The daily growth for non-pregnant heifers is estimated through an average daily gain to reach 55% of their manure bodyweight by the time they are pregnant. Because the actual age at pregnancy is not known, a user input for the target_heifer_preg_day is used to estimate the target average daily growth or ADG. A minimum daily growth rate of 0.5 kg/d is enforced so if the estimated daily_growth is less than 0.5 kg/d, the minimum value is used.

$$target\_average\_daily\_gain = min(\frac{0.55MSBW - SBW}{abs(targetheiferpregnantage - age)}, 0.5)$$

[AN.BWT.8]

Where MSBW is mature shrunk bodyweight (0.96 x mature bodyweight) and SBW is shrunk bodyweight (0.96 x bodyweight)

- **Pregnant Heifer Growth** Heifers are targeted to grow to 82% of their mature body weight by the time of their first calving. A gestation length is set at conception. The ADG of pregnant heifers is calculated based on the target heifer's body weight, gestation length, and days in pregnancy. The gestation length for each animal is set at conception by calculate_gestation_length in reproduction.py.

$$target\_average\_daily\_gain = \frac{0.82MSBW - SBW}{gestationlength - daysinpregnancy}$$

[AN.BWT.9]

- **Parity 1 and 2 Cow Growth** The daily growth of cows is calculated based on the target of reaching 92% of the mature body weight by the end of the 1st lactation, and full mature body weight at the end of the 2nd lactation. Before pregnancy, the daily growth is estimated based on the calving interval and during pregnancy the daily growth is calculated based on days until calving.

Parity 1 Animals: ***If not pregnant:***

$$target\_average\_daily\_gain = \frac{(0.92 - 0.82)MSBW}{calving\_interval}$$

[AN.BWT.10]

If pregnant:

$$target\_average\_daily\_gain = \frac{0.92MSBW - bodyweight}{gestation\_length - DIP}$$

[AN.BWT.11]

Parity 2 Animals ***If not pregnant:***

$$target\_average\_daily\_gain = \frac{(1 - 0.92)MSBW}{calving\_interval}$$

[AN.BWT.12]

If pregnant:

$$target\_average\_daily\_gain = \frac{(MSBW - bodyweight)}{gestation\_length - DIP}$$

[AN.BWT.13]

The gestation length for cows is estimated with the same methods that are used for heifers and the calving interval is set either as the average herd calving interval or the animal's own calving interval which is calculated as the age at her second calving minus her age at first calving as part of the daily_reproduction_update in animal.py.

- **Conceptus Growth** is estimated the same way for both heifers and cows using a non-linear model for animals that are greater than 50 days in pregnancy based on the methods developed by Korver, Arendonk, and Koops (1985).

  If the heifer or cow days in pregnancy is greater than 50 and less than the gestation length:

  First the expected total conceptus weight is estimated using an empirical equation based on the calf birth weight and the gestation length:

$$conceptus\_total\_weight = (0.0148 \ * \ gestation\_length) calf\_birth\_weight$$

[AN.BWT.14]

Then an animal specific conceptus parameter is calculated:

$$conceptus\_parameter = \frac{total\_conceptus\_weight^{\frac{1}{3}}}{gestation\_length - 50}$$

[AN.BWT.15]

Finally, the conceptus growth is estimated as:

$$conceptus\_growth = 3 \ * \ conceptus\ parameter^{3} \ * \ (DIP - 50)^{2}$$

[AN.BWT.16]

When days_in_pregnancy = gestation_length and the animal calves, then conceptus growth is set to the negative value of the current conceptus_weight such that the weight of the calf and placenta are subtracted from the cow's bodyweight when her bodyweight is updated.

$$conceptus\_growth = conceptus\_weight$$

[AN.BWT.17]

- **Tissue Change Due to Lactation** Lactation related tissue changes are estimated by a non-linear model presented by Galvão et al. (2013) that estimates the daily tissue mobilized during lactation.

$$bodyweight\_tissue\_change = \frac{P1}{P2} * exp\ (1 - \frac{DIM}{P2}) + DIM\ x\ exp\ (1 - \frac{DIM}{P2})$$

[AN.BWT.18]

Where P_1 and P_2 are parameters with distinct values for primiparous and multiparous cows:

Parameter	Primiparous	Multiparous
P_1	20	65
P_2	40	70

During the dry period, the net tissue change is assumed to be restored during the dry period. The net tissue change is estimated on the last day of lactation as:

$$tissue\_changed = P_1 * \frac{DIMlast}{P_2} * exp\ (1 - \frac{DIMlast}{P_2}) + DIM * exp\ (1 - \frac{DIM}{P_2})$$

[AN.BWT.19]

$$bodyweight\_tissue\_change = \frac{tissue\ changed}{gestation\ length - DIPdry}$$

[AN.BWT.20]

3.b Animal Reproduction

Introduction The reproduction component of the RuFaS model is designed based on reproductive program options recommended by the Dairy Cattle Reproduction Council (Dairy Cattle Reproduction Council, 2018) and experts in the field. It offers three pre-defined reproductive program options for nulliparous heifers (estrus detection [ED], timed artificial insemination [TAI], and synchronized estrus detection [Synch-ED]) and three options for lactating cows (ED, TAI, and a combined ED-TAI approach). Users can select appropriate reproductive protocols and specify reproductive performance variables for both nulliparous heifers and lactating cows.

This reproduction component simulates key reproductive events, including estrus occurrence, estrus detection, hormone administration, artificial insemination (AI), pregnancy checks, and pregnancy outcomes (abortion or successful delivery). Reproductive culling decisions are also modeled, based on an animal's inability to conceive or sustain a pregnancy. These processes are modeled using Monte Carlo simulations to represent variability in reproductive events.

The simulation generates and tracks detailed reproductive performance metrics at the animal level. These outputs provide insights into estrus detection, synchronization treatments, breeding efficiency, and pregnancy outcomes. The key recorded variables include:

- Estrus Detection (ED) Metrics: Number of days an animal participates in the ED program (ED_-days) and the total estrous events detected (estrus_count).
- Hormonal Treatments: Counts of synchronization protocol injections, including GnRH (GnRH_-injections), PGF (PGF_injections), and CIDR (CIDR_injections).

- Breeding Interventions: Number of semen straws used (semen_number) and total AI procedures performed (AI_times).
- Reproductive Efficiency: Time from first breeding attempt to pregnancy confirmation (breeding_to_pregnancy_time) and the interval from calving to confirmed pregnancy (calving_to_pregnancy_time).
- Pregnancy Monitoring: Total number of pregnancy checks performed (pregnancy_diagnoses).

Required User Inputs

Table 24: Below are the key user inputs that determine the **heifer reproductive breeding program** and impact reproductive efficiency and age at first calving.

Section	Variable	Definition	Default	Min	Max
management_decisions	breeding_start_day_h	Age when RuFaS initiates estrous protocol (days)	-	380	0
	heifer_repro_method	TAI (Timed AI), ED (Estrus Detection), Synch-ED (Synchronized ED)	TAI	TAI, ED, Synch-ED	
	heifer_repro_cull_time	Age when a heifer is culled if not pregnant (days)	-	500	0
farm_level_heifers	estrus_detection_rate	Proportion of estrus detected in ED protocol	0.9	0	1
	estrus_conception_rate	Conception rate in ED protocol	0.6	0	1
	repro_sub_protocol	Hormone protocol used; varies by method (e.g., 5dCG2P, CP, N/A)	5dCG2P	5dCG2P, 5dCGP, 2P, CP, N/A	
	conception_rate	Conception rate for sub-protocol	0.6	0	1
	estrus_detection_rate	Detection rate in sub-protocol	0.6	0	1

Table 25: The following inputs determine reproductive efficiency of cows and influence culling, calving interval, and calf output.

Section	Variable	Definition	Default	Min	Max
management_decisions	cow_repro_-method	TAI, ED, ED-TAI protocols	TAI	TAI, ED, ED-TAI	
	do_not_breed_-time	Days after calving to stop breeding if not pregnant	185	0	-
farm_level	voluntary_waiting_period	Days before breeding starts for ED/ED-TAI; ignored in TAI	50	0	-
	estrus_detection_protocol	Estrous protocol type	Double OvSynch	Double OvSynch, PreSynch, G6G, None	
farm_level_cows	estrus_conception_rate	Conception rate for ED/ED-TAI	0.5	0	-
	presynch_program	Presynch method used in TAI	OvSynch 56	OvSynch 48, OvSynch 56, CoSynch 72, 5d CoSynch, None	
	presynch_program_start_day	Day of first hormone injection for presynch	70	0	-
	ovsynch_program	OvSynch protocol used in TAI or ED-TAI	0.6	0	1
	ovsynch_program_start_day	Start day for OvSynch protocol	TAIafterPD	TAIafterPD, TAIbeforePD, PGFatPD, None	
	ovsynch_program_conception_rate	Conception rate for OvSynch program	-	-	-
	resynch_program	Resynch protocol around pregnancy check	-	-	-

NOTE: Anytime an insemination is performed after an OvSynch program, the conception rate will be changed to the ovsynch_program_conception_rate.

Table 26: The following inputs can be used to determine if there is a **decrease in the conception rate** with parity and rebreeding attempts.

Section	**Variable**	**Definition**	**Default**	**Min**	**Max**
farm_level	conception_rate_decrease	Percent decrease per breeding attempt after first	0.026	0.1	0
	decrease_conception_rate _-in_rebreeding	Whether to decrease rate with rebreeding	FALSE	-	-
	decrease_conception_rate _-by_parity	Whether to decrease rate by parity	FALSE	-	-

Relevant Outputs

Overall reporting for reproductive measures can be summarized and split into scheduled events and updates on an individual basis for heifers and cows.

These include:

- Events for Estrus (detection and scheduling)
- Hormone Administration
- Artificial Insemination (AI) (scheduling and status)
- Pregnancy Checks and Status
- Days Carry Calf
- Abortion Events
- Updates to Lactation
- Updates to Estrous
- Updates to Pregnancy
- Updates to Reproductive Status or Culling Decisions
- Updates to Protocols

Methodology ***Heifer Breeding***

Heifer II - The heiferII_reproduction_update represents the breeding-eligible heifer stage, beginning when the animal enters a reproductive management program and ending when she reaches a user-defined day in pregnancy that marks the start of the pre-fresh period (prefresh_day). The breeding process follows the assigned reproductive management program and is simulated as described below (Figure 2). Each program includes a user-defined conception rate (conception_rate), determining the probability of pregnancy initiation at the time of insemination.

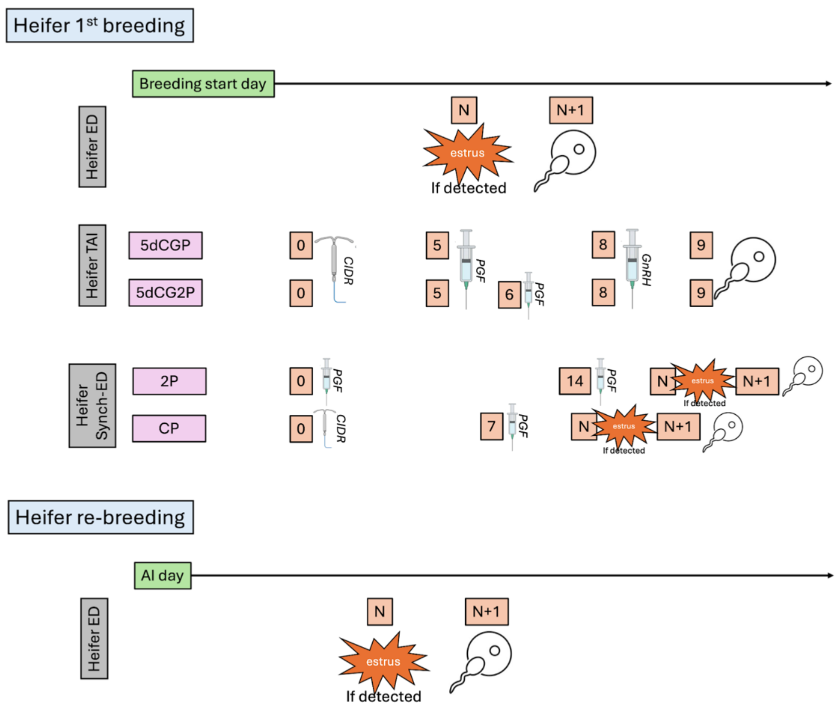


Figure 2: This is a high level schematic to illustrate the methodology behind the heifer breeding and re-breeding in RuFaS Animal Module.

Pregnancy diagnosis and reconfirmation occur after each insemination [heifer_pregnancy_update]. If a heifer remains nonpregnant beyond a user-defined age threshold (heifer_repro_cull_time, e.g., 500 days), she is culled due to reproductive failure. At a user-defined day in pregnancy (prefresh_day, e.g., one month before calving), the animal either transitions to the Heifer III stage or is sold as a replacement heifer.

The first-service reproductive management strategy for a heifer follows one of three strategies and are described in more detail below: estrus detection (ED), synchronized estrus detection (Synch-ED), or timed artificial insemination (TAI).

1. **ED [execute_heifer_ed_protocol]**: The ED strategy simulates spontaneous estrous cycles. Estrous cycle length is determined using a random draw from a normal distribution (De Vries et al., 2006). On the day of estrus predicted by the model, a random draw from a uniform distribution U(0,1) is compared with the estrus detection rate (estrus_detection_rate). If the drawn value is lower than the detection rate, AI is scheduled on that day. If estrus is not detected, the next estrus event is scheduled based on the normal estrous cycle. On the scheduled AI day, pregnancy success is determined using a random draw from a uniform distribution U(0,1), which is compared against the conception rate (conception_rate).
2. **Synch-ED [execute_heifer_synch_ed_protocol]**: In the Synch-ED strategy, hormonal treatment dates are scheduled according to the assigned program (repro_sub_protocol, 2P or CP), and estrus is simulated based on a program-specific estrous cycle following the hormonal treatment. Estrus detection and AI scheduling follow the same procedure as in the ED program.
3. **TAI [execute_heifer_tai_protocol]**: The TAI strategy simulates hormonal treatments administered on scheduled days from the start of the synchronization protocol. The timing of treatments and AI is modeled according to the assigned specific program (5dCGP or 5dCG2P) defined as a sub-protocol in the user inputs (repro_sub_protocol). On the scheduled AI day, pregnancy success

is determined using a random draw from a uniform distribution U(0,1), which is compared against the conception rate (conception_rate).

For all heifers who fail to conceive after the first insemination or experience pregnancy loss [open_heifer], the ED strategy is applied for subsequent breeding attempts.

Cow Breeding The breeding process for cows begins after calving and the voluntary waiting period (VWP), following one of three reproductive management strategies available for cows which are similar to those described in the heifer section and are pictured in Figure 3: ED, TAI, or ED-TAI. The program assigned to a cow determines the timing of insemination and the probability of pregnancy initiation.

- **ED [execute_cow_ed_protocol]**: In the ED strategy, spontaneous estrous cycles are simulated. The first estrous cycle after calving is determined on the calving date using a random draw from a normal distribution with user-specified mean and standard deviation (Kalantari, 2015). The subsequent estrous cycles are modeled similarly. If the scheduled estrus occurs before the VWP, the estrus is ignored. If the scheduled estrus occurs after the VWP, estrus detection is simulated following the same procedure as in the heifer ED strategy - a random draw from a uniform distribution U(0,1) is compared with the estrus detection rate (estrus_detection_rate). If the drawn value is lower than the detection rate, AI is scheduled on that day. If estrus is not detected, the next estrus event is scheduled according to the estrous cycle length. On the scheduled AI day, pregnancy success is determined using a random draw from a uniform distribution U(0,1), which is compared against the conception rate (ED_conception_rate).
- **TAI [execute_cow_tai_protocol]**: The TAI strategy simulates hormonal treatments administered on scheduled days according to the assigned synchronization protocol after the start of the breeding date (ovsynch_program_start_day). The timing of treatments and AI follow the predefined protocols (ovsynch_program, OvSynch-48, OvSynch-56, CoSynch-72, or 5d CoSynch). These protocols can be implemented alone or in combination with presynchronization protocols (presynch_program, PreSynch, Double OvSynch, or G6G). On the scheduled AI day, pregnancy success is determined using a random draw from a uniform distribution U(0,1), compared against the conception rate.
- **ED-TAI [execute_cow_ed_tai_protocol]**: The ED-TAI strategy integrates ED with TAI-based synchronization programs. Estrus detection is simulated before TAI begins. If estrus is detected after VWP (voluntary_waiting_period) and before the initiation of the TAI (ovsynch_-program_start_day), AI is scheduled according to the estrus detection protocol. If no estrus is detected, the cow follows the TAI protocol for insemination.

Important Note

For rebreedings, cows can be assigned to any of the three rebreeding strategies (resynch_program, TAIbeforePD, TAIafterPD, or PGFatPD) [open_cow].

- **TAIbeforePD**: The TAIbeforePD rebreeding strategy initiates a timed AI synchronization protocol before the first pregnancy diagnosis (PD). Cows that are due for pregnancy diagnosis but have not been confirmed pregnant yet start one of the predefined synchronization protocols (OvSynch-48, OvSynch-56, CoSynch-72, or 5-day CoSynch) six days before PD. This approach ensures that if a cow is found open (not pregnant) on PD day, she is already progressing through a synchronization protocol, reducing the waiting period before the next AI attempt.
- **TAIafterPD**: The TAIafterPD rebreeding strategy schedules synchronization for cows that are diagnosed as open on the PD day. If a cow is not pregnant, she is immediately enrolled in one of the available timed AI protocols (OvSynch-48, OvSynch-56, CoSynch-72, or 5-day CoSynch). This ensures timely rebreeding but does not initiate synchronization in advance of PD.

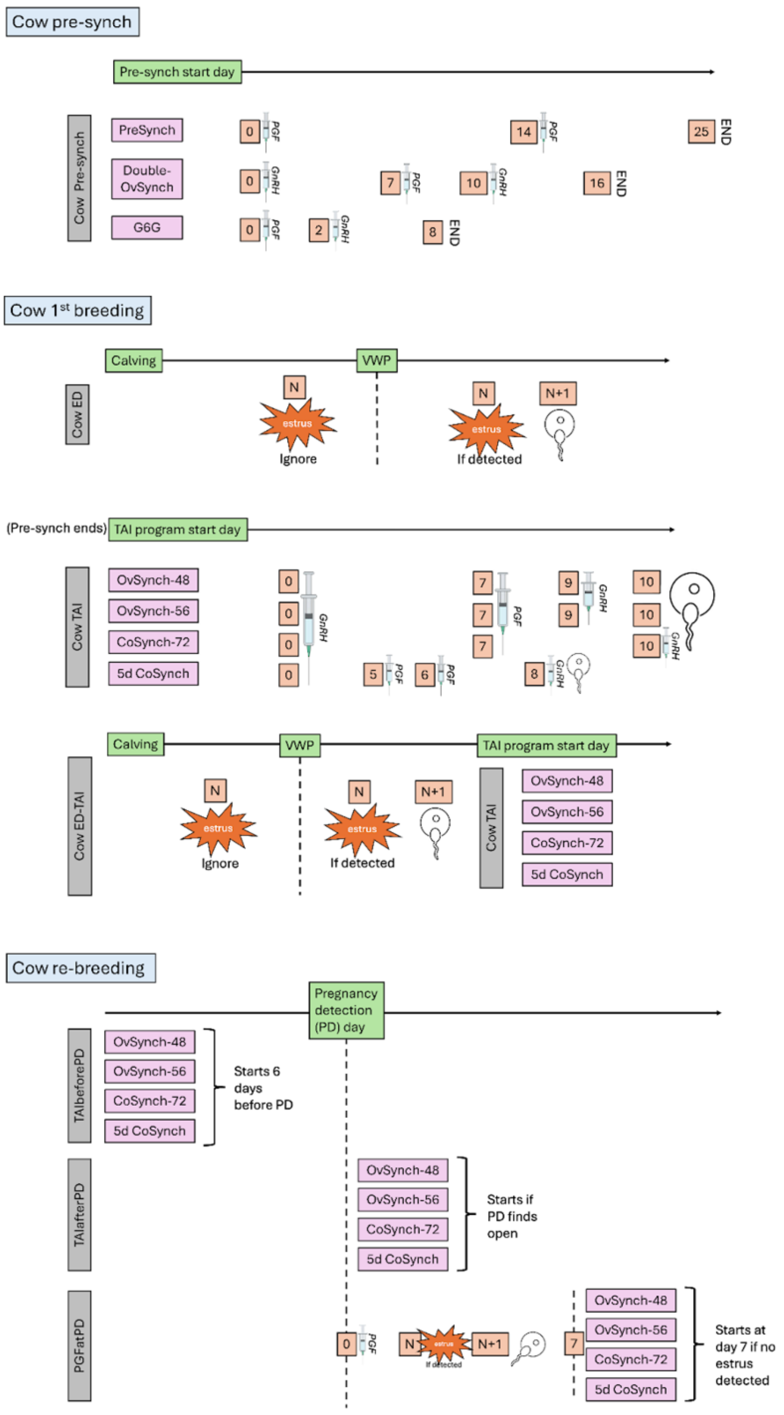


Figure 3: This is a high level schematic to illustrate the methodology behind the cow breeding and re-breeding in RuFaS Animal Module.

- **PGFatPD**: The PGFatPD rebreeding strategy uses prostaglandin (PGF) administration following PD for open cows to induce estrus. After PGF treatment (day 0), cows are monitored for natural estrus. If estrus is detected, AI is performed accordingly. If no estrus is detected within seven days, the cow is enrolled in a timed AI protocol (OvSynch-48, OvSynch-56, CoSynch-72, or 5-day CoSynch) to ensure insemination. This program balances estrus detection with timed AI, allowing for natural breeding opportunities before initiating synchronization.

Pregnancy Diagnosis (PD) and Pregnancy Loss For both heifers and lactating cows [heifer_-pregnancy_update, cow_pregnancy_update], pregnancy diagnoses are scheduled at three intervals after AI (preg_check_day_1, preg_check_day_2, preg_check_day_3). The default settings simulate the first at 32 d after insemination, which, if confirmed positive for pregnancy, is followed at 60 d and then at 200 d (Wiltbank et al., 2016); however, the number and day of pregnancy checks can be adjusted by the model user according to farm practices. Between AI and three pregnancy checks, pregnancy losses are simulated to occur according to random draws from a U (0,1) compared with thresholds according to the expected pregnancy losses in each of those periods (preg_loss_rate_1, preg_loss_rate_2, preg_-loss_rate_3). The default values are 0.02, 0.096, and 0.017 (Wiltbank et al., 2016). Users can adjust these numbers according to their herd performance.

On the next page are some important functions for the reproduction section of this document

Reproduction Functions

Table 27: Important functions for the reproduction section.

Function	Description	Parameters	Input Origins	Outputs
reproduction_-update	Updates the reproductive status of all animals daily, including estrus occurrence, detection, AI scheduling, pregnancy status, and culling decisions.	day: (int) Current simulation day. animal_list: (list) List of all animals in the herd.	day: Updated by the simulation engine. animal_-list: Provided as model input, updated dynamically.	Updated reproductive status for all animals in the herd. Scheduled events for estrus, AI, and pregnancy checks.
heiferII_reproduction _update	Manages reproduction processes for Heifer II animals, including estrus detection, breeding attempts, and pregnancy checks.	heifer: (object) Individual Heifer II animal.	heifer: Passed from the herd's current Heifer II group.	Updated estrus, AI status, and pregnancy details for the heifer.
cow_reproduction _update	Handles reproduction for lactating cows, including estrus detection, AI, and transitioning reproductive status.	cow: (object) Individual cow object. day: (int) Current simulation day.	cow: Passed from the herd's lactating group. day: Updated by the simulation engine.	Updated estrus, AI, pregnancy status, and culling decisions for the cow.
cow_give_birth	Manages calving events, updates the reproductive status, and initializes the lactation process.	cow: (object) Individual cow object. day: (int) Current simulation day.	cow: Passed from the pregnant group. day: Updated by the simulation engine.	Updated lactation and reproductive status for the cow.
execute_heifer _-ed_protocol	Executes the estrus detection protocol for heifers, simulating natural estrus occurrence and detection probability.	heifer: (object) Individual Heifer II animal. day: (int) Current simulation day.	heifer: Passed dynamically during estrus updates. day: Updated by the simulation engine.	Estrus detection and AI scheduling for the heifer.
execute_heifer _-tai_protocol	Manages the TAI protocol for heifers, simulating hormone administration and AI scheduling.	heifer: (object) Individual Heifer II animal. protocol: (dict) User-defined TAI protocol details.	heifer: Passed dynamically during protocol execution. protocol: Provided as user input.	AI scheduling and protocol updates for the heifer.

Function	Description	Parameters	Input Origins	Outputs
execute_heifer_-synch_ed_protocol	Executes synchronized estrus detection protocol for heifers, combining hormonal synchronization and estrus detection.	heifer: (object) Individual Heifer II animal. protocol: (dict) User-defined Synch-ED protocol details.	heifer: Passed dynamically during protocol execution. protocol: Provided as user input.	Estrus detection, hormone administration, and AI scheduling for the heifer.
open_heifer	Handles open status updates for heifers, reassigning protocols or marking for culling if reproduction fails.	heifer: (object) Individual Heifer II animal.	heifer: Passed dynamically during daily updates.	Updated reproduction status or culling decision for the heifer.
heifer_pregnancy_update	Updates pregnancy status for heifers, including pregnancy checks and abortion events.	heifer: (object) Individual Heifer II animal. day: (int) Current simulation day.	heifer: Passed dynamically during daily updates. day: Updated by the simulation engine.	Pregnancy status, days in pregnancy, or abortion event for the heifer.
execute_cow_-ed_protocol	Executes estrus detection protocol for cows, simulating natural estrus occurrence and detection probability.	cow: (object) Individual cow object. day: (int) Current simulation day.	cow: Passed dynamically during estrus updates. day: Updated by the simulation engine.	Estrus detection and AI scheduling for the cow.
execute_cow_-tai_protocol	Manages the TAI protocol for cows, simulating hormone administration and AI scheduling.	cow: (object) Individual cow object. protocol: (dict) User-defined TAI protocol details.	cow: Passed dynamically during protocol execution. protocol: Provided as user input.	AI scheduling and protocol updates for the cow.
execute_cow_-ed_tai_protocol	Executes ED-TAI protocol for cows, combining estrus detection with synchronized TAI scheduling.	cow: (object) Individual cow object. protocol: (dict) User-defined ED-TAI protocol details.	cow: Passed dynamically during protocol execution. protocol: Provided as user input.	Estrus detection, hormone administration, and AI scheduling for the cow.
cow_pregnancy_update	Updates pregnancy status for cows, including pregnancy checks and abortion events.	cow: (object) Individual cow object. day: (int) Current simulation day.	cow: Passed dynamically during daily updates. day: Updated by the simulation engine.	Pregnancy status, days in pregnancy, or abortion event for the cow.
open_cow	Handles open status updates for cows, reassigning protocols or marking for culling if reproduction fails.	cow: (object) Individual cow object.	cow: Passed dynamically during daily updates.	Updated reproduction status or culling decision for the cow.

3.c Milk Production

Introduction This section describes how RuFaS determines the amount of milk and milk components produced by a particular cow on a particular day.

Milk yield is largely determined by lactation curve parameters. Wood's curve is the only lactation model implemented in RuFaS currently, but there is flexibility to potentially include other models as well. At the beginning of the simulation, the herd is assigned distributions for the Wood's curve parameters l, m, and n for each parity of cows (see more information in the Lactation Curve section). At the beginning of each new lactation, each cow is assigned a set of parameters for that lactation based on the herd's distributions for her current parity.

When the cow starts lactating, she produces milk from the calving day until dry or culled according to the assigned lactation curve model. Each day, each cow's milk production is calculated and recorded. Milk production is calculated from the Wood's curve equation using the given lactation curve parameters and days in milk, and then adjusted by applying random variation and any milk reduction required by other parts of the model. Milk components (protein, fat, and lactose) are calculated based on milk production and percent of each component for each cow.

Currently, the only source of milk reduction is if there is no possible ration using the feeds available that would support the baseline level of milk production. Ration formulation is attempted with reduced milk production levels until reaching an acceptable ration (that meets the nutrient requirements for the pen's average cow) or reaching a threshold (floor) milk production defined by some combination of user inputs and pre-defined constants.

Required User Inputs

Table 28: The following user inputs are used to estimate customized herd-level lactation curve parameters. See the Methodology section of this document and the Lactation Curve Scientific Documentation for more detail about how these inputs are used with relation to daily milk production.

Section	Variable	Definition	Default	Min	Max
config _properties	end_date	The year and Julian day on which the simulation will end	2009:100	- (cutoff at 2006)	– (cutoff at 2016)
	FIPS_county_code	Unique 5-digit code that represents a specific US county	-	1000	56045
animal _properties	cow_times_milked_-per_day	Number of Milkings (per day) – The average or most common number of times cows are milked per day (1, 2, or 3 times daily)	3	0	-
	parity_fractions	Fractions of the milking animal population that are parity 1, 2, and 3 and beyond. The sum of these fractions must be 1.0	1: 0.346; 2: 0.272; 3: 0.379	1	0
	annual_milk_yield	The total milk yield generated by the farm in one year (kg). If this information is not available, it can be input as null	null	-	0
lactation_properties	milking_cow_fraction	Fraction of cows assumed to be in milk at any given time	0.8365 (305/365)	1	0

Table 29: The following inputs are used to determine the milk fat and milk protein percentages for each lactating cow

Section	Variable	Definition	Default	Min	Max
animal_properties	milk_fat_percent	The average or most common milk fat percentage in cow milk	3.5	0	
	milk_protein_percent	The average or most common milk protein percentage in cow milk	3.2	0	

Table 30: The following inputs are used to constrain how much an average cow's daily milk production will be reduced due to an inadequate ration. See the Methodology section of this document and the Ration Formulation scientific documentation for more detail about how these inputs are used with relation to daily milk production.

Section	Variable	Definition	Default	Min	Max
feed_properties	milk_reduction_maximum	Allowable amount of milk reduction (kg) when dietary nutrient supply cannot meet animal requirements		0	50
	tolerance	Allowable +/- percentage variance in each of the defined ration inclusion percentage values. *This variable defaults to 0.0 and currently has no effect due to no implementation of an automatic ration optimizer. RuFaS is hard-coded to use the user-defined ration at this time.*	0	0	1

Relevant Outputs

- report_life_cycle_manager_data:
  - Information added daily from herd statistics
  - daily_milk_production
  - herd_milk_fat_percent
  - herd_milk_fat_kg
  - herd_milk_protein_percent
  - herd_milk_protein_kg
- report_average_nutrient_requirements:
  - Reports the average ration per animal of each pen each time a new ration is formulated
  - avg_milk_production_reduction_pen (kg per animal)

Methodology

1. ***Herd Level Lactation Curve Parameters***

   At the beginning of the simulation, the herd is assigned distributions for the lactation curve parameters l, m, and n for each parity of cows. It calculates Wood's lactation curve parameter mean values by parity, adjusted based on the location, production, management practices, +/- reported milk production of the farm being simulated.

   User Inputs and Constants

   The following inputs are used to generate the herd's values for mean lactation curve parameters by parity.

   - year: end_date in the input config.json. Uses 2016 if the end year is 2016 or later.
   - region: assigns the appropriate region based on FIPS_county_code entered in the input config.json
   - milking frequency: cow_times_milk_per_day in input animal.json
   - parity distribution: parity_fractions in input animal.json
   - annual milk yield (optional): annual_milk_yield in animal.json
   - Milking_cow_fraction: 0.8356 default value
     - Is in lactation_curve_adjustment_inputs.json
     - Fraction of lactating cows in adult cows, between 0 and 1.
     - Used to calculate adjusted lactation curve parameters to match a herd's reported annual milk production

   Parameters and Equations

   These are comprehensively outlines in the Lactation Curve Parameters Determination Section.

   LactationCurve.set_lactation_parameters() is called by:

   - HerdFactory.initialize_herd() to Animal.setup_lactation_curve_parameters()
   - HerdManager._init_()

2. ***Cow Level Lactation Curve Parameters***
   Each cow is assigned a set of parameters for each lactation based on the herd's distributions of l, m, and n for her current parity. This occurs when a cow is initialized at the beginning of the simulation, when a heiferIII (close-up heifer) transitions into a cow, and each time a cow has a calf. It selects Wood's curve parameters for the cow for each parameter for each parity from the distributions based on the means calculated for the herd and constant standard deviations.

   User Inputs and Constants

   No additional inputs to consider.

   Parameters and Equations

   - LactationCurve.get_wood_parameters() is called by
     - Animal._initialize_cow
     - Animal.daily_reproduction_update
     - Animal.transition_heiferIII_to_cow
   - Parity groupings are as follows: 1, 2, and 3+
   - Immediately followed in each case by self.milk_production.set_wood_parameters() which assigns the l, m, and n values.

3. ***Daily Milk Production***

   Each day, each cow's milk production is calculated and recorded. Milk production is calculated from the Wood's curve equation using the given lactation curve parameters and days in milk. The calculated production is then adjusted by applying random variation and any milk reduction required by other parts of the model.

   User Inputs and Constants

   No additional inputs to consider but constants are below:

   - DAILY_MILK_VARIATION_MEAN: 0 kg/d
   - DAILY_MILK_VARIATION_STD_DEV: 1.0 kg/d

   Parameters and Equations

$$Calculated\ milk\ production = l * t^{m} * e^{[-nt]}$$

[AN.MLK.9]

   - l, m, n = lactation curve parameters
   - t = DIM

   Adjusted milk production = maximum of (calculated milk production + milk production variance - milk production reduction) or 0.

   - MilkProduction.calculate_daily_milk_production()
     - **Called by** HerdManager.daily_routines() to Animal.daily_routines() to MilkProduction.perform_daily_milking_update()
     - **Parameters**

    * Days_in_milk: Argument supplied from MilkProductionInputs (updated in Animal.daily_routines)
    * L_param, m_param, n_param: Wood_l, wood_m, wood_n supplied as arguments and assigned to individual on the day of calving via set_wood_parameters()
  - **Returns** calculated milk yield on the provided day in kg

- MilkProduction.daily_milk_produced()
  - **Inputs**
    * _Daily_milk_produced (calculated above)
    * Milk_production_variance is generated based on the mean and standard deviation for daily milk variation (stored as constants in AnimalModuleConstants)
    * Milk_production_reduction
- **Returns** adjusted_milk_production (unless the milk yield calculated above was 0, in which case it returns 0)

4. ***Daily Components Production***

Milk components (true protein, fat, and lactose) are calculated daily based on that day's milk production and percent of each component.

User Inputs and Constants

- milk_fat_percent – single constant value for all cows in the herd
- milk_protein_percent – single constant value for all cows in the herd
- AnimalModuleConstants.MILK_LACTOSE, current value 4.85

Parameters and Equations

- $Daily\ nutrient\ amount\ in\ milk = milk\ production * nutrient\ percentage * 0.01$
- MilkProduction._calculate_nutrient_content()
  - **Called by** HerdManager.daily_routines() to Animal.daily_routines() to MilkProduction.perform_daily_milking_update()
  - **Parameters**
    * Days_milk_produced as calculated in the Daily Milk Production section above
    * Nutrient_percentage: cow's percent true protein, fat, or lactose for each of the respective calculations
  - **Returns** Amount of a given nutrient in milk in kg

Please note that currently, true protein is the only milk protein currently used by the model. .

5. ***Milk Production and Reduction***

Currently, the only source of milk reduction is if there is no possible ration using the feeds available that would support the baseline level of milk production.

Each time the ration is formulated, the nutrition requirements of the pen are calculated and used to determine the ration given to each animal. The ration is checked against the nutrition requirements of the pen. If the pen contains lactating cows and the ration does not meet the average cow's requirements, then each cow's milk production is reduced until one of three conditions is met:

- The ration meets the animal's requirement (i.e. no additional reduction).
- The milk production of every animal is reduced by the maximum allowed (a user-defined value with default of 0.5 kg/head/d). Then the final ration and milk reduction values are used.

- The average milk production of the pen falls below the minimum allowable average milk production. Then the final ration and milk reduction values are used.

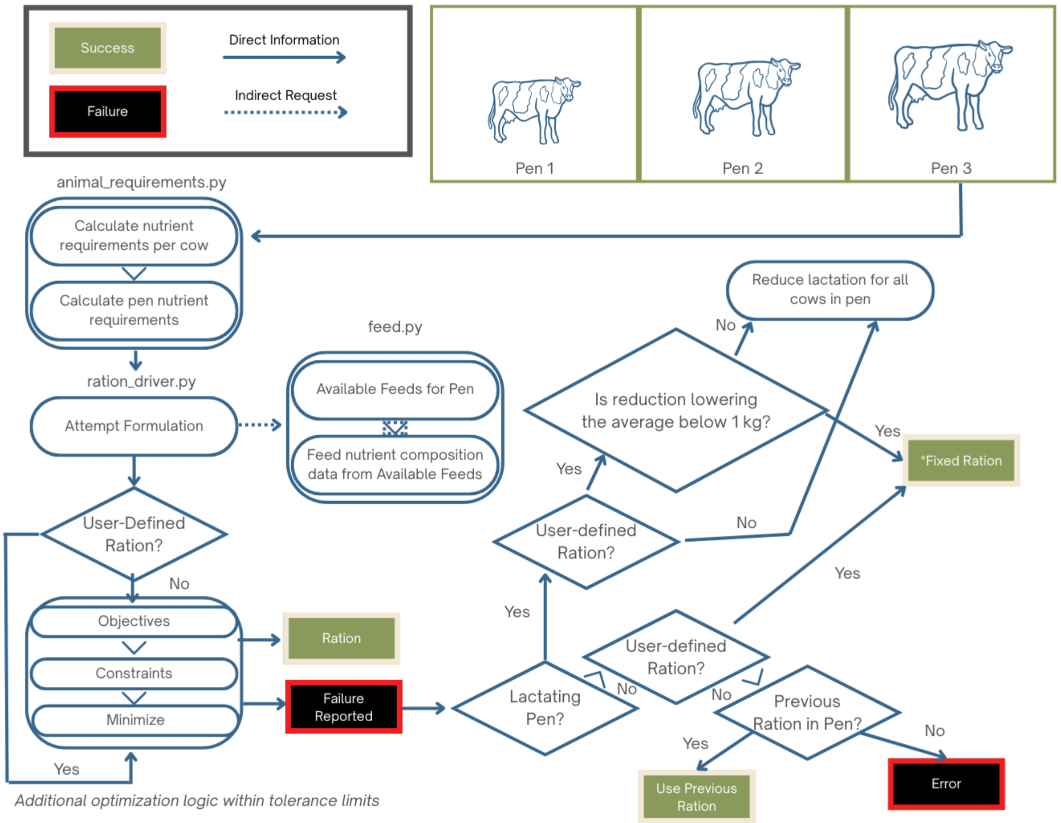


Figure 4: A schematic illustrating the flow of milk reduction in ration formulation logic.

User Inputs and Constants

- Milk_reduction_maximum: Allowable amount of milk reduction (kg) when dietary nutrient supply cannot meet animal requirements
- Tolerance: Allowable +/- percentage variance in each of the defined ration inclusion percentage values. This variable is default to 0.0 and it currently has no effect due to no implementation of automatic ration optimizer.
- Milk_reduction_kg is set to 0.25 kg: Milk reduction amount for each failed ration attempt, kg.
- Minimum_avg_pen_milk is set to 15 kg: Minimum allowable average milk production, for a given pen, as used in ration formulation, kg/animal

Parameters and Equations

- Pen.reduce_milk_production()
  - **Called by** Pen.use_user_defined_ration() when NutritionEvaluator determines that the ration is inadequate to support milk production
  - Attempts to reduce the milk production of all animals in the pen; returns False if all animals have already reached their maximum

- Animal.reduce_milk_production()
  - **Called by** Pen.reduce_milk_production for each animal in the pen
  - Attempts to reduce milk production of an animal in a pen; returns True if reduction was successful and False otherwise. When reduction is successful, increments milk_-production_reduction by the value

> Currently, all cows in a pen will have the same milk reduction amount. This function allows future flexibility if desired to have individual reductions rather than applying the average..

6. ***Starting and Ending Lactation***

   Starting Lactation [Animal.transition_heiferIII_to_cow]: When a heifer reaches the point where her days pregnant is equal to gestation length, she is transitioned to a lactating cow and configuration data for a newborn calf is returned.

   [Reproduction.cow_give_birth] and **[Animal.daily_reproduction_update]** When a cow's days pregnant is equal to gestation length, she gives birth and a newborn calf is born. Her parity is incremented plus one, her days in milk is reset to 1, and she is assigned new Wood's curve parameters for the new lactation.

   > For more information, refer to the Reproduction section of this document.

   Ending Lactation [MilkProduction.perform_daily_milking_update]: When a cow's days pregnant is equal to her dry off days of pregnancy, her milk production and days in milk are set to 0 and remain there until completing her gestation. Cows with days_in_milk = 0 are considered not to be milking and do not have any milk production values calculated.

   > For more details on milk changes and reductions, go to the Milk Production and Reduction portion of this section. .

3.d Lactation Curve: Parameters and Definitions

Introduction The Lactation Curve Parameter Determination module calculates and adjusts Wood's lactation curve parameters (l, m, n) based on farm-specific attributes such as geographic location, management practices, and production levels. These parameters shape the milk production curve for cows across different parities (1st, 2nd, 3rd + lactations).

This module is part of the Animal Module and integrates with the Milk Production calculations. The outputs influence various downstream models, including nutrient requirements, manure excretion, and greenhouse gas emission calculations.

The primary class in this module is:

LactationCurve: Manages and adjusts Wood's lactation curve parameters.

Required User Inputs

The module retrieves input data from simulation configuration files and the Input Manager. These inputs are organized into groups based on their functional roles. Each input is detailed by its parameter name, definition (including units where applicable), and any default values or constraints. There are three categories on inputs that should be considered in this section:

1. Wood's Lactation Curve Base Parameters

2. Adjustment Factors
3. Farm Specific 305 d Milk Production

Lactation Curve Parameters and Inputs

Table 31: Wood's Lactation Curve Base Parameters – These inputs provide the baseline values and variability for Wood's lactation curve parameters as derived from Li, Rosa, et al. (2022).

Variable	Definition	Units	Default
parameter_l_mean	Baseline value for parameter l of Wood's lactation curve	Unitless	19.9
parameter_m_mean	Baseline value for parameter m of Wood's lactation curve	Unitless	0.247
parameter_n_mean	Baseline value for parameter n of Wood's lactation curve	Unitless	0.003376
parameter_l_std_dev	Standard deviation for parameter l	Unitless	Parity 1: 0.28; Parity 2: 0.54; Parity 3+: 0.51
parameter_m_std_dev	Standard deviation for parameter m	Unitless	Parity 1: 0.0046; Parity 2: 0.0064; Parity 3+: 0.0060
parameter_n_std_dev	Standard deviation for parameter n	Unitless	Parity 1: 3.77e-5; Parity 2: 5.82e-5; Parity 3+: 5.54e-5

Lactation Curve Adjustment Factors

Table 33: Adjustment Factors – These factors adjust the base parameters based on geographical, temporal, management, and parity considerations.

Variable	Definition	Units	Default
region_adjustments	Regional adjustments based on FIPS codes (‘fips_-code‘) in the configuration file	Unitless	–
year_adjustments	Yearly adjustments based on calving year, defined by simulation time period (‘time.end_date.year‘)	Unitless	2006–2016
milking_frequency _-adjustments	Adjustments based on milking frequency (‘cow_-times_milked_per_day‘) in the animal input file	Unitless	2 or 3 *(Classified as twice-daily ($<$ 2.5 milkings/day) or thrice-daily ($\geq$ 2.5 milkings/day))*
parity_adjustments	Adjustments to lactation curve based on parity (first, second, third+)	Unitless	1–3

Farm Specific 305-Day Milk Production Inputs

Table 35: Farm Specific 305-Day Milk Production — These inputs help tailor the lactation curve to the farm's production targets and herd composition.

Variable	Definition	Units	Default
parity_fractions	Fraction of cows by lactation parity	Fraction	Parity 1: 0.346; Parity 2: 0.272; Parity 3+: 0.379
annual_milk_yield	Target annual milk yield to fit the lactation curve	kg	User-defined, default: null
milking_cow_fraction	Fraction of cows that are milking	Fraction	0.8365 (User-defined)

Relevant Outputs

Table 36: Summary of important functions for the reproduction section.

Parameter	Unit	Description
l	Unitless	Farm-specific scaling factor of the lactation curve by parity.
m	Unitless	Farm-specific rate of milk yield increase post-calving by parity.
n	Unitless	Farm-specific rate of milk yield decline after peak lactation by parity.

Methodology

1. Determining Base Lactation Curve Parameters (set_lactation_parameters). The Wood's lactation curve parameters are initialized with base values retrieved from lactation_inputs["parameter_-mean_values"] in LACTATION_CURVE_ADJUSTMENT_INPUTS.json.

2. Adjusting parameters (_calculate_adjusted_wood_parameters). Wood's lactation curve parameters are adjusted based on temporal, geographical, and management factors (Li, Rosa, et al., 2022)

   Each parameter (l, m, n) is adjusted using a series of modifications:

$$l_{adjusted} = l + \sum adjustments$$

[AN.MLK.1]

$$m_{adjusted} = m + scalingfactor_m * \sum adjustments$$

[AN.MLK.2]

$$n_{adjusted} = n + scaling\ factor_n * \sum adjustments$$

[AN.MLK.3]

Where:

$$scaling\ factor_m = 10^{-2}$$
$$scaling\ factor_n = 10^{-4}$$

Adjustments are applied based on:

- Region (using FIPS codes): _get_region_adjustments
- Year of simulation (adjusted for trends in milk production): _get_year_adjustments
- Milking frequency (twice vs. thrice daily): _get_milking_frequency_adjustments
- Parity (different lactation curves for different parities): animal_parity

3. Fitting Parameters to Target Milk Yield (_adjust_lactation_curve_to_milk_yield). If annual_-milk_yield is provided, the l parameter is optimized to match the expected 305-day milk yield:

   - Estimate Herd Average 305-day Milk Yield:

$$Milk\ Yield_{305} = \frac{Annual\ Milk\ Yield}{Number\ of\ Milking\ Cows} * \frac{305}{Days\ per\ Year}$$

[AN.MLK.4]

   - Estimate Parity Specific 305-day Milk Yield: (_estimate_305_day_milk_yield_by_parity)

$$M305_{Herd} = M305_{P1} * P1\% + M305_{\mathrm{P2}} * P2\% + M305_{P3} * P3^{+}\%$$

[AN.MLK.5]

$$M305_{P2} = M305_{P1} * 118\%$$

[AN.MLK.6]

$$M305_{P3+} = M305_{P1} * 125\%$$

[AN.MLK.7]

118% and 125% are assumptions made based on the work by Li, Rosa, et al. (2022).

   - Fit l Parameter Using Optimization: (_fit_wood_l_param_to_milk_yield). The error function is defined as: (_calculate_305_day_milk_yield_error)

$$Error = Predicted\ MilkYield305 - Target\ MilkYield305$$

[AN.MLK.8]

Using Scipy's minimize function, the l parameter is iteratively adjusted to minimize this error.

4. Final Lactation Curve Parameter Mapping. Once adjustments are made, parameters are stored in:

_parity_to_parameter_mapping = 1: "l": ..., "m": ..., "n": ...,
2: "l": ..., "m": ..., "n": ...,
3: "l": ..., "m": ..., "n": ...

Equations below are not in the LactationCurve class, but they are useful to understanding the bigger picture of how lactation curve parameter estimates are generated in RuFaS. Please see the Milk Production section for more details on these equations (Wood, 1967).

Wood's Lactation Curve

$$Y_t = l * t^m * e^{-nt}$$

[AN.MLK.9]

Where:

- Y(t): Daily milk yield.
- t: Time in days.
- l, m, n: Parameters adjusted for farm-specific conditions.

305-Day Milk Yield

$$\int_l^{305} lt^m e^{-nt} dt$$

[AN.MLK.10]

3.e Culling

Introduction This section describes the processes of animals leaving the herd in RuFaS. Reasons for leaving the herd are death or sale.

Sale may occur of:

- Calves (all male calves; female calves based on keep_female_calf_rate)
- Heifer II's (if the heifer does not have a successful conception or pregnancy and age (days born) exceeds heifer_repro_cull_time
- Heifer III's (if cow herd does not have room for additional heifers)
- Cows (through two possible mechanisms):
  - Disease-related culling, determined by a statistical distribution of occurrence, and
  - Production-related culling, which occurs when a cow has been labeled "Do Not Breed" due to reproductive failure past a designated number of days in milk and then her milk production drops below a designated threshold)

Deaths that may occur:

- Cows, determined by a statistical distribution of occurrence
- Stillborn calves
  - *Note:* stillborn calves are currently lumped together with calves sold at birth and reported as "calves sold"

Information about death and sale events are tracked and aggregated daily, including the day of death or sale, body weight, cull reason, days in milk, and parity.

Required User Inputs

Table 37: The following inputs are used to determine the probability and timing of cow death and sales events.

Variable	**Definition**	**Default**	**Min**	**Max**
parity_death_prob	Death probability by parity group (1st, 2nd, 3rd, 4th+ lactation)	-	0	1
parity_cull_prob	Cull probability by parity group (1st, 2nd, 3rd, 4th+ lactation)	-	0	1
parity_day_prob*	CDF values for likelihood of death over time; used with ‘cull_day_count‘	-	0	1
cull_day_count*	Defines time segments (days in milk) for the CDF curve	-	0	-
probability*	Conditional probability of culling due to each of six reasons; sums to 1	-	0	1
cull_day_prob*	CDF values for likelihood of [reason] culling over time	-	0	1
do_not_breed_time	Days after calving a cow will be culled if not pregnant	0	185	-
cull_milk_production	Milk threshold (kg/d) below which cows are culled	0	30	-
herd_num	Target number of dry and lactating cows on the farm	100	-	6
heifer_repro_cull_time	Max age (days) before a heifer is culled for not getting pregnant	500	0	-
keep_female_calf_rate	Fraction of female calves kept and raised on-farm	1	0	1
male_calf_rate_sexed_semen	Male calf rate when using sexed semen	0.1	0	1
male_calf_rate_conventional_semen	Male calf rate when using conventional semen	0.53	0	1
still_birth_rate	Stillbirth rate	0.065	0	1

Note: Variables marked with * are commonly left at default values. Adjusting them changes the timing of cow exit from the herd but does not affect the total number of exits.

Relevant Outputs

Information about death and sales events are tracked and aggregated daily by HerdManager, including the animal type, day of death or sale, body weight, cull reason, days in milk, and parity.

All of the following are reported by AnimalModuleReporter:

- **Report_life_cycle_manager_data** - is reported on a daily basis for the numbers of calves, HeiferII, HeiferIII, and Cow sold (calves born deceased tracked as newborns sold). It also reports HeiferIII purchased and documents other cow culls (i.e. sold and died)

  Culling data is also reported here documenting the average age of cows and HeiferII culled and the reason (e.g. death, poor production, lameness, injury, mastitis, disease, udder, or unkown/other).
- **Report_sold_animal_information** - this data is reported at the end of a simulation. It creates a list of all animals sold and reports their ID number, type, body weight, day sold (simulation day), cull reason, days in milk, and parity.
- **Report_sold_animal_information_sort_by_sell_day** - also reported at the end of a simulation to document the overall number and weight of animals sold on that simulation day. It reports zero's for days on which no animal of that type was sold. This enables using report filters to slice and summarize sales statistics by animal type for a specified time period (e.g., number and body weight of cows sold over the last 365 days of the simulation).

Methodology ***Cow Deaths***

At the beginning of each lactation, each cow "rolls the dice" to determine if +/- when she will die in the current lactation.

Probability of death associated with each lactation (1, 2, 3, 4+) is specified at the beginning of the simulation as a user input. At the beginning of each lactation for each cow, it is determined whether she will die during that lactation with a random draw compared with the appropriate probability. Then, for individuals assigned to die during that lactation, an empirical Cumulative Distributive Function (CDF) is used to determine on which day the death will occur based on the probability of death occurring during each interval of the lactation.

The values for the CDF determining probability of death by day of lactation are located in the animal input json file (death_day_prob; See Table 4).

User Inputs and Constants

- Parity death probability: "parity_death_prob"
  - Probability of death during the current lactation for cows beginning a new lactation, grouped into lactation groups 1, 2, 3, and 4+
- Death day probability: "death_day_prob"
  - Values to construct the CDF to determine future death day, given that death is going to occur (default values in the table below)
  - Values in the array correspond to the days in cull_day_count (another user input; default values are the "days" in the table below)

Parameters and Equations

Determination of death date (once already determined that death will occur this lactation):

Death day of Cow X is as follows

$$X = x_{\text{i-1}} + a_{\text{i}} * (R - c_{\text{i-1}})$$

[AN.ANM.1]

Where:

R = random float number between 0-1

$c_{\text{i-1}}, c_{\text{i}}$ (the upper and lower values around R in the death day prob CDF),

$x_{\text{i-1}}, x_{\text{i}}$ (days, correspond to $c_{\text{i-1}}, c_{\text{i}}$)

Empirical CDF: $a_{\text{i}} = \frac{x_{\text{i-1}}, x_{\text{i}}}{c_{\text{i-1}}, c_{\text{i}}}$

- Animal.determine_future_death_date()
  - **Called by** by Animal.daily_routines() at the beginning of each new lactation (on the day a new calf is born).
  - **Returns** calculated future death date in simulation days or sys.maxsize (effectively no death date).
  - Logic: Draw random value between 0-1 and compare it to probability of death in that lactation group (parity_death_probability).
    - If drawn value $\leq$ parity_death_probability, then go on to select date that death will occur - draws another random value between 0-1
    - Compares random draw to the CDF defined by death_day_prob array to determine number of days into the future at which death will occur - adds that number of days to current days born to determine future death date .
    - If drawn value > parity_death_probability, then future death date is assigned as sys.maxsize (a really really big number – effectively, she will never reach her assigned future death date).
- Animal.animal_life_stage_update()
  - **Called by** HerdManager.daily_routines() $\rightarrow$ Animal.daily_routines()
  - Updates the animal status to DEAD if days born == future death date

Cow Sales

1. *Disease Related*

   At the beginning of each lactation, each cow "rolls the dice" to determine if +/- when and for what attributed reason she will be sold for disease-related reasons (i.e., involuntary, not directly for production/reproduction reasons) in the current lactation.

Probability of disease sales associated with each lactation (1, 2, 3, 4+) is specified at the beginning of the simulation as a user input. At the beginning of each lactation for each cow, it is determined whether she will be sold for disease during that lactation with a random draw compared with the appropriate probability. Then, for individuals assigned to be sold during that lactation, another random draw determines the reason for that sale from one of six possibilities (See Table 4). Finally, an empirical CDF is used to determine the day of sale based on the probability of sale occurring during each interval of the lactation for that particular sale reason.

User Inputs and Constants

- Parity cull probability: "parity_cull_prob"
  - Probability of sale for disease during the current lactation for cows beginning a new lactation, grouped into lactation groups 1, 2, 3, and 4+
- PFor each of the six reasons:
  - "Probability" - Conditional probability that a sold animal is removed due to each group of issues. This probability is used to determine the reason for sale after it has been decided that the animal will be sold during the current lactation. The sum of probabilities for the six culling reasons equals 1. See Table 2 below for default values used to construct the empirical PDF.
  - "Cull_day_prob": Values to construct the CDF to determine future sale day, given that disease-related sale is going to occur. Values in the array correspond to the days in cull_-day_count (another user input; default values are the "days"; default values in Table 4)

Parameters and Equations

Determination of sale date (once already determined that death will occur this lactation):

We compute the sale day of Cow X as follows

$$X = x_{\text{i-1}} + a_{\text{i}} * (R - c_{\text{i-1}})$$

[AN.ANM.2]

Where:

R = random float number between 0-1

$c_{\text{i-1}}, c_{\text{i}}$ (the upper and lower values around R in the sale day prob CDF),

$x_{\text{i-1}}, x_{\text{i}}$ (days, correspond to $c_{\text{i-1}}, c_{\text{i}}$)

Empirical CDF: $a_{\text{i}} = \frac{x_{\text{i-1}}, x_{\text{i}}}{c_{\text{i-1}}, c_{\text{i}}}$

- Animal.determine_future_cull_date()
  - **Called by** by Animal.daily_routines() at the beginning of each new lactation (on the day a new calf is born)
  - **Returns** a tuple with future cull (sale) date in simulation days and reason for culling
  - Logic: Draw random value between 0-1 and compare it to probability of death in that lactation group (parity_death_probability)

        * If drawn value ≤ parity_death_probability, then go on to select date that death will occur - draws another random value between 0-1
        * Compares random draw to the CDF defined by death_day_prob array to determine number of days into the future at which death will occur - adds that number of days to current days born to determine future death date
        * If drawn value > parity_death_probability, then future death date is assigned as sys.maxsize (a really really big number – effectively, she will never reach her assigned future death date)
- Animal.animal_life_stage_update()
    - **Called by** HerdManager.daily_routines() → Animal.daily_routines()
    - Updates the animal status to SOLD if days born == future cull date

2. *Production Related*

If a cow is not pregnant when she reaches the days in milk cutoff specified by the user, she will be marked as Do Not Breed and no longer participate in reproductive protocols. When her milk production drops below the production cutoff specified by the user, she is sold for the reason of low production. Note that the only cows who are sold for "low production" are those flagged as DNB for reproductive failure; no low-producing cow who is pregnant or has fewer DIM than the cutoff will be sold for low production.

User Inputs and Constants

- Do_not_breed_time: The length of the breeding period after parturition for cows after which reproduction protocols stop if they fail to get pregnant
- Cull_milk_production: Milk production threshold at which 'do not breed' cows are culled if they fall below (in kg/d)

Parameters and Equations

- Reproduction._check_do_not_breed_flag()
    - Checks if a cow should be marked as do-not-breed if not pregnant beyond the user-defined breeding window
    - Called by cow reproduction update() in Reproduction.py
    - If not pregnant and DIM > do_not_breed_time and not already do_not_breed:
        * Updates reproduction_data_stream with do not breed event
        * Updates self.do_not_breed to true
- Animal.life_stage_update()
    - Updates the life stage of an animal based on its type and current simulation time
    - Called by Animal.daily_routines()
    - Includes the following logic:
        * If a cow is do_not_breed and daily milk produced is less than user-defined cull_-milk_production:
            · Sold_at_day = simulation_day
            · Cull_reason = low_prod_cull
            · Animal_status = sold

Table 38: Cumulative distribution function (CDF) values associated with the conditional probability that a sold animal is removed due to each group of issues and/or the conditional probability of death over time. Corresponds to the "probability" value for each cull reason in the animal input json file. The 'cull_day_count' array defines the segments of the CDF

Variable	**0**	**5**	**15**	**45**	**90**	**135**	**180**	**225**	**270**	**330**	**380**	**430**	**480**	**530**
Cumulative Probability of Death	0	0.18	0.32	0.42	0.48	0.54	0.60	0.65	0.70	0.77	0.83	0.89	0.95	1
Mastitis CDF	0	0.06	0.12	0.19	0.30	0.43	0.56	0.68	0.78	0.85	0.90	0.94	0.97	1
Feet & Leg CDF	0	0.03	0.08	0.16	0.25	0.36	0.48	0.59	0.69	0.78	0.85	0.90	0.95	1
Injury CDF	0	0.08	0.18	0.28	0.38	0.47	0.56	0.64	0.71	0.78	0.85	0.90	0.95	1
Disease CDF	0	0.04	0.12	0.24	0.34	0.42	0.50	0.57	0.64	0.72	0.81	0.89	0.95	1
Udder CDF	0	0.12	0.24	0.33	0.41	0.48	0.55	0.62	0.68	0.76	0.82	0.89	0.95	1
Unknown CDF	0	0.05	0.11	0.18	0.27	0.37	0.45	0.54	0.62	0.70	0.77	0.84	0.82	1

If a cow is not pregnant when she reaches the days in milk cutoff specified by the user, she will be marked as Do Not Breed and no longer participate in reproductive protocols. When her milk production drops below the production cutoff specified by the user, she is sold for the reason of low production. Note that the only cows who are sold for "low production" are those flagged as DNB for reproductive failure; no low-producing cow who is pregnant or has fewer DIM than the cutoff will be sold for low production.

