FarmDyn

A highly detailed template model for dynamic optimization of farms

19.04.2018

The dynamic single farm model FARMDYN documented in here presents a framework which allows simulating in farm management and investment under changes in boundary conditions such as prices or policy instruments, for a wide range of different farming systems found in Germany and beyond. Given the complex interplay of management and investment decisions - such as adjustments of herd sizes, crop shares and yields, feeding practise, fertilizer management and manure treatment -- FARMDYN is implemented as a fully dynamic bio-economic simulation model template building on Mixed-Integer Programming. It is complemented by a Graphical User Interface to steer simulations and to exploit results.

The framework is the outcome of several research activities. Its first version (named DAIRYDYN) was developed in the context of a research project financed by the German Science Foundation focusing in marginal abatement costs of dairy farms in. That project contributed the overall concept and the highly detailed description of dairy farming and GHG accounting, while it had only a rudimentary module for arable cropping. That version of the model was used by Garbert (2013) as the starting point to develop a version for pig farms, however with far less detail with regard to feeding options compared to cattle. Garbert also developed a first phosphate accounting module. Activities in spring 2013 for a scientific paper (Remble et al. 2013) contributed a first version with arable crops differentiated by intensity level and tillage type, along with more detailed machinery module which also considered plot size and mechanisation level effect on costs and labour needs. Based on nitrogen response functions, nitrogen loss factors were differentiated for the different intensity and related yield levels. After these extensions the model was renamed to FARMDYN (farm dynamic). David Schäfer, then a master student, developed in 2014 a bio-gas module for the model which reflects the German renewable energy legislation. In 2016, an extension allowing for stochastic programming was added.

Table of Contents

# List of abbreviations

|  |  |
| --- | --- |
| Abbreviation |  |
| CH4 | Methane |
| CO2 | Carbon dioxide |
| CVaR | Conditional Value at Risk |
| EEG | German Renewable Energy Acts |
| GAMS | General Algebraic Modelling System |
| GHG | Greenhouse Gas |
| GUI | Graphical User Interface |
| LHS | Latin Hypercube Sampling |
| MAC | Marginal Abatement Costs |
| MOTAD | Minimization of Total Absolute Deviations |
| MRP | Mean Reverting Processes |
| KWK | Kraft-Wärme-Kopplung |
| N | Nitrogen |
| N2O | Nitrous Oxide |
| NaWaRo | Renewable Energies |
| N2 | Elemental Nitrogen |
| NH3 | Ammonia |
| Norg | Organic Nitrogen |
| NP | Non-deterministic Polynomial-time |
| NOx | Nitrogen Oxides |
| NPV | Net Present Value |
| NTAN | Total Ammonia Nitrogen |
| P | Phosphate |
| RMIP | Relaxed Mixed Integer Programming |
| RHS | Right Hand Side |
| SON | State of Nature |
| SP | Stochastic Programming |
| VaR | Value at Risk |

# FarmDyn - A highly detailed template model for dynamic optimisation of farms



Universität Bonn Institute for Food and Resource Economics [Economic Modeling of Agricultural Systems Group](http://www.ilr.uni-bonn.de/em/em_e.htm)

The dynamic single farm model FarmDyn is the outcome of several, partially on-going research activities. It provides a modular and extendable template model to simulate in detail economically optimal production and investment decisions at single farm scale. FarmDyn depicts various farm branches (arable cropping, pig fattening, piglet production, dairy, beef fattening and biogas plants). Its default layout maximises the deterministic net present value (NPV) over a longer simulation horizon; alternatively, short-run, comparative static or stochastic layouts are available. In the latter case, all variables are state contingent and different types of risk behaviour can be modelled. Integer variables depict indivisibilities in labour use and investment decisions. Constraints reflect in rich detail (1) the inventory of and requirements for machines, stables and other structures, (2) demographic relations between different herds, (3) labour and feed requirements and nutrient flows as well as (4) the financial sphere of the farm, with a temporal resolution between two weeks and a year. The constraints can depict various environmental standards linked to detailed environmental accounting for nitrogen, phosphate and greenhouse-gases (GHG). A state-of-the-art software implementation based on GAMS in combination with MIP industry solvers and a graphical user interface (GUI) allows for efficient analysis. FarmDyn consists of several interacting modules (Figure 1).

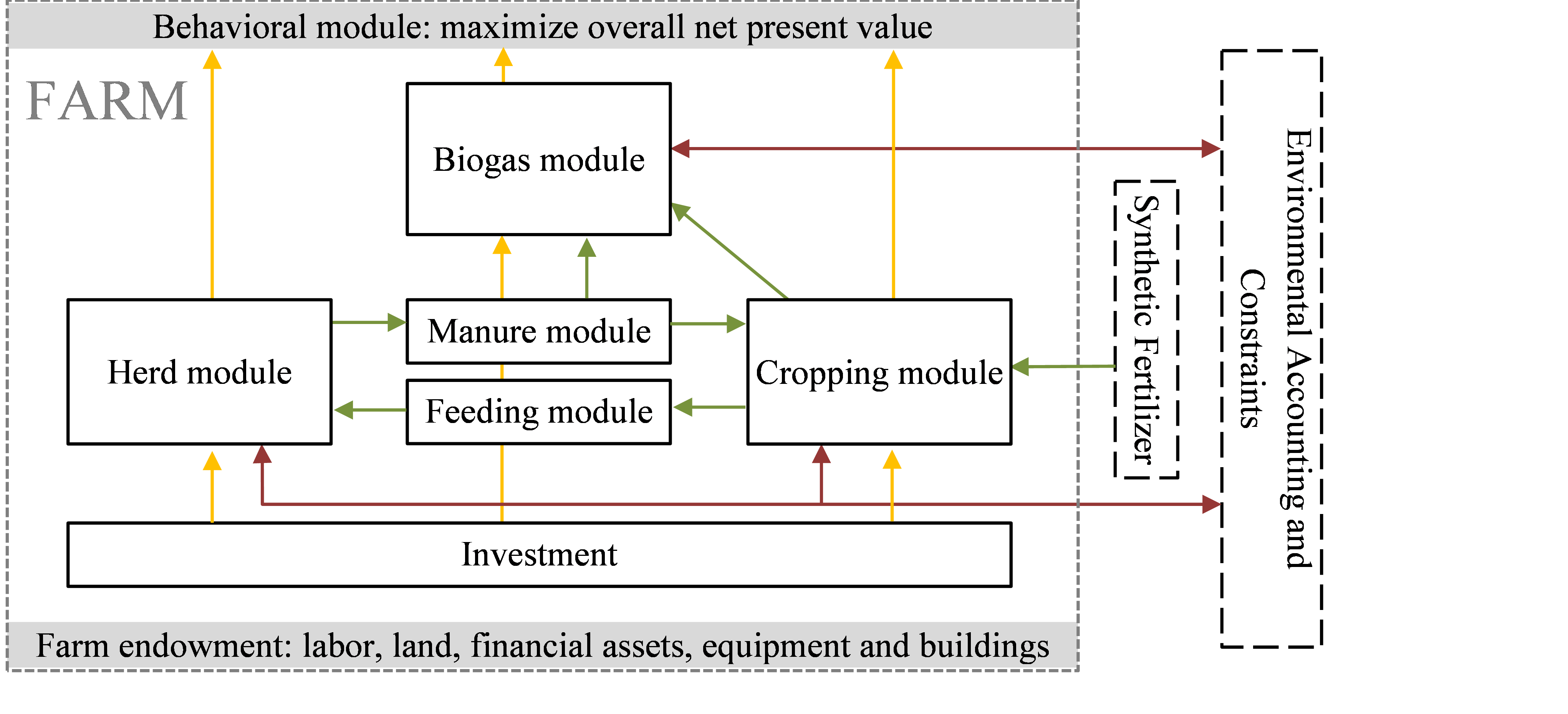


Figure 1. Overview of template model.

The **herd module** captures the intra-temporal demographic relations between different herds (number of animals born, replacement rates, raising periods etc.), at a maximal intra-yearly resolution of single months. The temporal resolution can be increased by aggregation on demand to reduce model size. Herds can be differentiated by animal types - such as cow, heifer, calf -, breeds, and feeding regimes. Cattle animal types can be broken down in different fattening phases. The pig module distinguishes between fattening- and piglet production systems. Fattening pigs are subdivided into different phases to account for different feeding requirements and excretion values. The piglet production system differentiates between sows, young piglets and piglets, which are separated from their mother after two weeks.

The **feed module** distinguishes between pig and cattle feeding requirements. For the different cattle, it captures a cost minimal feed mix from own produced fodder and different types of concentrates at given requirements per head and intra-year feeding periods (energy, protein, dry matter). For pigs it determines a cost minimal feed mix from own produced and purchased fodder and concentrates such as soybean meal and soy oil. For both branches, different feeding phases for reduced nitrogen and phosphorus output can be used.

The **cropping module** optimises the cropping pattern subject to land availability, reflecting yields, prices, machinery and fertilising needs and other variable costs for a selectable list of arable crops. The crops can be differentiated by production system (conventional, organic), tillage type (plough, minimal tillage, no tillage) and intensity level (normal and reduced fertilisation in 20% steps). Machinery use is linked to field working days requirements depicted with a bi-weekly resolution during the relevant months. Crop rotational constraints can be either depicted by introducing crop rotations or by simple maximal shares. The model can capture plots which are differentiated by soil, size and land type (gras, arable).

The **labour module** (not shown in Figure 1) optimises work use on- and off-farm with a monthly resolution, depicting in detail labour needs for different farm operations, herds and stables as well as management requirements for each farm branch and the farm as a whole. Off farm work distinguishes between half and full time work (binaries) and working flexibly for a low wage rate.

The **investment module** depicts investment decisions in machinery, stables and structures (silos, biogas plants, storage) as binary variables with a yearly resolution. Physical depreciation can be based on lifetime or use. Machinery use can be alternatively depicted as continuous re-investment rendering investment costs variable, based on a Euro per ha threshold. Investment can be financed out of (accumulated) cash flow or by credits of different length and related interest rates. For stables and biogas plants, maintenance investment is reflected as well.

Manure excretion from animals is calculated in the **manure module** based on fixed factors, differentiated by animal type, yield level and feeding practice. For biogas production, the composition of different feed stock is taken into account. Manure can be stored subfloor in stables and in different types of silos. Application of manure has to follow legal obligations and interacts with plant nutrient need from the cropping module. Different N losses are accounted for in stable, storage and during application, differentiating by spreading technology (broadspread, trailing hose etc).

The **environmental accounting module** allows quantifying gaseous emissions of Ammonia (NH3), nitrous oxide (N2O), nitrogen oxides (NOx), methane (CH4) and carbon dioxide (CO2). Nitrogen losses in the form of elemental nitrogen (N2) are not considered as emissions but are still a loss from the specified system and therefore part of the environmental accounting. For nitrogen (N) and phosphate (P), soil surface balances are calculated indicating potential nitrate leaching and phosphate losses. Environmental impacts are related to relevant farming operation. Furthermore, emissions are summarized in impact categories using characterization factors from RECIPE (2016).

The **biogas module** defines the economic and technological relations between components of a biogas plant with a monthly resolution, as well as links to the farm. Thereby, it includes the statutory payment structure and their respective restrictions according to the German Renewable Energy Acts (EEGs) from 2004 up to 2014. The biogas module differentiates between three different sizes of biogas plants and accounts for three different lifespans of investments connected to the biogas plant. Data for the technological and economic parameters used in the model are derived from KTBL (2013) and FNR (2013). The equations within the template model related to the biogas module are presented in the following section.

# Introduction

An economic template model uses a declarative approach which depicts in rather generic terms the physical and financial relations in the system to analyse. It describes their relations based on a set of decision variables, exogenous parameters and equations. Template models in that sense have a long-standing tradition in economics. In macro-applications, template based computable general equilibrium models, such as GTAP (Hertel 1997) or the IFPRI [[1]](#footnote-27) CGE\_template (Lofgren et al. 2002), are quite common. For regional and farm type applications, programming model templates are underlying e.g. the regional or farm type model in CAPRI (Britz & Witzke 2008) or the bio-economic typical farm type models in FFSIM (Louhichi et al. 2010). The aim of a template model is to differentiate between structural elements which are common to any instance of the system analysed and attributes of a specific instance. A specific instance of a farm would capture those attributes which are specific to e.g. location, firm and time point or period analysed, including attributes of the farmer (and its family) such as his management abilities and preferences.

A template model can be coded and documented independently from a specific instance. It also features clearly defined inputs and outputs so that generic interfaces to other modules can be developed. These modules could e.g. deliver the necessary inputs to generate instances or to use the template model's results as inputs, e.g. for reporting purposes or systematic analysis.

For our purposes, a suitable template must be able to generate instances representing farm characteristics by differing initial conditions and further attributes, specific to the firm and farmer. Initial conditions are for example the existing herds, available family labour, capital stock, such as stables, machinery or storage facilities and its age, land owned and rented by the farm or his equity. Further attributes could describe the firm's market environment such as input and output prices, yield potentials, household expenditures, the willingness of the farmer and family members to work off-farm and the potential farm branches.

Farming is characterised by long lasting and relatively expensive stationary capital stock, especially in form of stables and related equipment. High sunk costs related to past investments can lead to sticky farm programs, as key management possibilities such as reducing the herd size lead to modest saving of variable costs compared to losses in revenues. Consequently, strategies of farms as a response to changes in market and policy environment such as GHG emission ceilings are path dependent on investment decisions in the past. Whereas all farms can implement certain short term adjustments regarding herd-, feed- or fertiliser-management, investment based strategies are not very likely to be adjusted for farms which invested recently in new buildings or expensive machinery. These characteristics imply for individual farms, and for the industry as a whole that optimal short and long term strategies might differ considerably.

Accordingly, a framework is needed which covers a longer planning period to capture (re)investment decisions and their impact on the farm program and on externalities such as nutrient surpluses or GHG emissions. Figure 1 depicts the basic structure of the template model with different module interactions.

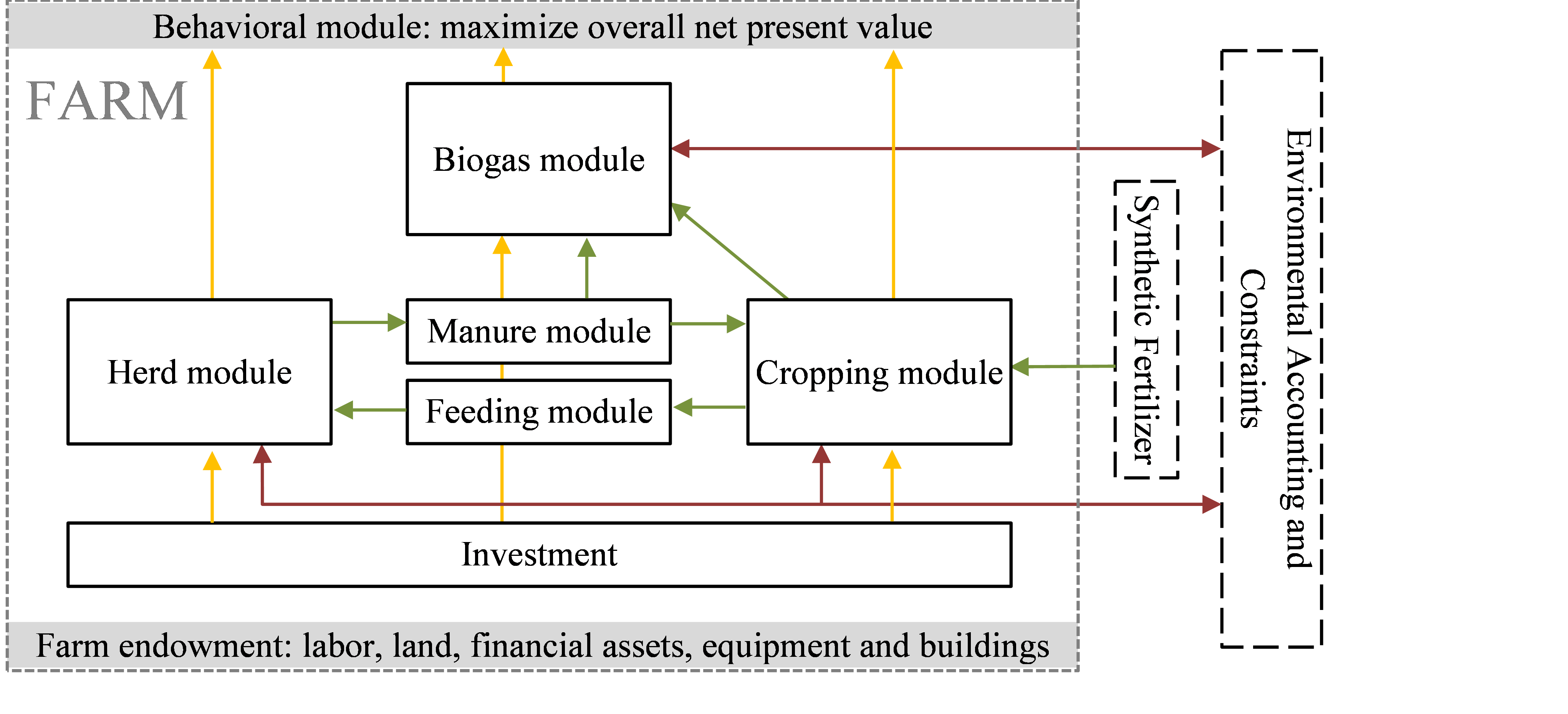


Figure 1: Overview of template model

In the following, the GAMS code is directly used to document the equations in the different modules to avoid a second layer of mnemonics. The following naming conventions are used in the GAMS code and also in the documentation. All decision variables of the farmers start with a *v\_*. They are endogenous to the simultaneous solution of all equations when maximising the objective function and hence depend on each other. Exogenous parameters start with a *p\_*. They can typically be changed in an experiment. Sets, i.e. collection of index elements, do not carry a specific prefix.

The model equations are defined in *model\templ.gms*, declarations of parameters and sets also used outside of the model equations can be found in *model\templ\_decl.gms*.

# Herd Module

Animals are dealt with in three parts of the model: the general herd module, the cattle module and the pig module. The general herd module depicts the herd demography while the latter two add aspects specific to cattle and pigs.

## General Herd Module

!!! abstract The herd module captures the intra-temporal demographic relations between different herds (number of animals born, replacement rates, raising periods etc.), at a maximal intra-yearly resolution of single months. The temporal resolution can be increased by aggregation on demand to reduce model size.

The general herd module depicts relations between herds of different animal types on farm. Specifically, herds are differentiated by age, gender, breeds, production objectives, month in each year.

The model uses two different variables to describe herds: *v\_herdStart* describes the number of animals by type which enter a production process at a certain time, while *v\_herdSize* describes the number of animals by type at the farm at a specific time. More precisely the standing herd, *v\_herdSize*, can be described as animals which joint the herd since the beginning of the production process, *v\_herdStart,* minus sold and slaughtered ones, as can be seen in the following equation. The parameter *p\_mDist* in the equation describes the difference in months between two time points defined by year, *t, t1*, and month, *m, m1*, *p\_prodLength* depicts the length of the production process in months.

herdSize\_(herds,breeds,tCur(t),nCur,m) $ (sum(FeedRegime,actHerds(herds,breeds,feedRegime,t,m))  
  
 $ sum( (t\_n(t1,nCur1),feedRegime,m1)  
 $ ( ( (-p\_mDist(t,m,t1,m1) le (p\_prodLength(herds,breeds)-1)  
 $ (p\_mDist(t,m,t1,m1) le 0))  
  
 or  
 ( (abs(p\_mDist(t,m,t1,m1)-12) le (p\_prodLength(herds,breeds)-1))  
 $ (p\_mDist(t,m,t1,m1)-12 le 0)) $ p\_compStatHerd  
 )  
 $ actHerds(herds,breeds,feedRegime,t1,m1)  
 $ (balherds(herds) or remonte(herds) or sameas("remonte",herds))  
 $ t\_n(t,nCur) $ isNodeBefore(nCur,nCur1)),1)  
 ) ..  
  
 sum(feedRegime $ actHerds(herds,breeds,feedRegime,t,m),v\_herdSize(herds,breeds,feedRegime,t,nCur,m))  
  
 =E=  
\*  
\* --- herds which started in the months before the production length, in case for piglets a separate construct is used  
\*  
  
 + sum( (t\_n(t1,nCur1),m1) $ (  
 (( (-p\_mDist(t,m,t1,m1) le (p\_prodLength(herds,breeds)-1))  
 $ (p\_mDist(t,m,t1,m1) le 0))  
  
 or  
 ( (abs(p\_mDist(t,m,t1,m1)-12) le (p\_prodLength(herds,breeds)-1))  
 $ (p\_mDist(t,m,t1,m1)-12 le 0)) $ p\_compStatHerd  
 )  
 $ sum(feedRegime,actHerds(herds,breeds,feedRegime,t1,m1)) $ isNodeBefore(nCur,nCur1)  
 $$iftheni.sows "%farmBranchSows%" == "on"  
 $(not sameas(herds,"piglets"))  
 $$endif.sows  
 ),  
 v\_herdStart(herds,breeds,t1,nCur1,m1)  
  
  
 $$iftheni.ch %cowHerd%==true  
\*  
\* --- minus, in case of cows, slaughtered before reaching the final age  
\*  
 -sum( (slgtCows,cows) $ (sum(feedRegime, actHerds(slgtCows,breeds,feedRegime,t1,m1))  
 $ sameas(cows,herds) $ (slgtCows.pos eq cows.pos)),  
 v\_herdStart(slgtCows,breeds,t1,nCur1,m1))  
 $$endif.ch  
 )  
\*  
\* --- Herd size dynamic for piglets separately to depict a correct transfer from year t to year t1 as well as account for temporal resolution adjustments  
\*  
  
 $$iftheni.sows "%farmBranchSows%" == "on"  
 + sum( (t\_n(t1,nCur1),m1) $ ( (abs(p\_mDist(t,m,t1,m1)) le (p\_prodLengthB(herds,breeds) -1 $ (p\_prodLengthB(herds,breeds) eq 1)))  
 $ (p\_mDist(t,m,t1,m1) le 0) $ isNodeBefore(nCur,nCur1)  
 $ sum(feedRegime,actHerds(herds,breeds,feedRegime,t1,m1))  
 $ (not sameas(herds,"sows"))  
 ${ ( sameas(t,t1) $ (not sameas(m - p\_prodLengthB(herds,breeds),m1)))  
 or ((not sameas(t,t1)) $ (sameas("Jan",m))$ (sameas( m + 11, m1)))}),  
  
 v\_herdStart(herds,"",t1,nCur1,m1))  
 $$endif.sows  
 ;

The definition of the number of animals being added to the herd, *v\_herdStart*, is described in the equation *herdBal\_*. In the simplest case, where a 1:1 relation between a delivery and a use process exists, the number of new animals entering the different use processes *balherds* is equal to the number of new animals of the delivery process *herds*. This relation is depicted by the *herds\_from\_herds* set.

One possible extension is that animals entering the herd can be alternatively bought from the market, defined by the set *bought\_to\_herds*. The symmetric case is when the raised/fattened animals are sold, which is described by the *sold\_from\_herds* set.

For the case where several delivering processes are available, for example heifers of a different process length replacing cows, the set *herds\_from\_herds* describes a 1:n relation. A similar case exists if one type of animal, say a raised female calve, can be used for different processes such as replacement or slaughter, such that the expression turns into a n:1 relation. This case is captured by second additive expression in the equation.

In comparative static mode *p\_compStatHerd*, all lags are removed such that a steady-state herd model is described.

herdsBal\_(balHerds,breeds,tCur(t),nCur,m) $ ( sum(feedRegime,actherds(balHerds,breeds,feedRegime,t,m)) $ t\_n(t,nCur)  
\*  
 $ (p\_Year(t) le p\_year("%lastYear%"))  
 $ (sum( (herds\_from\_herds(balHerds,herds,breeds),t1,m1)  
 $ ( (-p\_mDist(t,m,t1,m1) eq round(p\_prodLengthB(herds,breeds)/(12/card(herdM)))\* (12/card(herdM)) )  
 $ sum(feedRegime,actHerds(herds,breeds,feedRegime,t1,m1))),1)  
 $$iftheni.compStat "%dynamics%" == "comparative-static"  
 or (sum( (herds\_from\_herds(balHerds,herds,breeds),t1,m1)  
 $ ( (-p\_mDist(t,m,t1,m1)+12 eq round(p\_prodLengthB(herds,breeds)/(12/card(herdM)))\* (12/card(herdM)) )  
\* and not (-p\_mDist(t,m,t1,m1) eq round(p\_prodLengthB(herds,breeds)/(12/card(herdM)))\* (12/card(herdM)) )  
 $ sum(feedRegime,actHerds(herds,breeds,feedRegime,t1,m1))),1))  
 $$endif.compStat  
 or sum((bought\_to\_herds(herds,breeds,balherds),feedRegime) $ actherds(herds,breeds,feedRegime,t,m),1)  
 or sum((sold\_from\_herds(herds,breeds,balherds),feedRegime) $ actherds(herds,breeds,feedRegime,t,m),1) )  
 ) ..  
\*  
\* --- herd starting at current time point  
\*  
 v\_herdStart(balHerds,breeds,t,nCur,m)/p\_herdYearScaler(balHerds,breeds)  
  
\*  
\* --- plus herd starting at current time point which compete for the same input herds  
\*  
  
 + sum( herds1 $ [ (sum(herds\_from\_herds(herds1,herds,breeds)  
 $ herds\_from\_herds(balHerds,herds,breeds),1)  
 or sum(bought\_to\_herds(herds,breeds,herds1)  
 $ bought\_to\_herds(herds,breeds,balherds),1))  
 $ (not sameas(balHerds,herds1)) $ sum(feedRegime,actherds(herds1,breeds,feedRegime,t,m))],  
 v\_herdStart(herds1,breeds,t,nCur,m)/p\_herdYearScaler(herds1,breeds))  
  
\*  
\* --- sold animals from the process (e.g. female calv one year old)  
\*  
 + sum( sold\_from\_herds(herds,breeds,balherds) $ sum(feedRegime,actherds(herds,breeds,feedRegime,t,m)),  
 v\_herdStart(herds,breeds,t,nCur,m))  
  
  
 =e=  
\*  
\* --- equal to the starting herd of the process wich generates these herds  
\*  
 + sum( (herds\_from\_herds(balHerds,herds,breeds),t\_n(t1,nCur1),m1)  
 $ (( (-p\_mDist(t,m,t1,m1) eq round(p\_prodLengthB(herds,breeds)/(12/card(herdM)))\* (12/card(herdM)) )  
 $$iftheni.compStat "%dynamics%" == "comparative-static"  
 or (-p\_mDist(t,m,t1,m1)+12 eq round(p\_prodLengthB(herds,breeds)/(12/card(herdM)))\* (12/card(herdM)) )  
 $$endif.compStat  
 ) $ sum(feedRegime,actHerds(herds,breeds,feedRegime,t1,m1)) $ isNodeBefore(nCur,nCur1)),  
 v\_herdStart(herds,breeds,t1,nCur1,m1))  
  
\*  
\* --- bought to herd (e.g. heifers bought from market)  
\*  
 + sum( (bought\_to\_herds(herds,breeds,balherds))  
 $ sum(feedRegime,actherds(herds,breeds,feedRegime,t,m)), v\_herdStart(herds,breeds,t,nCur,m));

## Cattle Module

!!! abstract Herds can be differentiated by animal types - such as cow, heifer, calf -, breeds, and feeding regimes.

The cattle module is closely related to the general herd module. It describes the demographic relations between cattle types (dairy cows, mother cows, male and female calves, heifers, young bulls) on the farm. New-born calves can be sold immediately or after one year or being raised to a heifer or young bulls, respectively. The heifer process, starting with a female calf raised for one year, is available in three intensity levels, leading to different process lengths (12, 21, 27 month) and thus first calving ages (12, 33 and 40 months) for the replacement. In Figure 3 the general concept of the cattle module and its decision points are illustrated.

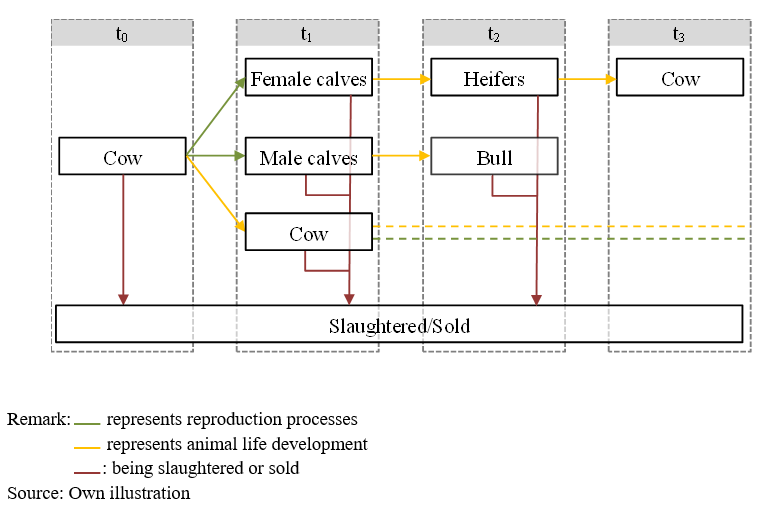


Figure 3: Cattle herd module management decisions

The number of new calves *v\_herdStart* are differentiated by gender and breed, in a year *t*, and specific month *m*, and depend on the herd size of cows of each breed and a specific calving coefficients. *ActHerds* is a flag set to define which herds might enter the solution for a specific year.

newCalves\_("%basBreed%",t,nCur,m) $ ( sum( (calvs,feedRegime), actHerds(calvs,"%basBreed%",feedRegime,t,m))  
 $ (p\_Year(t) le p\_year("%lastYear%")) $ t\_n(t,nCur)) ..  
\*  
\* --- new born calves (for females by genetic potential for milk yield) are born  
\* from the current herd of cows  
\*  
 v\_herdStart("fCalvsRais","%basBreed%",t,nCur,m) $ sum(feedRegime,actHerds("fCalvsRais","%basBreed%",feedRegime,t,m))  
 + v\_herdStart("fCalvsSold","%basBreed%",t,nCur,m) $ sum(feedRegime,actHerds("fCalvsSold","%basBreed%",feedRegime,t,m))  
 + v\_herdStart("mCalvsSold","%basBreed%",t,nCur,m) $ sum(feedRegime,actHerds("mCalvsSold","%basBreed%",feedRegime,t,m))  
 + v\_herdStart("mCalvsRais","%basBreed%",t,nCur,m) $ sum(feedRegime,actHerds("mCalvsRais","%basBreed%",feedRegime,t,m))  
 $$iftheni.crossBreed "%crossBreeding%"=="true"  
 + v\_herdStart("fCalvsRais","%crossBreed%",t,nCur,m) $ sum(feedRegime,actHerds("fCalvsRais","%crossBreed%",feedRegime,t,m))  
 + v\_herdStart("fCalvsSold","%crossBreed%",t,nCur,m) $ sum(feedRegime,actHerds("fCalvsSold","%crossBreed%",feedRegime,t,m))  
 + v\_herdStart("mCalvsSold","%crossBreed%",t,nCur,m) $ sum(feedRegime,actHerds("mCalvsSold","%crossBreed%",feedRegime,t,m))  
 + v\_herdStart("mCalvsRais","%crossBreed%",t,nCur,m) $ sum(feedRegime,actHerds("mCalvsRais","%crossBreed%",feedRegime,t,m))  
 $$endif.crossBreed  
  
  
 =e= sum( (cows,t1,nCur1,m1,mDist) $ (sum(feedRegime,actHerds(cows,"%basBreed%",feedRegime,t1,m1))  
 $ ( (mDist.pos eq -p\_mDist(t,m,t1,m1))  
 or (mDist.pos eq -p\_mDist(t,m,t1,m1)+12) $ p\_compStatHerd)  
 $ t\_n(t1,nCur1)),  
 v\_herdStart(cows,"%basBreed%",t1,nCur1,m1) \* p\_calvCoeff(cows,"%basBreed%",mDist));

The calving coefficients are defined in the *Cows* tab of the GUI. Here, the amount of births per lactation, living calves per birth, calf losses, and days between births can be set for the different breeds Holstein-Friesian (HF), Simmental (SI, which stands a placeholder for the individual breed defined in the GUI), and mother cows. The values are stored in the parameter p\_calvAttr.

The amount of calves that are born in a given month is derived from the information entered in the GUI with the help of an entropy estimator. For the sake of simplicity, but without loss of generality, it is assumed that birth is equally likely in the two months surrounding the average calving interval.

set curCycleLength / l11\*l15 /;  
\*  
\* --- entropy estimator  
\*  
 variable v\_ent "Entropy";  
 positive variables v\_prob(cowTypes,curCycleLength)  
  
 parameter p\_cycleLength(curCycleLength);  
 p\_cycleLength(curCycleLength) = 10 + curCycleLength.pos;  
  
  
 equation e\_ent "Entropy definition"  
 e\_daysBetweenBirths "Recover given information on inter calving interval"  
 e\_sumUnity "Probs add up to unity"  
 ;  
  
 e\_ent .. -v\_ent =E= sum( (cowTypes,curCycleLength),v\_prob(cowTypes,curCycleLength)  
 \* log(v\_prob(cowTypes,curCycleLength)/card(curCycleLength)));  
  
 e\_daysBetweenBirths(cowTypes) ..  
 p\_calvAttr(cowTypes,"daysBetweenBirths")/30.5  
 =E= sum(curCycleLength, v\_prob(cowTypes,curCycleLength)\* p\_CycleLength(curCycleLength));  
  
 e\_sumUnity(cowTypes) .. sum(curCycleLength, v\_prob(cowTypes,CurCycleLength)) =E= 1;  
  
 v\_prob.up(cowTypes,curCycleLength) = 1;  
 v\_prob.lo(cowTypes,curCycleLength) = 1.E-5;  
  
 v\_prob.fx(cowTypes,curCycleLength) $ (p\_calvAttr(cowTypes,"daysBetweenBirths")/30.5 lt p\_CycleLength(curCycleLength)-1) = 1.E-6;  
 v\_prob.fx(cowTypes,curCycleLength) $ (p\_calvAttr(cowTypes,"daysBetweenBirths")/30.5 gt p\_CycleLength(curCycleLength)+1) = 1.E-6;  
  
 v\_prob.l(cowTypes,CurCycleLength) = 1/card(CurCycleLength);  
  
 model m\_ent / e\_ent,e\_daysBetweenBirths,e\_sumUnity /;  
 solve m\_ent maximizing v\_ent using NLP;

The calving probabilities are then mapped to the actual endogenous calving distribution in the parameter p\_calvCoeff, which is subsequently used in the herdStart\_ equation.

p\_calvCoeff(dCows,"%basBreed%",mDist)  
 $ ( (mDist.pos ge 12)  
 $ (mDist.pos/12 le (ceil(p\_nlac(dcows)))\*p\_calvAttr("%cowType%","daysBetweenBirths")/365))  
 = sum( curCycleLength $ (mod(mDist.pos-1,curCycleLength.pos+11) eq 0),  
 p\_livingCalvesPerYear(dCows,"%basBreed%") \* v\_prob.l("%cowType%",curCycleLength) );

For a cow with a lifespan of four lactations, the calving distribution is depicted in the following figure. Notice how the distribution widens with increasing amounts of lactations.

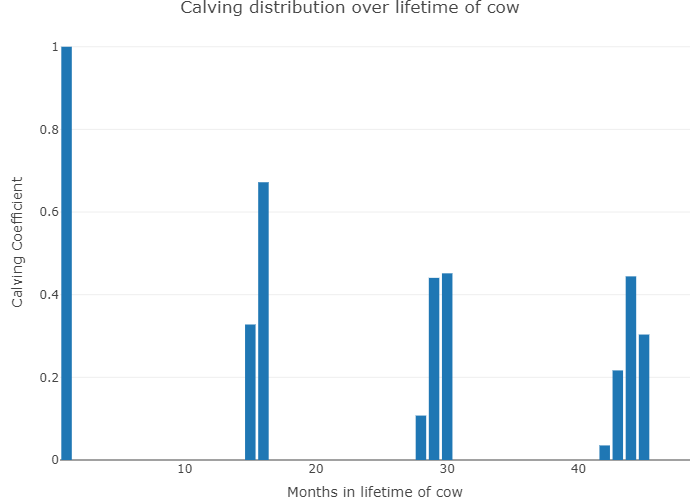
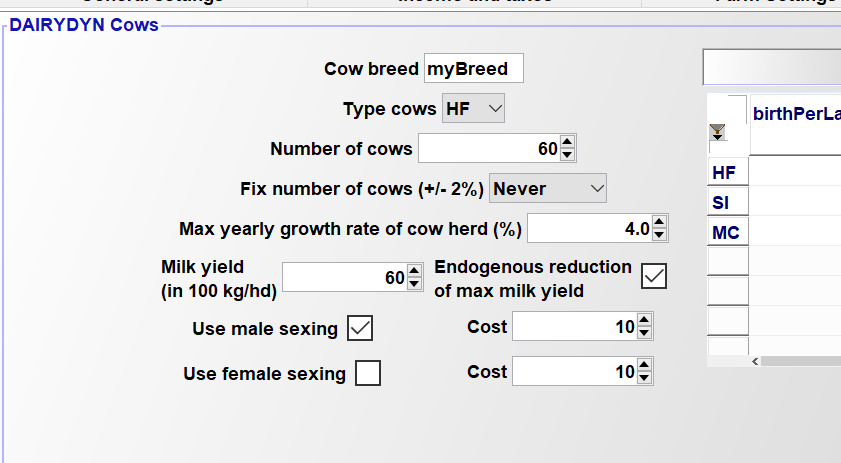


Figure 3: Calving distribution of a cow with four lactations, according to the endogenous calculation. Source: Own representation.

In order to provide a better overview of model results in the equation listing, a yearly average herd size is calculated in the equation sumHerds\_:

sumHerds\_(sumHerds,breeds,feedRegime,t,nCur,m) $ (t\_n(t,nCur)  
\* $ sum(sum\_herds(sumHerds,PossHerds) $ (p\_prodLength(possHerds,breeds) gt 1),1)  
 $ sum(sum\_herds(sumHerds,possHerds) $ actHerds(possHerds,breeds,feedRegime,t,m),1)) ..  
  
 v\_herdSize(sumHerds,breeds,feedRegime,t,nCur,m)  
 =e= sum(sum\_herds(sumHerds,possHerds) $ actHerds(possHerds,breeds,feedRegime,t,m) ,  
 v\_herdSize(possHerds,breeds,feedRegime,t,nCur,m) $ (p\_prodLength(possHerds,breeds) gt 1)  
 + v\_herdStart(possHerds,breeds,t,nCur,m) $ (p\_prodLength(possHerds,breeds) eq 1));

## Sexing

On interface: define if it used to generate males and/or females and define costs 

In the “model.gms”, male and female sexing are treated as inputs with their prices:

buy\_(curinputs(inputs),tCur(t),nCur) $ (t\_n(t,nCur)) ..

\* --- costs of sexing  
\*  
 + sum((breeds,m),v\_sexingF(breeds,t,nCur,m)) $ sameas(inputs,"femaleSexing")  
 + sum((breeds,m),v\_sexingM(breeds,t,nCur,m)) $ sameas(inputs,"maleSexing")

The prices are taken from the interface and introduced in “coeffgen.gms”:

p\_inputPrices("maleSexing","price") = %costMaleSexing%;

p\_inputPrices("femaleSexing","price") = %costfemaleSexing%;

If the user has sexing switched off, the variable is fixed to zero in “model\_starting\_bounds”:

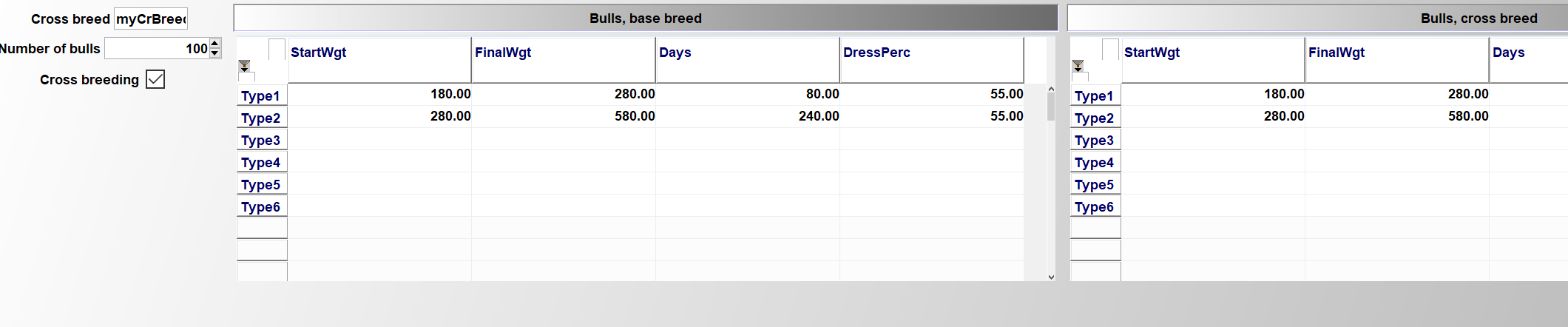
\* --- switch off sexing if not activated by user  
  
 $$ifi not "%useFemaleSexing%"=="true" v\_sexingF.fx(breeds,t\_n(t,nCur),m) = 0;  
 $$ifi not "%useMaleSexing%"=="true" v\_sexingM.fx(breeds,t\_n(t,nCur),m) = 0;

Sexing changes the male-female balance equation (see model\_module.gms):

maleFemaleRel\_(breeds,t,nCur,m) $( sum( (feedRegime,calvs), actHerds(calvs,breeds,feedRegime,t,m))  
 $ (p\_Year(t) le p\_year("%lastYear%")) $ t\_n(t,nCur)) ..  
  
  
 [  
 v\_herdStart("fCalvsRais",breeds,t,nCur,m) $ sum(feedRegime,actHerds("fCalvsRais",breeds,feedRegime,t,m))  
 + v\_herdStart("fCalvsSold",breeds,t,nCur,m) $ sum(feedRegime,actHerds("fCalvsSold",breeds,feedRegime,t,m))  
 ] /0.495  
  
 + v\_sexingF(breeds,t,nCur,m)\*0.5  
 - v\_sexingM(breeds,t,nCur,m)\*0.5  
  
 =E=  
 - v\_sexingF(breeds,t,nCur,m)\*0.5  
 + v\_sexingM(breeds,t,nCur,m)\*0.5  
  
 +[  
 v\_herdStart("mCalvsRais",breeds,t,nCur,m) $ sum(feedRegime,actHerds("mCalvsRais",breeds,feedRegime,t,m))  
 + v\_herdStart("mCalvsSold",breeds,t,nCur,m) $ sum(feedRegime,actHerds("mCalvsSold",breeds,feedRegime,t,m))  
 ] / 0.505;

If sexing is switched off, the number of female siblings (LHS) must be (approximately) equal to the males ones (RHS). Sexing an insemination to male will take 0.5 female out and increase the number of males of 0.5. Female sexing leads to the opposite effect.

## Cross-Breeding

Cross breeding can be switched on the interface on the “bulls” tab: 

As consequence, a second table is offered where data for the cross-breed can be entered.

The cross-breeds enter the calves balance (\*model\_module.gms\*) The left hand side adds up all male and female calves, if cross-breeding is switched on, adding the cross-breeds.

newCalves\_("%basBreed%",t,nCur,m) $ ( sum( (calvs,feedRegime), actHerds(calvs,"%basBreed%",feedRegime,t,m))  
 $ (p\_Year(t) le p\_year("%lastYear%")) $ t\_n(t,nCur)) ..  
\*  
\* --- new born calves (for females by genetic potential for milk yield) are born  
\* from the current herd of cows  
\*  
 v\_herdStart("fCalvsRais","%basBreed%",t,nCur,m) $ sum(feedRegime,actHerds("fCalvsRais","%basBreed%",feedRegime,t,m))  
 + v\_herdStart("fCalvsSold","%basBreed%",t,nCur,m) $ sum(feedRegime,actHerds("fCalvsSold","%basBreed%",feedRegime,t,m))  
 + v\_herdStart("mCalvsSold","%basBreed%",t,nCur,m) $ sum(feedRegime,actHerds("mCalvsSold","%basBreed%",feedRegime,t,m))  
 + v\_herdStart("mCalvsRais","%basBreed%",t,nCur,m) $ sum(feedRegime,actHerds("mCalvsRais","%basBreed%",feedRegime,t,m))  
 $$iftheni.crossBreed "%crossBreeding%"=="true"  
 + v\_herdStart("fCalvsRais","%crossBreed%",t,nCur,m) $ sum(feedRegime,actHerds("fCalvsRais","%crossBreed%",feedRegime,t,m))  
 + v\_herdStart("fCalvsSold","%crossBreed%",t,nCur,m) $ sum(feedRegime,actHerds("fCalvsSold","%crossBreed%",feedRegime,t,m))  
 + v\_herdStart("mCalvsSold","%crossBreed%",t,nCur,m) $ sum(feedRegime,actHerds("mCalvsSold","%crossBreed%",feedRegime,t,m))  
 + v\_herdStart("mCalvsRais","%crossBreed%",t,nCur,m) $ sum(feedRegime,actHerds("mCalvsRais","%crossBreed%",feedRegime,t,m))  
 $$endif.crossBreed  
  
  
 =e= sum( (cows,t1,nCur1,m1,mDist) $ (sum(feedRegime,actHerds(cows,"%basBreed%",feedRegime,t1,m1))  
 $ ( (mDist.pos eq -p\_mDist(t,m,t1,m1))  
 or (mDist.pos eq -p\_mDist(t,m,t1,m1)+12) $ p\_compStatHerd)  
 $ t\_n(t1,nCur1)),  
 v\_herdStart(cows,"%basBreed%",t1,nCur1,m1) \* p\_calvCoeff(cows,"%basBreed%",mDist));

The activation also affect *coeffgen-herds.gms*:

$$iftheni.crossBreed "%crossBreeding%"=="true"  
  
 actHerds("fCalvsRais","%crossBreed%",feedRegimeCattle,t,m) = yes;  
 actHerds("fCalvsSold","%crossBreed%","",t,m) = yes;  
 actHerds("mCalvsRais","%crossBreed%",feedRegimeCattle,t,m) = yes;  
 actHerds("mCalvsSold","%crossBreed%","",t,m) = yes;  
 herds\_from\_herds("mCalvsRaisSold","mCalvsRais","%crossBreed%") = yes;  
\* herds\_from\_herds(bulls,"mCalvsRais","%crossBreed%") = yes;  
\* herds\_from\_herds(heifsCross,"fCalvsRais","%crossBreed%") = yes;  
  
 $$endif.crossBreed

By setting the *actHerds* indicator set active for the cross-breeds. Accordingly, the sets for bulls work on a set which can include the cross-breed:

set bullBreeds(Breeds) / "%basBreed%" /;  
 $$ifi "%crossBreeding%"=="true" bullBreeds("%crossBreed%") = yes;

## Pig Module

!!! abstract The pig module distinguishes between fattening- and piglet production systems. Fattening pigs are subdivided into different phases to account for different feeding requirements and excretion values. The piglet production system differentiates between sows, young piglets and weaners.

The pig module, similar to the cattle module, is closely linked with the general herd module. The herd dynamics of the pig module are shown in Figure 4.

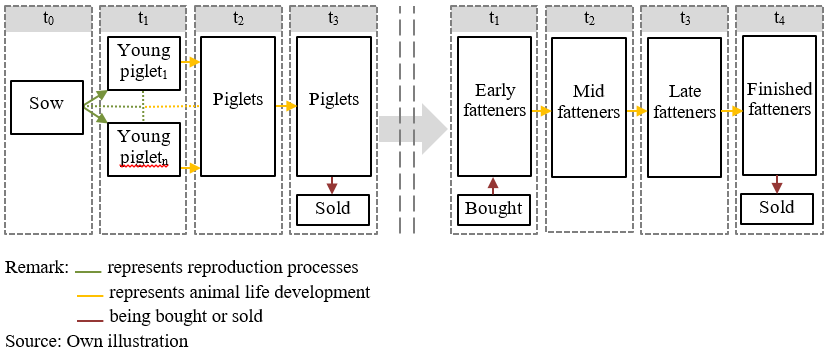


Figure 4: Pig module management decisions.

The piglet production process starts with the production of young piglets born to sows, shown in the following equation:

newPiglets\_(tCur(t),nCur,herdm) $ (sum(feedRegime,actHerds("sows","",feedRegime,t,herdm)) $ t\_n(t,nCur)) ..  
  
 v\_herdStart("youngPiglets","",t,nCur,herdm)  
 =e= sum(feedRegime $ actHerds("sows","",feedRegime,t,herdm),  
 v\_herdSize("sows","",feedRegime,t,nCur,herdm) \* p\_OCoeff("sows","youngPiglet","",t))/card(herdM);

Each sow produces on average 26.7 young piglets per year in the default parameterisation. After one month young piglets become weaners and remain 2 months within the herd before they are sold or transferred to the fattener branch. Labour and feed requirements are chosen according to a growing period of 41 days and a weight gain from 8 to 30 kg. The feeding-, stable- and labour requirements of the piglet production branch are steered by the sows and piglets herd size.

The fattener farm branch distinguishes between four different stages of fatteners to account for different feeding and excretion values during the production process. Feeding levels and excretion values are connected via the set *feedregime*. This set allows to adapt feeding patterns, for instance to adjust nutrient output in response to legislatively given fertiliser restrictions. For a more thorough explanation of the feeding options, please refer to the pig feeds module in section 2.2.2. The piglets bought in a month are immediately transferred into early fatteners and are transferred to the next fattening stage after a month until they become fatteners and are sold as fattened pigs. Each stage lasts for one month. The weight development during the fattening process is assumed from 28 to 118kg live weight.

As mentioned in the general herd module, the equations such as herd balance *herdsBal\_* and herd size, *herdSize\_* are used for the herd dynamic in the pig module. The following model code shows the elements of the herd used in the farm branch for sows.

$$iftheni.sows "%farmBranchSows%" == "on"  
  
 herds\_from\_herds("piglets","youngPiglets","") = yes;  
  
 bought\_to\_herds("youngSows","","sows") = yes;  
  
 actHerds("piglets","",feedRegimePigs,t,m) = yes;  
 actHerds("sows","",feedRegimePigs,t,m) = yes;  
 actHerds("youngPiglets","",feedRegimePigs,t,m) = yes;  
 actHerds("youngSows","",feedRegimePigs,t,m) = yes;  
 $$endif.sows

The statements below show the elements of the herd used in the farm branch for fatteners.

$iftheni.pigHerd %pigHerd% == true  
 $$iftheni.fattners "%farmBranchFattners%" == "on"  
  
 actHerds("Fattners","",feedRegimePigs,t,m) = yes;  
 actHerds("earlyFattners","",feedRegimePigs,t,m) = yes;  
 actHerds("midFattners","",feedRegimePigs,t,m) = yes;  
 actHerds("lateFattners","",feedRegimePigs,t,m) = yes;  
 actHerds("pigletsBought","",feedRegimePigs,t,m) = yes;  
  
 bought\_to\_herds("pigletsBought","","earlyFattners") = yes;  
  
 herds\_from\_herds("midfattners","earlyfattners","") = yes;  
 herds\_from\_herds("lateFattners","midFattners","") = yes;  
 herds\_from\_herds("Fattners","lateFattners","") = yes;  
  
 $$endif.fattners

# Feeding module

!!! abstract The feed module distinguishes between pig and cattle feeding requirements. For a dairy herd, it captures a cost minimal feed mix from own produced fodder and different types of concentrates at given requirements per head and intra-year feeding periods (energy, protein, dry matter) for each cattle herd. For pigs it determines a cost minimal feed mix from own produced and purchased fodder and concentrates such as soybean meal and soy oil. For both branches, different feeding phases for reduced nitrogen and phosphorus output can be used.

## Cattle Feed Module

The feeding module for cattle consists of two major elements:

1. **Requirement functions** and related constraints in the model template
2. **Feeding activities**, which ensure that requirements are covered and link the animal to the cropping sector as well as to purchases of concentrates

The requirements are defined in *coeffgen\requ.gms*. Requirements for dairy cows are differentiated by annual milk yield and by lactation period. The model differentiates 5 lactations period with different lengths (30 -- 70 -- 100 -- 105 -- 60 days, where the last 60 days are the dry period). The periods are labelled according to their last day, e.g. *LC200* is the period from day 101 to day 200, *LC305* is the period from the 201st to the 305th day and *dry* denotes the last 60 days of lactation.

This excurse describes the derivation of the output coefficient for each lactation phase, hence how much of yearly milk yield is produced by each cow on one day.

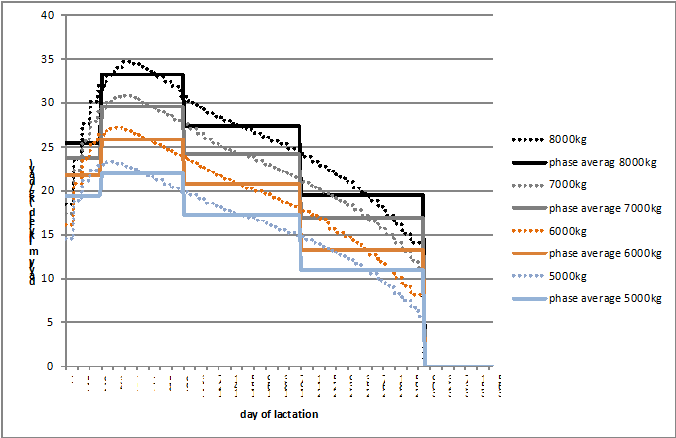


Figure 6: Lactation curves of different yearly milk yield potentials and average milk yield in different lactation phases (30-70-100-105-60). Remark: Calculation based on Huth (1995:pp.224-226) Source: own illustration

Using the above shown lactation functions, the daily fraction of the yearly milk yield in each lactation phase can be derived. The mean over the four milk yield potentials of the coefficients are shown in Table 1.

Daily fraction of whole lactation milk yield in different lactation phases Remark: Own calculation based on Huth (1995, pp.224-226)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | LC30 | LC100 | LC200 | LC3005 | Dry |
| Daily fraction | 0.00356 | 0.0043 | 0.00333 | 0.00233 | 0 |

Following these outputs, e.g. on each of the first 30 days of lactation, the cow produces 0.356% of the yearly milk yield (e.g. 28 kg per day for a cow which produces 8000 kg per year). In a next step, these coefficients are used to calculate the sum of milk output in each lactation phase to further calculate feed requirements stemming from the herds in each phase.

The daily milk yield in each period is based on the following statements which define milk yield in ton/year, stored on the general output coefficient parameter *p\_OCoeff*. The coefficient is scaled to match total yearly milk yield.

p\_mlkPerDay(dcows,"%basBreed%","LC30") = 0.003555556 \* sum(t $ (t.pos eq 1), p\_OCoeff(dcows,"milk","%basBreed%",t) \* 1000);  
p\_mlkPerDay(dcows,"%basBreed%","LC92") = 0.004333333 \* sum(t $ (t.pos eq 1), p\_OCoeff(dcows,"milk","%basBreed%",t) \* 1000);  
p\_mlkPerDay(dcows,"%basBreed%","LC213") = 0.003333333 \* sum(t $ (t.pos eq 1), p\_OCoeff(dcows,"milk","%basBreed%",t) \* 1000);  
p\_mlkPerDay(dcows,"%basBreed%","LC305") = 0.002333333 \* sum(t $ (t.pos eq 1), p\_OCoeff(dcows,"milk","%basBreed%",t) \* 1000);

The model differentiates between requirements for energy in net-energy for lactation, raw protein and maximum dry matter. The feeding requirements are described by the parameter p\_reqsPhase for each herd and a certain requirement phase. As described earlier, the requirement phases of cows are differentiated into specific, fixed stages during lactation. For bulls, heifers, and calves, the amount of feeding/requirement phases are defined over the GUI. For each feeding phase, the daily requirements during the production process are identical.

The requirement functions account for differing start and final weights, as well as daily weight gains of the animals. The underlying regression models were kindly provided by the Institut für Tierernährung und Futterwirtschaft of the Bayerische Landesandstalt für Landwirtschaft (LfL)[[2]](#footnote-44).

The requirements per requirement phase p\_reqsPhase are subsequently converted into values per month, in order to ensure that the animals are correctly fed throughout the requirement period.

p\_reqsPhaseMonths(herds,curBreeds,reqsPhase,reqs) $ p\_reqsPhaseLength(herds,curBreeds,reqsPhase)  
 = p\_reqsPhase(herds,curBreeds,reqsPhase,reqs)/p\_reqsPhaseLength(herds,curBreeds,reqsPhase) \* 30.5;

The monthly requirements per planning period, p\_reqsPhaseMonths, enter the equation structure of the model. The equations are differentiated by herd, year, planning period and state-of-nature (SON), and ensure the requirements are covered by an appropriate feed mix made out of different feeding stuff [^3]. The composition of the feed mix is determined endogenously. The total feed requirements for a farm in the different intra-yearly planning periods depend on the distribution of calving dates in the cow herd, therefore, cows of the same milk yield potential can be in different lactation phases during the year. The requirements of tons of feed, *v\_feeding*, are differentiated by herd, breed, planning period (lactation phase of cow), SON and year, if the requirement phases are not defined for specific time spans after the herd start:

reqs\_(possHerds,breeds,feedRegime,reqs,reqsPhase,m,t\_n(tCur,nCur))  
 $ sum( m\_to\_herdm(m,herdm)  
 $ (p\_reqsPhaseLengthMonths(possHerds,breeds,"gen")  
 $ p\_reqsPhaseMonths(possHerds,breeds,reqsphase,reqs)),  
 actHerds(possHerds,breeds,feedRegime,tCur,herdm)) ..  
\*  
\* --- herd size times requirements per head, minus year and SON specific reduction in milk yield  
  
 sum((m\_to\_herdm(m,herdm)) $ actHerds(possHerds,breeds,feedRegime,tCur,herdm),  
 v\_herdSize(possHerds,breeds,feedRegime,tCur,nCur,herdm)  
 \*p\_corrHerdm(possHerds,breeds)\*p\_reqsPhaseMonths(possHerds,breeds,reqsPhase,reqs))  
\*  
\* --- must be covered by feeding times the content of the feed stuff  
\*  
 =L= sum( feeds,v\_feeding(possHerds,breeds,feedRegime,reqsPhase,m,feeds,tCur,nCur) \* p\_feedContFMton(feeds,reqs))$(sameas(feedregime,"noGraz"))  
 +sum( feeds $(not sum( sameas(feeds,roughage),1)),v\_feeding(possHerds,breeds,feedRegime,reqsPhase,m,feeds,tCur,nCur) \* p\_feedContFMton(feeds,reqs))$(not sameas(feedregime,"noGraz"))  
 ;

Alternatively, requirements can be linked to the start point of an animal process to break down the total requirement during the length of the production processes in phases. The equation is only switched on if the parameter p\_reqsPhaseLength is non-zero:

reqsPhase\_(possHerds,breeds,feedRegime,reqs,reqsPhase,m,t\_n(tCur,nCur))  
 $ (sum(m\_to\_herdm(m,herdm),actHerds(possHerds,breeds,feedRegime,tCur,herdm))  
 $ (not p\_reqsPhaseLengthMonths(possHerds,breeds,"gen"))  
 $ p\_reqsPhase(possHerds,breeds,reqsPhase,reqs)) ..  
\*  
\* --- herds which started in the months before the production length  
\*  
\* -- number of months that herd in that requirement phase during that period  
\* multiplied with monthly requirements  
\*  
 sum((m\_to\_herdm(m,herdm)) $ actHerds(possHerds,breeds,feedRegime,tCur,herdm),  
 v\_herdsReqsPhase(possHerds,breeds,feedRegime,reqsphase,herdm,tCur,nCur)  
 \* p\_corrHerdm(possHerds,breeds) \* p\_reqsPhaseMonths(possHerds,breeds,reqsPhase,reqs))  
\*  
\* --- must be covered by feeding times the content of the feed stuff  
\*  
 =L= sum( feeds,v\_feeding(possHerds,breeds,feedRegime,reqsPhase,m,feeds,tCur,nCur) \* p\_feedContFMton(feeds,reqs))$(sameas(feedregime,"noGraz"))  
 +sum( feeds $(not sum(sameas(feeds,roughage),1)),v\_feeding(possHerds,breeds,feedRegime,reqsPhase,m,feeds,tCur,nCur) \* p\_feedContFMton(feeds,reqs))$(not sameas(feedregime,"noGraz"))  
 ;

In a next step feeding amounts are aggregated to total feed use, *v\_feeduse*, per each product and for each year, feed and planning period.

feedUse\_(feedsY,t\_n(tCur,nCur)) ..  
  
 v\_feedUse(feedsY,tCur,nCur)  
  
 =e= sum( (possHerds,breeds,feedRegime,reqsPhase,m) $ (actHerds(possHerds,breeds,feedRegime,tCur,m)  
 $ p\_reqsPhase(possHerds,breeds,reqsPhase,"DMMX")),  
 v\_feeding(possHerds,breeds,feedRegime,reqsPhase,m,feedsY,tCur,nCur));

For own produced feed which is not storable and shows a variable availability over the year, such as grass from pasture, an aggregation to the intra-year periods is done.

feedUseM\_(feedsM,m,t\_n(tCur,nCur)) ..  
  
 v\_feedUseM(feedsM,m,tCur,nCur)  
  
 =e= sum( (possHerds,breeds,feedRegime,reqsPhase)  
 $ (p\_reqsPhase(possHerds,breeds,reqsPhase,"DMMX")  
 $ sum(m\_to\_herdm(m,herdm),actHerds(possHerds,breeds,feedRegime,tCur,herdm))),  
 v\_feeding(possHerds,breeds,feedRegime,reqsPhase,m,feedsM,tCur,nCur));

## Pigs Feed Module

The feeding requirements for the piglet production branch differentiate between sows with the attached young piglets and the piglets after separation from the sows. Requirements are set for energy, crude protein, lysin, phosphorus feed and dry matter. Further, minimum and maximum requirements are set for certain feeds in order to reflect realistic feeding patterns. For example, a minimum requirement for oil in the feed intake is assumed to assure a correct viscosity.

The fattening branch distinguishes between four fattening stages to provide the option of nitrogen and phosphorus reduced feeding (N/P). It includes the stages *earlyFattners*, *midFattners*, and *lateFattners*, Fattners.\* Three feeding regimes are applicable, which are: normal feed, reduced N/P feed and highly reduced N/P feed. The primary differences between the feeding schemes are the adjustments of daily nutrient requirements depending on the stage a fattening pig is currently in. For instance, with the normal feed there are only two different feeding requirements; a daily requirement for the weight range from 28-40 kg which is in the early fattening phase and a daily requirement from 40-118 kg which assumes daily feed requirements in the mid, late and finishing fattening stage. In contrast the N/P reduced feeding phase differentiates between daily nutrient requirements for the weight ranges 28-40kg, 40-70kg and 70-118kg. Thus, all stages require different daily nutrient requirements. In accordance with the piglet production branch, the fattening branch also imposes maximal and minimal values for certain products to account for digestibility, correct feeding textures and mineral provision.

The requirements are used to determine the optimal feeding mix shown in the equation *reqPigs\_*. Hence, it can be seen which feeding products are used by which herd type at a certain time. The equation *feedSourcePig\_* determines the source of feed, i.e. whether it is purchased or produced on farm.

reqPigs\_(possHerds,feedAttr,feedRegime,tCur(t),nCur,m) $( sum(actHerds(herds,"",feedRegime,t,m),1) $ t\_n(t,nCur)  
 $ (not sameas(possherds,"pigletsBought")) $ (not sameas(possherds,"youngSows"))  
 $ (not sameas(possherds,"youngPiglets"))) ..  
  
 v\_herdSize(possHerds,"",feedRegime,t,nCur,m) \* p\_feedReqPig(possHerds,feedRegime,feedAttr)  
  
\* --- Accounting for temporal resolution for sows  
  
$iftheni.sows "%farmBranchSows%" == "on"  
 \* 12/card(herdM)  
$endif.sows  
 =L=  
 sum(feedspig , v\_feedingPig(possherds,feedsPig,feedRegime,t,nCur,m) \* p\_feedAttrPig(feedsPig,feedAttr));

feedSourcePig\_(feedspig,tCur(t),nCur) $ ( sum(actHerds(herds,"",feedRegime,t,m),1) $ t\_n(t,nCur)) ..  
  
 (v\_feedOwnPig(feedspig,t,nCur) $ (sum(sameas(curProds,feedspig),1))) + v\_feedPurchPig(feedspig,t,nCur)  
 =E=  
 sum((possherds,feedRegime,m) $ ((not sameas(possherds,"pigletsBought")) $ (not sameas(possherds,"youngSows"))  
 $ (not sameas(possherds,"youngPiglets"))),  
 v\_feedingPig(possherds,feedsPig,feedRegime,t,nCur,m));

The upper and lower bound for the feeding mix are then determined by *feedTot\_, feedmax\_, feedMin\_* (not additionally shown here) which allows certain flexibility in the feeding mix.

However, for the fatteners the feeding mix is fixed for different feeding regimes to precisely reproduce empirically found feeding ratios.

p\_feedMinPigday(feedRegime,massPhases,feedspig)  
  
 soybeanMeal rapeSeedMeal PlantFat MinFu MinFu2 MinFu3 MinFu4  
  
 normFeed.stg28\_40 0.23 0.015 0.037  
 normFeed.stg40\_118 0.195 0.015 0.032

# Cropping, Land and Land Use

!!! abstract The cropping module optimises the cropping pattern subject to land availability, reflecting yields, prices, machinery and fertilising needs and other variable costs for a selectable list of arable crops. The crops can be differentiated by production system (conventional, organic), tillage (plough, minimal tillage, no tillage) and intensity level (normal and reduced fertilisation in 20% steps). Machinery use is linked to field working-day requirements depicted with a bi-weekly resolution during the relevant months. Crop rotational constraints can be either depicted by introducing crop rotations or by simple maximal shares. The model can capture plots which are differentiated by soil and land type (grassland, arableland and pasture) and size.

Crop activities are differentiated by crop, *crops*, soil types, *soil,* management intensity, *intens*, and tillage type, *till*. The use of different management intensities and tillage types is optional. Management intensities impact yield levels (see chapter 2.11.1.1). Necessary field operations and thus variable costs, machinery and labour needs reflect intensity and tillage type as well.

## Cropping Activities in the Model

The farmer is assumed to be able to adjust on a yearly basis its land use as long as the labour, machinery and further restrictions allow for it. Land is differentiated into arable and permanent grass land, *landType*. Land use decisions can be restricted by maximal rotational shares for the individual crops. The set *plot* differentiates the land with regard to plot size, soil type and climate zone. The attributes of plots and the number of plots from 1 to 20, is defined in the GUI.

The total land endowment is calculated in the equation *totPlotLand\_* as the sum of the initial endowment, *p\_plotSize(plot)*, and land purchased, *v\_buyLand*, in the past or current year.

totPlotLand\_(plot,tCur(t),nCur) $ (p\_plotSize(plot) $ t\_n(t,nCur)) ..  
  
 v\_totPlotLand(plot,t,nCur)  
  
 =E=  
\*  
\* --- initialize of plots  
\*  
 p\_plotSize(plot)  
\*  
\* --- plus bought adjacent plots (= merged)  
\*  
$ifi %landBuy% == true + sum(t\_n(t1,nCur1) $ (tcur(t1) $ isNodeBefore(nCur,nCur1) $ (ord(t1) le ord(t))), v\_buyPlot(plot,t1,nCur1))  
 ;

Total cropped land is defined by the land occupied by the different crops, *v\_cropH*a. The *c\_s\_t\_i* set defines the active possible combinations of crops, soil type, tillage type/system and management intensity.

croppedLand\_(landType,soil,tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_croppedLand(landType,soil,t,nCur)  
 =e= sum( (curCrops(crops),plot\_lt\_soil(plot,landType,soil),till,intens)  
 $ c\_s\_t\_i(crops,plot,till,intens), v\_cropHa(crops,plot,till,intens,t,nCur)  
 $( not sameas (crops,"catchCrop")));

The total land *v\_totPlotLand* can be either used for cropping (including permanent grassland), *v\_croppedLand*, or rented out, v*\_rentOutLand*, on a yearly basis. The option to rent out land can be activated in the GUI:

plotland\_(plot,tCur(t),nCur) $ (p\_plotSize(plot) $ t\_n(t,nCur)) ..  
\*  
 v\_croppedPlotLand(plot,t,nCur)  
\*  
$ifi %landLease% == true + v\_rentOutPlot(plot,t,nCur)\*p\_plotSize(plot)  
\*  
 =L= v\_totPlotLand(plot,t,nCur);

That a farm stays within a maximum stocking rate ceiling, expressed in livestock units per ha, is ensured by the following equation. The maximal allowed stocking rate can be adjusted in the GUI:

luLand\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 sum( plot $ p\_plotSize(plot), v\_totPlotLand(plot,t,nCur)  
$ifi %landLease% == true -v\_rentOutPlot(plot,t,nCur) \* p\_plotSize(plot)  
 ) \* p\_maxStockRate =G=  
  
 sum(actHerds(possHerds,breeds,feedRegime,t,m) $ p\_prodLength(possHerds,breeds),  
 v\_herdSize(possHerds,breeds,feedRegime,t,nCur,m)  
  
 $$iftheni.branchF not "%farmBranchFattners%" == "on"  
 \* 1/min(12,p\_prodLength(possHerds,breeds)) \* 12/card(herdM)  
 $$endif.branchF  
  
 \* p\_lu(possHerds));

## Optional Crop Rotational Module

Alternatively to the use of maximum rotational shares (see previous section) the model offers an option of a three year crop rotation system. The rotation names (shown in the following list, see *model\templ\_decl.gms*), set *rot*, displays the order of the crops in the rotations. Each line depicts sequences of three crop types (do not have to be different) in a rotation with only the order being differently. This avoids unnecessary rigidities in the model.

set rot "Rotations" / WC\_WC\_PO,WC\_PO\_WC,PO\_WC\_WC  
 WC\_WC\_SC,WC\_SC\_WC,SC\_WC\_WC  
 WC\_WC\_SU,WC\_SU\_WC,SU\_WC\_WC  
 WC\_WC\_OT,WC\_OT\_WC,OT\_WC\_WC  
 WC\_WC\_ID,WC\_ID\_WC,ID\_WC\_WC  
  
 WC\_SC\_PO,SC\_PO\_WC,PO\_WC\_SC  
 WC\_SC\_SU,SC\_SU\_WC,SU\_WC\_SC  
 WC\_SC\_OT,SC\_OT\_WC,OT\_WC\_SC  
 WC\_SC\_ID,SC\_ID\_WC,ID\_WC\_SC  
  
 SC\_WC\_SC,SC\_SC\_WC,WC\_SC\_SC  
 SC\_SC\_ID,SC\_ID\_SC,ID\_SC\_SC  
 SC\_SC\_PO,SC\_SC\_SU,SC\_SC\_OT  
 WC\_PO\_ID,WC\_SU\_ID,WC\_OT\_ID  
 SC\_PO\_ID,SC\_SU\_ID,SC\_OT\_ID  
 WC\_ID\_ID,ID\_WC\_ID,ID\_ID\_WC  
 SC\_ID\_ID,ID\_SC\_ID,ID\_ID\_SC  
 PO\_ID\_ID,SU\_ID\_ID,OT\_ID\_ID  
 ID\_ID\_ID  
 PO\_OT\_WC,OT\_WC\_PO,WC\_PO\_OT  
 SU\_OT\_WC,OT\_WC\_SU,WC\_SU\_OT  
 PO\_OT\_SC,OT\_SC\_PO,SC\_PO\_OT  
 SU\_OT\_SC,OT\_SC\_SU,SC\_SU\_OT  
 SU\_OT\_PO,OT\_PO\_SU,PO\_SU\_OT  
 /;

Remark: WC: winter cereals, SC: summer cereals, PO: potatoes, SU: sugar beets, ID: idling land, OT: other

The *rotations* are linked to groups of crops in the first, second and third year of the rotation as can be seen in the following equation (only cross-set definitions *rot\_cropTypes* for the first rotation are shown).

set rot\_cropTypes(rot,cropTypes,cropTypes,cropTypes) "Rotation, first / second / third year crop type"  
 /  
 WC\_WC\_PO.WinterCere.WinterCere.potatoes  
 WC\_PO\_WC.WinterCere.potatoes.WinterCere  
 PO\_WC\_WC.potatoes.WinterCere.WinterCere  
  
 WC\_WC\_OT.WinterCere.WinterCere.other  
 WC\_OT\_WC.WinterCere.other.WinterCere  
 OT\_WC\_WC.other.WinterCere.WinterCere  
  
 WC\_WC\_ID.WinterCere.WinterCere.idle  
 WC\_ID\_WC.WinterCere.idle.WinterCere  
 ID\_WC\_WC.idle.WinterCere.WinterCere  
  
 WC\_WC\_SU.WinterCere.WinterCere.sugarBeet  
 WC\_SU\_WC.WinterCere.sugarBeet.WinterCere  
 SU\_WC\_WC.sugarBeet.WinterCere.WinterCere  
  
 WC\_WC\_SC.WinterCere.WinterCere.summerCere  
 WC\_SC\_WC.WinterCere.summerCere.WinterCere  
 SC\_WC\_WC.summerCere.WinterCere.WinterCere  
  
 WC\_SC\_PO.WinterCere.summerCere.potatoes  
 SC\_PO\_WC.summerCere.potatoes.WinterCere  
 PO\_WC\_SC.potatoes.WinterCere.summerCere  
  
 WC\_SC\_SU.WinterCere.summerCere.sugarBeet  
 SC\_SU\_WC.summerCere.sugarBeet.WinterCere  
 SU\_WC\_SC.sugarBeet.WinterCere.summerCere  
  
 WC\_SC\_ID.WinterCere.summerCere.idle  
 SC\_ID\_WC.summerCere.idle.WinterCere  
 ID\_WC\_SC.idle.WinterCere.summerCere  
  
 WC\_SC\_OT.WinterCere.summerCere.other  
 SC\_OT\_WC.summerCere.other.WinterCere  
 OT\_WC\_SC.other.WinterCere.summerCere  
  
  
 SC\_WC\_SC.summerCere.WinterCere.summerCere  
 WC\_SC\_SC.WinterCere.summerCere.summerCere  
 SC\_SC\_WC.summerCere.summerCere.WinterCere  
  
 WC\_ID\_ID.WinterCere.idle.idle  
 ID\_WC\_ID.idle.WinterCere.idle  
 ID\_ID\_WC.idle.idle.WinterCere  
  
 SC\_ID\_ID.summerCere.idle.idle  
 ID\_SC\_ID.idle.summerCere.idle  
 ID\_ID\_SC.idle.idle.summerCere  
  
 SC\_SC\_ID.summerCere.summerCere.idle  
 SC\_ID\_SC.summerCere.idle.summerCere  
 ID\_SC\_SC.idle.summerCere.summerCere  
  
 SC\_SC\_PO.summerCere.summerCere.potatoes  
 WC\_PO\_ID.WinterCere.potatoes.idle  
 SC\_PO\_ID.summerCere.potatoes.idle  
 ID\_ID\_ID.idle.idle.idle  
 PO\_ID\_ID.potatoes.idle.idle  
  
 SC\_SC\_SU.summerCere.summerCere.sugarBeet  
 WC\_SU\_ID.WinterCere.sugarBeet.idle  
 SC\_SU\_ID.summerCere.SugarBeet.idle  
 SU\_ID\_ID.sugarBeet.idle.idle  
  
  
  
 SC\_SC\_OT.summerCere.summerCere.other  
 WC\_OT\_ID.WinterCere.other.idle  
 SC\_OT\_ID.summerCere.other.idle  
 OT\_ID\_ID.other.idle.idle  
  
 PO\_OT\_WC.potatoes.other.WinterCere  
 OT\_WC\_PO.other.WinterCere.potatoes  
 WC\_PO\_OT.WinterCere.potatoes.other  
  
 PO\_OT\_SC.potatoes.other.summerCere  
 SU\_OT\_WC.SugarBeet.other.WinterCere  
 OT\_WC\_SU.other.WinterCere.SugarBeet  
 WC\_SU\_OT.WinterCere.SugarBeet.other  
  
 SU\_OT\_SC.SugarBeet.other.summerCere  
 OT\_SC\_SU.other.summerCere.SugarBeet  
 SC\_SU\_OT.summerCere.SugarBeet.other  
  
 SU\_OT\_PO.SugarBeet.other.potatoes  
 OT\_PO\_SU.other.potatoes.SugarBeet  
 PO\_SU\_OT.potatoes.SugarBeet.other  
  
 /;

The link between individual crops and crop types used in the rotation definitions is as follows:

set cropTypes\_crops(cropTypes,crops) / winterCere.(winterWheat,winterBarley)  
 summerCere.(summerCere,maizCorn,maizCCM,WheatGPS)  
 other.(winterrape,summerBeans,summerPeas,CatchCrop)  
 potatoes.potatoes  
 sugarbeet.sugarbeet  
 idle.idle  
 /;

In order to use the crop rotations in the model equations, three cross sets are generated which define the crop type in the first, second and third year for each rotation:

set cropType0\_rot(cropTypes,rot);cropType0\_rot(cropTypes,rot) $ sum(rot\_cropTypes(rot,cropTypes,cropTypes1,cropTypes2),1) = YES;

set cropType1\_rot(cropTypes,rot);cropType1\_rot(cropTypes,rot) $ sum(rot\_cropTypes(rot,cropTypes1,cropTypes,cropTypes2),1) = YES;

set cropType2\_rot(cropTypes,rot);cropType2\_rot(cropTypes,rot) $ sum(rot\_cropTypes(rot,cropTypes1,cropTypes2,cropTypes),1) = YES;

For each simulation, crops can be selected to be available for cropping on farm, therefore, it is possible that not all rotations are operational. Accordingly, in *coeffgen\coeffgen.gms*, the set of available crop rotations is defined:

cropType0\_rot(cropTypes,rot) $ (not sum( (cropType0\_rot(cropTypes1,rot),curCrops) $ cropTypes\_crops(cropTypes1,curCrops),1)) = NO;

cropType0\_rot(cropTypes,rot) $ (not sum( (cropType1\_rot(cropTypes1,rot),curCrops) $ cropTypes\_crops(cropTypes1,curCrops),1)) = NO;

cropType0\_rot(cropTypes,rot) $ (not sum( (cropType2\_rot(cropTypes1,rot),curCrops) $ cropTypes\_crops(cropTypes1,curCrops),1)) = NO;

cropType1\_rot(cropTypes,rot) $ (not sum( (cropType1\_rot(cropTypes1,rot),curCrops) $ cropTypes\_crops(cropTypes1,curCrops),1)) = NO;

cropType1\_rot(cropTypes,rot) $ (not sum( (cropType0\_rot(cropTypes1,rot),curCrops) $ cropTypes\_crops(cropTypes1,curCrops),1)) = NO;

cropType1\_rot(cropTypes,rot) $ (not sum( (cropType2\_rot(cropTypes1,rot),curCrops) $ cropTypes\_crops(cropTypes1,curCrops),1)) = NO;

cropType2\_rot(cropTypes,rot) $ (not sum( (cropType2\_rot(cropTypes1,rot),curCrops) $ cropTypes\_crops(cropTypes1,curCrops),1)) = NO;

cropType2\_rot(cropTypes,rot) $ (not sum( (cropType0\_rot(cropTypes1,rot),curCrops) $ cropTypes\_crops(cropTypes1,curCrops),1)) = NO;

cropType2\_rot(cropTypes,rot) $ (not sum( (cropType1\_rot(cropTypes1,rot),curCrops) $ cropTypes\_crops(cropTypes1,curCrops),1)) = NO;

The rotations enter the model via three constraints (*see model\templ.gms*). The right hand side sums up the crop hectares of a certain crop type in the current year in all three constraints, while the left hand side exhausts these hectares in the current, next and year after next year based on the rotations grown in these years.

rotHa0\_(cropTypes,plot,tCur(t),nCur) $ ( (not sum(plot\_lt\_soil(plot,"gras",soil),1) $ t\_n(t,nCur))  
  
 $ (sum(cropType0\_rot(cropTypes,curRot(rot)),1)  
 $ sum( (cropTypes\_crops(cropTypes,crops),c\_s\_t\_i(crops,plot,till,intens))  
 $ (v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0),1))) ..  
  
 sum( (cropTypes\_crops(cropTypes,crops),c\_s\_t\_i(crops,plot,till,intens)), v\_cropHa(crops,plot,till,intens,t,nCur))  
  
 =E= sum(cropType0\_rot(cropTypes,curRot(rot)), v\_rotHa(rot,plot,t,nCur));

rotHa1\_(cropTypes,plot,tCur(t),nCur) $ ((not sum(plot\_lt\_soil(plot,"gras",soil),1) )  
  
 $ (sum(cropType1\_rot(cropTypes,curRot(rot)),1)  
 $ sum( (cropTypes\_crops(cropTypes,crops),c\_s\_t\_i(crops,plot,till,intens))  
 $ (v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0),1)  
 $ tCur(t+1)) $ t\_n(t,nCur) ) ..  
  
 sum( (cropTypes\_crops(cropTypes,crops),c\_s\_t\_i(crops,plot,till,intens)), v\_cropHa(crops,plot,till,intens,t,nCur))  
  
 =E= sum((cropType1\_rot(cropTypes,curRot(rot)),t\_n(t+1,nCur1)), v\_rotHa(rot,plot,t+1,nCur1));

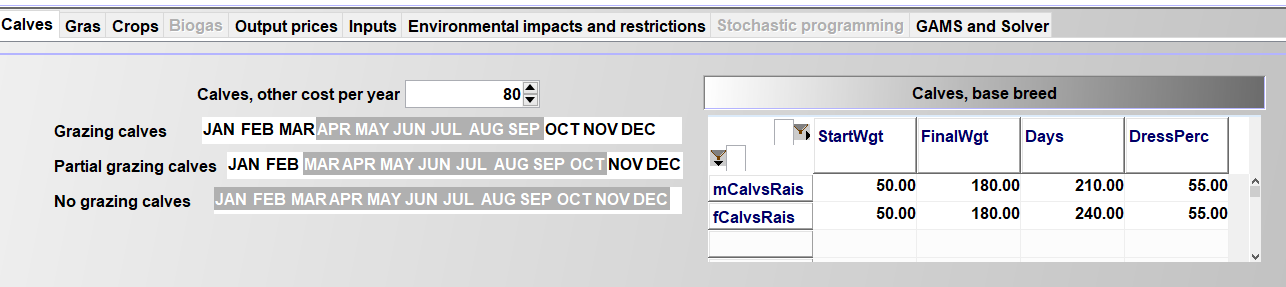
rotHa2\_(cropTypes,plot,tCur(t),nCur) $ ((not sum(plot\_lt\_soil(plot,"gras",soil),1))  
  
 $ (sum(cropType2\_rot(cropTypes,curRot(rot)),1)  
 $ sum( (cropTypes\_crops(cropTypes,crops),c\_s\_t\_i(crops,plot,till,intens))  
 $ (v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0),1)  
 $ tCur(t+2)) $ t\_n(t,nCur) ) ..  
  
 sum( (cropTypes\_crops(cropTypes,crops),c\_s\_t\_i(crops,plot,till,intens)), v\_cropHa(crops,plot,till,intens,t,nCur))  
  
 =E= sum((cropType2\_rot(cropTypes,curRot(rot)),t\_n(t+2,nCur1)), v\_rotHa(rot,plot,t+2,nCur1));

The rotations restrict the combination of crops and enter into the optional soil pool balancing approach.

## Grassland management

Grassland management was so far based on two type of pasture (past22 and past33) and three types of gras for silage (gras20, gras24 and gras33). The names were originally referring to annual fresh weigth yields, but actual yields could be changed via the interface. Content were fixed. The approach was completely overhauled as described in the following.

### Grazing of herds

For cows, bulls, heifers and calves, the user can define on the interface if no grazing (= all day long in stable), partial grazing (= half day in stable) or full grazing (= no time in stable) can be used. Figure XY illustrates the grazing of calves as specified in the GUI. The entries would imply that calves have to kept in stable during JAN,FEB,NOV,DEC as no other option is open, partial grazing is additionally possible in MAR and OCTOBER, and during the period APR-SEP, the farmer has the choice of all three types. 