4 Methane Emission and Manure Excretion

4.a Manure Excretion

Introduction The Manure Excretion Calculator module computes manure excretion for different animal categories, including calves, growing heifers, lactating cows, and dry cows. It calculates the quantities of total mass and nutrient composition in manure. The implementation is designed for flexibility, enabling integration with other RuFaS modules for modeling farm management scenarios. This module includes:

- *Classes:* ManureExcretionCalculator
- *Key Methods:*
  - calculate_calf_manure
  - calculate_heifer_manure
  - calculate_cow_manure
  - _calculate_lactating_cow_manure
  - _calculate_dry_cow_manure
  - _calculate_phosphorus_excretion_values

This module integrates with the process_digestion method in the DigestiveSystem class to calculate manure excretion on a daily and per-animal basis. It plays a critical role in linking animal-level digestion processes with downstream environmental and nutrient management models.

The module takes inputs that include general animal properties (e.g., body weight, nutrient intake, and nutrient compositions), milk production characteristics (e.g., fat and protein content), specific nutrient properties (e.g., urine phosphorus requirements), and animal categories (e.g., CALF, HEIFER, COW). The outputs from this module are detailed manure excretion metrics, such as total manure mass, nutrient content, and volatile solids. In addition, an animal-class specific methane potential is assigned to manure upon excretion. These outputs are used by the Manure Module for downstream calculations, including manure methane (CH_4) emissions and nutrient balance tracking.

Required User Inputs
The Manure Excretion Calculator module computes manure excretion metrics for various animal categories. Inputs are drawn from pen ration files and animal module constants. They are organized into the following groups:

1. General Animal Inputs
2. Cow Specific Inputs
3. Phosphorus Specific Inputs

General Inputs Manure Excretion

Table 39: These inputs define basic animal and diet properties used to calculate nutrient intake and manure production.

Variable	Definition	Units
body_weight	Body weight of the current animal	kg
dry_matter_intake	Dry matter intake from pen ration	kg
dry_matter_concentration	Dry matter concentration of the diet	%
crude_protein_conc	Crude protein concentration in the diet	%
potassium_concentration	Potassium concentration in the diet	%
ash_diet_content	Ash content of the diet	%
neutral_detergent_fiber_concentration	Neutral detergent fiber concentration	%
acid_detergent_fiber_concentrations	Acid detergent fiber concentration	%
nutrient_concentrations	Nutrient concentrations (e.g., CP, NDF, ADF, ash)	%

Cow Specific Inputs – Manure Excretion

Table 40: These inputs are used when modeling cows and are particularly relevant for lactating animals.

Variable	Definition	Units	Default	Note
is_lactating	Indicates the cow's lactating status	Boolean	-	-
days_in_milk	Days post-parturition; for lactating cows only	days	-	Intermediate output
milk_protein	Milk protein content; for lactating cows only	%	3.2	-
daily_milk_production	Daily milk production; for lactating cows only	kg	-	-
body_weight	Body weight of the cow	kg	-	-
milk_fat	Milk fat content	%	3.5	-
metabolizable_energy_intake	Metabolizable energy intake per kg DM	Mcal/kg DM	-	-

Phosphorus Excretion Inputs – Manure Excretion

Table 41: These inputs define phosphorus-related outputs relevant to lactating cows.

Variable	Definition	Units
fecal_phosphorus	Amount of fecal phosphorus excreted by the current animal; Intermediate model output	g
urine_phosphorus_required	Phosphorus required for urine production; Intermediate model output	g
total_manure_excreted	Total manure excreted by the animal; Intermediate model output	kg

Expected Outputs

The outputs from the Manure Excretion Calculator are organized into two main categories: manure excretion values and manure nutrient composition. These outputs are consistently generated across animal categories.

Table 42: Expected outputs from this section.

Category	Parameter	Definition	Units
Manure Excretion	Urea	Concentration of urea in the manure	g/L
	Urine	Amount of urine excreted	kg
	Manure Mass	Total mass of manure excreted	kg
	Total Solids	Dry matter content of the manure	kg
	Degradable Volatile Solids	Portion of volatile solids that are degradable	kg
	Non-degradable Volatile Solids	Portion of volatile solids that are non-degradable	kg
Manure Nutrient Composition	Manure Methane Potential	The maximum methane producing capacity of volatile solids from manure	m^3 CH_4 / kg VS
	Manure Total Ammoniacal Nitrogen	Total ammoniacal nitrogen excreted in the manure	kg
	Urine Nitrogen	Amount of nitrogen excreted via urine	kg
	Manure Nitrogen	Total nitrogen excreted in manure	kg
	Inorganic Phosphorus Fraction	Fraction of manure phosphorus that is inorganic	-
	Organic Phosphorus Fraction	Fraction of manure phosphorus that is organic	-

Continued on next page

Table continued from previous page

Category	Parameter	Definition	Units
	Non-water Inorganic Phosphorus Fraction	Portion of inorganic phosphorus not water-extractable	-
	Non-water Organic Phosphorus Fraction	Portion of organic phosphorus not water-extractable	-
	Total Phosphorus Excreted	Total phosphorus excreted by the animal	g
	Phosphorus Fraction	Overall fraction of phosphorus in the manure	-
	Potassium	Amount of potassium excreted in the manure	g

Methodology

- **Calf Manure Excretion (calculate_calf_manure)**

  Manure excretions Manure: amount of feces and urine excreted daily by a calf (kg/d)

$$total\_manure\_excreted = 3.45 * dry\_matter\_intake$$

[AN.EXC.1]

Urine: amount of urine excreted in kg. This is an assumption.

$$urine = 2.0$$

[AN.EXC.2]

Manure total solids: amount of dry material excreted by the calf (kg/d).

$$total\_solids = 0.393 * dry\_matter\_intake$$

[AN.EXC.3]

Total volatile solids (kg/d)

$$total\_volatile\_solids = 0.0023 * body\_weight$$

[AN.EXC.4]

Degradable volatile solids excretion (kg/d)

$$degradable\_volatile\_solids = 0.9 * total\_volatile\_solids$$

[AN.EXC.5]

Non-degradable volatile solids excretion (kg/d)

$$non\_degradable\_volatile\_solids = total\_volatile\_solids - degradable\_volatile\_solids$$

[AN.EXC.6]

Manure nutrient composition Manure N excretion (kg/d):

$$manure\_nitrogen = \frac{(112.55 * dry\_matter\_intake * \frac{crudeprotein\_concentration}{100})}{1000}$$

[AN.EXC.7]

Urine N excretion (kg/d)

$$urine\_nitrogen = 0.45 * manure\_nitrogen$$

[AN.EXC.8]

Manure total ammoniacal nitrogen (kg/d)

$$manure\_total\_ammoniacal\_nitrogen = urine\_nitrogen$$

[AN.EXC.9]

- **Heifer Manure Excretion (calculate_heifer_manure)**

Manure excretions Urine excretion (kg/d)

$$urine = 9.0$$

[AN.EXC.10]

Total manure excretion (kg/d)

$$total\_manure\_excreted = 4.158 * dry\_matter\_intake - 0.0246 * body\_weight$$

[AN.EXC.11]

Because the equations that predict total manure excreted and total solids excreted rely on different inputs, in some extreme cases, the total mass of manure excreted can be predicted to be very close to the predicted total solids excreted which is unrealistically dry. For this reason, we set a maximum manure dry matter content of 20% for heifers and dry cows. If the mass predicted by equation **[AN.EXC.11]** is not greater than (total solids/minimum dry matter) then,

$$total\_manure\_excreted = \frac{total\_solids}{dry\_matter\_manure}$$

[AN.EXC.#]

Total solids excretion (kg/d) - this is the same equation used for dry cows.

$$total\_solids = 0.178 * dry\_matter\_intake + 2.733$$

[AN.EXC.12]

Total volatile solids excretion (kg/d)

$$total\_volatile\_solids = 0.0073 * body\_weight$$

[AN.EXC.13]

Degradable volatile solids excretion (kg/d)

$$degradable\_volatile\_solids = 0.9 * total\_volatile\_solids$$

[AN.EXC.14]

Non-degradable volatile solids excretion (kg/d)

$$non\_degradable\_volatile\_solids = total\_volatile\_solids - degradable\_volatile\_solids$$

[AN.EXC.6]

Manure nutrient composition Manure N excretion (kg/d)

$$manure\_nitrogen = (15.1 + 0.83 * \frac{(dry\_matter\_intake * 1000) * \frac{crude\_protein\_concentration}{\frac{6.25}{100}}}{1000})$$

[AN.EXC.14]

Fecal N excretion (kg/d)

$$fecal\_nitrogen = (0.345 + 0.317 * \frac{(dry\_matter\_intake * 1000) * \frac{crude\_protein\_concentration}{\frac{6.25}{100}}}{1000})$$

[AN.EXC.15]

Urine N excretion (kg/d)

$$\text{urine}_{\text{nitrogen}} = \text{manure}_{\text{nitrogen}} - \text{fecal}_{\text{nitrogen}}$$

[AN.EXC.16]

Manure total ammoniacal nitrogen (kg/d)

$$manure\_total\_ammoniacal\_nitrogen = urine\_nitrogen$$

[AN.EXC.9]

- **Heifer Manure Excretion (calculate_heifer_manure)**

  Manure K excretion (g/d)

$$potassium = 1000 * dry\_matter\_intake * \frac{potassium\_concentration}{100}$$

[AN.EXC.17]

- **Lactating Cow Manure Excretion (_calculate_lactating_cow_manure)**

  Manure excretions Fecal water (kg/d): Calculate the amount of fecal water excreted (kg/d)

$$fecal\_water = 1.987 * dry\_matter\_intake + 0.348 * acid\_detergent\_fiber\_concentrations - 0.412 * crude\_protein\_concentration - 0.074 * dry\_matter\_concentration - 0.0057 * days\_in\_milk$$

[AN.EXC.18]

Total solids/Fecal dry matter (kg/d): Calculate the amount of fecal dry matter excreted (kg/d). Fecal dry matter is assumed to be equivalent to total solids.

$$fecal\_solids = 0.576 + 0.370 * dry\_matter\_intake - 0.075 * crude\_protein\_concentration + 0.059 * acid\_detergent\_fiber\_concentrations$$

[AN.EXC.19]

Urine (kg/d)

$$urine = -7.742 + 0.388 * dry\_matter\_intake + 0.726 * crude\_protein\_concentration + 2.066 * milk\_protein$$

[AN.EXC.20]

Total manure excretion (kg/d)

$$total\_manure\_excreted = fecal\_water + fecal\_solids + urine$$

[AN.EXC.21]

Organic matter intake (kg/d)

$$organic\_matter\_intake = dry\_matter\_intake - ash\_diet\_content$$

[AN.EXC.22]

Degradable volatile solids (kg/d)

$$degradable\_volatile\_solids = -1.017 + 0.364 * organic\_matter\_intake + 0.029 * neutral\_detergent\_fiber\_concentration - 0.023 * crude\_protein\_concentration$$

[AN.EXC.23]

Total volatile solids excreted by a lactating cow (kg/d)

$$total\_volatile\_solids = -1.201 + 0.402 * organic\_matter\_intake + 0.036 * neutral\_detergent\_fiber\_concentration - 0.024 * crude\_protein\_concentration$$

[AN.EXC.24]

Non-degradable volatile solids excretion (kg/d)

$$non\_degradable\_volatile\_solids = total\_volatile\_solids - degradable\_volatile\_solids$$

[AN.EXC.6]

Manure nutrient composition Total manure nitrogen (kg/d)

$$manure\_nitrogen = (20.3 + 0.654 * \frac{(dry\_matter\_intake*1000)*\frac{\frac{crude\_protein\_concentration}{6.25}}{100})}{1000}$$

[AN.EXC.25]

Fecal nitrogen (kg/d)

$$fecal\_nitrogen = \frac{(10.1*dry\_matter\_intake-18.5)}{1000}$$

[AN.EXC.26]

Urine N excretion (kg/d)

$$\text{urine}_{\text{nitrogen}} = \text{manure}_{\text{nitrogen}} - \text{fecal}_{\text{nitrogen}}$$

[AN.EXC.16]

Manure total ammoniacal nitrogen (kg/d)

$$manure\_total\_ammoniacal\_nitrogen = urine\_nitrogen$$

[AN.EXC.9]

Manure K excretion (g/d)

$$Potassium = 7.21 * dry\_matter\_intake + 15944 * \frac{potassium\_concentration}{100} - 164.5$$

[AN.EXC.27]

- **Dry Cow Manure Excretion (_calculate_dry_cow_manure)**

  Manure excretions Urine excretion (kg/d): Amount of urine excreted in kg with assumption that 1.038 kg urine is approximately 1 L. Due to lack of information, average excretion rate from dry cows is assumed.

$$urine = 15.4$$

[AN.EXC.28]

Total manure excretion (kg/d): Amount of feces and urine excreted daily by dry cows (kg/d).

$$total\_manure\_excreted = 0.00711 * body\_weight + 0.324 * crude\_protein\_concentration + 0.259 * neutral\_detergent\_fiber\_concentration + 8.05$$

[AN.EXC.29]

Because the equations that predict total manure excreted and total solids excreted rely on different inputs, in some extreme cases, the total mass of manure excreted can be predicted to be very close to the predicted total solids excreted which is unrealistically dry. For this reason, we set a maximum manure dry matter content of 20% for heifers and dry cows. If the mass predicted by equation AN.EXC.11 is not greater than (total solids/maximum manure dry matter content) then, $total\_manure\_excreted = \frac{total\_solids}{dry\_matter\_manure}$
Total solids excretion (kg/d) - this is the same equation used for dry cows.

$$total\_solids = 0.178 * dry\_matter\_intake + 2.733$$

[AN.EXC.12]

Organic matter intake (kg/d)

$$organic\_matter\_intake = dry\_matter\_intake - ash\_diet\_content$$

[AN.EXC.22]

Degradable volatile solids (kg/d)

$$degradable\_volatile\_solids = -1.017 + 0.364 * organic\_matter\_intake + 0.029 * neutral\_detergent\_fiber\_concentration - 0.023 * crude\_protein\_concentration$$

[AN.EXC.23]

Total volatile solids excreted by a lactating cow (kg/d)

$$total\_volatile\_solids = -1.201 + 0.402 * organic\_matter\_intake + 0.036 * neutral\_detergent\_fiber\_concentration - 0.024 * crude\_protein\_concentration$$

[AN.EXC.24]

Non-degradable volatile solids excretion (kg/d)

$$non\_degradable\_volatile\_solids = total\_volatile\_solids - degradable\_volatile\_solids$$

[AN.EXC.6]

- **Manure Phosphorus Excretion (_calculate_phosphorus_excretion_manure)**
  Total phosphorus fraction of feces

$$manure\_phosphorus\_fraction = \frac{fecal\_phosphorus + urine\_phosphorus\_required}{total\_manure_excreted * 1000 g/kg}$$

[AN.EXC.30]

Inorganic phosphorus fraction

$$inorganic\_phosphorus\_fraction = 0.50 * manure\_phosphorus\_fraction$$

[AN.EXC.31]

Organic phosphorus fraction

$$organic\_phosphorus\_fraction = 0.05 * manure\_phosphorus\_fraction$$

[AN.EXC.32]

Milk phosphorus

$$phosphorus\_in\_milk = 0.0009 * daily\_milk\_production * 1000 g/kg$$

[AN.EXC.33]

Manure P excreted by a cow (g/d)

$$manure\_phosphorus\_excreted = fecal\_phosphorus + urine\_phosphorus\_required$$

[AN.EXC.34]

Total P excreted by a cow (g/d)

$$\begin{gathered} total\_phosphorus\_excreted = \\ phosphorus\_in\_milk + fecal\_phosphorus + urine\_phosphorus\_required \end{gathered}$$

[AN.EXC.35]

- **Manure Methane Potential** The methane potential of the manure volatile solids are set by the **get_manure_streams()** method based on the Animal Combination of the pen. If the Animal Combination is CALVES or GROWING, the methane potential of the manure is set to 0.17. If the Animal Combination is LAC_COW or CLOSE_UP, the methane potential is set to 0.24 (Hanson, Itle, and Edquist, 2024).

Important Note

All references may be found at the end of this document or are based on expert communication with V. Sempali from 2023 to 2024.

4.b Enteric Methane Emission

Introduction The Enteric Methane Calculator module calculates daily enteric methane emissions for calves, heifers, lactating cows, and dry cows. Methane emissions are estimated based on animal characteristics, dietary intake, and a variety of animal category specific methane models. The IPCC, 2006 model is available for all animal classes except calves. The equation from Mills et al., 2003 is also available for dry and lactating cows and the Niu, E. Kebreab, Hristov, et al., 2018 equation is available for lactating cows. Finally, a separate equation published by Pattanaik et al., 2003 is used for calves. Simulation of enteric methane mitigation via mitigation supplements 3-NOP or monensin are also available for lactating cows The module integrates with the DigestiveSystem class and supports methane mitigation strategies using feed additives.

This module includes:

- **Classes:** EntericMethaneCalculator
- **Key Methods:**
  - calculate_calf_methane
  - calculate_heifer_methane
  - calculate_cow_methane
  - The lactating and dry cow methods are called within the calculate_cow_methane method
  - _calculate_lactating_cow_manure
  - _calculate_dry_cow_manure

The calculated enteric methane emissions are directly output to the Output Manager to generate the final reports.

Required User Inputs General Inputs

Table 43: These inputs define basic animal and diet properties used to calculate nutrient intake and manure production.

Variable	Definition	Units
dry_matter_intake	Dry matter intake from pen ration	kg
nutrient_concentrations	Nutrient concentrations (e.g., CP, NDF, ADF, ash)	%

Cow Specific Inputs

Table 44: These inputs are used when modeling cows and are particularly relevant for lactating animals.

Variable Name	Definition	Units	Default	Max
body_weight	Body weight of the current animal	kg	-	-

Continued on next page

Table continued from previous page

Variable Name	Definition	Units	Default	Max
is_lactating	Indicates the cow's lactating status	Boolean	-	-
milk_fat	Milk fat content	%	3.5	-
metabolizable_energy_intake	Metabolizable energy intake per kg DM	Mcal/kg DM	-	-

Methane Model-Specific and Mitigation Inputs

Table 45: Inputs specific to the methane emissions model and selected mitigation strategies.

Variable Name	Definition	Default and Range
methane_model	Selected methane model for emission estimation	"Mutian", "Mills", "IPCC"
methane_mitigation_method	Selected feed additive for methane mitigation	"3-NOP", "Monensin", "Essential Oils", "Seaweed"
methane_mitigation_additive_amount	Amount of additive used for methane mitigation (mg/kg DMI)	3-NOP (70; 40–100), Monensin (24; 20–36), Essential Oils (0; 0–100), Seaweed (0; 0–100)

Expected Outputs

The module predicts the daily enteric methane emissions for individual animals:

Table 46: Expected outputs from this section.

Category	Parameter	Definition	Units
Methane Emissions	Daily enteric methane emissions	Estimated methane production per animal per day	g CH_4/head/day

Methodology

- **Required Diet Composition Inputs**

  Based on dry matter intake and gross energy concentration from nutrient composition:

$$soluble\_residue = 100 - ash\_concentration - neutral\_detergent\_fiber\_concentration - crude\_protein\_concentration - ethyl\_ester\_concentration$$

[AN.MET.1]

$$gross\_energy\_concentration = 0.263 * crude\_protein\_concentration + 0.522 * ethyl\_ester\_concentration + 0.198 * neutral\_detergent\_fiber\_concentration + 0.160 * soluble\_residue$$

[AN.MET.2]

$$starch\_to\_acid\_detergent\_fiber\_concentration\_ratio = -0.0011 * \frac{starch\_concentration}{acid\_detergent\_fiber\_concentrations}$$

[AN.MET.3]

- **Calf Methane Emissions (calculate_calf_methane)** Methane emissions are calculated using body weight. The selected methane model for other animal categories does not apply here. Pattanaik et al., 2003 reported no difference in calf methane production in crossbred calves. Methane emissions by calves are estimated from mean values observed in that study.

$$methane\_emission = \frac{0.013 * body\_weight^{0.75} * 4.184 MJ/Mcal}{0.05565}$$

[AN.MET.4]

- **Heifer Methane Emissions (calculate_heifer_methane)**
  Methane Emission

$$methane\_emission = \frac{(0.065 * gross\_energy\_concentrationdry\_matter\_intake)}{0.05565}$$

[AN.MET.5]

- **Lactating Cow Methane Emissions (calculate_cow_methane)**
  For lactating cows, the Mutian, Mills, or IPCC model is applied.

  Mutian Model

$$methane\_emission = -126 + 11.3 * dry\_matter\_intake + 2.30 * neutral\_detergent\_fiber\_concentration + 28.8 * milk\_fat + 0.148 * body\_weight$$

[AN.MET.6]

Mills Model Mills et al., 2003 parameterized Mitscherlich equations for a nonlinear model of methane emissions. The CNCPS and IFSM models use the Mitscherlich 3 equation.

$$methane\_emission = \frac{45.98*(1-exp(-[(starch\_to\_acid\_detergent\_fiber\_concentration\_ratio)+0.0045]*metabolizable\_energy\_intake*4.184))}{0.05565}$$

[AN.MET.7]

IPCC Model

$$methane\_emission = \frac{(0.065*gross\_energy\_concentrationdry\_matter\_intake)}{0.05565}$$

[AN.MET.5]

- **Dry Cow Methane Emissions (_calculate_dry_cow_manure)**

  Mills Model Mills et al., 2003 parameterized Mitscherlich equations for a nonlinear model of methane emissions. The CNCPS and IFSM models use the Mitscherlich 3 equation.

$$methane\_emission = \frac{45.98*(1-exp(-[(starch\_to\_acid\_detergent\_fiber\_concentration\_ratio)+0.0045]*metabolizable\_energy\_intake*4.184))}{0.05565}$$

[AN.MET.7]

IPCC Model

$$methane\_emission = \frac{(0.065*gross\_energy\_concentrationdry\_matter\_intake)}{0.05565}$$

[AN.MET.5]

- **Methane Mitigation**
  - Methane mitigation strategies are applied exclusively to lactating and dry cows. However, caution is advised when interpreting the outcomes for dry cows, as the equations used were developed based on datasets from lactating cows.
  - Currently, the implemented methane mitigation feed additives include 3-NOP and monensin, with essential oils and seaweed planned for future integration.
  - The detailed methodology for methane mitigation calculations will be provided in a separate document titled Methane Mitigation Calculator.

4.c Enteric Methane Mitigation

Introduction The Methane Mitigation Calculator module evaluates the reduction in enteric methane yield due to the inclusion of specific feed additives in animal diets. This module calculates methane mitigation effects on a per-animal basis by considering dietary composition, feed additive dosage, and the selected mitigation strategy. Currently, the module supports the following mitigation strategies:

- 3-NOP (3-Nitrooxypropanol)
- Monensin

Future updates will include additional strategies such as essential oils and seaweed. The mitigation effects are first calculated as a percentage reduction in methane yield (methane production / DMI). Then this module integrates with the Enteric Methane Calculator to adjust the initially predicted enteric methane.

Required User Inputs

The Methane Mitigation Calculator module evaluates the reduction in enteric methane yield based on dietary composition and the use of specific feed additives. The inputs are organized into the following groups:

Dietary Composition

Table 47: General dietary fiber and fat concentration inputs.

Variable Name	Definition	Default and Range
neutral_detergent_fiber_concentration	Neutral detergent fiber concentration, % DM	-
ethyl_ester_concentration	Ether extract (crude fat) concentration, % DM	-
starch_concentration	Starch concentration in the diet, % DM	-

Table 48: Feed-based methane mitigation strategies.

Variable Name	Definition	Defaults and Units	Min	Max
methane_mitigation_method	Selected feed additive for methane mitigation	“3-NOP”, “Monensin”	-	-

Table 49: Dosage of methane mitigation additives.

Variable name	Definition	Default and range
methane_mitiga-tion_additive_-amount	Dosage per kg of dry matter intake (DMI).	3-NOP: 70 (40–100) mg/kg DMI; Monensin: 24 (20–36) mg/kg DMI

Table 50: Expected outputs from this section.

Parameter	Definition	Units
Methane Yield Reduction	Change in methane production per unit DMI; negative indicates reduction	%

Methodology

- **3-NOP Methane Yield Reduction**

  Reduction is modeled as a function of additive dosage, NDF, EE, and starch concentrations.

$$methane\_yield\_reduction = -30.8 - 0.226 * (methane\_mitigation\_additive\_amount - 70.5) + 0.906 * neutral\_detergent\_fiber\_concentration - 32.9 + 3.871 * ethyl\_ester\_concentration - 4.2 - 0.337 * (starch\_concentration - 21.1)$$

[AN.MET.8]

- **Monensin Methane Yield Reduction** Reduction is based on additive dosage and starch concentration.

$$methane\_yield\_reduction = 6.35 - 0.277 methane\_mitigation\_additive\_amount - 0.182 * starch\_concentration$$

[AN.MET.9]

- **Default Case** If no valid mitigation method is selected, the reduction defaults to 0%.
- **Adjust the Initial Predicted Enteric Methane (EntericMethaneCalculator.calculate_-cow_methane)**

  First, the initially predicted methane yield of the basal diet is calculated

$$methane\_yield = \frac{methane\_emission}{dry\_matter_intake} * 100$$

[AN.MET.10]

Then the adjusted of methane production is calculated based on methane yield reduction (%)

$$adjusted\_methane\_emission = \\ methane\_yield * (1 + \frac{methane\_yield\_reduction}{100}) * dry\_matter\_intake$$

[AN.MET.11]

5 References

References

Appuhamy, J.A.D.R.N., L.E. Moraes, C. Wagner-Riddle, D.P. Casper, J. France, et al. (2014). "Development of mathematical models to predict volume and nutrient composition of fresh manure from lactating Holstein cows". In: *Animal Production Science* 54.12.

Appuhamy, J.A.D.R.N., L.E. Moraes, C. Wagner-Riddle, D.P. Casper, and E. Kebreab (2018). "Predicting manure volatile solid output of lactating dairy cows". In: *Journal of Dairy Science* 101.1, pp. 820–829.

ASABE (2005). *Manure Production and Characteristics.* St. Joseph, MN: ASABE, pp. 1–32.

Boadi, D. et al. (2004). "Mitigation strategies to reduce enteric methane emissions from dairy cows: Update review". In: *Canadian Journal of Animal Science* 84.3, pp. 319–335. DOI: 10.4141/A03-109.

Boer, I.J. de et al. (2002). "Prediction of ammonia emission from dairy barns using feed characteristics part 1: Relation between feed characteristics and urinary urea concentration". In: *Journal of Dairy Science* 85.12, pp. 3382–3388. DOI: 10.3168/jds.S0022-0302(02)74425-1.

Dairy Cattle Reproduction Council (2018). *Reproductive Management Strategies for Dairy Cows and Heifers.* Accessed: 2023-06-17. URL: https://www.dcrcouncil.org/protocols/.

De Vries, A. et al. (2006). "Economic comparison of timed artificial insemination and exogenous progesterone as treatments for ovar- ian cysts." In: *J Dairy Sci* 89.8, pp. 3028–3037.

Galvão, K. N. et al. (2013). "Economic comparison of reproductive programs for dairy herds using estrus detection, timed artificial insemination, or a combination". In: *J Dairy Sci* 96.4, pp. 2681–2693.

Hanson, W. L., C. Itle, and K. Edquist (2024). *Quantifying Greenhouse Gas Fluxes in Agriculture and Forestry: Methods for Entity-Scale Inventory.* Forthcoming or gray literature if no publisher listed. USDA or relevant institution.

IPCC (2006). *Guidelines for National Greenhouse Gas Inventories.* https://www.ipcc-nggip.iges.or.jp/public/2006gl/. Accessed 2025-04-17.

Johnson, A.C.B., K.F. Reed, and E. Kebreab (2016). "Short communication: Evaluation of nitrogen excretion equations from cattle". In: *Journal of Dairy Science* 99.9, pp. 7669–7678. DOI: 10.3168/jds.2015-10730.

Kalantari, Afshin Samia (2015). *Using mathematical modeling techniques for optimized dairy herd management and decision making.* The University of Wisconsin-Madison.

Kebreab, Ermias et al. (2023). "A meta-analysis of effects of 3-nitrooxypropanol on methane production, yield, and intensity in dairy cattle". In: *Journal of Dairy Science* 106.2, pp. 927–936. DOI: 10.3168/jds.2022-22211.

Korver, S., J.A.M. van Arendonk, and W.J. Koops (1985). "A function for live-weight change between two calvings in dairy cattle". In: *Anim Sci* 40.2, pp. 233–241.

Li, M., K.F. Reed, et al. (2023). "A stochastic animal life cycle simulation model for a whole dairy farm system model: Assessing the value of combined heifer and lactating dairy cow reproductive management programs". In: *J Dairy Sci* 106.5, pp. 3246–3267.

Li, M., G.J.M. Rosa, et al. (2022). "Investigating the effect of temporal, geographic, and management factors on US Holstein lactation curve parameters". In: *J Dairy Sci* 105.9, pp. 7525–7538.

Mills, J.A.N. et al. (2003). "Alternative approaches to predicting methane emissions from dairy cows". In: *Journal of Animal Science* 81.12, pp. 3141–3150. DOI: 10.2527/2003.81123141x.

Monteny, G.J. et al. (2002). "Prediction of ammonia emission from dairy barns using feed characteristics part II: Relation between urinary urea concentration and ammonia emission". In: *Journal of Dairy Science* 85.12, pp. 3389–3394. DOI: 10.3168/jds.S0022-0302(02)74426-3.

National Academies of Sciences, Engineering, and Medicine (2021). *Nutrient Requirements of Dairy Cattle: Eighth Revised Edition.* Washington, DC: The National Academies Press.

National Research Council (2001). *Nutrient Requirements of Dairy Cattle.* Washington, D.C.: National Academies Press.

Nennich, T.D. et al. (2005). "Prediction of manure and nutrient excretion from dairy cattle". In: *Journal of Dairy Science* 88.10, pp. 3721–3733. DOI: 10.3168/jds.S0022-0302(05)73058-7.

Niu, M., E. Kebreab, A.N. Hristov, et al. (2018). “Prediction of enteric methane production, yield, and intensity in dairy cattle using an intercontinental database”. In: *Global Change Biology* 24, pp. 3368–3389. DOI: 10.1111/gcb.14094.

Pattanaik, A.K. et al. (2003). “Influence of grain processing and dietary protein degradability on nitrogen metabolism, energy balance and methane production in young calves”. In: *Asian-Australasian Journal of Animal Sciences* 16.10, pp. 1443–1450. DOI: 10.5713/ajas.2003.1443.

Reed, K.F. et al. (2015). “Predicting nitrogen excretion from cattle”. In: *Journal of Dairy Science* 98.5, pp. 3025–3035.

Reuven, Y.R. and D.P. Kroese (2017). *Simulation and the Monte Carlo Method. In: Wiley Series in Probability and Statistics.* John Wiley and Sons Inc.

Vadas, P.A. et al. (2007). “A model for phosphorus transformation and runoff loss for surface-applied manures”. In: *Journal of Environmental Quality* 36.1, pp. 324–332.

Wilkerson, V. A., D. R. Mertens, and D. P. Casper (1997). “Prediction of excretion of manure and nitrogen by Holstein dairy cattle”. In: *Journal of Dairy Science* 80.12, pp. 3193–3204.

Wiltbank, M.C. et al. (2016). “Pivotal periods for pregnancy loss during the first trimester of gestation in lactating dairy cows”. In: *Theriogenology* 86.1, pp. 239–253.

Wood, P. D. P. (1967). “Algebraic Model of the Lactation Curve in Cattle”. In: *Nature* 216.5111, pp. 164–165. DOI: 10.1038/216164a0.

Part III
Manure Module

Contents

List of Figures

Contents

List of Tables

Contents

6 Module Introduction

The responsibility of the manure module in RuFaS is to simulate the loss and/or gain of manure mass and nutrients on a daily timestep at each step of the manure management chain on a dairy farm. On the majority of farms, manure represents an important link in the cycling of nutrients through animals, land, crops, and back to animals. Therefore, modeling both greenhouse gas (GHG) emissions as well as non-GHG nutrient and losses is crucial in capturing nutrient flows through the whole-farm system.
The manure module accomplishes its responsibilities by tracking a critical, core set of variables called "ManureStream" variables, which include key agronomic nutrients (N, P, K), carbon (e.g. total and volatile solids), water, ash, mass, and volume. The Manure Module receives manure excretion and bedding information from the animal module, then passes manure through the manure management chain, until manure is either applied to fields or exported. The specific steps of the manure management chain are user-defined, and the individual steps/options in manure management chains are referred to as "processors". The exact chemical, physical, or other processes that occur at each step along the management chain depend on what type of processor the manure is being held in. For example, parlor cleaning processors, which represent removing manure from the milking parlor and holding areas, principally add water to manure but do not estimate GH emissions or other nutrient losses. A slurry storage outdoor processor (a type of manure storage), however, estimates daily methane (CH_4) and ammonia (NH_3-N) losses, which are then reflected by reducing the quantity of nutrients remaining in the stored manure at the end of the day.

6.a Structural setup of the Manure Module

Processors in the Manure Module fall into four basic classes, with multiple types available within each class. The processor classes are intended to represent the common, primary steps in manure management. They include manure handling (which refers to the daily activities of cleaning and removing manure from facilities or managing it in place), storage, and manure treatment, such as anaerobic digestion or mechanical solid liquid separation.

Table 51: Examples of processor types by class.

Classes	Types
Handler	Manual Scraping; Alley Scraper; Flush System; Parlor Cleaning
Digester	Continuous Mix
Separator	Screw Press; Rotary Screen
Storage	Composting; Open Lot; Bedded Pack; Slurry Storage Outdoor; Slurry Storage Underfloor; Anaerobic Lagoon

How manure moves between processors is based on defining the destination processor that any one processor should send its manure to, and the proportion(s) of manure from the originating processor that should go there. For example, if 50% of manure from a single pen is routed to storage A, and 50% to storage B, the manure handler processor assigned to that pen will have two destinations defined (storage A and storage B), with the proportion going to each defined as 0.50. By controlling destinations and proportions, the user is able to define aggregating (i.e., assigning manure from two separate processors to the same destination) and splitting (assigning manure from one processor to two or more locations) behavior. A visual example is below and illustrates the manure management chain without RuFaS-specific syntax. The next figure illustrates RuFaS-specific information (e.g. specific processor names, creation of manure streams in the animal module, etc.). Note that the splitting of manure generated from singular pens in the Animal Module illustrated in Figure 8 is covered in the Animal to Manure Connection document.

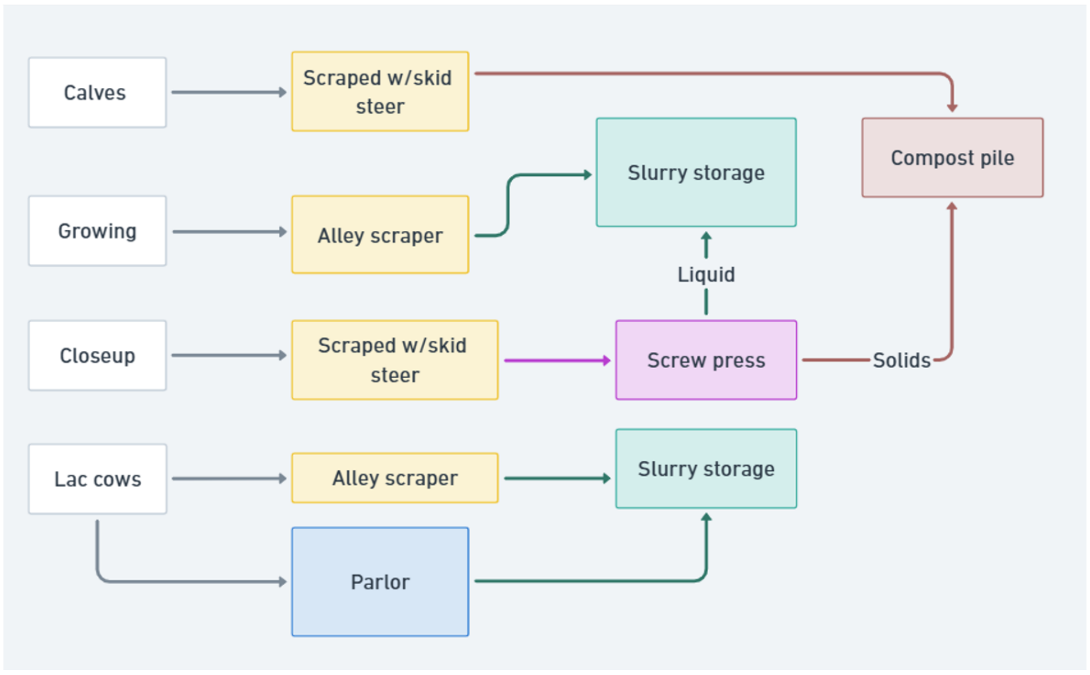


Figure 5: A real-world example of a simplistic manure management chain.

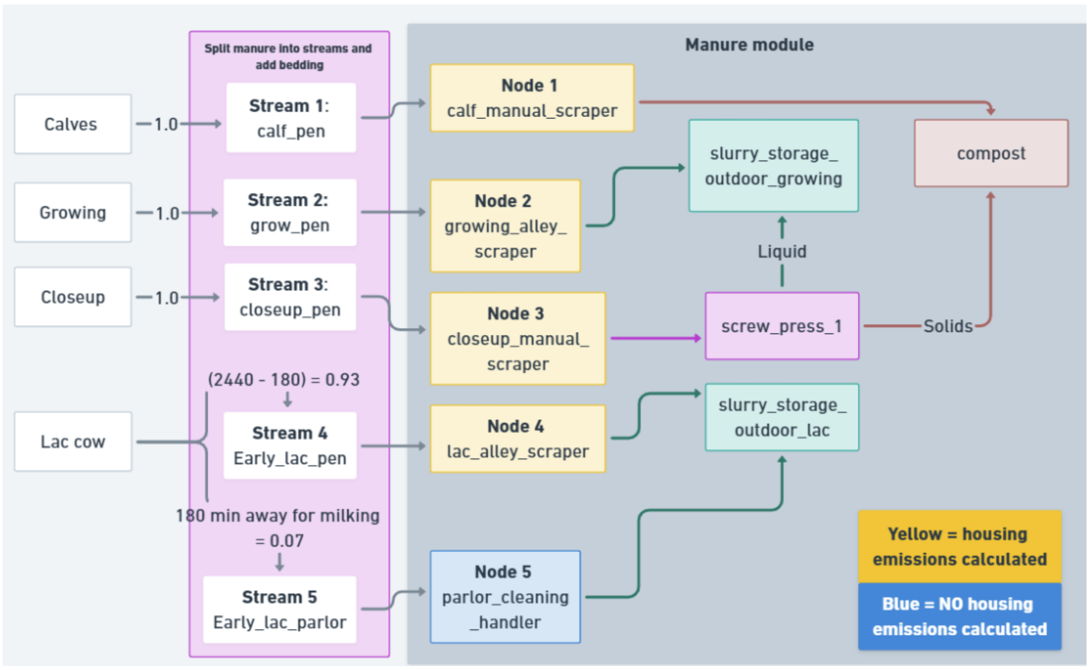

Split manure into streams and add bedding
Manure module
Calves
1.0
Stream 1: calf_pen
Node 1 calf_manual_scraper
compost
Growing
1.0
Stream 2: grow_pen
Node 2 growing_alley_scraper
slurry_storage_outdoor_growing
Liquid
Closeup
1.0
Stream 3: closeup_pen
Node 3 closeup_manual_scraper
screw_press_1
Solids
(2440 - 180) = 0.93
Lac cow
Stream 4 Early_lac_pen
Node 4 lac_alley_scraper
slurry_storage_outdoor_lac
180 min away for milking = 0.07
Stream 5 Early_lac_parlor
Node 5 parlor_cleaning_handler
Yellow = housing emissions calculated
Blue = NO housing emissions calculated

Figure 6: The manure management chain illustrated in the previous figure, with RuFaS-specific information reflected.

6.b Overview of Required Inputs

There are two general categories of input required from the user:

1. Which processors are used and how do they work (processor-specific configurations) E.g., storage time length, use of a cover
2. Destination and proportion(s) allocated to destination(s) of manure passed from each processor

Information on inputs required and options available for each individual processor are outlined in the individual processor documentation.

For defining destinations and proportions, there are very few rules and restrictions on how manure can be moved between processors. The primary rule for defining processor destinations is that users cannot define loops, i.e., manure from a processor "downstream" in a manure management chain cannot return its manure to a processor "upstream".

The assignment of manure generated in the Animal Module (which also reflects the quantity of bedding used) to its destination in the Manure Module occurs in the Animal Module. See the Animal to Manure Connection section for more information.

6.c General Assumptions of the Manure Module

1. All manure is recovered unless otherwise specifically stated. For example, all manure excreted by animals is assumed to be captured by manure handlers, all manure in a manure storage is assumed to be completely removed when the storage is emptied (unless explicitly noted in the specific processor's documentation), etc.

2. Manure moves through the manure management chain once, and only once, per day. With this, values are reported on a daily timestep, and represent the current state at the end of the given day (e.g. CH_4 emissions from this day, amount of N left in storage on this day, etc.).

   - Storage processors include a user-defined "storage time", during which manure accumulates in the storage until reaching the end of the time interval, at which point it is passed to the next processor or exported, if the storage comes last in the manure chain. The minimum storage time value is 1, thus manure cannot move through more than one storage per day.
   - For processors without a storage time option, manure is assumed to move through the processor immediately. E.g., on a single day, excreted manure may move through a handler, continuous mix digester, separator, and into composting storage, with each processor reporting the quantity and composition of manure they either passed to the next processor or are holding in storage.

3. The Manure Module adheres to the principle of mass balance, meaning, the quantity of manure nutrients remaining in manure is proportional to the quantity of the nutrient lost through biological or physical processes. The same is true for additions of mass/nutrients.

   - One current exception to this is N loss. N losses are not currently reflected in the total mass of manure; this exception will be addressed and corrected in the near future.

Manure Storage Assumptions

1. Received manure nutrients entering a storage are added to accumulated mass/nutrient quantities prior to calculating gas emissions each day.

2. When the end of the storage's user defined storage time is reached, the accumulated manure in storage is removed completely and entirely, and is passed to the next processor in the chain (e.g., a subsequent storage, field application, etc.).

3. Emptying intervals (in days) are defined by the user in equal lengths at this time, e.g., manure storage can be emptied every 3 months (e.g. in April, July, October, January) but cannot be emptied in April, then 2 months later in June, and then 3 months later in September. However, timing of storage emptying (e.g. April and October emptying vs. May and November emptying) can be set according to the general simulation dates to simulate more realistic manure removal behavior.

7 Animal and Manure Module Connection

7.a Introduction

Though animals are typically fed and managed in groups, referred to as pens in RuFaS, manure from a single group of animals may not always remain in or be transported to the same location on the farm. For example, lactating cows who must travel to a milking parlor deposit some proportion of their manure outside of their pen. Or, in open lot systems, manure deposited in the concrete feed alley apron is often removed and managed separately from manure deposited on the lot surface. Additionally, more than one type of bedding may be used in a pen (e.g., no bedding during summer but straw provided in winter), which also eventually becomes part of manure. Considering this, RuFaS accommodates routing of manure from a single group (pen) of animals to multiple locations, which are then received by the Manure module. This document describes how a pen's total manure excretion and bedding are allocated to one or more discrete quantities of manure, referred to as manure streams, and passed to the Manure module. How the user configures this manure routing and bedding options and addition of bedding to manure are also described here.

Implementation in RuFaS

The Animal module is responsible for calculating manure excretion mass and composition based on animal attributes, ration composition and intake, and other factors. The Animal module must then pass this manure on to the Manure module, where manure nutrient gains and losses are quantified through field application or export. The Animal module currently calculates manure excretion at the pen level (i.e., values representative of the total manure from animals in a single pen), and several additional steps must occur before manure information can be sent to the Manure module. These steps include:

- Splitting manure from individual pens into one or more user-defined manure streams
- Repackaging manure information into the form and units utilized by the Manure module
- Applying bedding mass or nutrients to manure streams

The direct connection between Animal module, which captures the animal lifecycle, nutrient intake and excretion, and other factors, and the Manure module is a crucial part of the nutrient cycle in RuFaS. This document describes how the user defines the partitioning of manure from a single pen of animals into one or more manure streams, and the basic mathematical procedure for generating manure streams from excreted manure. Later it will review the options for and application of bedding to manure streams.

Classes

Table 52: List of Python classes for Animal to Manure Module connection.

Handlers	Description
Pen	pen.py; Translates pen-level manure information into the format and units required by the Manure module; Translates pen-level manure information into one or more ManureStream instance(s), if applicable, and creates accompanying PenManureData instances
PenManureData	animal_to_manure_connection.py; Generates PenManureData and ManureStream

Required User Inputs

Table 53: User inputs for Stream Formation and Bedding sections.

	Variable	Units	Description
Stream Formation	minutes_away_for_milking	minutes	The total time (minutes) per day that animals spend walking to/from the parlor, waiting in holding areas, or being actively milked. For non-lactating animals, this value may be omitted or entered as "null".
	first_parlor_processor	–	The name of the processor defined in the Manure module inputs that manure from the milking parlor should enter.
	parlor_stream_name	–	The user-defined name of this pen's parlor stream.
	stream_name	–	The user-defined name of this stream.
	bedding_name	–	The name of the specific bedding configuration used with this stream. See section 2 for more information.
	stream_proportion	0–1	The proportion of daily manure excreted outside of the parlor and while traveling to/from the parlor that enters this stream. All stream proportions within a pen must sum to 1.0. Optional if only one stream.
	first_processor	–	The name of the processor defined in the Manure module inputs that this ManureStream should enter.
Bedding	name	–	Unique identifier to reference this bedding configuration.
	bedding_type	–	The type of bedding material. Options: sand, straw, sawdust, manure solids, or none.
	bedding_mass_per_day	kg/day	Daily mass of fresh bedding added per animal per day (wet weight). Bedding mass is applied based on the total number of animals in the pen, and is not based on the stream proportion of the manure stream the bedding is assigned to.
	bedding_density	kg/m^3	Density of the bedding on a wet weight basis.
	bedding_dry_matter_content	fraction	Bedding dry matter content as a fraction of total mass.
	bedding_carbon_fraction	fraction	Bedding carbon content as a fraction of total mass.
	bedding_phosphorus_content	fraction	Bedding phosphorus content as a fraction of total mass.
	sand_removal_efficiency	fraction	Fraction of sand assumed to be removed immediately after manure removal.

Other inputs

- Instances of PenManure objects that represent manure excreted by single pens of animals in the Animal module. See Animal Excretions documentation for further details.

- Instances of Pen objects that represent pen information for single pens of animals in the Animal module.

Expected Outputs

Stream Formation

- Instance(s) of a ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane potential, $m^3/kgVS$):
  - water
  - ammoniacal_nitrogen
  - nitrogen
  - phosphorus
  - potassium
  - ash
  - manure_degradable_volatile_solids
  - manure_non_degradable_volatile_solids
  - bedding_non_degradable_volatile_solids
  - total_solids
  - mass (equal to sum of water and total solids)
  - total volatile solids (equal to sum of degradable and non-degradable volatile solids)
  - volume
  - methane_production_potential
- A PenManureData class instance for each manure stream defined by the user, that represents the attributes of the animals and pen from which the manure originated. PenManureData instances include the following variables, reported by the Animal module:
  - num_animals: number of animals present in the pen from which the manure stream originated. Note that this variable does not correspond to the number of animals that generated the manure contained in the stream.
  - manure_deposition_surface_area: the surface area soiled with manure, used in the determination of housing emissions. See the Manure Handler document for information on determination of this value (MN.HDL.4).
  - animal_combination: the type of animal contained in the pen (calf, growing, closeup, or lactating).
  - pen_type: the type of pen (freestall, tiestall, open lot, or bedded pack barn) the manure in this stream originated from.
  - manure_urine_mass (kg): mass of urine in the manure.
  - manure_urine_nitrogen (kg): mass of urine N in the manure.
  - steam_type: the type of ManureStream (parlor stream or general stream).
  - first_processor: the name of the processor defined in the Manure module inputs that this ManureStream should enter.
  - total_bedding_mass (kg): the total mass of bedding added to this stream.
  - total_bedding_volume (m^3): the total volume of bedding added to this stream.

Bedding

- total_bedding_mass: The total mass of bedding added to the ManureStream each day (kg).
- total_bedding_volume: The total mass of bedding added to the ManureStream each day (kg).

7.b Methodology

Stream Formation

How many discrete locations/quantities (i.e., manure streams) a single pen's manure is split into is determined by the user. There are two types of manure streams in the model:

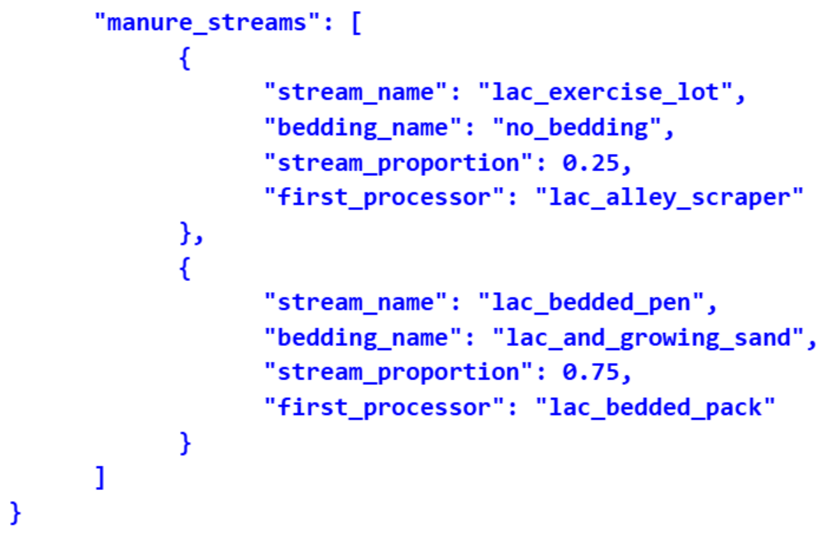
Parlor: A stream representing manure deposited in or while traveling to/from the milking parlor, along with (fresh, non-recycled) wash water.

- A parlor stream is not directly defined by the user for each pen in the way general streams are; parlor streams are created automatically for each lactating pen. The user specifies the proportion of each lactating pen's total manure that enters the parlor stream via the minutes away for milking input. Time away from the pen for milking can vary greatly between farms but also is typically known by the farm management, as fetching and pushing cows to the parlor is an intentional daily activity. This makes time away from the pen for milking (minutes_away_for_milking) a simple and relatively reliable metric for determining the basic split of manure in the housing area vs. milking parlor.
- Only one parlor stream can be created per lactating cow pen. However, more than one milking parlor (destination for milking parlor manure) may be defined for a farm; the user simply assigns parlor streams to differing instances of a parlor cleaning handler, which represents the milking parlor. See Manure Handler documentation for more information.
- No bedding can be applied to parlor streams.

General: A stream representing all or a portion of non-parlor manure and bedding. Manure in this stream or streams represents manure excreted outside the milking parlor or traveling to/from the parlor.

- All pens must have at least one general stream defined.
- Theoretically, an infinite number of general streams can be defined per pen, as long as stream proportions are provided for all of them. However, most practical farm simulation scenarios will require only one or two general manure streams per pen.
- The number of general streams and the proportion of non-parlor manure entering each stream is directly defined by the user. Stream proportion logic is covered below.

General-type manure streams are defined in the animal inputs, within each Pen blob. Each pen includes a "manure_streams" array input in which the user defines the one or more general manure streams generated by this specific pen. The required inputs and their definitions are described above in the Inputs section; these inputs must be provided for each manure stream as outlined in the figure below.



```
                "manure_streams": [
                    {
                        "stream_name": "lac_exercise_lot",
                        "bedding_name": "no_bedding",
                        "stream_proportion": 0.25,
                        "first_processor": "lac_alley_scraper"
                    },
                    {
                        "stream_name": "lac_bedded_pen",
                        "bedding_name": "lac_and_growing_sand",
                        "stream_proportion": 0.75,
                        "first_processor": "lac_bedded_pack"
                    }
                ]
            }
```

Figure 7: If manure from a pen is routed to two different general manure streams, two sets of manure stream inputs must be defined. An example of a single pen's manure stream inputs is above - in this example, manure from this pen is routed to two locations, one representing an exercise lot, the other representing a bedded pack area.

Stream proportion logic

The stream proportion refers to the user-defined proportion (0 to 1) of manure generated by animals in a single pen that enters a specific manure stream. The translation of user-defined stream proportions into the effective stream proportion utilized by the model is described below. This effective proportion is referred to as the split ratio in the model code - it is the value by which total manure excretions are multiplied by to determine the quantity of any single nutrient allocated to a stream of any type (e.g., 20% (0.20) to parlor stream, 40% (0.40) to general stream 1, 40% (0.40) to general stream 2. The values are the split ratios).

Non-lactating pens If more than one general stream is defined for a single pen, a stream proportion must be provided for each stream, and manure will be allocated to streams according to these proportions. Essentially, the split ratio is equal to the user-defined stream proportion.

Lactating pens

Parlor streams: The user-defined minutes away for milking is used to calculate the stream proportion value for the milking parlor:

When:

$$parlor\_stream\_proportion = \frac{minutes\_away\_for\_milking}{1440}$$

[AN.PEN.1]

- minutes_away_for_milking: user-defined total minutes per day lactating cows spend out of the pen traveling to/from the parlor or being milked.
- 1440: minutes per day.

General streams Each general stream proportion represents the fraction of manure—excluding that deposited in or during travel to/from the milking parlor—that enters a specific manure stream. For example, if all manure outside the parlor enters one stream, its proportion would be 1.0. If manure outside the parlor was split evenly between two general streams, each stream's proportion would be 0.50. With

this, the effective proportion of total manure entering this stream (the split ratio) must be calculated according to the parlor stream proportion.
First, we determine the proportion of manure entering any and all general streams vs. the parlor stream.

$$total\_general\_stream\_portion = 1 - parlor\_stream\_proportion$$

[AN.PEN.2]

When:

- parlor_stream_proportion: proportion of total manure allocated to the parlor stream, calculated in **[AN.PEN.1]**.

Once we have derived the proportion of manure entering any/all general streams, we multiply this proportion by the user-defined general stream proportion(s), to determine the split ratio for each general stream.

$$general\_stream\_proportion = total\_general\_stream\_proportion * stream\_proportion$$

[AN.PEN.3]

To continue with our example from above, if cows spent 200 minutes per day in the milking parlor, their split ratios would be as follows:

Parlor:

$\frac{200}{1440} = 0.140$
$lac\_exercise\_lot : (1 - 0.139) * 0.25 = 0.215$
$lac\_bedded\_pen : (1 - 0.139) * 0.75 = 0.645$

Recommended defaults

Although the time away from a pen is highly variable and specific to each farm's infrastructure and management, some recommended defaults for situations where actual information is not available are provided below.

Minutes away for milking

Recommended minutes_away_for_milking values for lactating cows are based on cow time budgets from published literature and are not official model defaults. The purpose of these time budgets at the time of writing is strictly related to estimating manure deposition in specific locations and is not related to larger Animal module activities. The use of time budgets for this purpose relies on the assumption that cows excrete urine and feces at a constant rate throughout the day and across various locations.

Open lot and bedded pack systems

The management of open lot and bedded pack barn pens often results in manure from a single pen being managed in multiple ways. These types of pens often have designated resting areas (i.e. the harrowed lot or bedded pack) where manure is managed in place for an extended period, as well as a separate feeding area, such as a concrete apron along a feed alley. Manure deposited in this area may be scraped back into the resting area, or it may be flushed, vacuumed or scraped to another separate storage area. Two distinct manure management routines within a single pen indicate that two manure streams should likely be used. If this is the case, the user may reference the tables below as a starting point for defining stream proportions representing the resting vs. feeding area. Defining two separate manure streams is recommended where appropriate to ensure that manure is not over-allocated to the parlor manure stream (where it does not contribute to housing emission estimations) and bedding is allocated appropriately.
The following streams and proportions are based on literature data and are not official model defaults. They are provided for user informational purposes; they may be used as a starting point to adjust proportions specific to a farm, or can be used for theoretical modeling scenarios.

Passing information to the Manure module

Once split ratios have been determined for each manure stream in each pen, Animal module manure and pen data can be translated into ManureStream objects. The variables contained in ManureStream are below; this dataclass is used both to move information from Animal to Manure module, as well as move information between processors within the Manure module.
The majority of the variables are calculated in the Animal module and translated directly into a ManureStream object. Subbullets below a variable indicate how variables not directly reported by the Animal module are calculated.

Bedding

Bedding is assigned and applied by manure stream, not by pen. The following calculations described how bedding mass, nutrients, and volume are calculated, and how bedding is added to a manure stream.

Calculate total bedding mass

calc_total_bedding_mass

$$total\_bedding\_mass(kg) = \\ bedding\_mass\_per\_day(kg/animal) * num\_animals * sand\_removal\_efficiency$$

[AN.PEN.7]

When:

- bedding_mass_per_day: The user-inputted bedding mass applied per animal per day. Note that this input is per total number of animals housed in the pen.
- num_animals: Animals in the pen from which the manure stream originated.
- sand_removal_efficiency: User-defined proportion of sand bedding removed via sand settling lane or other sand recovery technology. This value is only utilized if bedding type $\bar{s}$and.

Calculate total bedding volume

calc_total_bedding_volume

$$total\_bedding\_volume(m^3) = \frac{total\_bedding\_mass(kg)}{bedding\_density(kg/m^3)}$$

[AN.PEN.8]

When:

Table 54: Recommended minutes_away_for_milking by pen type based on published literature. Remember, this does not necessarily reflect the model defaults.

Pen Type	minutes_away_for_milking	Proportion	Special Notes
Freestall[1]	200 m (3.33 h)	0.139	
Tiestall	200 m (3.33 h)	0.139	Limited data; value is based on the assumption that daily time away from pen for milking will be broadly similar between freestall and tiestall barns.[2]
Open lot	300 m (5 h)	0.208	Assumption that open lot pens tend to be considerably larger than indoor-housed systems (greater travel distance).[3]
Compost bedded pack barn	200 m (3.33 h)	0.139	Limited data; value is based on the assumption that daily time away from pen for milking will be broadly similar between freestall and tiestall barns.[2]

[1] **gomez2010time**, **espejo2007herd**

[2] **jewell2019prevalence**

[3] **meyer2019contract**

Table 55: Proportions of differnt streams for different animal classes.

Pen Type	Animal Class	Lactating	Closeup	Growing	Calves
Open lot or bedded pack barn [1]	General stream, lot	0.796	0.76	0.76	1
	General stream, feed alley	0.204	0.24	0.24	N/A
	minutes_away_for_milking	300	N/A	N/A	N/A
Bedded pack barn [2]	General stream, bedded pack	0.792	0.819	0.819	1
	General stream, feed alley	0.208	0.179	0.179	N/A
	minutes_away_for_milking	200	N/A	N/A	N/A

[1] Based on data from **meyer2019contract** and the assumption that open lot pens tend to be considerably larger than indoor-housed systems (greater travel distance).

[2] Based on data from **endres2007behavior**, which suggests behavior of cows in compost bedded pack barns is broadly similar to those in other housing systems. With this, freestall data from **gomez2010time** was used, which reported 4.3 hours of feeding time, 3.33 hours away for milking, and the remainder spent in the housing area.

- total_bedding_mass (kg): the total mass of bedding applied to the manure stream each day, calculated in **[MN.PEN.7]**.
- bedding_density (kg/m^3): user-defined density of the specified bedding.