The labour needs differ between the three options:

p\_herdLab("heifs","noGraz",m) = 9 / card(m);

p\_herdLab("heifs","partGraz",m) = 9 / card(m) + 0.5;

p\_herdLab("heifs","fullGraz",m) = 5 / card(m);

The additional work load for partial grazing is calculated as follows: It is assumed that it takes one hour a day to move the herd form the stable to the pasture and back (= 15 hours in total in a month). The average herd driven is assumed to be equal to 60 animals for cows and 30 animals for heifers/bulls. For calves, which are assumed to be driven with other herds, 0.25 hours a month are added. The introduction of these grazing feeding regimes (part of the set “feedRegime”) requires a change in the logic of the program. The *v\_herdStart* variable now is no longer indexed with the feed regime – reflecting that e.g. a heifer entering the cow herd might during its lifetime as a cow sometimes be grazed and sometimes not. The *herdsSize\_* equation (below an incomplete screen shot, see \*model\_herd\_module\*) equilibrates the herd sizes in the different months (LHS) to the herds starting in the yearly and months before:

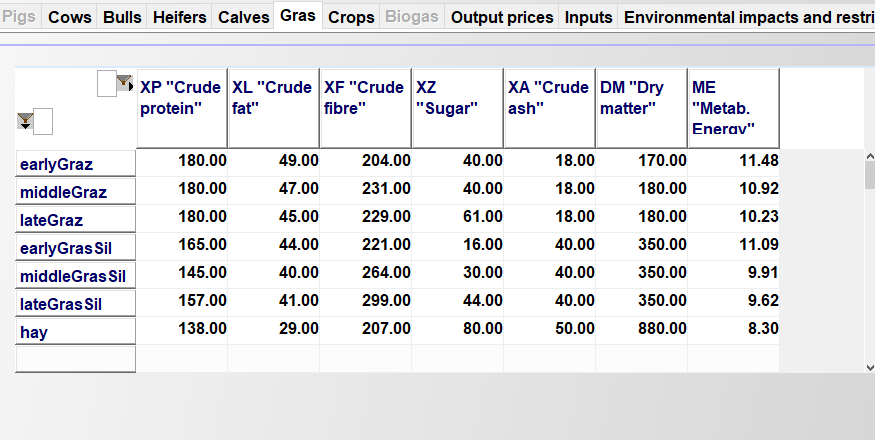
sum(feedRegime $ actHerds(herds,breeds,feedRegime,t,m),v\_herdSize(herds,breeds,feedRegime,t,nCur,m))  
  
 =E=  
\*  
\* --- herds which started in the months before the production length, in case for piglets a separate construct is used  
\*  
  
 + sum( (t\_n(t1,nCur1),m1) $ (  
 (( (-p\_mDist(t,m,t1,m1) le (p\_prodLength(herds,breeds)-1))  
 $ (p\_mDist(t,m,t1,m1) le 0))  
  
 or  
 ( (abs(p\_mDist(t,m,t1,m1)-12) le (p\_prodLength(herds,breeds)-1))  
 $ (p\_mDist(t,m,t1,m1)-12 le 0)) $ p\_compStatHerd  
 )  
 $ sum(feedRegime,actHerds(herds,breeds,feedRegime,t1,m1)) $ isNodeBefore(nCur,nCur1)  
 $$iftheni.sows "%farmBranchSows%" == "on"  
 $(not sameas(herds,"piglets"))  
 $$endif.sows  
 ),  
 v\_herdStart(herds,breeds,t1,nCur1,m1)  
  
  
 $$iftheni.ch %cowHerd%==true  
\*  
\* --- minus, in case of cows, slaughtered before reaching the final age  
\*  
 -sum( (slgtCows,cows) $ (sum(feedRegime, actHerds(slgtCows,breeds,feedRegime,t1,m1))  
 $ sameas(cows,herds) $ (slgtCows.pos eq cows.pos)),  
 v\_herdStart(slgtCows,breeds,t1,nCur1,m1))  
 $$endif.ch  
 )  
\*  
\* --- Herd size dynamic for piglets separately to depict a correct transfer from year t to year t1 as well as account for temporal resolution adjustments  
\*  
  
 $$iftheni.sows "%farmBranchSows%" == "on"  
 + sum( (t\_n(t1,nCur1),m1) $ ( (abs(p\_mDist(t,m,t1,m1)) le (p\_prodLengthB(herds,breeds) -1 $ (p\_prodLengthB(herds,breeds) eq 1)))  
 $ (p\_mDist(t,m,t1,m1) le 0) $ isNodeBefore(nCur,nCur1)  
 $ sum(feedRegime,actHerds(herds,breeds,feedRegime,t1,m1))  
 $ (not sameas(herds,"sows"))  
 ${ ( sameas(t,t1) $ (not sameas(m - p\_prodLengthB(herds,breeds),m1)))  
 or ((not sameas(t,t1)) $ (sameas("Jan",m))$ (sameas( m + 11, m1)))}),  
  
 v\_herdStart(herds,"",t1,nCur1,m1))  
 $$endif.sows  
 ;

An additional equation (see \*model\_module.gms\*) ensures that the feeding phase variable is linked to herd in a specific feed regime:

herdsByFeedRegime\_(herds,breeds,feedRegime,t,n,m) "Distribute herds to feed regimes"  
 herdsreqsPhase\_(herds,breeds,reqsPhase,m,t,n) "Animal herds in a certain phase in a certain month"  
 reqsPhase\_(herds,breeds,feedRegime,reqs,reqsPhase,m,t,n) "Animal requirements need to be covered"  
 sumReqs\_(reqs,t,n) "Total requirements per year"  
 sumReqsBought\_(reqs,t,n) "Total requirements per year from bought feed"  
 feedUse\_(feeds,t,n) "Definition of total feed use"  
 feedUseHerds\_(herds,feeds,t,n) "Definition of total feed use"  
 feedUseM\_(feeds,m,t,n) "Definition of total feed use"  
 prodsM\_(prods,m,t,n) "Monthly feed use definition"  
 herdsBefore\_(herds,breeds,feedRegime,t,t,n,m) "First two years"  
 herdsStartBefore\_(herds,breeds,t,t,n,m) "First two years"  
 sumHerds\_(sumHerds,breeds,feedRegime,t,n,m) "Summary herd definition, per month"  
 sumHerdsY1\_(sumHerds,breeds,t,n) "Summary herd definition, per year, sold herds"  
 avgLactations\_(breeds,t,n) "Recover average lactation length from short and long"  
 herdsLast\_(herds,breeds,feedRegime,t,n,m) "Prolongate herd size"  
 herdsLast1\_(herds,breeds,feedRegime,t,n,m) "Prolongate herd size"  
 maxHerdChange1\_(herds,breeds,feedRegime,t,n,n) "Special restricton for heifer and calves raisingherd"  
 maxHerdChange2\_(herds,breeds,feedRegime,t,n,n) "Special restricton for heifer and calves raisingherd"  
 hasHerdOrderDairy\_(t,n)  
 hasHerdOrderMotherCows\_(t,n)  
  
$iftheni.dh %cowherd%==true  
 newCalves\_(breeds,t,n,m) "Born calves (male and female)"  
 maleFemaleRel\_(breeds,t,n,m) "Born calves, keep male/female relation"  
$endif.dh  
  
 nutExcrPast\_(allNut,t,n,m) "N and P excretion on pasture"  
 nut2ManurePast\_(nut2,t,n,m)  
 FixGrasLand\_(t,n) "Ensures that there is no grassland on arable land"  
 FixPastLand\_(t,n) "Distribution of gras and past land on total land "  
  
;

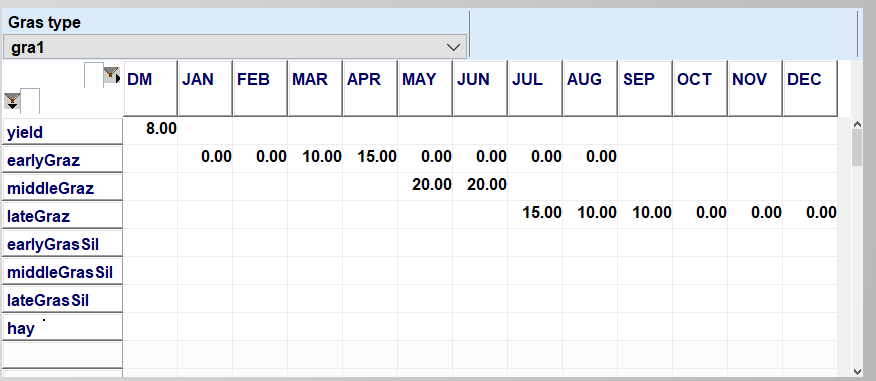
### Nutrient content of different grassland outputs

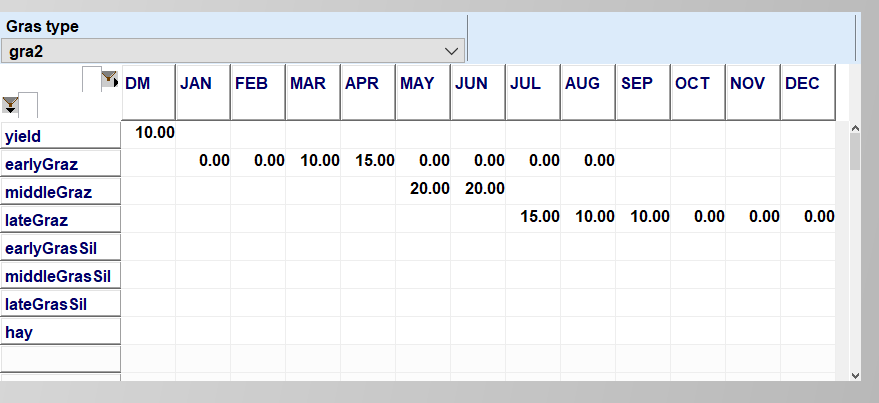
The model now supports three types of fresh gras (labelled early – middle – late), three types of gras silage (labelled early – middle – late) and hay based on their feed attributes per unit of dry matter. Dry matter content is inputted as well:



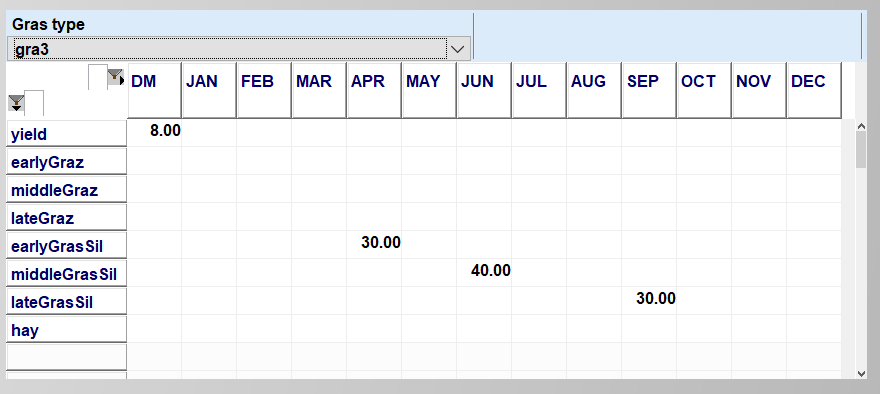
### Grassland management

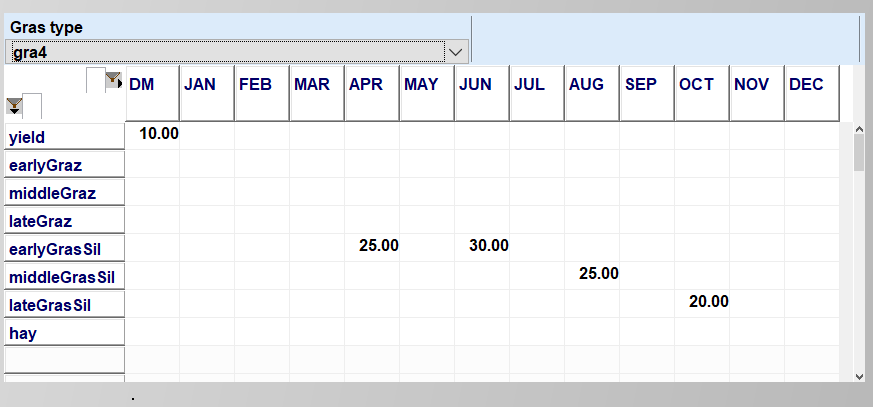
The user can define up to 10 different types of grassland management by the following two attributes: 1. Total dry matter output 2. Distribution of outputs (see above) over months The default setting defines two differently intensive grazing schemes, which only differ in dry matter output:





Two different silage use schemes:





Embedded phyton code (see *util.gms*) introduces more easily interpretable labels for reporting:

xx

After the original label follows the annual dry matter yield, followed by the number of cuts (where applicable) and the share of biomass used for grazing resp. silage or hay. The assignment of machinery needs and (related) labour hours in defined in “*coeffgen.gms*” and “*coeffgen.gms*”:

\* --- definition of basic field operations for graslands  
\*  
 set.gras . soilSample . SEP2 0.25 0.25 0.25  
 set.gras . weederlight . MAR2 0.25 0.25 0.25  
 set.gras . sowMachine . MAR2 0.75 0.75 0.75  
 set.gras . grasReSeeding . APR1 0.25 0.25 0.25  
 set.gras . roller . APR1 0.25 0.25 0.25  
 ;

The operations shown above occur on all type of grasslands in the given frequency. The operations related to cuts are defined as follows:

\* --- definition of cuts for grasland  
\*  
  
set toSilage(noPastOutputs) / earlyGrasSil,middleGrasSil,lateGrasSil /;  
set toHay(noPastOutputs) / hay /;  
  
 set grasToOutput(crops,grasOutputs);  
 grasToOutput(crops,grasOutputs) $ sum((m) $(p\_grasAttr(crops,grasOutputs,m)), 1) = YES;  
  
 table p\_opPerCut(operation,noPastOutputs,till) "Field operations for one gras cut per cutting process (either silo or bales)"  
  
 silo bales  
 mowing.set.noPastOutputs 1.00 1.00  
 tedding.set.noPastOutputs 1.00 1.00  
 raking.set.noPastOutputs 1.00 1.00  
\*  
\* --- these operations are changed by harvested biomas  
\*  
 closeSilo.middleGrasSil 1.00  
 silageTrailer.middleGrasSil 1.00  
 balePressWrap.middleGrasSil 1.00  
 balePressHay.hay 1.00  
 baleTransportSil.hay  
 baleTransportHay.hay 1.00  
 ;  
  
  
parameter p\_bioMassOpsFac(operation) "Factor in order to correct dry matter content to witted silage content (35% DM) or hay (86% DM)"  
 /  
 silageTrailer 0.35  
 balePressWrap 0.35  
 baleTransportSil 0.35  
 balePressHay 0.86  
 baleTransportHay 0.86

And are used to define the machinery needs:

\* --- count lab period where gras is cut  
\*  
parameter p\_cutPeriod(crops,\*) "Count # of labour period where grass is cut";  
  
p\_cutPeriod(gras,m)  
 = sum( (labPeriod\_to\_month(labPeriod,m),noPastOutputs) $ p\_grasAttr(gras,noPastOutputs,m),1);  
  
p\_cutPeriod(gras,labPeriod) = sum( labPeriod\_to\_month(labPeriod,m),p\_cutPeriod(gras,m));  
\*  
\* --- silo cut for silage  
\*  
 crop\_op\_per\_till(gras,operation,labPeriod,"silo")  
 $ (p\_cutPeriod(gras,labPeriod) $ p\_opPerCut(operation,"middleGrasSil","silo") $ grasToOutput(gras,"middleGrasSil"))  
 = sum( (labPeriod\_to\_month(labPeriod,m),toSilage) $ p\_grasAttr(gras,toSilage,m),  
 p\_opPerCut(operation,"middleGrasSil","silo")  
\*  
\* --- change machinery needs (or not) depending on harvested dry matter  
\*  
 \* ( 1 $ (not p\_bioMassOpsFac(operation))  
 + (p\_grasAttr(gras,toSilage,m)/p\_bioMassOpsFac(operation)/op\_attr(operation,"67kw","2ha","amount")) $ p\_bioMassOpsFac(operation))  
 )/ p\_cutPeriod(gras,labPeriod);

# Labour

!!! abstract The labour module optimises work use on- and off-farm with a monthly resolution, depicting detailed labour needs for different farm operations, herds and stables, management requirements for each farm branch, and the farm as a whole. Off-farm work distinguishes between half- and full-time work (binaries) and working flexibly for a low wage rate.

## General Concept

The template differentiates between three types of labour on farm:

1. **General management and further activities for the whole farm,** *p\_labManag("farm","const"*), which are needed as long as the farm is not abandoned ,*v\_hasFarm* = 1, *binary variable*, and not depending on the level of individual farm activities.
2. **Management activities and further activities depending on the size of farm branches** such as arable cropping, dairying, beef fattening, pig fattening, and piglet production. The necessary working hours are broken down into a base need, *const* which is linked to having the respective farm branch, *v\_hasBranch*, *integer*, and a linear term depending on its size, *slope*.
3. **Labour needs for certain farm operations** (aggregated to *v\_totLab*).

The sum of total labour needs cannot exceed total yearly available labour (see following equation). As discussed below, there are further restrictions with regard to monthly labour and available field working days.

LabTot\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 sum(m, v\_labTot(t,nCur,m)) =L= p\_yearlyLabH(t);

The maximal yearly working hours, *p\_yearlyLabH,* are defined in the statement shown below. The maximal labour hours for the first, second and further labour units can be entered via the GUI.

p\_yearlyLabH(t) = %AkhFirst% \* min(1,%Aks%)  
 + %AkhSecond% \* min(1,%Aks%-1) $ (%Aks% > 1)  
 + %AkhFurther% \* (%Aks%-2) $ (%Aks% > 2);

The maximaum work hours per month is defined in the following statement, represented by the parameter *p\_monthlyLabH*:

p\_monthlyLabH(t,m) = max(p\_yearlyLabH(t) / 365 \* p\_daysPerMonth(m)\*1.2, %Aks% \* 12 \* p\_daysPerMonth(m) \* 5/7);

The template considers the sum of labour needs for each month, *m,* and each SON, *s*. Farm labour needs are related to certain farm activities on field and in stable. The labour need for work on-farm and flexibly off-farm is defined by the following equation. The variables that enter in the equation are explained in the next section of the labour section.

labTotSM\_(tCur(t),nCur,m) $ t\_n(t,nCur) ..  
\*  
\* --- sum of work in hours in current month  
\*  
 v\_LabTot(t,nCur,m) =e=  
\*  
\* --- leisure time  
\*  
 v\_leisure(t,nCur,m)  
\*  
\* --- labour use for crops and herds  
\*  
 + v\_labCropSM(t,nCur,m)  
  
$ifi %herd%==true + v\_labHerdM(t,nCur,m)  
\*  
\* --- Management  
\*  
 + v\_labManag(t,nCur)/card(m)  
\*  
\* --- off farm labour - per month: p\_workTime are weekly hours,  
\* p\_commTime is the commuting time in weekly hours, assumption of  
\* 46 weeks work in each year (binary variables)  
\*  
 + v\_labOffFixed(t,nCur)/card(m)  
\*  
\* --- small scale work on a hourly basis (continous)  
\*  
 + v\_labOffHourly(t,nCur)  
\*  
\* --- labour use for biogas plant  
\*  
$ifi %biogas%== true + sum((curBhkw(bhkw)), v\_labBioGas(bhkw,t,nCur,m))  
 ;

## Labour Need for Farm Branches

FarmDyn comprises currently five different farm branches: cropping, cattle, fatteners, sows and biogas. The (management) labour needs for the biogas branch is accounted for in the biogas module. For the other branches, their size *v\_branchSize*, is endogenously defined from activity levels mapped to it:

branchSize\_(branches,tCur(t),nCur) $ ( (sum(branches\_to\_acts(branches,possActs) ,1) $ t\_n(t,nCur))  
 $$ifi %biogas%==true or sum(sameas(branches,"biogas"),1) $ t\_n(t,nCur)  
 ) ..  
  
 v\_branchSize(branches,t,nCur) =E= sum((branches\_to\_acts(branches,curCrops(crops)),plot,till,intens)  
 $( c\_s\_t\_i(crops,plot,till,intens)$( not sameas (crops,"catchCrop"))),  
 v\_cropHa(crops,plot,till,intens,t,nCur))  
  
$iftheni %herd% == true  
 + sum( (branches\_to\_acts(branches,possHerds),breeds,feedRegime,m)  
 $ (actHerds(possHerds,breeds,feedRegime,t,m) $ p\_prodLength(possHerds,breeds)),  
 v\_herdSize(possHerds,breeds,feedRegime,t,nCur,m)  
 \* 1/min(12,p\_prodLength(possHerds,breeds))  
 \* ( (12/card(herdM)) $ (not sameas(branches,"fatPig")) + 1 $ sameas(branches,"fatPig"))  
 )  
$endif  
  
$iftheni %biogas% == true  
 + [sum( (curBhkw,curEeg,t\_n(tCur,nCur),m), v\_prodElec(Curbhkw,curEeg,tCur,nCur,m))/100000]  
 $ sameas(branches,"biogas")  
$endif  
 ;

Where the cross-set, *branches\_to\_acts,* defines which activities count to a certain branch:

set branches\_to\_acts(branches,acts) /  
 cashCrops.(winterWheat,winterBarley,summerCere,winterRape,summerBeans,summerPeas,  
 MaizCorn,potatoes,sugarBeet,MaizCCM,  
 $$ifthen.cattle not %cattle%==true  
 MaizSil,WheatGPS,  
 $$endif.cattle  
 CatchCrop)  
 dairy.cows  
 motherCows.motherCow  
 sowPig.sows  
 fatPig.fattners  
 beef.bulls  
 /;

The binary variable *v\_hasBranch* which relates to the general management need for branch is triggered as follows:

hasBranch\_(branches,tCur(t),nCur) $ (sum(branches\_to\_acts(branches,acts) ,1) $ t\_n(t,nCur)  
 $$ifi %biogas%==true or sum(sameas(branches,"biogas"),1) $ t\_n(t,nCur)  
 ) ..  
 v\_branchSize(branches,t,nCur) =l= v\_hasBranch(branches,t,nCur) \* p\_maxBranch(branches);

The *hasFarm* trigger depends on the trigger for the individual branches:

hasFarm\_(branches,tCur(t),nCur) $ ((not sameas(branches,"farm")) $ (v\_hasBranch.range(branches,t,nCur) ne 0) $ t\_n(t,nCur)) ..  
  
 v\_hasBranch(branches,t,nCur) =l= v\_hasFarm(t,nCur);

The hours needed for yearly farm management are defined using a constant and the branch specific values:

labManag\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_labManag(t,nCur) =e=  
\*  
\* -- two hundredth hours independent from number of branches or farm size  
\*  
 + v\_hasFarm(t,nCur) \* p\_labManag("Farm","const")  
  
 + sum(branches $ sum(branches\_to\_acts(branches,acts), 1),  
 v\_hasBranch(branches,t,nCur) \* p\_labManag(branches,"const")  
 + v\_branchSize(branches,t,nCur) \* p\_labManag(branches,"slope"));

## Labour Need for Herd, Cropping, Operations and Off-Farm Work

### Herd Activities and Cropping

The labour need for animals, *v\_herdLabM,* is defined by an animal type specific requirement parameter, *p\_herdLab,* in hours per animal and month (see in the next equation, working hours per animal and month) and by the time requirement per stable place, which differs with the stable type. This formulation allows labour saving scale effects related to the stable size:

labHerdM\_(tCur(t),nCur,m) $ t\_n(t,nCur) ..  
 v\_labHerdM(t,nCur,m) =e=  
\*  
\* --- labour for animal activities, expressed per animal and month  
\* of standing herd  
\*  
 sum(actHerds(sumHerds,breeds,feedRegime,t,m1) $ m\_to\_herdm(m,m1),  
 v\_herdSize(sumHerds,breeds,feedRegime,t,nCur,m1) \* p\_herdLab(sumHerds,feedRegime,m))  
\*  
\* --- labour for animal activities, per starting animal (hours for giving birth and similar)  
\*  
 + sum( (sumHerds,breeds,m1) $ (sum(feedRegime, actHerds(sumHerds,breeds,feedRegime,t,m1)) $ m\_to\_herdm(m,m1)),  
 v\_herdStart(sumHerds,breeds,t,nCur,m1)\* p\_herdLabStart(sumHerds,m))  
\*  
\* --- fixed amount of hours for stables (maintenance, cleansing),  
\* captures also labour saving effects of large stables  
\*  
 + sum(stables $ v\_stableInv.up(stables,"long",t,nCur),  
 v\_stableShareCost(stables,t,nCur) \* p\_stableLab(stables,m) );

A similar equation exists for crops. The parameter *p\_cropLab* defines the labour hours per hectare and month for each crop. In addition, the parameters *p\_manDistLab* and *p\_syntDistLab* multiplied by the *N type* applied to each crop are added to the overall crop labour demand for the application of synthetic fertiliser and manure:

labCropSM\_(tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_labCropSM(t,nCur,m) =e=  
\*  
\* --- labour need for crops, expressed per ha of land  
\* (will probably change to specific acticities later)  
\*  
 sum( c\_s\_t\_i(curCrops(crops),plot,till,intens),  
 v\_cropHa(crops,plot,till,intens,t,nCur) \* p\_cropLab(crops,till,intens,m))  
  
\* --- labour need for application of N (fertilizer and manure N)  
  
$iftheni.man %manure% == true  
 + sum((c\_s\_t\_i(curCrops(crops),plot,till,intens),manApplicType\_manType(ManApplicType,curManType))  
 $ ((v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0)  
 $ ( v\_manDist.up(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m) ne 0)),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m) \* p\_manDistLab(ManApplicType))  
$endif.man  
  
 + sum((c\_s\_t\_i(crops,plot,till,intens),syntFertilizer),  
 v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m) \* p\_syntDistLab(syntFertilizer));

### Farm Operations

Field working days define the number of days available in a labour period of half a month, *labPeriod,* during which soil conditions allow specific types of operations, *labReqLevl*:

fieldWorkHours\_(plot,labReqLevl,labPerSum,tCur(t),nCur)  
 $ (p\_plotSize(plot) $ plot\_landType(plot,"arab") $ t\_n(t,nCur) ) ..  
  
 v\_fieldWorkHours(plot,labReqLevl,labPerSum,t,nCur)  
  
 =e=  
\*  
 sum(labPerSum\_ori(labPerSum,LabPeriod),  
\*  
\* --- operations requiring a tractor, with the exemption top of  
\* fertilizer dsitribution  
\*  
 sum( c\_s\_t\_i(curCrops(crops),plot,till,intens),  
 v\_cropHa(crops,plot,till,intens,t,nCur)  
 \* p\_fieldWorkHourNeed(crops,till,intens,labPeriod,labReqLevl)  
\*  
\* --- distribution of synthetic fertilizer  
  
 + sum( (syntFertilizer,labPeriod\_to\_month(labPeriod,m)),  
 v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m)  
 \* p\_machNeed(syntFertilizer,"plough","normal","tractor","hour") ) \* sameas(labReqLevl,"rf3")  
 )  
 );

The number of field working hours cannot exceed a limit which is defined by the available field working days, *p\_fieldWorkingDays.* Field working days depend on climate zone, soil type (*light, middle, heavy*) and distribution of available tractors to the soil type, *v\_tracDist*. It is assumed that farm staff will be willing to work up to 15 hours a day, still with the total work load per month being restricted:

tracRestrFieldWorkHours\_(plot,labReqLevl,labPerSum,tCur(t),nCur)  
 $ (p\_plotSize(plot) $ plot\_landType(plot,"arab") $ t\_n(t,nCur)) ..  
  
 v\_fieldWorkHours(plot,labReqLevl,labPerSum,t,nCur)  
  
 =L=  
 sum(labPerSum\_ori(labPerSum,LabPeriod),  
 sum(plot\_soil(plot,soil),  
 sum(curClimateZone, p\_fieldWorkingDays(labReqLevl,labPeriod,curClimateZone,soil)) \* 12)  
 \* v\_tracDist(plot,labPerSum,t,nCur));

Furthermore, the distribution of tractors is determined endogenously:

tracDistribution\_(labPerSum,tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 sum(plot $ p\_plotSize(plot), v\_tracDist(plot,labPerSum,t,nCur)) =L= ceil(%Aks%);

It implicitly assumes that farm family members are willing to spend hours for on-farm work even if working off-farm, e.g. by taking days off.

### Off-Farm Work

Farm family members can optionally work half- or full-time, *v\_workoff*, or on an hourly basis off-farm, *v\_workHourly*. Half- and full-time work are realised as integer variables. In the normal setting the wage per hour for working half time exceeds the wage of short time hourly work. Moreover, the wage per hour of full time work is higher than of working half time. For half- and full-time work commuting time can be considered:

offFarmHoursPerYearFixed\_(tCur(t),nCur) $ (t\_n(t,nCur) $ sum(workOpps(workType), v\_labOff.up(t,nCur,workType))) ..  
  
 v\_labOffFixed(t,nCur) =e=  
\*  
\* --- off farm labour - per month: does not fit with the actual hours worked,  
\* but assumes the actual willingness to work on farm  
\* is reduced (typically farm more compared to what is worked!)  
  
 + sum( workOpps(workType),  
 v\_labOff(t,nCur,workType) \* p\_workTimeLost(workType));

The set *workType* lists the possible combinations:

p\_workTime(workType) = (p\_workT("Half")+p\_workT("Full")\*floor(workType.pos/2)) $ ( mod(workType.pos,2) eq 1)  
 + p\_workT("Full")\*(workType.pos/2 ) $ ( mod(workType.pos,2) eq 0);

It is assumed that decisions about how much to work flexibly on an hourly basis are taken on a yearly basis (i.e. the same number of hours is inputted in each month).

The total number of hours worked off-farm is defined as:

offFarmWorkTot\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
\*  
 v\_labOffTot(t,nCur) =e=  
  
 v\_labOffHourly(t,nCur) \* card(m)  
  
 + v\_labOffFixed(t,nCur) $ sum(workOpps(workType), v\_labOff.up(t,nCur,workType));

# Stables

Different types of stables are implemented in FarmDyn. In general, stables are differentiated between calf, cow, mother cow, young cattle (including beef fattening), piglet, fattener, and sow stables. These stables are available in different sizes. The requirement of an animal for a certain stable type is defined by the parameter *p\_stableNeed*. Through this approach, it is possible for some herds to share a stables place, e.g. calves and heifers, or bulls and heifers by assigning their stable need to the same stable type. Stable types can further be specified through their manure management system, depicted by the set *stableStyle*. As a default, all stable types will employ slatted floor manure handling, resulting in only liquid manure handling on the farm. Choosing a straw stable for a herd will consequently affect costs, manure handling, and labour requirement.

The stable inventory (*v\_stableInv*) for each type of stable, (*stables*) is defined as seen in the following equation *stableInv\_*. *p\_iniStables* is the initial endowment of stables in the construction year, *p\_lifeTimeS* is the maximal physical lifetime of the stables, and *v\_buyStables* are newly constructed stables.

stableInv\_(stables,hor,tFull(t),nCur)  
 $ ( (p\_priceStables(stables,hor,t) gt eps)  
 $ ( sum( t\_n(t1,nCur1) $ (isNodeBefore(nCur,nCur1) or sameas(nCur,nCur1)),  
 v\_buyStables.up(stables,hor,t1,nCur1))  
 or sum( tOld, p\_iniStables(stables,hor,tOld))) $ t\_n(t,nCur) ) ..  
  
 v\_stableInv(stables,hor,t,nCur)  
  
 =L=  
\*  
\* --- old stables according to building date and lifetime  
\* (will drop out of equation if too old)  
\*  
 sum( tOld $ ( ((p\_year(tOld) + p\_lifeTimeS(stables,hor)) ge p\_year(t))  
 $ ( p\_year(told) le p\_year(t))),  
 p\_iniStables(stables,hor,tOld))  
  
\*  
\* --- plus (old) investments - de-investments  
\*  
 + sum( t\_n(t1,nCur1) $ ( isNodeBefore(nCur,nCur1)  
 $ ( ((p\_year(t1) + p\_lifeTimeS(stables,hor) ) ge p\_year(t))  
 $ ( p\_year(t1) le p\_year(t)))),  
 v\_buyStables(stables,hor,t1,nCur1));

The variable *v\_stableUsed* is defined as the size of a herd multiplied with the herds requirement for stable places in a particular stable type.

stables\_(stableTypes,tFull(t),nCur,m)  
 $ (sum(actHerds(sumHerds,breeds,feedRegime,t,m) $ ((not sameas(feedRegime,"fullGraz"))  
 $ v\_herdSize.up(sumHerds,breeds,feedRegime,t,nCur,m)), p\_stableNeed(sumHerds,breeds,stableTypes))  
 $ t\_n(t,nCur) ) ..  
\*  
\* --- herd sizes times their request for specific stable "types" (cow, calves, young cattle)  
\*  
  
 sum(actHerds(sumHerds,breeds,feedRegime,t,m) $ (not sameas(feedRegime,"fullGraz")),  
 v\_herdSize(sumHerds,breeds,feedRegime,t,nCur,m)  
 \* p\_stableNeed(sumHerds,breeds,stableTypes))  
 =L=  
\*  
\* --- must be covered by current stable inventory (not fully depreciated building),  
\* mutiplied with the stable places they offer  
\*  
 sum(stables $ ( sum( (t\_n(t1,nCur1),hor) $ (isNodeBefore(nCur,nCur1) or sameas(nCur,nCur1)),  
 v\_buyStables.up(stables,hor,t1,nCur1))  
 or sum( (tOld,hor), p\_iniStables(stables,hor,tOld))),  
 v\_stableUsed(stables,t,nCur) \* p\_stableSize(stables,stableTypes));

Eventually, the utilised stable places (*v\_stableUsed*), need to be covered by the available stable inventory (*v\_stableInv*):

stableUsed\_(stables,hor,tFull(t),nCur) $ ( ( sum( t\_n(t1,nCur1) $ (isNodeBefore(nCur,nCur1) or sameas(nCur,nCur1)),  
 v\_buyStables.up(stables,hor,t1,nCur1))  
 or sum( (tOld), p\_iniStables(stables,hor,tOld))) $ t\_n(t,nCur)) ..  
  
 v\_stableUsed(stables,t,nCur) =L= v\_stableInv(stables,hor,t,nCur);

The investment horizon set *hor* differentiates between the initial investment into the building, assumed to last for 30 years, and certain equipment for which maintenance investments are necessary after 10 or 15 years for cow stables.

A stable can only be used, if short and middle term maintenance investments are done.

As certain maintenance costs are linked to stables, the share of the used stable is restricted to minimum 75%, which assumes that maximal 25% of the maintenance costs can be saved when the stable is not fully used:

stableCostLo\_(stables,hor,tFull(t),nCur)  
 $ (( sum( t\_n(t1,nCur1) $ (isNodeBefore(nCur,nCur1) or sameas(nCur,nCur1)),  
 v\_buyStables.up(stables,hor,t1,nCur1))  
 or sum( (tOld), p\_iniStables(stables,hor,tOld))) $ t\_n(t,nCur) ) ..  
  
 v\_stableShareCost(stables,t,nCur) =G= v\_stableInv(stables,hor,t,nCur) \* 0.75;

The different stable attributes are defined in "*coeffgen\stables.gms*".

# Other Type of Buildings

Besides stables the model currently includes silos for more manure, bunker silos for maize or grass silage and storages for potatoes.

Each type of manure silo is linked to an inventory equation:

siloInv\_(curManChain(manChain),silos,tCur(t),nCur)  
 $ ( ( sum(t\_n(t1,nCur1) $ isNodeBefore(nCur,nCur1), v\_buySilos.up(manChain,silos,t1,nCur1))  
 or sum(tOld, p\_iniSilos(manChain,silos,tOld))) $ t\_n(t,nCur) ) ..  
  
 v\_siloInv(manChain,silos,t,nCur)  
  
 =e=  
\*  
\* --- Old silo according to building date and lifetime  
\* (will drop out of equation if too old)  
\*  
 sum(tOld $ ( ((p\_year(tOld) + p\_lifeTimeSi(silos)) ge p\_year(t))  
 $ ( p\_year(told) le p\_year(t))),  
 p\_iniSilos(manChain,silos,tOld))  
  
\*  
\* --- Plus (old) investments - de-investments  
\*  
 + sum(t\_n(t1,nCur1) $ (tcur(t1) $ isNodeBefore(nCur,nCur1)  
 and ( ((p\_year(t1) + p\_lifeTimeSi(silos)) ge p\_year(t))  
 $ ( p\_year(t1) le p\_year(t)))),  
 v\_buysilos(manChain,silos,t1,nCur1));

The manure silos are linked to the manure storage needs, which are described in chapter [Manure](manure.md)Manure. A similar inventory equation as for manure silos is implemented for the other buildings:

buildingInv\_(curBuildings(buildings),tCur(t),nCur)  
 $ ( ( sum(t\_n(t1,nCur1) $ isNodeBefore(nCur,nCur1), v\_buyBuildings.up(buildings,t1,nCur1))  
 or (sum(tOld, p\_iniBuildings(buildings,tOld)))) $ t\_n(t,nCur) ) ..  
  
 v\_buildingsInv(buildings,t,nCur)  
  
 =e=  
\*  
\* --- old silo according to building date and lifetime  
\* (will drop out of equation if too old)  
\*  
 sum(tOld $ ( ((p\_year(tOld) + p\_lifeTimeBuild(buildings)) ge p\_year(t))  
 $ ( p\_year(told) le p\_year(t))),  
 p\_iniBuildings(buildings,tOld))  
  
\*  
\* --- plus (old) investments - de-investments  
\*  
 + sum(t\_n(t1,nCur1) $ ( ((p\_year(t1) + p\_lifeTimeBuild(buildings)) ge p\_year(t))  
 $ ( p\_year(t1) le p\_year(t))  
 $ tcur(t1) $ isNodeBefore(nCur,nCur1)),  
 v\_buyBuildings(buildings,t1,nCur1));

The buildings included in the model are:

set s\_bunkerSilos /  
 bunkerSilo450  
 bunkerSilo900  
 bunkerSilo1620  
 bunkerSilo2640  
 bunkerSilo3630  
 bunkerSilo4620  
 bunkerSilo8580  
 bunkerSilo11870  
 bunkerSilo26550  
 /;

set buildings / potaStore500t  
 set.s\_bunkerSilos  
 /;

The attributes of the buildings are defined in *coeffgen\buildings.gms*:

table p\_building(buildings,buildAttr)  
 invSum capac\_t capac\_m3 lifeTime varCost  
  
 potaStore500t 195850 500 12 323  
\*  
\* --- KTBL 2014/15 p.144  
\*  
 bunkerSilo450 34176 450 20  
 bunkerSilo900 60900 900 20  
 bunkerSilo1620 84490 1620 20  
 bunkerSilo2640 115770 2640 20  
 bunkerSilo3630 127110 3630 20  
 bunkerSilo4620 138450 4620 20  
 bunkerSilo8580 218250 8580 20  
 bunkerSilo11870 284970 11870 20  
 bunkerSilo26550 482000 26550 20  
 ;

The inventory of the buildings is linked to building needs of certain activities:

buildingNeed\_(curBuildType(buildType),buildCapac,tCur(t),nCur)  
 $ (sum(curProds(prods),p\_buildingNeed(prods,buildType,buildCapac)) $ t\_n(t,nCur) ) ..  
  
 sum(buildType\_buildings(buildType,buildings)  
 $ ( ( sum(t\_n(t1,nCur1) $ isNodebefore(nCur,nCur1), v\_buyBuildings.up(buildings,t1,nCur1))  
 or sum(tOld, p\_iniBuildings(buildings,tOld)))  
 $ curBuildings(buildings)),  
  
 v\_buildingsInv(buildings,t,nCur) \* p\_building(buildings,buildCapac))  
  
 =G= v\_buildIngNeed(buildType,t,nCur);

# Farm Machinery

The model includes farm machineries in quite some detail:

set machType / tractor  
 tractorSmall  
 plough "Pflug"  
 chiselPlough "Schwergrubber"  
 sowMachine "S�maschine"  
 directSowMachine "DirektS�maschine"  
 seedBedCombi "Saatbeetkombination"  
 circHarrow "Scheibenegge"  
 springTineHarrow "Federzinkenegge"  
 fingerHarrow "Hackstriegel"  
 combine "Maehdrescher"  
 cuttingUnitCere "Getreideschneidwerk"  
 cuttingAddRape "Zusatzausruestung Rapsernte"  
 cuttingUnitMaiz "Maispflueckeinrichtung f�r M�hdrescher"  
 rotaryHarrow "Kreiselegge"  
 mulcher "Mulcher"  
 potatoPlanter "Kartoffellegegeraet"  
 potatoLifter "Kartoffelroder"  
 hoe "Hackmachine, 5-reihig"  
 ridger "Haeufler"  
 haulmCutter "Krautschlaeger"  
 forkLiftTruck "Gabelstapler"  
 threeWayTippingTrailer "Dreiseitenkippanhaenger"  
 Sprayer "Feldspritze"  
 singleSeeder "Einzelkornsaehgeraet (Rueben/Mais)"  
 beetHarvester "Ruebenroder"  
 fertSpreaderSmall "Duengerstreuer, 0.8cbm"  
 fertSpreaderLarge "Duengerstreuer, 4.0cbm"  
 chopper "Feldhaecksler"  
 cornHeader "Maisgebiss fuer Haecksler"  
 mowerConditioner "Maehaufbereiter"  
 grasReseedingUnit "Gasnachsaemaschine"  
 rotaryTedder "Kreiselzettwender"  
 rake "Schwader"  
 roller "Walze"  
 silageTrailer "Silage trailer, service"  
 balePressWrap "Baler and bale wrapper, service"  
 balePressHay "Baler"  
 closeSilo  
  
 manbarrel,draghose,injector,trailingshoe  
  
 solidManDist "Miststreuer"  
 frontLoader "Frontlader"  
 siloBlockCutter "Siloblockschneider"  
 shearGrab "Schneidzange"  
 dungGrab "Dungzange"  
 fodderMixingVeh8 "Futtermischwagen, 8m3, horizontale Schnecke, mit Befuellschild"  
 fodderMixingVeh10 "Futtermischwagen, 10m3, vertikale Schnecke, mit Befuellschild"  
 fodderMixingVeh16 "Futtermischwagen, 16m3, 2 vertikale Schnecken, mit Befuellschild"  
 /;

For further information see Appendix A1.

Each machinery type is characterised by set of attributes *p\_machAttr* (see *coeffgen\mach.gms*), for example:

table p\_machAttr(machType,machAttr) "Machinery attribute for default size (67kw, 2 ha)"  
  
\*  
\* --- Data from KTBL 2014/2015, if not otherwise stated  
\*  
\*  
\* --- KTBL. 82, 4 Schare, 140 cm  
\*  
 price hour ha m3 t varCost\_ha varCost\_t varCost\_h diesel\_h fixCost\_h fixCost\_t years varCost\_m3  
 Plough 13000 2000 12.0  
\*  
\* --- KTBL. 84, Schwergrubber, angebaut, 2.5m  
\*  
 ChiselPlough 5600 2600 5.0

## Farm Operations: Machinery Needs and Related Costs

Machinery is linked to specific farm operations (see *tech.gms*):

set operation "Field operators as defined by KTBL"  
  
 /  
 soilSample "Bodenprobe"  
 manDist "G�lleausbringung"  
 basFert "P und K Duengung, typischerweise Herbst"  
 plow "Pfl�gen"  
 chiselPlow "Tiefengrubber"  
 seedBedCombi "Saatbettkombination"  
 herb "Herbizidma�nahme"  
 sowMachine "Saemaschine"  
 directSowMachine "Direktsaatmaschine"  
 circHarrowSow "Kreiselegge u. Drillmaschine Kombination"  
 springTineHarrow "Federzinkenegge"  
 weedValuation "Unkrautbonitur"  
 weederLight "Striegeln"  
 weederIntens "Hacken"  
 plantvaluation "Bestandsbonitur"  
 NFert320  
 NFert160  
 combineCere "M�hdrusch, Getreide"  
 combineRape "M�hdrusch, Raps"  
 combineMaiz "M�hdrusch, Mais"  
 cornTransport "Getreidetransport"  
 store\_n\_dry\_8  
 store\_n\_dry\_4  
 store\_n\_dry\_beans  
 store\_n\_dry\_rape  
 store\_n\_dry\_corn  
 lime\_fert "Kalkung"  
 stubble\_shallow "Stoppelbearbeitung flach"  
 stubble\_deep "Stoppelbearbeitung tief"  
 rotaryHarrow "Kreiselegge"  
 NminTesting "Nmin Probenahme"  
 mulcher "Mulcher"  
 chitting "Vorkeimen"  
 solidManDist "Miststreuer"  
 seedPotatoTransp "Pflanzkartoffeltransport"  
 potatoLaying "Legen von Kartoffeln"  
 rakingHoeing "Hacken, striegeln"  
 earthingUp "h�ufeln"  
 knockOffHaulm "Kartoffelkraut schlagen"  
 killingHaulm "Krautabt�ten"  
 potatoHarvest "Kartoffeln roden"  
 potatoTransport "Kartoffeln zum Lager transportieren"  
 potatoStoring "Kartoffeln lagern"  
 singleSeeder "Einzelkornlegeger�t f�r Zuckerr�ben/Mais"  
 weederHand "von Hand hacken"  
 uprootBeets "Zuckerr�ben roden"  
 DiAmmonium "Diammonphosphat streuen"  
 grinding "KornMahlen"  
 disposal "Erntegut festfahren"  
 coveringSilo "Silo reinigen und mit Folie verschliessen, Maiz"  
 chopper "H�ckseln"  
 grasReSeeding "Grasnachs�en"  
 roller "Walzen"  
 mowing "M�hen mit M�haufbereiter"  
 raking "Schwaden"  
 tedding "Wenden mit Kreiselzettwender"  
 silageTrailer "Anwelkgut bergen mit Ladewagen"  
 closeSilo "Silo reinigen und mit Folie verschliessen"  
\* Hay/Bale specific tasks  
 balePressWrap "Ballen pressen und wickeln, Silage (Anwelkgut)"  
 baleTransportSil "Ballentransport Silageballen"  
 baleTransportHay "Ballentransport Heuballen"  
 balePressHay "Bodenheu pressen"  
 /;

For more details see Appendix A2.

Labour needs, diesel, variable and fixed machinery costs are linked to these operations. An extraction is shown in the following:

table op\_attr(operation,machVar,plotSize,opAttr)  
  
 labTime diesel fixCost varCost nPers amount  
 soilSample .67kw.2ha 0.2 0.5 1.05 0.30  
 manDist .67kw.2ha 1.7 6.7 20.20 24.65  
 basFert .67kw.2ha 0.25 0.9 2.04 2.11  
\*  
\* --- page 153, KTBL 2010/2011  
\*  
 plow .67kw.2ha 1.89 23.0 20.39 40.76  
 chiselPlow .67kw.2ha 1.09 15.1 9.02 22.92  
 SeedBedCombi .67kw.2ha 0.58 6.0 7.98 12.05  
 sowMachine .67kw.2ha 0.84 4.9 9.44 10.62  
 directSowMachine .67kw.2ha 0.71 6.5 23.01 22.59  
 circHarrowSow .67kw.2ha 1.29 12.9 16.96 27.16  
 springTineHarrow .67kw.2ha 0.75 7.3 6.56 13.60  
 weedValuation .67kw.2ha 0.16 0.3 1.59 0.35  
 herb .67kw.2ha 0.28 1.0 4.37 3.25  
 weederLight .67kw.2ha 0.42 2.6 3.93 6.22  
 weederIntens .67kw.2ha 0.73 3.8 13.10 9.70  
 plantValuation .67kw.2ha 0.13 0.1 0.91 0.18  
 NFert320 .67kw.2ha 0.23 0.9 1.75 1.95  
 NFert160 .67kw.2ha 0.19 0.8 1.16 1.58  
 lime\_fert .67kw.2ha 0.48 3.6 12.54 6.51  
 combineCere .67kw.2ha 1.20 20.8 66.43 31.94  
 combineRape .67kw.2ha 1.25 22.83 86.11 40.73  
 combineMaiz .67kw.2ha 1.32 23.99 115.57 54.54  
 cornTransport .67kw.2ha 0.23 0.8 5.28 3.41  
 store\_n\_dry\_8 .67kw.2ha 1.29 100.81 29.28  
 store\_n\_dry\_4 .67kw.2ha 0.64 50.41 14.64  
 store\_n\_dry\_beans .67kw.2ha 0.47 33.42 11.56  
 store\_n\_dry\_rape .67kw.2ha 0.64 49.38 40.52  
 store\_n\_dry\_corn .67kw.2ha 1.50 107.36 255.20  
\*  
\* --- page 152 KBL 2010/2011  
\*  
 stubble\_shallow .67kw.2ha 0.85 8.4 7.54 16.59  
 stubble\_deep .67kw.2ha 0.92 9.8 7.99 18.04  
\*  
\*--- KTBL 12/13 S. 420 [TK,24.07.13]  
\*  
 rotaryHarrow .67kw.2ha 1.17 9.40 8.27 22.06  
 NminTesting .67kw.2ha 0.51 0.18 1.32 0.34  
 mulcher .67kw.2ha 1.40 8.39 14.51 20.59  
 chitting .67kw.2ha 2.36 481.82 97.80  
 solidManDist .67kw.2ha 1.61 10.88 32.73 30.99  
 seedPotatoTransp .67kw.2ha 0.26 0.94 2.77 2.72  
 potatoLaying .67kw.2ha 1.19 11.84 23.94 31.60  
 rakingHoeing .67kw.2ha 0.73 4.12 11.65 10.80  
 earthingUp .67kw.2ha 0.70 3.49 7.67 10.03  
 knockOffHaulm .67kw.2ha 1.92 8.41 22.24 23.46  
 killingHaulm .67kw.2ha 0.23 1.15 5.48 3.09  
 potatoHarvest .67kw.2ha 19.94 55.23 189.53 133.98 3  
 potatoTransport .67kw.2ha 1.61 5.37 31.63 22.82  
\*  
\* --- fix costs covered by potaStore type buildings  
\*  
 potatoStoring .67kw.2ha 10.00 148.50  
  
  
\*  
\*--- KTBL 12/13 S.437 und 445 (BL 10.02.2014)  
\*  
 singleSeeder .67kw.2ha 1.0 4.26 28.3 18.39  
 weederHand .67kw.2ha 71.52 0.35 1.26 1.09  
 uprootBeets .67kw.2ha 4.41 49.73 149.98 134.33  
  
\*  
\*--- KTBL 12/13 S.348 (BL 10.02.2014)  
\*  
 DiAmmonium .67kw.2ha 0.16 0.65 0.86 1.48  
 grinding .67kw.2ha 84  
 disposal .67kw.2ha 0.7 3.57 4.19 7.55  
\*--- KTBL 14/15 S.331 (WB 27.07.2016)  
\* coveringSilo .67kw.2ha 4.2 265.15 60.61  
 coveringSilo .67kw.2ha 4.2 000.00 60.61  
  
\* H?cksler wird bei KTBL nur als Dienstleistung gef?hrt, nicht zur Eigenanschaffung  
\*  
 chopper .67kw.2ha 410  
\*  
\*--- KTBL 14/15 S.453 (CP 28.02.2018)  
\*  
\* labTime diesel fixCost varCost nPers amount  
 mowing .67Kw.2ha 0.64 5.47 8.48 11.39  
 tedding .67kw.2ha 0.43 2.78 3.56 6.88  
 raking .67kw.2ha 0.51 3.12 4.45 8.02  
 silageTrailer .67kw.2ha 98.00 11.9  
 closeSilo .67kw.2ha 1.09 69.42 15.87  
 grasReSeeding .67kw.2ha 0.27 2.07 3.63 4.44  
 roller .67kw.2ha 0.34 1.72 3.91 4.36  
\*--- KTBL 14/15 S.458 (Silage)/S.515 (Hay) (CP 27.02.2018)  
\*--- Ballenpressen mit Wickeln wird bei KTBL als Dienstleistung aufgeführt  
 balePressWrap .67kw.2ha 240.00 11.9  
 balePressHay .67kw.2ha 0.5 3.02 15.45 14.19 4.8  
 baleTransportSil .67kw.2ha 1.65 3.29 21.66 16.27 11.9  
 baleTransportHay .67kw.2ha 1.62 3.02 15.45 14.19 4.8  
;

Furthermore, the model considers the effect of different plot sizes and the mechanisation levels:

table p\_plotSizeEffect(crops,machVar,opAttr,plotSize)  
  
 1ha 2ha 5ha 20ha  
  
 winterWheat. 67kw .labTime 12.4 10.5 9.3 8.0  
 winterWheat. 67kw .diesel 90 83 78 73  
 winterWheat. 67kw .varCost 205 188 176 168  
 winterWheat. 67kw .fixCost 282 258 241 231  
  
 winterWheat.102kw .labTime 11.1 9.1 7.6 6.8  
 winterWheat.102kw .diesel 95 86 78 74  
 winterWheat.102kw .varCost 209 188 172 164  
 winterWheat.102kw .fixCost 315 284 262 249  
  
 winterWheat.200kw .labTime 11.9 8.6 6.3 4.9  
 winterWheat.200kw .diesel 118 99 84 75  
 winterWheat.200kw .varCost 240 201 173 157  
 winterWheat.200kw .fixCost 396 334 292 267  
 ;

p\_plotSizeEffect("winterWheat",machVar,"nPers",plotSize) = 1;

p\_plotSizeEffect("winterWheat",machVar,"amount",plotSize) = 1;

p\_plotSizeEffect(crops,machVar,opAttr,plotSize) $ (not p\_plotSizeEffect(crops,machVar,opAttr,plotSize))  
 = sum( crops1, p\_plotSizeEffect(crops1,machVar,opAttr,plotSize))  
 /sum( crops1 $ p\_plotSizeEffect(crops1,machVar,opAttr,plotSize),1);

The farm operations are linked to cropping activities (below an example for potatoes):

table crop\_op\_per\_till(crops,operation,labPeriod,till)  
 plough minTill noTill eco silo bales  
  
 potatoes . soilSample . AUG1 0.2 0.2 0.2  
 potatoes . basFert . AUG1 1.0 1.0 1.0  
 potatoes . solidManDist . AUG2 1.0  
 potatoes . plow . AUG2 1.0  
 potatoes . chiselPlow . AUG2 1.0 1.0  
 potatoes . sowmachine . AUG2 1.0 1.0  
 potatoes . mulcher . NOV1 1.0 1.0 1.0  
 potatoes . plow . NOV1 1.0 1.0  
 potatoes . chiselPlow . NOV1 1.0  
 potatoes . NminTesting . FEB1 1.0 1.0  
 potatoes . NFert320 . MAR1 1.0 1.0 1.0  
 potatoes . chitting . MAR1 1.0  
 potatoes . seedBedCombi . MAR2 1.0  
 potatoes . rotaryHarrow . MAR2 1.0 1.0  
 potatoes . seedPotatoTransp . APR1 1.0 1.0 1.0  
 potatoes . potatoLaying . APR1 1.0 1.0 1.0  
 potatoes . rakingHoeing . APR2 1.0  
 potatoes . earthingUp . APR2 1.0 1.0  
 potatoes . weedValuation . MAY1 1.0 1.0 1.0  
 potatoes . earthingUP . MAY1  
 potatoes . plantvaluation . JUN1 1.0 1.0  
 potatoes . herb . JUN1 1.0  
 potatoes . plantValuation . JUN2 2.0 2.0 1.0  
 potatoes . herb . JUN2 2.0 2.0 2.0  
 potatoes . plantValuation . JUL1 2.0 2.0  
 potatoes . herb . JUL1 2.0 2.0 1.0  
 potatoes . plantValuation . JUL2 1.0 1.0  
 potatoes . herb . JUL2 1.0 1.0 1.0  
 potatoes . plantValuation . AUG1 1.0 1.0  
 potatoes . herb . AUG1 1.0 1.0 1.0  
 potatoes . plantValuation . AUG2 1.0 1.0  
 potatoes . herb . AUG2 1.0 1.0  
 potatoes . knockOffHaulm . AUG2 1.0  
 potatoes . killingHaulm . AUG2 1.0 1.0  
 potatoes . potatoHarvest . SEP1 0.5 0.5 0.5  
 potatoes . potatoTransport . SEP1 0.5 0.5 0.5  
 potatoes . potatoStoring . SEP1 0.5 0.5 0.5  
 potatoes . potatoHarvest . SEP2 0.5 0.5 0.5  
 potatoes . potatoTransport . SEP2 0.5 0.5 0.5  
 potatoes . potatoStoring . SEP2 0.5 0.5 0.5  
 potatoes . lime\_fert . OCT1 0.333

These information on farm operations determine

1. The **number of necessary field working days** and *monthly labour need* per ha (excluding the time used for fertilising, which is determined endogenously)
2. The **machinery need** for the different crops
3. Related **variable costs**

The labour needs per month are determined by summing up over all farm operations, considering the labour period, the effect of plot size and mechanisation (*coeffgen\labour.gms*):

p\_cropLab(crops,till,intens,m) $ sum(plot,c\_s\_t\_i(crops,plot,till,intens))  
  
 = sum( (operation,actmachVar,actPlotSize,labPeriod\_to\_month(labPeriod,m)),  
 crop\_op\_per\_till(crops,operation,labPeriod,till)  
 \* op\_attr(operation,"67kW","2ha","labTime")  
\*  
\* -- effect of plot size and mechanisation on labour time  
\*  
 \* p\_plotSizeEffect(crops,actMachVar,"labTime",actPlotSize)  
 /p\_plotSizeEffect(crops,"67kW","labTime","2ha") );

## Endogenous Machine Inventory

The inventory equation for machinery is shown in *machInv\_*, where *v\_machInv* is the available inventory by type, *machType,* in operation hours. *v\_machNeed* is the machinery need of the farm in operating hours and *v\_buyMach* are investments in new machines.

machInv\_(curMachines(machType),machLifeUnit,tFull(t),nCur)  
 $ ( (v\_machInv.up(machType,machLifeUnit,t,nCur) ne 0)  
 $ p\_lifeTimeM(machType,machLifeUnit) $ p\_priceMach(machType,t)  
 $ (not sameas(machLifeUnit,"years")) $ t\_n(t,nCur) ) ..  
\*  
\* --- inventory end of current year (in operating hours, hectares etc.)  
\*  
 v\_machInv(machType,machLifeUnit,t,nCur)  
  
 =e=  
\*  
\* --- inventory end of last year (in operating hours)  
\*  
 sum(t\_n(t-1,nCur1) $ anc(nCur,nCur1), v\_machInv(machType,machLifeUnit,t-1,nCur1))  
\*  
\* --- new machines, converted in operation time  
\*  
 + (v\_buyMach(machType,t,nCur)+v\_buyMachFlex(machType,t,nCur)) \* p\_lifeTimeM(machType,MachLifeUnit)  
\*  
\* --- minus operating hours in current year if in normal planning period  
\*  
 - v\_machNeed(machType,machLifeUnit,t,nCur) $ tCur(t)  
\*  
\* --- minus operating hours of weighted average over normal planning period  
\* if beyond the normal planning period  
\*  
 - [sum( (t\_n(t1,nCur1)) $ ( (p\_year(t1) lt p\_year(t)) $ tCur(t1) $ isNodeBefore(nCur,nCur1)),  
 v\_machNeed(machType,machLifeUnit,t1,nCur1)  
 \* 1/(p\_year(t)+5 - p\_year(t1)) )  
 /sum( (t1) $ ( (p\_year(t1) lt p\_year(t)) $ tCur(t1)), 1/(p\_year(t)+5 - p\_year(t1)) )  
 ]  
 $ ( (not tCur(t)) and p\_prolongCalc)  
 ;

The last expression is used when the farm program for the simulated period is used to estimate the machinery needs for all years until the stables are fully depreciated.

The machinery need in each year is defined from activities or processes requiring machinery::

machNeedHerds\_(curMachines(machType),machLifeUnit,tCur(t),nCur)  
 $ (sum(actHerds(sumHerds,breeds,feedRegime,t,m),  
 p\_machNeed(sumHerds,"plough","normal",machType,machLifeUnit)) $ t\_n(t,nCur)) ..  
  
 v\_machNeedHerds(machType,machLifeUnit,t,nCur)  
  
 =e=  
\*  
\* --- herd sizes times their request for specific machine type  
\*  
 sum(actHerds(sumHerds,breeds,feedRegime,t,m) $ p\_prodLength(sumHerds,breeds),  
 v\_herdSize(sumHerds,breeds,feedRegime,t,nCur,m)  
 \* p\_machNeed(sumHerds,"plough","normal",machType,machLifeUnit)  
 \* 1/min(12,p\_prodLength(sumHerds,breeds)) \* 12/card(herdM));

machines\_(curMachines(machType),machLifeUnit,tCur(t),nCur) $ (p\_lifeTimeM(machType,machLifeUnit) $ t\_n(t,nCur)) ..  
\*  
\* --- crops times their request for specific machine type  
\*  
 + sum( c\_s\_t\_i(curCrops(crops),plot,till,intens),  
 v\_cropHa(crops,plot,till,intens,t,nCur)  
 \* p\_machNeed(crops,till,intens,machType,machLifeUnit))  
  
 + sum( (c\_s\_t\_i(curCrops(crops),plot,till,intens),syntFertilizer,m),  
 v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m)  
 \* p\_machNeed(syntFertilizer,"plough","normal",machType,machLifeUnit))  
  
\* ---- machine need for the application of N (manure/fertilizer)  
  
$iftheni.man %manure% == true  
  
 + sum((c\_s\_t\_i(curCrops(crops),plot,till,intens),manApplicType\_manType(ManApplicType,curManType),m)  
 $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m) ne 0),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m)  
 \* p\_machNeed(ManApplicType,"plough","normal",machType,machLifeUnit))  
  
$endif.man  
  
$iftheni.herd %herd%==true  
  
 + v\_machNeedHerds(machType,machLifeUnit,t,nCur)  
 $ sum(actHerds(sumHerds,breeds,feedRegime,t,m),  
 p\_machNeed(sumHerds,"plough","normal",machType,machLifeUnit))  
$endif.herd  
\*  
\* --- total machinery need  
\*  
 =L= v\_machNeed(machType,machLifeUnit,t,nCur)  
 ;

A small set of machinery, such as the front loader, dung grab, shear grab or fodder mixing vehicles are depreciated by time and not by use:

machInvT\_(curMachines(machType),tFull(t),nCur)  
 $ ( (v\_machInv.up(machType,"years",t,nCur) ne 0)  
 $ p\_lifeTimeM(machType,"years")  
 $ p\_priceMach(machType,t) $ t\_n(t,nCur) ) ..  
\*  
\* --- inventory end of current year (in operating hours)  
\*  
 v\_machInv(machType,"years",t,nCur)  
  
 + sum( t\_n(t1,nCur1) $ ( (p\_year(t1) gt smax( tOld $ p\_iniMachT(machType,told),  
 p\_year(tOld) + p\_lifeTimeM(machType,"years")))  
 $ (p\_year(t1)+p\_prolongLen gt p\_year(t))  
 $ tCur(t1) $ isNodeBefore(nCur,nCur1)),  
 v\_machInv(machType,"years",t1,nCur1)/p\_proLongLen)  
 $ ( (not tCur(t)) and p\_prolongCalc)  
  
 =L=  
\*  
\* --- old machines according to investment dates  
\* (will drop out of equation if too old)  
\*  
 sum( tOld $ ( ((p\_year(tOld) + p\_lifeTimeM(machType,"years")) ge p\_year(t))  
 $ ( p\_year(told) le p\_year(t))),  
 p\_iniMachT(machType,tOld))  
  
\*  
\* --- plus (old) investments - de-investments  
\*  
 + sum( t\_n(t1,nCur1) $ ( ((p\_year(t1) + p\_lifeTimeM(machType,"years") ) ge p\_year(t))  
 $ ( p\_year(t1) le p\_year(t))  
 $ isNodeBefore(nCur,nCur1)),  
 v\_buyMach(machType,t1,nCur1));

The aforementioned set of machinery, depreciated by time and not usage, are linked to the existence of stables, i.e. stables cannot be used if machinery is not present:

machInvStable\_(curMachines(machType),stables,tCur(t),nCur) $ ( (v\_machInv.up(machType,"years",t,nCur) ne 0)  
 $ ( sum( t\_n(t1,nCur1) $ isNodeBefore(nCur,nCur1),  
 v\_buyStables.up(stables,"long",t1,nCur1))  
 or sum( tOld, p\_iniStables(stables,"long",tOld)))  
 $ sum(stables\_to\_mach(stables,machType),1)  
  
 $ p\_lifeTimeM(machType,"years") $ p\_priceMach(machType,t) $ t\_n(t,nCur)) ..  
  
 sum(stables\_to\_mach(stables,machType), v\_stableInv(stables,"long",t,nCur))  
 =L= v\_machInv(machType,"years",t,nCur);

# Investments, Financing and Cash Flow Definition

!!! abstract The investment module depicts investment decisions in machinery, stables and structures (silos, biogas plants, storage) as binary variables with a yearly resolution. Physical depreciation can be based on lifetime or use. Machinery use can be alternatively depicted as continuous re-investment rendering investment costs variable, based on a Euro per ha threshold. Investment can be financed out of (accumulated) cash flow or by credits of different length and related interest rates. For stables and biogas plants maintenance investment are reflected as well.

The total investment sum *v\_sumInv* in each year is defined by:

invSum\_(tFull(t),nCur) $ t\_n(t,nCur) ..  
\*  
 v\_sumInv(t,nCur) =e=  
\*  
\* --- new land bought  
\*  
$ifi %landBuy% == true sum( plot, v\_buyPlot(plot,t,nCur)\*p\_pland(plot,t)) $ tCur(t)  
\*  
\* --- new stables bought  
\*  
$ifi %herd%==true + sum( (stables,hor), v\_buyStables(stables,hor,t,nCur)\*p\_priceStables(stables,hor,t))  
\*  
\* --- buildings and structures  
\*  
 + sum(curBuildings(buildings), v\_buyBuildings(buildings,t,nCur) \* p\_priceBuild(buildings,t))  
\*  
\* --- new machinery bought (integer and continous depreciation solutinN  
\*  
 + sum(curMachines(machType),  
 (v\_buyMach(machType,t,nCur)+v\_buyMachFlex(machType,t,nCur))\*p\_priceMach(machType,t))  
\*  
\* --- new manure silos bought  
\*  
$ifi %herd%==true + sum( (curManChain(manChain),silos), v\_buySilos(manChain,silos,t,nCur)\*p\_priceSilo(silos,t))  
  
\*  
\* --- new biogas plant bought  
\*  
$iftheni %biogas%==true  
  
 + sum((curBhkw(bhkw), curEeg(eeg)),  
 v\_buyBioGasPlant(bhkw,eeg,"ih20",t,nCur) $tCur(t)  
 \* p\_priceBioGasPlant(bhkw,"ih20"))  
  
 + sum((curBhkw(bhkw), ih),  
 v\_buyBioGasPlantParts(bhkw,ih,t,nCur)  
 \* ( p\_priceBioGasPlant(bhkw,ih) $ (not(ih20(ih)))  
\* + p\_priceFlexBioGasPlant(bhkw,eeg,ih)$eegDM(eeg) )  
 ))  
$endif  
;

Investments can be financed either by equity or by credits, and enters accordingly the cash balance definition, *v\_liquid*. The cash balance represents the cash at the end of the forgone year plus the net cash flow, *v\_netCashFlow*, in the current year plus new credits, *v\_credits*, minus fixed household expenditures, *p\_hcon*, and new investments, *v\_sumInv*:

Liquid\_(tFull(t),nCur) $ t\_n(t,nCur) ..  
\*  
 v\_liquid(t,nCur) =e=  
\*  
\* --- last years liquidity  
\*  
 + sum(t\_n(t-1,nCur1) $ anc(nCur,nCur1), v\_liquid(t-1,nCur1))  
\*  
\* --- total cash flow  
\*  
 + v\_netCashFlow(t,nCur)  
 ;

The model differentiates credits by repayment period, *p\_payBackTime*, and interest rate. Credits are paid back in equal instalments over the repayment period, hence, annuities decrease over time. The amount of outstanding credits is defined by the following equation:

credSum\_(creditType,tFull(t),nCur) $ t\_n(t,nCur) ..  
\*  
 v\_sumCredits(creditType,t,nCur) =e=  
\*  
 sum( t\_n(t1,nCur1) $ ( (((p\_year(t1) + p\_payBackTime(creditType)) ge p\_year(t))  
 $ ( p\_year(t1) le p\_year(t)))  
 $ tCur(t1) $ isNodeBefore(nCur,nCur1)),  
 v\_credits(creditType,t1,nCur1)  
 \* (1-1/p\_payBackTime(creditType) \* (p\_year(t)-p\_year(t1))));

The net cash flow is defined as the sum of the gross margins, *v\_objeTS* plus received interest and revenue from liquidation (selling equipment or land) minus storing costs for manure, interest paid on outstanding credits and repayment of credits:

netCashFlow\_(tFull(t),nCur) $ t\_n(t,nCur) ..  
\*  
 v\_netCashFlow(t,nCur) =e=  
\*  
\* --- financial and investment link cash flows  
\*  
 + v\_finCashFlow(t,nCur)  
 + v\_InvCashFlow(t,nCur)  
\*  
\* --- operation cash flow  
\*  
 + v\_opCashFlow(t,nCur)  
 ;

Revenues from liquidation are only assumed to take place in the last year (of the simulation):

liquidation\_(nCur) $ t\_n("%lastYearCalc%",nCur) ..  
  
 v\_liquidation(nCur) =e=  
\*  
\* --- assume that past credits  
\* are paid back in the last year (to prevent over-investments)  
\*  
 - sum(creditType, v\_sumCredits(creditType,"%lastYearCalc%",nCur))  
\*  
\* --- liquidity at the end of the simulation horizon  
\*  
\*  
\* --- sell machinery (assumption: linear according to  
\* operation time minus 33%)  
\*  
 + [ sum( (curMachines(machType),machLifeUnit) $ p\_lifeTimeM(machType,machLifeUnit),  
 v\_machInv(machType,machLifeUnit,"%lastYearCalc%",nCur)  
 /p\_lifeTimeM(machType,machLifeUnit)  
 \* p\_priceMach(machType,"%lastYearCalc%") \* 2/3)  
 / sum( (curMachines(machType),machLifeUnit)  
 $ p\_lifeTimeM(machType,machLifeUnit), 1) ] $ card(curMachines) $ p\_liquid  
\*  
\* --- sell land (transaction costs set to 4 times the yearly land rent)  
\* (only in case land can be bought or sold - eases the interpreation of the average objective value in each year)  
\*  
$iftheni.lb %landBuy% == true  
 + sum( plot, v\_totPlotLand(plot,"%lastYear%",nCur)  
 \* ( p\_pland(plot,"%lastYear%") - 4 \* p\_landRent(plot,"%lastYear%"))) $ p\_liquid  
$endif.lb  
$iftheni.dh %cowherd%==true  
\*  
\* --- sell cows, heifers, calves for raising in last year  
\*  
  
\*  
\* -- cows at 60% of value of a young cow  
\*  
 + sum( actHerds("cows",curBreeds,feedRegime,"%lastYear%",herdm),  
 sum(m\_to\_herdm("dec",herdm), v\_herdSize("cows",curBreeds,feedRegime,"%lastYear%",nCur,herdm))  
 \* p\_price("youngCow","conv","%lastYearCalc%") \* 0.6 ) $ p\_liquid  
\*  
\* -- heifers at 30% of value of a young cow  
\*  
 + sum( actHerds("heifs",curBreeds,feedRegime,"%lastYear%",herdm),  
 sum(m\_to\_herdm("dec",herdm), v\_herdSize("heifs",curBreeds,feedRegime,"%lastYear%",nCur,herdm))  
 \* p\_price("youngCow","conv","%lastYearCalc%") \* 0.3 ) $ p\_liquid  
\*  
\* -- raising cavles at 10% of value of a young cow  
\*  
 + sum( actHerds("fCalvsRais",curBreeds,feedRegime,"%lastYear%",herdm),  
 sum(m\_to\_herdm("dec",herdm), v\_herdSize("fCalvsRais",curBreeds,feedRegime,"%lastYear%",nCur,herdm))  
 \* p\_price("youngCow","conv","%lastYearCalc%") \* 0.1 ) $ p\_liquid  
$endif.dh  
 ;

Liquidation is active if the model runs in fully dynamic mode and not in comparative static and short run mode.

The gross margin for each year is defined as revenues from sales, *v\_salRev*, income from renting out land, *v\_rentOutLand*, and salary from working off-farm minus variable costs. The latter relate to costs of buying intermediate inputs such as fertiliser, feed or young animals comprised in the equations structure of the model template, *v\_buyCost*, and other variable costs, *v\_varCosts*. For off-farm work (full-and half-time, v*\_workOff*) the weekly work time in hours, *p\_weekTime*, is given. In addition, it is assumed that off-farm work covers 46 weeks each year, so that income is defined from multiplying these two terms with hourly wage, *p\_wage*.

objeTS\_(t,s) .. v\_objeTS(t,s)  
 =e= sum(c, v\_cropHa(c,t,s) \* p\_cropGrossMarg(c,t,s))  
 + v\_salRev(t,s);

The sales revenues, *v\_salRev*, that enter the equation above are defined from net production quantities, *v\_prods*, and given prices in each year and SON, *p\_price*:

salRev\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
\*  
 v\_salRev(t,nCur) =e= sum( (curProds(prodsYearly),sys) $ (v\_saleQuant.up(prodsYearly,sys,t,nCur) ne 0),  
 p\_price(prodsYearly,sys,t)  
$iftheni.sp "%stochProg%"=="true"  
\*  
\* --- a product is both output and input, use price of inputs to avoid a situation  
\* where the product can be bought cheaper than it is sold  
\*  
 \* ( 1 + (p\_randVar("priceOutputs",nCur)-1) $ (randProbs(prodsYearly) and (not sum(sameas(prodsYearly,inputs),1)))  
 + (p\_randVar("priceInputs",nCur)-1) $ (randProbs(prodsYearly) and ( sum(sameas(prodsYearly,inputs),1)))  
 )  
$endif.sp  
 \* v\_saleQuant(prodsYearly,sys,t,nCur));

The sale quantity, *v\_saleQuant*, plus feed use, *v\_feedUse*, must exhaust the production quantity, *v\_prods*:

saleQuant\_(curProds(prodsYearly),tCur(t),nCur) $ (sum(sys,p\_price(prodsYearly,sys,t)) $ t\_n(t,nCur)) ..  
  
 sum(sys $ p\_price(prodsYearly,sys,t),v\_saleQuant(prodsYearly,sys,t,nCur))  
  
$iftheni.dh %cattle%==true  
  
 + sum( sameas(prodsYearly,feedsY), v\_feedUseProds(feedsY,t,nCur))  
  
  
$endif.dh  
\*  
$iftheni %biogas%==true  
 + sum( sameas(prodsYearly,crM),  
 sum( (curBhkw(bhkw),curEeg(eeg),m),  
 v\_feedBiogas(bhkw,eeg,crM,t,nCur,m) ) )  
  
  
$endif  
$iftheni.p %pigherd%==true  
 + sum(sameas(prodsYearly,feedsPig),  
 v\_feedOwnPig(feedspig,t,nCur))  
$endif.p  
 =L= v\_prods(prodsYearly,t,nCur)  
  
  
  
\*  
\* --- buying of products which are also produced on farm  
\* (silage, cereals)  
\*  
\* + sum(sameas(prodsYearly,curinputs(inputs)) $ p\_inputprice(inputs,t) ,  
\* v\_buy(inputs,t,nCur))  
;

The production quantities are derived by summing the production quantities of animal and crop production. Additionally, for milk quantities reduction of yield for specific cows and phases is considered:

prods\_(prodsYearly,tCur(t),nCur) $ (sum(sameas(prodsYearly,curProds),1) $ t\_n(t,nCur)) ..  
  
 v\_prods(prodsYearly,t,nCur)  
 =e=  
\*  
\* --- crop output  
\*  
 sum( c\_s\_t\_i(crops,plot,till,intens), v\_cropHa(crops,plot,till,intens,t,nCur)  
 \* sum(plot\_soil(plot,soil),p\_OCoeffC(crops,soil,till,intens,prodsYearly,t)))  
  
\*  
\* ---- removed residues  
\*  
 + sum( c\_s\_t\_i(crops,plot,till,intens) $ (cropsResidueRemo(crops)  
$iftheni.BWA "%branchMode%" == "BWA"  
 $ intensResRem(intens)  
$endif.BWA  
 )  
 , v\_residuesRemoval(crops,plot,till,intens,t,nCur)  
 \* sum(plot\_soil(plot,soil), p\_OCoeffResidues(crops,soil,till,intens,prodsyearly,t)) )  
\*  
\* --- residues used for own Consumption  
\*  
$iftheni.straw %strawManure% == true  
 - v\_residuesOwnConsum(prodsYearly,t,nCur) $ (sum(sameas (prodsYearly,prodsResidues),1))  
$endif.straw  
$iftheni.herd %herd% == true  
\*  
\* --- animal output  
\*  
 + sum( (possHerds,breeds) $ (sum((feedRegime,m),actherds(possHerds,breeds,feedRegime,t,m))  
 $ p\_OCoeff(possHerds,prodsYearly,breeds,t)),  
\*  
\* -- herd size in different month times output yearly coefficient (milk, young animals ..)  
\*  
 ( sum(actHerds(possHerds,Breeds,feedRegime,t,m),v\_herdSize(possHerds,breeds,feedRegime,t,nCur,m))  
 \* ( 1/min(12,p\_prodLength(possHerds,breeds))  
 \* ( (12/card(herdM)) $ (not sameas(possHerds,"fattners")) + 1 $ sameas(possHerds,"fattners")))  
 $ ( (p\_prodLength(possHerds,breeds) gt 1)  
$ifi "%farmBranchFattners%" == "on" or ((p\_prodLength(possHerds,breeds) le 1) $ sameas(possHerds,"fattners"))  
$ifi "%farmBranchSows%" == "on" and (p\_prodLength(possHerds,breeds) gt 2)  
 )  
 + sum(m $ sum(feedRegime,actherds(possHerds,breeds,feedRegime,t,m)), v\_herdStart(possHerds,breeds,t,nCur,m))  
 $ ( (p\_prodLength(possHerds,breeds) le 1)  
  
$ifi "%farmBranchFattners%" == "on" and (not (sum((feedRegime,m),actHerds("Fattners","",feedRegime,tCur,m))))  
$ifi "%farmBranchSows%" == "on" or (p\_prodLength(possHerds,breeds) le 2)  
 )  
 )  
  
 \* p\_OCoeff(possHerds,prodsYearly,breeds,t)  
 )  
$endif.herd  
 ;

# Manure

!!! abstract Manure excretion from animals is calculated based on fixed factors, differentiated by animal type, breed and feeding practice. For biogas production, the composition of different feed stock is taken into account. Manure is stored subfloor in stables and in silos. Application of manure has to follow legal obligations and interacts with plant nutrient need from the cropping module. Different N losses are accounted for in stable, storage and during application.

## Manure Excretion

With regard to excretion of animals, relevant equations and variables can be found in the *general\_herd\_module.gms*. *v\_manQuantM* is the monthly volume in cubic meter of manure produced. As a default, liquid manure is considered in the model. If a herd is utilising a straw stable, semi-solid and solid manure are excreted additionally. The monthly manure excretion, *v\_manQuantM*, is computed in the following equation:

manQuantM\_(curManChain(manChain),tCur(t),nCur,m) $(t\_n(t,nCur)$(not sameas(curManChain,"LiquidBiogas"))) ..  
  
 v\_manQuantM(manChain,t,nCur,m)  
  
 =e=  
 sum( actherds(possHerds,breeds,feedRegime,t,m1) $ manChain\_herd(curManChain,possHerds),  
 p\_manQuantMonth(possHerds,curManChain) \* ( 1 - 1 $ sameas(feedRegime,"fullGraz")  
 - 0.5 $ sameas(feedRegime,"partGraz"))  
 \* sum(m\_to\_herdm(m,m1), v\_herdSize(possHerds,breeds,feedRegime,t,nCur,m1)));

Furthermore, the monthly excretion of nutrients, NTAN (total ammonia nitrogen), Norg (organic nitrogen) and P (phosphorus) is calculated, multiplying *v\_herdsize* and *p\_nut2ManMonth*. For cows, excretion rate depends on animal category, feeding regime and yield level. For fatteners and sows excretion depends on animal category and feeding regime. Corresponding parameters can be found in *coeffgen\manure.gms* (not shown here). For dairy cows, excretion on pasture is subtracted. Depending on the stable inventory of a herd, manure excretion can be differentiated by liquid, semi-solid, and solid manure. If a herd is kept in a straw stables, the manure excretion is split among the different manure types accordingly.

nut2ManureM\_(curManChain(manChain),nut2,tCur(t),nCur,m) $(t\_n(t,nCur)$(not sameas(curManChain,"LiquidBiogas"))) ..  
  
 v\_nut2ManureM(manChain,nut2,t,nCur,m) =e=  
 sum(actherds(possHerds,breeds,feedRegime,t,m1)  
 $ ( (not sameas(feedRegime,"fullGraz"))  
 $ manChain\_herd(curManChain,possHerds)),  
 p\_nut2ManMonth(possHerds,feedRegime,nut2)  
 \* ( 1 - 1 $ sameas(feedRegime,"fullGraz")  
 - 0.5 $ sameas(feedRegime,"partGraz"))  
 \* sum(herd\_stableStyle(possHerds,stableStyles), p\_nutShare(manChain,stableStyles,nut2))  
 \* sum(m\_to\_herdm(m,m1), v\_herdSize(possHerds,breeds,feedRegime,t,nCur,m1)))  
 ;

Biogas production involves the production of digestates. Four feed sources can be differentiated depending on their origin: use of manure produced on farm, manure imported to the farm, crops grown on farm and crops imported to the farm. Manure produced on farm is treated as not fermented manure, as though it is not entering the biogas plant.

For digestates from imported manure and from crops, volume of digestates in cubic meter is calculated in the *biogas\_module.gms* by multiplying amount of used feed stock, *v\_usedCropBiogas* and *v\_purchManure*, and a fugal factor. The latter represents the decrease of volume during the fermentation process.

biogasVolCropDigestate\_(crm(biogasfeedM),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_volDigCrop(crM,t,nCur,m) =E= sum( (curBhkw(bhkw), curEeg(eeg)),  
 v\_usedCropBiogas(bhkw,eeg,crM,t,nCur,m)\* p\_fugCrop(crM));

biogasVolManDigestate\_(tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_volDigMan(t,nCur,m) =E= sum( (curBhkw(bhkw), curEeg(eeg), curmaM) ,  
 v\_purchManure(bhkw,eeg,curmaM,t,nCur,m) $ selPurchInputs(curmaM) \* p\_fugMan);

The amount of nutrients produced in the biogas plant and entering the manure storage is computed by multiplying the amount of feed stock and the corresponding nutrient content. It is assumed, that N and P is not lost during fermentation. Furthermore, nutrients from crop inputs are calculated as an annual average since no short term changes are common.

nutCropBiogasY\_(curmanchain,nut2,tCur(t),nCur) $ (t\_n(t,nCur) $ sameas(curmanchain,"LiquidBiogas")) ..  
  
 v\_nutCropBiogasY(curmanchain,nut2,t,nCur) =E=  
 sum( ( crM(biogasFeedM),m,curBhkw(bhkw), curEeg(eeg) ),  
 v\_usedCropBiogas(bhkw,eeg,crM,t,nCur,m)  
 \* p\_nutDigCrop(curmanchain,nut2,crM));

nutCropBiogasM\_(curmanchain,nut2,tCur(t),nCur,m) $(t\_n(t,nCur) $ sameas(curmanchain,"LiquidBiogas")) ..  
  
 v\_nutCropBiogasM("LiquidBiogas",nut2,t,nCur,m) =E= v\_nutCropBiogasY("LiquidBiogas", nut2,t,nCur) / card(m);

nut2ManurePurch\_(curmanchain,nut2,curmaM,tCur(t),nCur,m) $( t\_n(t,nCur) $ sameas(curmanchain,"LiquidBiogas")) ..  
  
 v\_nut2ManurePurch(curmanchain,nut2,curmaM,t,nCur,m)  
 =E= sum ( (curBhkw(bhkw), curEeg(eeg)),  
 v\_purchManure(bhkw,eeg,curmaM,t,nCur,m) \* p\_nut2manPurch("LiquidBiogas",nut2,curmaM) ) ;

## Manure Storage

Equations related to manure storage serve mainly for the calculation of the needed storage capacity, linked to investment, and for the calculation of emissions during storage. The equations related to manure storage in *manure\_module.gms* are activated when fatteners, sows, dairy and/or biogas is activated in the GUI.