Calculate total bedding dry solids

calc_total_bedding_dry_solids

$$total\_bedding\_dry\_solids(kg) = total\_bedding\_mass(kg) * bedding\_dry\_matter\_content$$

[AN.PEN.9]

When:

- total_bedding_mass (kg): the total mass of bedding applied to the manure stream each day, calculated in **[MN.PEN.7]**.
- bedding_dry_matter_content: user-defined dry matter content of the specified bedding.

Calculate total bedding water

calc_total_bedding_water

$$total\_bedding\_water(kg) = total\_bedding\_mass(kg) * (1 - bedding\_dry\_matter\_content)$$

[AN.PEN.10]

When:

- total_bedding_mass (kg): the total mass of bedding applied to the manure stream each day, calculated in **[MN.PEN.7]**.
- bedding_dry_matter_content: user-defined dry matter content of the specified bedding.

Update ManureStream values to reflect bedding

_apply_bedding (pen.py)

Table 56: Calculated manure and bedding characteristics

Variable	Units	Calculation
water	kg	ManureStream water (kg) + bedding_water (kg)
ash	kg	if bedding type = sand, ash (kg) = total_bedding_mass (kg); otherwise, ash = 0
phosphorus	kg	ManureStream phosphorus + total_bedding_mass (kg) * bedding_phosphorus_content)
non_degradable _volatile_solids	kg	If bedding type = sand, ManureStream non_degradable_volatile_solids (kg); If bedding type is not sand, ManureStream non_degradable_volatile_solids (kg) + total_bedding_dry_solids (kg)
total_solids	kg	ManureStream total_solids (kg) + total_bedding_dry_solids (kg)
volume	m^3	ManureStream volume (m^3) + total_bedding_volume (m^3)

8 Manure Handler

8.a Introduction

Manure handling refers to the method of managing manure in animal living spaces. For some housing/pen types, such as freestalls and tiestalls, this involves removing the manure from barn alleyways, and is usually accomplished using some form of scraping or flushing with water. Flush systems utilize water to remove the manure from alleyways, whereas scrape systems use little or no water. For other housing types like open lots and compost bedded pack barns, tillage or harrowing are commonly used to instead incorporate the manure, urine, bedding (if applicable) and air into the pack and allow it to dry.

The specific manure handler options in the RuFaS model are:

- **Flushing**

  Flushing involves using water to carry manure and soiled bedding away from housing areas. A series of pipes and channels direct water to wash the manure into a collection pit. Flushing helps maintain a cleaner environment for animals and reduces the buildup of manure on the floor. Flushing/recirculating may also be used to “lubricate” transport/pumping of manure from underfloor collection pits to the next manure management step. Flushing is commonly performed using recycled water from a solid liquid separator, liquid manure storage, or other liquid collection area, rather than using fresh water.

- **Manual Scraping**

  Scraping involves physically removing the manure and soiled bedding using equipment like tractor-mounted scrapers or skid steers, or the manual use of shovels/scrapers, etc. This method is typically used in barns with slatted or solid floors. Regular scraping prevents manure accumulation and provides a clean area for animals.

- **Alley Scraper**

A scraper blade, typically pulled along by a chain or cable, moves slowly down a pen or alleyway on a set schedule several times a day, pushing or pulling manure out of the pen/alley. The function and outcome is similar to manual scraping, but alley scrapers are mechanized and typically operate automatically.

- **Parlor Cleaning**

  Parlor cleaning is a RuFaS-specific handler that represents cleaning of manure from milking units, parlor stall floors, and other areas in the milking parlor that require the use of fresh, non-recycled water. Parlor cleaning also can include an optional flush system capability, which functions in the same manner as the flush system described above and is used to wash manure from holding area floors.

Manure handlers in RuFaS have two primary functions: calculate and add cleaning water if used, and calculate indoor housing emissions of CH_4, CO_2, and NH_3 if applicable.

The following are important assumptions made in the manure handler submodules, as well as context on how manure handlers are used in the larger manure management chain:

1. All manure excreted and all bedding applied each day (if applicable) is assumed to be recovered by the manure handler.

2. Manure handlers in RuFaS are intended to be utilized, at this time, with manure deposited in housing areas that is cleaned frequently (e.g., removed daily or every few days), such as indoor barn floors, or concrete feed alleys in open lot or compost bedded pack pens. Handlers should not be used in manure streams where manure resides for a long period of time, such as streams representing an open lot or bedded pack manure pack, that have a designated processor (i.e., Open Lot or Bedded Pack). Users can consider the manure handler functionality to be "built in" to these processors. Defining a handler prior to one of these types of processors will result in double-counting of housing emissions.

3. If a manure handler is used, it must be defined first in the manure management chain. This prevents inappropriate calculation of housing emissions "downstream" when manure is no longer in the animal housing area. Handlers are meant to represent manure handling in animal living spaces; other manure handling/transport systems such as pumps, augers, belts, etc. are not explicitly modeled in RuFaS.

Classes

Table 57: The classes of handlers that are used in this module.

Handlers	Description
Handler (Processor)	handler.py contanins basic methods that are applicable to all handlers
Single StreamHandler (Handler)	single_stream_handler.py contains alley screaper, manual scraping, and flush system-specific methods
ParlorCleaning (Handler)	parlor_cleaning.py contains parlor cleaning-specific methods

Required User Inputs

Table 58: Required inputs for the Manure Handler (refreshed_manure_management.json)

Variable	Units	Description
Name	none	Unique identifier of the specific handler configuration used.
Type	none	The type (alley scraper, manual scraping, flushing, or parlor scraping) of manure handler.
cleaning_water_use_-amount	L/animal/day	The rate of water use (L per animal per day, total of fresh and/or recycled water) by a specified manure handler.
cleaning_water_recycle_-fraction	none	The fraction (0.XX) of cleaning water used by a specified handler that is from recycled (not fresh) water sources.
use_parlor_flush	true/false	True/false indication for if a parlor flush (which may be partially or fully recycled water) is used in addition to routine parlor water cleaning with fresh water.

Other inputs
Instance(s) of ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane potential, $m^3/kgVS$):

- water
- ammoniacal_nitrogen
- nitrogen
- phosphorus
- potassium
- ash
- manure_degradable_volatile_solids

- manure_non_degradable_volatile_solids
- bedding_non_degradable_volatile_solids
- total_solids
- mass (equal to sum of water and total solids)
- total volatile solids (equal to sum of degradable and non-degradable volatile solids)
- volume
- methane_production_potential

Manure handlers also make use of information contained in the PenManureData subclass. This includes the following information specific to the pen from which the manure was generated:

- num_animals: number of animals in the pen
- animal_combination: type of animals in the pen (calves, growing post-weaning heifers through closeup period), closeup (cows and heifers within user-defined closeup period), or lactating cows)
- Manure deposition surface area (m2): surface area soiled with manure. See section below on determining soiled surface area
- pen_type: the type of pen the manure was generated in (freestall, tiestall, open lot, or bedded pack barn)
- daily_urine (kg): mass of urine in the manure
- urine_nitrogen (kg): mass of urine N in the manure
- stream_type: the type of manure stream the manure is in (parlor or general stream)

Note that stream_type is not a user input, it is automatically assigned. Parlor streams are created automatically and all other streams are assigned "general".

Expected Outputs

- All handler types:
  - ManureStream variables
  - total_cleaning_water_volume (m^3): total volume of fresh (non-recycled) cleaning water used for, and ultimately added to, a single manure stream on a single simulation day by the manure handler
  - barn_temperature ($\circ C$): ambient temperature in barn
- Alley scraper, manual scraping, flush system:
  - housing_methane (kg): daily CH_4 (kg) emitted from manure on barn floors
  - housing_ammonia_N_emissions: (kg): daily NH_3-N (kg) emitted from manure on barn floors
  - housing_carbon_dioxide (kg): daily CO_2 (kg) emitted from manure on barn floors

8.b Methodology

Calculate barn temperature

_determine_barn_temperature
Air temperature inside barns is, in general, slightly cooler than the outdoors during the day, and slightly-to-considerably warmer than the outdoor temperature overnight, depending on the season. With this, basic bounds were implemented in the determination of barn temperature. These bounds were suggested by manure SMEs and are supported by barn temperature ranges reported in Bucklin et al., 2009 (Florida, upper limit). The lower bound (5°C) suggested by SMEs was based on general industry standards/conditions.

If ambient (outdoor) air temperature < 5°C, barn temperature $= 5$°C
If ambient (outdoor) air temperature > 5° & < 30°C, barn temperature = air temperature
If ambient (outdoor) air temperature > 30°C, barn temperature $= 30$°C

Calculate cleaning/flushing fresh water addition

determine_handler_cleaning_water_volume
The following method calculates the volume of fresh (non-recycled) cleaning water used for, and ultimately added to, a single manure stream on a single simulation day by the manure handler. This method is used in all handler types; however, for parlor cleaning handlers, this method is only utilized if "use_-parlor_flush" is entered as true. If no water is used with the handler (user inputted cleaning water use rate = 0), the function will return 0.

$$cleaning\_water\ (L) = \\ num\_animals * (cleaning\_water\_use\_rate * (1 - cleaning\_water\_recycle\_fraction))$$

[MN.HDL.1]

When:

- num_animals: the number of animals in the pen assigned to the handler
- cleaning_water_use_rate (L/animal/d): the user-provided cleaning water use rate for the handler
- cleaning_water_recycle_fraction: the proportion of cleaning water that is from recycled, non-fresh sources

Total fresh water added to manure (L) each day is calculated as follows:

$$total\_cleaning\_water(m^3) = (cleaning\_water(L) + \\ fresh\_water\_volume\_used\_for\_milking(L)) * LITERS\_TO\_CUBIC\_METERS$$

[MN.HDL.2]

When:

- LITERS_TO_CUBIC_METERS: a constant set to 0.997 kg/m^3

Calculate milking parlor fresh water addition

The following method calculates the volume of fresh water used for, and ultimately added to, a single manure stream on a single simulation day by the manure handler for parlor cleaning/sanitation purposes. This calculation occurs only in the parlor_cleaning handler and water added is considered to be 100% fresh, non-recycled water, as fresh water is required for cleaning of milking units, floors, etc. The following function is implicit (occurs automatically with any manure stream assigned to the parlor cleaning handler type) and is based at this time on a default value fresh water use rate.

$$fresh\_water\_volume\_used\_for\_milking\ (L) = \\ num\_animals * MILKING\_FRESH\_WATER\_USE\_RATE$$

[MN.HDL.3]

When:

- num_animals: the number of animals in the pen assigned to the handler
- MILKING_FRESH_WATER_USE_RATE (30.0 L/animal/d): the default fresh water use rate for cleaning equipment/floors in the milking parlor

Calculate housing emissions

The calculation of housing emissions according to the methods described below is applicable only to areas where manure resides for short periods of time, i.e., tiestall and freestall housing systems, or, in open lot or bedded pack pens, manure deposited in a feed alley, exercise lot, etc. Housing emissions from bedded packs and open lot manure packs, where manure accumulates over weeks or months, are determined according to alternative methods due to the longer-term residence of manure in these pen types; see respective bedded pack barn and open lot documentation files for further info.

The methodology used to calculated housing emissions is adapted from the IFSM (Rotz, Corson, et al., 2023). Of note is that the equations described here were determined based on emissions in tiestall and freestall housing systems. In RuFaS, these equations may also be applied to determine emissions from manure deposited in other concrete areas and retained for a short time, such as open lot or bedded pack feed alleys. These systems were not represented in the original datasets used to generate these equations, and results should be interpreted with caution; however, the portion of manure deposited in feed alleys or other areas is relatively small, and emissions from this area are expected to be a small contributor to total GHG and nutrient losses.

Additionally, the following calculations are only utilized by the manual scraping, flush system, and alley scraper handler types. For the parlor cleaning handler type, manure in a parlor is assumed to be cleaned from floors/surfaces frequently and thus housing emissions should not be estimated.

Calculate housing CH_4 emissions

Manure deposited on barn floors or other housing areas is a small source of CH_4 emissions. The following method, from Chianese, Rotz, and Richard, 2009, is an empirically-derived equation consisting of a simple relationship between manure CH_4 emission rate, ambient temperature in the barn, and surface area covered by manure.

The soiled surface area (m2) value is calculated in the Animal module, using pen-level information (number of stalls, number of cows, etc.) and the soiled surface area value is passed to the manure module directly. As no housing emissions are considered to be emitted in the milking parlor, the total soiled surface area for each pen is apportioned to the pen's general manure streams(s) based on the user-provided stream proportion value(s).

The total surface area soiled with manure, based on the total number of animals, is as follows:

$$soiled\_surface\_area\ (m^2) = num\_stalls * surface\_area\ (m^2/animal)$$

[MN.HDL.4]

When:

- num_stalls: the number of stalls in the pen assigned to the handler

Table 59: The surface area per stall that is covered with manure is a function of the pen type and type of animal housed.

System Type	Lactating Cows	Closeup, Growing, or Calves
Tiestall	1.2 m^2	1.0 m^2
Freestall	3.5 m^2	2.5 m^2
Bedded Pack	5.0 m^2	5.0 m^2
Open Lot	3.0 m^2	3.0 m^2

- surface_area (m^2/animal): surface area per stall, provided in Table 8.

We use this surface area value to determine CH_4 emissions (kg) each day

$$housing\ CH_4\ (kg) = max^1(0.0, 0.13 * barn\_temperature(C)) * \frac{soiled\_surface\_area\ (m^2)}{1000}$$

[MN.MET.1]

When:

- barn_temperature ($°C$): ambient air temperature in the barn. Note that the max notation ensures that the temperature scaling parameter cannot be less than 0.13.
- soiled_surface_area (m^2): the total surface area soiled with manure

Calculate housing CO_2 emissions Housing CO_2 emissions are estimated similarly to CH_4 emissions, using an empirical equation based on soiled surface area and barn air temperature.

$$housing\ CO_2\ (kg) = max^1(0.0, 0.0065 + barn\_temperature(C)) * soiled\_surface\_area\ (m^2)$$

[MN.HDL.5]

When:

- barn_temperature ($°C$): ambient air temperature in the barn. Note that the max notation ensures that the temperature scaling parameter cannot be less than 0.13.
- soiled_surface_area (m^2): the total surface area soiled with manure

Calculate volatile solids losses The total quantity of VS loss through microbial degradation is estimated based on CH_4 emissions.

$$total\_VS\_loss\ (kg) = housing\ CH_4\ (kg) * VS\_TO\_METHANE\_LOSS\_RATIO$$

[MN.HDL.6]

When:

- housing CH_4 (kg): daily emissions of CH_4 from manure on barn floors, calculated in **[MN.MET.1]**.

- VS_TO_METHANE_LOSS_RATIO (kg): a constant set to 6.665 representing the kg of total VS degraded per kg of CH_4 emitted from manure. See the Slurry Storage section for more information on derivation of this constant.

The fraction of volatile solids lost from the manure degradable (VSd) and non-degradable VS (VSnd) fractions is assumed to be equal to the proportion of these fractions in manure entering the processor. No VS losses are attributed to bedding VSnd in manure handlers.

$$degradable\_VS\_frac = (\frac{degradable\_VS}{total\_VS})$$

[MN.ADG.7]

Calculate housing NH_3 emissions Ammonia N loss from manure is predicted using methods based on Rotz and Oenema, 2006. Here, estimated NH_3-N volatilized from the surface of manure is based on a function where the NH_3-N is transported to the free atmosphere through a pathway with a finite resistance. Daily NH_3-N loss depends primarily on pen type (which determines soiled surface area), manure TAN content, and barn air temperature. The same base equation for NH_3-N volatilization is utilized for estimation of both housing and storage NH_3-N emissions.

In the original work of Rotz and Oenema, 2006, housing NH_3-N emissions are based only on urine (not manure) TAN concentration, rather than total manure (urine + feces) TAN. Within RuFaS, urine nutrients are not tracked separately from feces/manure. Therefore, the model assumes, for simplicity, that all TAN originates from urine and thus animal-excreted urine TAN = manure TAN. However, to maintain consistency with documentation, what would ordinarily in RuFaS be referred to as manure TAN within RuFaS is referred to here as urine TAN, though urine mass here (kg/d) is reflective of actual estimated urine mass, not manure (urine + feces) mass.

- First, we need to derive the value of the equilibrium coefficient Q for the NH_3 gas in the air for a given concentration of TAN in stored slurry using Henry's law of distribution.

The concentration of NH_3 in the free atmosphere is assumed to be zero. Since Q is a function of the Henry's law coefficient K_h and a dissociation of ammonium coefficient K_a, we will calculate those first.

Henry's law coefficient (K_h)

$$K_h = 10^{\frac{1478}{barn\ temperature - 1.69}}$$

[MN.AMM.1]

When:

- Barn temperature (K): ambient air temperature in the barn

Dissociation coefficient of ammonium (K_a)

$$K_a = 1 + 10^{(0.09018 + (\frac{2729.9}{barn\ temperature} - pH)}$$

[MN.AMM.2]

When:

- Barn temperature (K): ambient air temperature in the barn
- pH: the pH of the manure solution, set to 7.5 by default

Equilibrium coefficient (Q)

$$Q = K_h * K_a$$

[MN.AMM.3]

When:

- K_h: Henry's law coefficient, calculated in **[MN.AMM.1]**.
- K_a: Dissociation coefficient of ammonium, calculated in **[MN.AMM.2]**.

- Second, we determine the effective total resistance value of NH_3 transport from manure solution to the surface and from the manure surface to the atmosphere.

$$ammonia\_resistance = HOUSING\_SPECIFIC\_CONSTANT * (1 - 0.027 * (20.0 - barn\ temperature))$$

[MN.AMM.4]

  When:

  - HOUSING_SPECIFIC_CONSTANT: a housing-specific constant value, set to 260 s/m.
  - Barn temperature ($^{\circ}C$): ambient air temperature in the barn.

- Next, the rate of NH_3-N loss in kg N/m^2 from manure on the barn floor is calculated

$$NH_3N\ emission\ rate\ (kg\ N/m^2) = \frac{(TAN*c*y)}{(r*M*Q)}$$

[MN.AMM.5]

  When:

  - TAN (kg N/m^2): Mass of ammoniacal N in barn floor manure
  - c: time conversion constant (86400 s per d)
  - y: manure density, set to 990 kg/m^3
  - r: the resistance of NH_3 transfer from solution to manure surface, and from manure surface to atmosphere, determined in **[MN.AMM.4]**
  - M: urine mass (received from animal module)
  - Q: equilibrium coefficient, calculated in **[MN.AMM.3]**

- Lastly, we calculate total NH_3-N emissions (kg), based on the emission rate we just calculated and the soiled surface area.

$$NH_3N\ emissions\ (kg) = NH_3N\_rate * soiled\_surface\_area$$

[MN.AMM.7]

When:

- NH_3N_rate (kg N/m^2): Rate of NH_3-N loss (kg/m^2) from manure, calculated in **[MN.AMM.5]**
- soiled_surface_area (m^2): the total surface area soiled with manure

8.c Manure composition update

Below is a summary of updates to ManureStream variables.

The formulas below may be a summarization of multiple steps detailed above, and are intended to provide an overview of what mass losses/gains are reflected in the value of each variable.

Equations in the table below are in the format of "Output/exiting value" = "Entering processor value" +/- XYZ. "Entering" refers to the manure entering the processor (i.e., being loaded into the digester) and output variables reflect the manure leaving the processor (i.e., digestate leaving the digester). The Calculation column describes how the variable is updated in the processor.

Table 60: Calculated manure storage variables and their units

Variable	Units	Calculation
water	kg	Entering water (kg) + (total_cleaning_water (m^3) × WATER_DENSITY_KG_PER_M3)
total_ammoniacal_nitrogen	kg	min(0, entering ammoniacal_nitrogen (kg) − NH_3-N emissions (kg))
nitrogen	kg	min(0, entering nitrogen (kg) − NH_3-N emissions (kg))
phosphorus	kg	Entering phosphorus (kg)
potassium	kg	Entering potassium (kg)
ash	kg	Entering ash (kg)
non_degradable_volatile_-solids		Entering non_degradable_volatile_solids − (total_VS_loss × (1 − degradable_VS_frac))
degradable_volatile_solids		Entering degradable_volatile_solids − (total_VS_loss × degradable_VS_frac)
total_solids	kg	Entering total_solids − total_VS_loss
volume	m^3	$\frac{\text{Entering volume} - \text{total\_VS\_loss}}{\text{ManureConstants.SLURRY\_MANURE\_DENSITY}}$ [1,2]

[1] WATER_DENSITY_KG_PER_M3: a constant with a value of 0.997 kg/m^3.

[2] Note that for parlor cleaning handlers, the value of all housing emissions variables will be reported as 0.

9 Solid Liquid Separators

9.a Introduction

A solid-liquid separator (SLS) is a specialized piece of equipment or system designed to separate larger, solid particles from the liquid component of manure. This type of system may be implemented for a

variety of reasons, such as to improve ease of handling of manure liquid, improve the nutrient concentration in manure liquid, reduce storage volume required for manure lagoons or other storage systems, prevent solid accumulation in a covered manure storage, or reclaim manure solids for use as bedding or compost.
The solid fraction of manure slurry is composed of fibers originating primarily from manure, but also from feed, bedding, and other environmental sources. Once mechanically separated, the moisture content of the solid fraction ranges from 70% to 90%, depending on the specific separator system used. To further reduce moisture content of the separated solids, an additional dewatering or drying step may be incorporated. Manure solids may be directly used or sold for animal bedding (after stabilization), composted, or land applied.
After the manure solids are mechanically separated, the remaining liquid fraction is composed of small particles, water, and most of the nutrients present in the manure. Because the bulk of relevant nutrients (e.g., P, N, K) are mainly associated with the small particles which remain with the liquid fraction after separation occurs, separating the larger, less nutrient-dense particles out from the slurry liquid enriches the concentration of these nutrients in the liquid fraction. Solid-liquid separation also makes the manure slurry easier to pump, transport, and apply to fields, as large particles that may clog lines and sprayers are removed.

Implementation in RuFaS

In RuFaS, solid liquid separators currently represent mechanical, short retention time pieces of equipment. GHG emissions and other nutrient transformations from long-retention separators such as weeping walls or settling basins are not yet represented in the model. Because of this, the impact of these longer retention separators can only be approximated via modification of the separation efficiencies of the short retention time methods.

Currently, SLS inputs are simply nutrient-specific separation efficiencies that reflect the proportion of a specific nutrient that is separated into the solids fraction. With this, essentially any type of mechanical manure separation system can be modeled, as long as effective separation efficiencies are known. Manure is assumed to be loaded, processed, and passed to the next processor within a single day. Default separation efficiencies exist for two common types of SLS - a screw press and a rotary screen. The separated manure solids and remaining liquid fraction are quantified and can be managed in separate, downstream manure management systems in RuFaS. However, separated solids cannot be directly recycled (i.e., utilized with the specific nutrient composition of those separated solids) "upstream" as bedding.

Classes

Table 61: List of Python classes for solid liquid separators.

Handlers	**Description**
Separator (Processor)	separator.py

Required User Inputs

Table 62: The required inputs of this section.

Variable	Definition (units)	Description
Name	none	Unique identifier of the specific separator configuration used.
type	rotary screen or screw press	The type of solid liquid separator ; selection of this input indicates which default separations efficiencies should be used, however, any and all separation efficiencies may also be overwritten by user input.
separated_solids_dry_matter	none	percent dry matter by mass of the manure solids post solid-liquid separation.
total_solids_efficiency	none	the proportion of manure total solids (TS) that are separated into the manure solids fraction by the separator.
volatile_solids_efficiency	none	the proportion of manure volatile solids (VS) that are separated into the manure solids fraction by the separator.
nitrogen_efficiency	none	the proportion of manure total nitrogen that is separated into the manure solids fraction by the separator.
ammoniacal_nitrogen_efficiency	none	the proportion of manure ammoniacal nitrogen that is separated into the manure solids fraction by the separator.
phosphorus_efficiency	none	the proportion of manure phosphorus that is separated into the manure solids fraction by the separator.
potassium_efficiency	none	the proportion of manure potassium that is separated into the manure solids fraction by the separator.
ash_efficiency	none	the proportion of manure ash that is separated into the manure solids fraction by the separator.

Other inputs

Instance(s) of ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane potential, $m^3/kgVS$):

- water
- ammoniacal_nitrogen
- nitrogen
- phosphorus
- potassium
- ash
- manure_degradable_volatile_solids

- manure_non_degradable_volatile_solids
- bedding_non_degradable_volatile_solids
- total_solids
- mass (equal to sum of water and total solids)
- total volatile solids (equal to sum of degradable and non-degradable volatile solids)
- volume
- methane_production_potential

Expected Outputs

- Two sets of ManureStream variables, representing the separated solids ("SeparatedSolids") and liquid fraction ("SeparatedLiquid"), respectively

9.b Methodology

Separate nutrients

The equations below describe the general process for separating nutrients between liquid and solid fractions. The general equation for nutrient removal is as follows:

$$Separated\ solids\ nutrient\ content = received\ manure\ nutrient * separation\ efficiency$$

[MN.SEP.1]

$$Separated\ liquid\ nutrient\ content = received\ manure\ nutrient * (1 - separationefficiency)$$

[MN.SEP.2]

When:

- Received manure nutrient (kg): quantity of the specific nutrient in manure being loaded into the separator on a single day.

Nutrients that are separated according to this pattern are presented in the table below. The default separation efficiencies, as well as minimum and maximum allowable separation efficiencies, for the rotary screen and screw press separators are also provided (Hegg, R. E. Larson, and Moore, 1981; Mukhtar, Sweeten, and Auvermann, 1999; Varma et al., 2021). Note that separate removal efficiencies for manure degradable and non-degradable VS or bedding non-degradable VS are not specifiable at this time; all VS fractions are assumed to be removed at the rate of total VS removal.

Table 63: The default, minimums, and maximums for various separation efficiencies.

Type	Variable	Default	Min	Max
Rotary Screen	Total solids	0.35	0.25	0.4
	Nitrogen	0.3	0.25	0.35
	Ammoniacal N	0.15	0.1	0.2
	Phosphorus	0.4	0.3	0.45
	Potassium	0.15	0.05	0.2
	Ash	0.2	0.05	0.3
	Volatile solids	0.35	0.3	0.45
Screw Press	Total solids	0.25	0.15	0.35
	Nitrogen	0.3	0.2	0.35
	Ammoniacal N	0.15	0.05	0.2
	Phosphorus	0.2	0.1	0.35
	Potassium	0.23	0.15	0.35
	Ash	0.2	0.05	0.3
	Volatile solids	0.25	0.2	0.4

1 (Jørgensen and Jensen, 2009; Fournel et al., 2019)

Calculate total mass and water

Separated solids fraction Total mass of the separated solids fraction is determined by dividing the mass of separated solids by the user-inputted separated solids dry matter content.

$$Separated\ solids\ mass\ (kg) = \frac{Separated\ solids\ TS\ (kg)}{Separated\ solids\ \%DM}$$

[MN.SEP.3]

When:

- Separated solids TS (kg): the mass of solids in the separated manure solids fraction
- Separated solids %DM: user-inputted separated solids dry matter content

The mass of water in the separated solids fraction can then be determined as follows, given the assumption that total mass is equal to the sum of solids plus water:

$$Separated\ solids\ water\ (kg) = Separated\ solids\ mass\ (kg) - Separated\ solids\ TS\ (kg)$$

Separated liquid fraction

The total mass of the separated liquid fraction, as well as the mass of water, are simply equal to the mass and water in manure loaded into the separator, minus the quantity of each respective value partitioned into the separated solids fraction.

$$Separated\ solids\ mass\ (kg) = \\ Received\ mass\ (kg) - Separated\ solids\ mass\ (kg) Separated\ solids\ water\ (kg) = \\ Received\ water\ (kg) - Separated\ solids\ water\ (kg)$$

Calculate volume

Volume of each separated fraction is determined by dividing the mass of each fraction by its respective density:

$$Separated\ solids\ volume\ m^3 = \frac{Separated\ solids\ mass}{SOLID\_MANURE\_DENSITY}$$

[MN.SEP.4]

$$Separated\ liquid\ volume\ m^3 = \frac{Separated\ liquid\ mass}{LIQUID\_MANURE\_DENSITY}$$

[MN.SEP.5]

When:

- SOLID_MANURE_DENSITY (kg/ m^3): The default density of solid manure, set to 700 kg/ m^3
- LIQUID_MANURE_DENSITY(kg/ m^3): The default density of liquid manure, set to 900 kg/ m^3

9.c Manure composition update

Below is a summary of updates to ManureStream variables. Note that the formulas below may be a summarization of multiple steps detailed above, and are intended to provide an overview of what mass losses/gains are reflected in the value of each variable.
Equations in the table below are in the format of "Output/exiting value" = "Entering processor value" +/- XYZ. "Entering" refers to the manure entering the processor (i.e., being loaded into the digester) and output variables reflect the manure leaving the processor (i.e., digestate leaving the digester). The Calculation column describes how the variable is updated in the processor.

Table 64: Solids and liquid fraction calculations for separated manure components

Variable	Units	Calculation (Solids Fraction)	Calculation (Liquid Fraction)
water	kg	Separated solids mass (kg) − Separated solids TS (kg)	Entering water (kg) − Separated solids water (kg)
total_ammoniacal _-nitrogen	kg	Entering ammoniacal nitrogen (kg) × TAN removal efficiency	Entering ammoniacal nitrogen (kg) × (1 − TAN removal efficiency)
nitrogen	kg	Entering nitrogen (kg) × N removal efficiency	Entering nitrogen (kg) × (1 − N removal efficiency)
phosphorus	kg	Entering phosphorus (kg) × phosphorus removal efficiency	Entering phosphorus (kg) × (1 − phosphorus removal efficiency)
potassium	kg	Entering potassium (kg) × potassium removal efficiency	Entering potassium (kg) × (1 − potassium removal efficiency)
ash	kg	Entering ash (kg) × potassium removal efficiency	Entering ash (kg) × (1 − potassium removal efficiency)

Continued on next page

Variable	Units	Calculation (Solids Fraction)	Calculation (Liquid Fraction)
manure_degradable _volatile_solids	-	Entering manure VSd (kg) $\times$ VS removal efficiency $\times$ degradable_VS_frac	Entering VSd (kg) $\times$ (1 $-$ VS removal efficiency) $\times$ degradable_VS_frac
manure_non_degrad-able _volatile_solids	-	Entering manure VSnd (kg) $\times$ VS removal efficiency $\times$ (1 $-$ degradable_VS_frac)	Entering VSnd (kg) $\times$ (1 $-$ VS removal efficiency) $\times$ (1 $-$ degradable_VS_frac)
bedding_non_degrad-able _volatile_solids	-	Entering manure VSnd (kg) $\times$ VS removal efficiency $\times$ (1 $-$ degradable_VS_frac)	Entering VSnd (kg) $\times$ (1 $-$ VS removal efficiency) $\times$ (1 $-$ degradable_VS_frac)
total_solids	kg	Entering TS (kg) $\times$ TS removal efficiency	Entering TS (kg) $\times$ (1 $-$ TS removal efficiency)
volume	m^3	$\frac{\text{Separated solids mass (kg)}}{\text{SOLID\_MANURE\_DENSITY}}$	$\frac{\text{Separated liquid mass (kg)}}{\text{LIQUID\_MANURE\_DENSITY}}$

10 Anaerobic Digestion

10.a Introduction

Anaerobic digestion is a process where dairy cow manure is treated in an oxygen-free (anaerobic) environment to produce biogas, containing approximately 60% CH_4, 40% CO_2, which can be utilized as an on-farm or exported energy source. Additional benefits of anaerobic digestion include reduced manure odor, improved stability and quality of manure for fertilization purposes, and reductions in nutrient losses via undesirable greenhouse gas emissions.
Manure can undergo anaerobic digestion in a specialized anaerobic digestion chamber/system, usually referred to simply as a 'digester', or can occur in other manure storage systems that create anaerobic conditions, such as anaerobic lagoons.

Implementation of RuFaS

The anaerobic digestion submodule in RuFaS currently represents anaerobic digestion within an enclosed, mechanical digester system. The following assumptions are made in the RuFaS anaerobic digestion submodule:

- The digester represented is assumed to be a mesophilic, continuous stirred-tank reactor (CSTR).
- Manure is loaded into the digester and removed from the digester once daily.
- The digester is fully functional at the start of the simulation, i.e., the digester is full of substrate and the microbial population has stabilized.
- Residence time of manure in the digester is not modeled. As a result, the composition of effluent exiting the digester each day is identical to the influent composition, with the exception of nutrient losses or transformations that occur in the digester.
- At this time, CH_4 generation and volatile solids destruction is based strictly on the quantity of manure volatile solids loaded; other factors are not considered at this time.

The anaerobic submodule has two primary functions in RuFaS:

- Estimate daily methane production (kg/d) based on the daily mass of manure volatile solids (VS) loaded into the digester.
  - VS loading is dependent on the number and type of animals contributing manure to the digester, the diet of the animals, quantity and type of bedding, and any upstream manure handling processes, e.g., solid liquid separation, water addition, etc.
- Update the composition of the liquid manure effluent leaving the digester, which enters the anaerobic lagoon.
  - Reflecting VS loss in effluent exiting the digester is essential to capture the reduction in methane emissions from digestate compared to undigested, liquid manure, as well as changes in proportion of inorganic (ammoniacal) to total nitrogen.

Classes

Table 65: List of Python classes for anaerobic digestors.

Digester	Description
AnaerobicDigester	anaerobic_digester.py inherits behavior from the base class, Digester; however, at this time, there is no functionality included in the base Digester class. Therefore all methods are contained within the anaerobic_digestion.py child class.

Required User Inputs

Table 66: Required inputs for this section.

Variable	Units	Description
name	none	Unique identifier of the specific anaerobic digester configuration used.
hydraulic_retention_time	days	Number of days manure spends in the anaerobic digester. Note that this variable is not utilized directly by the module, but is utilized by the Economics, Emissions and Energy module.
anaerobic_digestion _temperature_set_point	°C	Temperature set point for the anaerobic digestion. This input is utilized in the conversion of CH_4 and CO_2 generation volume to mass; it does not directly influence CH_4 emissions.
biogas_leakage_fraction	none	Fraction of biogas generated in the anaerobic digester that escapes to the atmosphere through unintended leakage and is not collected by the gas capture system .

Other inputs

Instance(s) of ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane production potential, $m^3/kgVS$):

- water
- ammoniacal_nitrogen
- nitrogen
- phosphorus
- potassium
- ash
- manure_degradable_volatile_solids
- manure_non_degradable_volatile_solids
- bedding_non_degradable_volatile_solids

- total_solids
- mass (equal to sum of water and total solids)
- total volatile solids (equal to sum of degradable and non-degradable volatile solids)
- volume
- methane_production_potential

Expected Outputs

- captured_biogas_volume (m^3): Captured biogas (assumed to be composed of 40% CO_2, 60% CH_4) volume after accounting for leakage on the current day
- captured_methane_volume (m^3): Captured methane volume on the current day, after accounting for leakage
- methane_leakage_mass (kg): Mass of CH_4 lost to the atmosphere through unintended leakage on the current day. This variable is expressed as mass as opposed to volume as CH_4 emissions are reported in kg in the rest of the manure module and other RuFaS modules

10.b Methodology

Calculate Daily Methane Generation

Description: calculates volume of methane (CH_4) generated from a CSTR digester. Methane generation is estimated from the daily loading of manure volatile solids (VS). Degradable (VSd) and non-degradable (VSnd) VS are tracked separately but the ratio of VSd : VSnd does not affect CH_4 estimation; only the total quantity of VS is considered. To perform this calculation, we make several assumptions/simplifications:

- The ratio of chemical oxygen demand (COD): VS in dairy manure is assumed to be 1.2 to 1. 1 kg of COD can generate 0.4 kg of methane. Therefore, 1 kg VS reduction/degradation in the anaerobic digester = 1.2 kg COD = 480 L CH_4. In other words, each kg VS of reduced is assumed to generate 480 L of CH_4.
- A CSTR reduces manure VS content by approximately 50%. This is a generalization across CSTR digesters of varying efficiencies, based on expert opinion from W. Liao (MSU) and A. Leytem (USDA-ARS).
- Considering that 480 L of CH_4 are generated per kg of VS destroyed, and approximately 50% of VS loaded are anticipated to be destroyed, we estimate CH_4 generation by assuming 240 L CH_4 are generated per VS kg loaded into the digester; this is also in alignment with the IPCC (2019) Tier II manure CH_4 Bo value.
- Lastly, the volumetric ratio of CH_4 to CO_2 generation in the CSTR is assumed to be 6:4 (e.g. generation of 60% CH_4, 40% CO_2 biogas) based on commonly cited digester performance metrics (e.g., EPA, Fernández et al., 2015). The total quantity of VS destroyed in anaerobic digestion is then assumed to be equal to the quantity of CH_4 and CO_2 generated in the digester. This assumption is made due to a lack of data on destruction of degradable vs. non-degradable VS in anaerobic digestion.

$$generated\_methane\_volume\ (m^3) = \\ ACHIEVABLE\_METHANE\_EMISSION * total\_volatile\_solids\ (kg)$$

[MN.ADG.1]

When:

- ACHIEVABLE_METHANE_EMISSION = achievable methane generation constant (m^3 CH_4 per kg VS loaded into digester); Constant Value: 0.24 m^3 CH_4/kg VS
- total_volatile_solids = daily mass (kg) of manure total volatile solids loaded into the digester, received from ManureStream(s)

Lastly, we need to convert daily CH_4 volume (generated_methane_volume) to CH_4 mass. First we calculate CH_4 density according to the user-provided digestion setpoint temperature, then we apply the density value to the volume of CH_4 generated.

$$methane\_density = \frac{METHANE\_MOLAR\_MASS}{(IDEAL\_GAS\_LAW\_R*(temperature\_set\_point+CELSIUS\_TO\_KELVIN)}$$

[MN.ADG.2]

When:

- METHANE_MOLAR_MASS (16.04 g/mol): molar mass of CH4
- IDEAL_GAS_LAW_R (0.0821 L atm/mol K): ideal gas law R value
- temperature_set_point ($°C$): user-provided digestion set point temperature
- CELSIUS_TO_KELVIN (273.15): value to convert $°C$ temperature values to K

$$generated\_methane\_mass\ (kg) = generated\_methane\_volume\ (m^3) * methane\_density$$

[MN.ADG.3]

When:

- generated_methane_volume (m^3): volume of CH_4 generated in the digester on a single day, calculated in **[MN.ADG.1]**.

Calculate destroyed volatile solids

_destroy_volatile_solids

Description: Microbes convert (destroy) VS during anaerobic digestion and produce biogas containing primarily CH_4 and CO_2. Therefore, the quantity of VS is assumed to be equal to the mass of CH_4 and CO_2 generated. In this section, we calculate the total mass of CO_2 and CH_4 generated, which is used later to update the degradable and non-degradable VS values of the ManureStream instance passed to the next processor. First, we calculate the density of CO_2 based on the user-provided digestion setpoint

temperature.

$$carbon\_dioxide\_density = \frac{CARBON\_DIOXIDE\_MOLAR\_MASS}{(IDEAL\_GAS\_LAW\_R*(temperature\_set\_point+CELSIUS\_TO\_KELVIN))}$$

[MN.ADG.4]

When:

- CARBON_DIOXIDE_MOLAR_MASS (44.01 g/mol): molar mass of CO_2
- IDEAL_GAS_LAW_R (0.0821 L atm/mol K): ideal gas law R value
- temperature_set_point ($°C$): user-provided digestion set point temperature
- CELSIUS_TO_KELVIN (273.15): value to convert $°C$ temperature values to K

Second, we determine the quantity of CO_2 volume and mass generated, assuming digester biogas contains a 60:40 volumetric ratio of CH_4 to CO_2.

$$generated\_carbon\_dioxide\_mass = (generated\_methane\_volume * CARBON\_DIOXIDE\_TO\_METHANE\_RATIO * CARBON\_DIOXIDE\_DENSITY)$$

[MN.ADG.5]

When:

- CARBON_DIOXIDE_MOLAR_MASS (44.01 g/mol): molar mass of CO_2; Constant Value: = 4/6 L/L (0.667)
- carbon_dioxide_density (kg per m^3) = ideal gas law value to convert CO_2 from mass to volume

Third, we calculate the total destruction of total VS.

$$total\_volatile\_solids\_destruction = generated\_methane\_mass + generated\_carbon\_dioxide\_mass$$

[MN.ADG.6]

We then utilize the value of total_volatile_solids_destruction (kg) to update VSd, manure VSnd, and bedding VSnd. As mentioned above, the total quantity of VS destroyed is partitioned between the three VS fractions according to the proportion of each fraction in manure entering the digester. E.g., if manure entering contained 60% VSd, 10% manure VSnd, and 30% bedding VSnd, 60% of destroyed VS will be subtracted from VSd, 10% from manure VSnd, and 30% from bedding VSnd. To do this, we calculate the ratio of VSd to VS (degradable_volatile_solids_frac) and manure VSd to VS and apply these fractions to the total_volatile_solids_destruction value to determine the updated VS fraction values.

$$degradable\_VS\_frac = \frac{degradable\_VS\ (kg)}{total\_VS\ (kg)}$$

[MN.ADG.7]

$$manure\_non\_degradable\_VS\_frac = \frac{manure\_non\_degradable\_VS\ (kg)}{total\_VS\ (kg)}$$

[MN.ADG.8]

$$degradable\_VS = degradable\_VS\ (kg) - (total\_VS\_destruction\ (kg) * degradable\_VS\_frac)$$

[MN.ADG.9]

$$manure\_non\_degradable\_VS = manure\_non\_degradable\_VS\ (kg) - total\_VS_destruction * manure\_non\_degradable\_VS\_frac$$

[MN.ADG.10]

$$bedding\_non\_degradable\_VS = bedding\_non\_degradable\_VS\ (kg) - total\_VS_destruction * (1 - manure\_non\_degradable\_VS\_frac\ + degradable\_VS\_frac)$$

[MN.ADG.11]

Calculate methane leakage

_calculate_methane_leakage

Description: Calculates the volume of CH_4 generated that is lost to the atmosphere via leakage. The leakage fraction is currently a user input with a default value of 1%, which represents a conservative leakage rate. Leakage is largely dependent on the age and type of digester and should ideally be provided by the user for specificity.

$$methane\_leakage\_volume\ (m^3) = generated\_methane\_volume\ (m^3) * biogas\_leakage\_fraction$$

[MN.ADG.12]

When:

- generated_methane_volume (m^3): volume of CH_4 generated in the digester on a single day, calculated in **[MN.ADG.1]**
- biogas_leakage_fraction (%): fraction of biogas generated in the anaerobic digester that escapes to the atmosphere through unintended leakage; Default Value: 0.01 (1%)

Update digester effluent composition

_report_anaerobic_digester_outputs

The calculations below are some additional prerequisites to generating anaerobic digestion-specific outputs.

- The equation below is used to calculate the quantity of net, captured gas according to the quantity of biogas leakage.

$$captured\_methane\_volume\ (m^3) = generated\_methane\_volume\ (m^3) - methane\_leakage\_volume\ (m^3)$$

[MN.ADG.13]

When:

- generated_methane_volume (m^3): total volume of CH_4 generated in the digester on a specific day
- methane_leakage_volume (m^3): the volume of CH_4 generated that is lost to the atmosphere via leakage

- The equation below is used to calculate the updated volume of manure in the digester, accounting for the volume of destroyed VS.

$$updated\_volume(m^3) = incoming\_volume(m^3) - (\frac{total\_volatile\_solids\_destruction(kg)}{ManureConstants.SLURRY\_MANURE\_DENSITY})$$

[MN.ADG.14]

When:

- updated_volume (m^3) = incoming_volume (m^3) - (total_volatile_solids_destruction (kg) / ManureConstants.SLURRY_MANURE_DENSITY)

- Microbial processes during anaerobic digestion are known to increase the proportion/mass of total ammoniacal N (TAN), though the total mass of N is generally not different pre and post-digestion (Aguirre-Villegas, R. A. Larson, and Sharara, 2019). To reflect this, the proportion of TAN in manure in the digester is multiplied by a fixed factor. The factor (TAN_INCREASE_FACTOR, 1.60) was chosen based on a target of 50% loss of total N via NH_3-N emissions from a subsequent digestate lagoon, as reported by the USDA GHG estimation guidelines for uncovered digestate storage (Hanson, Itle, and Edquist, 2024).

$$updated\_ammoniacal\_nitrogen(kg) = min(ammoniacal\_nitrogen(kg) * TAN\_INCREASE\_FACTOR, nitrogen(kg))$$

[MN.ADG.15]

When:

- ammoniacal_nitrogen (kg): the mass of ammoniacal N in manure loaded into the digester on a single day
- TAN_INCREASE_FACTOR: factor by which total ammoniacal nitrogen content is increased by the anaerobic digestion process, set to 1.60
- Note that the "min" notation prevents manure TAN from exceeding manure total N.

10.c Manure composition update

Below is a summary of updates to ManureStream variables. Note that the formulas below may be a summarization of multiple steps detailed above, and are intended to provide an overview of what mass losses/gains are reflected in the value of each variable. Equations in the table below are in the format of "Output/exiting value" = "Entering processor value" +/- XYZ. "Entering" refers to the manure entering the processor (i.e., being loaded into the digester) and output variables reflect the manure leaving the processor (i.e., digestate leaving the digester). The Calculation column describes how the variable is updated in the processor.

Table 67: Calculated anaerobic digestion outputs.

Variable	Units	Calculation
water	kg	Entering water (kg)
total_ammoniacal_nitrogen	kg	min(entering ammoniacal_nitrogen (kg) $\times$ TAN_INCREASE_FACTOR, nitrogen (kg))
nitrogen	kg	Entering nitrogen (kg)
phosphorus	kg	Entering phosphorus (kg)
potassium	kg	Entering potassium (kg)
ash	kg	Entering ash (kg)
degradable_volatile_solids		Entering degradable_volatile_solids (kg) $-$ total_VS_destruction (kg) $\times$ degradable_VS_frac
manure_non_degradable_-volatile_solids		Entering non_degradable_volatile_solids (kg) $-$ total_VS_destruction (kg) $\times$ manure_non_degradable_VS_frac
bedding_non_degradable_-volatile_solids		Entering non_degradable_volatile_solids (kg) $-$ total_VS_destruction (kg) $\times$ (1 - degradable_VS_frac - manure_non_degradable_VS_frac)
total_solids	kg	Entering total_solids (kg) $-$ total_VS_destruction (kg)
volume	m^3	Entering volume (m^3) $-$ $\frac{\text{total\_VS\_destruction (kg)}}{\text{ManureConstants.SLURRY\_MANURE\_DENSITY}}$

11 Slurry Storage

11.a Introduction

Manure that is stored and managed at 7 to 12% dry matter is generally considered to be "slurry" manure. Several common options exist for storing manure at this %DM range. Slurry may be stored in underfloor pits, where manure is deposited directly into the pit through slatted floors, or is moved to an underfloor storage via scrapers or other manure handling systems. Manure may also be transported to outdoor storage tanks or pits/basins, which may be covered or uncovered.

Compared to anaerobic lagoons, slurry storages are typically of a smaller capacity, and do not result in controlled treatment (e.g., reduction of odor, N content reduction, organic matter decomposition) of manure. Slurry storages are typically emptied more frequently than liquid manure storages like anaerobic lagoons and contain more concentrated manure. In general, the biological processes in slurry storages and anaerobic lagoons are similar: microbes break down manure carbohydrates and proteins, resulting in CO_2, CH_4, and N_2O emissions, and mineralization of organic to inorganic N, and N losses occur through NH_3 volatilization at the manure surface. However, management of these two types of liquid manure storages differs as described above, which results in differences in emissions and nutrient losses.

Implementation in RuFaS
There are two options for slurry storage in RuFaS, slurry storage outdoor and slurry storage underfloor, which function almost identically. The key differences between the two methods are presented in Table 13. Because the two methods are highly similar, both are covered in this document.

Table 68: Key differences for two options of slurry storage.

	Slurry Storage Underfloor	**Slurry Storage Outdoor**
Temperature	Barn temperature method	Modeled air temperature method
Cover options	Only "no cover" permitted	Cover, cover and flare, crust, or no cover
Precipitation volume	Precipitation excluded from storage	Precipitation excluded or included depending on cover option selection

In the slurry storage submodules, accumulated manure in storage is modeled on a daily timestep. Nutrient/mass gains from daily addition of manure (feces/urine, bedding, wash water) to storage, and precipitation volume entering storage, are tracked. Gas emissions are calculated daily based on the quantity of nutrients in stored manure, manure temperature, storage type, use of a cover, and storage duration. Manure composition is then updated according to net nutrient losses/gains. Manure accumulates in storage until the end of the user-defined storage interval is reached. However, quantities of manure may additionally be removed from storage according to the user-defined manure application schedule. Note that at this time, water and nutrients from surface runoff, and water evaporation from storage manure, are not captured in slurry storage submodules.

Classes

Note that both slurry storage methods inherit some behavior from the base class, Storage. Because the two methods are highly similar, the content below is representative of both types of slurry storage, with specific differences between the two noted explicitly (see Table 18).

Slurry Storage	Description
SlurryStorageOutdoor(Storage)	slurry_storage_outdoor.py
SlurryStorageUnderfloor(Storage)	slurry_storage_underfloor.py

Required User Inputs

Table 69: Required inputs for the slurry storage section.

Variable	Units	Description
Name	none	Unique identifier of the specific slurry storage configuration used.
Capacity	m^3	The volumetric capacity of the slurry storage, in m^3. Note that this variable is a placeholder at this time, and does not influence model calculations.
Cover	-	Fefault value is “no cover” for slurry storage underfloor, as these storages do not generally have a synthetic cover, and typically receive too much surface disturbance to form a crust. Note that precipitation is always excluded from slurry storage underfloor given they are assumed to be completely covered or indoors. Default value for slurry storage outdoor is “no cover”. Cover (represents either a synthetic, precipitation-excluding cover, or an enclosed tank) or crust (a naturally-forming crust over $>50\%$ of the stored manure surface) are also options.
Surface_area	m^2	The surface area of the slurry storage. If not provided by the user, surface area is calculated based on the number of mature cows in the herd; see Calculate Surface Area section below.
Storage_time_period	days	The number of days that manure is stored between emptying events. At the end of this interval, the manure storage is emptied completely.

Other inputs

Instance(s) of ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane production potential, $m^3/kgVS$):

- water
- ammoniacal_nitrogen
- nitrogen

- phosphorus
- potassium
- ash
- manure_degradable_volatile_solids
- manure_non_degradable_volatile_solids
- bedding_non_degradable_volatile_solids
- total_solids
- mass (equal to sum of water and total solids)
- total volatile solids (equal to sum of degradable and non-degradable volatile solids)
- volume
- methane_production_potential

Expected Outputs

- ManureStream variables representing manure loaded (received) into storage each day, and accumulated manure after accounting for nutrient and mass gains/losses
- storage_methane (kg): Daily emission of CH_4 from accumulated manure in slurry storage
- storage_ammonia (kg): Daily emission of NH_3 from accumulated manure in slurry storage
- storage_nitrous_oxide (kg): Daily emission of N_2O from accumulated manure in slurry storage.

11.b Methodology

Calculate manure temperature
Slurry storage underfloor

_determine_barn_temperature

Temperature of stored manure is assumed to be equal throughout the entire mass of manure. In slurry storage underfloor, manure temperature is assumed to be equal to air temperature, but is bounded to 5 to $30°C$. See barn temperature determination method in Manure Handler section for more information.

Slurry storage outdoor
_determine_outdoor_storage_temperature

Daily temperature of manure in slurry storage outdoors is determined using a sine/cosine least squares fit to user-provided weather data, with a fixed amplitude damping and lag (phase shift) factor. For more information, see the *Calculate Stored Manure Temperature* in the Anaerobic Lagoon section.

Calculate storage surface area

Exposed surface area (m^2) of the manure in storage is important in determining NH_3-N emissions, as well as in determining precipitation volume added to storage if the storage is not covered or indoors. Wherever possible, this value should be provided by the user if modeling a real farm. If farm-specific information is unavailable or the farm being modeled is theoretical, the surface area should be estimated using tools like the USDA's Animal Waste Management Version 2.4.1. However, the RuFaS team recognizes that minimizing required inputs is desirable, though a fixed storage surface area is undesirable due to the variability in storage structure size and surface area. With this, an equation was developed that estimates storage surface area based on the following assumptions:

- All manure excreted by animals on the farm enters the specified storage. At this time, the Manure module is not capable of assessing the proportion of manure excreted that is stored in the defined storages, therefore, all manure is assumed to be stored in the current storage, for the purposes of surface area estimation.
- The storage is 15 ft deep, with vertical walls.
- The storage receives 2500 mm of precipitation per year.
- Herd composition, and thus manure excretion, is fixed, and the number of animals in each life stage class is proportional to the number of mature cows.

A constant value was derived to calculate estimated manure excretion based on the number of mature cows housed on the farm (a user input). The average number of animals in each class was determined according to default RuFaS animal lifecycle inputs, and the total mass and volume of manure excreted by the herd was calculated. This resulted in an estimated daily herd-wide manure excretion of 168.6 kg or 0.118 m^3 of manure per mature cow housed on the farm. The resulting equation is used to calculate storage surface area (m^2).

$$surface\_area(m^2) = \frac{(cow\_num * MANURE\_CONVERSION\_CONSTANT \times storage\_time \times FREEBOARD\_CONSTANT)}{(DEPTH\_CONSTANT - PRECIPITATION\_CONSTANT)}$$

[MN.STO.1]

When:

- cow_num: user-inputted number of mature cows housed on the farm
- MANURE_CONVERSION_CONSTANT: Factor to estimate m^3 of herd-wide manure produced per day per mature cow housed on the farm, set to 0.1175 m^3.
- storage_time (days): user-inputted number of days that manure is stored in this storage for before being emptied.
- FREEBOARD_CONSTANT: the volume allowance above the maximum volume of a slurry or liquid manure storage, set to 1.20 (20%).
- DEPTH_CONSTANT: value for slurry or liquid manure storage depth, set to 4.572 m (15 feet).
- PRECIPITATION_CONSTANT: The annual precipitation constant value, used only in determination of storage surface area if surface area is not provided by the user, set to 0.25 m.

Calculate precipitation volume

Covers have implications for inclusion or exclusion of precipitation volume, as well as for N_2O emissions. Four cover options exist for slurry storages:

- "Cover": An impermeable cover that does not permit precipitation to enter the storage.
- "Cover and flare": An impermeable cover with flaring of methane produced in storage. See Cover and Flare section.

- “Crust”: A naturally forming crust exists on the surface of the slurry storage.
- “No cover”: Storage is not covered or indoors.

The cover type for slurry storage underfloor in the default manure management file is “uncovered”, as these storages are typically not enclosed. However, precipitation is always excluded from underfloor slurry storages, regardless of the cover type – see Precipitation below.
Precipitation volume for uncovered outdoor slurry storages is calculated as follows:

$$Daily\ precipitation\ volume(m^3) = storage\ surface\ area(m^2) \times precipitation(m)$$

[MN.STO.2]

When:

- Storage surface area: the user-defined or model-estimated storage surface area (m^2).
- Precipitation: the daily amount of precipitation (m).

Calculate methane emissions

_calculate_methane_emissions

We use an adaptation of a method originally conceived by Sommer, Petersen, and Moller, 2004 to calculate daily emissions of CH_4 from degradable and non-degradable VS in slurry. These equations focus on the degradation of degradable and non-degradable volatile solids (VS) present in the manure. Factors like degradable and non-degradable VS (VSd and VSnd) content in storage, temperature, and location (indoor/outdoor) affect estimated CH_4 emissions. We apply the original method from Sommer, Petersen, and Moller, 2004 with updated dairy manure Arrhenius and activation energy values from Elsgaard, Olsen, and Petersen, 2016 and Petersen et al., 2024.

Methane emissions are calculated in the same way for slurry storage outdoor and underfloor, with the exception that temperature of manure in outdoor vs. underfloor slurry storage is determined differently, as described below. The same equation is utilized to calculate CH_4 emissions from VSd and VSnd (from both manure and bedding sources), though the rate-correcting factor differs between the two.

First, we must calculate the value of the Arrhenius exponent (_calculate_arrhenius_exponent). This value directly represents the responsiveness of biological reaction speed to temperature, and in the context of this empirical equation, may also be related to the methane potential of manure in storage and activity of the microbial population:

$$Arrh\_exp\ g\ (CH_4\ kg^{-1}\ VS\ h^{-1}) = e^{Ln(A) - \frac{ACTIVATION\_ENERGY}{(Gasconstant * manuretemperature)}}$$

[MN.MET.2]

When:

- Ln(A): The natural log of the Arrhenius parameter (NATURAL_LOG_ARRHENIUS_CONSTANT constant), set at 30.7 based on Petersen et al., 2024. This is an empirically-derived value determined based on observed manure CH_4 emission values.
- ACTIVATION_ENERGY: the apparent activation energy of methanogenesis in cattle slurry (J/-mol), set at 81,000 J/mol, based on Elsgaard, Olsen, and Petersen, 2016.

- Gas constant: ideal gas constant, set at 8.314 J K/mol.
- Manure temperature (K): temperature of manure in storage.

Now we can calculate actual daily CH_4 emission, based on the total quantity of VSd and VSnd in stored manure. The basic equation, used to calculate CH_4 emissions for each VS fraction, is as follows:

$$CH_4\ emission\ from\ VS_{d\ or\ nd}\ (kg\ d^{-1}) = 24 * Arrh\_exp * VS_{d\ or\ nd} * rate\_factor$$

[MN.MET.3]

When:

- 24: conversion factor from hours to day.
- Arrh_exp: Arrhenius parameter for CH_4 emission rate (g CH_4 kg^{-1} VS h^{-1}), calculated in **[MN.MET.2]**.
- $VS_{d\ or\ nd}$: The mass (kg) of VSd or VSnd in manure in slurry storage.
- rate_factor: The unitless rate-correcting factor, set to 1 for VS_d and 0.01 for VS_{nd}.

The total daily CH_4 emission is equal to the sum of emissions from the VS_d and VS_{nd} fractions.

Calculate cover and flare methane emissions

_calculate_cover_and_flare_methane

The cover and flare option is applicable to slurry storage outdoor only (i.e., not usable with slurry storage underfloor). If the cover and flare option is selected, daily CH_4 emission from slurry storage is multiplied by a methane destruction efficiency value. The set value for methane destruction efficiency is 81%, based on a white paper commissioned by Dairy Management, Inc. on cover and flare efficiency (**wallaceDMI**). The updated daily CH_4 emission (kg) from a cover and flare slurry storage is as follows:

$$Daily\ storage\ CH_4\ (kg) : storage\ CH_4 \times (1 - METHANE\_DESTRUCTION\_EFFICIENCY)$$

[MN.MET.4]

When:

- Storage CH_4 (kg): total daily kg of CH_4 emitted from stored manure, calculated in **[MN.MET.3]**.
- METHANE_DESTRUCTION_EFFICIENCY: coefficient for destruction of methane by the flare, set to 0.81.

Calculate volatile solids losses
_apply_methane_emissions

Daily emissions of CH_4 and CO_2 from slurry storage occur through microbial degradation of VS in slurry manure, among other processes (Petersen et al., 2024). Therefore, gaseous emissions from slurry storage result in a decrease in the quantity of VS in stored slurry. VSd and VSnd remaining in manure are updated separately according to their respective loss via CH_4 **[MN.STO.4]**. Here, we assume a fixed 1:3 molar ratio of CH_4-C to CO_2-C emissions from stored slurry from Petersen et al., 2024. This enables calculation of the total amount of C and thus VS_d and VS_{nd} lost through CH_4 and CO_2 emissions based on the quantity of CH_4 emitted from each VS fraction.