The amount of manure in the storage in cubic meter is described in the following equation. Manure is emptied by field application, *v\_volManApplied*. When activated in the GUI, manure can also be exported from the farm.

volInStorage\_(curManChain(manChain),tCur(t),nCur,m) $ ( t\_n(t,nCur)$ ( not sameas (manchain,"LiquidImport")) ) ..  
  
 v\_volInStorage(manChain,t,nCur,m) =e= [sum(t\_n(t-1,nCur1) $ anc(nCur,nCur1),  
 v\_volInStorage(manChain,t-1,nCur1,"Dec")) $ (sameas(m,"Jan") $ tCur(t-1))  
 + v\_volInStorage(manChain,t,nCur,m-1) $ (not sameas(m,"Jan"))]  
  
  
\* ---- in comparative static setting, manure in Jan includes manure from Dec, assuming steady flow  
  
$iftheni.cs "%dynamics%" == "comparative-static"  
 + v\_volInStorage(manChain,t,nCur,"Dec") $ sameas(m,"Jan")  
$endif.cs  
  
  
$iftheni.herd %herd% == true  
\*  
\* --- m3 excreted per year divied by # of month: monthly inflow  
\*  
 + v\_manQuantM(manChain,t,nCur,m) $ (not sameas(manchain,"LiquidBiogas"))  
$endif.herd  
  
\* --- m3 coming from biogas plant s energy crops and purchased manure  
$iftheni.b %biogas% == true  
  
\* --- Diogas digestate based on energy crops  
  
 + sum(crm(biogasfeedM), v\_volDigCrop(crM,t,nCur,m)) $ sameas(manchain,"LiquidBiogas")  
  
\* --- Biogas digestate based on manure  
  
 + v\_volDigMan(t,nCur,m) $ sameas(manchain,"LiquidBiogas")  
$endif.b  
\*  
\* --- m3 taken out of storage type for application to crops  
\*  
 - v\_volManApplied(manChain,t,nCur,m)  
  
$iftheni.ExMan "%AllowManureExport%"=="true"  
  
\* --- m3 exported from farm  
  
 - sum (manChain\_Type(manChain,curManType), v\_manExport(manChain,curManType,t,nCur,m))  
$endif.ExMan  
  
$iftheni.emissionRight not "%emissionRight%"==0  
\* --- m3 exported through manure emission rights  
  
 - sum (manChain\_Type(manChain,curManType), v\_manExportMER(manChain,curManType,t,nCur,m))  
$endif.emissionRight  
 ;

Following the same structure as the equation above, there is a nutrient pool for NTAN, Norg and P in the storage. Losses of NTAN and Norg during storage are calculated in the environmental accounting and subtracted from the respective pool.

nutPoolInStorage\_(curManChain(manChain),nut2,tCur(t),nCur,m) $ ( t\_n(t,nCur)$ ( not sameas (manchain,"LiquidImport")) ) ..  
  
 v\_nutPoolInStorage(manChain,nut2,t,nCur,m)  
  
 =e= [sum(t\_n(t-1,nCur1) $ anc(nCur,nCur1),  
 v\_nutPoolInStorage(manChain,nut2,t-1,nCur1,"Dec")) $ (sameas(m,"Jan") $ tCur(t-1))  
 + v\_nutPoolInStorage(manChain,nut2,t,nCur,m-1) $ (not sameas(m,"Jan"))]  
  
  
\* ---- in comparative static setting, nutrient pool in Jan includes nutrient pool from Dec, assuming steady flow  
  
$iftheni.cs "%dynamics%" == "comparative-static"  
  
 + v\_nutPoolInStorage(manChain,nut2,t,nCur,"Dec") $ sameas(m,"Jan")  
  
$endif.cs  
  
  
  
$iftheni.herd %herd% == true  
  
 + v\_nut2ManureM(manChain,nut2,t,nCur,m) $ (not sameas(manchain,"LiquidBiogas"))  
  
$endif.herd  
  
$iftheni.biogas %biogas% == true  
  
 + sum( (curBhkw(bhkw),curEeg(eeg),curmaM), v\_nut2ManurePurch("LiquidBiogas",nut2,curmaM,t,nCur,m) ) $ sameas(manchain,"LiquidBiogas")  
  
 + v\_nutCropBiogasM("LiquidBiogas",nut2,t,nCur,m) $ sameas(manchain,"LiquidBiogas")  
$endif.biogas  
  
  
  
\* --- When environmental accounting is switched on, storage losses are calculated in envir\_acc\_module.gms  
  
 - v\_nutLossInStorage(manChain,nut2,t,nCur,m)  
  
$ontext  
  
\* --- When environmental accounting is switched off, only standard losses for NH3 are substracted from NTAN  
  
 $$iftheni.herd %herd% ==true  
  
 - v\_nut2ManureM(manChain,nut2,t,nCur,m) \* p\_nutLossFacNoEnvAcc $ ( not sameas(nut2,"P") )  
  
 $$endif.herd  
  
 $$iftheni.biogas %biogas% == true  
  
 - sum( (curBhkw(bhkw),curEeg(eeg),maM), v\_nut2ManurePurch(nut2,maM,t,nCur,m)  
 \* (1 - p\_nutEffectivDueVAlBiogasPurchMan(maM) )) $ ( not sameas(nut2,"P") )  
  
 - v\_nutCropBiogasM(nut2,t,nCur,m) \* (1 - p\_nutEffectivDueVAlBiogasPlantDig) $ ( not sameas(nut2,"P") )  
  
 $$endif.biogas  
  
$offtext  
  
\* --- Nutrients applied  
  
 - v\_nut2ManApplied(manChain,nut2,t,nCur,m)  
  
\* --- Nutrients exported from farm  
  
$iftheni.ExMan "%AllowManureExport%"=="true"  
  
 - v\_nut2export(manChain,nut2,t,nCur,m)  
  
$endif.ExMan  
$iftheni.emissionRight not "%emissionRight%"==0  
  
\* --- Nutrient exported via manure emission rights  
  
 - v\_nut2ExportMER(manChain,nut2,t,nCur,m)  
  
$endif.emissionRight  
 ;

Storage losses of reactive nitrogen are calculated in the equation *nutLossInStorage\_*, using emission factors from the environmental impact accounting module.

nutLossInStorage\_(curManChain(manChain),nut2,tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_nutLossInStorage(manChain,nut2,t,nCur,m) =E=  
  
\* --- NH3 losses in stable and storage, only related to N TAN  
  
 + v\_emissions(manChain,"staSto","NH3",t,nCur,m) $( sameas(nut2,"NTAN") $ (not sameas(manchain,"LiquidBiogas")))  
  
  
 $$iftheni.herd %herd% == true  
  
\* --- N2O, N2 and NO losses in stable and storage, related to NTAN and Norg  
  
 + v\_nut2ManureM(manChain,"NTAN",t,nCur,m) \* ( p\_EFStaSto("N2O",curManChain) + p\_EFStaSto("NOx",curManChain) + p\_EFStaSto("N2",curManChain) ) $(sameas(nut2,"NTAN") $(not sameas(manchain,"LiquidBiogas")))  
  
 + v\_nut2ManureM(manChain,"NOrg",t,nCur,m) \* ( p\_EFStaSto("N2O",curManChain) + p\_EFStaSto("NOx",curManChain) + p\_EFStaSto("N2",curManChain) ) $(sameas(nut2,"NOrg") $(not sameas(manchain,"LiquidBiogas")))  
  
  
 $$endif.herd  
  
\* --- N2O, N2 and NO losses from storage from digestate, related to NTAN and Norg  
  
 $$iftheni.biogas %biogas% == true  
  
 + [ ( v\_nutCropBiogasM(manchain,"NTAN",t,nCur,m) + sum (curmaM(mam), v\_nut2ManurePurch(manchain,"NTAN",curmaM,t,nCur,m) ) )  
 \* (p\_EFStaSto("N2O",curManChain) + p\_EFStaSto("NOx",curManChain) + p\_EFStaSto("N2",curManChain)) ] $( sameas(nut2,"NTAN") $ sameas(manchain,"LiquidBiogas"))  
  
 + [ ( v\_nutCropBiogasM(manchain,"NOrg",t,nCur,m) + sum (curmaM(mam), v\_nut2ManurePurch(manchain,"NOrg",curmaM, t,nCur,m) ) )  
 \*( p\_EFStaSto("N2O",curManChain) + p\_EFStaSto("NOx",curManChain) + p\_EFStaSto("N2",curManChain)) ] $( sameas(nut2,"NOrg") $ sameas(manchain,"LiquidBiogas"))  
  
 $$endif.biogas  
 ;

The amount of manure in the storage needs to fit to the available storage capacity which is calculated in the equation *totalManStorCap\_*. The total storage capacity is the sum of the sub floor storage in stables, silos and silos for digestates from biogas production. Note: when the biogas branch is active without herds, the storage concept is simplified.

totalManStorCap\_(curManChain(manChain),tCur(T),nCur) $ t\_n(t,nCur) ..  
  
 v\_TotalManStorCap(manChain,t,nCur) =e=  
  
$iftheni.herd %herd% == true  
 v\_SubManStorCap(manChain,t,nCur) $ (not sameas ("LiquidBiogas",manchain))  
 + v\_SiloManStorCap(manChain,t,nCur) $ (not sameas ("LiquidBiogas",manchain))  
$endif.herd  
$ifi %biogas% == true + v\_siloBiogasStorCap(t,nCur) $ sameas ("LiquidBiogas",manchain)  
 ;

The storage capacity of silos *v\_SiloManStorCap* is derived by multiplying the silo inventory with parameters characterising the corresponding storage capacity.

siloManStorCap\_(curManChain(manChain),tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_SiloManStorCap(manChain,t,nCur)  
  
 =e= sum(silos $ ( sum(t\_n(t1,nCur1) $ isNodeBefore(nCur,nCur1), v\_buySilos.up(manChain,silos,t1,nCur1))  
 or sum(tOld, p\_iniSilos(manChain,silos,tOld))),  
 v\_SiloInv(manChain,silos,t,nCur) \* p\_ManStorCapSi(silos)) ;

The subfloor storage capacity of stables *v\_SubManStorCap* is calculated in the *general\_herd\_module.gms*. The stable inventory is multiplied with parameters characterising the corresponding subfloor storage capacity. The amount of manure which can be stored in the stable building, *p\_ManStorCap*, depends on the stable system. Slurry based systems with a plane floor normally only have small cesspits which demand the addition of manure silo capacities. The manure storage capacity of stables with slatted floor depends on the size of the stable, where a storage capacity for manure of three month in a fully occupied stable is assumed. A set of different dimensioned liquid manure reservoirs is depicted in the parameter, *p\_ManStorCapSi*, ranging from 500 to 4000 m³.

subManStorCap\_(curManChain(manChain),tCur(t),nCur) $(t\_n(t,nCur)$(not sameas(curManChain,"LiquidBiogas"))) ..  
  
 v\_SubManStorCap(manChain,t,nCur) =e=  
 sum(stables $ ( sum( (t\_n(t1,nCur1),hor) $ isNodeBefore(nCur,nCur1),  
 v\_buyStables.up(stables,hor,t1,nCur1))  
 or (sum( (tOld,hor), p\_iniStables(stables,hor,tOld)))),  
  
 v\_StableInv(stables,"long",t,nCur)\*p\_ManStorCap(manChain,stables));

The storage capacity for digestates from biogas plants, *v\_siloBiogasStorCap*, is linked to the size of the biogas plant and calculated in the *biogas\_module.gms*.

invSiloBiogas\_(tCur(t), nCur) $ t\_n(t,nCur) ..  
  
 v\_siloBiogasStorCap(t,nCur) =E= sum((curbhkw(bhkw), curEeg(eeg)), v\_invBiogas(bhkw,eeg,t,ncur) \* p\_siloBiogas(bhkw));

The total volume is distributed to the different storage type based on the following equations.

storageDistr\_(curManChain(manChain),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_volInStorage(manChain,t,nCur,m) =e=  
  
 sum (manStorage,v\_volInStorageType(manChain,ManStorage,t,nCur,m)) ;

For the silos related to animal husbandry, different coverage of silos can be applied. The type of silo cover used for a certain type of silo, *v\_siCovComb*, is a binary variable, i.e. one type of silo must be fully covered or not.

siloCoverInv\_(curManChain(manChain),silos,tCur(t),nCur)  
 $ ( ( sum(t\_n(t1,nCur1) $ isNodeBefore(nCur,nCur1), v\_buySilos.up(manChain,silos,t1,nCur1))  
 or sum(tOld, p\_iniSilos(manChain,silos,tOld))) $ t\_n(t,nCur)) ..  
  
 v\_siloInv(manChain,silos,t,nCur) =e= sum(siloCover, v\_siCovComb(manChain,silos,t,nCur,siloCover));

The amount of storage capacity is prescribed by environmental law. FarmDyn allows applying different regulations with regard to required storage capacity, changed in the GUI. Thereby FarmDyn allows to precisely capture the requirements of the German Fertilisation Ordinance 2007 and 2017, which is further specified in the [Fertilisation Ordinance chapter](fertilization_ordinance.md#required-manure-storage-capacities).

The total manure storage capacity *v\_TotalManStorCap* must be greater than the required storage capacity *v\_ManStorCapNeed*.

manStorCap\_(curManChain(manChain),tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_TotalManStorCap(manChain,t,nCur) =g= v\_ManStorCapNeed(manChain,t,nCur);

Besides legal requirements for the storage capacity, there are equations which make sure that the storage is emptied in certain points of time. Every spring, the storage has to be emptied completely with regard to nutrients and volume, what is made sure of in the equations *emptyStorageVol\_* and *emptyStorageNut\_*. On the one hand, this represents typical manure management of farms. On the other hand, the restriction is necessary to make sure that the correct relation between mass and nutrients is maintained when nutrients and volume changes due to nutrient losses during storage (see chapter 2.9.3).

emptyStorageVol\_(curManChain(manChain),tCur(t),nCur,m) $(sameas(m,"apr") $ t\_n(t,nCur)) ..  
  
 v\_volInStorage(manChain,t,nCur,m) =L= 0;

emptyStorageNut\_(curManChain(manChain),nut2,tCur(t),nCur,m) $(sameas(m,"apr") $ t\_n(t,nCur)) ..  
  
  
 v\_nutPoolInStorage(manChain,nut2,t,nCur,m) =L= 0;

Furthermore, at the end of the time period modelled, only 1/3 of the annual excreted manure is allowed to remain in the storage, to avoid unrealistic behaviour in the last year modelled. This is made sure in the following equations.

maxManVolStorLastMonth\_(curManChain(manChain),"%lastYear%",nCur,"Dec") $ t\_n("%lastYear%",nCur) ..  
  
 (  
  
$ifi %herd% == true v\_manQuant(manChain,"%lastYearCalc%",nCur)$ (not sameas(manchain,"LiquidBiogas"))  
  
$ifi %biogas% == true + sum((crm(biogasFeedM),m), v\_voldigCrop(crM,"%lastYearCalc%",nCur,m)+ v\_volDigMan("%lastYearCalc%",nCur,m) ) $ sameas(manchain,"LiquidBiogas")  
  
 ) \* 8/12  
  
 =G= v\_volInStorage(manChain,"%lastYear%",nCur,"Dec");

maxManNutStorLastMonth\_(curManChain(manChain),nut2,"%lastYear%",nCur,"Dec") $ t\_n("%lastYear%",nCur) ..  
  
 (  
  
$ifi %herd% ==true sum(m, v\_nut2ManureM(manChain,nut2,"%lastYear%",nCur,m) $ (not sameas(manchain,"LiquidBiogas")))  
  
$iftheni.b %biogas% == true  
  
 + sum((curmaM,m), v\_nut2ManurePurch(manchain,nut2,curmaM,"%lastYear%",nCur,m) ) $ sameas(manchain,"LiquidBiogas")  
  
 + sum(m, v\_nutCropBiogasM("LiquidBiogas",nut2,"%lastYear%",nCur,m)) $ sameas(manchain,"LiquidBiogas")  
  
  
$endif.b  
 ) \* 8/12  
  
 =G= v\_nutPoolInStorage(manChain,nut2,"%lastYear%",nCur,"Dec");

## Manure Application

Different application procedures for manure N are implemented, *ManApplicType*, including broad spread, drag hose spreader, injection of manure, and solid manure spread. The core variable is *v\_mandist* that represents the amount of manure in distributed cubic meter. The different techniques are related to different application costs, labour requirements as well as effects on different emissions. Furthermore, manure application is linked to the nutrient balance (see chapter 2.11.2 and 2.11.3) and the manure storage (see chapter 2.9.2).

The application of manure links nutrient with volumes. The nutrient content of the manure is depending on the herd's excretion as well as on the losses during storage. The parameter *p\_nut2inMan* contains the amount of NTAN, Norg and P per cubic meter of manure applied. The parameter is differentiated for the manure types linked to the present herd. Relevant parameters are calculated in *coeffgen\manure.gms*.

As a first step, the amount of different nutrients per cubic meter without losses is calculated in *p\_nut2inManNoLoss*. Here, the nutrient excretion of the animals is related to their volume excretion depending on the stables present on the farm.

In a second step, the nutrients per cubic meter are corrected for the storage losses in *p\_nut2inMan*. Varying storage time of manure, and hence varying nutrient content, can be taken into account by activating the "Nutrient loss depending on storage time" control in the GUI. In this case, for the manure of every herd, two types of manure are calculated, representing the maximum and minimum possible amount of losses during one year. This allows a complete emptying of the storage in a linear programming setting. In the default case, only the minimum losses are assumed.

The total manure distributed in cubic meter and in nutrients per month is summarised in the following equations according to:

nut2ManApplied\_(curManChain(manChain),nut2,tCur(t),nCur,m) $ ((v\_volManApplied.up(manChain,t,nCur,m) ne 0) $ t\_n(t,nCur)) ..  
  
 v\_nut2ManApplied(manChain,nut2,t,nCur,m) =e= sum( (c\_s\_t\_i(curCrops(crops),plot,till,intens),  
 manChain\_applic(manChain,ManApplicType),curManType)  
 $ (manApplicType\_manType(ManApplicType,curManType)  
 $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m) ne 0)  
 $ (not sameas (curCrops,"catchcrop")) ),  
  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m)  
 \* p\_nut2inMan(nut2,curManType,manChain));

volManApplied\_(curManChain(manChain),tCur(t),nCur,m) $ ((v\_volManApplied.up(manChain,t,nCur,m) ne 0) $ t\_n(t,nCur)) ..  
  
 v\_volManApplied(manChain,t,nCur,m)  
 =e= sum( (c\_s\_t\_i(curCrops(crops),plot,till,intens),  
 manChain\_applic(manChain,ManApplicType),curManType)  
 $ (manApplicType\_manType(ManApplicType,curManType)  
 $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m) ne 0)  
 $ (not sameas (curCrops,"catchcrop")) ),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m));

There are several restrictions with regard to the application of manure. First of all, the application of manure is not possible in some crops in some months, e.g. in maize at certain height of growth.

set doNotApplyManure(crops,m) /  
 (potatoes,sugarbeet,maizSil) .(Jun,Jul,Aug)  
 (WinterWheat,SummerBeans) .(Apr,May,Jun,Jul,Aug)  
 (WinterBarley) .(Apr,May,Jun,Jul)  
 (SummerPeas,SummerCere,WinterRape) .(May,Jun,Jul)  
 (WheatGPS) .(May,Jun)  
 (MaizCorn, MaizCCM) .(Jun,Jul,Aug)  
 /;

For these months, *v\_manDist* is forced to be zero.

# Synthetic Fertilisers

To meet the N and P demand of crops, synthetic fertiliser, *v\_syntDist*, can be applied besides manure. Synthetic fertiliser application enters equations with regard to the buying of inputs, *buy\_* and *varcost\_*, the labour need for application, *labCropSM\_*, the field work hours and machinery, *fieldWorkHours\_* and *machines\_*, and with regard to plant nutrition (see chapter 2.11). The equation *nMinMan\_* makes sure that minimum amounts of mineral fertiliser are applied for certain crops. It represents the limitation meeting the plant need with nutrients from manure, e.g. fertilising short before harvest for baking wheat cannot be done with manure.

nMinMin\_(c\_s\_t\_i(curCrops(crops),plot,till,intens),nut,tCur(t),nCur)  
 $ ( (v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0)  
 $ p\_minChemFert(crops,nut) $ (not sameas(till,"eco")) $ t\_n(t,nCur) ) ..  
  
 sum ((syntFertilizer,m),  
 v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m)  
 \* p\_nutInSynt(syntFertilizer,nut))  
 =G=  
  
\* sum(plot\_soil(plot,soil),  
\* p\_nutNeed(crops,soil,till,intens,nut,t))  
 v\_cropHa(crops,plot,till,intens,t,nCur) \* p\_minChemFert(crops,nut);

# Plant Nutrition

!!! abstract The equations related to plant nutrition make sure that the nutrient need of crops is met. Nutrient need can be derived from N response functions or from planning data for fixed yield levels. Furthermore, FarmDyn is loosely connected to the crop modelling framework SIMPLACE which provides data on cropping activities. Needed nutrients are provided by manure and synthetic fertiliser.

The template supports two differently detailed ways to account for plant nutrition need.

1. A **fixed factor approach** with yearly nutrient balances per crop
   1. Using N response curves
   2. Using planning data
2. Using data output of the crop modelling framework SIMPLACE

*p\_nutNeed* is the nutrient need for different crops tat enters the equation for fixed factor approach and the flow model. For the fixed factor approach, nutrient need can be calculated based on N response curves and alternatively based on planning data. In the detailed flow model, nutrient need is calculated based on N response curves. All relevant calculations can be found in *coeffgen\cropping.gms*.

## The fixed factor approach

The fixed factor approach is used in combination with the use of N response curves and planning data. Generally, the plant need in *p\_nutneed* has to be met with manure and synthetic fertiliser. There is the option to allow manure application over plant need as manure nutrients on livestock farms with high stocking densities partly treated as waste.

NutBalCrop\_(c\_s\_t\_i(curCrops(crops),plot,till,intens),nut,tCur(t),nCur)  
 $ ((v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0) $ t\_n(t,nCur) $( not sameas (crops,"catchCrop")) ) ..  
  
\* --- crop need based on plant uptake and calculated further need  
  
 sum(plot\_soil(plot,soil),  
 p\_nutNeed(crops,soil,till,intens,nut,t) \* v\_cropHa(crops,plot,till,intens,t,nCur)  
 \* (1 + p\_nutLossUnavoidable(soil,till,intens,nut)))  
  
 $$iftheni.man %manure% == true  
\* --- application over plant need of organic fertilizer is possible  
 + v\_nutOrganicOverNeed(crops,plot,till,intens,nut,t,nCur)  
 $$endif.man  
  
 =E=  
  
 $$iftheni.dh "%cattle%" == "true"  
\*  
\* --- manure excreted during grazing on pasture: N , different calculation of losses [TK 01.03.16 revised]  
\*  
 [sum( (nut2,m) $ ( sameas(nut2,"norg") or sameas(nut2,"ntan") ),  
\*  
\* --- excretion by herds which graze only for a part of the year  
\*  
 v\_nut2ManurePast(crops,plot,till,intens,nut2,t,nCur,m)  
 )  
 $$iftheni.NorgAcc "%NorgAccounting%" == "Interface"  
 \* %NOrgAccountedInt%  
 $$elseifi.NorgAcc "%NorgAccounting%" == "PlanningDueV16"  
 \* 0.8  
 $$else.NorgAcc  
\*  
\* WB: here, something needs to change ... cannot work with several pasture options  
 - v\_niEmissionsPast(crops,plot,till,intens,t,nCur)  
  
 $$ontext  
 \* p\_nutEffectivPastDueVNv  
 $$offtext  
 $$endif.NorgAcc  
  
 ] $ (past(crops) and sameas(nut,"N"))  
  
\* --- manure excreted during grazing pasture: P [TK 01.03.16 revised]  
  
 + sum(m,v\_nut2ManurePast(crops,plot,till,intens,"P",t,nCur,m)) $ (past(Crops) and sameas(nut,"P"))  
  
 $$endif.dh  
  
  
 $$iftheni.man "%manure%" == "true"  
  
\*  
\* -- application of N and P with organic fertilizer [TK 09.02.15 revised]  
\*  
\* + sum( (nut2\_nut(nut2,nut),manApplicType\_manType(ManApplicType,curManType),m)  
\* $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m) ne 0),  
\* v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m)  
\* \* sum(manChain\_applic(manChain,ManApplicType), p\_nut2inMan(nut2,curManType,manChain))  
\* \* p\_nut2UsableShare(crops,curManType,ManApplicType,nut2,m))  
  
  
 + sum( (nut2\_nut(nut2,nut),manApplicType\_manType(ManApplicType,curManType),m)  
 $(not sameas(plot,"plot7") $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m) ne 0 )),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m)  
 \* sum(manChain\_applic(manChain,ManApplicType), p\_nut2inMan(nut2,curManType,manChain))  
 \* p\_nut2UsableShare(crops,curManType,ManApplicType,nut2,m))  
  
  
 $$endif.man  
  
\* -- mineral N application  
  
 + sum ((syntFertilizer,m)$(not sameas(plot,"plot7")),  
 v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m)  
 \* p\_nutInSynt(syntFertilizer,nut) )  
 ;

## N response curves

The yield level of different crops is chosen in the GUI. The following equations show, using the example of winter cereals, that the yield, *p\_OCoeffC*, equals the yield given by the GUI, *p\_cropYieldInt* , and takes a growth rate given by the GUI into account.

p\_OCoeffC("winterCere",soil,till,intens,"winterCere",t) $ sum(soil\_plot(soil,plot),c\_s\_t\_i("winterCere",plot,till,intens)) = 8 \* (1.00 + p\_cropYieldInt("winterCere","GrowthRateY")/100) \*\*t.pos;

In the next step, the nutrient needs for crops are linked to the different cropping intensities. There are five different intensity levels with regard to the amount of N fertiliser applied:

set intens / normal "Full N fertilization"  
 fert80p "80 % N"  
 fert60p "60 % N"  
 fert40p "40 % N"  
 fert20p "20 % N"  
  
 bales  
 silo  
 Graz  
  
 /;

These nutrient needs for the different intensities are based on nitrogen response functions from field trials. The intensity can be reduced from 100 % to an N fertiliser application of 80 %, 60 %, 40 % and 20 %. The yield level is reduced to 96 %, 90 %, 82 % and 73 %, respectively. These steps reflect the diminishing yield increases from increased N fertiliser application.

p\_OCoeffC(arabCrops,soil,till,"fert80p",prods,t) $ (not sameas(till,"eco")) = p\_oCoeffC(arabCrops,soil,till,"normal",prods,t) \* 0.96;  
 p\_OCoeffC(arabCrops,soil,till,"fert60p",prods,t) $ (not sameas(till,"eco")) = p\_oCoeffC(arabCrops,soil,till,"normal",prods,t) \* 0.90;  
 p\_OCoeffC(arabCrops,soil,till,"fert40p",prods,t) $ (not sameas(till,"eco")) = p\_oCoeffC(arabCrops,soil,till,"normal",prods,t) \* 0.82;  
 p\_OCoeffC(arabCrops,soil,till,"fert20p",prods,t) $ (not sameas(till,"eco")) = p\_oCoeffC(arabCrops,soil,till,"normal",prods,t) \* 0.73;  
  
 p\_OCoeffC(arabCrops,soil,till,"fert80p",prods,t) $ (not sameas(till,"eco")) = p\_oCoeffC(arabCrops,soil,till,"normal",prods,t) \* 0.95;  
 p\_OCoeffC(arabCrops,soil,till,"fert60p",prods,t) $ (not sameas(till,"eco")) = p\_oCoeffC(arabCrops,soil,till,"normal",prods,t) \* 0.85;  
 p\_OCoeffC(arabCrops,soil,till,"fert40p",prods,t) $ (not sameas(till,"eco")) = p\_oCoeffC(arabCrops,soil,till,"normal",prods,t) \* 0.71;  
 p\_OCoeffC(arabCrops,soil,till,"fert20p",prods,t) $ (not sameas(till,"eco")) = p\_oCoeffC(arabCrops,soil,till,"normal",prods,t) \* 0.53;

The output coefficient, *p\_OCoeffC*, represents the yields per hectare. It is used to define the nutrient uptake by the crops, *p\_nutNeed,* based on the nutrient content, *p\_nutContent*. Values for *p\_nutContent* are taken from the German Fertiliser Directive (DüV 2007, Appendix 1).

p\_nutNeed(crops,soil,till,intens,nut,t) $ sum(soil\_plot(soil,plot), c\_s\_t\_i(crops,plot,till,intens))  
 = sum( prods, p\_OCoeffC(crops,soil,till,intens,prods,t) \* (p\_nutContent(crops,prods,nut)\*10));

For different intensities, the corresponding amount of nutrient applied has to fulfil the need *p\_nutNeed*.

The parameter *p\_basNut* defines the amount of nutrients coming from other sources than directly applied fertiliser, for example mineralization and atmospheric deposition. The curve suggests that for a 53%-level of yield, only 20% of the N dose at full yield is necessary. Assuming a minimum nutrient loss factor that allows defining how much N a crop takes up from other sources:

p\_basNut(crops,soil,till,nut,t) $ (sum(prods, p\_OCoeffC(crops,soil,till,"normal",prods,t)) $ sameas(nut,"N"))  
 = smax( (soil\_plot(soil,plot),c\_s\_t\_i(crops,plot,till,"normal"),intens),  
 sum(prods, p\_OCoeffC(crops,soil,till,intens,prods,t) \* (p\_nutContent(crops,prods,nut)\*10))  
 - p\_nutNeed(crops,soil,till,"normal",nut,t)\*(1 + p\_FracGaseF + p\_FracLeach) \* ( 0.2 $ sameas(intens,"fert20p")  
 + 0.4 $ sameas(intens,"fert40p")  
 + 0.6 $ sameas(intens,"fert60p")  
 + 0.8 $ sameas(intens,"fert80p")  
 + 1.0 $ sameas(intens,"normal")) );

The amount of nutrients applied, *p\_nutApplied,* is estimated as shown in the following equation. It is assumed that at least 20% of the default leaching and NH3 losses will occur.

p\_nutApplied(crops,soil,till,"fert20p","N",t) $ sum(soil\_plot(soil,plot),c\_s\_t\_i(crops,plot,till,"fert20p"))  
 = p\_nutNeed(crops,soil,till,"normal","N",t)\*(1 + p\_FracGaseF + p\_FracLeach)\*0.2;

p\_nutApplied(crops,soil,till,"fert40p","N",t) $ sum(soil\_plot(soil,plot),c\_s\_t\_i(crops,plot,till,"fert40p"))  
 = p\_nutNeed(crops,soil,till,"normal","N",t)\*(1 + p\_FracGaseF + p\_FracLeach)\*0.2 \* 1.5;

p\_nutApplied(crops,soil,till,"fert60p","N",t) $ sum(soil\_plot(soil,plot),c\_s\_t\_i(crops,plot,till,"fert60p"))  
 = p\_nutNeed(crops,soil,till,"normal","N",t)\*(1 + p\_FracGaseF + p\_FracLeach)\*0.2 \* 2;

p\_nutApplied(crops,soil,till,"fert80p","N",t) $ sum(soil\_plot(soil,plot),c\_s\_t\_i(crops,plot,till,"fert80p"))  
 = p\_nutNeed(crops,soil,till,"normal","N",t)\*(1 + p\_FracGaseF + p\_FracLeach)\*0.2 \* 2.5;

p\_nutApplied(crops,soil,till,"normal","N",t) $ sum(soil\_plot(soil,plot),c\_s\_t\_i(crops,plot,till,"normal") )  
 = p\_nutNeed(crops,soil,till,"normal","N",t)\*(1 + p\_FracGaseF + p\_FracLeach)\*0.2 \* 3;

The nutrient application, *p\_nutApplied,* in combination with the basis delivery from soil and air, *p\_basNut,* allows defining the loss rates for each intensity level, *p\_nutLossUnavoidable,* as the difference between the deliveries and the nutrient uptake, *p\_nutNeed,* by the plants:

p\_nutLossUnavoidable(soil,till,intens,nut)  
 $ (sum( (crops,t) $ p\_nutNeed(crops,soil,till,intens,nut,t), 1))  
 = sum( (crops,t) $ p\_nutNeed(crops,soil,till,intens,nut,t),  
 max(0,Min(50, p\_nutApplied(crops,soil,till,intens,nut,t)  
 + p\_basNut(crops,soil,till,nut,t) - p\_nutNeed(crops,soil,till,intens,nut,t)))  
 / p\_nutNeed(crops,soil,till,intens,nut,t))  
  
 /sum( (crops,t) $ p\_nutNeed(crops,soil,till,intens,nut,t), 1);

*p\_nutLossUnavoidable* enters the Standard Nutrient Fate Model (see chapter 2.11.2). It represents the factor that has to be applied over the plant removal, *p\_nutNeed*, to reach a certain yield level. It indicates the nutrient efficiency of the fertiliser management.

## Planning Data

The nutrient need can also be derived from planning data from the revised Fertiliser Directive (BMEL 2015). The proposed directive includes compulsory fertiliser planning to increase N use efficiency on farms. This measure is included in FarmDyn. When fertiliser management follows the planning data, different intensities do not exist, and yield levels are fixed, i.e. cannot be changed by the GUI.

The yield level *p\_OCoeffC* is fixed in the following equation, showing the example of winter cereals.

p\_OCoeffC("winterWheat",soil,till,intens,"winterWheat",t) $ sum(soil\_plot(soil,plot),c\_s\_t\_i("winterWheat",plot,till,intens)) = 8 ;

The yield corresponds to a certain amount of needed N, *p\_nutNeed*, given by the directive.

p\_nutNeed("winterWheat",soil,till,intens,"N",t) $ sum(soil\_plot(soil,plot), c\_s\_t\_i("winterWheat",plot,till,intens)) = 230 - p\_basNut("winterWheat",soil,till,"N",t) ;

In the case of P, it is assumed that the nutrient need corresponds to the nutrients removed by the harvested product.

p\_nutNeed(crops,soil,till,intens,"P",t) $ sum(soil\_plot(soil,plot), c\_s\_t\_i(crops,plot,till,intens))  
 = sum( prods, p\_OCoeffC(crops,soil,till,intens,prods,t) \* (p\_nutContent(crops,prods,"P")\*10));

The directive prescribes that nutrients delivered from soil and air have to be taken into account. This reduces the amount of fertiliser that needs to be applied, i.e. p\_nutNeed is lowered. `

p\_basNut(crops,soil,till,"N",t) $ arableCrops(crops) = 50 ;  
 p\_basNut(crops,soil,till,"N",t) $ grassCrops(crops) = 10 + p\_NfromLegumes(crops);

## Using data output of the crop modelling framework SIMPLACE

FarmDyn is loosely connected to the crop modelling framework [SIMPLACE](http://www.simplace.net/Joomla/index.php). This crop model provides cropping activities consisting of different managements and corresponding yields and externalities. They are provided as a gdx file and loaded into FarmDyn. The parameter *p\_simres* contains all information from the crop model for different crops, crop rotations (represented in the set *till*) and intensities. Intensities represent a whole range of management, consisting of different amounts of fertiliser, straw removal and catch crop growing. The elements of the set contain the information on yields and externalities for the different cropping activities. The use of the SIMPLACE data is activated in the GUI by selecting the BWA mode. It requires to choose specific farm types and their location in different soil-climate regions. Currently, SIMPLACE data are available for the German Federal State of North Rhine-Westphalia.

First, the shares of different crops in FarmDyn have to equal the crop rotation represented in the SIMPLACE data. Crop rotations can be selected at the GUI.

SimplaceRot\_(c\_s\_t\_i(curCrops(crops),plot,curRotTill(till),intens),tCur(t),nCur)  
 $ ( (v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0) $ t\_n(t,nCur)  
 $ (not sameas (crops,"idle") )  
 $ (not sameas (crops,"catchcrop") ) ) ..  
  
 v\_cropHa(crops,plot,till,intens,t,nCur)  
  
 =e=  
  
 sum ( crops1 $ (c\_s\_t\_i(crops1,plot,till,intens) $ (curCrops(crops1)  
 $ (not sameas(crops1,"idle"))  
 $ (not sameas(crops1,"catchcrop")))),  
 v\_cropHa(crops1,plot,till,intens,t,nCur)) \* p\_cropShare(till,crops);

The synthetic fertiliser need linked to cropping activities in the SIMPLACE data has to be provided by synthetic fertiliser distribution in FarmDyn.

NMineralSim\_(c\_s\_t\_i(curCrops(crops),plot,curRotTill(till),intens),"N",tCur(t),nCur)  
 $ ((v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0) $ t\_n(t,nCur) $ (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle") ) ) ..  
  
 sum ( (syntFertilizer,m) , v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m) \* p\_nutInSynt(syntFertilizer,"N")  
 \* (1 - p\_EFApplMinNH3(syntFertilizer) - p\_EFApplMin("N2O") - p\_EFApplMin("NOx") ) )  
  
 =e=  
  
 v\_cropHa(crops,plot,till,intens,t,nCur) \* p\_SimRes(till,crops,intens,"Nchem")  
;

The cropping activities in the SIMPLACE data correspond to the month January to June. Accordingly, manure application has to be conducted in those months. Note that other restrictions such as the Fertilisation Ordinance may restrict application in certain months.

NOrgSpringSim\_(c\_s\_t\_i(curCrops(crops),plot,curRotTill(till),intens),"N",tCur(t),nCur)  
 $ ((v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0) $ t\_n(t,nCur) $ (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle") ) ) ..  
  
  
\* --- NOrg Applied  
  
 sum( (manApplicType\_manType(ManApplicType,curManType),m\_spring(m) )  
 $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m\_spring) ne 0),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m\_spring )  
 \* sum(manChain\_applic(manChain,ManApplicType), p\_nut2inMan("NOrg",curManType,manChain)) \* (1 - ( p\_EFApplMin("N2O") + p\_EFApplMin("NOx"))) )  
  
\* -- NTAN applied minus losses with application  
  
 + sum( (manApplicType\_manType(ManApplicType,curManType),m\_spring(m) )  
 $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m\_spring) ne 0),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m\_spring)  
 \* sum(manChain\_applic(manChain,ManApplicType), p\_nut2inMan("NTan",curManType,manChain))  
 \* (p\_nut2UsableShare(crops,curManType,ManApplicType,"NTAN",m) - ( p\_EFApplMin("N2O") + p\_EFApplMin("NOx")) )  
 )  
 =e=  
  
 v\_cropHa(crops,plot,till,intens,t,nCur) \* p\_SimRes(till,crops,intens,"NOrgS")  
  
 ;

In line with manure application in spring, the manure application in autumn linked to cropping activities in the SIMPLACE data has to be provided by manure application in FarmDyn in the months July to December.

NOrgAutumnSim\_(c\_s\_t\_i(curCrops(crops),plot,curRotTill(till),intens),"N",tCur(t),nCur)  
 $ ((v\_cropHa.up(crops,plot,till,intens,t,nCur) ne 0) $ t\_n(t,nCur)  
 $ (not sameas (crops,"catchcrop") ) $ (not sameas (crops,"idle") ) ) ..  
  
  
\* --- NOrg Applied  
  
 sum( ( manApplicType\_manType(ManApplicType,curManType),m\_autumn(m) )  
 $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m\_autumn) ne 0),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m\_autumn )  
 \* sum(manChain\_applic(manChain,ManApplicType), p\_nut2inMan("NOrg",curManType,manChain)) \* (1 - ( p\_EFApplMin("N2O") + p\_EFApplMin("NOx"))) )  
  
\* -- NTAN applied minus losses with application  
  
 + sum( ( manApplicType\_manType(ManApplicType,curManType),m\_autumn(m) )  
 $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m\_autumn) ne 0),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m\_autumn)  
 \* sum(manChain\_applic(manChain,ManApplicType), p\_nut2inMan("NTan",curManType,manChain))  
 \* ( p\_nut2UsableShare(crops,curManType,ManApplicType,"NTAN",m) - ( p\_EFApplMin("N2O") + p\_EFApplMin("NOx")) )  
 )  
  
 =e=  
  
 v\_cropHa(crops,plot,till,intens,t,nCur) \* p\_SimRes(till,crops,intens,"NOrgA") ;

The cropping activities provided by SIMPLACE do not contain information on P2O5 fertiliser need. Therefore, the following equation ensures that P2O5 removal with the harvested product has to be meet by P2O5 from manure and chemical fertiliser.

PFertilizingSim\_("P",tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 sum( (prods,c\_s\_t\_i(curcrops(crops),plot,till,intens)) $ ( ( not sameas (prods,"WCresidues")) $ ( not sameas (prods,"WBresidues")) $ (not sameas (prods,"SCresidues"))  
 $ (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle") ) )  
 , p\_SimRes(till,crops,intens,"yield") \* p\_nutContent(crops,prods,"P") \* 10/1000  
 \* v\_cropHa(crops,plot,till,intens,t,nCur) )  
  
 =l=  
  
$iftheni.man %manure% == true  
  
 sum( (manApplicType\_manType(ManApplicType,curManType),m,c\_s\_t\_i(curCrops(crops),plot,till,intens)) $ ( (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle") )  
 $ ( v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m) ne 0) ),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m)  
 \* sum(manChain\_applic(manChain,ManApplicType), p\_nut2inMan("P",curManType,manChain))  
  
 )  
  
 +  
  
$endif.man  
  
 sum ( (syntFertilizer,m,c\_s\_t\_i(curcrops(crops),plot,till,intens) ) $( (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle") ) )  
 , v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m) \* p\_nutInSynt(syntFertilizer,"P") )  
 ;

Some crops require minimum chemical fertiliser doses such as the starter fertilisation of maize. For N, minimum chemical fertiliser needs are reflected in the SIMPLACE results. For P2O5, the following equations ensures that the minimum chemical fertiliser needs is met.

MinChemFertSimplace\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 sum( (c\_s\_t\_i(curcrops(crops),plot,till,intens)) $ ( (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle") ) )  
 , v\_cropHa(crops,plot,till,intens,t,nCur) \* p\_minChemFert(crops,"P")  
 )  
  
 =l=  
  
 sum ( (syntFertilizer,m,c\_s\_t\_i(curcrops(crops),plot,till,intens) ) $ ( (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle") ) )  
 , v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m) \* p\_nutInSynt(syntFertilizer,"P") )  
 ;

The SIMPLACE results contain scenarios, captured in the set intensities, with and without residue removal. Thereby, it is assumed that straw from cereal production can be sold. The following equation maps the cropping activities on the variable *v\_residuesRemoval* which is used in other parts of FarmDyn to calculate the costs and revenues related to residue removal.

ResidRemovalSim\_(curCrops(crops),plot,till,intens,tCur(t),nCur)  
 $ ( t\_n(t,nCur) $ c\_s\_t\_i(crops,plot,till,intens) $ (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle"))  
 $ intensResRem(intens) $ cropsResidueRemo(crops) ) ..  
  
 v\_residuesRemoval(crops,plot,till,intens,t,nCur) =e= v\_cropHa(crops,plot,till,intens,t,nCur) ;

The SIMPLACE results contain scenarios, captured in the set intensities, with and without catch crops. They are linked to the catch crop growing represented in *v\_cropHa*.

CatchCropsSimHa\_(plot,curRotTill(till),intens,tCur(t),nCur)  
 $ ( t\_n(t,nCur) $ (sum (crops, c\_s\_t\_i(crops,plot,till,intens))) ) ..  
  
 sum (c\_s\_t\_i("catchCrop",plot,till,intens), v\_cropHa("catchCrop",plot,till,intens,t,nCur) )  
  
 =e=  
 sum( c\_s\_t\_i(curCrops(crops),plot,till,intens) $ intensCatchCro(intens),  
 v\_cropHa(crops,plot,till,intens,t,nCur)  
 \* p\_SimRes(till,crops,intens,"catCroShare") )  
 ;

The SIMPLACE results contain nitrate leaching for the different cropping activities. This externality is summarised in the following equation for the environmental accounting in FarmDyn.

NleachSim\_(curCrops(crops),tCur(t),nCur)  
 $ ( t\_n(t,nCur) $ sum ( (plot,till,intens), c\_s\_t\_i(crops,plot,till,intens)) $ (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle") ) ) ..  
  
  
 v\_NleachSim(crops,t,nCur)  
  
 =e=  
  
 sum( c\_s\_t\_i(curCrops,plot,curRotTill,intens), v\_cropHa(crops,plot,curRotTill,intens,t,nCur)  
 \* p\_SimRes(curRotTill,crops,intens,"Nleach") ) ;

Furthermore, the SIMPLACE results contain a N balance which is summarised in the following equation. Please note that the calculation of this balance differs from the balance calculation under the Fertilisation Ordinance.

NSurplusSim\_(curCrops(crops),tCur(t),nCur)  
 $ ( t\_n(t,nCur) $ sum ( (plot,till,intens), c\_s\_t\_i(crops,plot,till,intens)) $ (not sameas (curCrops,"catchcrop") ) $ (not sameas (curCrops,"idle") ) ) ..  
  
  
 v\_NSurplusSim(crops,t,nCur)  
  
 =e=  
  
 sum( c\_s\_t\_i(curCrops,plot,curRotTill,intens), v\_cropHa(crops,plot,curRotTill,intens,t,nCur)  
 \* p\_SimRes(curRotTill,crops,intens,"NSur") ) ;

fertiliser # Fertilisation Ordinance

!!! abstract The German Fertilisation Ordinance implements the EU Nitrates Directive in Germany together with other environmental regulations. It consists of numerous measures which prescribe how farmers are allowed to use nutrients from manure and chemical fertiliser along with further management specifications. The most prominent measures of the Fertilisation Ordinance are included in FarmDyn, being (1) nutrient balance restrictions, (2) an organic nitrogen application threshold, (3) required manure storage capacities, (4) banning periods for fertiliser application, (5) restrictions of fertiliser application in autumn, (6) a binding fertiliser planning, (7) compulsory low-emission manure application techniques.

The equations regarding the Fertilisation Ordinance are mainly found in the Fertilisation Ordinance module (*duev\_module.gms*). Measures with regard to the storage capacity are partly found in the *manure module*. FarmDyn is used to asses the revision of the Fertilisation Ordinance in 2017. Therefore, the Ordinance from 2007 and 2017 can be directly selected in the GUI to activate the corresponding measures. In addition, thresholds and requirements can be modified separately in the GUI.

## Nutrient balance restrictions

The German Fertilisation Ordinance requires that farms calculate a nutrient balance on an annual basis for nitrogen and phosphate (DüV 2007;DüV 2007). This balance combines nutrient inputs via manure and synthetic fertiliser with nutrient removal via the harvested crops. The surplus, i.e. the balance, is not allowed to exceed a certain threshold.

Nutrient removal via harvested product is calculated, depending on its yield level and nutrient content. The main harvested product, as well as straw from cereal production which can be sold in FarmDyn, are covered.

nutRemovalDuev\_(nut,tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_nutRemovalDuev(nut,t,nCur)  
 =e=  
  
 sum( (c\_s\_t\_i(crops,plot,till,intens)), v\_cropHa(crops,plot,till,intens,t,nCur)  
 \* sum( (plot\_soil(plot,soil),curProds), p\_OCoeffC(crops,soil,till,intens,curProds,t)  
 \* p\_nutContent(crops,curProds,nut)\*10 ) )  
  
  
 + sum( (c\_s\_t\_i(crops,plot,till,intens)) $ cropsResidueRemo(crops), v\_residuesRemoval(crops,plot,till,intens,t,nCur)  
 \*sum( (plot\_soil(plot,soil),curProds), p\_OCoeffResidues(crops,soil,till,intens,curProds,t)  
 \* p\_nutContent(crops,curProds,nut) \* 10 ) )  
  
 ;

Nutrient input via synthetic fertiliser is calculated.

synthAppliedDueV\_(nut,tCur(t),nCur) $ t\_n(t,nCur)..  
  
 v\_synthAppliedDueV(nut,t,nCur) =e=  
  
 sum( (c\_s\_t\_i(crops,plot,till,intens),syntFertilizer,m),  
 v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m)  
 \* p\_nutInSynt(syntFertilizer,nut) ) ;

Input via animal manure is calculated.

nutExcrDueV\_(nut,tCur(t),nCur) $ t\_n(t,nCur)..  
  
 v\_nutExcrDuev(nut,t,nCur) =e=  
  
 sum((actHerds(possHerds,breeds,feedRegime,t,m)),  
 v\_herdSize(possHerds,breeds,feedRegime,t,nCur,m)  
 \* ( 1 - 1 $ sameas(feedRegime,"fullGraz")  
 - 0.5 $ sameas(feedRegime,"partGraz"))  
 \* 1/card(herdM)  
  
 \* p\_nutExcreDueV(possHerds,feedRegime,nut) );

Input via digestates from biogas production is calculated (only digestate from plant origin as for instance silage maize).

nutBiogasDuev\_(nut,tCur(t),nCur) $ t\_n(t,nCur)..  
  
 v\_nutBiogasDuev(nut,t,nCur) =e=  
  
  
 sum( (curmanchain, m,nut2) $ (not sameas (nut2,"P")),  
 v\_nutCropBiogasM(curmanchain,nut2,t,nCur,m) + sum(curmaM, v\_nut2ManurePurch(curmanchain,nut2,curmaM,t,nCur,m) )) $ (sameas (nut,"N") $ sum(sameas(manchain,"LiquidBiogas"),1))  
  
 + sum( (curmanchain,m) , v\_nutCropBiogasM(curmanchain,"P",t,nCur,m) + sum(curmaM, v\_nut2ManurePurch(curmanchain,"P",curmaM,t,nCur,m))) $ (sameas (nut,"P") $ sum(sameas(manchain,"LiquidBiogas"),1))  
 ;

In the equation *nutBalDuev\_*, nutrient inputs and outputs are combined. Manure N is accounted with factors defined by the Fertilisation Ordinance. As a supplement to nutrient inputs and outputs, the import and export of manure nutrients are included into the equation.

nutBalDueV\_(nut,tCur(t),nCur) $ t\_n(t,nCur) ..  
  