Given that C is assumed to be lost via CH_4 and CO_2 emissions in a ratio of 1:3, we assume for each C lost as CH_4, 3 C are lost as CO_2. CH_4 is 75% C by mass, thus for each kg of CH_4 emitted, 0.7498 C are lost via CH_4 and (3 x 0.7498) are lost from CO_2, for a total of 2.992 kg C per kg of CH_4 emitted. We assume manure VS are 45% C (Petersen et al., 2024); therefore, 2.9992 kg C / 45% C = 6.665 kg VS are lost per kg of CH_4 emitted.

$$VS_{d\ or\ nd}\, loss\ (kg) = CH_4\, emission\, from\, VS_{d\ or\ nd}\ (kg) * VS\_TO\_METHANE\_LOSS\_RATIO$$

[MN.STO.3]

When:

- CH_4 emission from VS_d or $_{nd}$ (kg): total daily kg of CH_4 emitted from VS_d or VS_{nd}, calculated in **[MN.MET.3]**
- VS_TO_METHANE_LOSS_RATIO: default ratio of VS degraded per kg of CH_4 emitted from slurry storage, set to 6.665

Calculate ammonia emissions
_calculate_ammonia_emissions

Emission of NH_3-N from stored slurry is determined using equations from Rotz and Oenema, 2006, which are also utilized in the IFSM (Rotz, Corson, et al., 2023). Ammonia emissions are influenced by the quantity of total ammoniacal N (TAN) accumulated in manure storage, manure temperature, and manure storage surface area. First, we must derive the various parameters utilized in the calculation.

- First, we need to derive the value of the equilibrium coefficient Q for the NH_3 gas in the air for a given concentration of TAN in stored manure using Henry's law. Note that the concentration of NH_3 in the free atmosphere is assumed to be zero. Since Q is a function of the Henry's law coefficient K_h and a dissociation of ammonium coefficient K_a, we will calculate those first.

  Henry's law coefficient (K_h)

  $$K_h = 10^{\frac{1478}{manure\ temperature}} - 1.69$$

  [MN.AMM.1]

  When:

  – Manure temperature (K): temperature of manure storage.

  Dissociation coefficient of ammonium (K_a)

  $$K_a = 1 + 10^{(0.09018 + \frac{2729.9}{manure\ temperature} - pH)}$$

  [MN.AMM.2]

  When:

  – Manure temperature (K): temperature of stored manure.

- DEFAULT_STORED_MANURE_PH: the pH of the manure in storage, set to 7.5 by default

Equilibrium coefficient (Q)

$$Q = K_h * K_a$$

[MN.AMM.3]

When:

- K_h: Henry's law coefficient, calculated in **[MN.AMM.1]**.
- K_a: Dissociation coefficient of ammonium, calculated in **[MN.AMM.2]**.

- Next, the rate of NH_3-N loss in kg N/m^2 from stored manure is calculated:

$$NH_3N\ emission\ rate\ (kg\ N/m^2) = \frac{(TAN*c*y)}{(STORAGE\_RESISTANCE*M*Q)}$$

[MN.AMM.5]

When:

- TAN (kg): Mass of ammoniacal N in stored manure
- c: time conversion constant (86400 s per d)
- y: manure density, set to 990 kg/m^3
- STORAGE_RESISTANCE: A constant value representing the sum of resistance of NH_3 transfer from solution to manure surface, and from manure surface to atmosphere, set at 23.1 s/m.
- M (kg): Total mass of stored manure
- Q: Equilibrium coefficient calculated in **[MN.AMM.3]**

- Lastly, we calculate total NH_3-N emissions (kg), based on the emission rate we just calculated and the manure storage surface area.

$$NH_3N\ emissions\ (kg) = NH_3N\_rate * surface\_area$$

[MN.AMM.7]

When:

- NH_3N_rate (kg N/m^2): Rate of NH_3-N loss (kg/m^2) from manure, calculated in **[MN.AMM.5]**.
- surface_area (m^2): Total manure storage surface area.

Calculate nitrous oxide emissions

_calculate_nitrous_oxide_emissions
N_2O emissions (kg N_2O-N) are based on the daily quantity of manure N loaded into storage, whether the manure storage is covered or uncovered. This method is based on Intergovernmental Panel on Climate

Change, 2019; however, it should be noted that the original IPCC, 2006 method is based on daily manure N excretion by animals, whereas the current method is based on manure N loading into storage, which may reflect upstream N losses from NH_3 emissions in housing, solid liquid separation, etc. The calculation is as follows:

$$N_2O - N\ emissions\ (kg) = Received\_N * N_2O\ factor$$

[MN.NIT.1]

When:

- Received_N (kg): Quantity of manure total N loaded into storage on the current day
- N_2O factor: kg of N_2O-N emitted per kg of manure N added per day to storage, based on the following logic:
    - Cover type = crust OR cover; 0.005
    - Cover type = no cover; 0 (no N_2O emissions)

11.c Received, stored, and emptied outputs

Manure storages in RuFaS report two types of outputs to OutputManager each day: received manure and stored manure.

Received manure outputs represent the quantity of manure mass and nutrients added to the manure storage on a single day. No nutrient losses from gas or other emissions/losses are reflected in these output values.

Stored manure outputs represent the accumulated quantity of manure and nutrients present in storage on a single day. These values are the net quantity of mass/nutrients remaining each day after adding received manure values and subtracting any losses to gas emissions or other losses. In slurry storage processors, daily losses include CH_4, NH_3, and N_2O emissions. The order of operations in updating accumulated manure values is:

- Add received manure values to stored manure values
- Calculate gas emissions and total nutrient losses based on stored manure values
- Update stored manure values based on the day's nutrient losses. See the Manure composition update section for specific details on how nutrient gains and losses are accounted for on a daily timestep.

For slurry storage and all other storage processor types, the stored manure values (not received manure) are passed to the next processor in the chain (e.g. another storage, field application, export, etc.) when the storage time interval is complete.

Emptied manure
Manure may be removed from storage via requests made by the Crop and Soil module. The user specifies the days and years for manure removal (i.e. application), as well as the application type (liquid or solid) and quantity of N or P required for each application date within year. Note that these actions are the responsibility of the Crop and Soil module; more information on manure application inputs and methodology can be found in the Crop and Soil module documentation. When manure is removed from storage by the Crop and Soil module, emptied manure outputs report the quantity of manure and nutrients removed on that day, and Manure Stream attributes representing stored manure are updated accordingly to reflect post-removal amounts remaining in storage.

11.d Manure composition updates

Received manure:

In slurry storage processors, the following nutrient sources are represented in received manure values:

- ManureStream values, as received from the previous processor(s) in the manure management chain
- Precipitation water (kg), calculated in MN.STO.2 (if applicable), is added to the water value in ManureStream

Stored manure:

Below is a summary of updates to ManureStream variables representing the stored manure. Note that the formulas below may be a summarization of multiple steps detailed above, and are intended to provide an overview of what mass losses/gains are reflected in the value of each variable.

Equations in the table below (Calculation column) are in the format of: updated stored manure value = yesterday's stored manure value + today's manure value +/- XYZ. The updated stored manure values reflect the total quantity of manure/nutrients in storage on a single day after accounting for all gains/losses that occurred on that day. Received manure simply refers to the manure being loaded into the manure storage each day.

Table 70: Manure storage variable calculations

Variable	Units	Calculation
water	kg	stored manure water (kg) + received manure water (kg)
total_ammoniacal_nitrogen	kg	max(0, stored manure ammoniacal nitrogen (kg) + received ammoniacal nitrogen (kg) − NH_3N emissions (kg))
nitrogen	kg	stored manure nitrogen (kg) + received manure nitrogen (kg) − NH_3N emissions (kg) − N_2O-N emissions (kg)
phosphorus	kg	stored manure phosphorus (kg) + received manure phosphorus (kg)
potassium	kg	stored manure potassium (kg) + received manure potassium (kg)
ash	kg	stored manure ash (kg) + received manure ash (kg)
degradable_volatile_solids		stored degradable VS (kg) + received degradable VS (kg) − VSd loss (kg)
manure_non_degradable_-volatile_solids		stored manure non-degradable VS (kg) + received manure non-degradable VS (kg) − manure VSnd loss (kg)
bedding_non_degradable_-volatile_solids		stored bedding non-degradable VS (kg) + received bedding non-degradable VS (kg) − bedding VSnd loss (kg)
total_solids	kg	stored total solids (kg) + received total solids (kg) − VSd loss (kg) − VSnd loss (kg)
volume	m^3	stored volume (m^3) + Received volume (m^3) − $\frac{\text{VSd loss (kg)}+\text{VSnd loss (kg)}}{\text{SLURRY\_MANURE\_DENSITY}}$

12 Anaerobic Lagoon

12.a Introduction

Manure that is stored and managed at less than $\tilde{5}$% dry matter is generally considered to be liquid manure. Liquid manure is generated by either dilution of raw or slurry manure, generally through the addition of wash or flush water, or removal of a portion of manure solids through solid liquid separation methods (mechanical separator, settling basin, etc.) or anaerobic digestion. Liquid manure is generally stored in a type of large, outdoor storage structure called an anaerobic lagoon, or simply a lagoon. Anaerobic lagoons are not simply structures in which to store manure. Lagoons facilitate biological breakdown of organic materials, which reduces volatile solids content and odor, though also increases N mineralization and loss as ammonia, particularly if the lagoon is uncovered. Accordingly, anaerobic lagoons have specific design and management requirements to facilitate biological treatment activity (Natural Resources Conservation Service (NRCS), 2017). Some characteristics that separate an anaerobic lagoon from slurry or liquid manure storage are:

- Greater storage capacity
- Less frequent and less complete emptying, resulting in longer solids/sludge retention time
- Storage of liquid rather than slurry manure
- Controlled volatile solids loading rate
- Lagoons are generally a lined or unlined in-ground basin, whereas slurry storage may be either in-ground or above-ground tanks or other structures

Implentation in RuFaS

In RuFaS, the underlying biological and gas emission methods are identical for slurry storages vs. anaerobic lagoons, as the biological process of organic matter breakdown is very similar between the two in reality. However, the differences in size, dilution, management, and other factors differ between the two in reality, leading to generally greater GHG emissions from lagoons compared to slurry storages.
In the anaerobic lagoon submodule, accumulated manure in storage (i.e., held in the lagoon) is modeled on a daily timestep. Nutrient/mass gains from daily addition of manure (feces/urine, bedding, wash water) to storage, and precipitation volume entering storage, are tracked. Gas emissions are calculated daily based on the quantity of nutrients in stored manure, manure temperature, storage type, use of a cover, and storage duration. Manure composition is then updated according to net nutrient losses/gains. Manure accumulates in storage until the end of the user-defined storage interval is reached. However, quantities of manure may additionally be removed from storage according to the user-defined manure application schedule

Classes

Table 71: List of Python classes for anaerobic lagoon.

Anaerobic Lagoon	Description
AnaerobicLagoon(Storage)	anaerobic_lagoon.py

Required User Inputs

Table 72: Required inputs for the anaerobic lagoon section (refreshed_manure_management.json)

Variable	Units	Description
Name	none	Unique identifier of the specific anaerobic lagoon configuration used.
Capacity	m^3	The volumetric capacity of the anaerobic lagoon, in m^3. Note that this variable is a placeholder at this time, and does not influence model calculations.
Cover	-	The type of cover used with the anaerobic lagoon.
Surface_area	m^2	The surface area of the anaerobic lagoon at the minimum operating level.
Storage_time_-period	days	The number of days that manure is stored between emptying events. At the end of this interval, the manure storage is emptied completely.

Other inputs

Instance(s) of ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane production potential, $m^3/kgVS$):

- water
- ammoniacal_nitrogen
- nitrogen
- phosphorus
- potassium
- ash
- manure_degradable_volatile_solids
- manure_non_degradable_volatile_solids
- bedding_non_degradable_volatile_solids
- total_solids
- mass (equal to sum of water and total solids)
- total volatile solids (equal to sum of degradable and non-degradable volatile solids)

- volume
- methane_production_potential

Expected Outputs

- ManureStream variables representing manure loaded (received) into storage each day, and accumulated manure after accounting for nutrient and mass gains/losses
- storage_methane (kg): Daily emission of CH_4 from accumulated manure in an anaerobic lagoon.
- storage_ammonia (kg): Daily emission of NH_3 from accumulated manure an anaerobic lagoon.
- storage_nitrous_oxide (kg): Daily emission of N2O from accumulated manure an anaerobic lagoon.

12.b Methodology

Calculate manure temperature

_determine_outdoor_storage_temperature

Manure temperature is modeled using a cosine function whose parameters are derived from a least-squares fit of simulation-wide weather data. The air temperature amplitude is reduced using a damping factor to reflect the smaller annual variation in manure temperature relative to air. The phase shift (i.e., timing of peak temperature) is determined based on the least squares function and is adjusted by a fixed lag constant representing the delayed thermal response of manure temperature relative to air temperature.

First we determine the amplitude of the manure temperature function by applying the damping factor.

$$manure\_amplitude = amplitude \times MANURE\_DAMPING\_FACTOR$$

[MN.STO.13]

When:

- amplitude: modeled amplitude of the seasonal air temperature function, calculated from user-supplied, simulation-wide weather data
- MANURE_DAMPING_FACTOR: a fixed damping factor applied to the air temperature amplitude, set to 0.65.

Second, we use this amplitude in the following function to determine modeled manure temperature(°C) each simulation day. Note the function includes a 'max' term to implement a lower temperature bound for manure temperature.

$$manure\_temp = max((mean\_temp \times manure\_amplitude \times \cos \frac{2\pi}{365} \times (jday - phase\_shift - MANURE\_TEMPERATURE\_LAG)), min\_temp)$$

[MN.STO.14]

When:

- mean_temp (°C): simulation-wide mean air temperature
- manure_amplitude: amplitude of the manure temperature function, determined in **MN.STO.13**
- jday: Julian day of the simulation
- phase_shift: Julian date of peak air temperature in the simulation

- MANURE_TEMPERATURE_LAG (days): fixed lag constant representing the delayed thermal response of manure temperature relative to air temperature, set to 30.
- min_temp (°C): A fixed minimum manure temperature constant, dependent on the type of storage.
  - Anaerobic lagoon: 1 °C
  - Slurry storage outdoor: -20 °C

Calculate storage surface area
Exposed surface area (m^2) of the manure in storage is important in determining NH_3-N emissions, as well as in determining precipitation volume added to storage if the storage is not covered or indoors. Wherever possible, this value should be provided by the user if modeling a real farm. If farm-specific information is unavailable or the farm being modeled is theoretical, the surface area should be estimated using tools like Animal Waste Management. However, the RuFaS team recognizes that minimizing required inputs is desirable, though a fixed storage surface area is undesirable due to the variability in storage structure size and surface area. With this, an equation was developed that estimates storage surface area based on the following assumptions:

- All manure excreted by animals on the farm enters the specified storage. At this time, the Manure module is not capable of assessing the proportion of manure excreted that is stored in the defined storages, therefore, all manure is assumed to be stored in the current storage, for the purposes of surface area estimation.
- The storage is 15 ft deep, with vertical walls.
- The storage receives 2500 mm of precipitation per year.
- Herd composition, and thus manure excretion, is fixed, and the number of animals in each life stage class is proportional to the number of mature cows.

A constant value was derived to calculate estimated manure excretion based on the number of mature cows housed on the farm (a user input). The average number of animals in each class was determined according to default RuFaS animal lifecycle inputs, and the total mass and volume of manure excreted by the herd was calculated. This resulted in an estimated daily herd-wide manure excretion of 168.6 kg or 0.118 m^3 of manure per mature cow housed on the farm. The resulting equation is used to calculate storage surface area (m^2).

$$surface\_area(m^2) = \frac{(cow\_num * MANURE\_CONERSION\_CONSTANT * storage\_time * FREEBOARD\_CONSTANT)}{(DEPTH\_CONSTANT - PRECIPITATION\_CONSTANT)}$$

[MN.STO.1]

When:

- cow_num: user-inputted number of mature cows housed on the farm
- MANURE_CONVERSION_CONSTANT: Factor to estimate m^3 of herd-wide manure produced per day per mature cow housed on the farm, set to 0.1175 m^3.
- storage_time (days): user-inputted number of days that manure is stored in this storage for before being emptied
- FREEBOARD_CONSTANT: the volume allowance above the maximum volume of a slurry or liquid manure storage, set to 1.20 (20%).

- DEPTH_CONSTANT: value for slurry or liquid manure storage depth, set to 4.572 m (15 feet).
- PRECIPITATION_CONSTANT: The annual precipitation constant value, used only in determinatio

Calculate precipitation volume
The use of covers has implications for inclusion or exclusion of precipitation volume, as well as for N^2O emissions. Four cover options exist for anaerobic lagoons.

- Cover
- Cover and flare
- Crust
- No cover

Detailed descriptions are outlined in the Slurry Storage section of this module. Precipitation volume for anaerobic lagoons that are uncovered or have a crust is calculated as follows:

$$Daily\ precipitation\ volume\ (m^3) = storage\ surface\ area\ (m^2) *\ precipitation\ (m)$$

[MN.STO.2]

When:

- Storage surface area: the user-defined or model-estimated storage surface area (m^2).
- Precipitation: the daily amount of precipitation (m).

Calculate methane emissions
_calculate_methane_emissions

We use an adaptation of a method originally conceived by Sommer, Petersen, and Moller, 2004 to calculate daily emissions of CH_4 from degradable and non-degradable VS in anaerobic lagoons. These equations focus on the degradation of degradable and non-degradable volatile solids (VS) present in the manure. Factors like degradable and non-degradable VS (VSd and VSnd) content in storage, temperature, and location (indoor/outdoor) affect estimated CH_4 emissions. We apply the original method from Sommer et al., 2022 with updated dairy manure Arrhenius and activation energy values from Elsgaard, Olsen, and Petersen, 2016 and Petersen et al., 2024. The same equation is utilized to calculate CH_4 emissions from VSd and VSnd, though the rate-correcting factor differs between the two. First, we must calculate the value of the Arrhenius exponent (_calculate_arrhenius_exponent). This value directly represents the responsiveness of biological reaction speed to temperature, and in the context of this empirical equation, may also be related to the methane potential of manure in storage and activity of the microbial population:

$$Arrh\_exp\ (g\ CH_4\ kg^{-1}\ VS\ h^{-1}) = e^{Ln(A) - \frac{ACTIVATION\_ENERGY}{(Gasconstant*manuretemperature)}}$$

[MN.MET.2]

When:

- Ln(A): The natural log of the Arrhenius parameter (NATURAL_LOG_ARRHENIUS_CONSTANT), set at 30.6 based on Petersen et al., 2024. This is an empirically-derived value determined based on observed manure CH_4 emission values.
- ACTIVATION_ENERGY: the apparent activation energy of methanogenesis in cattle slurry (J/-mol), set at 81,000 J/mol, based on Elsgaard, Olsen, and Petersen, 2016.
- Gas constant: ideal gas constant, set at 8.314 J K/mol.
- Manure temperature (K): temperature of manure in storage.

Now we can calculate actual daily CH_4 emission, based on the total quantity of VSd and VSnd in stored manure. The basic equation, used to calculate CH_4 emissions for each VS fraction, is as follows:

$$CH_4\ emission\ from\ VS_{d\ or\ nd}\ (kg\ d^{-1}) = 24 * Arrh\_exp * VS_{d\ or\ nd} * rate\_factor$$

[MN.MET.3]

When:

- 24: conversion factor from hours to day.
- Arrh_exp: Arrhenius parameter for CH_4 emission rate (g CH_4 kg^{-1} VS h^{-1}), calculated in **[MN.MET.2]**.
- S_d or $_{nd}$: The mass (kg) of VS_d or VS_{nd} (from both manure and bedding) in manure in storage.
- rate_factor: The unitless rate-correcting factor, set to 1 for VS_d and 0.01 for VS_{nd}.

The total daily CH_4 emission is equal to the sum of emissions from the VS_d and VS_{nd} fractions.

Calculate cover and flare methane emissions
_calculate_cover_and_flare_methane

If the cover and flare option is selected, daily CH_4 emission from an anaerobic lagoon is multiplied by a methane destruction efficiency value. The set value for methane destruction efficiency is 81%, based on a white paper commissioned by Dairy Management, Inc. on cover and flare efficiency authored by Jim Wallace and co-authors. The updated daily CH_4 emission (kg) from a cover and flare lagoon is as follows:

$$Daily\ storage\ CH_4\ (kg) : storage\ CH_4 \times (1 - METHANE\_DESTRUCTION\_EFFICIENCY)$$

[MN.MET.4]

When:

- Storage CH_4 (kg): total daily kg of CH_4 emitted from stored manure, calculated in **[MN.MET.4]**.
- METHANE_DESTRUCTION_EFFICIENCY: destruction efficiency: coefficient for destruction of methane by the flare, set to 0.81.

Calculate volatile solids losses
_apply_methane_emissions

Daily emissions of CH_4 and CO_2 from anaerobic lagoons occur through microbial degradation of VS in manure, among other processes (Petersen et al., 2024). Therefore, gaseous emissions from lagoons result in a decrease in the quantity of VS in stored manure. VSd and VSnd remaining in manure are updated separately according to their respective loss via CH_4 **[MN.STO.4]**. Here, we assume a fixed 1:3 molar ratio of CH_4-C to CO_2-C emissions from stored manure from Petersen et al., 2024. This enables calculation of the total amount of C and thus VS_d and VS_{nd} lost through CH_4 and CO_2 emissions based on the quantity of CH_4 emitted from each VS fraction.

Given that C is assumed to be lost via CH_4 and CO_2 emissions in a ratio of 1:3, we assume for each C lost as CH_4, 3 C are lost as CO_2. CH_4 is 75% C by mass, thus for each kg of CH_4 emitted, 0.7498 C are lost via CH_4 and (3 x 0.7498) are lost from CO_2, for a total of 2.992 kg C per kg of CH_4 emitted. We assume manure VS are 45% C (Petersen et al., 2024); therefore, 2.9992 kg C / 45% C = 6.665 kg VS are lost per kg of CH_4 emitted.

$$VS_{d\ or\ nd}\, loss\ (kg) = CH_4\, emission\, from\, VS_{d\ or\ nd}\ (kg) * VS\_TO\_METHANE\_LOSS\_RATIO$$

[MN.STO.3]

When:

- CH_4 emission from VSd or nd (kg): total daily kg of CH_4 emitted from VS_d or VS_{nd}, calculated in MN.MET.3
- VS_TO_METHANE_LOSS_RATIO: default ratio of VS degraded per kg of CH_4 emitted from slurry storage, set to 6.665

Calculate manure retention at emptying
_emptying_fraction

Anaerobic lagoons, through their settling action, accumulate and retain a bottom layer of solids often referred to as "sludge". Additionally, depending on the frequency and extent of lagoon agitation, retention time of volatile solids in lagoons is typically explicitly managed to promote biological degradation of solids. These factors contribute to the generally greater CH_4 emissions per unit of volatile solids loaded into anaerobic lagoons compared to in-ground basin or tank manure storages. To directly capture the greater retention of manure at emptying events, and to indirectly capture the greater biological activity in anaerobic lagoons, a default manure retention factor is implemented in RuFaS. This factor dictates the portion of manure which, when the storage time interval is reached, is retained in the lagoon. This factor is applied evenly to all manure constituents (i.e., ManureStream variables).

$$retained\_manure_i = \\ accumulated\_manure_i \times ANAEROBIC\_LAGOON\_MANURE\_RETENTION$$

[MN.STO.15]

When:

- i: manure consistituent i
- accumulated_manure_$_i$: quantity of manure consistituent i present in the accumulated anaerobic lagoon manure when the
- ANAEROBIC_LAGOON_MANURE_RETENTION: constant fraction of the accumulated stored manure that is retained in the anaerobic lagoon when the storage time interval is reached, set to 0.10

Calculate ammonia emissions
_calculate_ammonia_emissions

Emission of NH_3-N from anaerobic lagoons is determined using equations from Rotz and Oenema, 2006, which are also utilized in the IFSM (Rotz, Corson, et al., 2023). Ammonia emissions are influenced by the quantity of total ammoniacal N (TAN) accumulated in manure storage, manure temperature, and manure storage surface area. First, we must derive the various parameters utilized in the calculation.

- First, we need to derive the value of the equilibrium coefficient Q for the NH_3 gas in the air for a given concentration of TAN in stored manure using Henry's law. Note that the concentration of NH_3 in the free atmosphere is assumed to be zero. Since Q is a function of the Henry's law coefficient K_h and a dissociation of ammonium coefficient K_a, we will calculate those first.

  Henry's law coefficient (K_h)

  $$K_h = 10^{\frac{1478}{manure\ temperature}} - 1.69$$

  [MN.AMM.1]

  When:

  – Manure temperature (K): temperature of manure storage.

  Dissociation coefficient of ammonium (K_a)

  $$K_a = 1 + 10^{(0.09018 + \frac{2729.9}{manure\ temperature} - pH)}$$

  [MN.AMM.2]

  When:

  – Manure temperature (K): temperature of manure storage.
  – DEFAULT_STORED_MANURE_PH: the pH of the manure in storage, set to 7.5 by default

  Equilibrium coefficient (Q)

  $$Q = K_h * K_a$$

  [MN.AMM.3]

  When:

  – K_h: Henry's law coefficient, calculated in **[MN.AMM.1]**.
  – K_a: Dissociation coefficient of ammonium, calculated in **[MN.AMM.2]**.

- Next, the rate of NH_3-N loss in kg N/m2 from stored manure is calculated:

  $$NH_3N\ emission\ rate\ (kg\ N/m^2) = \frac{(TAN*c*y)}{(STORAGE\_RESISTANCE*M*Q)}$$

[MN.AMM.5]

When:

- TAN (kg): Mass of ammoniacal N in stored manure
- c: time conversion constant (86400 s per d)
- y: manure density, set to 990 kg/m^3
- STORAGE_RESISTANCE: A constant value representing the sum of resistance of NH_3 transfer from solution to manure surface, and from manure surface to atmosphere, set at 23.1 s/m.
- M (kg): Total mass of stored manure
- Q: Equilirbium coefficient calculated in **[MN.AMM.3]**

- Lastly, we calculate total NH_3-N emissions (kg), based on the emission rate we just calculated and the manure storage surface area.

$$NH_3N\ emissions\ (kg) = NH_3N\_rate * surface\_area$$

[MN.AMM.7]

When:

- NH_3N_rate (kg N/m^2): Rate of NH_3-N loss (kg/m^2) from manure, calculated in **[MN.AMM.5]**
- surface_area (m^2): total manure storage surface area

Calculate nitrous oxide emissions
_calculate_nitrous_oxide_emissions

N2O emissions (kg N2O-N) are based on the daily quantity of manure N loaded into the lagoon, and whether the lagoon is covered or uncovered. This method is based on Intergovernmental Panel on Climate Change, 2019; however, it should be noted that the original IPCC method is based on daily manure N excretion by animals, whereas the current method is based on manure N loading into storage, which may reflect upstream N losses from $_3$ emissions in housing, solid liquid separation, etc. The calculation is as follows:

$$N_2ON\ emissions\ (kg) = Received\_N * N_2O\ factor$$

[MN.NIT.1]

When:

- Received_N (kg): Quantity of manure total N loaded into storage on the current day
- N_2O factor: kg of N_2O-N emitted per kg of manure N added per day to storage, based on the following logic:
  - Cover type = crust OR cover; 0.005
  - Cover type = no cover; 0 (no N_2O emissions)

12.c Received, stored, and emptied outputs

Manure storages in RuFaS report two types of outputs to OutputManager each day: received manure and stored manure.

Received manure outputs represent the quantity of manure mass and nutrients added to the manure storage on a single day. No nutrient losses from gas or other emissions/losses are reflected in these output values.

Stored manure outputs represent the accumulated quantity of manure and nutrients present in storage on a single day. These values are the net quantity of mass/nutrients remaining each day after adding received manure values and subtracting any losses to gas emissions or other losses. In anaerobic lagoon processors, daily losses include CH_4, NH_3, and N_2O emissions. The order of operations in updating accumulated manure values is:

- Add received manure values to stored manure values
- Calculate gas emissions and total nutrient losses based on stored manure values
- Update stored manure values based on the day's nutrient losses. See the Manure composition update section for specific details on how nutrient gains and losses are accounted for on a daily timestep.

For anaerobic lagoons and all other storage processor types, the stored manure values (not received manure) are passed to the next processor in the chain (e.g. another storage, field application, export, etc.) when the storage time interval is complete.

Emptied manure
Manure may be removed from storage via requests made by the Crop and Soil module. The user specifies the days and years for manure removal (i.e. application), as well as the application type (liquid or solid) and quantity of N or P required for each application date within year. Note that these actions are the responsibility of the Crop and Soil module; more information on manure application inputs and methodology can be found in the Crop and Soil module documentation. When manure is removed from storage by the Crop and Soil module, emptied manure outputs report the quantity of manure and nutrients removed on that day, and Manure Stream attributes representing stored manure are updated accordingly to reflect post-removal amounts remaining in storage.

12.d Manure composition updates

Received manure:

In anaerobic lagoon processors, the following nutrient sources are represented in received manure values:

- ManureStream values, as received from the previous processor(s) in the manure management chain
- Precipitation water (kg), calculated in MN.STO.2 (if applicable), is added to the water value in ManureStream

Stored manure:

Below is a summary of updates to ManureStream variables representing the stored manure. Note that the formulas below may be a summarization of multiple steps detailed above, and are intended to provide an overview of what mass losses/gains are reflected in the value of each variable.

Equations in the table below (Calculation column) are in the format of: updated stored manure value = yesterday's stored manure value + today's manure value +/- XYZ. The updated stored manure values reflect the total quantity of manure/nutrients in storage on a single day after accounting for all gains/losses that occurred on that day. Received manure simply refers to the manure being loaded into the manure storage each day.

Table 73: Manure storage variable calculations

Variable	Units	Calculation
water	kg	stored manure water (kg) + received manure water (kg)
total_ammoniacal_nitrogen	kg	max(0, stored manure ammoniacal nitrogen (kg) + received ammoniacal nitrogen (kg) − NH_3N emissions (kg))
nitrogen	kg	stored manure nitrogen (kg) + received manure nitrogen (kg) − NH_3N emissions (kg) − N_2O-N emissions (kg)
phosphorus	kg	stored manure phosphorus (kg) + received manure phosphorus (kg)
potassium	kg	stored manure potassium (kg) + received manure potassium (kg)
ash	kg	stored manure ash (kg) + received manure ash (kg)
degradable_volatile_solids		stored degradable VS (kg) + received degradable VS (kg) − VSd loss (kg)
manure_non_degradable_volatile_-solids	kg	stored manure non-degradable VS (kg) + received manure non-degradable VS (kg) − manure VSnd loss (kg)
bedding_non_degradable_volatile_-solids	kg	stored bedding non-degradable VS (kg) + received bedding non-degradable VS (kg) − bedding VSnd loss (kg)
total_solids	kg	stored total solids (kg) + received total solids (kg) − VSd loss (kg) − VSnd loss (kg)
volume	m^3	stored volume (m^3) + Received volume (m^3) − $\frac{\text{VSd loss (kg)}+\text{VSnd loss (kg)}}{\text{SLURRY\_MANURE\_DENSITY}}$

13 Bedded Pack

13.a Introduction

Traditional dairy housing systems typically involve concrete flooring and limited bedding, which can lead to cow discomfort, suboptimal health, and laborious manure management. Bedded pack pens aim to address these issues by providing a soft, comfortable, and dry bedding surface for cows, enhancing their comfort and overall well-being.

Bedded packs fall into one of two management methods: a traditional bedded pack, or a compost bedded pack. See UW-Madison Dairyland Initiative Report for further detail. Though the names are often used interchangeably, note here that the primary difference between the two types is the presence or absence of active mixing, and the length of time between complete removal of the accumulated bedding/manure mixture from the pen.

Traditional bedded pack Traditional bedded packs, referred to here as simply "bedded packs", are typically bedded using straw. Fresh material is added daily without any mixing activity, which causes the manure/bedding mix (pack) to compact and become anaerobic. The pack material is typically removed every 4 to 6 weeks.

Compost Bedded Pack Compost bedded packs are typically bedded with sawdust, shavings, or other fine, dense, absorbent material. Bedding is added daily, accompanied by mixing of the pack material to promote aeration and aerobic decomposition (composting). This composting action leads to the production of heat and microbial activity that promotes breakdown of organic components and reductions in moisture content, resulting in a drier and more stable pack. The pack material is typically removed after several months.

Implementation in RuFaS
The two types of bedded pack are modeled differently in RuFaS as their decomposition conditions and thus emission profiles differ. Bedded packs, where mixing does not occur, promote anaerobic decomposition, whereas compost bedded packs promote aerobic decomposition through regular mixing and aeration of the pack. Sawdust or shavings bedded pens may be used in situations where pens are cleaned more frequently, e.g. weekly, but these pens are not considered to be bedded packs in RuFaS.

Classes

Table 74: List of classes for bedded pack.

Class	Description
CompostBeddedPackBarn(Storage)	bedded_pack.py

Required User Inputs

Table 75: Required inputs for the bedded pack section (refreshed_manure_management.json)

Variable	Definition (units)	Description
Name	none	Unique identifier of the specific bedded pack configuration used.
storage_time_period	days	The number of days that the manure/bedding pack accumulates in the pen before being removed and replaced with fresh bedding.
is_mixed	true/false	A boolean indicator for whether the bedded pack is routinely mixed to intentionally promote composting activity.

Other inputs

Instance(s) of ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane production potential, $m^3/kgVS$):

- water
- ammoniacal_nitrogen
- nitrogen
- phosphorus
- potassium
- ash
- manure_degradable_volatile_solids
- manure_non_degradable_volatile_solids
- bedding_non_degradable_volatile_solids
- total_solids
- mass (equal to sum of water and total solids)
- total volatile solids (equal to sum of degradable and non-degradable volatile solids)
- volume
- methane_production_potential

Expected Outputs

- ManureStream variables representing manure loaded (received) into storage each day, and accumulated manure after accounting for nutrient and mass gains/losses
- storage_methane (kg): Total mass of CH_4 emitted from the bedded pack each day.
- storage_ammonia_N (kg): Total mass of NH_3-N emitted from the bedded pack each day.
- storage_nitrous_oxide_N (kg): Total mass of N2O-N emitted from the bedded pack each day.
- storage_nitrogen_leached (kg): Total mass of N leached from the bedded pack each day. Leached N is assumed to be lost to the environment, and is not captured in runoff that may enter a manure storage.
- carbon_decomposition (kg): the total quantity of manure C lost through microbial degradation of volatile solids.

13.b Methodology

Calculate daily methane generation

Description: Calculates the daily mass of methane emitted from the bedded pack based on daily manure VS added to the bedded pack (through animal excretion and bedding addition) and an emission factor based on mixing activity and simulation average temperature (Hanson, Itle, and Edquist, 2024). Here and in other equations, 'daily' denotes the value associated with received manure added to the bedded pack on a specified simulation day. **Note that the quantity of volatile solids utilized in determination of CH_4 includes only manure-excreted volatile solids; bedding volatile solids are excluded.**

Table 76: Methane conversion factor values for bedded pack pens, based on presence or absence of mixing and average annual air temperature.

	Average air temperature (°C)				
Mixing, T/F	≤ 4.6 °C	4.7 - 5.8 °C	5.8 - 13.9 °C	14.0 - 25.1 °C	≥ 25.2 °C
Mixing	0.5	0.5	1.0	1.0	1.5
No mixing	21	26	37	41	74

$$methane\ (kg) = B_0 * MCF * tVS$$

[MN.MET.6]

When:

- B_0 = methane production potential (kg CH_4 per kg manure VS) of manure excreted onto the lot on a specified simulation day
- Constant Value: 0.24 m^3 CH_4/kg VS
- MCF = methane conversion factor (Table 22), based on simulation average ambient temperature

- tVS (kg) = daily mass (kg) of manure-excreted total VS in the bedded pack manure, received from ManureStream(s); note that bedding VS are not included in this value

Calculate carbon decomposition

Description: In addition to microbial processes that occur in anaerobic conditions, which generate primarily CH_4 and CO_2, carbon in the manure/bedding mixture is also degraded through aerobic microbial processes. This process is a function of substrate availability/degradability, temperature, moisture, aeration, and microbial population. This series of calculations is based on the IFSM composting simulation method (Bonifacio, Rotz, and Richard, 2017a; Bonifacio, Rotz, and Richard, 2017b). Simplifications/assumptions that have been made which diverge from the original method are explicitly noted below.

- First, we calculate the maximum decomposition rate per day, and decomposition rate of the slow fraction per day. We use the same equation for both rates, however, the temperature value used in calculating maximum decomposition rate is 60 ℃, versus 30 ℃ in calculating slow fraction degradation. The maximum decomposition rate value is set to 0.04195 and the slow fraction decomposition rate is set to 0.00846, but the equations and set values are shown below for reference.

$$max\_decomp\_rate = EFFECTIVE\_MICROBIAL\_DECOMP\_RATE * (1.066^{DECOMPOSITION\_TEMPERATURE-10} - 1.21^{DECOMPOSITION\_TEMPERATURE-50})$$

[MN.STO.4]

When:

  - EFFECTIVE_MICROBIAL_DECOMP_RATE (unitless): The effectiveness of microbial decomposition rate per day, set to 0.00237
  - DECOMPOSITION_TEMPERATURE: temperature of the inner compost layer, set to 60 $°C$ (reflective of temperature at which microbial growth, and thus decomposition, is maximized)

$$slow\_decomp\_rate = EFFECTIVE\_MICROBIAL\_DECOMP\_RATE * (1.066^{DEFAULT\_LAYER\_TEMPERATURE-10} - 1.21^{DEFAULT\_LAYER\_TEMPERATURE-50})$$

[MN.STO.5]

When:

  - EFFECTIVE_MICROBIAL_DECOMP_RATE (unitless): The effectiveness of microbial decomposition rate per day, set to 0.00237
  - DEFAULT_LAYER_TEMPERATURE: temperature of the pack layer, set to 30$°C$ Setting the layer temperature to a constant value is a simplification as manure pack temperature is not modeled dynamically at this time.

- Second, we calculate the carbon decomposition rate per day (calculate_carbon_decomposition_-rate). The value of this parameter is equal to 0.03876, but the equation and set values are included below for reference.

$$C\_decomp\_rate = (max\_decomp\_rate - slow\_decomp\_rate) * e^{FIRST\_ORDER\_DECAYING\_COEFFICIENT*(DEFAULT\_DAYS\_SINCE\_LAST\_MIXING - DEFAULT\_LAG\_TIME)} + slow\_decomp\_rate$$

[MN.STO.6]

When:

- max_decomp_rate and slow_decomp_rate calculated with **[MN.STO.4]**, set to 0.04195 and 0.00846, respectively
- FIRST_ORDER_DECAYING_COEFFICIENT: First-order decaying coefficient constant, set to 0.10
- DEFAULT_DAYS_SINCE_LAST_MIXING: number of days from the start of pack formation or last mixing event, set to 1 by default
- lag: lag time in days to reach maximum decomposition rate, set to 2

- Third, we calculate the anaerobic effect coefficient, related to the effect of the degree of aeration in the manure pack on decomposition. This value is set to 0.9664, but the equation and fixed values are provided below for reference.

$$anaerobic\_effect = \frac{oxygen\_mole\_fraction}{oxygen\_half\_saturation\_constant + oxygen\_mole\_fraction} * \frac{oxygen\_half\_saturation\_constant + oxygen\_ambient\_air_mole\_fraction}{oxygen\_ambient\_air_mole\_fraction} = \frac{0.15}{0.02+0.15} * \frac{0.02+0.21}{0.21} = 0.9664$$

When:

- oxygen_mole_fraction: mole fraction of oxygen in the air within the bedded pack, unitless, set at 0.15. This is a simplification as oxygen content of the manure pack is not currently modeled.
- oxygen_half_saturation_constant: the half-saturation constant, unitless, set at 0.02 by the original publication.
- oxygen_ambient_air_mole_fraction: the mole fraction of oxygen in ambient air, unitless, set at 0.21 (ambient air is approximately 21% oxygen).

- Fourth, we calculate total carbon in the manure/bedding pack available for decomposition. Here we make some assumptions on the carbon content of manure degradable vs. non-degradable volatile solids. Degradable volatile solids, which originate from fecal excretion by animals, are considered to be 50% carbon by weight (Larney, Ellert, and Olson, 2011). Non-degradable volatile solids, which originate primarily from bedding addition, are assumed to contain 35% carbon by weight. The total carbon available is the sum of these two quantities.

$$carbon\_from\_VS_d\ (kg) = degradable\_volatile\_solids * DEFAULT\_CARBON\_FRACTION\_AVAILABLE\_IN\_VSD$$

[MN.STO.7]

When:

- degradable_volatile_solids: The degradable volatile solids (kg) in the daily manure added to the bedded pack.
- DEFAULT_CARBON_FRACTION_AVAILABLE_IN_VSD: the carbon content (%) of manure degradable volatile solids, set to 50% by default.

$$carbon\_from\_VSnd\ (kg) = non\_degradable\_volatile\_solids * DEFAULT\_CARBON\_FRACTION\_AVAILABLE\_IN\_VSND$$

[MN.STO.8]

When:

- nondegradable_volatile_solids: The non-degradable volatile solids (kg) in the daily bedding and manure added to the bedded pack.
- DEFAULT_CARBON_FRACTION_AVAILABLE_IN_VSD: the carbon content (%) of manure degradable volatile solids, set to 35% by default.

$$total\_carbon\ (kg) = carbon\_from\_VSnd + carbon\_from\_VSd$$

[MN.STO.9]

- Finally, we calculate total carbon decomposition in kg/d using the coefficients and values calculated in the steps above (calculate_carbon_decomposition):

$$total\_carbon\_decomposition\ (kg) = (total\_carbon\ *\ C\_decomp\_rate\ *\ DEFAULT\_MOISTURE\_EFFECT\_MICROBIAL\_DECOMP\ *\ anaerobic\_effect)$$

[MN.STO.10]

When:

- total_carbon: total carbon available in manure pack (kg); **[MN.STO.9]**
- C_decomp_rate: carbon decomposition rate per day; **[MN.STO.6]**
- DEFAULT_MOISTURE_EFFECT_MICROBIAL_DECOMP: The effect of moisture on microbial decomposition, set at 0.65. This is a simplification as moisture content of the manure pack is not currently modeled.
- anaerobic_effect: the anaerobic effect coefficient, related to the effect of the degree of aeration in the manure pack on decomposition. Set to 0.9664 by default.

Calculate total VS Loss
_apply_dry_matter_loss
The quantity of total and volatile solids remaining in the accumulated bedded pack must be updated according to estimated CH_4 and C decomposition losses. To do this, we calculate the total loss of VS through CH_4 emission and C decomposition. Loss of mass through CH_4 emissions is assumed to be equal to the mass of CH_4 emitted. Manure volatile solids are assumed to be 50% C, therefore, to determine total mass loss through C decomposition, we divide the mass of C decomposition by 0.50. Importantly, volatile solids destruction is attributed only to manure-excreted volatile solids; bedding volatile solids destruction is not considered at this time.

$$total\_volatile\_solids\_loss(kg) = methane + \frac{total\_carbon\_decomposition}{0.50}$$

[MN.STO.11]

When:

- methane (kg): The daily methane loss, calculated with **[MN.MET.6]**, based on the daily quantity of manure VS added to the bedded pack
- total_carbon_decomposition (kg): quantity of C lost through microbial decomposition (kg), calculated with **[MN.STO.10]**, based on the daily quantity of manure VS added to the bedded pack

Calculate N loss to ammonia
_calculate_cbpb_ammonia_emission

Manure nitrogen being deposited and accumulating in the bedded pack results in NH_3 emissions. Here we utilize an emission factor based on mixing activity (Hanson, Itle, and Edquist, 2024) to estimate total kg of ammonia loss based on the daily manure N deposition by animals.

$$storage\_ammonia\_N\ (kg) = daily\_manure\_N\ (kg) * ammonia\_coefficient$$

[MN.AMM.8]

When:

- daily_maure_N: Daily kg of manure N added to the bedded pack
- ammonia_coefficient: kg of NH_3-N emitted per kg of manure N added per day to the bedded pack, based on the style of management:
  - Bedded pack (no mixing) = AMMONIA_EMISSION_COEFFICIENT_WITH_UNTILLED_-BEDDING (0.25)
  - Compost bedded pack (mixing) = AMMONIA_EMISSION_COEFFICIENT_WITH_TILLED_-BEDDING (0.50)

Calculate N loss to nitrous oxide
_calculate_cbpb_nitrous_oxide_emission

In addition to NH_3-N emissions, manure nitrogen deposition in the bedded pack also results in N2O emissions. Similar to NH_3, we estimate daily N_2O-N loss using an emissions factor from Hanson et al. (2024) based on the type of management used.

$$storage\_nitrous\_oxide\_N\ (kg) = daily\_manure\_N\ (kg) * nitrous\_oxide\_coefficient$$

[MN.NIT.1]

When:

- daily_maure_N: Daily kg of manure N added to the bedded pack
- ammonia_coefficient: kg of NH_3-N emitted per kg of manure N added per day to the bedded pack, based on the style of management:
  - Bedded pack (no mixing) = NITROUS_OXIDE_EMISSION_COEFFICIENT_WITH_UNTILLED_BEDDING (0.01)
  - Compost bedded pack (mixing) = NITROUS_OXIDE_EMISSION_COEFFICIENT_WITH_TILLED_BEDDING (0.07)

Calculate N loss to leaching
calculate_nitrogen_loss_to_leaching

Manure nitrogen deposited in bedded packs may also be lost to leaching. Leaching of manure N may occur when fecal and urinary N are converted to nitrate in the soil beneath the bedded pack, if the pack is not concrete, lined, or otherwise sealed. Nitrate can then be carried away via water movement through the subsoil. Similar to NH_3 and N_2O, we estimate daily leaching-N loss using an emissions factor from Hanson et al. (2024), which is not influenced by management of the bedded pack (i.e., mixing activity).

$$storage\_leached\_N\ (kg) = daily\_manure\_N\ (kg) * LEACHING\_COEFFICIENT$$

[MN.STO.12]

When:

- daily_maure_N: Daily kg of manure N added to the bedded pack
- LEACHING_COEFFICIENT: kg of N leached per kg of manure N added per day to the open lot; set at 0.035.

13.c Received, stored, and emptied outputs

Manure storages in RuFaS report two types of outputs to OutputManager each day: received manure and stored manure.

Received manure

Received manure outputs represent the quantity of manure mass and nutrients added to the manure storage on a single day. No nutrient losses from gas or other emissions/losses are reflected in these output values.

Stored manure

Stored manure outputs represent the accumulated quantity of manure and nutrients present in storage on a single day. These values are the net quantity of mass/nutrients remaining each day after adding received manure values and subtracting any losses to gas emissions or other losses. In bedded pack processors, daily losses include CH_4, NH_3, and N_2O emissions, and N leaching. The order of operations in updating accumulated manure values is:

- Add received manure values to stored manure values
- Calculate gas emissions and total nutrient losses based on stored manure values
- Update stored manure values based on the day's nutrient losses. See the Manure composition update section for specific details on how nutrient gains and losses are accounted for on a daily timestep.

For bedded pack and all other storage processor types, the stored manure values (not received manure) are passed to the next processor in the chain (e.g. another storage, field application, export, etc.) when the storage time interval is complete.

Emptied manure
Manure may be removed from storage via requests made by the Crop and Soil module. The user specifies the days and years for manure removal (i.e. application), as well as the application type (liquid or solid) and quantity of N or P required for each application date within year. Note that these actions are the responsibility of the Crop and Soil module; more information on manure application inputs and methodology can be found in the Crop and Soil module documentation. When manure is removed from storage by the Crop and Soil module, emptied manure outputs report the quantity of manure and nutrients removed on that day, and Manure Stream attributes representing stored manure are updated accordingly to reflect post-removal amounts remaining in storage.

13.d Manure composition updates

Received manure

Received manure simply refers to the manure being loaded into the manure storage each day (i.e., deposited in the bedded pack). In bedded pack processors, the following nutrient sources are represented in received manure values:

- ManureStream values, as received from the previous processor(s) in the manure management chain. In bedded pack processors, which are placed first in the manure management chain as there are no intermediary steps between animal excretion and the bedded pack, this ManureStream instance typically represents manure and bedding received directly from the Animal module.
- Daily precipitation volume/mass is **not** currently represented in received bedded pack manure.

Stored manure

Below is a summary of updates to ManureStream variables representing the stored manure. Note that the formulas below may be a summarization of multiple steps detailed above, and are intended to provide an overview of what mass losses/gains are reflected in the value of each variable.

Equations in the table below (Calculation column) are in the format of: updated stored manure value = yesterday's stored manure value + today's manure value +/- XYZ. The updated stored manure values reflect the total quantity of manure/nutrients in storage on a single day after accounting for all gains/losses that occurred on that day. Received manure simply refers to the manure being loaded into the manure storage each day.

Table 77: Calculated manure storage variables and their units

Variable	Units	Calculation
water	kg	Stored manure water (kg) + received manure water (kg)
total_ammoniacal_nitrogen	kg	max(0, stored manure ammoniacal N + received ammoniacal N − NH_3N emissions)
nitrogen	kg	Stored manure N + received manure N − NH_3-N − N_2O-N emissions
phosphorus	kg	Stored manure P + received manure P
potassium	kg	Stored manure K + received manure K
ash	kg	Stored manure ash + received manure ash
degradable_volatile_solids		stored degradable VS (kg) + received degradable VS (kg) − VSd loss (kg)
manure_non_degradable_volatile_-solids	kg	stored manure non-degradable VS (kg) + received manure non-degradable VS (kg) − VSnd loss (kg)
bedding_non_degradable_volatile_-solids	kg	stored bedding non-degradable VS (kg) + received bedding non-degradable VS (kg)
total_solids	kg	Stored TS + received TS − VS loss
volume	m^3	Stored volume + received volume − $\left(\frac{\text{VS loss}}{\text{SOLID\_MANURE\_DENSITY}}\right)$

14 Open Lot

14.a Introduction

Open or dry lot dairy systems are systems in which cows are housed outdoors in earthen or concrete pens. The open lot surface typically consists of natural or compacted soil covered with accumulated dry manure. Open lot pens often contain some form of shelter to provide shade and protection from harsh weather conditions, under which bedding may be applied, especially during winter. In addition to the dirt/manure pack lot, these systems also typically have a segregated feed feeding area, which may include a concrete apron that cows stand and deposit manure upon while eating. Manure deposited in this area may be managed differently from manure deposited on the lot surface.

To manage the manure on the lot surface, farmers commonly harrow the surface to spread out, break up, and mix fresh manure into the existing dry pack, to facilitate drying and create a standing and lying surface mainly consisting of dry manure solids. Harrowing is usually performed daily, which results in mixing of the soil/dry manure pack and fresh manure and urine. When cows deposit urine and manure on the lot surface, the dry climate and low humidity typical to regions where these systems are common facilitate rapid drying and ammonia (NH_3) volatilization from urine. Harrowing activity also promotes volatilization of NH_3 through mixing of fresh urine and manure. Though NH_3 losses may be high, the harrowing/spreading action also serves to create a lighter, more aerated pack that emits less methane (CH_4) than other manure storage methods, such as slurry or liquid manure storage; however, these same aerobic conditions can also lead to greater nitrous oxide (N_2O) emissions from the lot surface.

Manure is allowed to accumulate on the lot surface for several weeks to months, and may be either spread flat or partially piled in the pen during this period. Lots are cleaned out typically via scraping with large equipment 1-2 times per year. Manure removed from lots may be immediately field applied, stacked/piled until field application, composted, or managed in other ways. Of note is that, while the majority of manure on open lot dairies may be managed as a solid, open lot dairies with milking animals will also utilize a slurry or liquid manure storage to store, at a minimum, milking parlor waste. Feed apron waste, lot runoff, or other waste may also be stored as a liquid.

Implementation in RuFaS

In the Open Lot processor, we receive manure information from the animal module and manure management information from user inputs. Given weather data, we then calculate daily nutrient losses through CH_4, N_2O, and NH_3 emissions, as well as N leaching and C decomposition. We update the composition of the accumulated manure mix each day as manure and urine are added and nutrients are lost through the processes mentioned.

Important assumptions:

- Harrowing is assumed to be performed daily. At this time, differences in lot manure harrowing or piling frequency are not reflected in calculation of emissions/losses.
- Gas emissions in this processor are based on annual emissions factors that are extrapolated to a daily timestep, meaning that calculations are made based on daily excretion of manure VS or N, rather than accumulated quantities of the nutrients. With this, for N and NH_3 emissions for example, manure is assumed to lose all estimated NH_3-N on the day it is excreted. Manure N accumulation in lot manure is tracked, but accumulated manure N does not influence NH_3 emissions.

Classes

Table 78: Classes available in open lots.

	Description
Storage(OpenLot)	open_lot.py.

Required User Inputs

Table 79: Required inputs for the open lots section

Variable	Units	Description
storage_time_period	days	The interval in days that the open lot pen is cleaned out, i.e., the majority of manure, either already in piles or spread across the lot surface, is scraped or otherwise collected from the lot and removed from the pen.

Other inputs

Instance(s) of ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane potential, $m^3/kgVS$):

- water
- ammoniacal_nitrogen
- nitrogen
- phosphorus
- potassium
- ash
- manure_degradable_volatile_solids
- manure_non_degradable_volatile_solids
- bedding_non_degradable_volatile_solids
- total_solids
- mass (equal to sum of water and total solids)
- total volatile solids (equal to sum of degradable and non-degradable volatile solids)
- volume
- methane_production_potential

Expected Outputs

- ManureStream variables representing manure loaded (received) into storage each day, and accumulated manure after accounting for nutrient and mass gains/losses
- storage_methane (kg): Total mass of CH_4 emitted from the open lot manure each day.
- storage_ammonia_N (kg): Total mass of NH_3-N emitted from the open lot manure each day.
- storage_nitrous_oxide_N (kg): Total mass of N2O-N emitted from the open lot manure each day.
- storage_nitrogen_leached (kg): Total mass of N leached from the open lot manure each day. Leached N is assumed to be lost to the environment, and is not captured in lot runoff that may enter a manure storage.
- carbon_decomposition (kg): the total quantity of manure C lost through microbial degradation of volatile solids.

14.b Methodology

Calculate methane conversion factor
calculate_ifsm_methane_emission
Calculates the methane conversion factor for open lot manure given the ambient air temperature using an equation from IFSM (Rotz, Corson, et al., 2023). This method is an adaptation of the IPCC, 2006 tier 2 approach. Note that the MCF cannot be lower than 0.

$$MCF = max(0, \frac{0.0625 * T - 0.25}{100})$$

[MN.MET.5]

When:

- T = ambient air temperature, 0 to 30°C

Calculate daily methane generation
calculate_ifsm_methane_emission

Calculates the daily mass of methane emitted from the open lot based on daily manure volatile solids (VS) added to lot manure (through animal excretion and bedding addition) and the emission factor calculated in **[MN.MET.5]**. **Note that the quantity of volatile solids utilized in determination of CH_4 includes only manure-excreted volatile solids; bedding volatile solids are excluded.** Here and in other equations, 'daily' denotes the value associated with received manure added to the open lot on a specified simulation day.

$$methane\ (kg) = B_0 * MCF * tVS * METHANE\_FACTOR$$

[MN.MET.6]

When:

- B_0 = methane production potential (kg CH_4 per kg manure VS) of manure excreted onto the lot on a specified simulation day
- MCF = methane conversion factor calculated in **[MN.MET.5]**, based on daily ambient temperature

- tVS (kg) = daily mass (kg) of manure-excreted volatile solids in the open lot pen, received from ManureStream(s); bedding volatile solids are not included in this value
- METHANE_FACTOR = 0.67; unit conversion factor for CH_4 volume to mass (kg).

Calculate carbon decomposition
calculate_carbon_decomposition

In addition to microbial processes that occur in anaerobic conditions, which generate primarily CH_4 and CO_2, open lot manure C is also degraded through aerobic microbial processes. This process is a function of substrate availability/degradability, temperature, moisture, aeration, and microbial population. This series of calculations is based on the IFSM composting simulation method, which is described in detail in Bonifacio, Rotz, and Richard, 2017a and Bonifacio, Rotz, and Richard, 2017b. Simplifications/assumptions that have been made which diverge from the original method are explicitly noted below.

- First, we calculate the maximum decomposition rate per day, and decomposition rate of the slow fraction per day. We use the same equation for both rates, however, the temperature value used in calculating maximum decomposition rate is $60°C$, versus $30°C$ in calculating slow fraction degradation. The maximum decomposition rate value is set to 0.04195 and the slow fraction decomposition rate is set to 0.00846, but the equations and set values are shown below for reference.

$$max\_decomp\_rate = EFFECTIVE\_MICROBIAL\_DECOMP\_RATE * 1.066^{(DECOMPOSITION\_TEMPERATURE-10)-1.21^{(}DECOMPOSITION\_TEMPERATURE-50)}$$

[MN.STO.4]

When:

- EFFECTIVE_MICROBIAL_DECOMP_RATE (unitless): The effectiveness of microbial decomposition rate per day, set to 0.00237
- DECOMPOSITION_TEMPERATURE: temperature of the inner compost layer, set to $60°C$ (reflective of temperature at which microbial growth, and thus decomposition, is maximized)

$$slow\_decomp\_rate = EFFECTIVE\_MICROBIAL\_DECOMP\_RATE * 1.066^{(DEFAULT\_LAYER\_TEMPERATURE-10)} - 1.21^{(DEFAULT\_LAYER\_TEMPERATURE-50)}$$

[MN.STO.5]

When:

- EFFECTIVE_MICROBIAL_DECOMP_RATE (unitless): The effectiveness of microbial decomposition rate per day, set to 0.00237
- DECOMPOSITION_TEMPERATURE: temperature of the inner compost layer, set to $30°C$. Setting the layer temperature to a constant value is a simplification as manure pack temperature is not modeled dynamically at this time.

- Second, we calculate the carbon decomposition rate per day (calculate_carbon_decomposition_-rate). The value of this parameter is equal to 0.03876, but the equation and set values are included below for reference.

$$C\_decomp\_rate = (max\_decomp\_rate - slow\_decomp\_rate) * e^{FIRST\_ORDER\_DECAYING\_COEFFICIENT*(DEFAULT_DAYS\_SINCE\_LAST\_MIXING - DEFAULT\_LAG\_TIME)} + slow\_decomp_rate$$

[MN.STO.6]

When:

  - max_decomp_rate and slow_decomp_rate calculated with **[MN.STO.4]**, set to 0.04195 and 0.00846, respectively
  - FIRST_ORDER_DECAYING_COEFFICIENT: First-order decaying coefficient constant, set to 0.10
  - DEFAULT_DAYS_SINCE_LAST_MIXING: number of days from the start of pack formation or last mixing event, set to 1 by default
  - lag: lag time in days to reach maximum decomposition rate, set to 2

- Third, we calculate the anaerobic effect coefficient, related to the effect of the degree of aeration in the manure pack on decomposition. This value is set to 0.9664, but the equation and fixed values are provided below for reference.

$$anaerobic\_effect = \frac{oxygen\_mole\_fraction}{(oxygen\_half\_saturation\_constant + oxygen\_mole\_fraction)} * \frac{(oxygen\_half\_saturation\_constant + oxygen\_ambient\_air\_mole\_fraction)}{oxygen\_ambient\_air\_mole\_fraction} = \frac{0.15}{(0.02+0.15)} * \frac{(0.02+0.21)}{0.21} = 0.9664$$

When:

  - oxygen_mole_fraction: mole fraction of oxygen in the air within the lot manure pack, unitless, set at 0.15. This is a simplification as oxygen content of the manure pack is not currently modeled.
  - oxygen_half_saturation_constant: the half-saturation constant, unitless, set at 0.02 by the original publication.
  - oxygen_ambient_air_mole_fraction: the mole fraction of oxygen in ambient air, unitless, set at 0.21 (ambient air is approximately 21% oxygen).

- Fourth, we calculate total carbon in the manure/bedding pack available for decomposition. Here we make some assumptions on the carbon content of manure degradable vs. non-degradable volatile solids. Degradable volatile solids, which originate from fecal excretion by animals, are considered to be 50% carbon by weight (Larney, Ellert, and Olson, 2011). Non-degradable volatile solids, which originate primarily from bedding addition, are assumed to contain 35% carbon by weight. The total carbon available is the sum of these two quantities.

$$carbon\_from\_VSd\ (kg) = degradable\_volatile\_solids * DEFAULT\_CARBON\_FRACTION\_AVAILABLE\_IN\_VSD$$

[MN.STO.7]

When:

- degradable_volatile_solids: The degradable volatile solids (kg) in the daily manure added to the open lot.
- DEFAULT_CARBON_FRACTION_AVAILABLE_IN_VSD: the carbon content (%) of manure degradable volatile solids, set to 50% by default.

$$carbon\_from\_VSnd\ (kg) = non\_degradable\_volatile\_solids * \\ DEFAULT\_CARBON_FRACTION\_AVAILABLE\_IN\_VSND$$

[MN.STO.8]

When:

- non_degradable_volatile_solids: The non-degradable volatile solids (kg) in the daily manure added to the open lot.
- DEFAULT_CARBON_FRACTION_AVAILABLE_IN_VSD: the carbon content (%) of manure non-degradable volatile solids, set to 35% by default.

$$total\_carbon\ (kg) = carbon\_from\_VS_{nd} + carbon\_from\_VS_d$$

[MN.STO.9]

- Finally, we calculate total carbon decomposition in kg/d using the coefficients and values calculated in the steps above (_apply_dry_matter_loss):

$$total\_carbon\_decomposition\ (kg) = (total\_carbon * C\_decomp\_rate * \\ DEFAULT\_MOISTURE\_EFFECT\_MICROBIAL\_DECOMP * anaerobic\_effect)$$

[MN.STO.10]

When:

- total_carbon: total carbon available in manure pack (kg); **[MN.STO.9]**
- C_decomp_rate: carbon decomposition rate per day; **[MN.STO.10]**
- DEFAULT_MOISTURE_EFFECT_MICROBIAL_DECOMP: The effect of moisture on microbial decomposition, set at 0.65. This is a simplification as moisture content of the manure pack is not currently modeled.
- anaerobic_effect: the anaerobic effect coefficient, related to the effect of the degree of aeration in the manure pack on decomposition. Set to 0.9664 by default.

Calculate total VS Loss

The quantity of total and volatile solids remaining in the accumulated bedded pack must be updated according to estimated CH_4 and C decomposition losses. To do this, we calculate the total loss of VS through CH_4 emission and C decomposition. Loss of mass through CH_4 emissions is assumed to be equal to the mass of CH_4 emitted. Manure volatile solids are assumed to be 50% C, therefore, to determine total mass loss through C decomposition, we divide the mass of C decomposition by 0.50. Importantly, volatile solids destruction is attributed only to manure-excreted volatile solids; bedding volatile solids destruction is not considered at this time.

$$total\_volatile\_solids\_loss(kg) = methane + \frac{total\_carbon\_decomposition}{0.50}$$

[MN.STO.11]

When:

- methane (kg): The daily methane loss, calculated with **[MN.MET.6]**, based on the daily quantity of manure VS added to the open lot
- total_carbon_decomposition (kg): quantity of C lost through microbial decomposition (kg), calculated with **[MN.STO.10]**, based on the daily quantity of manure VS added to the open lot

Calculate N loss to ammonia

_calculate_cbpb_ammonia_emission

Manure nitrogen being deposited and accumulating in the bedded pack results in NH_3 emissions. Here we utilize an emission factor based on mixing activity (Hanson, Itle, and Edquist, 2024) to estimate total kg of ammonia loss based on the daily manure N deposition by animals.

$$storage\_ammonia\_N\ (kg) = daily\_manure\_N * ammonia\_coefficient$$

[MN.AMM.8]

When:

- daily_maure_N: Daily kg of manure N added to the open lot
- AMMONIA_EMISSION_COEFFICIENT_IN_OPEN_LOTS: kg of NH_3-N emitted per kg of manure N added per day to the open lot; set at 0.36.

Calculate N loss to nitrous oxide

_calculate_cbpb_nitrous_oxide_emission

In addition to NH_3-N emissions, manure nitrogen deposition in the bedded pack also results in N_2O emissions. Similar to NH_3, we estimate daily N2O-N loss using an emissions factor from Hanson, Itle, and Edquist, 2024 based on the type of management used.

$$storage\_nitrous\_oxide\_N\ (kg) = daily\_manure\_N * nitrous\_oxide\_coefficient$$

[MN.NIT.1]

When:

- daily_maure_N: Daily kg of manure N added to the open lot
- NITROUS_OXIDE_COEFFICIENT_IN_OPEN_LOTS: kg of N_2-O-N emitted per kg of manure N added per day to the open lot; set at 0.02.

Calculate N loss to leaching
calculate_nitrogen_loss_to_leaching

Manure nitrogen deposited in bedded packs may also be lost to leaching. Leaching of manure N may occur when fecal and urinary N are converted to nitrate in the soil beneath the bedded pack, if the pack is not concrete, lined, or otherwise sealed. Nitrate can then be carried away via water movement through the subsoil. Similar to NH_3 and N_2O, we estimate daily leaching-N loss using an emissions factor from Hanson, Itle, and Edquist, 2024, which is not influenced by management of the bedded pack (i.e., mixing activity).

$$storage\_leached\_N\ (kg) = daily\_manure\_N\ (kg) * LEACHING\_COEFFICIENT$$

[MN.STO.12]

When:

- daily_maure_N: Daily kg of manure N added to the open lot
- LEACHING_COEFFICIENT: kg of N leached per kg of manure N added per day to the open lot; set at 0.035.

14.c Received, stored, and emptied outputs

Manure storages in RuFaS report two types of outputs to OutputManager each day: received manure and stored manure.

Received manure

Received manure outputs represent the quantity of manure mass and nutrients added to the manure storage on a single day. No nutrient losses from gas or other emissions/losses are reflected in these output values.

Stored manure

Stored manure outputs represent the accumulated quantity of manure and nutrients present in storage on a single day. These values are the net quantity of mass/nutrients remaining each day after adding received manure values and subtracting any losses to gas emissions or other losses. In open lot processors, daily losses include CH_4, NH_3, and N_2O emissions, and N leaching. The order of operations in updating accumulated manure values is:

- Add received manure values to stored manure values
- Calculate gas emissions and total nutrient losses based on stored manure values
- Update stored manure values based on the day's nutrient losses. See the Manure composition update section for specific details on how nutrient gains and losses are accounted for on a daily timestep.

For open lot and all other storage processor types, the stored manure values (not received manure) are passed to the next processor in the chain (e.g. another storage, field application, export, etc.) when the storage time interval is complete.

Emptied manure
Manure may be removed from storage via requests made by the Crop and Soil module. The user specifies the days and years for manure removal (i.e. application), as well as the application type (liquid or solid) and quantity of N or P required for each application date within year. Note that these actions are the responsibility of the Crop and Soil module; more information on manure application inputs and methodology can be found in the Crop and Soil module documentation. When manure is removed from storage by the Crop and Soil module, emptied manure outputs report the quantity of manure and nutrients removed on that day, and Manure Stream attributes representing stored manure are updated accordingly to reflect post-removal amounts remaining in storage.

14.d Manure composition updates

Received manure

Received manure simply refers to the manure being loaded into the manure storage each day (i.e., deposited on the lot). In open lot processors, the following nutrient sources are represented in received manure values:

- ManureStream values, as received from the previous processor(s) in the manure management chain. In open lot processors, which are placed first in the manure management chain as there are no intermediary steps between animal excretion and the open lot, this ManureStream instance typically represents manure and bedding received directly from the Animal module.
- Daily precipitation volume/mass is **not** currently represented in received open lot manure.

Stored manure

Below is a summary of updates to ManureStream variables representing the stored manure. Note that the formulas below may be a summarization of multiple steps detailed above, and are intended to provide an overview of what mass losses/gains are reflected in the value of each variable.

Equations in the table below (Calculation column) are in the format of: updated stored manure value = yesterday's stored manure value + today's manure value +/- XYZ. The updated stored manure values reflect the total quantity of manure/nutrients in storage on a single day after accounting for all gains/losses that occurred on that day. Received manure simply refers to the manure being loaded into the manure storage each day.

Table 80: Calculated manure storage variables and their units

Variable	Units	Calculation
water	kg	Stored manure water (kg) + received manure water (kg)
total_ammoniacal_nitrogen	kg	$\max(0,$ stored manure ammoniacal N + received ammoniacal N − NH_3N emissions$)$
nitrogen	kg	Stored manure N + received manure N − NH_3-N − N_2O-N emissions
phosphorus	kg	Stored manure P + received manure P
potassium	kg	Stored manure K + received manure K
ash	kg	Stored manure ash + received manure ash

Continued on next page

Variable	Units	Calculation
degradable_volatile_solids		stored degradable VS (kg) + received degradable VS (kg) − VSd loss (kg)
manure_non_degradable_volatile_-solids	kg	stored manure non-degradable VS (kg) + received manure non-degradable VS (kg) − VSnd loss (kg)
bedding_non_degradable_volatile_-solids	kg	stored bedding non-degradable VS (kg) + received bedding non-degradable VS (kg)
total_solids	kg	Stored TS + received TS − VS loss
volume	m^3	Stored volume + received volume − $\left(\frac{\text{VS loss}}{\text{SOLID\_MANURE\_DENSITY}}\right)$

[1] The “Max(0,)” notation prevents the ammoniacal N value from becoming negative. This is especially important early in a simulation when accumulated manure quantities that are very small compared to the fixed surface area value can lead to high ammonia emissions.

[2] degradable_volatile_solids_frac = received manure degradable_volatile_solids / received manure total_volatile_solids

15 Composting

15.a Introduction

Composting is a method of treating manure in which solid manure, generally with bedding or other carbon-rich material added, is purposefully managed in order to promote microbial decomposition. This naturally occurring decomposition process results in losses of both water and organic matter, resulting in a smaller volume and mass of manure the farm is required to store and handle. Composting also helps to stabilize manure by killing pathogens and weed seeds via heat generated from the decomposition process, and reduced odors. Finished compost is a useful end-product (like all well-managed manure) that can be land-applied, sold, or otherwise exported.
To promote optimal conditions for microbial decomposition, compost requires airflow through the pile to prevent anaerobic conditions from forming. While turning is the traditional way to achieve this, several methods to accomplish the necessary aeration of compost exist on farms today.

Implementation in RuFaS The following composting methods are represented in RuFaS:

- Static pile: material to be composted is piled and not turned/moved throughout the composting process. Aeration is maintained using either bulking agents to create airspace in the pile, or through forced air introduction via pipes.
- Intensive windrow: material to be composted is piled into long rows that are regularly turned to redistribute moisture, nutrients, and air through the pile.
- Passive windrow: material to be composted is piled into long rows, but is not regularly turned, and instead relies on passive air diffusion to aerate the pile.
- In-vessel: material to be composted is loaded into a large vessel, often a drum, which is mechanically rotated to promote uniform air circulation and heating.

At this time, turning frequency, pile composition, composting duration, C:N ratio, and other factors do not directly affect emissions or nutrient loss calculations. Only the general composting method and the basic composition (kg of N, volatile solids, etc.) of material added to the pile is factored into this submodule's calculations at this time.

Classes

Table 81: Classes available in composting.

	Description
Storage(Composting)	composting.py.

Required User Inputs

Table 82: Required inputs for the composting section (refreshed_manure_management.json)

Variable	Definition (units)	Description
Name	none	Unique identifier of the specific handler configuration used.
Storage_time_period	days	The number of days that manure is stored between emptying events. At the end of this interval, the manure storage is emptied completely.
composting_type	static pile, passive windrow, intensive windrow, or in-vessel	The type of method used for composting

Other inputs

Instance(s) of ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane production potential, $m^3/kgVS$):

- water
- ammoniacal_nitrogen
- nitrogen
- phosphorus
- potassium
- ash
- manure_degradable_volatile_solids
- manure_non_degradable_volatile_solids
- bedding_non_degradable_volatile_solids
- total_solids
- mass (equal to sum of water and total solids)
- total volatile solids (equal to sum of degradable and non-degradable volatile solids)
- volume
- methane_production_potential

Expected Outputs

- ManureStream variables representing manure loaded (received) into storage each day, and accumulated manure after accounting for nutrient and mass gains/losses
- storage_methane (kg): Total mass of CH_4 emitted from the compost each day.
- storage_ammonia_N (kg): Total mass of NH_3-N emitted from the compost each day.

- storage_nitrous_oxide_N (kg): Total mass of N_2O-N emitted from the compost each day.
- storage_nitrogen_leached (kg): Total mass of N leached from the compost each day. Leached N is assumed to be lost to the environment, and is not captured in runoff that may enter a manure storage.
- carbon_decomposition (kg): the total quantity of manure C lost through microbial degradation of volatile solids.

15.b Methodology

Calculate daily methane generation

_calculate_composting_methane_emissions

Calculates the daily mass of methane emitted from the compost based on manure volatile solids (VS) added to compost (from animal excretion and bedding addition) and an emission factor based on composting method and simulation average temperature described in Table 1 (Hanson, Itle, and Edquist, 2024). **Note that the quantity of volatile solids utilized in determination of CH_4 includes only manure-excreted volatile solids; bedding volatile solids are excluded.** Here and in other equations, 'daily' denotes the value associated with received manure added to the open lot on a specified simulation day.

Table 83: Methane conversion factor values for compost pens, based on composting method and average annual air temperature.

	Average air temperature (°C)		
Composting method	0 - 10 °C	10 - 18 °C	>18 °C
In-vessel	0.5	0.5	0.5
Static pile	1.0	2.0	2.5
Intensive windrow	0.5	1.0	1.5
Passive windrow	1.0	2.0	2.5

$$methane\ (kg) = B_o * MCF * tVS$$

[MN.MET.6]

When:

- B_0 = methane production potential (kg CH_4 per kg manure VS) of manure excreted into the bedded pack on a specified simulation day
- MCF = methane conversion factor, based on annual ambient temperature
- tVS (kg) = daily mass (kg) of manure-excreted volatile solids in the open lot pen, received from ManureStream(s); bedding volatile solids are not included in this value
- METHANE_FACTOR = 0.67; unit conversion factor for CH_4 volume to mass (kg).

Calculate carbon decomposition

calculate_carbon_decomposition

Carbon in the compost material is degraded through primarily aerobic microbial processes. This process is a function of substrate availability/degradability, temperature, moisture, aeration, and microbial population. This series of calculations is based on the IFSM composting simulation method, which is described in detail in Bonifacio, Rotz, and Richard, 2017a and Bonifacio, Rotz, and Richard, 2017b. Simplifications/assumptions that have been made which diverge from the original method are explicitly noted below.

- First, we calculate the maximum decomposition rate per day, and decomposition rate of the slow fraction per day. We use the same equation for both rates, however, the temperature value used in calculating maximum decomposition rate is $60°C$, versus the actual ambient air temperature in calculating slow fraction degradation. The maximum decomposition rate value is set to 0.04195, but the equation is shown below for reference.

$$max\_decomp\_rate = EFFECTIVE\_MICROBIAL\_DECOMP\_RATE * (1.066^{(DECOMPOSITION\_TEMPERATURE-10)} - 1.21^{(DECOMPOSITION\_TEMPERATURE-50)})$$

[MN.STO.4]

  - EFFECTIVE_MICROBIAL_DECOMP_RATE (unitless): The effectiveness of microbial decomposition rate per day, set to 0.00237
  - DECOMPOSITION_TEMPERATURE: temperature of the inner compost layer, set to 60∘C (reflective of temperature at which microbial growth, and thus decomposition, is maximized)

$$slow\_decomp\_rate = EFFECTIVE\_MICROBIAL\_DECOMP\_RATE * 1.066^{(daily\_temperature-10)} - 1.21^{(daily\_temperature-50)}$$

[MN.STO.5]

  - EFFECTIVE_MICROBIAL_DECOMP_RATE (unitless): The effectiveness of microbial decomposition rate per day, set to 0.00237
  - daily_temperature: average ambient air temperature on a single day (∘C).

- Second, we calculate the carbon decomposition rate per day (calculate_carbon_decomposition_-rate). The value of this parameter is equal to 0.03876, but the equation and set values are included below for reference.

$$C\_decomp\_rate = (max\_decomp\_rate - slow\_decomp\_rate) * e^{(FIRST\_ORDER\_DECAYING\_COEFFICIENT*(DEFAULT\_DAYS\_SINCE\_LAST\_MIXING-DEFAULT\_LAG\_TIME))} + slow\_decomp\_rate$$

[MN.STO.6]

  - max_decomp_rate and slow_decomp_rate calculated with **[MN.STO.4]** and **[MN.STO.5]**

- FIRST_ORDER_DECAYING_COEFFICIENT: First-order decaying coefficient constant, set to 0.10
- DEFAULT_DAYS_SINCE_LAST_MIXING: number of days from the start of composting or last turning/mixing event, set to 1 (i.e., assuming daily mixing)
- lag: lag time in days to reach maximum decomposition rate, set to 2

- Third, we calculate the anaerobic effect coefficient, related to the effect of the degree of aeration in the manure pack on decomposition. This value is set to 0.9664, but the equation and fixed values are provided below for reference.

$$anaerobic\_effect = (\frac{oxygen\_mole\_fraction}{(oxygen\_half\_saturation\_constant+oxygen\_mole\_fraction)}) * (\frac{(oxygen\_half\_saturation\_constant+oxygen\_ambient\_air\_mole\_fraction)}{oxygen\_ambient\_air\_mole\_fraction}) = (\frac{0.15}{(0.02+0.15)}) * (\frac{(0.02+0.21)}{0.21}) = 0.9664$$

  - oxygen_mole_fraction: mole fraction of oxygen in the air within the windrow, unitless, set at 0.15. This is a simplification as oxygen content of the manure pack is not currently modeled.
  - oxygen_half_saturation_constant: the half-saturation constant, unitless, set at 0.02 by the original publication.
  - oxygen_ambient_air_mole_fraction: the mole fraction of oxygen in ambient air, unitless, set at 0.21 (ambient air is approximately 21% oxygen).

- Fourth, we calculate total carbon in the compost material available for decomposition. Here we make some assumptions on the carbon content of manure degradable vs. non-degradable volatile solids. Degradable volatile solids, which originate from fecal excretion by animals, are considered to be 50% carbon by weight (Larney, Ellert, and Olson, 2011). Non-degradable volatile solids, which originate primarily from bedding addition, are assumed to contain 35% carbon by weight. The total carbon available is the sum of these two quantities.

$$carbon\_from\_VSd\ (kg) = degradable\_volatile\_solids * DEFAULT\_CARBON\_FRACTION\_AVAILABLE\_IN\_VSD$$

[MN.STO.7]

  - degradable_volatile_solids: The degradable volatile solids (kg) in the daily manure added to the compost.
  - DEFAULT_CARBON_FRACTION_AVAILABLE_IN_VSD: the carbon content (%) of manure degradable volatile solids, set to 50% by default.

$$carbon\_from\_VSnd\ (kg) = non\_degradable\_volatile\_solids * DEFAULT\_CARBON\_FRACTION\_AVAILABLE\_IN\_VSND$$

[MN.STO.8]

- non_degradable_volatile_solids: The non-degradable volatile solids (kg) in the daily bedding and manure added to the compost.
- DEFAULT_CARBON_AVAILABLE_IN_VSND: the carbon content (%) of manure non-degradable volatile solids, set to 35% by default.

$$total\_carbon\ (kg) = carbon\_from\_VSnd + carbon\_from\_VSd$$

[MN.STO.9]

- Finally, we calculate total carbon decomposition in kg/d using the coefficients and values calculated in the steps above (calculate_carbon_decomposition):

$$total\_carbon\_decomposition\ (kg) = (total\_carbon * C\_decomp\_rate * \\ DEFAULT\_MOISTURE\_EFFECT\_MICROBIAL\_DECOMP * anaerobic\_effect)$$

[MN.STO.10

- total_carbon: total carbon available in compost (kg); **[MN.STO.9]**
- C_decomp_rate: carbon decomposition rate per day; **[MN.STO.6]**
- DEFAULT_MOISTURE_EFFECT_MICROBIAL_DECOMP: The effect of moisture on microbial decomposition, set at 0.65. This is a simplification as moisture content of the compost is not currently modeled.
- anaerobic_effect: the anaerobic effect coefficient, related to the effect of the degree of aeration in the compost on decomposition. Set to 0.9664 by default.

Calculate total VS loss
_apply_dry_matter_loss

The quantity of total and volatile solids remaining in the compost each day must be updated according to estimated CH_4 and C decomposition losses. To do this, we calculate the total daily loss of VS through CH_4 emission and C decomposition. Loss of mass through CH_4 emissions is assumed to be equal to the mass of CH_4 emitted. Manure volatile solids are assumed to be 50% C, therefore, to determine total mass loss through C decomposition, we divide the mass of C decomposition by 0.50.

$$total\_volatile\_solids\_loss\ (kg) = methane + (\frac{total\_carbon\_decomposition}{0.50})$$

[MN.STO.11

- methane (kg): The daily methane loss, calculated with **[MN.MET.6]**, based on the daily quantity of manure VS added to the compost
- total_carbon_decomposition (kg): quantity of C lost through microbial decomposition (kg), calculated with **[MN.STO.10]**, based on the daily quantity of manure VS added to the compost

Table 84: NH_3-N emission factors for compost by composting method.

Composting method	**kg NH_3-N emitted per kg of N added to compost**
In-vessel	0.45
Static pile	0.50
Intensive windrow	0.50
Passive windrow	0.45

Calculate N loss to ammonia
_calculate_composting_ammonia_emissions

Description: Manure nitrogen being added to and accumulating in the compost results in NH_3 emissions. Here we utilize daily manure N addition to compost and an emission factor based on composting method (Hanson, Itle, and Edquist, 2024) to estimate total kg of NH_3-N loss.

$$storage\_ammonia\_N\ (kg) = daily\_manure\_N\ (kg) * ammonia\_coefficient$$

[MN.AMM.8

- daily_maure_N: Daily kg of manure N added to compost
- ammonia_coefficient: kg of NH_3-N emitted per kg of manure N added per day to compost, based on the composting method

Calculate N loss to nitrous oxide
_calculate_nitrous_oxide_emissions

Description: In addition to NH_3-N emissions, nitrogen added to and accumulating in the compost also results in N_2O emissions. Similar to NH_3, here we utilize daily manure N addition to compost and an emission factor based on composting method (Hanson, Itle, and Edquist, 2024) to estimate total kg of $N_2O - N$ loss.

Composting method	**kg NH_3-N emitted per kg of N added to compost**
In-vessel	0.006
Static pile	0.010
Intensive windrow	0.005
Passive windrow	0.005

$$storage\_nitrous\_oxide\_N\ (kg) = daily\_manure\_N\ (kg) * nitrous\_oxide\_coefficient$$

[MN.NIT.1

- daily_maure_N: daily kg of manure N added to the compost
- nitrous_oxide_coefficient: kg of $N_2O - N$ emitted per kg of manure N added per day to compost, based on the composting method

Calculate N loss to leaching
calculate_nitrogen_loss_to_leaching

Description: Nitrogen in compost may also be lost to leaching. Leaching of compost N may occur when fecal and urinary N are converted to nitrate in the soil beneath the compost, if the compost is not contained in a vessel. Nitrate can then be carried away via water movement through the subsoil. Similar to NH_3 and N_2O, we use daily manure N addition to compost and an emission factor based on composting method (Hanson, Itle, and Edquist, 2024) to estimate total kg of leaching N loss.

Table 85: N_2O-N emission factors for compost by composting method.

Composting method	kg NH_3-N emitted per kg of N added to compost
In-vessel	0.00
Static pile	0.06
Intensive windrow	0.06
Passive windrow	0.04

$$storage\_leached\_N\ (kg) = daily\_manure\_N\ (kg) * leaching\_coefficient$$

[MN.STO.12]

- daily_maure_N: daily kg of manure N added to the bedded pack
- leaching_coefficient: kg of N leached per kg of manure N added per day to compost, based on the composting method

15.c Received, stored, and emptied outputs

Received manure
Received manure outputs represent the quantity of manure mass and nutrients added to the manure storage on a single day. No nutrient losses from gas or other emissions/losses are reflected in these output values.

Stored manure
Stored manure outputs represent the accumulated quantity of manure and nutrients present in storage on a single day. These values are the net quantity of mass/nutrients remaining each day after adding received manure values and subtracting any losses to gas emissions or other losses. In composting processors, daily losses include CH_4, NH_3, N leaching, and N2O emissions. The order of operations in updating accumulated manure values is:

1. Add received manure values to stored manure values
2. Calculate gas emissions and total nutrient losses based on received manure values
3. Update stored manure values based on the day's nutrient losses. See the Manure composition update section for specific details on how nutrient gains and losses are accounted for on a daily timestep.

For composting and all other storage processor types, the stored manure values (not received manure) are passed to the next processor in the chain (e.g. another storage, field application, export, etc.) when the storage time interval is complete.

Emptied manure
Manure may be removed from storage via requests made by the Crop and Soil module. The user specifies the days and years for manure removal (i.e. application), as well as the application type (liquid or solid) and quantity of N or P required for each application date within year. Note that these actions are the responsibility of the Crop and Soil module; more information on manure application inputs and methodology can be found in the Crop and Soil module documentation. When manure is removed from storage by the Crop and Soil module, emptied manure outputs report the quantity of manure and nutrients removed on that day, and Manure Stream attributes representing stored manure are updated accordingly to reflect post-removal amounts remaining in storage.

15.d Manure composition updates

Received manure

In composting processors, the following nutrient sources are represented in received manure values: ManureStream values, as received from the previous processor(s) in the manure management chain. For composting processors, these ManureStream instance(s) typically represent an accumulated quantity of manure from an open lot or bedded pack, daily/weekly cleanout of the manure/bedding mix generated by other pen types, or daily addition of separated manure solids. Daily precipitation volume/mass is not currently represented in received manure added to compost (i.e., precipitation falling each day is not represented in the received manure outputs representing nutrient/mass/water additions to compost each day).

Stored manure

Below is a summary of updates to ManureStream variables representing the stored manure. Note that the formulas below may be a summarization of multiple steps detailed above, and are intended to provide an overview of what mass losses/gains are reflected in the value of each variable.
Equations in the table below (Calculation column) are in the format of: updated stored manure value = yesterday's stored manure value + today's manure value +/- XYZ. The updated stored manure values reflect the total quantity of manure/nutrients in storage on a single day after accounting for all gains/losses that occurred on that day.

Table 86: Manure storage variable calculations

Variable	Units	Calculation
water	kg	Stored manure water (kg) + received manure water (kg)
total_ammoniacal_nitrogen	kg	max(0, stored ammoniacal nitrogen (kg) + received ammoniacal nitrogen (kg) − NH_3N emissions (kg))
nitrogen	kg	stored nitrogen (kg) + received nitrogen (kg) − NH_3-N emissions (kg) − N_2O-N emissions (kg)
phosphorus	kg	stored phosphorus (kg) + received phosphorus (kg)
potassium	kg	stored potassium (kg) + received potassium (kg)

Continued on next page

Variable	Units	Calculation
ash	kg	stored ash (kg) + received ash (kg)
degradable_volatile_solids		stored degradable VS (kg) + received degradable VS (kg) − VSd loss (kg)
manure_non_degradable_volatile_solids	kg	stored manure non-degradable VS (kg) + received manure non-degradable VS (kg) − VSnd loss (kg)
bedding_non_degradable_volatile_solids	kg	stored bedding non-degradable VS (kg) + received bedding non-degradable VS (kg)
total_solids	kg	stored total solids (kg) + received total solids (kg) − total volatile solids loss (kg)
volume	m^3	stored volume (m^3) + received volume (m^3) − $\frac{\text{total volatile solids loss (kg)}}{\text{SOLID\_MANURE\_DENSITY}}$

[1] The “Max(0,) notation prevents the ammoniacal N value from becoming negative. This is especially important early in a simulation when accumulated manure quantities that are very small compared to the fixed surface area value can lead to high ammonia emissions.

[2] degradable_volatile_solids_frac = received manure degradable_volatile_solids / received manure total_volatile_solids

16 Daily spread

16.a Introduction

Daily spread is a method of manure management in which manure is collected daily (or several times per week) directly from animal housing areas and field-applied. This manure management strategy may be more labor intensive but can greatly reduce the quantity of manure a farm must store.

Implementation in RuFaS
Daily spread in RuFaS is implemented as essentially a "blank" manure processor. It receives manure but does not estimate any gas emissions, nutrient losses, or changes to manure composition. **Note that this section covers strictly the Manure Module daily spread functionality; the actual daily application of manure from the manure module occurs in the Crop and Soil module and is thus covered in the Crop and Soil documentation.**

Classes

Table 87: List of classes for daily spread.

Class	Description
DailySpread(Storage)	daily_spread.py

Required User Inputs

Table 88: Required inputs for the daily spread section (refreshed_manure_management.json)

Variable	Definition (units)	Description
Name	none	Unique identifier of the specific daily spread configuration used.

Other inputs

Instance(s) of ManureStream for each manure stream defined by the user that represent the attributes of the manure in the specific manure stream. ManureStream instances include the following variables (all in kg except for volume, m^3 and manure methane production potential, $m^3/kgVS$):

- water
- ammoniacal_nitrogen
- nitrogen
- phosphorus
- potassium
- ash
- manure_degradable_volatile_solids
- manure_non_degradable_volatile_solids
- bedding_non_degradable_volatile_solids
- total_solids

- mass (equal to sum of water and total solids)
- total volatile solids (equal to sum of degradable and non-degradable volatile solids)
- volume
- methane_production_potential

Expected Outputs

- ManureStream variables representing manure received by daily spread processor each day

17 Manure to Soil and Crop Module Connection

17.a Introduction

On most dairies, the amount of manure applied to a field is determined by the amount of nitrogen (N) and phosphorus (P) needed by the crop to support production, or by the maximum amount of that nutrient that can be applied based on environmental regulations. These amounts are often established in the farm's detailed nutrient management plan and updated with crop yields and manure quantities and compositions.

Often, application of manure alone cannot fulfill all crop nutrient needs due to a mismatch between the N:P ratio required by the crop and the N:P ratio of the available manure. The manure application is therefore referred to as being **limited** by the nutrient whose requirement is met first through manure application.

In RuFaS, the user provides the desired amount of manure N and P for each manure application and specifies whether the model should supplement the application with imported manure or fertilizer if the exact amounts of both N and P are not met by the manure available on-farm. If both N and P cannot be supplied by manure in storage on the day of application, the total amount of the limiting nutrient (N or P) is used to determine the amount of the other nutrient, the total manure mass, and other accompanying nutrients (e.g., volatile solids (VS), potassium (K)) based on manure composition. This information is then passed to the Soil and Crop Module for application of nutrients to fields and to the Manure Module for removal of stored nutrients.

17.b Implementation in RuFaS

In RuFaS, manure accumulates in storage until it is removed for field application, transferred to another storage, or exported off-farm. This accumulated manure represents the manure available for field application. Manure applications, referred to as *manure nutrient requests*, originate from the Crop and Soil Module and are scheduled by year and Julian day. Each request contains a specified quantity of manure N and P.

Nutrient requests may be made for either solid or liquid manure. Manure is designated as liquid or solid based on storage type: slurry storage and anaerobic lagoons contain liquid manure, while all other storage types contain solid manure.

Exchange of the information about the manure nutrient application request and the availability of manure nutrients is executed in the simulation engine to ensure independence of the Manure Module and the Soil and Crop Module. The flow of information between the Manure and Soil and Crop Modules for a manure application is illustrated in Figure 4.

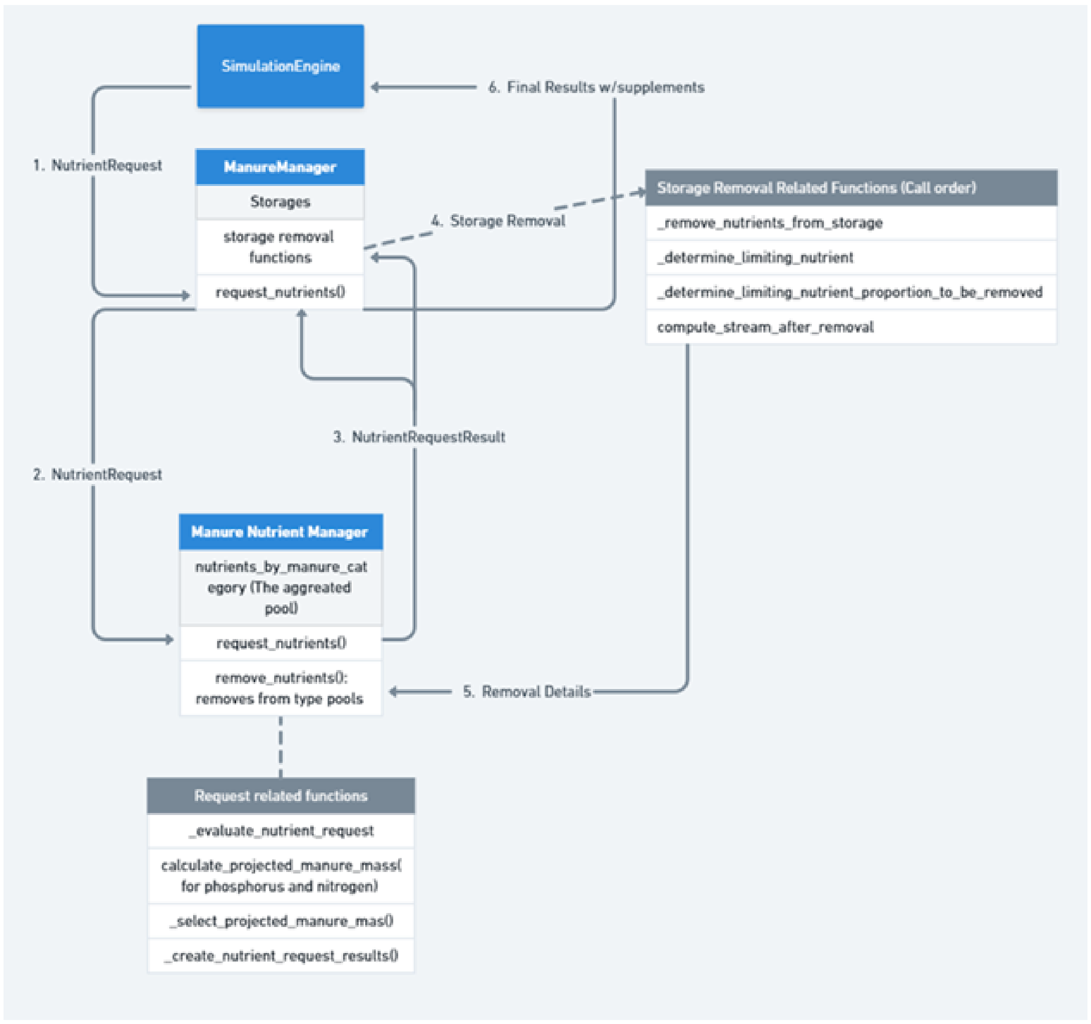

SimulationEngine
6. Final Results w/supplements
1. NutrientRequest
ManureManager
Storages
storage removal functions
request_nutrients()
4. Storage Removal
Storage Removal Related Functions (Call order)
_remove_nutrients_from_storage
_determine_limiting_nutrient
_determine_limiting_nutrient_proportion_to_be_removed
compute_stream_after_removal
3. NutrientRequestResult
2. NutrientRequest
Manure Nutrient Manager
nutrients_by_manure_category (The aggregated pool)
request_nutrients()
remove_nutrients(): removes from type pools
5. Removal Details
Request related functions
_evaluate_nutrient_request
calculate_projected_manure_mass(for phosphorus and nitrogen)
_select_projected_manure_mas()
_create_nutrient_request_results()

Figure 8

Each day, the SimulationEngine calls the generate_daily_manure_applications() method to check with the FieldManager to see if there is a manure application scheduled and, when there is, pass the manure nutrient request created by the FieldManager to the ManureManager. The ManureManager, in turn, checks with the ManureNutrientManager to see if the manure nutrient request can be fulfilled via the request_nutrients() in the ManureManager, which calls the evaluate_nutrient_request() function from the ManureNutrientManager class. The ManureNutrientManager then evaluates the nutrient request, determines the limiting nutrient to be used, and calculates the amount of each manure nutrient and the total mass that can be provided by the manure in storage using the following logic. At this time, nutrients from all storages are pooled by ManureNutrientManager to determine the total quantity of nutrients available. Once the nutrient request is returned, an identical proportion (not amount) of manure is removed from all storages. In other words, the user cannot specify at this time which storage(s) should be pulled from for a manure application.

17.c Methodology

Step 1: Determination of the Limiting Nutrient

Determine the nutrient that manure storage removal will be based on using the composition of N and P in the manure nutrient pool. This step is executed by evaluate_nutrient_request:

$$\text{manure\_mass\_N} = \frac{\text{mass\_N\_requested}}{\text{manure\_nitrogen\_frac}} \tag{1}$$

$$\tag{2}$$

$$\text{manure\_mass\_P} = \frac{\text{mass\_P\_requested}}{\text{manure\_phosphorus\_frac}} \tag{3}$$

The projected manure mass is then calculated as:

$$\text{projected manure mass} = \min\left(\text{mass\_N\_requested}, \text{mass\_P\_requested}\right) \tag{4}$$

[MN.MSC.1]

where:

- mass_N_requested is the mass of manure (kg) required to meet the requested mass of N.
- mass_P_requested is the mass of manure (kg) required to meet the requested mass of P.

The limiting nutrient is the nutrient (N or P) with a smaller projected mass required to meet the nutrient request. The projected manure mass based on the limiting nutrient is used to determine the removal of manure from the manure storages.

Step 2: Proportion of Limiting Nutrient Removed
Using the limiting nutrient identified in Step 1 and considering the total quantity of that nutrient available in the aggregated ManureNutrient pool, the proportion of the limiting nutrient to be removed from each storage is calculated as:

$$\text{nutrient\_proportion\_to\_be\_removed} = \min\left(\frac{\text{mass\_requested\_nutrient}}{\text{manure\_nutrient\_available}}, 1\right) \tag{5}$$

[MN.MSC.2]

Step 3: Removal of Nutrients from Individual Storages
Next, the amount of the limiting nutrient and projected mass of manure to be removed from the aggregated ManureNutrient pool are determined. If there is sufficient manure available to fulfill the nutrient request, the quantity of the limiting nutrient removed will be equal to the requested amount; otherwise, the amount removed will be lower than the requested amount.
For each nutrient and each storage, the nutrient removal is calculated as:

$$\text{nutrient\_to\_be\_removed\_from\_storage} = \text{nutrient\_in\_storage} \times \text{nutrient\_proportion\_to\_be\_removed} \tag{6}$$

[MN.MSC.3]

where:

- nutrient_to_be_removed_from_storage is the mass of the nutrient removed from a specific manure storage (kg).
- nutrient_in_storage is the mass of the nutrient in the storage prior to manure application (kg).

The ManureNutrientManager then passes the result of the manure nutrient request with the total amount of the limiting nutrient to be supplied and the ManureManager calculates the mass of manure and nutrients to be removed from each individual storage. The ManureManager returns the final results of the manure application to the SimulationEngine, including:

- Total nitrogen mass supplied (kg)
- Total phosphorus mass supplied (kg)
- Total manure mass supplied (kg)
- Total dry matter supplied (kg)
- Dry matter fraction of the manure (%)
- A Boolean indicator of whether the nutrient request was fulfilled

After the nutrient request is partially or fully fulfilled, the corresponding quantities of mass and nutrients are subtracted from each individual manure storage.

18 Key Constants

Table 89: The key constants for all equations in the Manure Module

Variable	Value	Definition
MANURE_DAMPING_FACTOR	0.65	Fixed damping factor applied to the air temperature amplitude.
MANURE_TEMPERATURE_LAG	30	Lag constant representing delayed thermal response of manure temperature relative to air temperature (days).
ANAEROBIC_LAGOON_MANURE_-RETENTION	0.1	Fraction of stored manure retained in anaerobic lagoon when storage interval is reached.
ACTIVATION_ENERGY	81,000 J/mol	Apparent activation energy of methanogenesis in dairy manure.
DEFAULT_STORED_MANURE_PH	7.5	Default pH of manure in slurry storage or anaerobic lagoon.
FREEBOARD_CONSTANT	1.2	20% volume allowance above max volume if surface area not user-defined.
DEPTH_CONSTANT	4.572	Depth of slurry/liquid manure storage used for surface area calculation (m).
PRECIPITATION_CONSTANT	0.25	Annual precipitation constant for determining storage surface area (m).
MANURE_CONVERSION_CONSTANT	0.1175	Factor to estimate manure volume per cow per day (m^3).
METHANE_DESTRUCTION_EFFICIENCY	0.81	Percent methane destroyed with cover and flare system.
VS_TO_METHANE_LOSS_RATIO	6.665 kg	Mass ratio of CO_2+CH_4 to CH_4 from storage (Petersen et al., 2024).
NATURAL_LOG_ARRHENIUS_CONSTANT	30.6 g CH_4/kg VS/h	Log of Arrhenius parameter for methane emissions from stored manure.
STORAGE_RESISTANCE	23.1	Default resistance to ammonia volatilization (s/m).
SLURRY_MANURE_DENSITY	990 kg/m^3	Default density of manure as excreted.
AMMONIA_EMISSION_COEFFICIENT_-UNTILLED	0.25	Ammonia emission coefficient (no mixing).
AMMONIA_EMISSION_COEFFICIENT_-TILLED	0.5	Ammonia emission coefficient (with mixing).

Continued on next page

Table continued from previous page

Variable	**Value**	**Definition**
DEFAULT_DAYS_SINCE_LAST_MIXING	1 d	Days since last mixing, for C decomposition.
NITROUS_OXIDE_EMISSION_UNTILLED	0.01	N_2O-N emitted per kg manure N/day (no mixing).
NITROUS_OXIDE_EMISSION_TILLED	0.07	N_2O-N emitted per kg manure N/day (with mixing).
HOUSING_SPECIFIC_CONSTANT	260.0 s/m	Default constant for ammonia emissions from housing.
DEFAULT_PH_FOR_HOUSING_AMMONIA	7.7	Default pH for manure on housing floors.
MILKING_FRESH_WATER_USE_RATE	30 L/animal/day	Milking water use rate per animal.
WATER_DENSITY_KG_PER_M3	0.997 kg/m^3	Default water density.
CARBON_DIOXIDE_MOLAR_MASS	44.01 g/mol	Molar mass of CO_2.
CARBON_DIOXIDE_TO_METHANE_RATIO	4–6	Volumetric ratio of CO_2 to CH_4 during digestion.
IDEAL_GAS_LAW_R	0.0821 L atm/mol K	Ideal gas law constant.
METHANE_MOLAR_MASS	16.04 g/mol	Molar mass of CH_4.
TAN_INCREASE_FACTOR	1.60	TAN increase from anaerobic digestion (unitless).
DEFAULT_LAYER_TEMPERATURE	30 °C	Default layer temperature for decomposition.
DECOMPOSITION_TEMPERATURE	60 °C	Temperature for peak microbial activity.
DEFAULT_CARBON_FRACTION_VSD	0.5	Carbon content of degradable VS.
DEFAULT_CARBON_FRACTION_VSND	0.35	Carbon content of non-degradable VS.
DEFAULT_MOISTURE_EFFECT	0.65	Moisture effect on microbial decomposition.
DEFAULT_LAG_TIME	2 d	Lag time for C decomposition.
EFFECTIVE_MICROBIAL_DECOMP_RATE	0.00237	Microbial decomposition rate (unitless).
FIRST_ORDER_DECAY_COEFFICIENT	0.01	First-order decay coefficient.
LEACHING_COEFFICIENT	–	N leached per kg manure N/day.
DEFAULT_DAYS_SINCE_LAST_HARROW	1 d	Days since last harrow event.
ACHIEVABLE_METHANE_EMISSION	0.24 m^3 CH_4/kg VS	Achievable methane generation.
SOLID_MANURE_DENSITY	700 kg/m^3	Default solid manure density.

Continued on next page

Table continued from previous page

Variable	Value	Definition
LIQUID_MANURE_DENSITY	1000 kg/m^3	Default liquid manure density.

19 References

References

Aguirre-Villegas, H. A., R. A. Larson, and M. A. Sharara (2019). “Anaerobic digestion, solid-liquid separation, and drying of dairy manure: Measuring constituents and modeling emission”. In: *Science of The Total Environment* 696, p. 134059. DOI: 10.1016/J.SCITOTENV.2019.134059. URL: https://doi.org/10.1016/J.SCITOTENV.2019.134059.

Bonifacio, H. F., C. A. Rotz, and T. L. Richard (2017a). “A Process-Based Model for Cattle Manure Compost Windrows: Part 1. Model Description”. In: *Transactions of the ASABE* 60.3, pp. 877–892.

– (2017b). “A Process-Based Model for Cattle Manure Compost Windrows: Part 2. Model Performance and Application”. In: *Transactions of the ASABE* 60.3, pp. 893–913.

Bucklin, R. A. et al. (2009). “Environmental Temperatures in Florida Dairy Housing”. In: *Applied Engineering in Agriculture* 25.5, pp. 727–735. DOI: 10.13031/2013.28851.

Chianese, D. S., C. A. Rotz, and T. L. Richard (2009). “Simulation of methane emissions from dairy farms to assess greenhouse gas reduction strategies”. In: *Transactions of the ASABE* 52.4, pp. 1313–1323. URL: https://elibrary.asabe.org/abstract.asp?aid=27781.

Elsgaard, L., A. B. Olsen, and S. O. Petersen (2016). “Temperature response of methane production in liquid manures and co-digestates”. In: *Science of The Total Environment* 539, pp. 78–84. DOI: 10.1016/j.scitotenv.2015.07.145. URL: https://www.sciencedirect.com/science/article/pii/S0048969715304861.

Fernández, Y. B. et al. (2015). “Biological carbon dioxide utilisation in food waste anaerobic digesters”. In: *Water Research* 87, pp. 467–475. DOI: 10.1016/j.watres.2015.06.011. URL: https://doi.org/10.1016/j.watres.2015.06.011.

Fournel, S. et al. (2019). “Production of recycled manure solids for bedding in Canadian dairy farms: I. Solid–liquid separation”. In: *Journal of Dairy Science* 102, pp. 1832–1846. DOI: 10.3168/jds.2018-15330.

Hanson, W. L., C. Itle, and K. Edquist (2024). *Quantifying Greenhouse Gas Fluxes in Agriculture and Forestry: Methods for Entity-Scale Inventory*. Forthcoming or gray literature if no publisher listed. USDA or relevant institution.

Hegg, R. O., R. E. Larson, and J. A. Moore (1981). “Mechanical liquid-solid separation in beef, dairy and swine waste slurries”. In: *Transactions of the ASAE* 24.1, pp. 159–163. DOI: 10.13031/2013.34204.

Intergovernmental Panel on Climate Change (2019). *2019 Refinement to the 2006 IPCC Guidelines for National Greenhouse Gas Inventories*. Accessed: 2024-05-17. URL: https://www.ipcc-nggip.iges.or.jp/public/2019rf/index.html.

IPCC (2006). *Guidelines for National Greenhouse Gas Inventories*. https://www.ipcc-nggip.iges.or.jp/public/2006gl/. Accessed 2025-04-17.

Jørgensen, K. and L. S. Jensen (2009). “Chemical and biochemical variation in animal manure solids separated using different commercial separation technologies”. In: *Bioresource Technology* 100, pp. 3088–3096. DOI: 10.1016/j.biortech.2009.01.056.

Larney, F. J., B. H. Ellert, and A. F. Olson (2011). “Carbon, ash and organic matter relationships for feedlot manures and composts”. In: *Canadian Journal of Soil Science* 85, pp. 261–264. DOI: 10.4141/S04-060.

Mukhtar, Saqib, John M. Sweeten, and Brent W. Auvermann (1999). *Solid-liquid separation of animal manure and wastewater*. Tech. rep. Publication L-5340. Texas A&M AgriLife Extension Service.

Natural Resources Conservation Service (NRCS) (2017). *Conservation Practice Standard Code 359: Waste Treatment Lagoon*. Accessed: 2024-05-17. URL: https://www.nrcs.usda.gov/sites/default/files/2022-10/Waste_Treatment_Lagoon_359_CPS_Oct_2017.pdf.

Petersen, S. O. et al. (2024). “In-vitro method and model to estimate methane emissions from liquid manure management on pig and dairy farms in four countries”. In: *Journal of Environmental Management* 353, p. 120233. URL: https://www.sciencedirect.com/science/article/pii/S0301479724002196.

Rotz, C. A., M. S. Corson, et al. (2023). *The Integrated Farm System Model. Reference Manual, Version 4.7*. Tech. rep. Pasture Systems and Watershed Management Research Unit, USDA-ARS.

Rotz, C. A. and J. Oenema (2006). “Predicting management effects on ammonia emissions from dairy and beef farms”. In: *Transactions of the ASABE* 49.4, pp. 1139–1149. URL: https://elibrary.asabe.org/abstract.asp?aid=21731&t=2&redir=&redirType=.

Sommer, S. G. et al. (2022). “Model for calculating ammonia emission from stored animal liquid manure”. In: *Biosystems Engineering* 223, pp. 41–55.

Sommer, S. G., S. O. Petersen, and H. B. Moller (2004). “Algorithms for calculating methane and nitrous oxide emissions from manure management”. In: *Nutrient Cycling in Agroecosystems* 69, pp. 143–154. DOI: 10.1023/B:FRES.0000029678.25083.fa. URL: https://link.springer.com/article/10.1023/b:fres.0000029678.25083.fa.

Varma, V. S. et al. (2021). “Dairy and swine manure management–Challenges and perspectives for sustainable treatment technology”. In: *Science of The Total Environment* 778, p. 146319. DOI: 10.1016/j.scitotenv.2021.146319.

Part IV
Soil and Crop Module

Contents

List of Figures

List of Tables

Contents

20 Module Introduction

The Soil and Crop Module simulates the daily changes in soil composition and crop growth based on nutrient availability, weather and soil types. The underlying biophysical processes are drawn from existing models of soil and plant biogeochemistry including the SWAT model, Daycent, and SurPhos. The management routines execute the daily biogeochemical processes and direct simulation of the user-provided management practices including application of manure or synthetic fertilizer, tillage, planting and harvesting. As part of the whole farm biophysical simulation in RuFaS, the Soil and Crop Module receives manure from the Manure Module and sends harvested crop biomass to the Feed Storage Module.

21 Field Management

21.a Introduction

The Field Manager (FieldManager) is analogous to the agronomist or field manager on a real dairy farm. It coordinates and manages all the different routines and processes that are executed in Soil and Crop Module across any number of fields. To start with, the field manager interprets and integrates all of the user's inputs about field management, crops, and soils to create a set of fields that each have their specified field and soil properties as well as schedules for each of the key management operations (manure application, fertilizer application, tillage, planting, and harvesting).

Once the fields and their schedules are set up, the FieldManager works through daily updates for each field. These daily updates include:

- Gathering the weather conditions for that day
- Checking if there is a manure application needed so that it can coordinate with Manure Module to provide the manure for the application
- Calling each field's daily update methods
- Recording the harvested crops and sending them to the Feed Storage Module

The FieldManager is also responsible for calling functions to record and reset any annual data records or processes for each field. Thus, the FieldManager is responsible for receiving and generating information from the user and other parts of the model and coordinating the simulation within the Soil and Crop Module.

Setting up Field Management RuFaS can simulate as many fields as the user wishes to specify and requires a unique input file for each field. However, individual fields can share soil, crop, and management schedule inputs so that the user can simulate multiple fields with, for example, the same soil configuration but different crop and management schedules or different crops with the same manure application schedules without having to replicate the inputs for fields with shared characteristics.

Field Management Initialization The FieldManager class, implemented in field_manager.py, has a series of functions to help package the user-provided configuration data and management schedule information required to initialize each field. These functions all start with the prefix "set_up" and are called by the wrapper function "set_up_field()." There is only one instance of the FieldManager class as it is used to set up and coordinate processes across all the fields. The following functions within the FieldManager class gather information and call other functions to help initialize key data classes required for initialization of each field:

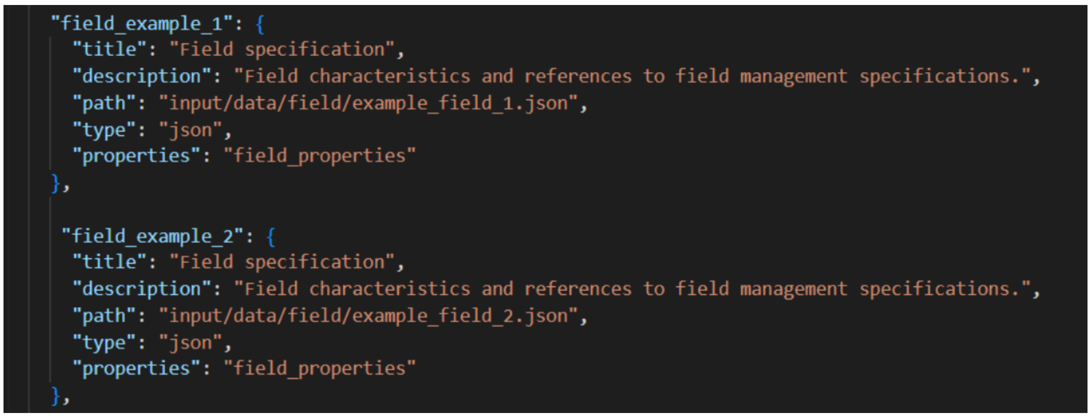
- "setup_field_data": This function sets up the field data parameters and returns an instance of the FieldData class populated with the values from the field_configuration_data
- "setup_soil_data": This function sets up a Soil instance that itself contains a SoilData instance configured to the user-provided specifications
- "setup_soil_layer_data": This function sets up a soil LayerData instance configured with provided data that is added to the SoilData instance

See the Individual Field Management section below for more information about the FieldData Class and the Soil submodule documentation for information related to the soil data classes.

The next group of "set_up" functions in FieldManager help initialize the management schedules for each field. These functions call initialization routines for a schedule class that is used to generate an instance of a schedule with all the information that is specific to each type of management practice for each field. These schedule classes translate the user input into detailed schedules that can be read and interpreted by the FieldManager and instances of RuFaS Fields on a daily basis to determine the operations that need to be simulated each day. The schedule set up functions are:

- "set_up_fertilizer_events": This function generates fertilizer application event schedules based on the user-provided fertilizer schedule and returns a fertilizer schedule as well as a dictionary with the available fertilizer mixes
- "set_up_manure_events": This function generates manure application event schedules based on the user-provided manure application schedules and returns a list of the planned manure applications.
- "set_up_tillage_events": This function generates tillage event schedules based on the user-provided tillage schedules and returns a list of the planned tillage events.
- "set_up_crop_events": This function generates all planting and harvest events based on the user-provided crop rotation schedule and returns a list of planting events and a list of the harvesting events that correspond to the planting events.
- "set_up_crop_schedules": This function is called by set_up_crop_events() and returns a single list of all the cropping schedules.

Outputs for each field can be identified by the character string used to define the field input blob in the metadata file.
For example, in the below excerpt from a metadata input file:



```
"field_example_1": {
  "title": "Field specification",
  "description": "Field characteristics and references to field management specifications.",
  "path": "input/data/field/example_field_1.json",
  "type": "json",
  "properties": "field_properties"
},

 "field_example_2": {
  "title": "Field specification",
  "description": "Field characteristics and references to field management specifications.",
  "path": "input/data/field/example_field_2.json",
  "type": "json",
  "properties": "field_properties"
},
```

Figure 9: Example of the metadata input file.

there are 2 different fields named "field_example_1" and "field_example_2" so the outputs generated by the first field would take the form:
Class.function.variable.field='field_example_1'
Where the pre-fix Class.function.variable is determined by the class, function that produced the variable, and the variable being reported. As a specific illustration, the output variable for the daily surface residue from "field_example_1" would be:
FieldDataReporter.send_field_daily_variables.current_residue.field='field_example_1'

Field Setup and Initialization When directed by the FieldManager, the Field class initializes a new Field object with the user-provided information that is packaged and shared by the FieldManager. These inputs include things like the field location, size, irrigation practices, pointers to the input files with management schedules, and options for turning crop growth stressors on and off.

Required Inputs

Table 90: Required user inputs for the FieldManager section.

Section	Variable	Definition	Default	Min	Max
Field	field_size	Size of the field (ha)	N/A	0	NA
	absolute_latitude	The absolute latitude of the center of this field.(degrees).	43.5	0	90
	longitude	The longitude of the center of this field (degrees).	-89.4	-180	180
	watering_amount_in_-liters	Amount of water to be applied as irrigation at the end of a specified interval (L).	0	0	N/A
	watering_interval	Number of days that make up the irrigation interval between irrigation events (d).	0	0	N/A
	simulate_water_stress	Whether water stress affects crops grown in this field.	TRUE	N/A	N/A
	simulate_temp_stress	Whether temperature stress affects crops grown in this field.	TRUE	N/A	N/A
	simulate_nitrogen_stress	Whether nitrogen stress affects crops grown in this field.	TRUE	N/A	N/A
	simulate_phosphorus_-stress	Whether phosphorus stress affects crops grown in this field.	TRUE	N/A	N/A
Crop Management	crop_species	Name of the crop being grown.	Must be selected from names of configurations defined in the crop configuration input file.		
	harvest_days	Julian day(s) of year to harvest	N/A	1	366

Continued on next page

Table continued from previous page

Section	Variable	Definition	Default	Min	Max
	harvest_years	Calendar years in which the harvesting occurs.	N/A	1	
	harvest_operations	Operation(s) with which this crop will be harvested.	-	• harvest_-only • harvest_-kill • kill_only	
	harvest_type	Whether the crop uses scheduled harvests or optimal harvests (“scheduled”, “optimal”).	scheduled		
	planting_days	Julian day(s) of year to plant.	N/A	1	366
	planting_years	Calendar years in which the planting occurs.	N/A	1	
	pattern_repeat	Number of times that this crop schedule should be repeated.	0	0	N/A
	planting_skip	Number of years to be skipped between planting schedule repetitions.	0	0	N/A
	harvesting_skip	Number of years to be skipped between harvest schedule repetitions.	0	0	N/A
Fertilizer Management	name	Name of the fertilizer mix.			
	N	Fraction of nitrogen contained in the fertilizer mix by mass.	1	0	1
	P	Fraction of phosphorus contained in the fertilizer mix by mass.	1	0	1

Continued on next page

Table continued from previous page

Section	Variable	Definition	Default	Min	Max
	K	Fraction of potassium contained in the fertilizer mix by mass.	1	0	1
	ammonium_fraction	Fraction fertilizer nitrogen that is ammonium. All non-ammonium nitrogen is assumed to be nitrate.	0.5	0	1
Fertilizer Schedule	mix_names	List of the mix names that will be used for the corresponding fertilizer application.			
	years	List of years in which fertilizer will be applied.		1	
	days	List of days on which fertilizer will be applied.		1	366
	nitrogen_masses	List of minimum nitrogen masses that the corresponding fertilizer applications should contain (kg).		0	
	phosphorus_masses	List of minimum phosphorus masses that the corresponding fertilizer applications should contain (kg).		0	
	potassium_masses	List of minimum potassium masses that the corresponding fertilizer applications should contain (kg).		0	
	application_depths	List of depths at which the fertilizer is injected into the soil (mm).		0	
	surface_remainder_-fractions	List of fractions of fertilizer which remain on the soil surface when applied via injection.		0	1
	pattern_repeat	Number of times that this fertilizer application schedule should be repeated.	0	0	

Continued on next page

Table continued from previous page

Section	Variable	Definition	Default	Min	Max
	pattern_skip	Number of years to be skipped between schedule repetitions.	0	0	
Manure Application	years	List of years in which tillage will occur.		1	
	days	List of days on which tilling will occur.		1	366
	tillage_depths	List of depths that the corresponding tillage applications reach (mm).		0.1	
	incorporation_fractions	List of fractions of surface pools that get incorporated into the soil during applications.		0	1
	mixing_fractions	List of fractions of soil layer pools that are available to mix into other soil layers.		0	1
	implements	List of tillage implements which will be used to execute the tillage operations.	-		• subsoiler • moldboard-plow • coulter-chisel-plow • disk-harrow • cultivator • seedbed-conditioner

Continued on next page

Table continued from previous page

Section	Variable	Definition	Default	Min	Max
	pattern_repeat	Number of times that this tillage schedule should be repeated.	0	0	
	pattern_skip	Number of years to be skipped between schedule repetitions.	0	0	
	years	List of years in which manure will be applied.		1	
	days	List of days on which manure will be applied.		1	366
	nitrogen_masses	List of minimum nitrogen masses that the corresponding manure applications should contain (kg).		0	
	phosphorus_masses	List of minimum phosphorus masses that the corresponding manure applications should contain (kg).		0	
	potassium_masses	List of minimum potassium masses that the corresponding manure applications should contain (kg).		0	
	coverage_fractions	List of fractions of how much of the field is covered by the corresponding manure application.		0.01	1
	application_depths	List of depths at which the manure is injected into the soil (kg).		0	
	surface_remainder_-fractions	List of fractions of manure which remain on the soil surface when applied.		0	1

Continued on next page

Table continued from previous page

Section	Variable	Definition	Default	Min	Max
	manure_types	The type of manure which will be requested for the application.		• liquid • solid	
	supplement_manure_-nutrient_deficiencies	Determines if nutrient deficient manure applications are supplemented with chemical fertilizer.	none	• none • manure • synthetic fertilizer • synthetic fertilizer and manure	
	pattern_repeat	Number of times that this manure application schedule should be repeated.	0	0	
	pattern_skip	Number of years to be skipped between schedule repetitions.	0	0	

21.b Methodology

The first step to initializing an instance of a field is to initialize an instance of the FieldData class for each field which is called by the setup_field_data() function in FieldManager. Each instance of the FieldData class contains key information about the field from the user-provided configuration data that remains fixed throughout the simulation as well as other field attributes that are updated dynamically throughout the simulation such as the amount of crop residue that is on the surface of the field (current_residue), the amount of water being applied to a field in an irrigation interval (watering_amount_in_mm) and the number of days since the start of the current watering interval (days_into_watering_interval).