$iftheni.h %herd% == true  
  
\* --- Nutrients excreted from animals time specific loss factor  
  
 v\_nutExcrDuev(nut,t,nCur) \* p\_nutEffectivDueVNv(nut)  
$endif.h  
  
\* --- Nutrients coming from biogas plant (including energy crops and purchased manure)  
  
$iftheni.b %biogas% == true  
 + v\_nutBiogasDuev(nut,t,nCur) \* p\_nutEffectivDueVNvBiogas(nut)  
  
$endif.b  
  
\* --- Applied synthetic fertilizer  
  
 + v\_synthAppliedDueV(nut,t,nCur)  
  
  
\* --- Nutrient from N fixation from legumes in grassland  
 + sum( (c\_s\_t\_i(crops,plot,till,intens)) ,  
 v\_cropHa(crops,plot,till,intens,t,nCur) \* p\_NfromLegumes(Crops) ) $ (sameas (nut,"N") )  
  
  
\* --- Import of manure  
  
 $$iftheni.im "%AllowManureImport%" == "true"  
  
 + sum ( (nut2\_nut(nut2,nut),m), v\_manImport(t,nCur,m) \* p\_nut2inMan(nut2,"manImport","LiquidImport") ) \* (1- (p\_nutEffectivDueVAl - p\_nutEffectivDueVNv("N") )) $ sameas (nut,"N")  
 + sum ( (nut2\_nut(nut2,nut),m), v\_manImport(t,nCur,m) \* p\_nut2inMan(nut2,"manImport","LiquidImport") ) $ sameas (nut,"P")  
  
 $$endif.im  
  
\* --- Crop output (nutrient removal)  
  
 - v\_NutRemovalDuev(nut,t,nCur)  
  
$iftheni.h %herd% == true  
  
\* --- Nutrients exported from farm  
  
 $$iftheni.ExMan %AllowManureExport%==true  
  
 - sum( (curManChain,m,nut2) $(not sameas (nut2,"P")), v\_nut2export(curManChain,nut2,t,nCur,m) ) $ sameas (nut,"N")  
 - sum( (curManChain,m), v\_nut2export(curManChain,"P",t,nCur,m) ) $ sameas (nut,"P")  
  
 $$endif.ExMan  
  
 $$iftheni.emissionRight not "%emissionRight%"==0  
  
 - sum( (curManChain,m,nut2) $(not sameas (nut2,"P")), v\_nut2exportMER(curManChain,nut2,t,nCur,m) ) $ sameas (nut,"N")  
 - sum( (curManChain,m), v\_nut2exportMER(curManChain,"P",t,nCur,m) ) $ sameas (nut,"P")  
  
 $$endif.emissionRight  
  
$endif.h  
  
 =e=  
  
 v\_surplusDueV(t,nCur,nut) ;

The surplus, *v\_surplusDueV,* is not allowed to exceed a certain threshold, which changes from Fertilisation Ordinance 2007 to 2017 and, in addition, can be defined in the GUI.

nutSurplusDueVRestr\_ (tCur(t),nCur,nut) $ (p\_surPlusDueVMax(t,nut) $ t\_n(t,nCur)) ..  
  
 v\_surplusDueV(t,nCur,nut)  
  
 =L=  
 p\_surplusDueVMax(t,nut) \* v\_croplandActive(t,nCur) \* ( 1 - p\_soilShareNutEnriched) $ sameas (nut,"P")  
  
 + p\_surplusDueVMax(t,nut) \* v\_croplandActive(t,nCur) $ sameas (nut,"N")  
  
 ;

## Organic nitrogen application threshold

Farms have to calculate the application of manure N and, under the Fertilisation Ordinance 17, N from biogas digestate. The derived value is not allowed to exceed a threshold related to farm area in ha.

As input for manure N, animal excretion *v\_nutExcrDuev(nut,t,nCur)\_* is included, and the input from biogas is calculated seperatly in the following equation.

nutBiogasDueVAccAL\_(tCur(t),nCur) $ t\_n(t,nCur)..  
  
 v\_nutBiogasDueVAccAL(t,nCur) =e=  
  
 sum( (curmanchain,m,nut2) $ (not sameas (nut2,"P")), v\_nutCropBiogasM(curmanchain,nut2,t,nCur,m) \* p\_nutEffectivDueVAlBiogasPlantDig  
  
\* --- Depending of the Fertilizer Ordinance, the inclusion of digestate N from plant origin can be switched on/off (GUI=optional, FO07 = off, FO17 = on)  
  
 \* p\_NincludeBioDigest )  
  
 + sum ( (curBhkw(bhkw), curEeg(eeg),curmaM,m,nut2) $ (not sameas(nut2,"P")),  
 v\_purchManure(bhkw,eeg,curmaM,t,nCur,m) \* p\_nut2manPurch("LiquidBiogas",nut2,curmaM) \* p\_nutEffectivDueVAlBiogasPurchMan(curmaM) ) ;

The N input from manure and biogas digestate and manure import is summarized in the following equation. The export of manure is substracted in the equation. The variable *v\_DueVOrgN* returns the accordance of nitrogen from organic sources at a farm level. The nitrogen has to be accounted with factors defined by the Fertilisation Ordinance.

DuevOrgN\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_DueVOrgN (t,nCur) =E=  
  
$iftheni.h %herd% == true  
  
 v\_nutExcrDuev("N",t,nCur) \* p\_nutEffectivDueVAl  
$endif.h  
$iftheni.dh %daidyherd% == true  
  
 v\_nutExcrPast("N",t,nCur) \* p\_nutEffectivDueVAlPast  
$endif.dh  
  
  
\* --- Nutrients imported to the farm  
\* [TK][TO DO] add coefficient for accounting for imported manure  
  
$iftheni.im "%AllowManureImport%" == "true"  
  
 + sum ( (nut2,m) $ (not sameas (nut2,"P")), v\_manImport(t,nCur,m) \* p\_nut2inMan(nut2,"manImport","LiquidImport") )  
  
$endif.im  
  
\* --- Nutrients exported from farm  
  
$iftheni.ExMan %AllowManureExport%==true  
  
 - sum( (curManChain,m,nut2) $(not sameas (nut2,"P")), v\_nut2export(curManChain,nut2,t,nCur,m) )  
  
$endif.ExMan  
  
  
$iftheni.emissionRight not "%emissionRight%"==0  
  
 - sum( (curManChain,m,nut2) $(not sameas (nut2,"P")), v\_nut2exportMER(curManChain,nut2,t,nCur,m) )  
$endif.emissionRight  
  
\* --- Nutrients coming from biogas plant, included depending on FD, calculated in fermenter tech  
  
$iftheni.b %biogas% == true  
  
 + v\_nutBiogasDueVAccAL(t,nCur)  
$endif.b  
  
;

The N input is not allowed to exceed a target value defined by the Fertilisation Ordinance, being 170 kg N/ha/a in most cases.

DuevOrgNLimit\_ (tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_DueVOrgN (t,nCur)  
  
 =L=  
  
 sum( (c\_s\_t\_i(curCrops(crops),plot,till,intens)) $ ( not (sameas(crops,"idle") or sameas (crops,"idlegras") or sameas (crops,"catchCrop") ) ),  
 p\_nutManApplLimit(crops,t)  
 \* v\_cropHa(crops,plot,till,intens,t,nCur)) ;

## Binding fertiliser planning

Under the Fertilisation Ordinance 2017, farms have to do an obligatory fertiliser planning based on the expected yields. The derived nutrient need with regard to nitrogen must not be exceeded. This allows to calculate a nitrogen quota which farms have to meet. The fertiliser quota is always calculated, if the Fertilisation Ordinance is switched on. However, it only becomes binding for fertiliser application under the Fertilisation Ordinance 2017.

The nutrient input is summarised in the equation *FertQuotaInput\_*. Nutrients from chemical fertiliser, manure and mineralisation from the soil are taken into account. Manure N is accounted with mineral fertiliser equivalents defined by the Ordinance.

FertQuotaInput\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_FertQuotaInput(t,nCur)  
  
 =e=  
  
\* --- Input of chemical N fertilizer which is fully accounted in the fertilizer quota  
  
 sum ( (c\_s\_t\_i(curCrops(crops),plot,till,intens),syntFertilizer,m) $( (not sameas (crops,"catchCrop")) $ ( not sameas (crops,"idle") ) $ (not sameas (crops,"idleGras") ) ),  
 v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m) \* p\_nutInSynt(syntFertilizer,"N") )  
  
\* --- Input of manure N which is accounted with prescribed mineral fertilizer equivalents  
  
$iftheni.man %manure% == true  
  
 + sum( ( c\_s\_t\_i(curCrops(crops),plot,till,intens),manApplicType\_manType(ManApplicType,curManType),nut2,manChain\_type(manChain,curManType),m)  
 $ ( ( not sameas (nut2,"P")) $ (not sameas (crops,"catchCrop")) $ ( not sameas (crops,"idle") ) $ (not sameas (crops,"idleGras") )  
 $ (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m) ne 0) ),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m)  
 \* p\_nut2inMan(nut2,curManType,manChain) \* p\_nutEffFOPlan(curManType) )  
  
$endif.man  
  
  
 ;

Furthermore, the plant nutrient need is calculated. It depends on the yield level and is precisely defined by the Fertilisation Ordinance.

FertQuotaNeed\_(tCur(t),nCur) ..  
  
  
 v\_FertQuotaNeed(t,nCur)  
  
 =e=  
  
\* --- N need is derived from FO 17, depending on yield level which is reflected in p\_NneedFerPlan  
  
 sum ( ( c\_s\_t\_i(curCrops(crops),plot,till,intens) ) $( (not sameas (crops,"catchCrop")) $ ( not sameas (crops,"idle") ) $ (not sameas (crops,"idleGras") ) )  
 , v\_cropHa(crops,plot,till,intens,t,nCur)  
 \* sum(plot\_soil(plot,soil), p\_NneedFerPlan(crops,plot,soil,till,intens,t) ) )  
  
\* --- Assumption that N min provided in spring is always 50;

The plant nutrient input is not allowed to exceed the estimated nutrient need. The restriction becomes only binding under the Fertilisation Ordinance 2017. *p\_bigNumberFO*, being a very large number if the Fertilisation Ordinance 2017 is not activated, ensures that the nutrient need is extremely high under the Fertilisation Ordinance 2007 and not binding.

FertQuota\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_FertQuotaInput(t,nCur) =l= v\_FertQuotaNeed(t,nCur) \* p\_bigNumberFO ;

## Required manure storage capacities

Farms are required to hold a minimum storage capacity to bridge the time in autumn and winter when manure application is not allowed. This storage capacity was defined at federal state level under the Fertilisation Ordinance 2007. Under the Fertilisation Ordinance 2017, it is defined at federal level.

Generally, farms have to hold a manure storage capacity to gap the amount of manure excretion corresponding to a certain time period, e.g. 6 months. Therefore, the required storage capacity is defined in the manure module. The parameter *p\_ManureStorageNeed* defines the required amount of months and is linked to the selected Fertilisation Ordinance or can be defined in the GUI.

manStorCapNeed\_(curManChain(manChain),tCur(t),nCur) $ (t\_n(t,nCur) $ p\_ManureStorageNeed) ..  
  
 v\_ManStorCapNeed(manChain,t,nCur) =e= p\_ManureStorageNeed \* (  
  
 $$ifi %herd% == true v\_manQuant(manChain,t,nCur) $ (not sameas (manchain, "LiquidBiogas"))  
  
\* --- required silo storage capacity for biogas plant digestate (including energy crops and purchased manure)  
  
 $$ifi %biogas% == true + sum((crM(biogasfeedM),m), v\_voldigCrop(crM,t,nCur,m) + v\_volDigMan(t,nCur,m)) $ sameas ("LiquidBiogas",manchain)  
  
 );

Under the Fertilisation Ordinance 2017, farms exceeding a stocking density of 3 livestock units per ha have to hold additional manure storage capacity. This is implemented with a binary trigger in FarmDyn. The variable *v\_triggerStorageGVha* becomes one when the farm exceeds the livestock unit threshold.

triggerStorageGVha\_(tCur(t),nCur) $t\_n(t,nCur) ..  
  
 ( v\_sumGV(t,nCur) / sum(plot, p\_plotSize(plot)) ) - 3 =l= v\_triggerStorageGVha(t,nCur) \* 16 ;

If the variable *v\_triggerStorageGVha* is one, the restriction in the following equation becomes binding.

manStorCapGVDepend\_(curManChain(manChain),tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_TotalManStorCap(manChain,t,nCur)  
 =g= v\_manQuant(manChain,t,nCur) \* p\_ManureStorageNeedGV  
 - ( (1 - v\_triggerStorageGVha(t,nCur) ) \* p\_bigNumber ) ;

## Banning periods for fertiliser application

During certain months of the year, the application of fertiliser is not allowed as there is no plant nutrient need and the risk of nitrate leaching is very high. This is implemented in FarmDyn by setting the variable *v\_mandist* and *v\_syntdist* to zero for certain months which disables fertiliser application in the model.

Depending on the Fertilisation Ordinance selected, sets are defined which include the months in which fertiliser application is forbidden (can also be defined via the GUI which is not shown here).

$$elseifi.fertGui %RegulationFert% == FD\_2007  
  
\* --- (2) Depending on regulation of FO 07  
  
 set monthApplicationForbidden(m) /Dec,Jan / ;  
 set monthApplicationForbiddenArab(m) /Nov,Dec,Jan / ;  
 set monthApplicationForbiddenGrass(m) / Dec,Jan / ;  
  
  
 $$elseifi.fertGui %RegulationFert% == FD\_2017  
  
\* --- (3) Depending on regulation of FO 17  
 set monthApplicationForbidden(m) /Nov,Dec,Jan / ;  
 set monthApplicationForbiddenArab(m) /Oct,Nov,Dec,Jan / ;  
 set monthApplicationForbiddenGrass(m) / Nov,Dec,Jan / ;  
  
 $$endif.fertGui

For the months which are defined in the described sets, the variables for fertiliser application are set to zero.

v\_syntDist.up(arabCrops(crops),plot,till,intens,syntFertilizer,t,n,monthApplicationForbiddenArab(m))  
 $ ( t\_n(t,n) $ c\_s\_t\_i(crops,plot,till,intens) ) = 0 ;  
  
 v\_syntDist.up(grassCrops(crops),plot,till,intens,syntFertilizer,t,n,monthApplicationForbiddenGrass(m))  
 $ ( t\_n(t,n) $ c\_s\_t\_i(grassCrops,plot,till,intens) ) = 0 ;  
  
  
  
 $$iftheni.v\_manDist declared v\_manDist  
  
 v\_volManApplied.up(manChain,t,n,monthApplicationForbidden) $ t\_n(t,n) = 0;  
 v\_nut2ManApplied.up(manChain,nut2,t,n,monthApplicationForbidden) $ t\_n(t,n) = 0;  
 v\_manDist.up(crops,plot,till,intens,manApplicType,manType,t,n,monthApplicationForbidden)  
 $ (t\_n(t,n) $ c\_s\_t\_i(crops,plot,till,intens)) = 0;  
  
 v\_manDist.up(arabCrops(crops),plot,till,intens,manApplicType,manType,t,n,monthApplicationForbiddenArab)  
 $ (t\_n(t,n) $ c\_s\_t\_i(crops,plot,till,intens)) = 0;  
  
 v\_manDist.up(grassCrops(crops),plot,till,intens,manApplicType,manType,t,n,monthApplicationForbiddenGrass)  
 $ (t\_n(t,n) $ c\_s\_t\_i(crops,plot,till,intens)) = 0;  
  
 $$endif.v\_manDist

## Restriction of fertiliser application in autumn

In addition to the fixed banning periods, the application of fertiliser in autumn is only legal for some crops and restricted to a defined amount of nitrogen per ha. The parameter *p\_NLimitInAutumn* contains the allowed amount of N and is defined depending on the Fertilisation Ordinance. Catch crops allow additional manure application in autumn.

NLimitAutumn\_ (c\_s\_t\_i(curCrops(arablecrops),plot,till,intens),tCur(t),nCur)  
 $ ( (v\_cropHa.up(arableCrops,plot,till,intens,t,nCur) ne 0) $ t\_n(t,nCur) ) ..  
  
  
$iftheni.man %manure% == true  
  
 sum( (manChain\_type(manChain,curManType),manApplicType\_manType(ManApplicType,curManType),nut2,m) $ monthHarvestBlock(arableCrops,m),  
 v\_manDist(arablecrops,plot,till,intens,ManApplicType,curManType,t,nCur,m)  
 \* p\_nut2inMan(nut2,curManType,manChain) $ (not sameas (nut2,"P")) )  
  
$endif.man  
  
 + sum( (syntFertilizer,m) $ monthHarvestBlock(arableCrops,m), v\_syntDist(arableCrops,plot,till,intens,syntFertilizer,t,nCur,m)  
  
 \* p\_nutInSynt(syntFertilizer,"N") )  
  
 =l= v\_cropHa(arableCrops,plot,till,intens,t,nCur) \* p\_NLimitInAutumn(arablecrops)  
  
\* --- Catch crop allow also the application of manure in autumn. Interpreted as possibility to get rid of manure in autumn as there is no nutrient need for catch crops. The nutrient of applied manure  
\* to catch crops is provided to the following main crop. Therefore, catch crops are not included into the nutrient need calculation but allows to "move the nutrient need of the main crops"  
\* to autumn. Note that this inclusion of catch crops makes them always listed under v\_cropha.  
\*  
  
  
 + v\_cropHa("catchCrop",plot,till,intens,t,nCur) \* p\_NLimitInAutumn("catchCrop") $ curCrops("catchCrop")  
 ;

## Low-emission manure application techniques

The Fertilisation Ordinance defines which manure application techniques are legally allowed. Under the Fertilisation Ordinance 2017, broadcast spreading is banned except on fallow land followed by direct incorporation. This measures is introduced in FarmDyn by setting the variable *v\_mandist* to zero for certain months and not allowed application techniques.

\* --- Broadcast spreader are banned on grassland  
  
 $$iftheni.dh %cattle%==true  
 v\_manDist.up(grassCrops(crops),plot,till,intens,"applSpreadPig",manType,t,nCur,m) $ t\_n(t,Ncur) = 0 ;  
 v\_manDist.up(grassCrops(crops),plot,till,intens,"applSpreadCattle",manType,t,nCur,m) $ t\_n(t,Ncur) = 0 ;  
 $$ifi %AllowManureImport% == true v\_manDist.up(grassCrops(crops),plot,till,intens,"applSpreadImport",manType,t,nCur,m) $ t\_n(t,Ncur) = 0 ;  
 $$endif.dh

\* --- Broadcast spreader are generally banned on arable land, except when there is no crop  
  
 v\_manDist.up(arabCrops(crops),plot,till,intens,"applSpreadPig",manType,t,nCur,m) $ (t\_n(t,Ncur) $ monthGrowthCrops(crops,m)) = 0 ;  
 v\_manDist.up(arabCrops(crops),plot,till,intens,"applSpreadCattle",manType,t,nCur,m) $ (t\_n(t,Ncur) $ monthGrowthCrops(crops,m)) = 0 ;

# Environmental Accounting Module

!!! abstract The environmental accounting module utilises commonly applied methodology for the quantification of methane (CH4), ammonia (NH3), nitrous dioxide (N2O), nitrogen oxides (NOx) and elemental nitrogen (N2), as laid down in IPCC (2006), Haenel (2018) and EMEP (2013, 2016). An extension of the scope of accounting to LCA methodology enables the consideration of emissions prior to on-farm activities such as the provision of major inputs (EcoInvent 2.X). Emissions are characterised at midpoint level using characterisation factors from ReCiPe (2016). A soil surface balance is calculated for nitrogen (N) and (P) indicating N and P prone to loss through run-off or leaching.

## Gaseous emissions

All calculations related to the environmental accounting are listed in *model\env\_acc\_module.gms* while the respective emission factors, characterisation factors and other input data are specified in *coeffgen\env\_acc.gms*. The calculation of emissions follows Haenel et al. (2018). An overview of the methodology, data and the respective (primary) sources used are presented in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Source/Emission | Methodology applied | Emission factor | Revised EF |
| CH4 enteric fermentation | IPCC(2006)-10.30 f. tier 2+3 | Haenel et al. (2018) p.140, p.145, p.155, p.168, p.214, p.194, IPCC p.10.30 | Haenel et al. (2018), DAMMGEN et al. (2013), IPCC (2006)- 10.30, DAMMGEN et al. (2012C) |
| CH4 stable, storage and pasture | Haenel et al. (2018) p. 42 No. 3.28 and 3.29 Following IPCC (2006) eq. 10.23 | Haenel et al.(2018) p.108 and p. 185. IPCC (2006) p.10.41 | DAMMGEN et al., (2012a), IPCC (2006)- |
| NH3 emissions from stable and storage | EMEP (2016) | Haenel (2018) p.108, p. 109, Haenel et al. (2018) p.186 p.187 | DAMMGEN et al. (2010a), DAMMGEN et al. (2010b) |
| N2O, NOx, N2 emissions from stable and storage | EMEP (2016), Haenel (2018) p. 53 | Haenel 2018 p. 110, HAENEL et al. (2012), JARVIS & PAIN (1994), Haenel et al. (2015) pp. 188 | IPCC (2006), DAMMGEN et al. (2010b) |
| NH3 from manure application | EMEP (2016) | Haenel et al. (2018), pp. 111-112, 189, 64 | DOHLER et al. (2002) |
| N2O, NOx, N2 emissions from manure application | EMEP (2016), Haenel et al. (2018), pp. 316-317 | Haenel et al. (2018) p.326, Stehfest and Bouwman (2006) N2 Roesemann et al. (2015) pp. 316-317 |  |
| NH3 from excreta from pasture | EMEP (2016), Haenel et al. (2018) p.55 | Haenel (2018) p.137/EMEP(2013): 3B , pp. 27 |  |
| N2O, NOx, N2 emissions from excreta on pastures | EMEP (2016), Haenel et al. (2018) p.55 | Haenel et al. (2018) p. 332; IPCC (2006) 11.11, table 11.1, Haenel et al. (2018) p. 332, STEHFEST UND BOUWMAN (2006) Roesemann et al. (2015), pp. 324 |  |
| NH3, N2O, NOx, N2 emissions from mineral fertiliser application | Haenel et al. (2018), pp. 316-317 | Haenel et al. (2018) p.325, Haenel et al. (2018) p.326, Stehfest and Bouwman(2006) N2 Roesemann et al. (2015) |  |
| Indirect N2O emissions from prior NOx, NH3 and NO3 emissions | IPCC (2006) | IPCC (2006)-11.24, Table 11.3 | IPCC (2006) |
| CO2 emission from provision of inputs |  | Ecoinvent |  |
| NO3-N leach | Agroscope |  |  |
| P-loss | Agroscope |  |  |

The considered emissions are listed in the set *emissions*, the included sources in the set *sources*. The cross set *source\_emissions* links emissions to relevant sources. The set *emCat* lists midpoint emission categories according to ReCiPe (2016).

set emissions / NO3,NH3,N2O,NOx,N2,N2Oind,NSoilSurplus,PsoilSurplus,CH4,CO2 /;

set source / entFerm,staSto,past,manAppl,minAppl,field,input / ;

set source\_emissions(source,emissions) /  
 staSto.(NH3,N2O,NOx,N2,N2Oind,CH4)  
 past.(NH3,N2O,NOx,N2,N2Oind,CH4)  
 manAppl.(NH3,N2O,NOx,N2,N2Oind)  
 minAppl.(NH3,N2O,NOx,N2,N2Oind)  
 field.(NSoilSurplus,PsoilSurplus,NO3,N2Oind)  
 entFerm.CH4  
 input.CO2  
 / ;

set emCat /GWP,PMFP,TAP,FEP,MEP/;

The actual calculation of the emissions is realised in the equation *emissions\_*. The timely resolution allows for reporting of emissions on a monthly basis. The different compartments of the equation represent the order of emission accounting by emissions and sources based on Haenel et al. (2018). Using conditional *sameas* statements only relevant emissions and sources are activated. The different compartments of the equation *emissions\_* are presented in the following in their order of appearance:

1. Methane emissions from enteric fermentation

* Emissions from enteric fermentation are calculated based on the actual feedintake, v\_feeduse, measured in gross energy. CH4 conversion factors, p\_Ym, represent animal specific emission rates for cattle and pig herds.

emissions\_(chain\_source(curChain,source),emissions,t\_n(t,nCur),m) $ (tCur(t) $ source\_emissions(source,emissions) ) ..  
  
 v\_emissions(curChain,source,emissions,t,nCur,m)  
  
 =E=  
  
\* --- Calculation of CH4 emissions from enteric fermentation linked to gross energy intake (IPCC, 2006, eq. 10.21)  
\* in kg CH4 per month (yearly emissions averaged for monthly reporting),  
  
$iftheni.h %herd% == true  
 + [ (  
 $$iftheni.ch %cattle% == true  
 + sum((feeds,dcows,n),  
 p\_feedContFMton(feeds,"GE") \* v\_feedUseHerds(dcows,feeds,t,n) \* p\_Ym("dcows"))  
  
 + sum((feeds,mcows,n),  
 p\_feedContFMton(feeds,"GE") \* v\_feedUseHerds(mcows,feeds,t,n) \* p\_Ym("mcows"))  
  
 + sum((feeds,heifs,n),  
 p\_feedContFMton(feeds,"GE") \* v\_feedUseHerds(heifs,feeds,t,n) \* p\_Ym("heifs"))  
  
 + sum((feeds,bulls,n),  
 p\_feedContFMton(feeds,"GE") \* v\_feedUseHerds(bulls,feeds,t,n) \* p\_Ym("bulls"))  
  
 + sum((feeds,calvs,n),  
 p\_feedContFMton(feeds,"GE") \* v\_feedUseHerds(calvs,feeds,t,n) \* p\_Ym("calvs"))  
 $$endif.ch  
 $$iftheni.fat "%farmBranchfattners%" == "on"  
 + sum((actHerds(fatHerd,breeds,feedRegime,t,m)),  
 p\_feedReqPig(fatHerd,feedRegime,"energ")\*1000 \* v\_herdsize(fatHerd,breeds,feedRegime,t,nCur,m) \* p\_YM("fatHerd"))  
 $$endif.fat  
 $$iftheni.sows "%farmBranchSows%" == "on"  
 + sum((actHerds(sows,breeds,feedRegime,t,m)),  
 p\_feedReqPig(sows,feedRegime,"energ")\*1000 \* v\_herdsize(sows,breeds,feedRegime,t,nCur,m) \* p\_YM("sows"))  
 $$endif.sows  
 )/(100 \* 55.65) \* 1/card(herdM)  
  
 ] $ ( sameas(emissions,"CH4") $ sameas(source,"entFerm") )

1. Methane emissions from stable and storage

CH4 emissions stemming from manure storage are calculated according to the volume in the different storage systems, *v\_volInStorageType*. The amount of volatile solids in the slurry is estimated based on the stored volume using the average dry matter, *p\_avDmMan*, and the share of volatile solids in the dry matter, *p\_oTSMan*. The effect of different slurry cover types on emissions is incorporated via different methane conversion factors, *p\_MCF*. Furthermore, different manure types are considered in the maximum methane producing capacity, *p\_BO*.

\* CH4 from manure storage:  
  
 + [ sum( (curManChain,manStorage), v\_volInStorageType(curManChain,manStorage,t,nCur,m)  
 \* 1000 \* p\_avDmMan(curManChain) \* p\_oTSMan(curManChain) \* p\_BO(curManChain)  
 \* p\_densM \* p\_MCF(Manstorage,curManChain)  
 /12)  
 ] $ ( sameas(emissions,"CH4") $ sameas(source,"staSto") )

1. Methane emissions from excreta on pastures

Excreta on pastures also emits CH4. The calculation of those emissions is conducted analog to the emissions from storage with a specific methane conversion factor, *p\_MCFPast*:

\* Pasture:  
  
 $$iftheni.ch %cattle% == true  
  
 + [ sum(curManChain, v\_manQuantPast(curManChain,t,nCur,m)  
 \* 1000 \* p\_avDmMan(curManchain) \* p\_oTSMan(curManChain) \* p\_BO(curManchain)  
 \* p\_densM \* p\_MCFPast  
 )  
 ] $ ( sameas(emissions,"CH4") $ sameas(source,"past") )

N-emissions are calculated using a mass-flow approach starting with the N excretion by farm animals. Three N-pools are considered, N-TAN, N-Org and total N. The correction of the N pools by previous losses are not part of the *env\_acc module* but are considered in the *manure\_module*. The considered N flows and emissions are depicted in the figure below :

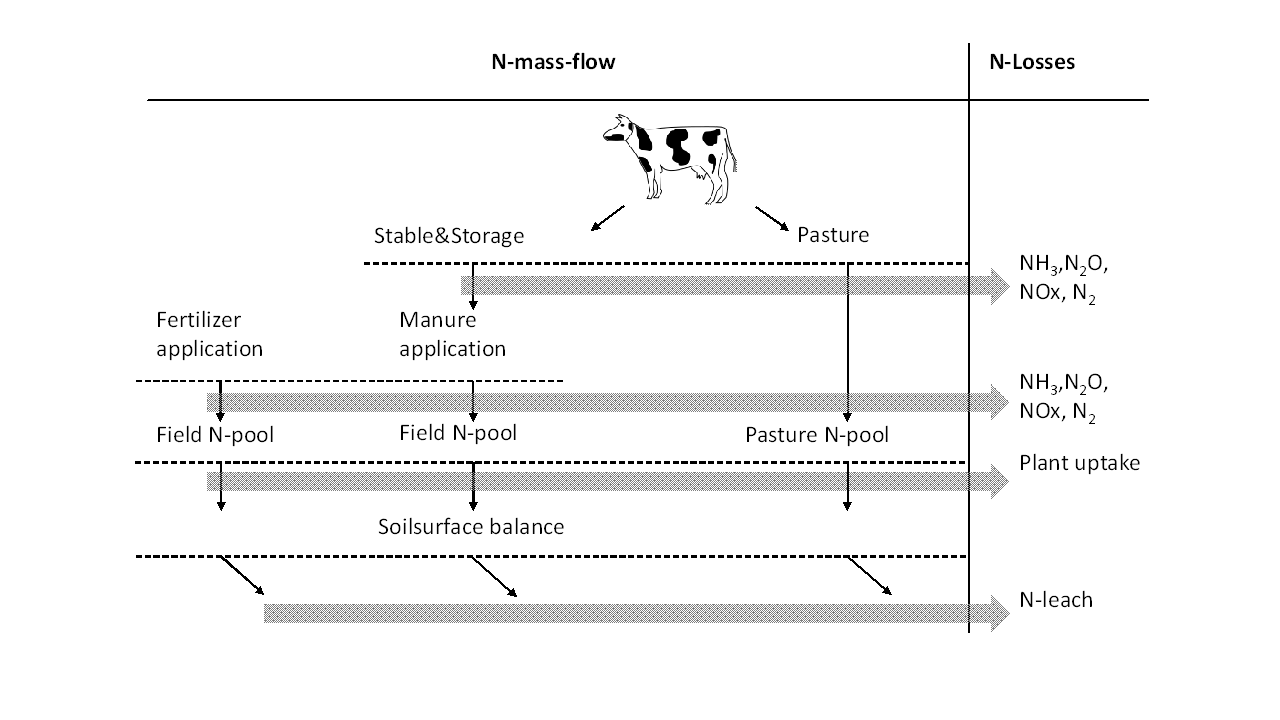


Figure 1: N massflow approach with considered stages and emissions in FarmDyn

1. N emissions from stable and storage

NH3 emissions at the stable stage are calculated according to the NTAN in manure as excreted by the animals, *v\_nut2ManureM*. NH3 emissions from storage are calculated based on the N-TAN pool in storage, *v\_nutPoolInStorage*. The emission factors differentiate between cattle and pig slurry. While NH3 emissions are based only on the N-TAN pool, other N emissions are based on the total N pool as depicted in *v\_nut2manureM*. Considered emissions are N2O, and NOx. N2 is generally not considered as an emission. For the completeness of the N-flow model N losses in the form of N2 are still calculated in the environmental accounting. Indirect N2O emissions (N2Oind) are calculated based on prior emissions of reactive N species, namely NH3 and NOx. For the sake of simplicity, the stages stable and storage are summarized in the calculation of emissions. Compared to total N2O and NOx emissions on farm the emissions at this stage are rather small and the generalisation is not expected to distort the results.

\* --- Calculation of NH3, N2O, NOx, N2, N2Oind from stable and storage (staSto)  
  
 + [  
 $$iftheni.loss "%nutLossStorageTime%" == true  
 sum(sameas(curManChain,curChain), (v\_nut2ManureM(curManChain,"NTAN",t,nCur,m)  
 \* p\_EFSta("NH3")  
 + v\_nutPoolInStorage(curManChain,"NTAN",t,nCur,m)  
 \* p\_EFSto("NH3"))) $ sameas(emissions,"NH3")  
 $$else.loss  
 sum(sameas(curManChain,curChain), (v\_nut2ManureM(curManChain,"NTAN",t,nCur,m) $ (not sameas(curmanchain,"LiquidBiogas"))  
 \* (p\_EFSta("NH3") + p\_EFSto("NH3")))) $( sameas(emissions,"NH3"))  
  
  
 $$endif.loss  
  
 + sum(sameas(curManChain,curChain), (v\_nut2ManureM(curManChain,"NTAN",t,nCur,m)$ (not sameas(curmanchain,"LiquidBiogas"))  
 + v\_nut2ManureM(curManChain,"NOrg",t,nCur,m)$ (not sameas(curmanchain,"LiquidBiogas")))  
 \* ( p\_EFStaSto("N2O",curManChain) $ sameas(emissions,"N2O")  
 + p\_EFStaSto("NOx",curManChain) $ sameas(emissions,"NOx")  
 + p\_EFStaSto("N2",curManChain) $ sameas(emissions,"N2")  
 ))  
  
 + (sum(sameas(curManChain,curChain) , v\_emissions(curChain,"stasto","NH3",t,nCur,m)  
 + v\_emissions(curChain,"stasto","NOx",t,nCur,m))  
 \* p\_EFN2Oind ) $ sameas(emissions,"N2Oind")  
  
 ] $ sameas(source,"staSto")

1. Calculation of NH3, N2O, NO and N2 losses from manure excretion on pasture for cattle

The calculation of N emissions from pastures follows the same logic as the calculation of emissions from the stable and storage stage. The emission factors represent the conditions of manure excreted on pastures.

\* --- Calculation of NH3, N2O, NO and N2 losses from manure excretion on pasture for cattle according to Haenel et al. (2018) p.55  
\* in kg NH3-N, N2O-N, NO-N and N2 per month  
  
$$iftheni.ch %cattle% == true  
 + [  
 + sum(c\_s\_t\_i(past,plot,till,intens),  
 v\_nut2ManurePast(past,plot,till,intens,"NTAN",t,nCur,m)  
 \* ( p\_EFpasture("NH3") $ sameas(emissions,"NH3") ))  
  
 + sum(c\_s\_t\_i(past,plot,till,intens),  
 (v\_nut2ManurePast(past,plot,till,intens,"NTAN",t,nCur,m)  
 + v\_nut2ManurePast(past,plot,till,intens,"Norg",t,nCur,m))  
 \* ( p\_EFpasture("N2O") $ sameas(emissions,"N2O")  
 + p\_EFpasture("NOx") $ sameas(emissions,"NOx")  
 + p\_EFpasture("N2") $ sameas(emissions,"N2"

1. Calculation of NH3, N2O, NOx, N2 from manure application

NH3 emissions from the application of manure are calculated based on the N-TAN pool in the slurry leaving the storage stage. The emission factors vary between grassland and arable land, different application devices and pig and cattle slurry. N2O, NOx and N2 emissions are calculated based on the total N pool at the application stage, *v\_nut2manApplied*. The emission factors are equal to the emission factors for the application of synthetic fertilisers, as proposed by EMEP(2016). Indirect N2O emissions are based on prior emissions of NH3 and NOx.

\* --- Calculation of NH3, N2O, NOx, N2 from manure application  
\* NH3 losses depending on technology, source EMEP (2016)  
\* in kg NH3-N, N2O-N, NO-N and N2 per month  
  
 + [ sum( (c\_s\_t\_i(curCrops(crops),plot,till,intens),manApplicType\_manType(ManApplicType,curManType))  
 $ ( sum(sameas(curChain,curManChain) $ manChain\_type(curManChain,curManType),1) $( not sameas (crops,"catchCrop"))),  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m)  
 \* sum(manChain,p\_nut2inMan("NTAN",curManType,manChain))  
 \* (1- p\_nut2UsableShare(crops,curManType,manApplicType,"NTAN",m))) $ sameas(emissions,"NH3")  
  
 + sum((sameas(curManChain,curChain),nut2) $ (not sameas(nut2,"P")),  
 v\_nut2ManApplied(curManChain,nut2,t,nCur,m) \* ( p\_EFApplMin("N2O") $ sameas(emissions,"N2O")  
 + p\_EFApplMin("NOx") $ sameas(emissions,"NOx")  
 + p\_EFApplMin("N2") $ sameas(emissions,"N2")))  
  
  
 + (sum(sameas(curManChain,curChain) , v\_emissions(curChain,"manAppl","NH3",t,nCur,m)  
 + v\_emissions(curChain,"manAppl","NOx",t,nCur,m))  
 \* p\_EFN2Oind ) $ sameas(emissions,"N2Oind")  
  
 ] $ sameas(source,"manAppl")

1. Calculation of NH3, N2O, NOx, N2 from mineral fertiliser application

N-emissions from the application of mineral fertiliser application follow the same logic as from the application of manure, except for the considered N-pool. In synthetic fertiliser all N is present as N-TAN. The emission factor for NH3 emissions distinguishes between different fertiliser types.

\* --- Calculation of NH3, N2O, NOx, N2 from mineral fertilizer application  
\* Based on Haenel et al. 2018, pp. 316-317  
\* in kg NH3-N, N2O-N, NO-N and N2 per month  
  
 + [sum( (c\_s\_t\_i(curCrops(crops),plot,till,intens),syntFertilizer),  
 v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m) \* p\_nutInSynt(syntFertilizer,"N")  
 \* ( p\_EFApplMinNH3(syntFertilizer) $ sameas(emissions,"NH3")  
 + p\_EFApplMin("N2O") $ sameas(emissions,"N2O")  
 + p\_EFApplMin("NOx") $ sameas(emissions,"NOx")  
 + p\_EFApplMin("N2") $ sameas(emissions,"N2")))  
  
 + (( v\_emissions(" ","minAppl","NH3",t,nCur,m) + v\_emissions(" ","minAppl","NOx",t,nCur,m))  
 \* p\_EFN2Oind ) $ sameas(emissions,"N2Oind")  
 ] $ sameas(source,"minAppl")

1. Calculation of CO2 emissions from bought inputs

Up-stream emissions stemming from the production of farm inputs are included to gain a more detailed, Life cycle assessment like, perspective. For the moment being, the global warming potential of the provision of mayor farm inputs is measured in CO2 equivalents. The emission factors are not part of the official part of the model, as they are subject of license fees.

\* --- Calculation of yearly CO2 emissions from bought inputs in kg CO2eq per month devided by 12 for monthly resolution  
  
 + [ sum((inputs,sys),  
 v\_buy(inputs,sys,t,nCur) \* p\_EFInput(inputs,emissions)/12)  
 ] $ ( sameas (emissions,"CO2") $ sameas (source,"input"))

To ease the calculation of the emissions along the N mass-flow N-emissions are calculated according to their N-weight. The equation *emissionsMass\_* converts the weight into the actual mass of the molecule as a preliminary step for further calculations, characterisations and weightings.

\* --- Calculation of actual weight of N-emissions in kg NH3, N2O, NO and N2 per month  
\*  
 emissionsMass\_(chain\_source(curChain,source),emissions,t\_n(t,nCur),m) $ (tCur(t) $ source\_emissions(source,emissions) ) ..  
  
 v\_emissionsMass(curChain,source,emissions,t,nCur,m) =e=  
  
 v\_emissions(curchain,source,emissions,t,nCur,m) \* p\_corMass(emissions)  
 ;

The equation *emissionsCat* relates the emissions to midpoint impact categories using characterisation factors from ReCiPe (2016). The emissions in the respective category are then summed over the manure chains, sources and month to gain an emission profile for the whole farm.

\* --- Characterization of emission via ReCiPe 2016 in kg eq per year  
\*  
 emissionsCat\_(chain\_source(curChain,source),emCat,t\_n(t,nCur))$ (tCur(t) $ t\_n(t,nCur) )..  
  
 v\_emissionsCat(curChain,source,emCat,t,ncur) =e=  
 sum(source\_emissions(source,emissions),  
 v\_emissionsYear(curChain,source,emissions,t,nCur) \* p\_emCat(emCat,emissions))  
 ;

## N and P surplus

The losses of N, mainly as Nitrate to groundwater bodies, and P, mainly via erosion and entry to surface waters, are the most relevant environmental threats of farming systems. Since they are highly depending on environmental and geographical conditions, fixed emission factors are less commonly used than for gaseous losses. Therefore, we calculate N and P surplus balances in the equation *SoilBal\_* as an indicator for potential loss of N and P after field application. . A more detailed depiction of those emissions can be achieved with the usage of the crop model Simplace. The linkage of the model to FarmDyn is described in the previous chapter: Using data output of the crop modelling framework SIMPLACE.

The balance is calculated as the difference between the nutrient input via organic and mineral fertiliser and the removal of nutrients via the harvested product.