Once the FieldData is initialized, the instances of the SoilData and SoilLayerData classes for that field are initialized. Following initialization of the soils for each field, the management schedules are generated. The user has some flexibility in how they provide information about the schedule for management operations so that they can either provide a list that provides information and dates for each operation individually, or if the same operation is repeated at an annual timestep or greater, they can provide information about the pattern with which that operation is repeated via the "pattern_repeat" and "pattern_skip" inputs. For example, if a user wanted to simulate an operation such as planting corn or fertilizer application that happened every year for 5 years on May 4th starting in 2021, they could specify this schedule with either of the two following combinations of inputs in a management schedule input file:
Option 1:

- "years": [2021, 2022, 2023, 2024, 2025],
- "days": [124, 124, 124, 124, 124],
- "pattern_repeat": 0,
- "pattern_skip": 0

Option 2:

- "years": [2021],
- "days": [124],
- "pattern_repeat": 5,
- "pattern_skip": 0

Similarly, if a user wanted to simulate something like cover crop planting or a manure application that happened every 4 years on September 5th starting in 1985, they could provide either of the following two inputs for that operation:

Option 1:

- "years": [1985, 1989, 1993, 1997, 2001, 2005],
- "days": [218]],
- "pattern_repeat": 0,
- "pattern_skip": 0

Option 2:

- "years": [1985],
- "days": [218],
- "pattern_repeat": 6,
- "pattern_skip": 4

To translate the user-provided information about field schedules into standard data formats, RuFaS generates management schedules via operation specific schedule classes. These schedule classes (CropSchedule, FertilizerSchedule, ManureSchedule, and TillageSchedule) are built as child classes from a generic parent class (Schedule) that contains functions to expand the user-provided information about the timing of management operations into a list with a Julian day and year for each management operation alongside any operation specific configuration data.

Crop specific management data includes specification of both the planting and harvesting schedule, and whether the harvest operation is a harvest only operation (meaning that the crop continues to grow after harvest) or if it is an event that kills the plant, with or without harvesting the crop.

Tillage specific management information includes the depth of the tillage operation, the amount of surface soil that is mixed into lower parts of the soil profile (incorporation_fractions), and the amount of mixing that occurs between soil layers (mixing_fractions).

Fertilizer application specific management information includes the proportions of N, P, and K in the fertilizer mixes being used, the proportion of N that is in the form of ammonium (ammonium_fraction), the total masses of N, P, and K applied at each application, the application depth, and the fraction of manure that remains on the surface (surface_remainder_fraction).

Manure application specific management information includes the total masses of N, P, and K applied at each manure application, the application depth, and the fraction of fertilizer that remains on the surface (surface_remainder_fraction), and an indication of how the model should handle cases where there is not enough manure in storage to meet the manure application request (supplement_manure_-nutrient_deficiencies).

After the user information about the management schedules are translated into complete lists where each list entry is the schedule information for each operation for the entire simulation, these multi-year schedules are then translated into lists where each entry is an individual event for each operation so that the field manager can iterate over this list. The lists of events are then combined with the field and soil data to initialize an instance of a field!

Daily Field Management Each day the main simulation engine calls on the FieldManager routine to run its daily_update_routine() function which returns a list of harvested crops that can be passed to the Feed Storage Module. The FieldManager daily_update_routine() function collects the current weather conditions and manure for application to the field where appropriate and calls the Field class manage_-field() routine which is the primary method in the Field class that steps through all the functions that execute the field management operations. The order of operations called is:

- Fertilizer Application
- Manure Application
- Tillage
- Soil Nutrient Cycling + Crop Growth
- Crop Management
  - Transition crops to dormancy
  - Plant crops
  - Harvest crops
  - Update crop and crop residue coverage

Once these daily processes have been executed for each field, the FieldManager daily_update_routine() function sends all the outputs to the OutputManager via the output_gatherer.send_daily_variables() function.

Fertilizer Application The first step in execution of a fertilizer application is to calculate the total mass of fertilizer required to meet the minimum N, P, and K application masses requested for each fertilizer given the specified nutrient composition of the fertilizer mix using the following two methods:

$$\begin{aligned} fert\_mass_{nitrogen} &= \frac{mass\_requested\_nitrogen}{fert\_nitrogen\_fracfert\_mass} \\ fert\_mass_{phosphorus} &= \frac{mass\_requested\_phosphorus}{fert\_phosphorus\_frac} \\ fert\_mass_{potassium} &= \frac{mass\_requested\_potassium}{fert\_potassium\_frac} \end{aligned}$$

[SC.FLD.1]

$$applied\_mass = max[fert\_mass_{nitrogen},\ fert\_mass_{phosphorus}\ fert\_mass_{potassium}]$$

[SC.FLD.2]

Where:

- applied_mass (kg) is the mass of the given fertilizer required to satisfy all nutrient requests.
- fert_mass$_{\text{nitrogen}}$ (kg) is the mass nitrogen requested.
- fert_mass$_{\text{phosphorus}}$ (kg) is the mass of phosphorus requested.
- fert_mass$_{\text{potassium}}$ (kg) is the mass of potassium requested.
- fert_nitrogen_frac (proportion) is the fraction of nitrogen in the fertilizer mix.
- fert_phosphorus_frac (proportion) is the fraction of phosphorus in the fertilizer mix.
- fert_potassium_frac (proportion) is the fraction of potassium in the fertilizer mix.

The mass of each nutrient applied is then calculated as:

$$\begin{aligned} applied\_nitrogen &= applied\_mass * fert\_nitrogen\_frac \\ applied\_phosphorus &= applied\_mass * fert\_phosphorus\_frac \\ applied\_potassium &= applied\_mass * fert\_potassium\_frac \end{aligned}$$

[SC.FLD.3]

Where:

- applied_nitrogen (kg) is the mass of nitrogen applied.
- applied_phosphorus (kg) is the mass of phosphorus applied.
- applied_potassium (kg) is the mass of potassium applied.

Once the total fertilizer mass and nutrients associated with a fertilizer application are determined, the fertilizer nitrogen and phosphorus masses are added to the appropriate soil nutrient pools via the apply_-fertilizer() function in the FertilizerApplication class in the fertilizer_application.py file.

Note that RuFaS does not currently track soil potassium cycling so the application of potassium with fertilizer is recorded solely as a means of tracking resource use. In the future, we hope to add a soil potassium cycling routine at which point potassium will be added to soil nutrient pools via the apply_-fertilizer() method in a similar manner as nitrogen and phosphorus.

Nitrogen and phosphorus are first added to the surface soil layer followed by distribution to the sub-surface layers according to the depth of application. Because soil nutrient pools are tracked in units of kg/ha, the total mass of each nutrient applied to the field is first divided by the field size.

The amount of each nutrient applied to the surface is determined as:

$$surface\_applied\_nutrient = \frac{(applied\_nutrient * surface\_remainder\_fraction)}{field\_size}$$

[SC.FLD.4]

Where:

- surface_applied_nutrient (kg/ha) is the mass per ha of each nutrient applied to the surface soil layer.
- surface_remainder_fraction (proportion) is the user-provided input defining the fraction of the fertilizer that remains on the soil surface.
- field_size (ha) is the size of the field.

The proportion of the subsurface fertilizer that is applied to each layer is calculated from the relationship between the bottom depth of each soil and the application depth. So, for each layer a "depth factor" is calculated as:

if layer_depth < application_depth:

$$depth\_factor = \frac{layer\_depth}{application\_depth}$$

[SC.FLD.5]

else

$$depth\_factor = 1.0 - depth\_factor\_sum$$

[SC.FLD.6]

Where:

- depth_factor (proportion) is the fraction of subsurface fertilizer added to each soil layer
- layer_depth (mm) is the bottom depth of each sub-surface soil layer
- depth_factor_sum (proportion) is the sum of the depth factors for all layers above the deepest layer receiving fertilizer

Thus, the amount of fertilizer added to each nutrient pool in a sub-surface soil layer is:

$$sub\_surface\_applied\_nutrient = \frac{(applied\_nutrient*(1-surface\_remainder\_fraction))}{field_size}$$

$$sub\_surface\_applied\_nutrient_{layer\_i} = sub\_surfaapplied\_nutrient * depth\_factor$$

The soil nutrient pools that receive nitrogen are the soil nitrate and ammonium pools. The proportion of fertilizer nitrogen distributed to each of these pools is determined by the user provided ammonium_-fraction.

Fertilizer phosphorus is intended to be added to the labile inorganic phosphorus pools to be available for plant uptake. However, phosphorus applied to the surface is not immediately available in this pool and until the first precipitation or irrigation event the phosphorus is gradually sorbed by the soil. Thus, at the time of application, 75% of surface applied fertilizer P is added to the available phosphorus pool for gradual sorption and 25% of the surface applied P is added to the recalcitrant pool which is solubilized during rainfall and either added to the labile phosphorus pool or lost through runoff and leaching.

$$available\_phorphorus\_pool = surface\_applied\_phosphorus * 0.75$$
$$recalcitrant\_phorphorus\_pool = surface\_applied\_phosphorus * 0.25$$

[SC.FLD.7]

Manure Application When the results of the manure nutrient request are passed into the FieldManager and then to the Field class daily update routine, the manage_field() function can then execute the manure application with the manure nutrients and amount provided by the ManureModule via the SimulationEngine. If there are any unmet nutrients from the initial, intended application, if the user indicated that these unmet nutrients can be met with synthetic fertilizers, these are supplied by an additional fertilizer application that is initiated by the handle_unmet_nutrients() function.

Once the amount of manure to be applied is known, the apply_and_record_manure_application() function first adds the water associated with the manure application.The amount of water in the manure application is determined by the total mass of manure applied and the dry matter fraction of the manure:

$$manure\_applied\_water = \frac{manure\_mass\_applied}{manure\_dry\_matter\_fraction}$$

[SC.FLD.8]

Where:

- manure_applied_water (L) is the amount of water applied to the filed with a manure application

The amount of water in liters is then converted to the amount in mm by a helper function convert_-liters_to_millimeters() .

After the water is applied, the phosphorus and nitrogen from manure are applied via the apply_-machine_manure() function in the ManureApplication class which is implemented in manure_application.py. The manure application process is broken out into the surface application which follows methods described by the SurPhos model (Vadas et al., 2007) and the subsurface application.

Surface Manure Application During manure application, the amount of manure applied to the surface is assumed to stay on the surface unless it is a liquid manure with a dry matter content less than 15% in which case infiltration is assumed and the manure nutrients are assumed to be added to the sub-surface layer. Following the methods from the SurPhos model (Vadas et al., 2007), first the manure solids remaining on the soil surface are calculated:

$$surface\_dry\_matter\_mass = manure\_dry\_matter\_mass * surface\_remainder\_fraction$$

[SC.FLD.9]

Where:

- surface_dry_matter_mass (kg) is the amount of manure solids or dry matter that is applied to the soil surface at that application.
- manure_dry_matter_mass (kg) is the amount of manure solids or dry matter that is applied at that application.
- surface_remainder_fraction (kg) is the user-provided value for the amount of the manure applied that remains on the surface.

Surface Phosphorus Application Phosphorus is then distributed to 4 pools (water extractable inorganic phosphorus, water extractable organic phosphorus, stable inorganic phosphorus, stable organic phosphorus) according to the following methods:

First the fractions of each type of phosphorus are calculated assuming that 55% of manure phosphorus is water extractable and 75% of the non-water extractable or stable phosphorus is inorganic:

$$\begin{gathered} water\_extractable\_inorganic\_phosphorus\_fraction = 0.50 \\ water\_extractable\_organic\_phosphorus\_fraction = 0.50 \\ stable\_phosphorus\_fraction = 1 - (water\_extractable\_inorganic\_phosphorus\_fraction + \\ water\_extractable\_organic\_phosphorus\_fraction) \\ stable\_inorganic\_phosphorus\_fraction = 0.75 stable\_phosphorus\_fraction \\ stable\_organic\_phosphorus\_fraction = 0.25 stable\_phosphorus\_fraction \end{gathered}$$

[SC.FLD.10]

Then these fractions are multiplied by the total phosphorus applied as well as the product of the infiltration indicator and the surface remainder fraction to obtain the mass of each phosphorus type applied to add the surface or subsurface soil phosphorus layers:

$$mass\_to\_add\_to\_labile\_phos = total\_phosphorus\_mass(water\_extractable\_inorganic\_phosphorus\_fraction + 0.95 * water\_extractable\_organic\_phosphorus\_fraction + 0.95 * stable\_organic\_phosphorus\_fraction) * soil\_infiltration * surface\_remainder\_fraction$$

[SC.FLD.11]

$$mass\_to\_add\_to\_active\_phos = total\_phosphorus\_mass * stable\_organic\_phosphorus\_fraction * soil\_infiltration * surface\_remainder\_fraction$$

[SC.FLD.12]

Where:

- mass_to_add_to_labile_phos (kg) is the phosphorus that will be added to the surface labile phosphorus pool during that application.
- mass_to_add_to_active_phos (kg) is the phosphorus that will be added to the surface active phosphorus pool during that application.
- total_phosphorus_mass (kg) is the total phosphorus applied during that application.
- soil_infiltration is equal to 0.4 if the manure applied is less than 15% dry matter, and 0 otherwise.

The SurPhos manure pool attributes for the manure dry matter mass, moisture factor and field coverage are then updated according to the new field coverage and surface dry matter mass for that application (Vadas et al., 2007).

Surface Nitrogen Application The composition of manure nitrogen is provided by the manure module as the fractions that are in inorganic nitrogen, ammonium nitrogen, and organic nitrogen. These fractions are then distributed to the soil nitrate, ammonium, fresh organic nitrogen, and stable organic nitrogen pools according to the following:

$$mass\_nitrates\_added = manure\_dry\_matter\_mass * inorganic\_nitrogen\_fraction * \frac{(1-ammonium\_fraction)}{field\_size}$$
$$mass\_ammonium\_added = manure\_dry\_matter\_mass * inorganic\_nitrogen\_fraction * \frac{(ammonium\_fraction)}{field\_size}$$
$$mass\_fresh\_organic\_nitrogen\_added = manure\_dry\_matter\_mass * organic\_nitrogen\_fraction * \frac{fresh\_fraction\_of\_organic\_nitrogen}{field\_size}$$
$$mass\_fresh\_organic\_nitrogen\_added = manure\_dry\_matter\_mass * organic\_nitrogen\_fraction * \frac{(1-fresh\_fraction\_of\_organic\_nitrogen)}{field\_size}$$

[SC.FLD.13]

Where:

- mass_nitrates_added (kg N/ha) is the mass of nitrate nitrogen applied to relevant soil layer
- mass_ammonium_added (kg N/ha) is the mass of ammonium nitrogen applied to relevant soil layer
- mass_fresh_organic_nitrogten_added (kg N/ha) is the mass of fresh organic nitrogen applied to relevant soil layer
- mass_stable_organic_nitrogen_added (kg N/ha) is the mass of stable organic nitrogen applied to relevant soil layer
- manure_dry_matter_mass (kg) is the mass of manure dry matter added to the soil layer
- ammonium_fraction (proportion) is the fraction of inorganic nitrogen that is in the form of ammoniacal nitrogen and is provided by the Manure Module
- fresh_fraction_of_organic_nitrogen (proportion) is the fraction of organic nitrogen that is more quickly mineralized and is set to a constant of 0.9286

The manure mass that is applied to the surface layer is set to the 100% of the surface_dry_matter_-mass calculated in **[SC.FLD.9]** if the manure dry matter content is greater than 15%. For liquid manure when the dry matter content is less than 15%, infiltration is assumed and the surface application is set to 60% of the surface_dry_matter_mass from **[SC.FLD.9]** and 40% of the surface_dry_matter_mass is assigned to be added to the first sub-surface layer.

Subsurface Manure Application For manure applications where a fraction or all of the manure is applied below the soil surface (i.e. injection) the sub-surface manure application method is followed for the proportion of the application applied below the soil surface.
The sub-surface fraction of manure application is simply calculated as:

$$subsurface\_fraction = 1 - surface\_remainder\_fraction$$

[SC.FLD.14]

to capture all the manure applied that is not accounted for in the surface application methods.This value, along with the manure nitrogen and phosphorus compositions and the total masses of manure applied in each application is passed into the apply_subsurface_manure() function.
Calculation of the manure phosphorus masses to be added to different soil phosphorus pools occurs according to the equations described in **[SC.FLD.9]**, **[SC.FLD.10]**, and **[SC.FLD.11]**.
Similarly, calculation of the manure N mass to be added to the subsurface layers follows the methods described in **[SC.FLD.13]**.
The method for distributing the total_phosphorus_mass and manure_mass_applied to the subsurface layers multiplies the manure application masses by the subsurface_fraction calculated in **[SC.FLD.9]** and depth_factors calculated using the method described in **[SC.FLD.5]** and **[SC.FLD.6]** as in the generic equation below:

$$nutrient\_mass\_added\_to\_layer = \\ mass\_applied * nutrient\_fractionsubsurface\_fraction * depth\_factor$$

[SC.FLD.15]

Tillage Application The tillage method mixes nutrient masses from the soil layers to the layers below them and ensures that the nutrients do not get mixed below the bottom of the soil profile. The till_soil() function aggregates all the soil nutrient pools needed to be mixed through tillage including:

- "labile_inorganic_phosphorus_content",
- "active_inorganic_phosphorus_content",
- "stable_inorganic_phosphorus_content",
- "nitrate_content",
- "ammonium_content",
- "active_organic_nitrogen_content",
- "stable_organic_nitrogen_content",
- "fresh_organic_nitrogen_content",
- "metabolic_litter_amount",
- "structural_litter_amount",
- "active_carbon_amount",
- "slow_carbon_amount",
- "passive_carbon_amount"

It then iterates through these soil nutrient pools and soil layers to redistribute nutrients between the soil pools according to the user provided tillage depth and mixing fraction.

Initiate Daily Biophysical Processes An important step in the Field classes manage_field() routine is to initiate the updates to the biophysical processes that are managed by the Soil and Crop classes. This is achieved by the execute_daily_processes() method and initiates the following processes:

- Snow accumulation and melting
- Crop residue and canopy cover update
- Soil temperature update
- Soil water and nutrient cycling updates
- Crop growth updates

Crop Management In managing the crop cycle, the daily updates called by the Field class first check if the crop is meant to go into a dormant stage.
After initiating any dormancy where appropriate, the Field class then checks to see if any crops are scheduled to be planted that day. If so, it calls on the Crop class to initialize Crop and ensures that the max_root_depth attribute of that crop does not allow its roots to go below the bottom of the soil profile.
The last management practice the Field class checks is the harvesting event schedule. If a harvest event is scheduled for that day, it initiates the harvest method in the Crop class and updates its list of crops to remove the harvested and killed crops from its list of crops.

22 Soil Management

22.a Introduction

The soil submodule spatially and dynamically models agronomic and biogeochemical processes taking place on the surface of and within the soil profile. It tracks plant roots/residue, hydrological movement, and dissolved/undissolved carbon, nitrogen, and phosphorus cycling. Carbon and nitrogen nutrient cycling and hydrology are based on the SWAT+ model and phosphorus nutrient cycling on the SurPhos model (Vadas et al., 2007).

Soil Initialization RuFaS can simulate soils of any combination of textures in as many layers delineated by depth as the user wishes to specify. A single soil type and profile can represent all fields in a simulation or can be specific to individual fields. Each specified soil is stored by RuFaS as a SoilData object consisting of a list of LayerData objects that in turn represent a set of variables specific to each layer of soil. At the start of the simulation the user's soil specifications generate the soil layers for each field and pools of water, carbon, nitrogen, and phosphorus are initialized in each layer.

Required Inputs

Table 91: Required inputs for the Soil section.

Variable	Definition	Default	Min	Max
second_moisture _condition_parameter	Curve number parameter for average moisture condition equation (unitless)	85	20	99
average_subbasin_slope	Slope of the field, measured as rise over run (unitless)	0.05	0	
slope_length	Slope of the field, measured as rise over run (m)	50	0	
manning_roughness_coefficient	Roughness coefficient for overland flow (unitless)	0.4	0.001	
support_practice_factor	Ratio of soil loss with a specific support practice based on slope characteristics. This value adjusts for terracing, contouring, and stripcropping, and these features are not present in V1, so this value IS NOT used (unitless)	0.08		
albedo	The ratio of reflected to total incident short-wave solar radiation. Related primarily to soil color and cover (unitless)	0.16	0	1
residue_fresh_organic _mineralization_rate	Fraction of residue that will decompose in a day given optimal conditions (unitless)	0.05	0	
soil_evaporation _compensation_coefficient	Modifies depth distribution used to meet soil evaporative demand (unitless)	0.95	0.01	1
initial_residue	Residual plant biomass remaining after a crop is harvested (kg h^{-1})	0	0	
soil_layers	The soil layers that this profile contains. Each element of the array should be a 'soil_layer' object		1	
bottom_depth	The bottom depth of each soil layer object within the soil_layers array (mm)		20	500
wilting_point_water_-concentration	Condition of soil moisture at which soil water becomes unavailable and water held in the soil at a tension of 1.5 Mpa (mm H_2O/mm soil)	0.2	0	0.95

Continued on next page

Table continued from previous page

Variable	Definition	Default	Min	Max
field_capacity_water_-concentration	Soil moisture condition at which the soil matric potential is zero or water held by the soil at 0.033 MPa (mm H_2O/mm soil)	0.3	0	0.95
saturation_point_water _concentration	Concentration of water in a layer when it has become saturated (mm H_2O/mm soil)	0.5	0	0.95
saturated_hydraulic_-conductivity	Measure of ease of water movement through the soil (mm hr-1)	9.5	0.01	15
bulk_density	Ratio of mass of solid particles to total volume of soil (g cm^{-3})	1.4	1.1	1.9
organic_carbon_fraction	Fraction of soil weight that is organic carbon at beginning of the simulation.	0.012	0	1
clay_fraction	Fraction of soil mass that is clay (unitless)	0.225	0	1
silt_fraction	Fraction of soil mass that is silt (unitless)	0.625	0	1
sand_fraction	Fraction of soil mass that is sand (unitless)	0.125	0	1
rock_fraction	Fraction of soil mass that is rock (unitless)	0.013	0	1
soil_water_concentration	Concentration of water in a soil layer at the beginning of the simulation (mm H_2O/mm soil)	0.25	0	0.95
pH	pH of the soil layer (pH)	7	0	14
initial_temperature	Temperature of the layer at the beginning of the simulation ($°C$)	15.05		
initial_labile_inorganic _phosphorus_concentration	Concentration of labile inorganic phosphorus in the layer at the beginning of the simulation (mg kg^{-1} soil)	null	0	
initial_fresh_organic _phosphorus_concentration	Concentration of initial fresh organic phosphorus in the layer at the beginning of the simulation (mg kg^{-1} soil)	0	0	
initial_soil_nitrate_-concentration	Concentration of nitrate in the layer at the beginning of the simulation (mg kg^{-1} soil)	null	0	
initial_soil_ammonium_-concentration	Concentration of ammonium in the layer at the beginning of the simulation (mg kg^{-1} soil)	null	0	
humus_mineralization_rate_-factor	Rate factor for humus mineralization of active organic nutrients (N and P) (unitless)	0.003	0	0.003

Continued on next page

Table continued from previous page

Variable	Definition	Default	Min	Max
ammonium_volatilization _cation_exchange_factor	Controls the rate of ammonia volatilization (unitless)	0.45	0.01	
denitrification_rate_coefficient	Controls the rate of denitrification (unitless)	1.4	0	3
denitrification _threshold_water_content	Fraction of field capacity water content above which denitrification takes place. (unitless)	1	0	1
residue_fresh _organic_mineralization_rate	The fraction of residue that will decompose in a day given optimal conditions (unitless)	0.05	0	

22.b Methodology

Soil temperature and water content/flow are the main drivers of soil biogeochemical cycling.

Soil Temperature Soil temperature fluctuates diurnally and seasonally and is estimated daily as a function of the previous day's soil temperature, average annual air temperature, the current day's soil surface temperature and depth in the soil profile for each soil layer by using the following equations from SWAT 2009 documentation.

$$T_{soil}\ (z,\ d_n) = (L * T_{soil,(z,dn-1)}) + (1 - L) * [df * (T_{aair} - T_{surf}) + T_{surf}]$$

[SC.TMP.1]

Where:

- $T_{soil\ (z,d_n)}$ = soil temperature ($°C$) at depth z (mm) and day of the air $\mathrm{d_n}$
- L = lag coefficient(ranging from 0.0 to 1.0, default: 0.8) that controls the influence of the previous day's temperature on the current day's temperature
- $T_{soil(z,d_n-1)}$ =soil temperature ($°C$) at depth z (mm) on previous day
- df = depth factor that quantifies the influence of depth below surface on soil temperature
- T_{aair} = Average annual air temperature ($°C$)
- T_{surf} = Daily soil surface temperature ($°C$) on the day

$$df = \frac{zd}{zd + exp(-0.867 - 2.078 * zd})$$

[SC.TMP.2]

- zd = ratio of depth at the center of soil layer to damping depth

$$zd = \frac{z}{dd}$$

[SC.TMP.3]

Where:

- dd = damping depth (mm)
- z = depth at the center of the soil layer (mm)

Damping depth is a function of soil water and max damping depth. dd_{max} is the maximum damping depth (mm) to which the temperature wave penetrates and the amplitude decreases to 1/e or 0.37 of its initial value in the soil layers. This is largely dependent on soil thermal conductivity and thermal diffusivity, and is calculated using following equations

$$dd_{max} = 1000 + \frac{2500BD}{BD + 686\exp(-5.63BD)}$$

[SC.TMP.4]

$$dd = dd_{max} * exp\{ln(\frac{500}{dd_{max}}) * [\frac{(1-scale)}{(1+scale)}]^2\}$$

[SC.TMP.5]

- dd_{max} = maximum damping depth (mm)
- scale = scaling factor for soil water

$$scale = \frac{SW}{[(0.356-0.144*BD)*Ztot]}$$

[SC.TMP.6]

Where:

- BD = soil bulk density (g/cm^3)
- SW = total soil water in the profile (mm)
- Ztot = total soil profile depth (mm)

Soil surface temperature is a function of the previous day's temperature, the amount of ground cover, and the temperature of a bare soil surface. The temperature of a bare soil is calculated as:

$$T_{bare} = T_{av} + E_{sr}^{\frac{(Tmax-Tmin)}{2}}$$

[SC.TMP.7]

Where:

- T_{bare} = Temperature of a bare soil surface ($°C$) with no cover
- T_{av} = Average daily temperature ($°C$) on the day
- E_{sr} = radiation term

$$E_{sr} = \frac{[Hday*(1-albedo)-14]}{20}$$

[SC.TMP.8]

Where:

- H_{day} = daily solar radiation (user input, MJ/m^2/d)
- albedo = daily fraction of incoming sunlight that is reflected by a surface

Albedo should be determined by soil type and is set by user input.
The impact of plant and snow cover is quantified as a weighting factor, bcv, given by the maximum of the following two equations:

$$bcv = \frac{plant\_cover}{plant\_cover + exp(7.563 - 0.0001297) * plant\_cover}$$

[SC.TMP.9]

$$bcv = \frac{SNOW}{SNOW + exp(6.055 - 0.3002) * SNOW}$$

[SC.TMP.10]

Where:

- SNOW = snow water content on the current day (mm)
- plant_cover = Total aboveground plant biomass and residue on the current day (kg per hectare)

$$Tsurf = (bcv * T_{soil,d-1}) + [(1 - bcv) * T_{bare}]$$

[SC.TMP.11]

- Tsurf = surface temperature ($°C$)

Surface temperature is dependent on the previous day's soil temperature and is calculated before soil temperature on any given day. On the first day of a simulation the default soil temperature is used for $T_{soil,d-1}$.

Soil Hydrology

The soil hydrological processes calculated in RuFaS are limited to infiltration, the downward movement of water through the soil profile, and evapotranspiration, the movement of water from the soil profile into the atmosphere.

Infiltration (implemented in infiltration.py) The Infiltration class is implemented in infiltration.py and calculates both infiltration and runoff. Infiltration is the movement of soil water into and through the soil profile and surface runoff results when application of water to the ground surface exceeds the infiltration rate. Infiltration is calculated as the difference between the amount of water applied to the surface and the amount of surface water run off. Surface water is thus calculated first and then used to calculate soil water infiltration. Soil texture (percentage of sand, silt, and clay), bulk density, presence of pores, pore diameter, and continuity affect the soil infiltration rate. Slow soil infiltration rate results in ponding in the level areas, surface runoff, and erosion in sloping areas. High infiltration rates can lead to nutrient leaching. Surface runoff is estimated using the NRCS curve number method.
Runoff has a minimum of 0 and is calculated as:

$$ifR > 0.2S,\ runoff = R - 0.2S * 2R + 0.8S$$
$$if\ not,\ runoff = 0$$

[SC.INF.1]

Where:

- R = daily rainfall (mm H_2O)
- S = retention parameter (mm H_2O)

S is determined using an empirical value CN (curve number, unitless, user-defined). Three values of CN are used to determine S.

$$CN_1 = CN_2 - \frac{20*100-CN_2}{100-CN_2+exp[2.533-0.0636*(100-CN_2)]}$$

[SC.INF.2]

$$CN_3 = CN_2 * exp[0.00673 * (100 - CN_2)]$$

[SC.INF.3]

Where:

- CN_1, the first moisture condition, is the wilting point.
- CN_2, the second moisture condition, curve number is user-defined.
- CN_3, the third moisture condition, is field capacity.

S is calculated as

$$S = S_{max} * \{1 - \frac{SW_{available}}{SW_{available}+exp[w_1-(w_2*SW_{available})]}\}$$

[SC.INF.4]

- S_{max} = maximum value for S on any given day (mm H_2O)
- $SW_{available}$ = available soil profile water content (SW - WP) (mm H_2O)
- $w_1,\ w_2$ = shape coefficients

Smax is calculated as:

$$S_{max} = 25.4 * [\frac{1000}{CN_1} - 10]$$

[SC.INF.5]

w1 and w2 are calculated as:

$$w_1 = ln\{\frac{FC}{1-(S_3*S_{max}^{-1})} - FC\} + w_2 * FC$$

[SC.INF.6]

$$w_2 = \frac{ln\{\frac{FC}{1-(S_3*S_{max}^{-1})}-FC\}-ln\{\frac{SAT}{1-(2.54*S_{max}-1)}-SAT\}}{SAT-FC}$$

[SC.INF.7]

Where:

- FC = amount of water in soil profile at field capacity (mm H_2O)
- SAT = amount of water in soil profile at saturation (mm H_2O)

S3 is calculated as:

$$S_3 = 25.4 * [\frac{1000}{CN_3} - 10]$$

[SC.INF.8]

When the top layer of the soil is frozen (layerTemp $\leq 2°C$), S is modified using the following equation:

$$S_{frz} = S_{max} * [1 - exp(-0.000862 * S)]$$

[SC.INF.9]

Infiltration into the soil is daily rainfall less daily runoff.

$$Infiltration = R - runoff$$

[SC.INF.10]

Evapotranspiration Evapotranspiration (ET) is one of the most important hydrological processes of soil water balance in the soil and crop module. It is the sum of all processes by which water moves from the land surface to the atmosphere via soil evaporation, plant canopy transpiration, and sublimation. RuFaS adapts the Hargreaves method (**hargreaves1985**) to estimate ET which only requires temperature as an input.

Potential evapotranspiration (PET) is the maximum amount of water that could be removed due to surface evaporation and plant canopy transpiration with no limitation on water availability, while evapotranspiration (ET) is the actual amount of water lost due to evaporation and transpiration.

Potential Evapotranspiration (Implemented in field.py) is the larger of 0.001 or function of the radiation, daily temperature range, and heat of vaporization

$$ET_{max} = \max\{\frac{0.0023 * H_0 * (T_{max} - T_{min})^{0.5} * (T_{avg} + 17.8))}{LHV}, 0.001\}$$

[SC.EVP.1]

Where:

- LHV = latent heat of vaporization (MJ kg^{-1})
- ET$_{max}$ = potential evapotranspiration (otherwise known as PET) (mm d^{-1})
- H$_0$ = extraterrestrial radiation (MJ m^{-2} d^{-1}), input
- T$_{max}$ = maximum air temperature for a given day ($°C$)
- T$_{min}$ = minimum air temperature for a given day ($°C$)
- T$_{avg}$ = mean air temperature for a given day ($°C$)

LHV is calculated as:

$$LHV = 2.501 - 2.361 * 10^{-3} * T_{\text{avg}}$$

[SC.EVP.2]

Maximum crop transpiration (implemented in water_dynamics.py) is the proportion of potential evapotranspiration that is transpired through the plant canopy. LAI lower than 3.0 and available water in the rooting zone limit transpiration.

$$\begin{aligned} trans_{max} &= \frac{ET_{max} * LAI_{act}}{3.0} && if 0 \leq LAI_{act} \leq 3.0 \\ trans_{max} &= ET_{max} && if LAI_{act} > 3.0 \end{aligned}$$

[SC.EVP.3]

Where:

- Trans$_{max}$ = maximum transpiration on a given day (mm H_2O)
- LAI$_{act}$ = LAI (Leaf Area Index) actual (calculated in Crop Growth)

Sublimation and soil evaporation (implemented in field.py) refer to the phase changes of solid and liquid water to gas which transfer water from land to atmosphere.

$$evap_{max} = ET_{max} * SoilCov$$

[SC.EVP.4]

Where:

- evap$_{max}$= maximum soil evaporation/sublimation on a given day (mm H_2O)
- SoilCov = soil cover index

$$SoilCov = exp(-5.0 * 10^{-5} * BioMass)$$

[SC.EVP.5]

Where:

- BioMass = aboveground biomass and residue (kg ha^{-1})

Potential evaporation is adjusted based on trans$_{\max}$ as:

$$evap_{\max} = min \left\{ \frac{evap_{\max} \cdot ET_{\max}}{evap_{\max} + trans_{\max}},\ evap_{\max} \right\}$$

[SC.EVP.6]

Partition evaporation within soil profile (implemented in evaporation.py) vertically subsets the maximum evaporation to calculate evaporation demand within each layer of the soil profile

$$evap_z = evap_{max} * [\tfrac{z}{z+exp(2.374-0.00713*z)}]$$

[SC.EVP.7]

Where:

- evap$_z$ = evaporation demand at depth z (mm H_2O)
- z = depth below soil surface (mm)

The evaporation demand for a given soil layer is the difference between evaporation demands at the top and bottom of the layer.

$$evap_{ly} = evap_{bottom} - evap_{top}$$

[SC.EVP.8]

When the water content of a soil layer is below field capacity, the evaporative demand for the layer is reduced according to the following equation

$$evap_{ly}\_reduced = evap_{ly} * exp[2.5 * \tfrac{SW-FC}{FC-WP}] \qquad if\ SW < FC$$

[SC.EVP.9]

Where:

- evap$_{ly}$ = evaporation demand of the layer

- SW = soil water content for the layer (mm H_2O)
- WP = soil water content held at wilting point (mm H_2O)
- FC = field capacity for soil water (mm H_2O)

In addition, the daily amount of water removed by evaporation is limited to 80% of the plant available water (SW-WP): evaporation.py

$$evap_{ly} = min\left\{0.8 * SW - WP,\ evap_{ly}\right\}$$

[SC.EVP.10]

The total actual amount of water lost in evapotranspiration (ETact) is equal to the sum of evaporation from each layer

Percolation (implemented in percolation.py) Percolation is the process of water movement downward through the soil profile. When the soil water content in a layer is less than or equal to field capacity no percolation occurs. Percolation is not calculated for frozen layers.

$$perc = SW_{perc} * [1 - exp(\frac{-t}{TT})]$$

[SC.PER.1]

Where:

- SWperc = water available for percolation (mm H_2O)
- perc = amount of water that percolates to the underlying soil layer (mm H_2O)
- t = time step (24 h)
- TT = travel time for percolation (h)

$$SW_{perc} = SW - FC \qquad\qquad if SW > FC$$

[SC.PER.2]

TT for each soil layer is calculated as:

$$TT = \frac{SAT-FC}{Ksat}$$

[SC.PER.3]

Where:

- Ksat = saturated hydraulic conductivity (mm/h)
- SAT = Amount of water in the soil layer when completely saturated (mm H_2O)

22.c Soil Erosion (Implemented as soil_erosion.py)

Soil erosion is a geological process whereby surface soil is removed by water/wind and anthropogenically. In RuFaS, erosion is modeled from individual crop fields using the Modified Universal Soil Loss Equations (MUSLE, SWAT 2009 documentation). The MUSLE input variables include rainfall (rainfall amount and intensity), soil erodibility factor, soil cover/management, best management practices, and the topographic factor.

Base Equation, Rainfall Intensity, and Peak Runoff

$$sed = 11.8 * (Qsurf * qpeak * field\_size)^{0.56} * K * C * P * LS * CFRG$$

[SC.ERO.1]

Where:

- sed = sediment yield on a given day (metric tons)
- Qsurf = surface runoff volume (m^3)
- qpeak = peak runoff rate (m^3/sec)
- field_size = Area of the cropping field (ha)
- K = USLE soil erodibility factor (Mg MJm^{-1}mmm^{-1}))
- C = USLE cover and management factor
- P = USLE support practice factor
- LS = USLE topographic factor
- CFRG = the coarse fragment factor

q_{peak} is calculated as:

$$qpeak = \frac{RC * I * Area}{3.6}$$

[SC.ERO.2]

Where:

- RC = runoff coefficient
- I = rainfall intensity (mm/hr)
- Area = field size (ha)

RC is the fraction of daily runoff to daily precipitation

$$RC = \frac{runoff}{R}$$

[SC.ERO.3]

Where:

- runoff = runoff (mm H_2O)
- R = rainfall (mm H_2O)

Rainfall intensity is rainfall over time for the time of concentration

$$I = \frac{Rtc}{Tconc}$$

[SC.ERO.4]

Where:

- Rtc = rain amount during time of concentration (mm H_2O)
- Tconc = time of concentration (h)

$$Tconc = \frac{Length^{0.6} * n^{0.6}}{18 * slope^{0.3}}$$

[SC.ERO.5]

Where:

- Length = slope length of field (m)
- n = manning's roughness coefficient (Refer Table 2)
- slope = field slope (m/m)

$$Rtc = alpha * R$$

[SC.ERO.6]

Where:

- alpha = fraction of daily rain during time of concentration

$$alpha = 1 - exp[2 * Tconc * ln(1 - alpha_{0.5})]$$

[SC.ERO.7]

Where:

- $\text{alpha}_{0.5}$ = fraction of daily rain in in $\frac{1}{2}$ hour of highest intensity

$$alpha_{0.5} = \frac{\{0.02083+[1-exp(\frac{-125}{R+5})]\}}{2}$$

[SC.ERO.8]

Soil erodibility factor K is calculated as:

$$K = f_{csand} * f_{cl-si} * f_{orgc} * f_{sand}$$

[SC.ERO.9]

Where:

- F_{csand} = gives low factors for soils with high sand contents and high values for soils with little sand
- $\text{F}_{\text{cl-si}}$ = gives low factors for soils with high clay to silt ratios
- F_{orgc} = reduces soil erodibility for soils with high organic carbon content
- F_{sand} = reduces soil erodibility for soils with high sand contents

Factors are calculated as:

$$f_{csand} = 0.2 + 0.3 * exp[-0.256 * sand * (1 - \frac{silt}{100})]$$

[SC.ERO.10]

$$f_{cl-si} = [\frac{silt}{clay+silt}]^{0.3}$$

[SC.ERO.11]

$$f_{orgc} = 1 - \frac{0.25orgC}{orgC+exp(3.72-2.95*orgC)}$$

[SC.ERO.12]

Where:

- orgC = organic Carbon percentage for the first soil layer.

$$f_{sand} = 1 - \frac{0.7*(1-\frac{sand}{100}}{(1-\frac{sand}{100})+exp(-5.51+22.9*(\frac{sand}{100})^{-1})}$$

[SC.ERO.13]

The cover and management factor, C, is the ratio of soil loss from land cropped under specified conditions to loss from clean-tilled, continuous fallow. The equation below essentially allows C to vary between 0.8 and 0.05 as:

$$C = exp\{[ln(0.8) - ln(0.05)] * exp(-0.00115 * surface\ residue) + ln(0.05)\}$$

[SC.ERO.14]

Where:

- Cover = amount of residue and growing biomass covering the soil surface (kg/ha)
- 0.05 = the estimated minimum value for C

LS is a topographic factor and is the expected ratio of soil loss per unit area from a field slope to that from a 22.1-m length of uniform 9 percent slope under identical conditions. LS is estimated as:

$$LS = \frac{Lhill}{22.1}^{m} * [65.41 * \sin^2(alphahill) + 4.56 * \sin(alphahill) + 0.065]$$

[SC.ERO.15]

Where:

- Lhill = hill slope length (user input, m)
- Alphahill = angle of the hill slope, defined as tan^{-1}(average subasin slope) (user input, m/m)
- average subbasin slope = average slope for the hydrological subbasin (user input, m/m)

The exponent m is calculated as

$$m = 0.6 * [1 - exp(-35.835 * average\ subbasin\ slope)]$$

[SC.ERO.16]

The P factor is a user defined support practice factor that reduces erosion.

If there is snow on the ground, this will heavily affect sediment yield. sed is adjusted for snowfall on the range day > 350, or day < 60 as:

$$sed_{snow} = \frac{sed}{exp(\frac{3*snow\ water\ content}{25.4})}$$

[SC.ERO.17]

Where:

- sedsnow = snow adjusted sediment yield.

Coarse Fragment factor (CFRG) is calculated as :

$$CFRG = exp(-0.053 * rock)$$

[SC.ERO.18]

Where:

- rock is the percent of rock in the first soil layer

22.d Soil Nitrogen Cycling

RuFaS tracks soil N as five pools based on nitrogen form and lability. Three pools are organic N; fresh, active, and stable. Two are inorganic N; NH_4^+ and NO_3^-. Nitrogen enters the cycle as synthetic fertilizer, manure, and crop residue. Synthetic fertilizers are partitioned into nitrate and ammonium. Manure organic N is partitioned into active and stable organic N pools by 92.86% and 7.14% respectively. Active and stable organic N can interconvert. N in the active pool can also get percolated to the second layer. Furthermore, manure dry matter mass goes into ammonium and nitrate pools using certain equations. Residue biomass also gets converted into fresh and active organic N pools via mineralization. If residue is in top 2 cm it goes into the fresh organic N pool while if residue is below 2 cm it goes into the active organic N pool. 80% of fresh organic N is converted into nitrate and 20% into active pool via decomposition. Active N is converted into NH_4^+ after mineralization. NH_4^+ can be converted into NO_3^- via nitrification, taken up by plants, volatilized, percolated, or lost via runoff. NO_3^- can be taken up by plants, percolate, runoff, or denitrify to NO, N_2O, and N_2 gas.

Initialize soil N Pools (Implemented in layer_data.py)

$$initial\_soil\_nitrate\_concentration = 7 * e^{\frac{-depth\_of\_layer\_center}{1000}}$$

[SC.SON.1]

Where:

- initial_soil_nitrate_concentration is the initial concentration of nitrate in the soil (mg kg^{-1} or ppm) at depth.
- depth_of_layer_center is the depth of the middle of the soil layer from the soil surface (mm).

The ammonium pool for soil N is initially set by the user in the soil input file.

Organic N values in the soil profile are set by the following equation:

$$humic\_organic\_nitrogen\_concentration = 10^4 * \frac{organic\_carbon\_fraction * 100}{14}$$

[SC.SON.2]

Where:

- humic_organic_nitrogen_concentration is the concentration of humic organic N in the layer (mg kg^{-1} or ppm).
- organic_carbon_fraction is the percent organic C in the layer (%, user input).

Humic organic N is partitioned between the active and stable pools for each layer using the following equations:

$$activeN = humic\_organic\_nitrogen\_concentration * FractN$$
$$stableN = humic\_organic\_nitrogen\_concentration * (1 - FractN)$$

[SC.SON.3]

Where:

- activeN, initial_active_organic_nitrogen_concentration, is the concentration of N in the active organic pool (mg kg^{-1}).
- humic_organic_nitrogen_concentration is the concentration of humic organic N in the layer (mg kg^{-1}).
- FractN, fraction_of_humic_nitrogen_in_active_pool, is the fraction of humic N in the active pool; General Constant - 0.02.
- stableN, initial_active_organic_nitrogen_concentration, is the concentration of N in the stable organic pool (mg kg^{-1}).

Fresh N is initialized in the top soil layer only (user defined, usually 0-20 mm), and is 0.15% of the initial amount of residue on the soil surface.

$$fresh\_organic\_nitrogen\_content = 0.0015 * residue$$

[SC.SON.4]

Where:

- fresh_organic_nitrogen_content is the fresh organic pool in the top soil layer (kg N ha^{-1}).
- residue is material in the residue pool for the top soil layer (kg ha^{-1}).

While pool sizes are initialized as concentration (mg kg^{-1}), all calculations are done in units of mass (kg ha^{-1}). The conversion is:

$$nitrogen\_mass\_kg\_ha = \frac{(nitrogen\_mass\_mg\_kg * BD * depth * fieldsize)}{fieldsize}$$

[SC.SON.5]

$$nitrogen\_mass\_kg\_ha = nitrogen\_mass\_mg\_kg * BD * depth * 10$$

[SC.SON.6]

Where:

- BD is the bulk density for the given layer (g cm^{-3} or mg m^{-3}).
- depth is the thickness of a given layer (mm).
- field size is the field area (ha).

All pools have a lower limit of 0. If a transfer of N from one pool to another is greater than the N in the pool of origin, the destination pool receives only the amount that depletes the pool of origin. The rest of the N cycle is in the nitrogen_cycling folder and run by nitrogen_cycling.py, which is called from the soil routine.

Nitrification and Volatilization (Implemented in nitirification_volatilization.py) Nitrification is the conversion of NH_4^+ to NO_3^- via microbial oxidation of NH_4^+ under suitable soil temperature and moisture conditions. Nitrification occurs only at soil temperatures greater than 5°C and is modeled as a function of soil temperature and water factors, temp_factor and water_factor.

$$temp\_factor = min(1.0,\ 0.41\ X^{\frac{temperature-5}{10}})\ if\ temperature > 5$$

[SC.NNV.1]

Where:

- temperature is the temperature in the soil layer (°C).

If:

$$water\_content \geq 0.25 * field\_capacity - 0.75 * wilting\_point\ water\_factor = 1$$

Else:

$$water\_factor = \frac{water\_content - wilting\_point}{0.25 * (field\_capacity - wilting\_point)}$$

Where:

- water_content is the water content of the soil layer on a given day (mm).
- field_capacity is the field capacity water content (mm).
- wilting_point is the wilting point water content (mm).

The volatilization of aqueous soil ammonium to ammonia gas (NH_3) occurs even when nitrification does not. NH_3 volatilization is influenced by temperature, moisture, fertilizer application, pH, and depth (**mu-extsoil**).

The volatilization depth factor (DepthFac) is unitless and calculated as:

$$depth\_factor = 1 - \{\frac{depth\_of\_layer\_center}{depth\_of\_layer\_center + exp(4.706 - 0.0305 * depth\_of\_layer\_center)}\}$$

[SC.NNV.2]

Where:

- depth_of_layer_center is the depth from the surface of the soil to the middle of the layer (mm).

The impact of environmental factors on nitrification and NH_3 volatilization are estimated by nitrification and volatilization regulator equations. Nitrification and volatilization only occur if the soil temperature is greater than $5°C$.

A nitrification regulator (nitrification_regulator) and a volatilization regulator (volatilization_regulator) are calculated as:

$$nitrification\_regulator = temp\_factor * water\_factor$$
$$volatilization\_regulator = temp\_factor * depth\_factor * cation\_exchange\_factor$$

[SC.NNV.3]

Where:

- cation_exchange_factor is provided by the soil input file..

The total mass of NH_4 (kg N ha^{-1}) converted through nitrification and volatilization is given by:

$$total\_ammonium\_lost = NH_4 * [1 - exp(-1 * nitrification\_regulator - volatilization\_regulator)]$$

[SC.NNV.4]

The estimated fraction of total_ammonium_lost that is converted to NO_3 by nitrification is:

$$nitrification\_loss\_fraction = 1 - exp(-1 * nitrification\_regulator)$$

[SC.NNV.5]

The estimated fraction of total_ammonium_lost that is lost by volatilization (FracVolatil) is:

$$volitilization\_loss\_fraction = 1 - exp(-1 * volitilization\_regulator)$$

[SC.NNV.6]

The amount of N removed from the NH_4 pool via nitrification and converted to NO_3 is (kg N ha^{-1}) calculated as:

$$nitrification\_loss = [\frac{nitrification\_loss\_fraction}{nitrification\_loss\_fraction + volitilization\_loss\_fraction}] * total\_ammonium\_lost$$

[SC.NNV.7]

If:

$$nitrification\_loss\_fraction + volitilization\_loss\_fraction \leq 0$$
$$nitrification\_loss\ is\ 0$$

N loss in leaching and runoff (Implemented in leaching_runoff_erosion.py) N can be transported with the movement of water through and out of the soil profile through leaching, runoff, and erosion. Leaching is the process by which N ions move down the soil profile with percolating soil water (Wang and Li, 2019). Leaching affects inorganic and active organic pools. If percolation is 0, then percolated N is also 0. NO_3, NH_4 and active N lost to percolation in each layer is removed from the layer of origin and added to the layer below. The water and N percolated from the bottom layer is lost from the soil profile to the vadose layer. Runoff and erosion N is removed exclusively from the surface soil layer. Lateral flow is not taken into account.

The concentration of NO_3 and NH_4 (kg N mm^{-1} H_20) in the mobile water fraction for a given layer ($NConc_1$) is:

$$mobile\_water\_nitrogen\_concentration = \frac{NO_3\ or\ NH_4 * [1 - exp(\frac{-total\_mobile\_water}{(1-\theta) * SAT})]}{total\_mobile\_water}$$

[SC.NNV.8]

Where:

- NO_3 or NH_4 is the amount of nitrate or ammonium in the layer (kg N ha^{-1})
- total_mobile_water is the amount of mobile water in the layer (mm H_2O)
- θ is the fraction of porosity from which anions are excluded
- SAT is the saturated water content of the soil layer (mm H_2O).

The amount of mobile water in the layer (mm H_2O) is the amount of water lost by surface runoff and percolation given as:

$$\text{for top layer: } total\_mobile\_water = runoff\_water\_amount + percolated\_water\_amount$$

[SC.NNV.9]

$$\text{for lower soil layers: } total\_mobile\_water = percolated\_water\_amount$$

[SC.NNV.10]

Where:

- runoff_water_amount is the surface runoff generated on a given day (mm H_2O)
- percolated_water_amount is the amount of water percolating to the underlying soil layer on a given day (mm H_2O).

The amount/mass (kg N ha^{-1}) of NO_3 and NH_4 lost in surface runoff is:

$$nitrates\_lost\_to\_runoff = \\ nitrate\_conc\_in\_mobile\_h20 * nitrate\_runoff\_coefficient * runoff\_water\_amount \\ ammonium\_lost\_to\_runoff = \\ ammonium\_conc\_in\_mobile\_h20 * ammonium\_runoff\_coefficient * runoff\_water\_amount$$

[SC.NNV.11]

Where:

- xxx_conc_in_mobile_h20 is the concentration of nitrate or ammonium in the surface layer mobile water (kg N mm^{-1} H_2O)
- xxx_runoff_coefficient is the nitrate or ammonium percolation coefficient (GeneralConstants; ammonium = 0.2, nitrate = 0.2)
- runoff_water_amount is the surface runoff generated on a given day (mm H_2O)

N erosion (Implemented in leaching_runoff_erosion.py) Soil N is eroded from the organic N pools: fresh, active, and stable.

$$nitrogen\_erosion\_concentration = \frac{100 * N(kgN/ha)}{bulk\_density * depth}$$

[SC.LCH.1]

Where:

- bulk_density is the soil layer bulk density (Mg m^{-3})
- depth is the thickness of the soil layer (mm).

Organic N mass loss in erosion (kg N ha^{-1}) is calculated as:

$$erosion\_nitrogen\_loss = \\ 0.001 * nitrogen\_erosion\_concentration * daily\_soil\_lost * enrichment\_ratio$$

[SC.LCH.3]

Where:

- daily_soil_lost is the daily soil loss per hectare (metric tons soil ha^{-1})
- nitrogen_erosion_concentration is the concentration of organic N in the top soil layer (g N metric ton^{-1} soil)
- enrichment_ratio is the ratio of concentration of organic N transported with the sediment to that found in the soil surface layer.

$$enrichment\_ratio = exp[1.21 - 0.16 * log(daily\_soil\_lost * 1000)]$$

[SC.LCH.4]

$$Percolated\_ammonium = ammonium\_conc\_in\_mobile\_h20 * percolated\_water$$
$$Percolated\_nitrate = nitrate\_conc\_in\_mobile\_h20 * percolated\_water$$

[SC.LCH.5]

Where:

- xxx_conc_in_mobile_h20 is the concentration of nitrate or ammonium in the surface layer mobile water (kg N mm^{-1} H_2O) as determined in equation **[SC.NNV.8]**
- Percolated_water is the water percolated from the layer (mm)

Denitrification (Implemented in denitrification.py) Denitrification is the bacterial conversion of NO_3 to denitrified nitrates (N_2, NO_2, NO, N_2O) under anaerobic conditions. The denitrification submodule calculates total denitrified nitrates and partitions these emissions further into N_2O and N_2. Denitrification is a function of soil water content, temperature, and induced by the presence of NO_3 and carbon (C).

The amount of nitrate that is lost to denitrification (DenitrN) (kg N ha^{-1}) is calculated for each soil layer by the function calculate_denitrification_amount according to the following equation:

If nutrient_cycling_soil_water_factor$_{\text{layer[i]}}$ > soil_water_threshold:

$$denitrified\_nitrates_{layer[i]} = nitrate\_content_{layer[i]} * max(min(1.0, 1 - exp(-denitrification\_rate\_coefficient * temp\_factor_{layer[i]} * organic\_carbon\_percent_{layer[i]})),$$

[SC.NDN.1]

Else:

- denitrified_nitrates$_{\text{layer[i]}} = 0$

Where:

- nitrate_content$_{\text{layer[i]}}$ is the amount of nitrate in layer i (kg N h^{-1})

- denitrification_rate_coefficienct is the user defined denitrification rate coefficient which can have values between 0-3 with a default value of 1.4
- temp_factor$_{\text{layer[i]}}$ is the nutrient cycling temperature factor for each layer,
- organic_carbon_percent$_{\text{layer[i]}}$ is the amount of organic carbon in the layer (%) calculated as organic_carbon_fraction$_{\text{layer[i]}}$ * 100
- nutrient_cycling_soil_water_factor$_{\text{layer[i]}}$ is the nutrient cycling water factor for the layer i
- soil_water_threshold is the nutrient_cycling_water_factor threshold value for the soil profile and is a user input with a default of 1.0. According to Wen et al. (2024) the value for the soil_water_-threshold is typically set to 1.0 so denitrification is allowed to occur when water content is above field capacity.

The denitrified_nitrates$_{\text{layer[i]}}$ is removed from the layer's NO_3 pool and partitioned into N_2O and N_2 which are then lost to the atmosphere.

Partitioning factor is used to partition denitrified nitrates into N_2O and N_2

$$Partitioning\_factor = max((min\ (NO3\_effect, C\_effect) * moisture\_effect * pH\_effect), 0.0)$$

[SC.NDN.2]

$$NO3\_effect = (1.0 - [0.5 + \arctan(\pi * 0.01 * \frac{(NO3\_content - 190)}{\pi})\}] * 25)$$

[SC.NDN.3]

$$C\_effect = 13 + \{\frac{[30.78 * \arctan[\pi * 0.07 * (C_{resp} * 1000 - 13)]}{\pi}\}$$

[SC.NDN.4]

$$moisture\_effect = \frac{1.4}{(13^{17/(13^{(2.2*WFPS)})})}\ if\ WFPS = 0,\ moisture\ effect = 0$$

[SC.NDN.5]

$$pH\_effect = \frac{1}{(1470 * e^{(-1.1*pH)})}$$

[SC.NDN.6]

Where:

- NO3_effect is the effect of nitrate on partitioning (unitless)
- C_effect is the effect of carbon on partitioning (unitless)
- arctan= the arc tangents measured in radians of x (values are between -pi/2 and pi/2)

- NO3_content is the amount of nitrate in the layer (ug N/g soil)
- C_{resp} is carbon respiration from the soil layer (kg ha^{-1})
- WFPS is the water filled pore space in the soil layer (unitless).

The amount of N_2O-N emissions (kg N ha^{-1}) are calculated as follows:

$$nitrous\_oxide\_nitrogen = (\frac{denitrified\_nitrates}{(1+partitioning_factor)}) * 0.001$$

[SC.NDN.7]

$$N_2(kg) = (\frac{denitrified\_nitrates}{-1(partitioning\_factor)}) * 0.001$$

[SC.NDN.8]

Where:

- denitrified_nitrates is the amount of nitrates that have been denitrified (kg ha^{-1})

Mineralization and Decomposition/Immobilization (Implemented in mineralization_ decomp.py)

Mineralization is the microbial conversion of organic N to simple, soluble forms of organic N (including amino acids) and ultimately to the inorganic N that can be taken up by a plant. Decomposition is the breakdown of fresh organic residue into simpler organic components. Immobilization is the uptake or assimilation of N by microbes making them unavailable (temporarily) to plants. Soil stoichiometry (C:N) determines whether organic N will be mineralized (resulting in more plant available N) or inorganic N will be immobilized (resulting in less plant available N).

The N Mineralization equations represent net mineralization which incorporate immobilization into the equations. Fresh and active N pools are subject to decomposition and mineralization respectively. Mineralization and decomposition are allowed to occur only if soil temperature is greater than $0°C$. Soil temperature and water are the two factors used in the mineralization and decomposition equations to account for the impacts of temperature and moisture on these processes.

N mineralized from the active N pool (Nminact) (kg N ha^{-1}) is calculated as:

$$N\_min\_active = Min\_rate * (temp\_factor * water\_factor)^{0.5} * activeN$$

[SC.NMN.1]

Where:

- humus_min_rate is the rate coefficient for mineralization of the humus active organic nitrogen
- temp_factor is the nutrient cycling temperature factor for the layer
- water_factor is the nutrient cycling water factor for the layer
- activeN is the amount of N in the active organic pool (kg N ha^{-1}).

N mineralized from the active pool is transferred from the active pool to the NH_4 pool. Decomposition and mineralization are a function of a daily decay rate constant that is calculated with the C:N and C:P ratios of the residue, and temperature and soil water factors. The C:N and C:P ratios of the residues in the soil layer are currently fixed at:

$$C : N = 25 \; C : P = 200$$

[SC.NMN.2]

The decay rate constant, the same as that used to calculate residue in crop defines the fraction of residue that is decomposed as:

$$decay\_rate\_constant = min\_coeff * residue\_comp\_factor * (temp\_factor * water\_factor)^{0.5}$$
$$ResComp = 1.0$$

[SC.NMN.3]

Where:

- residue_comp_factor is the nutrient cycling residue composition factor
- min_coeff is the rate coefficient for mineralization of the residue fresh organic nutrients (user-defined, default - 0.05)
- temp_factor is temperature's effect on nutrient cycling for the layer
- water_factor is water's effect on nutrient cycling for the layer

Mineralization of the residue fresh organic N (kg N ha^{-1}) is calculated as:

$$residue\_freshN\_mineralization = decay\_rate\_constant * freshN$$
$$ResComp = 1.0$$

[SC.NMN.4]

Where:

- decay_rate_constant is the residue decay rate constant
- freshN is the N in the fresh organic pool in the layer (kg N ha^{-1}).

20% of the N decomposed from the residue_freshN_mineralization is then transferred to the active organic N pool, and 80% is transferred to the NO_3 pool. In the top layer, 0.15% of residue is decomposed into fresh organic N pools.
Humus Mineralization (Implemented in humus_mineralization.py)

In general, the term humus describes the transformed products formed from a wide variety of organic substrates without any intent to assign chemical composition or structure (**hayes2020vindication**).

Humus mineralization allows N to move between the active and stable organic pools. N mineralized from the humus active organic pool is added to the NO_3 pool in the layer.

The amount of N transferred between the active and stable organic pools (Ntrans) (kg N h^{-1}) maintains an equilibrium, and is calculated as

$$amount\_transferred = humus\_mineralization\_rate * active\_organic\_nitrogen * [\frac{1}{fracN} - 1] - stable\_organic\_nitrogen$$

[SC.NMN.5]

Where:

- humus_mineralization_rate is the humus mineralization rate constant 10^{-5}
- active_organic_nitrogen is the amount of N in the active organic pool (kg N ha^{-1})
- fracN is the fraction_of_humic_nitrogen_in_the_active_ pool (0.02)
- stable_organic_nitrogen is the amount of N in the stable organic pool (kg N ha^{-1})

When Ntrans is positive, N moves from the active to stable organic pool. When Ntrans is negative, N moves from stable to active organic pool.

22.e Soil Phosphorus Cycling

Soil phosphorus cycling is based on the SurPhos model theoretical documentation (Vadas et al., 2007), which identifies eight soil phosphorus pools; two organic and six inorganic. Organic pools are divided into stable and water extractable and inorganic pools are divided into stable, water extractable, active, labile, recalcitrant, and available. RuFaS handles phosphorus cycling via the PhosphorusCycling class, which contains and manages the simulated aspects of phosphorus in a soil profile. These aspects are divided into fertilizer (implemented in fertilizer.py), manure (implemented in manure.py), mineralization (implemented in phosphorus_mineralization.py), and soluble phosphorus (implemented in soluble_-phosphorus.py). The cycle_phosphorus method calls each of the daily routines that manage surface and subsurface phosphorus.

Phosphorus Initialization (Implemented in layer_data.py) The P sorption parameter (PSP) is initialized before the simulation begins. The PSP represents how much of any inorganic P added to soil remains labile P upon reaching relative equilibrium. A PSP of 0.5 means 50% of added P remains labile P and 50% becomes active P.

PSP is initialized as:

$$PSP = max(0.05,\ min(0.7,\ -0.045 * log(clay) + 0.001 * labileP - 0.035 * orgC + 0.43)$$

[SC.PHO.1]

$$adjusted\_clay\_content = max(10^{-8},\ clay\ fraction * 100)$$

[SC.PHO.2]

Where:

- PSP is the P sorption parameter (unitless)
- clay is the adjusted_clay_content
- clay fraction is the fraction of the layer composed of clay
- orgC is the percentage of the layer composed of organic carbon, assumed to be 58% of user-defined organic matter content (%)
- labileP (labile_inorganic_phosphorus) is the concentration of labile inorganic phosphorus (mg kg^{-1})

$$initial\ active\ inorganic\ Pconc = \frac{labile\ inorganic\ P*(1.0-PSP)}{PSP}$$
$$initial\ stable\ inorganic\ Pconc = 4.0*activeP$$

[SC.PHO.3]

After initialization soil P pools are converted from mg phosphorus kg^{-1} soil to a kg ha^{-1} basis.

Soluble Phosphorus (Implemented in soluble_phosphorus.py) The SolublePhosphorus class calculates the P transferred via hydrological processes in the soil based on equations from the APLE and SurPhos models (Vadas et al., 2007). The daily_update_routine alters phosphorus levels in the SoilData object by tracking phosphorus in runoff (if any) from the first soil layer and phosphorus percolated downward through the soil profile.

Soluble Phosphorus Runoff

$$adjusted\ DRP_{runoff} = min(labileP,\ top\_soil\_layerP*extraction\ coefficient*runoff*10^{-6})$$

[SC.PSL.1]

Where:

- adjusted DRPrunoff is the dissolved reactive phosphorus in runoff from the soil (kg ha^{-1}) surface expressed as the lesser of labileP or calculated dissolved P
- extraction coefficient is 0.005
- Runoff is the amount of water lost from the surface layer (L ha^{-1})
- top_soil_layerP is:

$$top\_soil\_layerP = \frac{labileP*field\_size*kg-to-mg}{bulk\_density*Mg-to-kg*soil\_volume}$$

[SC.PSL.2]

Where:

- labile_P is the labile phosphorous content of the soil (mg/kg)
- field_size is the field area (hectares)
- kg-to-mg is the constant 10^6
- Mg-to-kg is the constant 10^3
- soil_volume is the volume of soil (m^3)

Soluble Phosphorus Leaching Phosphorus leaching is predicted by the clay isotherm function in the APLE documentation (Vadas et al., 2007).

$$isotherm\ slope = 173.51 * clay\_fraction + 8.48$$

[SC.PSL.3]

Where:

- soilP is the concentration of P in the soil layer (mg P kg^{-1} soil)
- isotherm_slope is the slope of the isotherm curve (unitless)
- clay_fraction is the fractional clay content of a soil layer (unitless)

$$isotherm\_inter = 4.726 * isotherm\_slope - 8.97$$

[SC.PSL.4]

Where:

- isotherm_inter is the intercept of P sorption isotherm (unitless)
- isotherm_slope is the slope of the P sorption isotherm (unitless)

$$DRP_{leachate} = min(20.0,\ exp(\frac{soilP*1.5-isotherm\_inter}{isotherm\_slope}))$$

[SC.PSL.5]

Where:

- $DRP_{leachate}$ is the dissolved reactive phosphorus leached from the soil layer (mg L^{-1})

The maximum leachate concentration specified in APLE is 20 (mg L^{-1}), and is applied here as the min term. Leached P is added to the soil layer immediately below the layer for which it is calculated.

The predicted value in (mg L^{-1}) is then converted to kg h^{-1} and the actual dissolved reactive phosphorus is expressed as the minimum of the layer labile P or the calculated DRP$_{\text{leachate}}$ to prevent negative values.

$$DRP_{leachate},\ actual = min(labileP,\ DRP_{leachate})$$

[SC.PSL.6]

Where:

- DRP$_{\text{leachate}}$ is the dissolved reactive leachate P (kg ha^{-1})
- DRP$_{\text{leachate}}$,actual is the actual dissolved reactive P leached (kg ha^{-1})

Summarized leachate values are the sum of values that leave the bottom layer of the soil profile.

Fertilizer Leaching and Runoff (Implemented in fertilizer.py) Simulates the leaching and runoff of phosphorus from fertilizer applied to the soil surface.

Available Fertilizer P pool During rainfall events, the fraction of fertilizer P lost is greatest during the first rain event and reduced for each consecutive rainfall event. When synthetic fertilizer P is applied, 75% of it goes into an available pool, and 25% goes into a recalcitrant pool. The equation below determines the fraction of P remaining in the available P pool (unitless).

$$fertPavail = max\ (0,\ -0.16 * log(fert_{CNT}) + cov)$$

[SC.PSL.7]

Where:

- fertP$_{\text{avail}}$ is the fraction of P remaining in the available P pool
- fertCNT is the number of days since the fertilizer was applied
- cov is the cover factor, adjusted based on cover type:
  - "BARE": 0.5333
  - "RESIDUE COVER": 0.6667
  - "GRASSED": 0.8

Note: the multiplicative and additive constants differ between the SurPhos documentation and here, the ones listed here should be preferred (Vadas et al., 2007).

fert$_{\text{CNT}}$ is then iterated to account for the logarithmic curve of fertilizer sorption without rain. The availability of P for runoff or leaching decreases as it sits on the soil surface.

Runoff and Leaching

At the first rainfall event, available P is split between runoff and leachate. Subsequent rain events target the recalcitrant pool, with 40% of released P leached, and all subsequent events leaching 7.5% of available P. The partitioning of P between runoff and leachate is a function of the amount of rainfall.

$$sorp\ \% = rain\_day\_factor$$

[SC.PSL.8]

Where:

- rain_day_factor is the rain day factor calibrated as described above

Sorbed fertilizer P is then computed and cannot exceed that in the available P pool:

$$fert_{sorp} = min(fertP_{avail},\ max(0.0,\ fertP_{avail} * sorp\ \%))$$

[SC.PSL.9]

$$fertP_{avail} = fertP_{avail} - fert_{sorp}$$

[SC.PSL.10]

$$sorp\% = max\ (0,\ -0.16 * log(fert_{CNT}) + cov)$$

[SC.PSL.11]

Where:

- Fert$_{\text{sorp}}$ is the fertilizer sorption

For the first layer, labileP is converted to kg to add adsorbed fertilizer P. labileP is then converted back to kg ha^{-1}. The concentration of fertilizer dissolved P in runoff (mg L^{-1}) (fertP$_{\text{dissolved in runoff}}$) is computed:

$$fertP_{dissolved\ in\ runoff} = \frac{(fertP * fraction\_P\_released * PD_{fact})}{total\ rainfall}$$

[SC.PSL.12]

$$PDfact = 0.034 * exp(3.4 * (\frac{runoff}{rainfall}))$$

[SC.PSL.13]

Where:

- fertP is the amount of fertilizer P in the pool that is going to be leached from (mg)
- fraction of P released is the fraction of P solubilized during the current rain event. 1 for first, 0.4 for second, and 0.075 (unitless).
- PD_{fact} is the P distribution factor is the value that determines P distribution between runoff and infiltration (unitless)
- total rainfall is rainfall on the day (L)

(Vadas et al., 2007) Leached fertilizer P is added to soil labileP for each layer to yield the updated labileP as a function of depth:

$$labileP\_updated = labileP + (fert_{leach} - fert_{runoff}) * DF$$

[SC.PSL.14]

Where:

- fert_leach is amount of fert P to be leached from one layer to the next
- fert_runoff is the amount of fert P lost to runoff

Fertilizer P lost to runoff and leaching are added to their cumulative sums

Manure Phosphorus

Manure is an important source of bioavailable P for crop uptake, runoff, and leaching losses. Organic and inorganic P contributions from manure applications are tracked and modeled, including decomposition of manure solids, labile P release and P sorption. Largely changes to manure P are by three main processes: leaching, decomposition, and assimilation.
Manure Phosphorus Leaching (implemented in manure_pool.py)

$$MIP_{leach} = max(0.0, manure_{extr} * WIP)$$
$$MOP_{leach} = max(0.0, \frac{manure_{extr} * WOP}{0.6})$$

[SC.PSL.31]

Where:

- MIP_{leach} is the manure inorganic P leached (kg ha^{-1}).
- manure_extr manure available for extraction
- MOP_{leach} is the manure organic P leached (kg ha^{-1})
- WIP is the water extractable inorganic P
- WOP is the water extractable organic P

Leached P is then removed from surface pools and added to respective annual sums.

The concentration of dissolved P in runoff (mg/L):

$$PDfact = (\frac{runoff}{rainfall})^{0.225}$$

[SC.PSL.32]

Where:

- PD_{fact} is the P distribution factor
- runoff is the amount of runoff from rainfall on the current day (mm)
- rainfall is the amount of rainfall on the current day (mm)

The concentration of water extractable phosphorus in runoff on the current day is calculated as

$$MIP_{runoff} = MIP_{leach}\ in\ mg * (\frac{1}{rainfall}) * (\frac{1}{field\ size\ in\ mm^2}) * (\frac{1}{mm^3\ to\ L}) * PD_{fact}$$

[SC.PSL.33]

$$MIP_{leach\ in\ mg} = MIP_{leach} * 1000000$$

[SC.PSL.34]

$$field\_size_{mmsq} = field\_size\ * 10000000000$$

[SC.PSL.35]

Where:

- MIP_{runoff} is the the concentration of water extractable P in runoff on the current day (mg L^{-1})
- MIP_{leach} in mg is the manure leached in mg
- PD_{fact} is the Factor accounting for runoff to rainfall ratio on the current day (unitless)

$$MOP_{runoff} = MOP_{leach\ in\ mg} * (\frac{1}{rainfall}) * (\frac{1}{field\ size\ in\ mm^2}) * (\frac{1}{mm^3\ to\ L}) * PDfact$$

[SC.PSL.36]

Manure runoff P is then converted to kg and the total manure leached P can be calculated with these terms as:

$$M_{leach} = MIP_{leach} - MIP_{runoff} + MOP_{leach} - MOP_{runoff}$$

[SC.PSL.37]

Leached manure P is then added to labileP for each layer as a function of depth (implemented in manure_application.py):

$$\begin{gathered}
labileP\_updated = labileP + M_{leach} * DF \\
labileP\_updated = labileP * DF \ (used\ when\ manure\ is\ applied\ in\ the\ field) \\
DF = \frac{ratio}{ratio + exp(-0.867 - (2.078 * ratio))} \\
ratio = \frac{center\ depth}{damping\ depth}
\end{gathered}$$

[SC.PSL.38]

Where:

- DF is the Depth Factor for a given layer of soil, a terminating geometric sequence modeling the decreasing fraction of manure P to be leached from one layer to the next
- center depth is the depth of the center of a given soil layer (mm)
- damping depth is the damping depth of the soil factor (mm)

Manure and leachate P are then added to their respective annual sums

Manure Decomposition (implemented in manure_pool.py)

$$\begin{gathered}
wet_{rate} = -0.3 * manure\_moisture + 0.27 \\
dry_{rate} = (-0.5 * (\frac{manure_{mass}}{mass}) + 0.075) * T_{fac}
\end{gathered}$$

[SC.PSL.39]

Where:

- wet$_{\text{rate}}$ is the wet rate of manure decomposition (unitless)
- dry$_{\text{rate}}$is the dry rate of manure decomposition (unitless)
- manure$_{\text{mass}}$is the total mass of manure on the field (kg/ha)
- mass is the mass of applied manure (kg)
- manure_moisture is the manure water content (unitless)

The moisture content of the manure is then adjusted as follows: If rainfall on a given day is above 4 mm:

$$moisture = moisture + wet_{rate},\ if\ rainfall \quad > \ 4mm$$

[SC.PSL.40]

If rainfall is below 1.0 mm:

$$moisture = moisture - dry_{rate}$$

[SC.PSL.41]

$$moisture = min(0.9,\ max(0.0,\ moisture))$$

[SC.PSL.42]

$$AWDCR = 0.003 * TFA^{0.5}$$
$$ASIM = 30.0 * exp(2.5 * moisture)$$

[SC.PSL.43]

$$TFA = \frac{(2 * 32^2 * T^2 - T^4)}{32^4}$$

[SC.PSL.44]

Where:

- AWDCR is the unitless decomposition factor
- ASIM is the unitless manure assimilation factor. unitless manure factor
- TFA is the unitless temperature factor
- T is the average daily air temperature in $^{\circ}C$

The decomposition factor, dcom is calculated and then adjusted by manure$_{\text{mass}}$

$$dcom = min(max(0.0,\ manure_{mass} * AWDCR),\ manure_{mass})$$

[SC.PSL.45]

The decomposition factor is then used to calculate the cover decomposition factor:

$$cov_{dcom} = min(manure_{cov},\ max(0.0,\ \frac{dcom}{manure_{mass}}))$$

[SC.PSL.46]

The P decomposition factor for each P pool (P$_{\text{dcom}}$) is then calculated:

$$P_{dcom} = min(P,\ max(0.0,\ Ppools * 0.01 * min(TFA,\ moisture)))$$

[SC.PSL.47]

Where:

- P_{pools} represents each of the tracked P pools: stable inorganic P (SIP), stable organic P (SOP), water-extractable organic P (WOP)

Manure mass and total cover also have assimilation factors calculated as:

$$manure_{ASIM} = min(manure_{mass},\ max(0.0,\ ASIM * TFA * manure_{cov}))$$

[SC.PSL.48]

$$cov_{ASIM} = min(manure_{cov},\ max(0.0,\ \frac{manure_{ASIM}}{manure_{mass} * manure_{cov}}))$$

[SC.PSL.49]

Each pool then also has an ASIM calculated as:

$$P_{ASIM} = min(P,\ max(0.0,\ \frac{manure_{ASIM}}{manure_{mass} * P}))$$

[SC.PSL.50]

Each ASIM and decomposition factor are then removed from the corresponding P pool.

P from decomposition is accounted for in the inorganic pool:

$$WIP = WIP + WOP_{dcom} + SOP_{dcom} * 0.75 + SIP_{dcom}$$
$$WOP = WOP + SOP_{dcom} * 0.25$$

[SC.PSL.51]

The total decomposed P is then:

$$decomposedP = SIP_{ASIM} + WOP_{ASIM} + SOP_{ASIM} + WIP_{ASIM}$$

[SC.PSL.52]

Where:

- SIP is stable inorganic P

- SOP is stable organic P
- WOP is water-extractable organic P
- WIP is water-extractable inorganic P

Manure mass and cover are updated to reflect decomposition and assimilation:

$$manure_{mass} = max(0.0,\ manure_{mass} - dcom - man_{ASIM})$$
$$manure_{cov} = max(0.0,\ manure_{cov}\ - cov_{dcom} - cov_{ASIM})$$

[SC.PSL.53]

Decomposed P is then added to the labile pool for each layer as a function of depth:

$$labileP = labileP + decomposedP * depth\_factor$$

[SC.PSL.54]

Where:

- depth_factor is a terminating geometric sequence modeling the decreasing fraction of decomposed P to be leached from one layer to the next

Manure Runoff (implemented in manure_pool.py) Manure and soil P in the first layer are susceptible to runoff and the manure dissolved reactive phosphorus is 0.5% of the top layer P concentration:

$$MDRP_{runoff} = top\_layer\_manureP * 0.005$$

[SC.PSL.55]

Where:

- $\mathrm{MDRP}_{\mathrm{runoff}}$ is the manure dissolved reactive P transported in runoff (mg/L)
- manureP is the amount of manure phosphorus

The total inorganic phosphorus in runoff is calculated as the sum of manure inorganic, dissolved reactive, and fertilizer P:

$$TIP_{runoff} = MIP_{runoff} + MDRP_{runoff} + fertP_{runoff}$$

[SC.PSL.56]

Where:

- $\mathrm{TIP}_{\mathrm{runoff}}$ is the total inorganic P runoff
- $\mathrm{MIP}_{\mathrm{runoff}}$ is the manure inorganic P runoff
- $\mathrm{MDRP}_{\mathrm{runoff}}$ is the manure dissolved reactive P runoff [SC.PSL.55]
- $\mathrm{fertP}_{\mathrm{runoff}}$ is the fertilizer P runoff

Phosphorus Mineralization (implemented in phosphorus_mineralization.py)

Phosphorus mineralization is the transfer of P from organic to inorganic pools and between labile and active inorganic pools.

The amount of P that mineralizes into water-extractable inorganic P on the current day from the given pool is a function of the sorption dynamics of the type of P being mineralized, soil temperature, and soil moisture. The maximum P sorption parameter (PSP) is calculated for each layer then adjusted to be between 0.05 and 0.7 and limited based on the average value of PSP.

$$PSP_{act} = max(0.05,\ min(0.7, (\frac{(meanPSP*364)+currentPSP)}{365}))$$

[SC.PMN.1]

Where:

- $\mathrm{PSP}_{\mathrm{actual}}$ is the actual P sorption parameter (unitless)
- meanPSP is the mean sorption parameter (unitless)
- currentPSP is the current sorption parameter (unitless)

The following variables are used to define the empirical sorption and desorption curves.

The current balance of P between labile and active inorganic pools is calculated as pbal. A negative value returned indicates that there is more active phosphorus than there should be, a positive value indicates that there is more labile phosphorus than there should be, and a return value of zero indicates that the two pools are balanced

$$pbal = labileP - activeP * \frac{PSP_{act}}{(1.0-PSPact)}$$

[SC.PMN.2]

Where:

- pbal is the current balance of P between labile and active inorganic pools
- labileP is the labile inorganic P content of the soil layer (kg/ha)
- activeP is the active inorganic P content of the soil layer (kg/ha)
- $\mathrm{PSP}_{\mathrm{act}}$ is the The actual P sorption parameter of the layer adjusted from the maximum (unitless)

The P desorption factor ($\mathrm{PD}_{\mathrm{fac}}$) calculates how much P should be desorped in the given soil layer and given as:

$$PD_{fac} = base * (day_{count}^{-0.32})$$

[SC.PMN.3]

$$base = (-1.0 * PSP_{act}) + 0.8$$

[SC.PMN.4]

The amount of P that should be removed from the active inorganic P pool to put it in equilibrium with the labile inorganic P pool (kg ha^{-1}) is given as labile$_{\text{pflow}}$

$$labile_{pflow} = PD_{fac} * pbal * -1.0$$

[SC.PMN.5]

Where:

- PD$_{\text{fac}}$ is the P desorption factor
- day$_{\text{count}}$ is the number of days that the active inorganic P pool has been greater than it would be when in equilibrium with the labile inorganic P pool.
- labile$_{\text{pflow}}$ is the amount of P moved from the active to labile inorganic P pool to maintain equilibrium (kg/ha).
- base is a value used to determine how much P is transferred from the active P pool to the labile inorganic P pool (unitless).
- PSP$_{\text{act}}$ is the actual P sorption parameter of the layer adjusted from the maximum (unitless)

P sorption factor (PD$_{\text{fac}}$) calculates how much P should be sorped in the given soil layer:

$$PS_{fac} = 0.918 * exp(-4.603 * PSP_{act}) * (day_{count}^{-0.238*log(0.918*exp(-4.603*PSP_{act}))-1.126})$$

[SC.PMN.6]

Where:

- PS$_{\text{fac}}$ is the P sorption factor
- scalar is the scalar used to calculate sorption factor. The scalar is used in determining how much phosphorus is removed from the labile inorganic phosphorus pool and transferred to the active inorganic phosphorus pool (unitless).
- exponent is a value used as an exponential term when determining the phosphorus sorption rate (unitless).

The amount of phosphorus that should be removed from the labile inorganic phosphorus pool to put it in equilibrium with the active inorganic phosphorus pool (kg / ha) ($\text{active}_{\text{pflow}}$) is given as:

$$active_{pflow} = PS_{fac} * pbal$$

[SC.PMN.7]

P also transfers from stable pool to the active pool and given as $\text{stable}_{\text{pflow}}$:

$$\begin{aligned} stable_{pflow} &= 0.0006 * (stableP - (4.0 * activeP)) \\ stable_{pflow} &= min(stableP, stable_{pflow}) \qquad if\ stable_{pflow} > 0 \\ stable_{pflow} &= (-1 * stable_{pflow}) \qquad if\ stable_{pflow} > activeP \end{aligned}$$

[SC.PMN.8]

Where:

- $\text{Stable}_{\text{pflow}}$ is the P transferred from the stable to the active inorganic P pool (kg ha^{-1})
- stableP is the stable inorganic P content of the soil layer (kg ha^{-1})
- activeP is the active inorganic P content of the soil layer (kg ha^{-1})

22.f Soil Carbon Cycling

The CarbonCycling class (implemented in carbon_cycle.py) manages carbon cycling processes within the SoilData object. Carbon cycling processes include decomposition of organic matter, processing of crop residue, and partitioning of carbon between soil pools and the atmosphere and are based on the DAYCENT model. There are five carbon pools in the soil:

- Plant Structural - Slowly decomposing plant mass with higher lignin content
- Plant Metabolic - Rapidly decomposing plant mass with low lignin content
- Active - Microbially active carbon yielded by decomposition
- Slow - Soil organic matter that turns over every 20-40 years
- Passive - Stable, recalcitrant soil organic matter with a turn over time between 200 and 1500 years

Decomposition occurs within the soil C pools resulting in the transfer of C between pools. In addition, crop residues, above and below ground, decompose into the soil C pools. Each decomposition event generates some CO_2 that enters the atmosphere.

Soil Carbon Partitioning (implemented in carbon_cycle.py) The mass and proportion of C in the soil C pools are set by the following equations:

$$\begin{aligned} C_{active}\ \% &= (\tfrac{C_{active}}{soil\ mass}) \\ C_{slow}\ \% &= (\tfrac{C_{slow}}{soil\ mass}) \\ C_{passive}\ \% &= (\tfrac{C_{passive}}{soil\ mass}) \end{aligned}$$

[SC.CAR.1]

Where:

- C_{active} % = active C composition of the soil (%)
- C_{slow} % = slow C composition of the soil (%)
- C_{passive} % = passive C composition of the soil (%)
- soil mass = the mass of soil (kg ha^{-1}

$$C_{\%} = C_{active\%} + C_{slow\%} + C_{passive\%}$$

[SC.CAR.2]

Where:

- C% = carbon composition of the soil (%)

$$C = C_{active} + C_{slow} + C_{passive}$$

[SC.CAR.3]

Where:

- C = soil carbon (kg ha^{-1})

Decomposition (implemented in decomposition.py) Carbon decomposition is the process of complex carbon molecules breaking down into simpler molecules and leaving behind more recalcitrant C species, represented here as a transfer of C between C pools. Decomposition is a function of temperature and moisture. The following factors are parameterized and calculated with the assumption of coarse soil:

$$T_d = \frac{teff_2 + (\frac{teff_3}{\pi}) * tan^{-1}(\pi * teff_4 * (T_{avg} - teff_1))}{normalizer}$$

[SC.CAR.4]

Where:

- T_d = unitless temperature effect on decomposition factor (empirically determined based on average soil temperature)
- $teff_1$ = x-value at inflection point (empirical factor set at 15.4)
- $teff_2$ = y location of inflection point (before normalization) (empirical factor set to 11.75)

- teff_3 = distance from maximum point to the minimum point (empirical factor set at 29.7)
- teff_4 = slope of line at inflection point (empirical factor set at 0.031)
- normalizer = empirical factor set at 20.80546

$$M_d = (\frac{water_{fac} - b}{a - b})^{e_1} * (\frac{water_{fac} - c^{e_2}}{a - c})$$

[SC.CAR.5]

Where:

- M_d = unitless moisture effect on decomposition factor (empirically determined based on soil water levels)
- $\text{water}_{\text{fac}}$ = relative water saturation (%)
- a = dimensionless empirical factor (0.55)
- b = dimensionless empirical factor (1.7)
- c = dimensionless empirical factor (-0.007)
- e_1 = dimensionless empirical factor (6.648115)
- e_2 = dimensionless empirical factor (3.22)

Decomposition proceeds at differing rates between the active, slow, and passive pools.

$$C_{active\ decomp\ rate} = K5 * (1 - 0.75 * silt : clay_{\%})$$

[SC.CAR.6]

Where:

- $C_{\text{active decomp rate}}$ = rate at which active C is decomposed into slow C, passive C and CO_2 (%) (kg year^{-1})
- K5 = maximum rate of C decomposition (0.14, **parton1987analysis**)
- silt:clay$_{\%}$ = ratio of percent silt to clay content in the soil

$$C_{active_{decomp}} = C_{active\ decomp\ rate} * M_d * T_d * C_{active}$$

[SC.CAR.7]

Where:

- $C_{\text{active decomp}}$ = active C decomposed into slow or passive C and CO_2 (kg ha^{-1})
- C_{active} = active C stored in the soil (kg ha^{-1})

$$C_{slow\ decomp} = K6 * M_d * T_d * C_{slow}$$

[SC.CAR.8]

Where:

- $C_{\text{slow decomp}}$ = slow C decomposed into active or passive carbon and CO_2 (kg ha^{-1})
- K6 = slow C decomposition factor (set at 0.0038, **parton1987analysis**)
- C_{slow} = slow C stored in the soil (kg ha^{-1})

$$C_{passive\ decomp} = K7 * M_d * T_d * C_{passive}$$

[SC.CAR.9]

Where:

- $C_{\text{passive decomp}}$ = passive C decomposed into active or passive carbon and CO_2 (kg ha^{-1})
- K7 = passive C decomposition factor (set at 0.0038, **parton1987analysis**)
- C_{passive} = passive C stored in the soil (kg ha^{-1})

CO_2 produced during soil C pool decomposition

$$Es = 0.85 - 0.68 * silt : clay_{\%}$$

[SC.CAR.10]

Where:

- Es = adjusted factor of CO_2 loss from the decomposition of active C (**parton1987analysis**)

$$C_{active\_to\_slow} = C_{active_{decomp}} * (1 - Es - 0.004)$$
$$C_{active\_to\_CO_2} = C_{active_{decomp}} * Es$$

[SC.CAR.11]

Where:

- $C_{\text{active:slow}}$ = active C decomposed into slow C (kg ha^{-1})
- $C_{\text{active loss}}$ = active C lost as CO_2 during decomposition into slow C (kg ha^{-1})

$$C_{active\_to\_C_{passive}} = C_{active_{decomp}} * 0.004$$

[SC.CAR.12]

Where:

- C_{active}:$C_{passive}$ = active C decomposed into passive carbon (kg ha^{-1})

$$C_{slow\_to\_active} = C_{slow_{decomp}} * (1 - CO2_{slow\ loss} - C_{slow\_to\_passive})$$
$$C_{slow\_to\_CO_2} = C_{slow\ decomp} * CO2\_to\_C_{slow\ loss}$$
$$C_{slow\_to\_passive} = C_{slow\ decomp} * frac\_C_{slow\ passive}$$

[SC.CAR.13]

Where:

- $C_{slow\_to\_active}$ = slow C decomposed into active C (kg ha^{-1})
- CO_2_to_$C_{slow\ loss}$ = slow C lost as CO_2 during decomposition (fraction)
- frac_$C_{slow_to_passive}$ = slow C decomposed into passive C (fraction)
- $C_{slow\_to\_passive}$ = slow C decomposed into passive C (kg ha^{-1})
- $C_{slow\_to\_CO_2}$ = slow C lost as CO_2 during decomposition (kg ha^{-1})

$$C_{passive\_to\_CO_2} = C_{passive\ decomp} * frac\_CO2_{passive\ loss}$$

[SC.CAR.14]

Where:

- $C_{passive:active}$ = passive C decomposed into active carbon (kg ha^{-1})
- frac_$CO2_{passive\ loss}$ = passive C lost as CO_2 during decomposition (fraction)
- $C_{passive\ loss}$ =passive C lost as CO_2 during decomposition (kg ha^{-1})

$$C_{CO2\ loss\ decomp} = C_{active\ loss} + C_{slow\ loss} + C_{passive\ loss}$$

[SC.CAR.15]

Where:

- $C_{CO_2\ loss\ decomp}$ =C lost as CO_2 during decomposition (kg ha^{-1})

Residue Partitioning (implemented in residue_partitioning.py) Crop residues are partitioned into above and below ground biomass representing stalks and roots left after harvest, respectively. Additionally, some above ground residue is incorporated into the soil profile in the case of tillage.

Above Ground: Lignin, Metabolic C, Structural C Above ground residue is further partitioned into metabolic and structural components based on the lignin composition of the residue.

$$AG_{lignin\ res\ frac} = 0.12 * rainfall * 0.1$$

[SC.CAR.16]

Where:

- $AG_{\mathrm{lignin\ res\ frac}}$ = fraction of lignin composition of above ground plant residue (unitless)
- rainfall = amount of rainfall on the current day (mm)

$$AG_{L:N} = \frac{\frac{AG_{lignin\ res\ frac}}{100}}{fr_N} \qquad \text{if } \mathrm{fr_N}{>}0$$

$$\text{Otherwise:}$$

$$AG_{L:N} = 0.4$$

[SC.CAR.17]

Where:

- $AG_{\mathrm{L:N}}$ = above ground plant lignin to nitrogen (N) ratio when nitrogen in plant residue at harvest is greater than 0 (unitless)
- $\mathrm{fr_N}$ = fraction of N in plant residue at harvest (unitless)

The ratio of lignin to N in residue determines the percentage of above ground residue in the metabolic pool, with a maximum of 85%.

$$AG_{met\ frac} = 0.85 - 0.18 * AG_{L:N}$$

[SC.CAR.18]

Where:

- $AG_{\mathrm{met\ frac}}$ = above ground residue that is metabolic (unitless)

The component of residue dry matter at harvest consisting of above ground metabolic carbon (C) is added to the above ground metabolic C pool.

$$AG_{met\_updated} = AG_{met} + AG_{DM\ res} * AG_{met\ frac}$$

[SC.CAR.19]

Where:

- AG_{met} = above ground metabolic carbon (kg ha^{-1})
- $AG_{met\ frac}$ = fraction of above ground residue that is metabolic (unitless)
- $AG_{DM\ res}$ = dry matter residue at harvest (kg ha^{-1})

Above ground metabolic C follows two paths into the soil. It is either incorporated below ground pool via tillage or decomposed into the active C. The calculated values are subtracted from the above ground metabolic pool and added to the relevant pool.

$$AG_{met}\_to\_C_{active} = AG_{met_{active\ decomp}} * M_d * T_d * AG_{met}$$

[SC.CAR.20]

Where:

- $AG_{met}\_to\_C_{active}$ = above ground metabolic C decomposed to active carbon (kg ha^{-1})
- $AG_{met\ active\ decomp}$ = rate of decomposition from metabolic to active C (set at 0.28, **parton1987analysis**)

Below ground metabolic C decomposes into active C.

$$AG_{met}\_to\_BG_{met} = AG_{met} * tillage_{frac}$$

[SC.CAR.21]

Where:

- AG_{met}:BG_{met} = above ground metabolic C decomposed to below ground metabolic C (kg ha^{-1})
- tillage = fraction metabolic C incorporated into soil during tillage (unitless)

Any dry matter residue at harvest that is not added to the above ground metabolic C pool is added to the above ground structural C pool:

$$AG_{DM\ res}\_to\_AG_{struct} = AG_{DM\ res} * (1 - AG_{metfrac})$$

[SC.CAR.22]

Where:

- AG_{struct} = above ground structural C (kg ha^{-1})

Similarly to above ground metabolic C, above ground structural C is incorporated into the belowground structural C pool via tillage or decomposed, in this case, equally into active and slow C pools. The calculated values are subtracted from the above ground structural pool and added to the relevant pool.

$$AG_{struct\ decomp} = K1 * e^{-3} * 1 - AG_{met\ frac}$$

[SC.CAR.23]

Where:

- $AG_{\text{struct decomp}}$ = rate at which above ground structural C decomposes into slow or active C
- K1 = structural decomposition factor, set at 0.076 (**parton1987analysis**)

$$AG_{struct}\_to\_C_{active} = AG_{struct\ decomp} * M_d * T_d * AG_{struct}$$
$$AG_{struct}\_to\_C_{slow} = AG_{struct_{decomp}} * M_d * T_d * AG_{struct}$$

[SC.CAR.24]

Where:

- AG_{struct}:C_{active} = above ground structural C decomposed into active C (kg ha^{-1})
- AG_{struct}:C_{slow} = above ground structural C decomposed into slow C (kg ha^{-1})

$$AG_{struct}\_to\_BG_{struct} = AG_{struct} * tillage_{frac}$$

[SC.CAR.25]

Where:

- AG_{struct}:BG_{struct} = transfer of structural carbon during tillage (kg ha^{-1})

Below Ground: Lignin, Metabolic C, Structural C

As above ground, the residue below ground is partitioned into metabolic and structural C pools based on the lignin content of the residue. In the case of tillage, some residue is incorporated from the above ground pool. In the case of tillage, below ground residue will be a combination of below ground and incorporated biomass.

$$BG_{DM\ res} = AG_{DM\ res} * tillage_{frac}$$

[SC.CAR.26]

$$BG_{lignin\ res\ frac} = \left(\frac{BG_{DM\ res}}{BG_{DM\ res} + BG_{bio}}\right) - 0.15 * rainfall * 0.01$$

[SC.CAR.27]

Where:

- $\mathrm{BG_{lignin\ res}}$= below ground residue comprised of lignin (fraction)
- $\mathrm{BG_{DM\ res}}$ is the below ground residue mass (kg ha^{-1})
- $\mathrm{BG_{bio}}$ = below ground biomass (kg ha^{-1})

The below ground ratio of lignin to N will then be a weighted sum of the lignin to N ratio in the below ground and the incorporated biomass.

$$BG_{L:N} = AG_{L:N} * (\frac{BG_{DM\ \ res}}{BG_{DM\ \ res}+BG_{bio}}) + (\frac{BG_{lignin\ \ res\ \ =frac}}{fr_N*100}) * (1 - (\frac{BG_{DM\ \ res}}{BG_{DM\ \ res}+BG_{bio}}))$$

[SC.CAR.28]

Where:

- $\mathrm{BG_{L:N}}$= below ground lignin to N ratio

The component of below ground residue that is metabolic is determined by the below ground lignin to N ratio, with a maximum composition of 85%.

$$BG_{met\ \ frac} = 0.85 - 0.18 * BG_{L:N}$$

[SC.CAR.29]

Where:

- $\mathrm{BG_{met\ frac}}$= below ground residue that is metabolic (fraction)

Incorporated metabolic C and the component of below ground biomass consisting of metabolic C are added to the below ground metabolic C pool.

$$BG_{met}\_updated = BG_{met} + AG_{met}\_to\_BG_{met} + (BG_{bio} * BG_{met\ \ frac})$$

[SC.CAR.30]

Where:

- $\mathrm{BG_{met}}$=below ground metabolic C (kg ha^{-1})

Below ground metabolic C decomposes and the calculated mass is removed from below ground metabolic C and added to below ground active C.

$$BG_{met}\_to\_C_{active} = BG_{met_{active\ \ decomp}} * M_d * T_d * BG_{met}$$

[SC.CAR.31]

Where:

- $BG_{met\_to\_C_{active}}$ = below ground metabolic C decomposed into active C (kg ha^{-1})
- $BG_{met\ active\ decomp}$ = rate at which below ground metabolic C decomposes into active C (set at 0.35, **parton1987analysis**)

Incorporated structural and the component of below ground biomass consisting of structural C are added to the below ground structural C pool.

$$BG_{struct}\_updated = BG_{struct} + AG_{struct}\_to\_BG_{struct} + BG_{bio} * (1 - BG_{met\ frac})$$

[SC.CAR.32]

Decomposition then reduces below ground structural C by the sum of the calculated values and equal amounts of mass are added to the below ground active and slow C pools.

$$BG_{struct}\_to\_C_{active} = BG_{struct\ decomp} * M_d * T_d * BG_{struct}$$
$$BG_{struct}\_to\_C_{slow} = BG_{struct_{decomp}} * M_d * T_d * BG_{struct}$$

[SC.CAR.33]

Where:

- $BG_{struct\_to\_C_{active}}$ = below ground structural C decomposed into active C (kg ha^{-1})
- $BG_{struct\_to\_C_{slow}}$ = below ground structural C decomposed into slow C (kg ha^{-1})
- $BG_{struct\ decomp}$ = rate at which below ground structural C decomposes into slow or active C (set at 0.094, **parton1987analysis**)
- BG_{struct} = below ground structural C (kg ha^{-1})

Pool and Gas Partitioning (implemented in pool_gas_partition.py) Following the residue biomass transfers detailed above, C pools are then updated to account for the imperfect transfer of decomposition between the metabolic and structural C pools to the active and slow C pools. The amount of C lost to the atmosphere as CO2 during decomposition processes are detailed below. Calculated CO2 losses are subtracted from their source pool and added to the relevant atmospheric pool.

$$AG_{met}\_to\_C_{active\ loss} = AG_{met}\_to\_C_{active} * frac\_CO2_{met\_to\_active}$$
$$AG_{met}\_to\_C_{active\ act} = AG_{met}\_to\_C_{active} * (1 - frac\_CO2_{met\_to\_active})$$

[SC.CAR.34]

Where:

- frac_CO2$_{\text{met\_to\_active}}$ = rate of C dioxide loss during transformation of metabolic to active C (0.55, **parton1987analysis**)
- AG$_{\text{met}}$_to_C$_{\text{active loss}}$ = above ground metabolic C being lost as carbon dioxide during decomposition into active C (kg ha^{-1})
- AG$_{\text{met}}$_to_C$_{\text{active act}}$ =above ground metabolic C decomposed to active carbon after accounting for C dioxide loss (kg ha^{-1})

$$AG_{struct\_to\_C_{active\ loss}} = AG_{struct\_to\_C_{active}} * frac\_CO2_{struct\_to\_active}$$
$$AG_{struct\_to\_C_{active\ act}} = AG_{struct\_to\_C_{active}} * (1 - frac\_CO2_{struct\_to\_active})$$

[SC.CAR.35]

Where:

- frac_CO2$_{\text{struct active}}$ = rate of CO_2 loss during transformation of structural to active C (0.45, (**parton1987analysis**))
- AG$_{\text{struct}}$_to_C$_{\text{active act}}$ = above ground structural C decomposed to active C after accounting for CO_2 loss C (kg ha^{-1})
- AG$_{\text{struct}}$_to_C$_{\text{active loss}}$ = above ground structural C being lost as CO2 during decomposition into active C (kg ha^{-1})

$$AG_{struct\_to\_C_{slow\ loss}} = AG_{struct\_to\_C_{slow}} * frac\_CO2_{struct\_to\_slow}$$
$$AG_{struct\_to\_C_{slow\ act}} = AG_{struct\_to\_C_{slow}} * (1 - frac\_CO2_{struct\_to\_slow})$$

[SC.CAR.36]

Where:

- AG$_{\text{struct}}$_to_C$_{\text{slow loss}}$ = decomposing above ground structural C being lost as CO2 during decomposition into slow C (kg ha^{-1})
- AG$_{\text{struct}}$_to_C$_{\text{slow act}}$ = above ground structural C decomposed to slow C after accounting for CO2 loss (kg ha^{-1})
- frac_CO2$_{\text{struct\_to\_slow}}$ = rate of CO2 loss during transformation of structural to slow C (0.3, Partonet al. 1987)

$$BG_{met\_to\_C_{active\ loss}} = BG_{met\_to\_C_{active}} * frac\_CO2_{met\_to\_active}$$
$$BG_{met\_to\_C_{active\ act}} = BG_{met\_to\_C_{active}} * (1 - frac\_CO2_{met\_to\_active})$$

[SC.CAR.37]

Where:

- $BG_{met}\_to\_C_{active\ loss}$ = below ground metabolic C being lost as CO_2 during decomposition into active C (kg ha^{-1})
- $BG_{met}\_to\_C_{active\ act}$ = below ground metabolic C decomposed to active C after accounting for CO_2 loss (kg ha^{-1})

$$BG_{struct}\_to\_C_{active\ loss} = BG_{struct}\_to\_C_{active} * (frac\_CO2_{struct\_to\_active})$$
$$BG_{struct}\_to\_C_{active\ act} = BG_{struct}\_to\_C_{active} * (1 - frac\_CO2_{struct\_to\_active})$$

[SC.CAR.38]

Where:

- $BG_{struct}\_to\_C_{active\ loss}$ = below ground structural C being lost as CO_2 during decomposition into active C (kg ha^{-1})
- $BG_{struct}\_to\_C_{active\ act}$ = below ground structural C decomposed to active C after accounting for CO_2 loss (kg ha^{-1})

$$BG_{struct}\_to\_C_{slow\ loss} = BG_{struct}\_to\_C_{slow} * (frac\_CO2_{struct\_to\_slow})$$
$$BG_{struct}\_to\_C_{slow\ act} = BG_{struct}\_to\_C_{slow} * (1 - frac\_CO2_{struct\_to\_slow})$$

[SC.CAR.39]

Where:

- $BG_{struct}\_to\_C_{slow\ loss}$ = below ground structural C being lost as CO_2 during decomposition into active C (kg ha^{-1})
- $BG_{struct}\_to\_C_{slow\ act}$ = below ground structural C decomposed to active C after accounting for CO_2 loss (kg ha^{-1})

Decomposition losses, mass transfers between pools, and totals are updated following pooling and gas partitioning:

$$AG\_to\_C_{active} = AG_{met}\_to\_C_{active\ act} + AG_{struct}\_to\_C_{active\ act}$$
$$BG\_to\_C_{active} = BG_{met}\_to\_C_{active\ act} + BG_{struct}\_to\_C_{active\ act}$$
$$C_{active} = AG\_to\_C_{active} + BG\_to\_C_{active} + C_{passive}\_to\_C_{active} + C_{slow}\_to\_C_{active} - C_{active\ decomp}$$

[SC.CAR.40]

Where:

- $AG\_to\_C_{active}$ = above ground C decomposed into the active carbon pool (kg ha^{-1})
- $BG\_to\_C_{active}$ = below ground C decomposed into the active carbon pool (kg ha^{-1})

$$C_{slow} = AG_{struct\_to\_C_{slow\ act}} + BG_{struct\_to\_C_{slow\ act}} + C_{active\_to\_slow} - C_{slow\ decomp}$$
$$C_{passive} = C_{slow\_to\_passive} + C_{active\_to\_passive} - C_{passive\ decomp}$$

[SC.CAR.41]

$$AG_{CO2\ loss} = AG_{met\_to\_C_{active\ loss}} + AG_{struct\_to\_C_{active\ loss}} + AG_{struct\_to\_C_{slow\ loss}}$$
$$BG_{CO2\ loss} = BG_{met\_to\_C_{active\ loss}} + BG_{struct\_to\_C_{active\ loss}} + BG_{struct\_to\_C_{slow\ loss}}$$

[SC.CAR.42]

Soil Carbon Aggregation (implemented in carbon_ cycling.py)

$$soil\ volume = depth * 10 * area$$
$$soil\ mass = \frac{BD}{0.001} / soil\ volume$$

[SC.CAR.43]

$$C_{active\ \%} = \left(\frac{C_{active}}{soil\ mass}\right)$$
$$C_{slow\ \%} = \left(\frac{C_{slow}}{soil\ mass}\right)$$
$$C_{passive\ \%} = \left(\frac{C_{passive}}{soil\ mass}\right)$$

[SC.CAR.44]

Where:

- C_{active} % = active C composition of the soil (%)
- C_{slow} % = slow C composition of the soil (%)
- C_{passive} % = passive C composition of the soil (%)

$$C_{\%} = C_{active\%} + C_{slow\%} + C_{passive\%}$$

[SC.CAR.45]

Where:

- C% = carbon composition of the soil (%)

$$C = C_{active} + C_{slow} + C_{passive}$$
$$C_{mg} = C * 1000000$$
$$C_g = C * 1000$$

[SC.CAR.46]

Where:

- C = soil carbon (kg ha^{-1})
- C_{mg} = soil carbon (mg ha^{-1})
- C_g = soil carbon (g ha^{-1})

$$AG_{CO2\ loss} = AG_{met} : C_{active\ loss} + AG_{struct} : C_{active\ loss} + AG_{struct} : C_{slow\ loss}$$
$$BG_{CO2\ loss} = BG_{met} : C_{active\ loss} + BG_{struct} : C_{active\ loss} + BG_{struct} : C_{slow\ loss}$$

[SC.CAR.47]

Where:

- $AG_{CO_2\ loss}$ = above ground C lost as CO_2 (kg ha^{-1})
- $BG_{CO_2\ loss}$ = below ground C lost as CO_2 (kg ha^{-1})

$$C_{CO2\ loss} = AG_{CO2\ loss} + BG_{CO2\ loss} + C_{CO2\ loss\ decomp}$$

[SC.CAR.43]

Where:

- $C_{CO_2\ loss}$ = total C lost as CO_2 (kg ha^{-1})

23 Crop Management

23.a Introduction

Crop growth and development in RuFaS is based on the SWAT model and is driven by biomass accumulation based on the available heat units adsorbed by the plant/plant parts, and available nutrient and water uptake from the soil by the crop roots. Absorbed heat units and leaf area index drive potential biomass accumulation and the availability of soil nutrients and water modulate the potential crop growth via stress functions.

We simulate three organic N pools (fresh, active, and stable) and two inorganic pools (NO_3^- and NH_4^+).

Initial NO_3 levels (mg kg^{-1}) in the soil are set by user input. If user input is null, initial nitrate concentrations are generated as:

Planting and Initialization When the Field class plants a new crop, it calls the create_crop() method of the Crop class to initialize a new crop with the appropriate crop configuration inputs. The crop configurations and the default input values for each of the default crop options are provided in the table below. The key inputs to ensure that the crop that is grown will be harvested, stored by the Feed Module, and fed to animals by the Animal Module as intended are:

- Crop category
- Crop type
- RuFaS feed IDs associated with this feed
- Storage type

All other inputs drive the biological processes and ultimately determine the rate and extent of biomass accumulation in response to weather and nutrient availability.

Daily Biophysical Processes The daily crop processes are called by the Field class and are executed in two methods. First the water cycling is executed by the Field class cycle_water() method in which the crop's water cycle is called last. Then the daily crop updates are called for each crop using the Crop class's perform_daily_crop_update() method.

Harvest At the end of the Field class's manage_field() routine, it checks for both user scheduled and optimal harvest events and initiates the Crop routine's harvest if it is a harvest day.

23.b Methodology

Crop Water Cycle Daily Updates The water cycle within a crop is executed by methods in the WaterUptake class and the WaterDynamics and are mostly called by the Crop class method cycle_water_-for_crop(). First, water is taken up from the soil into the crop by water_uptake.uptake() method and then water is distributed through the crop water cycle by the water_dynamics.cycle_water() method. The maximum potential water evaporation and transpiration from the crop, however, are called directly from the Field class's cycle_water() method because these values are needed to calculate the field water balance and soil evaporation.

Crop Evaporation The maximum potential crop canopy evaporation and transpiration are calculated as a function of the total potential evapotranspiration for the field on that day (see the Soil documentation).
After the total potential evapotranspiration (mm H_2O) is calculated in the Field class, evaporation from the crop canopy is estimated as:

$$crop\_evaporation = min(crop\_canopy\_water,\ max\_potential\_evapotranspiration)$$

[SC.CRP.1]

Crop Canopy Water The amount of water in the crop's canopy is calculated by the Crop class's handle_water_in_canopy() method. Each crop has a crop-specific maximum water capacity (mm) when the crop is fully matured and this value informs the current day's maximum water storage capacity (mm) based on the ratio of the current leaf area index (LAI) and the maximum LAI:

$$water\_canopy\_storage\_capacity\ =\ max\_water\_capacity * \frac{leaf\_area\_index}{LAImax}$$

[SC.CRP.2]

The potential excess storage capacity is calculated using the maximum water storage capacity and the actual amount of water in the canopy on that day:

$$canopy\_water\_excess\_capacity \ = \ water\_canopy\_storage\_capacity \ - canopy\_water$$

[SC.CRP.3]

The amount of additional water stored in the canopy is then calculated as:

$$water\_taken\_to\_be\_stored \ = \ min(canopy\_water\_excess\_capacity, \ precipitation)$$

[SC.CRP.4]

And added to the current canopy water:

$$canopy\_water = canopy\_water\_previous + water\_taken\_to\_be\_stored$$

[SC.CRP.5]

Finally, the remaining precipitation is passed by the Field class to the Soil as the precipitation that reaches the soil:

$$precipitation\_reaching\_soil \ = precipitation - water\_taken\_to\_be\_stored$$

[SC.CRP.6]

Crop Potential Transpiration The Field's cycle_water method then subtracts the amount of water evaporated from the crop to calculate the remaining evapotranspiration demand (mm H_2O) and that value is passed to the Crop class's method to calculate the maximum crop transpiration (mm H_2O):

$$max\_transpiration = remaining\_evapotranspiration\_demand * leaf\_area\_inde * 3.0 \ if \ 0 \ \leq LAI_{act} \leq 3.0$$
$$max\_transpiration = remaining\_evapotranspiration\_demand \ if \ LAI_{act} > 3.0$$

[SC.CRP.6]

Potential Water Uptake The potential water uptake from the soil surface to any specified depth in the root zone can be calculated as:

$$max\_water\_uptake_{depth} = \frac{max\_transpiration}{[1-exp(-water\_dist\_param)]} * [1 - exp(-water\_dist\_param * \frac{depth}{root\_depth})]$$

[SC.CRP.7]

where max_transpiration is the maximum plant transpiration on a given day (mm H_2O), water_dist_-param is the water-use distribution parameter, depth is the depth from the soil surface (mm), and root_depth is the depth of root development in the soil (mm).
The potential maximum water uptake from any soil layer, then, can be calculated by taking the difference in uptake between the top and bottom of the soil layer

$$max\_water\_uptakelayer = max\_water\_uptakebottom\_depth - max\_water\_uptaketop\_depth$$

[SC.CRP.8]

where $max\_water\_uptake_{bottom\_depth}$ is the potential water uptake for the lower boundary of the soil layer (mm H_2O) and $max\_water\_uptake_{top\_depth}$ is the potential water uptake for the upper boundary of the soil layer (mm H_2O).
Potential water uptakes from each layer are then calculated sequentially from the soil surface to the maximum depth by layer to allow for compensation for unmet demand set by the maximum water uptake by the available water in lower layers.
The unmet demands for water are calculated as sum of the unmet demands for that layer and all layers above it:

$$unmet\_demand_{layer\_i} = \sum_{x=1}^{i} max\_water\_uptakex - soil\_water$$

[SC.CRP.9]

The unmet demands are then used to calculate the potential water uptakes for each layer from the maximum water uptake and the unmet demands from each layer for every layer below the surface:

$$potential\_water\_uptakelayer = \\ max\_water\_uptake_{layer} + unmet\_demand_{layer} * uptake\_compensation\_factor$$

[SC.CRP.10]

where potential_water_uptake_layer is the adjusted potential water uptake for layer (mm H_2O), uptake_compensation_factor is the plant uptake compensation factor which is a user-defined input for each crop that ranges between 0.01 and 1.00. When the uptake_compensation_factor approaches 1.00 the model allows more of the water uptake demand to be met by lower layers in the soil; when it approaches 0.00, the model allows less variation from the original depth distribution to take place.
As the water content of the soil decreases, the soil holds remaining water more tightly, resulting in a decrease in the efficiency of plant water uptake. Thus, the potential water uptakes are adjusted one more time if the initial soil water content is below a threshold:

$$if soil\_water_{layer} < 0.25 \ * available\_water\_capcity_{ayer} : \\ adjusted\_potential\_water\_uptake_{layer} = \\ potential\_water\_uptake_{layer} * exp[5 * (\frac{soil\_water_{layer}}{0.25 * available\_water\_capcity_{layer}} - 1)]$$

[SC.CRP.11]

where $available\_water\_capcity_{layer}$ is the point at which plant available water in the soil layer begins to limit efficiency of plant uptake, $soil\_water_{layer}$ is the soil water in a given layer (mm H_2O)

Actual water uptake The actual water uptake for each layer are then calculated as the minimum of the adjusted potential water uptakes and the difference between the available soil water in that layer and the wilting point soil water.

$$actual\_water\_uptake\_layer = \\ min[adjusted\_potential\_water\_uptake_{layer}, (soil\_water_{layer} - wilting\_point_{layer})]$$

[SC.CRP.12]

where *actual_water_uptake_layer* is the actual water for the layer (mm H_2O), $wilting\_point_{layer}$ is the water content of the layer at wilting point (mm H_2O).
The total plant uptake for the entire profile for the day (mm H_2O) is calculated as the sum of these values for each layer in the profile.

$$total\_water\_uptake = \sum_{x=1}^{n} actual\_water\_uptake\_later_i$$

[SC.CRP.13]

Where $actual\_water\_uptake\_later_i$ is the actual water for layer (mm H_2O) and n is the number of layers in the soil profile.
The total plant water uptake is passed to WaterDynamics *cycle_water*() method as the actual amount of transpiration on the day (mm H_2O).

$$actual\_transpiration = total\_water\_uptake$$

[SC.CRP.14]

Crop Growth and Nutrient Cycle Daily Updates

The second set of daily process updates is called by the execute_daily_processes() method in the Field class which in turn calls perform_daily_crop_update() in the Crop class. Similar to other daily update functions, this method steps through a number of functions to initiate simulation of the biophysical processes that drive crop growth and development. These processes are:

- Check if the crop is mature or is dormant
- Absorb solar radiation in the form of heat units and determine the max potential growth for that day
- Develop roots
- Uptake nitrogen from the soil
- Uptake phosphorus from the soil
- Calculate and set physical constraints to growth
- Grow the canopy of the crop by increasing the leaf area index
- Increase and allocate total crop biomass

Maturity A crop's advance towards maturity is tracked through the accumulation of heat units which are a theoretical unit developed to estimate the amount of solar radiation absorbed by the plant. Each crop has a crop specific value for the potential heat units (PHU) required to reach maturity. The accumulated fraction of potential heat units (FrPHU) on a given day is calculated as:

$$fr_{PHU} = \frac{HU_{sum}}{PHU}$$

[SC.CRP.15]

where HU_{sum} is the heat units accumulated up to the current day and PHU is the crop-specific total heat units required for maturity. HU_{sum} is calculated from day 1 (planting day) to the day until the plant reaches maturity.

Dormancy Perennial crops like alfalfa enter a period of dormancy in which the plants do not grow as the day length nears the shortest or minimum day length for the year. The beginning and end of dormancy are defined by a day length threshold (hrs) which can be calculated as:

$$daylength\_threshold = daylength\_min + dormancy\_threshold$$

[SC.CRP.16]

where $daylength\_min$ is the minimum day length for the location during the year (hrs), $dormancy\_threshold$ is the dormancy threshold (hrs). Plants enter dormancy when the day length falls below the threshold and exit dormancy when day length exceeds the threshold again in the next year.
The dormancy threshold varies with latitude and is calculated as:

$$\begin{gathered} t_{dorm} = 1.0;\ if\ latitude\ >\ 40^\circ\ N\ or\ S \\ t_{dorm} = \frac{(latitude-20)}{20};\ if\ 20^\circ\ N\ or\ S \leq\ latitude \leq 40^\circ\ N\ or\ S \\ t_{dorm} = 0.0;\ if\ latitude\ <\ 20^\circ\ N\ or\ S \end{gathered}$$

[SC.CRP.17]

where the latitude is expressed as a positive value regardless of cardinality (degrees).
The total daylight hours (daylength, hrs) for each day is calculated by the Weather class each day and passed to the FieldManager:

$$daylength = \frac{2cos^{-1}[-tan(solar\_declination\_radians)tan(latitude)]}{earth\_angular\_velocity}$$

[SC.CRP.18]

where the solar declination in radians is the earth's latitude at which incoming solar rays are normal to the earth's surface and the angular velocity of the earth's rotation is a constant equal to 0.2618 rad h^{-1} or 15° h^{-1}.

$$solar\_declination\_radians = sin^{-1}\{0.4 * sin[\frac{2\pi}{year\ length}(day - 82)]\}$$

[SC.CRP.19]

where year length is the number of days in a given year, and the day is Julian day.
At the beginning of the dormant period for perennials (like alfalfa), 10% of the biomass is converted to residue and the LAI for the species is set to the minimum value allowed for the species (default 0.75). For cool season annuals like winter cereal grain species (rye, triticale, wheat, etc), their biomass is not converted to residue. When the day length first drops below the dormancy threshold, the dormancy routine is run once on the first day of the dormancy period.
If it is a perennial crop, the model checks whether the perennial should be cut:

$$if\ (fr_{PHU} \geq fr_{PHU,harvest,min}) : yield$$

[SC.CRP.20]

Where:

- $\mathrm{fr}_{\mathrm{PHU}}$ is the fraction of potential heat units.
- $\mathrm{fr}_{\mathrm{PHU,harvest,min}}$ is the minimum fraction of potential heat units acquired to warrant harvest (user input).