SoilBal\_(nut,t,nCur) $ (tCur(t) $ t\_n(t,nCur)) ..  
  
 v\_soilbalance(nut,t,nCur) =e=  
  
$iftheni.h %herd% ==true  
  
\* --- Calculation of manure applied minus losses from application  
  
 sum( (c\_s\_t\_i(curCrops(crops),plot,till,intens),manApplicType\_manType(ManApplicType,curManType),m)  
 $ ( (v\_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,m) ne 0)  
 $( not sameas (crops,"catchCrop")) ),  
  
 v\_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,m)  
  
 \* ( sum(manChain,p\_nut2inMan("NORG",curManType,manChain)) $ sameas(nut,"N")  
 + sum(manChain,p\_nut2inMan("NTAN",curManType,manChain)) $ sameas(nut,"N")  
 + sum(manChain,p\_nut2inMan("P",curManType,manChain)) $ sameas(nut,"P")  
 ))  
  
 - sum( (curChain,NiEmissions(Emissions),m) $ chain\_source(curChain,"manAppl"),  
 v\_emissions(curChain,"manAppl",emissions,t,nCur,m) ) $ sameas(nut,"N")  
  
$endif.h  
  
\* --- Calculation of mineral N and P applied minus losses from application  
  
 + sum( (c\_s\_t\_i(curCrops(crops),plot,till,intens),syntFertilizer,m),  
 v\_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,m) \* p\_nutInSynt(syntFertilizer,nut) )  
  
 - sum( (NiEmissions(Emissions),m), v\_emissions("","minAppl",emissions,t,nCur,m) ) $ sameas(nut,"N")  
  
\* --- Minus the removal of N and P by harvested product  
  
 - sum ( (plot\_soil(plot,soil),c\_s\_t\_i(curCrops(crops),plot,till,intens)),  
 p\_nutNeed(crops,soil,till,intens,nut,t) \* v\_cropHa(crops,plot,till,intens,t,nCur) )  
  
 + sum( (plot\_soil(plot,soil),c\_s\_t\_i(curCrops(crops),plot,till,intens)),  
 p\_basNut(crops,soil,till,nut,t) \* v\_cropHa(crops,plot,till,intens,t,nCur) )  
 ;

Note, that the calculated surplus differs from the surplus calculated for the threshold under the fertiliser directive (see chapter 2.11.4). For the environmental accounting, the estimated losses from storage, stable etc. are modelled precisely whereas fixed, prescribed values are used for the fertiliser directive.

# Biogas Module

!!! abstract The biogas module defines the economic and technological relations between components of a biogas plant with a monthly resolution, as well as links to the farm. Thereby, it includes the statutory payment structure and their respective restrictions according to the German Renewable Energy Acts (EEGs) from 2004 up to 2014. The biogas module differentiates between three different sizes of biogas plants and accounts for three different life spans of investments connected to the biogas plant. Data for the technological and economic parameters used in the model are derived from KTBL (2013) and FNR (2013). The equations within the template model related to the biogas module are presented in the following section.

## Biogas Economic Part

The economic part describes on the one hand the revenues stemming from the heat and electricity production of the biogas plant, and on the other hand investment and operation costs. The guaranteed feed-in tariff paid to the electricity producer per kWh, *p\_priceElec*, and underlying the revenues, is constructed as a sliding scale price and is exemplary shown in the next equation.

p\_priceElec(bhkw,eeg,tCur(t))$(eegRated(eeg)) = (p\_priceElecBase("150kW",eeg) \* (150/p\_powRate(bhkw,eeg))  
 + p\_priceElecBase(bhkw,eeg) \* ((p\_powRate(bhkw,eeg) - 150)/p\_powRate(bhkw,eeg)))  
 ;

p\_priceElecE2004("150kW","E2004")= 0.08;

*p\_priceElecBase*, used to calculate the guaranteed feed-in tariff differentiated by size, includes the base rate and additional bonuses [[3]](#footnote-98) according to the legislative texts of the EEGs. For the EEG 2012 it only contains the base rate. In addition, the guaranteed feed-in tariff is subject to a degressive relative factor, *p\_priceElecDeg,* which differs between EEGs and describes price reductions over time. The *p\_priceElecBase* is then used to calculate the electricity based revenue of the biogas operator by multiplying it with the produced electricity, *v\_prodElec*. In order to assure a correct representation of the EEG 2012 payment, the biogas module differentiates the electricity output by input source *v\_prodElecCrop* and *v\_prodElecManure* and multiplies it with its respective bonus tariffs *p\_priceElecInputclass* which are added to the base rate.

bioGasObje\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
 v\_salRevBioGas(t,nCur)  
  
 =e=  
  
\* --- Revenue stemming from electricity production with degression depending on EEG (excluding direct marketing)  
 sum( (curBhkw(bhkw),curEeg(eeg),m) $ (not(eegDM(eeg))),  
 v\_prodElec(bhkw,eeg,t,nCur,m) \* p\_priceElec(bhkw,eeg,t) )  
  
\* --- Revenue stemming from electricity production for EEG E2012 differentiated by input class  
 + sum( (curBhkw(bhkw),curEeg(eeg),m) $ (eegDif(eeg)) ,  
 v\_prodElecCrop(bhkw,eeg,t,nCur,m) \* p\_priceElecInputclass(bhkw,eeg,"inputCl1")  
 + v\_prodElecManure(bhkw,eeg,t,nCur,m) \* p\_priceElecInputclass(bhkw,eeg,"inputCl2") )  
  
\* --- Revenue stemming from heat  
 + sum( curEeg(eeg), v\_sellHeat(eeg,t,nCur) \* p\_priceHeat(t) )  
  
\* --- Revenue specification for EEG with direct marketing and flexible biogas production  
 + sum( (curBhkw(bhkw),curEeg(eeg),m)$(eegDM(eeg)),  
 + (v\_prodElec(bhkw,eeg,t,nCur,m) \* p\_shareEPEX(bhkw) )  
 \* (p\_dmMP(bhkw,eeg,t,m) + p\_dmsellPriceHigh(m) )  
 + (v\_prodElec(bhkw,eeg,t,nCur,m) \* (1 - p\_shareEPEX(bhkw) ) )  
 \* (p\_dmMP(bhkw,eeg,t,m) + p\_dmsellPriceLow(m) )  
 + (v\_prodElec(bhkw,eeg,t,nCur,m) \* p\_flexPrem(bhkw,eeg) ) )  
  
\* --- Revenue stemming from scenario premium  
 + sum( (curBhkw(bhkw), curEeg(eeg),m)$(eegScen(eeg)),  
 v\_prodElec(bhkw,eeg,t,nCur,m) \* p\_scenPremium(eeg)$(eegScen(eeg)))  
;

In addition to the *traditional* guaranteed feed-in tariff, the biogas module comprises the payment structure for the so-called *direct marketing option* which was implemented in the EEG 2012. The calculation of the revenue with a direct marketing option is defined as the product of the produced electricity, *v\_prodElec*, the sum of the market premium, *p\_dmMP*, and the price at the electricity spot exchange EPEX Spot, *p\_dmsellPriceHigh/Low.* The latter depends on the amount of electricity sold during high and low stock market prices. Additionally, it is accounted for a flexibility premium, *p\_flexPrem*.

Furthermore, the revenue stemming from heat is accounted for and is included as the product of sold heat, *v\_sellHeat*, times the price of heat, *p\_priceHeat*, which is set to two cents per kWh. The amount of head sold is set exogenously and depends on the biogas plant type.

The detailed steps of the construction of prices can be seen in *\coeffgen\prices\_eeg.gms.*

## Biogas Inventory

The biogas plant inventory differentiates biogas plants by size (set *bhkw*), which determines the engine capacity, the investment costs and the labour use. Three size classes are currently depicted.

set bhkw "different bhkw sizes" /  
 150KW "150kW engine"  
 250kW "250kW engine"  
 500KW "500kW engine"  
 /;

Moreover, in order to use a biogas plant, different components need to be present which differ by lifetime (investment horizon *ih*). For example, in order to use the original plant, the decision maker has to re-invest every seventh year in a new engine but only every twentieth year in a new fermenter.

iH "investment horizon" /  
 iH7 "reinvestment after seven years",  
 iH10 "reinvestment after ten years",  
 iH20 "reinvestment after twenty years"

The biogas plant and their respective parts can either be bought, *v\_buyBiogasPlant(Parts)*, or an already existing biogas plant can be used, *p\_iniBioGas*. Both define the size of the inventory of the biogas plant, *v\_invBioGas(Parts).* The model currently limits the number of biogas plants present on farm to unity.

invBioGasTot\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 sum( (curBhkw(bhkw),curEeg(eeg)), v\_invBioGas(bhkw,eeg,t,nCur)) =L= 1;

invBioGas\_(curBhkw(bhkw),curEeg(eeg),ih,tFull(t),nCur) $ (ih20(ih) $ t\_n(t,nCur)) ..  
  
 v\_invBioGas(bhkw,eeg,t,nCur)  
  
 =L=  
 sum( (tCur(t1),n1) $ (t\_n(t1,n1) $ isNodeBefore(nCur,n1)  
 $ (p\_year(t1) + p\_ih(ih)+1 ge p\_year(t)+1 )  
 and (p\_year(t1)+1 le p\_year(t)+1 ) ),  
  
 v\_buyBioGasPlant(bhkw,eeg,ih,t1,n1) )  
  
 + sum( tOld $ ( (p\_year(tOld) + p\_ih(ih) ge p\_year(t) ) and (p\_year(tOld) le p\_year(t) ) ),  
  
 p\_iniBioGas(bhkw,eeg,ih,tOld) );

invBioGasTotParts\_(curBhkw(bhkw),ih,tCur(t),nCur) $ (t\_n(t,nCur) $ (not ih20(ih)))..  
  
 v\_invBioGasParts(bhkw,ih,t,nCur) =G= sum(curEeg(eeg), v\_invBioGas(bhkw,eeg,t,nCur));

Furthermore, the inventory *v\_invBioGas* stores the information under which EEG the plant was original erected, either by externally setting the EEG for an existing biogas plant or the initial EEG is endogenously determined by the year of investment. In addition, the module provides the plant operator the option to switch from the EEG under which its plant was original erected to newer EEGs endogenously, such that the electricity and heat price of the newer legislation determines the revenues of the plant. For this purpose, the variable *v\_switchBioGas* transfers the current EEG from *v\_invBioGas* to the variable *v\_useBioGasPlant*. Hence, the *v\_invBioGas* is used to represent the inventory while *v\_useBioGasPlant* is used to determine the actual EEG under which a plant is used, i.e. payment structures and feedstock restrictions.

switchBioGas\_(curBhkw(bhkw),curEeg(eeg1),tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_invBioGas(bhkw,eeg1,t,nCur)  
  
 =G= sum(newEeg\_oldEeg(eeg,eeg1) $ curEeg(eeg), v\_switchBioGas(bhkw,eeg1,eeg,t,nCur));

useBioGas\_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur) $ t\_n(t,nCur) ..  
  
 v\_useBioGasPlant(bhkw,eeg,t,nCur)  
  
 =L= sum(newEeg\_oldEeg(eeg,eeg1) $ curEeg(eeg1), v\_switchBioGas(bhkw,eeg1,eeg,t,nCur));

## Production Technology

The production technology describes not only the production process, but also defines the limitations set by technological components such as the engine capacity, fermenter volume and fermentation process. As heat is only a by-product of the electricity production and therefore the production equations do not differ from those for electricity, the heat production is not explicitly described.

The size of the engine restricts with *p\_fixElecMonth* the maximal output of electricity in each month. According to the available size classes, the maximal outputs are 150kW, 250kW and 500kW, respectively, at 8.000 operating hours per year. This number of hours stems from the assumption that the biogas plant is not operating for 9% of the available time due to maintenance, etc.

fixkWel\_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ (t\_n(t,nCur) and (v\_prodElec.up(bhkw,eeg,t,nCur,m) ne 0)) ..  
  
 v\_prodElec(bhkw,eeg,t,nCur,m)  
  
 =l= v\_useBioGasPlant(bhkw,eeg,t,nCur) \* p\_fixElecMonth(bhkw,m) \* p\_scenRed(eeg);

The production process of electricity, *v\_prodElec,* is constructed in a two-stage procedure. First, biogas [[4]](#footnote-101), *v\_methCrop/Manure,* is produced in the fermenter as the product of crops and manure, *v\_usedCrop/Manure,* and the amount of methane content per ton fresh matter of the respective input. Second, the produced methane is combusted in the engine in which the electricity-output, *v\_prodElecCrop/Manure,* is calculated by the energy content of methane, *p\_ch4Con,* and the conversion efficiency of the respective engine, *p\_bhkwEffic*.

methCrop\_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_methCrop(bhkw,eeg,t,nCur,m)  
  
 =e= sum(crM(biogasFeedM), v\_usedCropBiogas(bhkw,eeg,crM,t,nCur,m) \* p\_crop(crM) );

methManure\_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_methManure(bhkw,eeg,t,nCur,m)  
  
 =e= sum(curmaM, v\_usedManBiogas(bhkw,eeg,curmaM,t,nCur,m) \* p\_manure(curmaM) );

kWel\_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ (t\_n(t,nCur) and (v\_prodElec.up(bhkw,eeg,t,nCur,m) ne 0)) ..  
  
 v\_prodElec(bhkw,eeg,t,nCur,m)  
  
 =l= v\_useBioGasPlant(bhkw,eeg,t,nCur) \* p\_fixElecMonth(bhkw,m) \* p\_scenRed(eeg);

kWelCrop\_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_prodElecCrop(bhkw,eeg,t,nCur,m)  
  
 =e= v\_methCrop(bhkw,eeg,t,nCur,m) \* p\_ch4Con \* p\_bhkwEffic(bhkw,"el") \* p\_transLosses;

kWelManure\_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_prodElecManure(bhkw,eeg,t,nCur,m)  
  
 =e= v\_methManure(bhkw,eeg,t,nCur,m) \* p\_ch4Con \* p\_bhkwEffic(bhkw,"el") \* p\_transLosses;

The bonus structure of the EEG 2012 requires a differentiation between the two input classes: crop and manure. Thus, the production process is separated in methane produced from the *Crop* input class and the *Manure* input class.

The production technology imposes a second bound by connecting a specific fermenter volume, *p\_volFermMonthly,* to each engine size. The fermenter volume is exogenously given under the assumption of a 90-day hydraulic retention time and an input mix of 70% maize silage and 30% manure. Hence, the input quantity derived from crops, *v\_usedCropBiogas,* and manure, *v\_usedManBiogas,* is bound by the fermenter size, *v\_totVolFermMonthly.*

fixKW\_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_totVolFermMonthly(bhkw,eeg,t,nCur,m)  
  
 =l= v\_useBioGasPlant(bhkw,eeg,t,nCur) \* p\_volFermMonthly(bhkw) \* p\_scenred(eeg);

totVolFerm\_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_totVolFermMonthly(bhkw,eeg,t,nCur,m) =g=  
  
 sum(crM(biogasFeedM), v\_usedCropBiogas(bhkw,eeg,crM,t,nCur,m))  
  
 + sum(curmaM, v\_usedManBiogas(bhkw,eeg,curmaM,t,nCur,m) );

The inputs for the fermentation process can be either externally purchased, *v\_purchCrop/Manure,* or produced on farm, *v\_feedBiogas/v\_volManBiogas*. Additionally, the module accounts for silage losses for purchased crops, as crops from own production already includes silage losses in the production pattern of the farm. Currently, the model includes only cattle manure, maize silage and grass silage as possible inputs.

usedCropBioGas\_(curBhkw(bhkw),curEeg(eeg),crM(biogasFeedM),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_usedCropBiogas(bhkw,eeg,crM,t,nCur,m)  
  
 =e= ( v\_purchCrop(bhkw,eeg,crM,t,nCur,m) $ selPurchInputs(crM) \* p\_silageLoss)  
 + v\_feedBioGas(bhkw,eeg,crM,t,nCur,m) $ SUM(sameas(curProds,crM),1);

manureTot\_(curBhkw(bhkw), curEeg(eeg),curmaM,tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_usedManBiogas(bhkw,eeg,curmaM,t,nCur,m)  
 =e=  
 v\_purchManure(bhkw,eeg,curmaM,t,nCur,m) $ selPurchInputs(curmaM)  
$ifi %herd%==true + sum(curmanchain $ (not sameas (curmanChain,"LiquidBiogas")) , v\_volManBiogas(curmanchain,bhkw,eeg,curmaM,t,nCur,m))  
 ;

volManBioGas\_(curmanchain, tCur(t),nCur) $ (t\_n(t,nCur) $ (not sameas (curmanchain,"LiquidBiogas"))) ..  
  
 v\_manQuant(curManChain,t,nCur) $ (not sameas (curmanchain,"LiquidBiogas"))  
  
 =G= sum( (manchain\_mam(curmanchain,curmam),curbhkw(bhkw),curEeg(eeg),m) $(not sameas (curmanchain,"liquidBiogas")), v\_volManBiogas(curmanchain,bhkw,eeg,curmaM,t,nCur,m)) ;

The third bound imposed by the production technology is the so called digestion load (*Faulraumbelastung*). The digestion load, *p\_digLoad,* restricts the amount of organic dry matter within the fermenter to ensure a healthy bacteria culture. The recommended digestion load of the three different fermenter sizes ranges from 2.5 to 3 [[5]](#footnote-102) and is converted into a monthly limit.

fixdigLoad\_(curBhkw(bhkw),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_digLoad(bhkw,t,nCur,m) =l= sum(curEeg(eeg), v\_useBioGasPlant(bhkw,eeg,t,nCur) \* p\_digLoad(bhkw,m)) ;

digLoad\_(curBhkw(bhkw),tCur(t),nCur,m) $ t\_n(t,nCur) ..  
  
 v\_digLoad(bhkw,t,nCur,m) =l= sum(curEeg(eeg), v\_useBioGasPlant(bhkw,eeg,t,nCur) \* p\_digLoad(bhkw,m)) ;

The data used for the fermenter technology can be seen in *\coeffgen\fermenter\_tech.gms*

## Restrictions Related to the Renewable Energy Act

Within the legislative text of the different Renewable Energy Acts different restrictions were imposed in order to receive certain bonuses or to receive any payment at all. In the biogas module most bonuses for the EEG 2004 and EEG 2009 are inherently included such as the KWK-Bonus and NawaRo-Bonus, i.e. the plant is already defined such that these additional subsidies on top of the basic feed-in tariff can be claimed. Additionally, the biogas operator has the option to receive the Manure-Bonus, if he ensures that 30% of his input quantity is manure based, as can be seen in the following code.

manureRes\_(curBhkw(bhkw),eegMan(eeg),tCur(t),nCur,m) $ (t\_n(t,nCur) $ curEeg(eeg)) ..  
  
 sum(curmaM, v\_usedManBiogas(bhkw,eeg,curmaM,t,nCur,m)) =g= v\_totVolFermMonthly(bhkw,eeg,t,nCur,m)\*0.3 ;

Furthermore, the EEG 2012 imposes two requirements which have to be met by the plant operator to receive any statutory payment at all. First, the operator must ensure that not more than 60% of the used fermenter volume, *v\_totVolFermMonthly,* is used for maize. Second, under the assumption that the operator uses 25% of the heat emitted by the combustion engine for the fermenter itself, he has to sell at least 35% of the generated heat externally;

maizeRes\_(curBhkw(bhkw),eegDif(eeg),biogasFeedM,tCur(t),nCur,m) $ (curEeg(eeg) $ t\_n(t,nCur)) ..  
  
 v\_usedCropBiogas(bhkw,eeg,"maizSil",t,nCur,m) =l= 0.6 \* v\_totVolFermMonthly(bhkw,eeg,t,nCur,m);

heatRes\_(curBhkw(bhkw),eegDif(eeg),tCur(t),nCur,m) $ (curEeg(eeg) $ t\_n(t,nCur)) ..  
  
 v\_sellHeat(eeg,t,nCur) =g= p\_minHeatSold \* v\_prodHeat(eeg,t,nCur);

Changes made in EEG 2014 and the amendment of 2016 has not been included in the model yet.

# Dynamic Character of FarmDyn

## The Fully Dynamic Version

As described in earlier sections, the model template optimises the farm production process over time in a fully dynamic setting, i.e. all time points are simultaneously considered. Connecting different modules over time (t1-tn) allows considering biologic and economic path dependencies.

As can be seen from Figure 6, the temporal resolution varies across different parts of the template module. Cropping decisions are annually implemented, whereas the intra-year resolution of the herd size module can be flexibly chosen by the user with a minimal resolution of one month.

Concerning fodder composition, decision points in each year are every three months. This provides the decision maker a more flexible adjustment to feed requirements of the herd (conditional on lactation phase), his resources and prices respectively availability of pasture, silage and concentrates. Furthermore, as stated in the manure module, the applications of manure or synthetic fertilisers, as well as the stored manure amounts on farm are implemented on monthly level.

The optimal production plan over time is not simulated in a recursive fashion from year to year, but all variables of the planning horizon are optimised at once. Consequently, decisions at some point in time also influence decisions before and not only after that point. For instance, an increase in the herd at some point might require increased raising processes before.

## Short-Run and Comparative Static Version

The short-run version considers only one year and does not comprise a liquidation of the enterprise. The comparative static version replaces the herd dynamics by a steady state model where, for example, the cows replaced in the current year are equal to the heifers in the current year, which in turn are equal to the calves raised in the current year. In the comparative static mode, the vintage model for investments in buildings and machinery is replaced by a setting in which the investment costs are related to one year. Nevertheless, the binary character can be maintained.

# Dealing with risk and risk behaviour: deterministic versus stochastic model versions

!!! danger "Prototype feature" This feature has not been thoroughly tested and should be used with caution.

!!!abstract The default layout of the model maximises the NPV over the simulation horizon in a deterministic setting. The stochastic programming extensions introduces decision trees based on mean reverting processes for the output and input price levels and renders all variable state contingents, calculating the option value of full flexibility in management and investment decision over the simulation horizon. A tree reduction algorithm allows exploiting the outcome of large-scale Monte-Carlo simulations while avoiding the curse of dimensionality. Besides risk neutral maximisation of the expected NPV, different types of risk behaviour such as MOTAD, Target MOTAD or value at risk can be used in conjunction with the stochastic programming extension.

## Overview

The FarmDyn model comprises since the first versions optionally stochastic components. In the current version, two set-ups are possible:

1. A deterministic version
2. A fully stochastic programming version where all variables are state contingent and unbalanced stochastic trees are used.

In the deterministic version, no parameter is stochastic and hence no variable state contingent. The equations, variables and certain parameter carry nevertheless indices for nodes in the decision tree and SON, but these refer in any year to a deterministic singleton. In the partly stochastic simulation version of the FarmDyn model, farm management and investment characters with a longer-term character are not state contingent and hence must allow managing all SONs in any year. For example, in case of machine depreciation based on use, the investment decisions must ensure the maximum use in any year and SON.

The fully Stochastic Programming (SP) version of the model introduces scenario trees and renders all variables in the model stage contingent to yield a fully dynamic stochastic approach. That is only feasible in conjunction with a tree reduction approach: even if we would only allow for two different states in each year (= decision nodes), we would end up after twenty years with 210 ~ 1 Million leaves in the trees. Given the number of variables and equations in any year, the resulting model would be impossible to generate and solve. In the following, we briefly discuss the changes to model structure and how the decision tree and the related random variable(s) are constructed.

The SP version of the model can be combined with a number of risk behavioural models to maximise the expected utility. Given the complex character of the remaining modules in the model, only those extensions were chosen which can be implemented in a MIP framework, hence, a non-linear approach such as an E-V approach are not considered. The available risk models (value at risk (var), conditional value at risk, MOTAD and target MOTAD) are discussed in Risk behaviour section below.

### Objective Function in the deterministic

In the deterministic version of the model, we consider a maximisation of NPV of profit under a discount rate. The farm is assumed to be liquidated at the end of the planning horizon, i.e. the cow herd, machinery, land are sold and loans are paid back. Any remaining equity is discounted to its NPV; therefore, a definition close to the flow-to-equity approach is used:

OBJE\_ ..  
\*  
 v\_obje =L=  
 v\_objeMean  
  
\*  
\* --- penalty for negative deviation from mean NPV (similar MOTAD) or target MOTAD / ES  
\*  
$ifi %stochProg%==true - v\_expNegDevNPV \* p\_negDevPen $ (not p\_expShortFall)  
$ifi %stochProg%==true - v\_expShortFall \* p\_negDevPen $ (not p\_expShortFall)  
$ifi %stochProg%==true + v\_expShortFall \* p\_negDevPen $ p\_expShortFall  
 ;

Further on, fully dynamic optimisation assumes that the decision maker is fully informed about the future such that the economically optimal farm plan over the chosen planning horizon is simulated.

## The Stochastic Programming version with full stage contingency

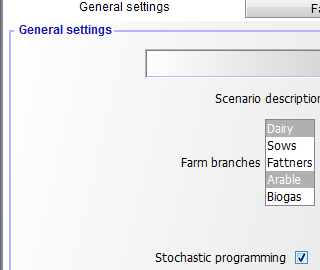
As opposed to the deterministic or partly stochastic version, in the stochastic programming version all variables are state contingent. The stochastic version considers different future developments over time, currently implemented for selected output and input prices, i.e. price paths. These paths do not need to have equal probability. The stochastic programming (SP) approach includes a decision tree that reflects decision nodes where each node has leaves with probability of occurrence. All decisions are contingent on the SON in the current year, and decisions in subsequent years depend on decisions made on previous nodes (=stages) on the path to a final leave. In the SP, all production and investment decisions in any year are hence depicted as state-contingent, i.e. they reflect at that time point the different futures which lay ahead, including future management flexibility. Also the timing of investments is hence state contingent.

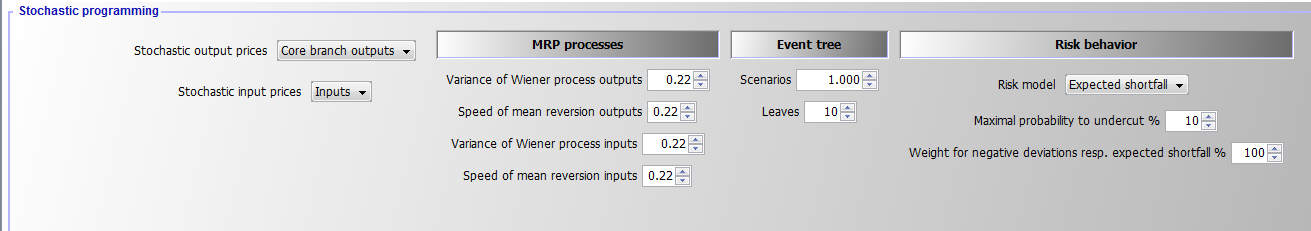
All variables and equations carry the index *nCur*, which indicates the current node in the decision tree. Equally, the node needs to be linked to the correct year, which is achieved by a dollar operator and the *t\_n* set, for instance as in the following equation which was already shown above. Whereas in the deterministic version, there is just one dummy node for each year, in the stochastic version, potentially different states and thus nodes are found for decision variables and equations in any one year.

The revised objective function maximises the probability weighted average of the final liquidity for each final leave in the decision tree:

OBJE\_ ..  
\*  
 v\_obje =L=  
 v\_objeMean

The number of uncompressed scenarios to start with and the desired number of leaves in the final reduced tree are defined via the GUI if the SP module is switched on:





That information enters the declarations in *model\templ\_decl.gms.* If the stochastic programming extension is switched off, there is only one node (which is indicated by a blank space, " ") and the model collapses to a deterministic one:

$iftheni.sp not %stochProg%==true  
\*  
\* --- dummy implementation of SP frameworK  
\* there is one universal node, i.e. that is the deterministic version  
\*  
  
 set n "Decision nodes in tree" / " " /;  
 set t\_n(t,n) "Link betwen year and decision node";  
 t\_n(t," ") = YES;  
  
 set anc(n,n) "Is the second node the node before first one?";  
 anc(" "," ") = YES;  
  
 set isNodeBefore(n,n) "Is the second node before first one?";  
 isNodeBefore(" "," ") = YES;  
  
 set sameScen(n,n) "The two nodes belong to the same scenario";  
 sameScen(" "," ") = YES;  
  
 set leaves(n) / " " /;  
  
  
 parameter p\_probN(n);  
 p\_probN(" ") = 1;

The changes in the listing are minimal compared to the previous version without the SP extension, only one point more in each variable or equation name is included, which indicates the blank common node (between the dots), for example as following:



With the SP extension, information is needed about ancestor nodes and nodes before the current one:

$else.sp  
  
$evalglobal nt %lastYear%-%firstYear%+1  
  
  
$evalGlobal nNode (%nt%-1) \* %nOriScen% + 1  
\*  
\* --- sets and parameters are population in coeffgen/stochProg.gms  
\*  
 set n /n1\*n%nNode%/;  
 set t\_n(t,n) "Link betwen year and decision node";  
 set anc(n,n) "Is the second node the node before first one?";  
 set isNodeBefore(n,n) "Is the second node before first one?";  
 set sameScen(n,n) "The two nodes belong to the same scenario";  
 set leaves(n);  
 parameter p\_probN(n);  
  
$endif.sp

### Generating Random Variable(s) and the decision tree

The generation of decision tree and related random variable(s) consists of three major steps:

1. **Generation of a predefined number of scenarios** which describe equally probable future developments for the random variables considered, i.e. in that uncondensed tree, the probabilities of the scenarios are identical.
2. **Generating a reduced decision tree** from all possible scenarios most of the nodes are dropped and the remaining nodes receive different probabilities.
3. **Defining the symbols in GAMS** according to step 1 and 2.

As GAMS can become quite slow with complex loops, the first step is implemented in Java. Currently, two random variables (one for output and one for input price changes) are generated based on two independent logarithmic mean-reverting processes (MRPs), the log is introduced to avoid negative outcomes. The variance and speed of reversion are defined on the GUI as shown above, under an expected mean of unity. The starting price multiplier is also set to unity. Each path of input and output prices are simulated once in the SP.

The Java program is called from GAMS to pass the information on the number of decision nodes (= simulated time points) and the desired number of scenarios to the program:

$iftheni.stochPrices not "%StochPricesOutputs%"=="None"  
  
 execute "java -Djava.library.path=..\gui\jars -jar ..\gui\mrpfan.jar %nt% %nOriScen% %scrdir%\\mrp.gdx 1 1 %varOutputs% %lambdaOutputs% 2>1"  
  
 execute\_load "%scrdir%\\mrp.gdx" p\_randVar,tn,anc;  
 p\_randVar("priceOutputs",n) = p\_randVar("Price",n);  
$endif.stochPrices

$iftheni.stochPrices not "%StochPricesInputs%"=="None"  
  
 execute "java -Djava.library.path=..\gui\jars -jar ..\gui\mrpfan.jar %nt% %nOriScen% %scrdir%\\mrp.gdx 1 1 %varInputs% %lambdaInputs% 2>1"  
  
 execute\_loadpoint "%scrdir%\\mrp.gdx" p\_randVar,tn,anc;  
 p\_randVar("priceInputs",n) = p\_randVar("Price",n);  
  
$endif.stochPrices

The Java process stores the generated random developments along with the ancestor matrix in a GDX file. The following Figure shows an example of a decision tree as generated by the Java program for five years and four scenarios, illustrated as a fan. The common root node *1*, the node in the first year, is on the left side of the Figure. The nodes 2, 5, 8, 11, 14 are in the second year. Each second year node has its own set of followers, and all nodes besides *1* have the same probability of 20%.

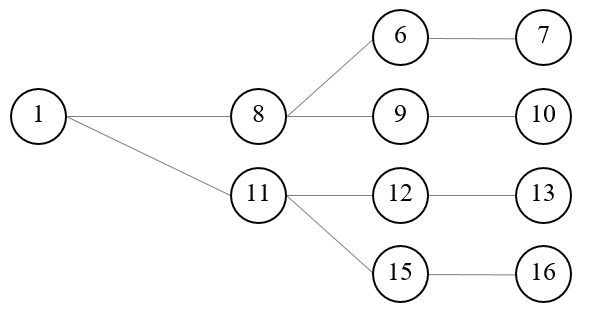


Figure 8: Example of an input decision tree organised as a fan. Source: Own illustration

Increasing the number of years leads to a proportional increase in the number of nodes. For complex stochastic processes such as MRPs, many paths are needed, each reflecting a Monte-Carlo experiment, to properly capture the properties of the stochastic process. This leads to the curse of dimensionality, as the number of variables and equations in the model increases quadratic with the number of years and number of Monte-Carlo experiments. As MIP models are NP-hard to solve, that quickly leads to models which cannot be solved in any reasonable time. Hence, in a next step, the tree must be reduced to avoid that curse of dimensionality, achieved by using the [*SCENRED2*](https://www.gams.com/help/index.jsp?topic=%2Fgams.doc%2Ftools%2Fscenred2%2Findex.html) utility comprised in GAMS (Heitsch & Römisch 2008, 2009). The algorithm deletes nodes from the tree and adds the probability of dropped nodes to a neighboring remaining one.

The example in the Figure below depicts a hypothetical tree from tree reduction with four final leaves generated from the tree given in Figure 8. Each scenario starts with the same root node, *1*, for which the information is assumed to be known for certain, i.e. the probability for this root node N*1,* which falls in the first year, is equal to unity and ends with one of the final leaves, 7, 10, 13 or 16. In the second year two nodes are kept in the example, each depicting possible states of nature with their specific followers while potentially differing in their probabilities. Node number 8 has a probability of 60% as it represents in the reduced tree three original nodes while node 11 has one of 40%. The strategy chosen for each of these nodes depends simultaneously on the possible future development beyond that node while being conditioned on the decisions in the root node (which itself depends on all follow up scenarios).

: Figure 9: Example of a reduced tree. Source: Own illustration

The example can also help to understand better some core symbols used in the code and relations in the SP extension. The nodes remaining in the reduced tree are stored in the set *nCur*. The set *t\_n* would match the first year with the first node, the second year with the nodes 8 and 11 etc. For the node 15, the ancestor set *anc* would be set to *anc* (*n15*,*n11*) to indicate that node 11 is the node before 15 on the scenario ending with leave 16. *Isbefore* (*n16*,*x*) would be true for x=16,15,11 and 1 and comprises the complete scenario ending with the final leave 16. The probabilities for node 8 and 11 must add up to unity as they relate to the same time point. The same holds for the node set (6, 9, 12, 15) for the third year. Hence, the decision at the root node 1 influences all subsequent scenarios, whereas the stage contingent decisions at node 8 influence directly the scenarios ending with leaves 7 and 10. The root node reflects all scenarios simultaneously and consequently an indirect influence between all nodes exists.

Furthermore, in a programming context no backward or forward recursion solution tactic is possible to find the best strategy as the number of strategies is normally not countable (the solution space is bounded, but there exist typically an infinite number of possible solutions). Finding a solution is further complicated by the fact that a larger number of variables have an integer or binary character. MIP problems are NP-hard, i.e. the solution time increases dramatically in the number of integers. This makes it especially important to find an efficient way to reduce the number of nodes considered to keep the solution time in an acceptable range

In the current implementation, the tree size which also determines the overall model size is steered by setting exogenously the number of final nodes.

$setglobal sr2prefix test  
$setglobal treeGen on  
  
$iftheni.runSR2 %treeGen%==on  
\*  
\* --- scenario tree construction from fan  
\*  
 $$libinclude scenRed2  
\*  
\* --- information for SCENRED: option file and options from interface  
\*  
 ScenredParms('sroption') = 1;  
 ScenredParms('num\_time\_steps') = %nt%;  
 ScenredParms('num\_nodes') = card(n);  
 ScenredParms('num\_random') = %MRP%;  
 ScenredParms('num\_leaves') = %nOriScen%;  
 ScenredParms('visual\_red') = 1;  
 $$libinclude runScenRed2 %sr2Prefix% tree\_con n anc p\_probN ancRed p\_probRed p\_randVar  
  
$endif.runSr2  
  
 ;  
\*  
\* --- load information from ScenRed2  
\*  
 execute\_load 'sr2%sr2Prefix%\_out.gdx' ancRed=red\_ancestor,p\_probRed=red\_prob;

Based on the information returned from the scenario reduction utility, the set of active nodes, *nCur,* is determined:

\* --- actives nodes are those which have an updated probability  
\*  
 option kill=nCur;  
 nCur(n) $ p\_probRed(n) = YES;  
\*  
\* --- cleanse link between time points and nodes from unused nodes  
\*  
 tn(tnum,n) $ (not nCur(n)) = no;  
\*  
\* --- map into year set used by model  
\*  
 t\_n(tCur,nCur) $ sum(tn(tnum,nCur) $ (tnum.pos eq tCur.pos),1) = YES;  
 t\_n(tBefore,"n1") = YES;  
  
\*  
\* --- take over cleansed ancestor matrix and probabilities  
\*  
 option kill=anc;  
 anc(nCur,nCur1) = ancRed(nCur,nCur1);  
 anc("n1","n1") = YES;  
 p\_probN(n) = p\_probRed(n);

A little bit trickier is to efficiently find *all* nodes that are before a given node in the same scenario (these are often nodes shared with other scenarios such as the root node, see Figure 9 above). This is achieved by an implicit backward recursion over a year loop:

loop(tCur,  
 loop(anc(nCur,nCur1),  
 isNodeBefore(nCur,nCur2) $ isNodeBefore(nCur1,nCur2) = YES;  
 );

As indicated above, the set *anc (nCur, nCur1)* indicates that decision node *nCur1* is the node before the node *nCur*, i.e. they belong to the same scenario. That is used in lag and lead operators, e.g.:

hasFarmOrder\_(tCur(t),nCur) $ (tCur(t-1) $ t\_n(t,nCur)) ..  
  
 v\_hasFarm(t,nCur) =L= sum(t\_n(t-1,nCur1) $ anc(nCur,nCur1), v\_hasFarm(t-1,nCur1));

The *isNodeBefore(nCur,nCur1)* relation depicts all nodes, *nCur1,* before node *nCur* in the same scenario, including the node *nCur* itself. An example gives:

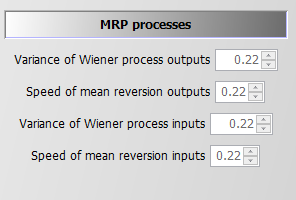
\* --- steady state: starting herds before the first fully simulated year are equal to that one  
\*  
 sum(t\_n(tBefore,nCur1) $ sameScen(nCur1,nCur),v\_herdStart(herds,breeds,tBefore,nCur1,m))  
 =E= v\_herdStart(herds,breeds,t,nCur,m);

**Important Aspects to remember!**

2. The normal case is that the objective value increases when  
considering stage contingency under risk neutrality. This is due to the effect that profits increase over-proportionally in output prices under profit maximisation.

|  |
| --- |
| 3. The solution time of the model can be expected to increase substantially with the SP extension switched on. MIP models are non-convex and NP-Complete problems. To our knowledge there is no existing sting polynomial-time algorithm, which means that the solution time to optimality increases typically dramatically in the number of considered integers. Even small problems can take quite long to be solved even towards moderate optimality tolerances and not fully optimality. This holds especially if the *economic signal* to choose between one of the two branches of a binary variable is weak, i.e. if the underlying different strategy yield similar objective values. Which is unfortunately exactly the case where the SP programming approach is most interesting (if there is one clearly dominating strategy rather independent e.g. of a reasonable range of output prices, considering different future inside that reasonable range is not necessary). |

The interface allows to define the parameters of the logarithmic Mean Reverting processes (MRP) with an expected mean and start value of log(1):



### Introduction of the Random Variable(s)

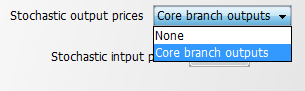
The notion *random variable* only implies that the variable has an underlying probability distribution and not that it is a decision variable in our problem. Consequently, random are parameters in GAMS and are not declared as variables. As mentioned in the section above, in the SP version of the model the MRPs are simulated in Java that generate deviations around unity, i.e. we can multiply a given mean price level for an output and/or an input (e.g. defined by user on the interface) with the node specific simulated random price multiplier. If two MRPs are used, they are currently assumed to be uncorrelated. One path from the root to a final leave thus depicts a time series of input and output price deviations from the mean of the stochastic version.

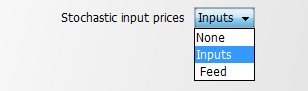
The random variable can impact either revenue, *salRev\_*, by introducing state specific output price(s) and/or cost for buying inputs, *buyCost\_*, by state specific input price(s):

salRev\_(tCur(t),nCur) $ t\_n(t,nCur) ..  
\*  
 v\_salRev(t,nCur) =e= sum( (curProds(prodsYearly),sys) $ (v\_saleQuant.up(prodsYearly,sys,t,nCur) ne 0),  
 p\_price(prodsYearly,sys,t)  
$iftheni.sp "%stochProg%"=="true"  
\*  
\* --- a product is both output and input, use price of inputs to avoid a situation  
\* where the product can be bought cheaper than it is sold  
\*  
 \* ( 1 + (p\_randVar("priceOutputs",nCur)-1) $ (randProbs(prodsYearly) and (not sum(sameas(prodsYearly,inputs),1)))  
 + (p\_randVar("priceInputs",nCur)-1) $ (randProbs(prodsYearly) and ( sum(sameas(prodsYearly,inputs),1)))  
 )  
$endif.sp  
 \* v\_saleQuant(prodsYearly,sys,t,nCur));

buyCost\_(curInputs(inputs),sys,tCur(t),nCur) $ (t\_n(t,nCur) $ p\_inputprice(inputs,sys,t) $ (v\_buy.up(inputs,sys,t,nCur) ne 0)) ..  
  
 v\_buyCost(inputs,sys,t,nCur) =e= p\_inputprice(inputs,sys,t)  
$iftheni.sp %stochProg%==true  
 \* ( 1 + (p\_randVar("priceInputs",nCur)-1) $ randProbs(inputs) )  
$endif.sp  
 \* v\_buy(inputs,sys,t,nCur);

The decision whether prices are treated as random variables is steered via the interface:





In the case where neither input nor output prices are random a run time error will occur.

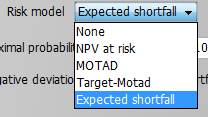
The core branches are defined in *coeffgen\stochprog.gms*:

$iftheni.stochPrices "%StochPricesOutputs%"=="Core branch outputs"  
  
\*  
 $$ifi "%farmBranchArable%" == "on" option kill = randProbs; randProbs(set\_crop\_prods) = yes;  
 $$ifi "%pigHerd%" == "on" option kill = randProbs; randProbs(set\_pig\_prods) = yes;  
 $$ifi "%farmBranchDairy%" == "on" option kill = randProbs; randProbs(set\_dairy\_prods) = yes;  
 $$ifi "%farmBranchBeef%" == "on" option kill = randProbs; randProbs(set\_beef\_prods) = yes;  
  
$endif.stochPrices

That means that dairy production takes precedence over other branches and pigs over arable cropping, assuming that arable crops are typically not the core farm branch in mixed enterprises.

## Risk Behaviour

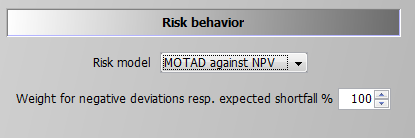
The model allows introducing four different risk behaviour options in the stochastic programming version in addition to risk neutral behaviour (None):



All risk measures relate to the distribution of the NPV, i.e. changes in expected returns aggregated over the full simulation horizon, and do not take fluctuations of the cash flow for individual years into account. This is reasonable as the farmer is assumed to have access to credits which can be used to overcome short-term cash constraints. The cost of using credits as a risk management option is considered endogenously in the model as farmers have to pay interest on these credits which reduces the NPV. Still, considering that risk is accessed here with regard to changes in accumulated final wealth over a long planning horizon is crucial when comparing the approach and results to risk analysis based e.g. on a comparative static analysis of yearly variance of gross margins.

### MOTAD for Negative Deviations against NPV

The first and simplest risk model modifies the objective function: it maximises a linear combination of the expected NPV and the expected mean negative deviation from the NPV.



The formulation builds on MOTAD (Minimization of Total Absolute Deviations) as a linear approximation of the quadratic E-V model proposed by Hazell 1971. The approach was developed at a time where quadratic programming was still not considered feasible for even medium sized problems. Under normality, it can be shown that the absolute deviations and the variance show approximately a linear relationship, the factor between the two depends however in a non-linear way on the number of observations. Mean absolute deviations can also be understood as a robust estimate for the variance.