The parameters that drive growth ($\mathrm{fr}_{\mathrm{PHU}}$, $\mathrm{HU}_{\mathrm{sum}}$, $\mathrm{fr}_{\mathrm{LAI,max}}$) are reset to 0 and crop biomass, nitrogen, and phosphorus are reduced by 10% of their value at the time of harvest. Biomass lost when entering dormancy is added to residue.

Accumulation of Heat Units The available heat units on a given day are calculated by the determine_new_heat_units() method in the HeatUnits class:

$$HU = T_{HU} - T_{min};\ when\ T_{HU} > T_{base,min}$$
$$tHU = O;\ if\ T_{HU} - T_{min} < 0$$

[SC.CRP.21]

where T_{HU} is either the mean daily temperature ($°C$) or the temperature used to define the maximum heat units for the day; T_{min} is the base or minimum temperature required for the plant growth ($°C$).
By default, T_{HU} is set to the mean daily ambient temperature. If the attribute, use_heat_unit_-temperature is set to True, then THU is defined by the equation below:

$$T_{HU} = \frac{T_{HU,max} + T_{HU,min}}{2}$$

[SC.CRP.22]

where $T_{HU,max}$ is the maximum heat unit temperature on a given day ($°C$), $T_{HU,min}$ is the minimum heat unit temperature on a given day ($°C$)

$$T_{HU,min} = T_{crop,min}$$
$$T_{HU,min} = T_{ambient,min};\ if\ T_{ambient,min} > T_{crop,min}$$

[SC.CRP.23]

$$T_{HU,max} = T_{crop,max}$$
$$T_{HU,max} = T_{ambient,max};\ if\ T_{ambient,max} > T_{crop,max}$$

[SC.CRP.24]

where $T_{crop,max}$and $T_{crop,min}$ are the crop-specific maximum and minimum temperatures required to sustain growth ($°C$) and $T_{ambient,max}$and $T_{ambient,min}$ are the maximum and minimum ambient temperatures that day.
The effect of using the alternative method on heat unit accumulation varies depending on the relationship between the air temperature range and the crop's growth temperature range:

1. If both min and max air temperatures are higher than the crop's min and max growth temperatures, accumulation is greater than the main method.
2. If both min and max air temperatures are lower than the crop's min and max temperatures, accumulation is greater than the main method.

3. If the air temperature range is entirely within the crop's temperature range, accumulation equals the middle of the crop temperature window.
4. If the crop's temperature range is entirely within the air temperature range, accumulation equals the middle of the air temperature range.

Once the heat units for that day have been calculated, they are added to the previously accumulated heat units by the *add_heat_units*() method.

Root Development The depth of the roots on a given day is estimated based on the fraction of accumulated heat units fr_{PHU} and the maximum rooting depth. First a root specific fraction is calculated:

$$root = 0.40 - 0.20 * fr_{PHU}; if\ 2 \geq fr_{PHU} \geq 0$$
$$root\_fraction = 0; if\ 2 < fr_{PHU} < 0$$

[SC.CRP.25]

where $root\_depth_{max}$ is the maximum depth of root development in the soil (mm).

$$root\_depth = root\_depth_{max}$$

[SC.CRP.26]

If the crop type is perennial, and it is not the establishment/planting year then the rooting depth of the given plant (root_depth) (mm) is defined as the depth of root development in the soil on a given day and calculated as:

$$root\_depth = root\_depth_{max}; if\ fr_{PHU} > 0.40$$
$$root\_depth = 2.5 * fr_{PHU} * root\_depth_{max}; if\ fr_{PHU} \leq 0.40$$

[SC.CRP.27]

where root_depthmax is the maximum depth of root development in the soil (mm).

Nitrogen Uptake

Each crop's nitrogen uptake routines are managed by the uptake() method in the NitrogenUptake class. This function executes all nitrogen incorporation routines. It calculates the amount of nitrogen the plant desires based on its current growth stage and the available nitrogen in the soil. Then, it extracts nitrogen from the accessible soil profile. If there's any unmet nitrogen demand, the plant may attempt to fix atmospheric nitrogen. The nitrogen from both extraction and fixation is then added to the plant's biomass, contributing to its growth.

Potential Nitrogen Uptake A crop's demand for nitrogen (and phosphorus), also known as the potential nitrogen uptake, at different stages of development is driven by the optimal composition or fraction of that nutrient in the plant's biomass at that stage of development, the total biomass, and the amount of that nutrient in the plant on the previous day.
The optimal composition of a crop for both nitrogen and phosphorus is predicted based on a curve between the crop-specific, user-provided compositions at emergence ($fr_{N,1}$) and maturity ($fr_{N,3}$).
Following this logic, the function for determining the optimal fraction of N in the plant biomass on a given day (fr_n) is:

$$fr_n = (fr_{N,1} - fr_{N,3}) * [1 - \frac{fr_{PHU}}{fr_{PHU} + exp(n_1 + n_2 * fr_{PHU})}] + fr_{N,3}$$

[SC.CRP.28]

where $fr_{N,1}$ is the fraction of N in the plant biomass at emergence, $fr_{N,3}$ is the fraction of N in the plant biomass at maturity, fr_{PHU} is the fraction of potential heat units accumulated for the plant on a given day in the growing season (described in **[SC.CRP.15]**), and n_1 and n_2 are known as the shape coefficients.
The shape coefficients are calculated by solving the below equations using the composition of the plant at two known points in crop development: half maturity ($fr_{PHU,50\%}$) and maturity ($fr_{PHU,100\%}$). The second shape coefficient is needed to calculate the first shape coefficient and is calculated as:

$$n_2 = \frac{\ln\left(\frac{fr_{PHU,50\%}}{1-\left(\frac{fr_{N,2}-fr_{N,3}}{fr_{N,1}-fr_{N,3}}-fr_{PHU,50\%}\right)}\right)-\ln\left(\frac{fr_{PHU,100\%}}{1-\left(\frac{fr_{N,3}-fr_{N,3}}{fr_{N,1}-fr_{N,3}}-fr_{PHU,100\%}\right)}\right)}{fr_{PHU,100\%}-fr_{PHU,50\%}}$$

[SC.CRP.29]

The first shape coefficient is then calculated as:

$$n_1 = \ln\left[\frac{fr_{PHU,50\%}}{1-(\frac{fr_{N,2}-fr_{N,3}}{fr_{N,1}-fr_{N,3}})} - fr_{PHU,50\%}\right] + n_2 * fr_{PHU,50\%}$$

[SC.CRP.30]

where n_1 is the first shape coefficient, n_2 is the second shape coefficient, $fr_{N,1}$ is the fraction of N in plant biomass at emergence, $fr_{N,2}$ is the the fraction of N in the plant biomass at 50% maturity, $fr_{N,\ 3}$ is the fraction of N in the plant biomass near maturity, $fr_{N,3}$ the normal fraction of N in plant biomass at maturity, $fr_{PHU,50\%}$ is the fraction of potential heat units accumulated for the plant at 50% maturity and $fr_{PHU,100\%}$ is the fraction of potential heat units accumulated for the plant at 100% maturity.
We follow the SWAT model assumption that the difference between the nutrient fraction near maturity and the nutrient fraction at maturity in the crop is equal to 0.00001 and the near mature nutrient fraction is calculated from the mature nutrient fraction according to this assumption.
Using the optimal fraction of N in the crop biomass calculated in **[SC.CRP.28]** the optimal mass of N stored in the plant biomass on a given day (kg N ha^{-1}) is given as:

$$optimal\_nitrogen\_mass = optimal\_nitrogen\_fractionplant\_biomass$$

[SC.CRP.31]

Where optimal_nitrogen_fraction is the optimal fraction of N in the plant biomass for the current growth stage, and plant_biomass is the total plant biomass on a given day (kg ha^{-1}).
Plant N demand (potential N uptake) on a given day (kg N ha^{-1}) is then calculated as:

$$optimal\_nitrogen\_mass = optimal\_nitrogen\_fractionplant\_biomass$$

[SC.CRP.32]

Where optimal_nitrogen_mass_previous is the actual mass of N stored in the plant material at the end of the previous day (kg N ha^{-1}), and biomass_growth_max is the potential increase in total plant biomass on a given day (kg N ha^{-1}), and is constrained by the condition that nitrogen_uptake ≥ 0

Actual Nitrogen Uptake Once the potential nitrogen uptake has been calculated, the request for nutrients from the crops is distributed through the soil layers, starting from the top down. First the soil layers accessible to the roots based on the depth of the roots on that day is determined by the method find_deepest_accessible_soil_layer() which updates each crop's accessible_soil_layers attribute - an integer that indicates the number of soil layers from the surface down to which the crop roots have access.
The number of accessible soil layers is then used to calculate the bottom depths of each layer and the amount of nitrates in each layer that are accessible to the plant by the methods access_layers in the NonWaterUptake class.
The potential N uptake from the surface to any given depth z ($N_{up,z}$; kg N ha^{-1}) is calculated as:

$$nitrogen\_uptake_{depth} = \frac{nitrogen\_uptake}{[1 - exp(-\beta_n)]} * 1 - exp(-\beta_n * \frac{depth}{root\_depth})$$

[SC.CRP.33]

where nitrogen_uptake_depth is the potential N uptake from the soil surface to the lower boundary of a given depth, nitrogen_uptake is the total potential N uptake (kg N ha^{-1}), β_n is the nutrient uptake distribution parameter, "depth" is the depth from soil surface (mm), and root_depth is the depth of root development in the soil on a given day (mm).
The β_n parameter does not significantly affect N uptake by plants, but it has a significant impact on the maximum amount of NO_3 removed from the upper 10 cm via surface runoff. Higher β_n values allow plants to extract a greater percentage of N_3up from the upper soil layers, which results in less NO_3 loss to runoff. We set this parameter to a default value of 10.
Potential N uptake for each layer is then calculated as the difference in the potential uptake to the bottom depth of the layer and the potential uptake to the top depth of each layer. For the surface layer, where the top depth or upper boundary of the soil layer is at 0 mm depth:

$$potential\_nitrogen\_uptake_{layer} = nitrogen\_uptake_{depth}$$

[SC.CRP.34]

where $nitrogen\_uptake_{layer}$ is the potential N uptake from the surface soil layer (kg N ha^{-1})
For every other layer, the potential N uptake is calculated for the upper and lower boundaries of the layer and $N_{up,ly}$ is taken as the difference between the two:

$$potential\_nitrogen\_uptake_{layer} = nitrogen\_uptake_{depth,bottom} - nitrogen\_uptake_{depth,upper}$$

[SC.CRP.35]

where potential_nitrogen_uptake$_{\text{depth,bottom}}$ is the potential N uptake from the soil surface to the lower boundary of the soil layer (kg N ha^{-1}) and nitrogen_uptake$_{\text{depth,upper}}$ is the potential N uptake from the soil surface to the upper boundary of the soil layer (kg N ha^{-1}).
The actual N uptake from a given soil layer (kg N ha^{-1}) is calculated as the minimum of potential N uptake plus demand, and available NO_3 in the layer and given as:

$$actual\_nitrogen\_uptake_{layer} = \\ min\{nitrogen\_uptake_{layer} + unmet\_nitrogen\_demand,\ soil\_nitrate_{layer}\}$$

[SC.CRP.36]

where unmet_nitrogen_demand is the potential N uptake from the overlying soil layers (kg N ha^{-1}) that was not met by the nutrients in those layers. For the first soil layer where there are no overlying layers, unmet_nitrogen_demand is 0. nitrogen_uptake$_{\text{layer}}$ is the potential N uptake for the soil layer (kg N ha^{-1}), and soil_nitrate is the nitrate content of the soil layer (kg N ha^{-1}).
For every other soil layer the unmet nitrogen demand is calculated as:

$$unmet\_nitrogen\_demand = max(sum\_potential\_nitrogen\_uptake_{above} - sum\_soil\_nitrate_{above}, 0)$$

[SC.CRP.37]

where N$_{\text{up}}$,over is the sum of potential N uptake for each layer overlying the given layer (kg N ha^{-1}) and NO$^{3}{}_{\text{over}}$ is the sum of nitrate content of each soil layer overlying the given layer (kg N ha^{-1}).

$$sum\_potential\_nitrogen\_uptake_{above} = \sum_{layer-1}^{surface} potential\_nitrogen\_uptake_{layer}$$
$$sum\_soil\_nitrate_{above} = \sum_{layer-1}^{surface} soil\_nitrate_{layer}$$

[SC.CRP.38]

where layer -1 is the soil layer above the current layer, S is the surface layer, potential_nitrogen_-uptake$_{\text{layer}}$ is the potential uptake for the given soil layer (kg N ha^{-1}),soil_nitrate$_{\text{layer}}$ is the nitrate content of the given soil layer (kg NO$_3$-N ha^{-1})
actual_nitrogen_uptake$_{\text{total}}$ for the entire profile on a given day is the sum of N$_{\text{actualup}}$ for each layer.

$$actual\_nitrogen\_uptake_{total} = \sum_{z}^{s} actual\_nitrogen\_uptake_{layer}$$

[SC.CRP.39]

Nitrogen Fixation The N fixation module amends the routine and equations described in the SWAT 2009 documentation by limiting calculations of f$_{\text{NO3}}$, f$_{\text{SW}}$, N$_{\text{demand}}$, and N$_{\text{fix}}$ to the portion of the soil profile that is accessible to root biomass. This module also only applies to those crops that have been designated/defined as capable of forming symbiotic N fixation associations (e.g. legumes).
After the amount of N available in the soil to meet the crop's nitrogen demands has been determined, the amount of N added to the plant biomass by fixation (kg N ha^{-1}) is calculated using the unmet demands from the soil:

$$nitrogen\_fixation = unmet\_demand * stage\_factor * min\{water\_factor,\ nitrate\_factor,\ 1\}$$

[SC.CRP.40]

where unmet_demand is the plant N demand not met by uptake from the soil (kg N ha^{-1}) and is also the maximum amount of N that can be fixed by the plant on a given day, stage_factor is the growth stage factor (0.0-1.0), *water_factor* is the soil water factor (0.0-1.0), and nitrate_factor) the soil nitrate factor (0.0-1.0).
The growth stage factor is dependent on the fraction of accumulated potential heat units (PHU) for the plant on a given day in the growing season and can be expressed as:

$$\begin{array}{ll}
stage\_factor = 0; & if\ fr_{PHU} \leq 0.15 \\
stage\_factor = 6.67 * fr_{PHU} - 1; & if\ 0.15 < fr_{PHU} \leq 0.30 \\
stage\_factor = 1; & if\ 0.30 < fr_{PHU} \leq 0.55 \\
stage\_factor = 3.75 - 5 * *fr_{PHU}; & if\ 0.55 < fr_{PHU} \leq 0.75 \\
stage\_factor = 0; & if\ fr_{PHU} > 0.75
\end{array}$$

[SC.CRP.41]

stage_factor reflects the growth and decline of N fixing bacteria in the plant roots during the growing season.
The soil nitrate factor is dependent on the sum of nitrate available in the soil layers accessible to root biomass (kg NO_3-N ha^{-1}) and calculated as:

$$accessible\_nitrates = \sum_{i=1}^{root\_layer} soil\_nitrates_{layer,i}$$

[SC.CRP.42]

where root_layer is the soil layer of lowest depth that is accessible to the root biomass, and soil_-nitrates$_{layer,i}$ is the nitrate content in the given soil layer (kg N ha^{-1}).
The soil nitrate factor is then determined along the continuum:

$$\begin{array}{ll}
nitrate\_factor = 1; & if\ accessible\_nitrates \leq 100 \\
nitrate\_factor = 1.5 - 0.0005 * NO3_{root}; & if\ 100 < accessible\_nitrates \leq 300 \\
nitrate\_factor = 0; & if\ accessible\_nitrates > 300
\end{array}$$

[SC.CRP.43]

The soil water factor controls N fixation as the soil dries out and is calculated in the SoilData class:

$$water\_factor = \frac{root\_accessible\_soil\_water}{0.85 * root\_accessible\_field\_capacity}$$

[SC.CRP.44]

where root_accessible_soil_water is the soil water content in the soil profile accessible to plant root biomass (mm H_2O) and root_accessible_field_capacity is the soil water content accessible to root biomass at field capacity (mm H_2O).
root_accessible_soil_water and $0.85 * root\_accessible\_field\_capacity\_t$ are calculated as the sum of total soil water and soil water field capacity in each layer for the soil layers accessible to the root biomass.

Store Nitrogen in Plant The actual nitrogen stored in plant biomass on a given day (kg N ha^{-1}) is calculated as:

$$crop\_nitrogen = previous\_crop\_nitrogen + N_{actualup} + N_{fix}$$

[SC.CRP.45]

where *previous_crop_nitrogen* is the N stored in plant biomass on the previous day (kg N ha^{-1}), $N_{actualup}$ is the actual N uptake for the entire soil profile (kg N ha^{-1}), andN_{fix} is the amount of N added to the plant biomass by fixation (kg N ha^{-1}) for perennial crops.

Phosphorus Uptake

The methods for estimating crop phosphorus uptake and addition to the plant's stored phosphorus are analogous to the methods for crop nitrogen uptake and both the PhosphorusUptake class and the NitrogenUptake class use the same generic methods that are inherited from the NonWaterUptake class to implement these calculations. The pool of phosphorus available for uptake by the crop is labile inorganic phosphorus.
To summarize:

- First the potential phosphorus uptake is estimated based on the demand from the crop given its current biomass, stage of development, and estimated phosphorus composition at that stage of development. These methods are described in **[SC.CRP.28] - [SC.CRP.32]** substituting phosphorus for nitrogen.
- Then the actual phosphorus uptake is calculated based on the number of soil layers that the roots have access to and the amount of labile inorganic phosphorus in each soil layer. These methods are described in **[SC.CRP.33] - [SC.CRP.44]** again, substituting phosphorus for nitrogen.
- Finally the phosphorus taken up by the roots are added to the amount of phosphorus stored in the plant as described in **[SC.CRP.45]** with the obvious exception that there is never any fixed phosphorus.

Growth Constraints

The GrowthConstraints class includes all of the methods that estimate the potential limitations on plant growth that a crop faces due to environmental conditions. Specifically, these methods calculate a stress factor for each of the following:

- Insufficient or excess water
- Inadequate nitrogen
- Inadequate phosphorus
- Extreme temperature

The stress factors have values between [0,1] and represent the proportional reduction in total potential biomass that is accumulated that day as a result of that environmental stressor. After calculating the stress factor for each condition separately, the largest stressor is used to define the growth factor for that day:

$$growth\_factor = 1 - max(water\_stress,\ temp\_stress,\ nitrogen\_stress,\ phosphorus\_stress)$$

[SC.CRP.46]

If the user wishes to run a simulation in which crop growth ignores these environmental stressors, they can do so through the user inputs "simulate_water_stress", "simulate_temp_stress", "simulate_nitrogen_-stress", and "simulate_phosphorus_stress" in each field. The default values for these are set to true, indicating that crop growth should take environmental stressors into account but setting any of these input values to false will "turn off" that stressor and return a value of 0 for the associated stress factor.

Water stress Water stress is the stress caused by the soil water conditions on a given day. The water stress for a given day is calculated as:

$$water\_stress = 1 - \frac{w_{actualup}}{E_t}$$

[SC.CRP.47]

where $w_{actualup}$ is the daily total plant water uptake (mm H_2O) and E_t is the actual amount of transpiration on a given day (mm H_2O).

Temperature stress Temperature stress is experienced by the crops when the mean air temperature on a given day diverges from the optimal temperature for the plant growth. If the average temperature is less than or equal to the minimum temperature required for crop growth, also called the base temperature, then the temperature stress for that day will be 1 and the crop will not accumulate any biomass.

$$temp\_stress = 1; \qquad if\ \bar{T}_{ave} \leq T_{base}$$

[SC.CRP.48]

where $\bar{T}_{ave}$ is the mean air temperature ($°C$) and T_{base} is the plant-specific minimum temperature for growth ($°C$).
If the base temperature is less than the average temperature and the average temperature is less than or equal to the optimal temperature, then temperature stress is calculated as:

$$temp\_stress = 1 - exp[\frac{-0.1054(T_{opt} - \bar{T}_{ave})^2}{(\bar{T}_{ave} - T_{base})^2}]; \quad if\ T_{base} < \bar{T}_{ave} \leq T_{opt}$$

[SC.CRP.49]

where T_{opt} is the plant-specific optimal temperature for growth ($°C$) If the optimal temperature is less than the average temperature and the average temperature is less than or equal to two times the optimal temperature minus the base temperature, then temperature stress for a given day is calculated as:

$$temp\_stress = 1 - exp[\frac{-0.1054(T_{opt} - \bar{T}_{ave})^2}{(2 * T_{opt} - \bar{T}_{ave} - T_{base})^2}]; \quad if\ T_{opt} < \bar{T}_{ave} \leq 2 * T_{opt} - T_{base}$$

[SC.CRP.50]

If the average temperature is greater than two times the optimal temperature minus the base temperature, then temperature stress is set to 1 and results in no biomass accumulation,

$$temp\_stress = 1; \qquad if\ \bar{T}_{ave} > 2 * T_{opt} - T_{base}$$

[SC.CRP.51]

Nitrogen and Phosphorus Stress Nitrogen and phosphorus stress are calculated using the same functions based on a non-linear relationship between the current nitrogen content of the plant and the optimal content of the plant on that day. Nitrogen stress, however, is not calculated only for legumes. Both nitrogen and phosphorus stress are 0.0 at optimal nutrient content and 1.0 when the nutrient content of the plant is 50% or less of the optimal value.
First a scaling factor is calculate that relates the current plant nutrient content to the optimal nutrient content:

$$scaling\_factor = 200(\frac{stored\_nutrient\_content}{optimal\_nutrient\_content} - 0.5); \qquad if\ optimal\_nutrient\_content \neq 0$$

[SC.CRP.52]

The nutrient stress for a given day is then calculated as:

$$nutrient\_stress = 1(\frac{scaling\_factor}{scaling\_factor + exp[3.535 - 0.02597 * scaling\_facotr]'}; \qquad if\ optimal\_nutrient\_content \neq 0$$
$$nutrient\_stress = 0; \qquad if\ optimal\_nutrient\_content = 0$$

[SC.CRP.53]

where stored_nutrient_content is the actual mass of nitrogen or phosphorus stored in plant material on the current day (kg N or P ha^{-1}) and optimal_nutrient_content is the optimal mass of nitrogen or phosphorus stored in plant material for the current growth stage (kg N or P ha^{-1}).

Grow Canopy

The amount of canopy cover is expressed as the leaf area index (LAI). LAI is defined as the area of green leaf per unit area of land. Each crop species has a maximum LAI and once that maximum is reached, the LAI will remain constant until leaf senescence begins to exceed leaf growth.
LAI is calculated using equations that relate the current fraction of accumulated heat units and crop-specific heat unit fractions associated with different stages of development. The methods for calculating the LAI are found in the LeafAreaIndex class.
First, two unitless shape coefficients, shape_1 and shape_2, are calculated by the method determine_-lai_shapes():

$$shape\_1 = \ln[\frac{fr_{PHU,1}}{fr_{LAI,1}} - fr_{PHU,1}] + shape\_2 * fr_{PHU,1}$$

[SC.CRP.54]

$$shape\_2 = \frac{[\ln(\frac{fr_{PHU,1}}{fr_{LAI,1}} - fr_{PHU,1}) - \ln(\frac{fr_{PHU,2}}{fr_{LAI,2}} - fr_{PHU,2})]}{fr\_PHU{,}2 - fr_{PHU,1}}$$

[SC.CRP.55]

where $fr_{PHU,1}$, $fr_{PHU,2}$, $fr_{LAI,1}$, and $fr_{LAI,2}$ are the crop-specific curve points. Specifically, $fr_{PHU,1}$ and $fr_{PHU,2}$ are the fractions of total potential heat units of the growing season corresponding to the 1st point and 2nd point on the optimal leaf area development curve and $fr_{LAI,1}$ and $fr_{LAI,2}$ are the fractions of the maximum plant LAI corresponding to the 1st and 2nd point on the optimal leaf area development curve.
Next, the optimal LAI fraction for a given day is calculated by the method determine_optimal_leaf_-area_fraction() as:

$$fr_{LAI,opt} = \frac{fr_{PHU}}{fr_{PHU}+exp(shape\_1-shape\_2*fr_{PHU})}$$

[SC.CRP.56]

where $fr_{LAI,opt}$ is the fraction of the plant's maximum potential LAI corresponding to a given fraction of potential heat units (PHU) the plant has accumulated on that day.
The optimal LAI fraction ($fr_{LAI,opt}$) is then used by the method determine_canopy_height() to set the canopy height for that day according to the following equation:

$$canopy\_height = min(canopy\_height_{max},\ canopy\_height_{max} * \sqrt{fr_{LAI,opt}})$$

[SC.CRP.57]

The optimal increase in LAI for annuals and perennials that are not in senescence is calculated by the method determine_max_leaf_area_change() using the following equation:

$$optimal\_leaf\_area\_index\_change = \\ (fr_{LAI,opt,d} - fr_{LAI,opt,d-1}) * LAI_{max} * (1 - exp(5 * (LAI_{act,d-1} - LAI_{max})))$$

[SC.CRP.58]

where LAImax is the crop-specific maximum LAI, $fr_{LAI,max,d}$ and $fr_{LAI,max,d-1}$are the fraction of the plant's maximum LAI accumulated on the given day d and the previous day respectively, and $LAI_{act,d-1}$LAIact,d-1 is the actual LAI on the previous day.
The actual LAI added is then calculated as:

$$actual\_leaf\_area\_index\_change = optimal\_leaf\_area\_index\_change * \sqrt{growth\_factor}$$

[SC.CRP.59]

where actual_leaf_area_index_change is the change in the LAI the given day and growth_factor is the plant's growth factor (0.0 - 1.0) calculated in **[SC.CRP.46]**.
The total LAI for the plant that day is then updated according to the following:

$$leaf\_area\_index = previous\_leaf\_area\_index + actual\_leaf\_area\_index\_change$$

If the plant is in a stage of senescence, the LAI for annuals is calculated as:

$$leaf\_area\_index = LAI_{max} * (\frac{1-fr_{PHU}}{1-fr_{PHU,sen}})$$

[SC.CRP.60]

where LAI_{max} is the maximum LAI, $fr_{PHU,sen}$ is the crop-specific fraction of PHU at which senescence becomes the dominant growth process, and fr_{PHU} is the fraction of PHU accumulated for the plant on a given day in the growing season.

Biomass Accumulation

Plant biomass is produced when intercepted solar radiation is converted into plant biomass via photosynthesis assuming a plant species-specific radiation use efficiency. The methods that estimate the conversion of solar radiation into plant biomass are collected in the BiomassAllocation class.
The amount of photosynthetically active radiation (MJ m^{-2}) that is intercepted by the leaf area is calculated by the method intercept_radiation using the following function:

$$intercepted\_radiation = 0.5 * radiation * (1 - exp(-extinction\_coef * LAI_{act}))$$

[SC.CRP.61]

where radiation is the incident total solar radiation (MJ m^{-2}), 0.5 X H_{day}, extinction_coef is the light extinction coefficient, and LAI_{act} is the leaf area index.
Using this value, the maximum potential biomass increase for that day is calculated as:

$$biomass\_growth\_max = light\_use\_efficiency * intercepted\_radiation$$

[SC.CRP.62]

where biomass_growth_max is the potential increase in total plant biomass on a given day (kg ha^{-1}), light_use_efficiency is the crop-specific radiation use efficiency (10^{-1} g MJ^{-1} or kg ha^{-1} x MJ m^{-2}), and intercepted_radiation is the amount of intercepted photosynthetically active radiation on a given day (MJ m^{-2}).
The actual increase in biomass (kg/ha) on a given day) is calculated as:

$$biomass\_growth = biomass\_growth\_max * growth\_factor$$

[SC.CRP.63]

Where growth_factor is the plant growth factor (0.0 - 1.0) calculated in **[SC.CRP.46]**.

Biomass Allocation

Once the total biomass growth for a given day has been calculated, the total accumulated biomass can be partitioned into above and below ground biomass depending on the development stage of the crop. The aboveground biomass (kg hha^{-1}) is calculated as:

$$aboveground\_biomass\_growth = (1 - root\_fraction) * biomass\_growth$$

[SC.CRP.64]

where root_fraction is the fraction of total biomass in the roots calculated in **[SC.CRP.25]** and *biomass_growth* is the actual plant biomass added that day (kg hha^{-1}).

$$belowground\_biomass\_growth = root\_fraction * biomass\_growth$$

[SC.CRP.65]

Crop Harvest

When the Field class initiates a harvest event via the manage_harvest() function of the Crop class, the CropManagement class method for harvesting is called which:

- Determines the harvest index for that crop which accounts for the type of feed and storage of that crop
- Cuts and collects the crop's harvested biomass according to the harvest index and harvest efficiency
- Kills the crop if it is an annual or the last harvest of a perennial crop
- Records the yield
- Transfers the residue to the soil

Harvest Index

If a custom harvest index is provided by the user (harvest index override), that value is used. Otherwise, the harvest index is calculated based on the crop's accumulated heat fraction and the crop-specific optimal harvest index. The method also adjusts the harvest index based on the crop's water deficiency. Harvest index is defined as the fraction of the above ground dry biomass of plant that is removed as dry yield. The potential harvest index (HI_{max}) for the plant for a given day is calculated as:

$$potential\_harvest\_index = optimal\_harvest\_index * \frac{100 * fr_{PHU}}{[100 * fr_{PHU} + exp(11.1 - 10 * fr_{PHU})]}$$

[SC.CRP.66]

where $optimal\_harvest\_index$ is the optimal harvest index for the plant at maturity, given ideal growing conditions and fr_{PHU} is the fraction of potential heat units accumulated for the plant on a given day in the growing season.
The actual harvest index is then adjusted for water deficiency:

$$actual\_harvest\_index = (potential\_harvest\_index - minimum\_harvest\_index) * \frac{water\_deficiency}{water\_deficiency + exp(6.13 - 0.883 * water\_deficiency)} + HI_{min}$$

[SC.CRP.67]

where potential_harvest_index is the potential harvest index, minimum_harvest_index is the harvest index for the plant in drought conditions and represents the minimum harvest index allowed for the plant, water_deficiency is the water deficiency factor.
As the water deficiency factor increases, the actual harvest index will decrease. The water deficiency factor is calculated in the WaterDynamics class during the cycle_water() method:

$$water\_deficiency = * \frac{\sum_{p}^{m} actual\_evapotranspiration}{\sum_{p}^{m} potential\_evapotranspiration}$$

[SC.CRP.68]

where p is the day of planting, m is the day of harvest if the plant is harvested before it reaches maturity or the last day of the growing season if the plant is harvested after it reaches maturity, actual_evaoptranspiration is the actual evapotranspiration on a given day (mm H_2O), and potential_evaoptranspiration is the potential evapotranspiration on a given day (mm H_2O).

Crop Cutting

After the harvest index has been set, the manage_harvest method calls the cup_crop() method that determines how much biomass is cut from the growing crop. The proportion of a crop that is cut is determined by the harvest index. A harvest index < 1 indicates that a proportion of aboveground biomass equal to the harvest index will be removed from the aboveground biomass of the plant. A harvest index > 1 will remove below ground biomass as well.
The cut biomass is removed from the plant's total biomass and the amount collected as yield is determined by the collected fraction. The remaining portion is left in the field as the crop residue created during harvest. Cut operations without a harvest, as in the case of cover cropping systems, are conducted by setting collected_fraction = 0 (the default).
If the harvest index is less than one, the cut biomass (kg ha^{-1}) is calculated as:

$$cut\_biomass = aboveground\_biomass * actual\_harvest\_index$$

[SC.CRP.69]

The actual harvested yield (kg DM ha^{-1}) and cut crop residue (kg DM ha^{-1}) left on the field are then calculated using the harvest efficiency which is defined as the fraction of cut biomass removed by the harvesting equipment.

$$\begin{gathered} dry\_yield\_collected = cut\_biomass * harvest\_efficiency \\ yield\_yield\_collected = cut\_biomass * (1 - harvest\_efficiency) \\ biomass_{update} = biomass_{previous} - cut\_biomass \end{gathered}$$

[SC.CRP.70]

After the total biomass and yields have been updated, the above and belowground biomass in the plant material that remains in the field are updated along with the plant's root fraction via the recalculate_-biomass_distribution() method.
If no roots are harvested:

$$\begin{gathered} aboveground\_biomass_{update} = aboveground\_biomass_{previous} - cut\_biomass \\ root\_fraction = \frac{root\_biomass}{biomass_{update}} \end{gathered}$$

If roots are harvested:

$$\begin{gathered} root\_biomass\_removed = cut\_biomass - aboveground\_biomass_{previous} \\ root\_biomass_{update} = root\_biomass_{previous} - root\_biomass\_removed \\ aboveground\_biomass_{update} = 0 \\ root\_fraction = 0 \end{gathered}$$

[SC.CRP.71]

In addition to total biomass, the nutrient content for both the collected and uncollected portions are updated. If the simulation is using the internally-derived harvest index for cutting, then nutrients are determined with the crop-specific yield nutrient fractions. Otherwise, (harvest index override), the optimal nutrient values are used.
The amount of nitrogen and phosphorus removed with the collected biomass, or the nitrogen and phosphorus yields (kg N or P ha^{-1}) are calculated as:

$$nitrogen\_yield = nitrogen\_fraction * dry\_yield\_collected$$
$$phosphorus\_yield = phosphorus\_fraction * dry\_yield\_collected$$

[SC.CRP.72]

where $nitrogen\_fraction$ is the fraction of nitrogen in the harvested biomass, and $phosphorus\_fraction$ fraction of phosphorus in the harvested biomass.
The cut_crop() method updates the crops leaf area index and accumulated heat units. This is especially important for perennial crops that will continue growing either the following day or after the end of their dormancy.
The plant's leaf area index and accumulated heat units are set back by the fraction of biomass removed in yield.

$$fraction\_cut = \frac{cut\_biomass}{aboveground\_biomass}$$

[SC.CRP.73]

$$leaf\_area\_index_{update} = leaf\_area\_index_{previous} * (1 - fraction\_cut)$$

[SC.CRP.74]

$$accumulated\_heat\_unit_{update} = accumulated\_heat\_unit_{previous} * (1 - fraction\_cut)$$

[SC.CRP.75]

Reducing the number of accumulated heat units shifts the plant's development to an earlier period in which growth is usually occurring at a faster rate.
Crop Kill

If the plant is killed at harvest, the plant biomass that was not removed by harvest and the nutrients in that biomass are added to the residue pools (kg DM, N or P ha^{-1}). So after the cut_crop() methods calculates the yield and the residue produced by the harvest practice, the kill() method, updates the status of the crop plant to indicate that it is no longer living and adds the remaining plant biomass and nutrients to the yield residue pools.

$$yield\_residue_{update} = yield\_residue_{previous} + biomass$$
$$yield\_nitrogen\_residue_{update} = yield\_residue_{update} * nitrogen\_fraction$$
$$yield\_phosphate\_residue_{update} = yield\_residue_{update} * phosphorus\_fraction$$

[SC.CRP.76]

Record Harvest and Transfer Residue

The last steps in the CropManagement method for managing the crop harvest is to record the harvested crop information in a data structure that can be passed to the OutputManager and the Feed Storage Module and then to add the crop residue to the soil residue pools. The latter is accomplished by the transfer_residue method.
If the crop is not killed and there is only surface residue and the soil pools in the surface layer are updated as follows:

$$
\begin{gathered}
soil\_plant\_residue = yield\_residue \\
fresh\_organic\_nitrogen\_content_{update} = \\
fresh\_organic\_nitrogen\_content_{previous} + yield\_nitrogen\_residue \\
labile\_inorganic\_phosphorus\_content_{update} = \\
labile\_inorganic\_phosphorus\_content_{previous} + yield\_phosphorus\_residue
\end{gathered}
$$

[SC.CRP.77]

If the crop is killed and there is both surface and belowground crop residue to add to the soil pools, the residue is distributed between the soil layers by the distribute_residue_nutrients() method.
First, the fraction of the residue that remains on the surface is calculated as:

$$surface\_residue\_fraction = \frac{yield\_residue - root\_biomass}{yield\_residue}$$

The surface residue, nitrogen residue, and phosphorus residues are then calculated by multiplying the total residue masses by the surface residue fractions and that mass is added to the soil pools as described above in **[SC.CRP.77]**.
Then the residue masses added to the subsurface soil layers calculated as the masses multiplied by (1-surface_residue_fraction). The subsurface residue masses are distributed to different soil layers by calculating the distribution of root biomass between the soil layers following a method described by **fan2016root**. The work by **fan2016root** propose a logistic dose response curve for the cumulative distribution of the root biomass up to a specific depth in the soil profile. We adapt the methods they proposed to estimate the proportion of biomass in each soil layer, rather than the cumulative proportion at a given soil depth.
Heuristically, the method is executed as follows:
Starting with the top layer:

1. Estimate the proportion of the root biomass distributed to the depth that is equal to the bottom of the soil layer using Equation 2 in **fan2016root** reproduced in **[SC.CRP.78]** below.
2. Multiply the proportion of the root biomass distributed up to the depth of the current soil layer from step 1 by remaining yield residue masses (total mass, nitrogen mass, and phosphorus mass).
3. Subtract the sum of the yield residue mass for all layers above the current layer from the total yield residue mass from the surface to the bottom of the soil layer to get the root biomass in that latter alone. (note, for the first sub-surface layer, this sum will be zero).

Repeat for each layer going down. The equation from **fan2016root** that determines the fraction of the root biomass up to a given depth is described below and broken up into 3 parts for clarity:

$$
\begin{gathered}
root\_biomass\_layer\_fraction = first\_term + (second\_term * third\_term) \\
first\_term = \frac{1}{1+(\frac{root\_depth}{root\_dist\_param\_da})^{root\_dist\_param\_c}} \\
second\_term = 1 - \frac{1}{1+(\frac{root\_depth}{root\_dist\_param\_da})^{root\_dist\_param\_c}} \\
third\_term = 1 - \frac{root\_depth}{root\_depth\_max}
\end{gathered}
$$

[SC.CRP.78]

This method relies on two crop-specific, empirically fitted parameters estimated by **fan2016root**: root_-dist_param_c and root_dist_param_c. The default values for these parameters are those reported in the original manuscript but they are available for adjustment as user inputs.

Part V

Feed Storage Module

Contents

List of Figures

Contents

List of Tables

Contents

24 Module Introduction

The Feed Module's primary purpose is to serve as an inventory of purchased and farm-grown feed. Feed is added via purchase (Animal Module) and from on-farm harvest (Crop and Soil Module). Feed is removed when it is fed out in the Animal Module. In addition, stored feed quantity and quality changes are simulated over the duration of storage depending on crop type, crop quality, user-specified storage conditions, and ambient climate conditions. The feed storage module currently accepts Corn (silage, dry/high-moisture grain), Alfalfa, Grass, Triticale/Cereal, and Soybean. Forage and/or feed quality can also be user-specified, offering flexibility and modularity of user needs. Feed component values, proportions, digestibility, and availability subject to change during storage are: mass, DM content, CP, NPN, nNDF, ADF, starch, and WSC. These changes must be tracked to accurately reflect the remaining feed and its qualities. Overall stored feed quality is assessed prior to consumption in the Animal Module.

24.a Feed Quality Assessment

colorbox This section allocates farm grown feeds to be used for single or multiple animal classes and is illustrated in Figure 1. Priority is to reserve high-quality forage for lactating cows. Feeds are generally categorized into grains or forages. If the feed is a grain then it is available to all animal classes. The model assumes that grains can be purchased to supplement farm grown grain inventory.
If the feed is a forage, then some farms will assess the quality of the forage and preserve the high quality forage for the lactating cows. If they do not differentiate their forages based on quality, then the forage is available to all animals.

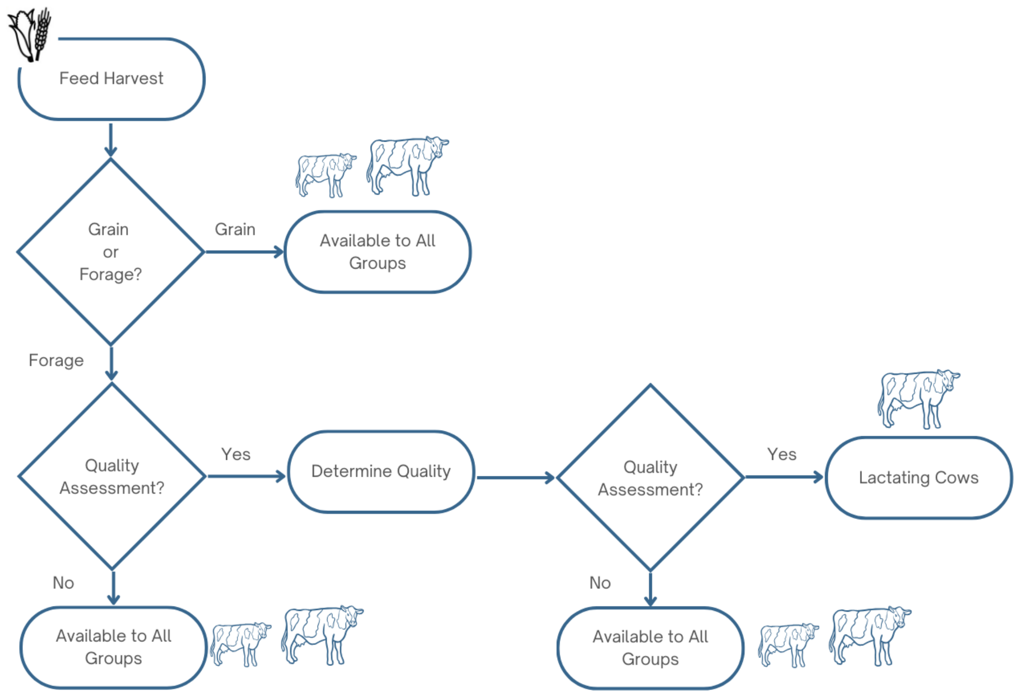


Figure 10: Farm grown feed allocation to cattle groups

We currently represent the quality assessment for the feeds based on a single differentiating nutrient composition value. The differentiating nutrient and levels of the nutrient are given in the feed library.
The feed library that includes both NRC and NASEM feeds does not have feed quality embedded in the

feed IDs. Instead, a quality assessment should be made independent of the feed ID number and added as an additional attribute of the stored feed.

25 Feed Storage Inputs

Farmgrown feeds have storage inputs that are described in depth below and consist of the following:

- storage classification The class of storage: Grain, Hay, Silage, Baleage.
- name Each storage has a unique name. Ex. "alfalfa_silage_storage_1"
- crop_name The name of the crop being stored.
- field_name The name of the field the crop was grown in.
- storage_type The sub-type of the storage class. Ex. "Dry" for grain storage
- dm_loss_coefficient An input used for grain storages that specifies a loss of DM. Ex. "0.01",
- initial_storage_dry_matter The dry matter content upon receipt by the feed storage module.
- post_wilting_moisture_percentage The DM content of silage or baleage after wilting.
- target_dry_matter The final DM content of hay.
- additional_dry_matter_loss_coefficient An optional input to represent DM loss in excess of normal processes.
- bale_size An input used for baleage to specify the diameter of the bale in meters.
- rufas_id The RuFaS feed ID of the stored crop.
- capacity 1e10 The maximum mass capacity of the storage.

26 Storage Classifications

The agricultural production of harvested animal feed is dependent on season, climate, and farm management. In most cases, this means that feed is produced in large quantities when growing conditions are suitable and must be preserved to meet daily animal feed requirements. Microbial spoilage of wet feed products rapidly degrades their nutritional quality and can harm livestock that consume spoiled feed. Viable, cost-effective methods of preservation depend on the crop type and climate, but practically are limited to drying feed beyond the ability to support microbial growth and fermenting feed in the absence of oxygen to produce acid and reduce pH to inhibit spoilage. Feed Storage functions are primarily based on those in the Integrated Farm Systems Model but modified to accommodate a daily time-step and remove explicit spatial components.

26.a Grain

Grains (e.g., corn, rice, soybeans, and wheat) under 15% moisture are considered dry grain. Higher moisture grains are considered high-moisture. Field harvested grains are not always sufficiently dry for safe storage (< 15% moisture) and supplemental drying is often used to achieve target moisture content. Air-drying is a lower-energy drying method for drying grain to ambient levels that consists of passing ambient air through wet grains via fan. Heated-air drying is higher-energy and faster than air-drying, but can reduce moisture below ambient levels. In practice, DM loss from grains are small and are modeled as a fixed percentage loss of dry matter; 1% for dry grain, 5% for high-moisture grain.

$$L\_\{gaseous\} = M\_i \; cdot0.01 \; tag$$

[FS.GRN.1]

$$L\_\{gaseous\} = M\_i\ cdot 0.05\ tag$$

[FS.GRN.2]

High-moisture grain as a feed must be managed (fed, fermented, or dried) quickly for this to be accurate, but that is a well-understood requirement and is not explicitly modeled. Nutritional quality changes are variable, but proportional to DM loss. Nutrient values of finished grain not currently tracked are referenced from the most recent edition of NASEM for use in the Animal Module.

26.b Hay

Hay is forage that has been dried with a safe preservation moisture content of 12-15%. Bale storage conditions are separated into protected and unprotected storage with a total of four storage categories in order of decreasing protection from the elements: protected-indoor, protected-wrapped, protected-tarped, and unprotected-outdoor. Less protected hay storages experience additional losses as compared to the most protected storage (indoor) due to exposure to increased airflow and weather according to the following equation. Baled hay loses DM and quality as a function of storage conditions and its moisture content. Bale density plays a role, but is too complex to model directly and is instead estimated as a function of moisture content.

$$L\_\{gaseous\} = L\_\{base\} + L\_\{additional\}\ tag$$

[FS.HAY.1]

Where we know that:
L_{base} (kg) is the minimum loss expected from protected-indoor as the sum of DM lost in the first 30 days of storage (L_{first 30 days}, kg) and that lost after 30 days (L_{post 30 days}, kg) in order to capture the differing rates of DM loss before and after the forage has reached its target DM content.

$$L\_\{base\} = L\_\{first\ 30\ days\} + L\_\{post\ 30\ days\}\ tag$$

[FS.HAY.2]

Where we know that:

$$L\_\{first\ 30\ days\} = \frac{Q + 2433 \cdot [moisture - \frac{MF(1-moisture)}{1-MF}]}{(1-moisture)(14206 - \frac{2433 \cdot MF}{1-MF})} \cdot \frac{min(30,d)}{30}\ tag$$

[FS.HAY.3]

$$Q = 104 \cdot moisture^{2.18} \cdot {b_{density}}^{0.5} + 5.72(moisture^{1.23} \cdot {b_{density}}^{0.94})\ tag$$

Q is the sensible heat generated in the hay (kJ/kg) based on bale density

[FS.HAY.4]

$$b\_\{density\} = 100 + 440 \cdot moisture\ tag$$

[FS.HAY.5]

moisture is the fraction of water mass at time of storage (unitless), MF is the target final moisture percent of hay (percent) as defined by the user (default: 12), and d is the number of days the hay has been stored for.

And

$$L\_\{post\ 30\ days\} = 0.0001 \cdot max(0, d - 30)\ tag$$

[FS.HAY.6]

$$L\_\{additional\} = \sum_{i=1}^{d} l \cdot rain_i \cdot max(0.0, \frac{high_i + low_i}{2}) \cdot \frac{1}{b_{density}} \cdot {b_{size}}^3\ tag$$

[FS.HAY.7]

- d is the number of days the hay has been stored for,
- i is the day of storage that dry matter loss is being calculated for,
- l is the loss coefficient (unitless) from Table 1; **[FS.HAY.7]**,
- rain_i is the amount of rain on day i (cm),
- high_i is the high temperature on day i ($\circ C$),
- low_i is the low temperature on day ($\circ C$),
- b_density is the bale density (kg/m^2) **[FS.HAY.5]**, and
- b_size is the diameter of the hay bale (m)

Table 92: Hay storage types and associated fractional loss coefficients

Hay Storage Type	Fractional Loss Coefficient (i)
Protected Wrapped	0.0000216
Protected Tarped	0.0000108
Unprotected Outdoor	0.00006
Protected Indoor	0

Cumulative water lost from forages stored as hay is determined by:

$$M = F_i \cdot \frac{max(0.0, moisture - MF)}{100} \cdot \frac{min(30, d)}{30}\ tag$$

[FS.HAY.8]

The lowest energy method for hay drying is passive drying in the field when conditions are suitable, but, similarly to grains, hay can receive supplemental drying via ambient or heated air. DM loss from protected, baled hay is preferentially from starch and water soluble carbohydrates (60%), and CP (40%) which, over time, leads to a proportional increase in NDF and a minor increase in CP over time. Unprotected bales experience increased proportional CP loss (50% of protected bale DM loss) due to leaching and an additional 17% loss of NDF dry matter compared to protected bales. Fractional loss coefficients for hay are defined by Table 2:

Table 93: Protected and unprotected storage, nutrients, and associated fractional loss coefficients

Hay Storage Type	**Nutrient**	**Fractional Loss Coefficient**
	Acid Detergent Fiber	0.0
Unprotected Hay	Neutral Detergent Fiber	0.17
	Crude Protein	0.4
	Acid Detergent Fiber	0.0
Protected or Indoor Hay	Neutral Detergent Fiber	0.0
	Crude Protein	0.0

26.c Silage

Silage is the product of anaerobic acidification of non-dry forage, a process called ensiling, and has traditionally been produced as a method of animal feed preservation viable in wet, cool climates where forage production cannot sustain livestock year-round and drying hay at the required scale is challenging (Pahlow et al., 2003). Typically, forage fermentation is the product of lactic acid bacteria (LAB) consuming WSC present in harvested forage and metabolizing it to a mix of organic products, including the desirable lactic and acetic acids, as well as CO_2 and ethanol. A well-preserved silage relies on a rapid fermentation for retention of optimal feed quality because insufficient or slow acidification is associated with prolonged proteolytic activity, additional dry matter loss, and increased risk of spoilage. (Pahlow et al., 2003; Muck, 2013).
Fermented forages experience DM loss and nutrient changes post-harvest, during fermentation, and during storage. Wilting losses are not currently modeled.
Alfalfa fermentation daily dry matter loss (fraction) is calculated with the equation:

$$L\_fermentation = \sum_{i=1}^{d} drymass_{i-1} \cdot (0.00052 - 0.001213(dryfrac_{i-1} - 0.2)) \; tag$$

[FS.SIL.1]

- d is the number of days the alfalfa has been ensiled for,
- i is the day since ensiling that dry matter loss is being calculated for,
- drymass_i-1 is the dry matter mass of ensiled alfalfa on day i-1,
- dryfrac_i-1 is the fraction of ensiled alfalfa fresh mass that is dry matter on day i-1.
- When i is 1, drymass_i-1 is the initial amount of ensiled alfalfa dry mass and

- dryfrac_i-1 is the initial dry matter fraction of the ensiled alfalfa. This equation is only appropriate for use if dryfrac_i-1 is in the range [0.2, 0.6] and the average temperature on day
- i is in the range [5, 45] ($\circ C$).

Corn, grass, and small grain daily dry matter loss (fraction) is calculated with the equation:

$$L_{fermentation} = \sum_{i=1}^{d} drymass_{i-1} \cdot (0.000288 - 0.000643(dryfrac_{i-1} - 0.15))\ tag$$

[FS.SIL.2]

- d is the number of days the forage has been ensiled for,
- i is the day since ensiling that dry matter loss is being calculated for,
- drymass_i-1 is the dry matter mass of ensiled forage on day i-1,
- dryfrac_i-1 is the fraction of ensiled forage fresh mass that is dry matter on day i-1.
- When i is 1, drymass_i-1 is the initial amount of ensiled forage dry mass and
- dryfrac_i-1 is the initial dry matter fraction of the ensiled forage. This equation is only appropriate for use if dryfrac_i-1 is in the range [0.15, 0.6] and the average temperature on day i is in the range [0, 40] ($\circ C$).

Table 94: Silage and the fractional loss coefficients

Nutrient	**Fractional Loss Coefficient**
Acid Detergent Fiber	0.0
Neutral Detergent Fiber	0.0
Crude Protein	0.0

Ensiled crops may also lose dry matter through effluent efflux. Crops that are ensiled with a dry matter content less than 30% lose both water and dry matter according to the following equations:

$$L_{effluent} = dryfrac\_\{effluent\} \cdot M\_\{effluent\} \cdot 0.1 \cdot max(10, d)\ tag$$

[FS.SIL.4]

Where we know that:

$dryfrac_{effluent}$ is the fraction of dry matter in the effluent that is lost, $M_{effluent}$ is the estimated maximum effluent (kg), and d is the number of days the crop has been ensiled for.

$$ML\_\{effluent\} = (1 - dryfrac_{effluent}) \cdot M_{effluent} \cdot 0.1 \cdot max(10, d)\ tag$$

[FS.SIL.5]

Where we know that: dryfrac_effluent is the fraction of dry matter in the effluent that is lost,

M_effluent is the estimated maximum effluent (kg), and d is the number of days the crop has been ensiled for. The fraction of dry matter in effluent is a constant defined as:

$$dryfrac_{effluent} = 0.1035 \; tag$$

[FS.SIL.6]

$$M_{effluent} = mass_{fresh} \cdot ((1 - dryfrac) - 0.7) \; tag$$

[FS.SIL.7]

Where we know that:
$mass_{fresh}$ is the fresh mass of the crop when it is ensiled (kg) and $dryfrac$ is the fraction of the crop's fresh mass which is dry matter when it is ensiled.

Dry matter lost in effluent is preferentially from CP, water soluble carbohydrates, and non-protein nitrogen each of which is recalculated by the following equations:

Percent crude protein is calculated as:

$$CP_{updated} = \frac{CP_{initial} \cdot 0.01 - 0.3 \cdot DL_{fraction}}{1 - DL_{fraction}} \cdot 100 \; tag$$

[FS.SIL.8]

Where we know that:
$CP_{initial}$ is the percentage of dry matter mass that is crude protein in the ensiled crop before accounting for dry matter loss to effluent and $DL_{fraction}$ is the fraction of dry matter mass lost to effluent. The percentage of crude protein is lower-bounded at 0.

The percentage of NPN is calculated with the equation:

$$NPN_{updated} = \frac{(NPN_{initial} \cdot 0.01) \cdot (CP_{initial} \cdot 0.01) - 0.3 \cdot DL_{fraction}}{(CP_{initial} \cdot 0.01) - DL_{fraction}} \cdot 100 \; tag$$

[FS.SIL.9]

Where we know that:
$NPN_{initial}$ is percentage of dry matter mass that is non-protein nitrogen in the ensiled crop before accounting for dry matter loss to effluent, $CP_{initial}$ is the percentage of dry matter mass that is crude protein before accounting for dry matter loss to effluent, and $DL_{fraction}$ is the fraction of dry matter mass lost to effluent. The percentage of non-protein nitrogen is lower-bounded at 0.

$DL_{fraction}$ is calculated with the equation:

$$DL_{fraction} = \frac{DM_{lost}}{DM_{initial}} \; tag$$

[FS.SIL.10]

Where we know that:
DM_lost is the amount of dry matter mass in the ensiled crop after accounting for dry matter lost to effluent, and DM_initial is the amount of dry matter mass in the ensiled crop before accounting for dry matter lost to effluent.

26.d Baleage

Baleage is an ensiled forage with a DM content between 50-65% that is baled and wrapped to allow ensiling. The higher DM content of baleage lowers the amount of acid needed to achieve effective preservation and allows it to be consolidated into a bale. Fermentation DM losses for baleage are calculated by **[FS.SIL.1]** and **[FS.SIL.2]** with fractional nutrient loss coefficients from Table 3. Baleage does not experience effluent losses.

26.e Nutrient Composition

Changes in dry matter and moisture of stored feed often alter their total and fractional nutrient composition. Recalculating nutrient fractions of stored feeds uses the following equation:

$$n_{updated} = \frac{max(0, \frac{n_{initial}}{100} - C) \cdot DL_{fraction}}{1 - DL_{fraction}} \cdot 100 \; tag$$

[FS.NUT.1]

Where we know that:
n_updated is the percentage of the target nutrient in the stored crop's dry matter mass after accounting for dry matter loss, n_initial is the percentage of the target nutrient in the stored crop's dry matter mass before accounting for dry matter loss, C is the fractional loss coefficient specific to the nutrient and storage type, and DL_fraction is the fraction of dry matter lost.

$$DL_{fraction} = \frac{DM_{updated}}{DM_{initial}} \; tag$$

[FS.NUT.2]

27 References

Feed Storage Module

Muck, R. E. (2013). “Recent advances in silage microbiology”. In: *Agricultural and Food Science* 22.1, pp. 3–15. DOI: 10.23986/afsci.6718.

Pahlow, G. et al. (2003). “Microbiology of ensiling”. In: *Silage Science and Technology*. Ed. by D. R. Buxton, R. E. Muck, and J. H. Harrison. Madison: American Society of Agronomy, Inc., pp. 31–93.

Part VI
Abbreviations

Table 95: Common abbreviations used throughout this document

Abbreviation	Full Term or Phrase
ADF	Acid detergent fiber (% of DM)
ADG	Average daily weight gain (g/d)
Age1stBred	First breeding age (month)
BCS5	Body condition score (1–5 basis)
BW	Body weight (kg)
Ca	Calcium content (% of DM)
CBW	Calf birth weight (kg)
CI	Calving interval (d)
cm	centimeter
Cost	Feed cost ($/kg of DM)
CP	Crude protein (% of DM)
d	days
DE	Digestible energy (standard value, Mcal/kg)
DIM	Days in milk
DM	Dry matter (% of as-fed basis)
DMI	Dry matter intake
DOP	Days of pregnancy (d)
dRUP	RUP degradability (% of RUP)
EE	Ether extract, in crude fat (% DM)
EndCP	Endogenous CP
EndN	Endogenous N
EQSBW	Equivalent shrunk body weight (kg)
FA	Fatty acids
FIPS	Federal Information Processing Standards
g	grams
GE	Gross energy (standard value, Mcal/kg)
GrUterW	Gravid uterine weight (kg)
Kd	Rumen protein degradation rate (%/h)
kg	kilograms
kJ	kilojoule
L	liters
m	meter
max	maximum
Mcal	Megacalorie

Continued on next page

Table continued from previous page

Abbreviation	**Full Term or Phrase**
MCP	Microbial Crude Protein
ME	Metabolizable energy (standard value, Mcal/kg)
MICP	Microbial CP
min	minimum
mL	milliliters
MN	Microbial N
MP	Metabolizable protein
MTP	Microbial TP
MW	Mature body weight (kg)
MY	Milk yield
N	Nitrogen (% of DM)
NASEM	National Academies of Sciences, Engineering, and Medicine
NDF	Neutral detergent fiber (% of DM)
NEG	Net energy for growth (standard value, Mcal/kg)
NEL	Net energy for lactation (standard value, Mcal/kg)
NEM	Net energy for maintenance (standard value, Mcal/kg)
nNDF	Nondigestible Neutral Detergent Fiber
NPN	Non-protein Nitrogen
NRC	National Research Council (updated in 2021 to NASEM)
OM	Organic matter (% of DM)
P	Phosphorus content (% of DM)
PAF	Processing adjustment factor (not used)
Parity	Number of lactations
PrevTemp	Average daily temperature of last month (°C)
RDP	Ruminally degraded CP
ROM	Residual OM (% of DM)
RUP	Ruminally undegraded CP
s	seconds
TAN	Total ammoniacal N
TDN	Total digestible nutrient (% of DM)
TP	True protein (% of DM)
USDA	United States Department of Agriculture
UterW	Uterine weight (kg)
WSC	Water Soluble Carbohydrates