Our approach builds on an often used modification by only considering down-side risk, i.e. only negative deviations from the simulated mean are taken into account:

negDevNPV\_(nCur) $ t\_n("%lastYearCalc%",nCur) ..  
  
 v\_objeN(nCur) + v\_negDevNPV(nCur) =G= v\_objeMean;

This approach is especially relevant if the deviation above and below the objective function are not by definition symmetric. However, as the distribution itself is determined in our stage contingent approach endogenously, symmetry makes limited sense. The expected mean deviation is calculated as:

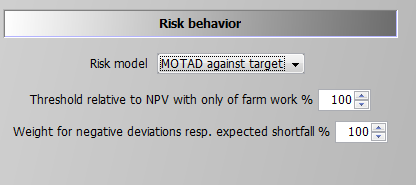
expNegDevNPV\_ ..  
  
 v\_expNegDevNPV =E= sum(nCur $ t\_n("%lastYearCalc%",nCur), v\_negDevNPV(nCur)\*p\_probN(nCur));

And subtracted from the objective function (see equation *OBJE\_*),

\* --- penalty for negative deviation from mean NPV (similar MOTAD) or target MOTAD / ES  
\*  
$ifi %stochProg%==true - v\_expNegDevNPV \* p\_negDevPen $ (not p\_expShortFall)  
$ifi %stochProg%==true - v\_expShortFall \* p\_negDevPen $ (not p\_expShortFall)  
$ifi %stochProg%==true + v\_expShortFall \* p\_negDevPen $ p\_expShortFall  
 ;

The reader should note that the standard MOTAD approach by Hazell and described in text books is based on expected gross margins and deviation thereof, whereas in this model an approach in the context of dynamic stochastic programming approach is used. The expected mean returns for each activity and related (co)variances are not known beforehand in our model such that an E-V approach would be numerically demanding. This holds especially for our large-scale MIP problem, such that avoiding quadratic formulations, as required by an E-V approach, has its merits. Finally, it should be noted that these equations are always active for information purposes. The weight in the objective is set to a very small number when other types of risk behaviour are simulated.

### MOTAD for Negative Deviations against Target



The only difference to the *MOTAD against NPV* option described before is that negative deviations are defined against a target set by the user. That target is based on a relative threshold multiplied with the simulated objective value in the case of no farming activity, therefore, income is only drawn from off-farm work, decoupled payments and interest. This income level is used as the absolute benchmark level which can be modified by the user with the percentage multiplier entered in the GUI. This effects the following equation:

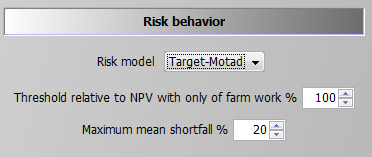
shortFall\_(nCur) $ (t\_n("%lastYearCalc%",nCur) $ (p\_npvAtRiskLim gt 1)) ..  
  
 v\_objeN(nCur) + v\_shortFall(nCur) =G= p\_npvAtRiskLim;

Using this information the expected shortfall is defined:

expShortFall\_ $ ( p\_expShortFall or p\_maxShortFall ) ..  
  
 v\_expShortFall =E= sum(nCur $ t\_n("%lastYearCalc%",nCur), v\_shortFall(nCur)\*p\_probN(nCur));

The expected shortfall then enters the objective function.

### Target MOTAD



The second option is what is called *Target MOTAD* in programming modelling. It has some relation to *MOTAD* as it also takes negative deviation from a pre-defined threshold into account, *p\_npvAtRiskLim*.

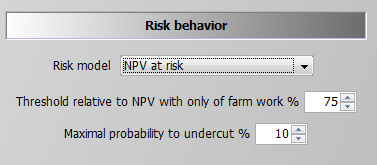
The difference to the approach above is that the expected shortfall below the predefined threshold does not enter the objective function, but acts as an upper bound. Hence, the shortfall of NPV cannot be lower than certain level:

maxShortFall\_ $ ( (p\_npvAtRiskLim gt 1) $ (p\_maxShortFall gt 0) ) ..  
  
 sum(nCur $ t\_n("%lastYearCalc%",nCur), v\_shortFall(nCur)\*p\_probN(nCur))  
 =L= p\_maxShortFall\*p\_npvAtRiskLim;

### Value at Risk Approach

Contrary to the *MOTAD* approaches discussed before, the *Value at Risk* *(VaR)* and *conditional value at risk (CVaR)* approaches (see next section) require additional binary variables and thus are numerically more demanding.

The value (NPV) at risk approach introduces a fixed lower quantile (i.e., introduced as parameter and determined by the user) for the NPV as shown in following illustration. It requires the following user input:



The second parameter defines the maximal allowed probability for simulated objective values to fall below the resulting threshold. The reader should be aware of the fact that only undercutting matters, not by how much income drops below the given threshold. For the *conditional value at risk* at approach see next section.

If the maximal probability is set to zero, the threshold acts as a binding constraint in any SON, i.e. the NPV at any leaf cannot fall below it. The NPV at risk approach does thus not change the equation for the objective function, but introduces additional constraints. The first one drives a binary indicator variable, v\_*npvAtRisk,* which is equal to one if the objective value at a final leaf falls below the threshold:

npvAtRisk\_(nCur) $ (t\_n("%lastYearCalc%",nCur) $ (p\_npvAtRiskLim gt 1)) ..  
  
 v\_objeN(nCur) =G= p\_npvAtRiskLim - v\_npvAtRisk(nCur) \* p\_npvAtRiskLim;

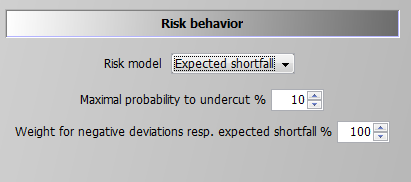
If v\_*npvAtRisk* is zero, the objective value (LHS) for each final leave must exceed the given threshold *p\_npvAtRiskLim*. The second constraint, shown below, adds up the probabilities for those final nodes which undercut the threshold (LHS) and ensures that their sum is below the given maximal probability:

maxProbNpvAtRisk\_ $ ( ( (p\_npvAtRiskLim gt 1) or p\_expShortFall) $ p\_npvAtRiskmaxProb) ..  
  
 sum(t\_n("%lastYearCalc%",nCur), v\_npvAtRisk(nCur) \* p\_probN(nCur)) =L= p\_npvAtRiskmaxProb;

As long as off-farm income is considered deterministic and the relative threshold is below 100%, a solution where only off-farm income is generated should always be a feasible.

### Conditional Value at Risk

The *Conditional Value at risk* approach is also referred to as the expected or mean shortfall. It is the most complex and numerically demanding of the options available and it can be seen as the combination of the VaR approach and target MOTAD with an endogenously determined limit. The decision taker defines hence a quantile, say 10% as in the screen shot below, and the model calculates endogenously the expected shortfall for the lowest 10% of the scenarios. The objective function in the model maximises a linear combination of the expected NPV and the endogenous mean shortfall, subject to a predefined lower quantile:



A first constraint, which is also used for the VaR option, ensures that the sum of the considered cases does not fall below the now endogenously defined limit (equation was already shown above in the section on the Value at Risk Approach):

maxProbNpvAtRisk\_ $ ( ( (p\_npvAtRiskLim gt 1) or p\_expShortFall) $ p\_npvAtRiskmaxProb) ..  
  
 sum(t\_n("%lastYearCalc%",nCur), v\_npvAtRisk(nCur) \* p\_probN(nCur)) =L= p\_npvAtRiskmaxProb;

Additionally, the expected shortfall for any of the final nodes which do not contribute to active lower quantile must be zero, based on a so-called BIGM formulation, i.e. the binary variable *v\_npvAtRisk* is multiplied with a very large number, here with 1.E+7. If *v\_npvAtRisk* for that final leave is zero (= it does not belong to the leaves with the worst NPVs), the left hand side must be zero as well. On the other hand, if *v\_npvAtRisk* is unity, i.e. the final leaves' NPV belongs to the x% worst cases, where x is set by the user, the shortfall for that leave can consider any number determined by the model as the RHS value of 1.E+7 in the case of *v\_npvAtRisk* equal unity never becomes binding.

shortFallTrigger1\_(nCur) $ ( t\_n("%lastYearCalc%",nCur) $ p\_expShortFall ) ..  
  
 v\_shortFall(nCur) =L= 1.E+7 \* v\_npvAtRisk(nCur);

Besides this, any leaves which is not in worst cases set (*v\_npvAtRisk* = 0) must at least generate a NPV which exceeds the best shortfall.

shortFallBound\_(nCur) $ ( t\_n("%lastYearCalc%",nCur) $ p\_expShortFall ) ..  
  
 v\_objeN(nCur) =G= v\_bestShortFall+1 - v\_npvAtRisk(nCur) \* 1.E+7;

For cases at or below the quantile which contributed towards the expected mean shortfall, both the own expected NPV and the best NPV act simultaneously as lower bounds:

shortFallTrigger2\_(nCur) $ ( t\_n("%lastYearCalc%",nCur) $ p\_expShortFall ) ..  
  
 v\_slackNPV(nCur) =L= 1.E+7 \* (1-v\_npvAtRisk(nCur));

shortFallTrigger3\_(nCur) $ ( t\_n("%lastYearCalc%",nCur) $ p\_expShortFall ) ..  
  
 v\_shortFall(nCur) + v\_slackNPV(nCur) =E= v\_objeN(nCur);

Accordingly, the *v\_bestShortFall* splits the expected NPVs in those below and above the relevant quantile.

The cases below that bound define the expected shortfall:

expShortFall\_ $ ( p\_expShortFall or p\_maxShortFall ) ..  
  
 v\_expShortFall =E= sum(nCur $ t\_n("%lastYearCalc%",nCur), v\_shortFall(nCur)\*p\_probN(nCur));

The expected shortfall adds to the objective (in opposite to target MOTAD).

The objective function is hence a trade-off between a higher expected mean NPV and the expected shortfall of cases x% relative to that endogenous mean.

# The Coefficient Generator

## Concept and File Structure

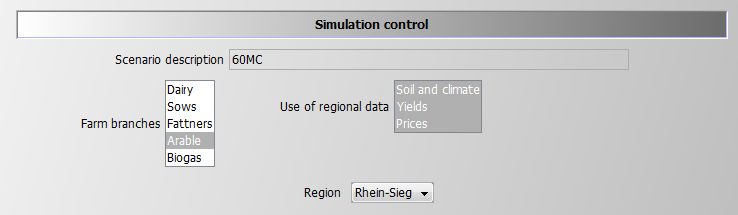
The coefficient generator comprises a number of small modules, realised in GAMS, which define the various exogenous parameters comprised in the template. It is designed such that it can generate from a few central characteristics of the farm (herd size, current milk yield, existing stables and their construction year, labour force and available land) and the realised crop yields a plausible set of coefficients for the template model. The coefficient generator can also be set-up to load parameters for a specific region.

The coefficient generator is divided in:

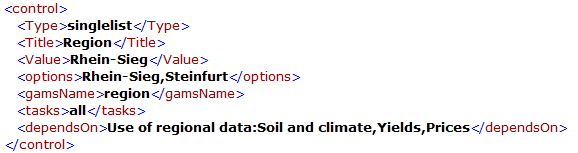
* **Buildings:** includes bunker silos for silage maize and potatoes.
* **Cows**: cows, heifers and calves are defined that have different milk yield potentials. Additionally, a maximum number of lactation is defined. It depends on the milk output level of the lactating cows (diminishes with increasing milk output potential).
* **Credit**: different credit types are defined. These vary by interest rate and payback time.
* **Cropping**: defines different activities for cash-crop production with specific restrictions concerning crop rotation, fertilizer demand and yield potentials.
* **Environmental accounting**: defines environmental impact due to manure and fertiliser application.
* **Farm constructor**: the farm constructor defines the relationships between benchmark data of the farms and production specific endowments e.g. of land, stables and machinery in the initial situation.
* **Farm\_Ini:** Initialises the farms land endowment and plot distribution
* **Feeds**: possible fodder compounds are listed with their specific contents of ingredients (N, C, DM, XP,...).
* **Fermenter\_tech**: includes all data regarding the technical aspects of the biogas fermenter, the different inputs and their related biogas yields.
* **Fertilising**: defines coefficients for various application techniques for organic and synthetic fertilisers.
* **Greening <not yet included>:** Adds the restrictions of the CAP Greening into the model.
* **Indicators**: this module gives a definition of the different GHG indicators and a description of the underlying calculation schemes and parameters. The majority is taken from IPCC methodology and completed by other literature findings.
* **Ini\_herds**: it defines the initial herds of the farm.
* **Labour**: defines labour needs on a monthly basis for herds and crops and wages for the off-farm work.
* **Mach**: defines the different types of machinery that are available for the farmer and it quantifies the useful lifetime (defined according to years or on hourly basis) as well as investments and variable costs.
* **Manure**: quantifies amount of animal excreta with respect to livestock category. For cows manure amount is controlled by yearly milk output level. Furthermore, coefficients for different manure storage and application types are derived by this module.
* **Pigs:** defines output coefficients, production lengths and other variable costs for fatteners and sows.
* **Prices**: different default values are defined if prices for variables are not defined by the GUI.
* **Prices\_eeg**: contains the prices applied in the different EEGs as well as investment prices for different biogas plant parts.
* **Requ**: definitions of requirement functions for lactating cows in relation to their milk yield, live weight etc., as well as for heifers and calves are included in this module.
* **Silos**: in this module the definition of different types of surface reservoirs for liquid manure is set. It differentiates concerning capacity and related investment costs. Furthermore, additional costs of specific coverage types of the surface manure reservoirs are defined for straw coverage and coverage with foil.
* **Stables:** stable types with stable places and required workload for the respective stables for all herd types
* **StochProg**: defines the decision tree and further GAMS symbols used in the stochastic programming version
* **Tech:** defines all machinery, crop specific operation requirements and field working days.

## Handling of Regional Data

The interface allows defining which data should be taken from the regional data base and, in case one or several of these options are selected, to choose a region:



The list of regions is defined in *gui\dairydyn\_default.xml*:



The regional data are stored in three files according to the available options:

* *regionalData\prices.gms* -- input and output prices
* *regionalData\yields.gms* -- crop yields
* *regionalData\Climate\_soil.gms* -- set of climate zone and soil shares

The code is set up in a way that in case no data is found the settings from the interface are used.

### Climate and Soil Data

The climate and soil data are read in by *coeffgen\farm\_ini.gms*. As soil shares determine potentially the size of the plots the information is used in many subsequent programs. The inclusion of the regional data is conditional on the interface settings:

curClimateZone("%curClimateZone%") = YES;  
\*  
\* --- scale soil shares edited by user to add up to unity  
\*  
 p\_soilShare(soil,"Share") = p\_soilShare(soil,"Share") \* 1 / sum(soil1, p\_soilShare(soil1,"Share"));  
  
  
\*  
\* --- Regional climate and soil data (overwrites the data given in the GUI for climate zone and soil)  
\*  
$ifi "%useRegionalDataSoilAndClimate%"=="ON" $include 'regionalData/Climate\_soil.gms'

Soil shares are entered via the interface or a batch file is overwritten if at least one of the soil types data are entered:

p\_soilShare("l","Share") = 0.2 ;

p\_soilShare("m","Share" ) = 0.5 ;

p\_soilShare("h","Share") = 0.3 ;

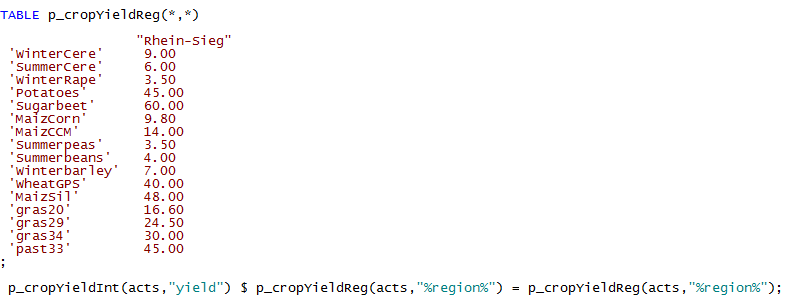
p\_soilShare(soil,"Share") = p\_soilShare(soil,"Share") \* 1 / sum(soil1, p\_soilShare(soil1,"Share"));

### Yield Data

Handling of yields is similar. The inclusion of the data is done by *coeffgen\cropping.gms*:



The crop yields data is entered in a table and overwrites the data from the interface, *p\_cropYieldInt,* only if a non-zero entry is found for the activity, *acts,* and the current region, *"%region%"*. It is important to highlight that in order to increase readability the table is not domain checked. This is despite the fact that the list of activities, *acts,* is defined in *model\templ\_decl* with *$If* conditions which would need to be repeated here as well.

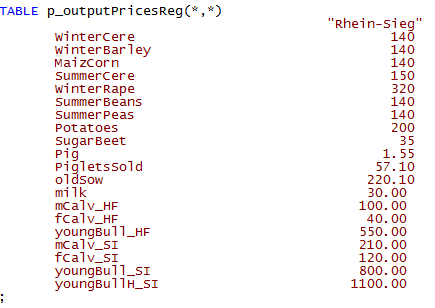


### Price Data

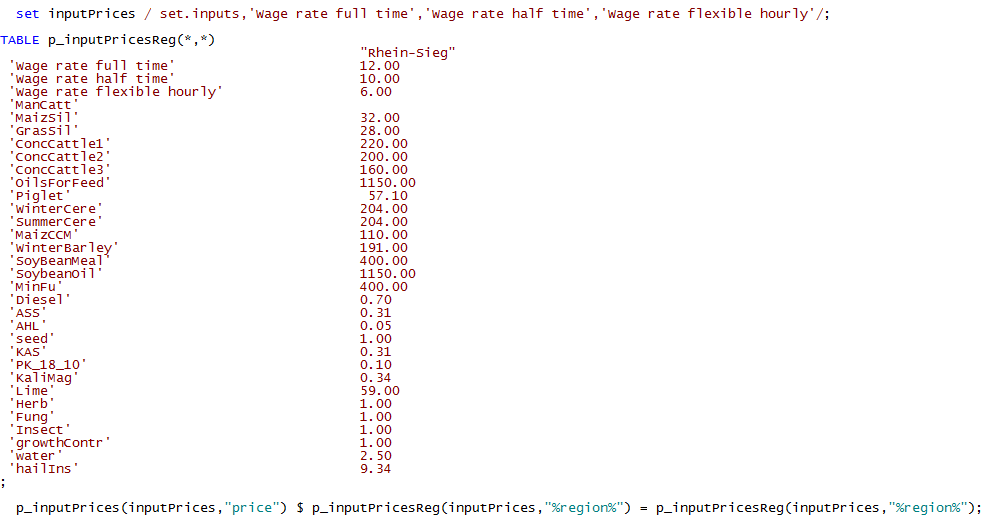
The information on regional prices is included in *coeffgen\prices.gms*:



The file comprises a section for output and another for input prices, both consisting of a table with regional prices and a statement which overwrites the information from the interface:



For price data there is as well no domain checking to increase readability. The section of the inputs is structurally identical to the one shown above:



The updated prices are used in a next step in *coeffgen\prices.gms*:

$ifi "%useRegionalDataPrices%"=="ON" $include 'regionalData/prices.gms'

# Technical Realisation

!!!abstract The model uses GAMS for data transformations and model generation and applies the industry LP and MIP solver CPLEX for solution. The code adheres to strict coding guidelines, for instance with regard to naming conventions, code structuring and documentation, including a modular approach. A set of carefully chosen compilation and exploitation tests is used to check the code. The code is steered by a GUI based on GGIG (ref., Java code) which also support result exploitation.

## Overview of the Technical Realisation

The model template and the coefficient generator are realised in GAMS (General Algebraic Modelling System), a widely used modelling language for economic simulation models. GAMS is declarative (as seen from the template discussion above), i.e. the structure of the model's equation is declared once, and from there different model instances can be generated. GAMS supports scripting for data transformation, extensively used by the coefficient generator and by the post-model reporting.

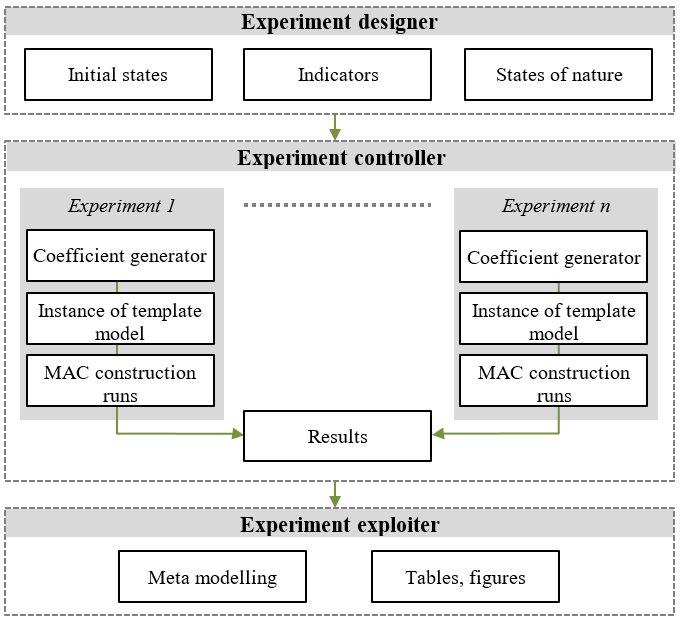


Figure 11: Overview of technical realisation. Source: Own illustration

Additionally, as an extension of the experiment exploiter, *machine learning* (for detailed description see Britz, 2011) can be used to derive correlations and dependencies between model results and available model variables.

## MIP Solution Strategy

In opposition to purely linear problems, Mixed-Integer problem models (MIPs) are far harder to solve. In order to find the optimum, in theory the combinatorial set of all binaries respectively general integer variables would need to be evaluated. Depending on the simulation horizon of FarmDyn, the number of farm branches considered and the time resolution for investment and labour use decisions, a model instance can comprise between a few dozens to more than a thousand binary variables, with often several ten thousand variables and equations in total.

There are huge differences in the quality of LP and more so MIP solvers. Industry solvers such as CPLEX or GUROBI reflect continuous investments into algorithmic improvements over decades. Fortunately, both offer free academic licenses. The code is set-up to work with both solvers to be secured should license conditions change as well as switch in cases one of the solvers outperforms considerably the other. Current tests seem to show a slight advantage for CPLEX. Both solvers can benefit from parallel processing. Model instances should therefore if possible be solved on a multi-core computing server. The option files for the solvers are currently defined such that one core is not used by the program and left free for other processing load.

The relaxed version of the model (where binaries and integers are removed and treated as continuous variables) can typically be solved in a few seconds, and once such a starting point is given, slight modifications to the model take very little time to solve despite the model size. However, regardless of tremendous algorithmic improvements in solving MIPs, the MIP version could take quite long to solve without some solution tactic.

The model code therefore integrates different strategies to speed up the solution process for the MIP. Some of those are generally applicable to MIP problems, typically offered by GAMS and/or the MIP solvers, others follow tactics proposed to speed up the solution time of MIP problems, but require a specific implementation reflecting the model structure. In the following, these strategies are roughly described, starting first with the model generic.

In order to define a lower bound on the objective which allows the solver to cut-off parts of the tree, the model is first solved in relaxed mode (RMIP) with the farm switched off such that income can only be generated by working off-farm (*v\_hasFarm* is fixed to zero). Solving that variant takes less than a second. The solution is used to define the lower cut-off for MIP solver. Next, the model is solved as RMIP with only one SON, and afterwards, the state contingent variables are copied to all other SON, before the RMIP is solved again. The main statements are given in the *exp\_starter.gms* file.

The relaxed (RMIP) solution defines the upper cut-off -- forcing certain variables to only take on integer values can only reduce the objective function. At the same time, it proves a basis for solving the MIP. However, in many instances it has not proven useful to use the solution of RMIP as MIP start starting point, both CPLEX and GUROBI seem to spend considerable time to construct a feasible integer solution from the RMIP solution.

As stated above, solving a MIP problem to its true optimum can be tremendously time consuming. Therefore, typically MIP problems are only solved given an optimality tolerance. The branch-and-cut algorithm used in MIP solvers always provide a safe upper limit for the objective value stemming from a relaxed version of the current tree node. Accordingly, they can quantify the maximal absolute and relative gap to the potentially maximal objective function. Typically, the smaller the desired gap, the larger the number of combination of integer variables the solver needs to test. Forcing the gap to zero requires more or less a test of all combination, i.e. ten-thousands of solves of a LP version of the model with binaries and integers fixed. In most production runs, a relative gap of 0.5% has proven as acceptable. The solver will then stop further search for a better solution once a MIP solution has been found which differs by less from the relaxed best node.

The problem with the gap is clearly that differences between two simulations can not only stem from different model inputs (prices, policy etc.), but also simply from the fact that the gap at the best solutions returned by the solver for each run differs.

MIP solvers can also "tune" their options based on one or several given model instance. Tuning is available both with CPLEX and GUROBI, and can be switched on via the interface. That process takes quite long, as the model is repeatedly solved with different solver options. The parameters from the tuning step are stored in an option and can be used by subsequent runs.

### Fractional investments of machinery

An option to reduce the number of binaries is to treat certain investment decisions as continuous. For machinery, the model allows to replace the binary variable *v\_buyMach* by a fractional replacement *v\_buyMachFlex*. The replacement depends on a threshold for the depreciation costs per ha or hour, which can be set by the interface. The larger the threshold, the lower is the number of integer variables and the higher the (potential) difference to the solution where more indivisibilities in machine investments are taken into account.

The relevant code section (*define\_starting\_bounds.gms*) is shown below:

$ifi "%dynamics%" == "comparative-static" $setglobal buyMachFlexThreshold 1E+6  
$ifi not setglobal buyMachFlexThreshold $setglobal buyMachFlexThreshold 3  
  
  
 v\_buyMach.fx(machType,t,nCur) $ (t\_n(t,nCur)  
 $ ( (p\_machAttr(machType,"depCost\_ha") le %buyMachFlexThreshold%) $ p\_machAttr(machType,"depCost\_ha")  
 or (p\_machAttr(machType,"depCost\_hour") le %buyMachFlexThreshold%) $ p\_machAttr(machType,"depCost\_hour")  
 ) $ (not p\_machAttr(machType,"years"))) = 0;

v\_buyMachFlex.fx(machType,t,nCur) $ (t\_n(t,nCur)  
 $ ( (p\_machAttr(machType,"depCost\_ha") gt %buyMachFlexThreshold%)  
 or (p\_machAttr(machType,"depCost\_hour") gt %buyMachFlexThreshold%)  
 or p\_machAttr(machType,"years"))) = 0;

### Heuristic reduction of binaries

On demand, the RMIP solution can be used in combination with some heuristic rules to reduce the set of endogenous variables. As the RMIP solution will e.g. build a fraction of larger stables and thus save costs compared to the MIP solution, the herd size in the MIP solution can be assumed to be upper bounded by the solution of the MIP. Similarly, as investment costs for machinery will be underestimated by the MIP, it can be assumed that machinery not bought in the RMIP solution will not be found in the optimal solution of the MIP.

An example is shown below for investment decision into stables. The program first defines the maximal amount of stable places used in any year. Investments into stables and their usage which are larger than the maximal size or smaller than 2/3 of the maximal size are removed from the MIP. Equally, investment in stables is set to zero if there was no investment in the RMIP solution.

\* --- exclude bigger stable investment under binary conditions  
\*  
 p\_maxBoughtStableSize(stableTypes,leaves)  
 = smax(t\_n(tCur,nCur) $ sameScen(leaves,nCur),  
 sum( (stables1) $ p\_stableSize(stables1,stableTypes),  
 v\_buyStables.l(stables1,"long",tCur,nCur)\*p\_stableSize(stables1,stableTypes)))  
 $ (smax(t\_n(tCur,nCur) $ sameScen(leaves,nCur),  
 sum( (stables1) $ p\_stableSize(stables1,stableTypes),  
 v\_buyStables.l(stables1,"long",tCur,nCur)\*p\_stableSize(stables1,stableTypes))) ne -INF);  
  
  
  
 p\_maxInvStableSize(stableTypes,leaves)  
 = max(smax(t\_n(tCur,nCur) $ sameScen(leaves,nCur),  
 sum(stables1 $ p\_stableSize(stables1,stableTypes),  
 v\_stableUsed.l(stables1,tCur,nCur)\*p\_stableSize(stables1,stableTypes)))  
 $ (smax(t\_n(tCur,nCur) $ sameScen(leaves,nCur),  
 sum(stables1 $ p\_stableSize(stables1,stableTypes),  
 v\_stableUsed.l(stables1,tCur,nCur)\*p\_stableSize(stables1,stableTypes))) ne -INF),  
  
 smax(tOld,  
 sum(stables1 $ p\_stableSize(stables1,stableTypes),  
 p\_iniStables(stables1,"long",tOld)\*p\_stableSize(stables1,stableTypes)))  
 $ (smax(tOld,  
 sum(stables1 $ p\_stableSize(stables1,stableTypes),  
 p\_iniStables(stables1,"long",tOld)\*p\_stableSize(stables1,stableTypes))) ne -INF));  
  
 p\_maxBoughtStableSize(stableTypes,leaves)  
 = smin(stables1 $ (p\_stableSize(stables1,stableTypes) ge p\_maxBoughtStableSize(stableTypes,leaves)),  
 p\_stableSize(stables1,stableTypes));  
  
 p\_maxBoughtStableSize(stableTypes,leaves) $ (p\_maxBoughtStableSize(stableTypes,leaves) eq INF)  
 = smax(stables1 $ p\_stableSize(stables1,stableTypes),p\_stableSize(stables1,stableTypes));  
  
  
 p\_maxInvStableSize(stableTypes,leaves)  
 = smin(stables1 $ (p\_stableSize(stables1,stableTypes) ge p\_maxInvStableSize(stableTypes,leaves)),  
 p\_stableSize(stables1,stableTypes));  
  
 p\_maxInvStableSize(stableTypes,leaves) $ (p\_maxInvStableSize(stableTypes,leaves) eq INF)  
 = smax(stables1 $ p\_stableSize(stables1,stableTypes),p\_stableSize(stables1,stableTypes));  
  
 v\_stableInv.up(stables,hor,t,nCur)  
 $ (t\_n(t,nCur) $ sum( (stableTypes,sameScen(nCur,leaves)) $ ( (p\_stableSize(stables,stableTypes) le p\_maxInvStableSize(stableTypes,leaves))  
 and (p\_stableSize(stables,stableTypes) ge p\_maxBoughtStableSize(stableTypes,leaves)\*6/10)  
 $ p\_stableSize(stables,stableTypes)),1 )) = 1;  
  
 v\_stableInv.lo(stables,hor,t,nCur)  
 $ (t\_n(t,nCur) $ sum( (stableTypes,sameScen(nCur,leaves)) $ ( (p\_stableSize(stables,stableTypes) le p\_maxInvStableSize(stableTypes,leaves))  
 and (p\_stableSize(stables,stableTypes) ge p\_maxBoughtStableSize(stableTypes,leaves)\*6/10)  
 $ p\_stableSize(stables,stableTypes)),1 )) = 0;  
  
 v\_stableInv.up(stables,hor,t,nCur)  
 $ ( sum( (stableTypes,sameScen(nCur,leaves)) $ ((p\_stableSize(stables,stableTypes) gt p\_maxInvStableSize(stableTypes,leaves))  
 $ p\_stableSize(stables,stableTypes)),1 ) ) = 0;  
  
 v\_buyStables.up(stables,hor,t,nCur)  
 $ (sum((stables1,stableTypes) $ (p\_stableSize(stables,stableTypes) $ p\_stableSize(stables1,stableTypes)), v\_buystables.l(stables1,hor,t,nCur))  
 $ sum( (stableTypes,sameScen(nCur,leaves)) $ ((p\_stableSize(stables,stableTypes) lt p\_maxInvStableSize(stableTypes,leaves))  
 $ (p\_stableSize(stables,stableTypes) ge p\_maxBoughtStableSize(stableTypes,leaves)\*6/10)  
 $ p\_stableSize(stables,stableTypes)),1 ) $ t\_n(t,nCur) ) = 1;  
  
  
 v\_stableUsed.up(stables,t,nCur)  
 $ ( sum((stableTypes,sameScen(nCur,leaves)) $ ((p\_stableSize(stables,stableTypes) gt p\_maxInvStableSize(stableTypes,leaves))  
 $ p\_stableSize(stables,stableTypes)),1 ) $ t\_n(t,nCur)) = 0;  
  
 v\_buyStables.up(stables,hor,t,nCur)  
 $ ( sum((stableTypes,sameScen(nCur,leaves)) $ ((p\_stableSize(stables,stableTypes) gt p\_maxInvStableSize(stableTypes,leaves))  
 $ p\_stableSize(stables,stableTypes)),1 ) $ t\_n(t,nCur)) = 0;  
  
 v\_stableInv.up(stables,hor,t\_n(t,nCur))  
 $ ( sum((stableTypes,sameScen(nCur,leaves)) $ ((p\_stableSize(stables,stableTypes) lt p\_maxBoughtStableSize(stableTypes,leaves)\*5/10)  
 $ p\_maxBoughtStableSize(stableTypes,leaves)  
 $ p\_stableSize(stables,stableTypes)),1 ) ) = 0;  
  
 v\_buyStables.up(stables,hor,t\_n(t,nCur))  
 $ sum((stableTypes,sameScen(nCur,leaves)) $ ((p\_stableSize(stables,stableTypes) lt p\_maxBoughtStableSize(stableTypes,leaves)\*5/10)  
 $ p\_maxBoughtStableSize(stableTypes,leaves)  
 $ p\_stableSize(stables,stableTypes)),1 ) = 0;  
  
 v\_buyStables.up(stables,hor,t,nCur)  
 $ ( sum((stableTypes,sameScen(nCur,leaves)) $ ((p\_stableSize(stables,stableTypes) lt p\_maxInvStableSize(stableTypes,leaves)\*5/10)  
 $ p\_stableSize(stables,stableTypes)),1 ) $ t\_n(t,nCur)) = 0;  
  
\*  
\* --- do not buy stables in MIP mode if never bought under RMIP

Similar statements are available for investments into manure silos, buildings and machinery. These heuristics are defined in "*model\reduce\_vars\_for\_mip.gms*". It is generally recommended to use these statements as they can considerably reduce solving time. However, especially after structural changes to the code, checks should be done if the rules do not actually prevent the model from finding the (optimal) MIP solution.

### Binary fixing heuristics

In order to speed up solution, the heuristics discussed above are coupled with repeated RMIP solves where integer variable from the last fractional solution are moved to zero or unity depending on the solution and heuristics rules. To give an example: if parts of machinery are bought over time such that their sum exceeds a threshold, for instance half a tractor, the *machBuy* variable in the first year where the machine is bought is fixed to zero. These pre-solves can lead to start point for the MIP solves where most integer variables are already no longer fractional which can speed up solution.

### Equations which support the MIP solution process

Another tactic to ease the solution of MIPs is to define equations, which decrease the solution space for the integer variables based on the level of fractional variables respectively defining logical ordering for the integer decisions. These equations are not necessarily truly restricting the solution space, they only reinforce existing relations between variables. The additional equations often reduce the overall solution time by improving the branching more than by increasing single LP iterations due to the increase in the constraints.

One way to improve the branching order is to link binaries with regard to dynamics. There are currently *three ordering equations over time*. The first two prescribes respectively that if a farm has a cow herd in t+1 this implies that a cow herd in the previous year existed:

hasHerdOrderDairy\_(tCur(t),nCur) $ (tCur(t-1) $ t\_n(t,nCur)) ..  
  
 v\_HasBranch("dairy",t,nCur) =L= sum(t\_n(t-1,nCur1) $ anc(nCur,nCur1), v\_hasBranch("dairy",t-1,nCur1));

hasFarmOrder\_(tCur(t),nCur) $ (tCur(t-1) $ t\_n(t,nCur)) ..  
  
 v\_hasFarm(t,nCur) =L= sum(t\_n(t-1,nCur1) $ anc(nCur,nCur1), v\_hasFarm(t-1,nCur1));

The third one implies that working off-farm in a year t implies also working off-farm afterwards:

workOrder\_(tCur(t),nCur) $ ((sum(workOpps(workType) $ (v\_labOff.up(t,nCur,workType) ne 0),1) $ tCur(t-1)) $ t\_n(t,nCur)) ..  
  
 v\_labOffB(t,nCur) =G= sum(t\_n(t-1,nCur1) $ anc(nCur,nCur1), v\_labOffB(t-1,nCur1));

Another tactic followed is to define logical high level binaries which dominate other. These *general binaries* are partly already shown above: the *v\_hasFarm* and *v\_workOffB* variables. The later one is linked to the individual off-farm working possibilities:

convLab\_(tCur(t),nCur) $ (sum(workOpps(workType)$ (v\_labOff.up(t,nCur,workType) ne 0),1) $ t\_n(t,nCur) ) ..  
  
 sum(workOpps(workType), v\_labOff(t,nCur,workType)) =E= v\_labOffB(t,nCur);

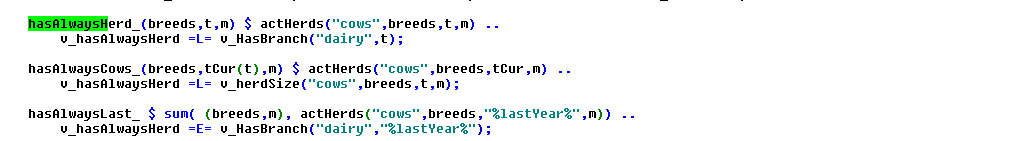
In order to support the solving process, *w\_workOff* is defined as a *SOS1* variable, which implies that at most one of the *workType* options is greater than zero in any year.

The *v\_hasFarm* variables dominates the *v\_hasBranch* variables:

hasFarmOrder\_(tCur(t),nCur) $ (tCur(t-1) $ t\_n(t,nCur)) ..  
  
 v\_hasFarm(t,nCur) =L= sum(t\_n(t-1,nCur1) $ anc(nCur,nCur1), v\_hasFarm(t-1,nCur1));

That equation is additionally linked to the logic of the model as *v\_hasFarm* implies working hours for general farm management.

Furthermore, general binary exists which controls if a herd is present in any year, *v\_hasAlwaysHerd*. If it is switched on, it will imply a dairy herd in any year.This is based on the equation *hasAlwaysLast\_* together with the order equation *hasHerdOrder\_* shown below.

 -> gibt es so nicht mehr

The equations which support the MIP solution process by linking fractional variables to binary ones relate to investment decisions. Firstly, investments in machinery are only possible if there is matching machinery need:

machBuy\_(curMachines(machType),machLifeUnit,t,nCur)  
 $ ( (v\_machInv.up(machType,machLifeUnit,t,nCur) ne 0)  
 $ (v\_buyMach.up(machType,t,nCur) ne 0)  
 $ p\_lifeTimeM(machType,machLifeUnit) $ p\_priceMach(machType,t)  
 $ (not sameas(machLifeUnit,"years")) $ t\_n(t,nCur) ) ..  
 v\_buyMach(machType,t,nCur)  
  
 =L= (v\_machNeed(machType,machLifeUnit,t,nCur) $ tCur(t)  
\*  
\* --- minus operating hours of weighted average over normal planning period  
\* if beyond the normal planning period  
\*  
 + [sum( (t\_n(t1,nCur1)) $ ( (p\_year(t1) lt p\_year(t)) $ tCur(t1) $ isNodeBefore(nCur,nCur1)),  
 v\_machNeed(machType,machLifeUnit,t1,nCur1)  
 \* 1/(p\_year(t)+5 - p\_year(t1)) )  
 /sum( (t1) $ ( (p\_year(t1) lt p\_year(t)) $ tCur(t1)), 1/(p\_year(t)+5 - p\_year(t1)) )  
 ] $ ( (not tCur(t)) and p\_prolongCalc)  
  
 ) \* 10;

Secondly, two equations link the dairy herd to investment decisions into stables and manure storage silos:

stableBuy\_(stables,hor,tCur(t),nCur) $ ( (v\_buyStables.up(stables,hor,t,nCur) gt 0) $ t\_n(t,nCur)) ..  
  
 v\_buyStables(stables,hor,t,nCur) =L= sum(stableTypes\_to\_branches(stableTypes,branches)  
 $ p\_stableSize(stables,stableTypes), v\_HasBranch(branches,t,nCur));

stableInvb\_(stables,hor,tCur(t),nCur)  
 $ ( ( (sum( t\_n(t1,nCur1) $ (isNodeBefore(nCur,nCur1) or sameas(nCur,nCur1)),  
 v\_buyStables.up(stables,hor,t1,nCur1)) gt 0)  
 or (sum( tOld, p\_iniStables(stables,hor,tOld)))) $ t\_n(t,nCur) ) ..  
  
 v\_stableInv(stables,hor,t,nCur) =L= sum(stableTypes\_to\_branches(stableTypes,branches)  
 $ p\_stableSize(stables,stableTypes), v\_HasBranch(branches,t,nCur));

These supporting restrictions can be switched off from the model via the interface, to check if they unnecessarily restrict the solution domain of the solver. It is generally recommended to use them as they have proven to speed up the solution process.

### Priorities

Finally, there are options to help the MIP solver to decide which branches to explore first. The variable field .prior in GAMS allows setting priorities which are passed to the MIP solver; lower priorities are interpreted as having precedence. The file "model\def\_priors.gms" defines such priorities.

The model is instructed to branch first on the decision to have a herd in any year, next on having a farm and the individual branches:

v\_hasAlwaysHerd.prior = %priorOperator% (p\_priorMax\*20);

v\_hasFarm.prior(t,n) $ t\_n(t,n) = %priorOperator% (p\_priorMax\*6 + %timeWeight%);

v\_hasBranch.prior("dairy",t,n) $ t\_n(t,n) = %priorOperator% (p\_priorMax\*4 + %timeWeight%);

v\_hasBranch.prior("farm",t,n) $ t\_n(t,n) = %priorOperator% (p\_priorMax\*5 + %timeWeight%);

Generally, early years are given precedence:

$setglobal timeWeight (card(t)-ord(t)+1)/card(t) \* p\_priorMax \* 10

The *p\_priorMax* is the maximal priorities assigned to stables, which is defined by a heuristic rule: large stables are tried before smaller ones, cow stable before young cattle and calves stables, and finally long-term investment in the whole building done before maintenance investments:

parameter p\_priorStables(stables);  
  
 p\_priorStables(stables) $ sum(stableTypes $ p\_stableSize(stables,stableTypes), stableTypes.pos)  
 = sqr(1/sum(stableTypes $ p\_stableSize(stables,stableTypes), stableTypes.pos)\*10)  
  
 \* sqrt( sum(stableTypes $ p\_stableSize(stables,stableTypes),  
 p\_stableSize(stables,stableTypes))  
 / smax((stables1,stableTypes) $ p\_stableSize(stables,stableTypes),  
 p\_stableSize(stables1,stableTypes)));  
  
 p\_priorMax = smax(stables, p\_priorStables(stables)) \* card(hor);  
 p\_priorStables(stables) = p\_priorStables(stables)/p\_priorMax;  
 p\_priorMin = smin(stables, p\_priorStables(stables));  
 p\_priorMax = 1;

Off-farm work decisions currently receive a lower priority compared to investments into stables:

v\_buyStables.prior(stables,hor,t,n) $ t\_n(t,n) = %priorOperator% (p\_priorStables(stables)\*hor.pos + %timeWeight%);

v\_stableInv.prior(stables,hor,t,n) $ t\_n(t,n) = %priorOperator% [(p\_priorStables(stables)\*hor.pos + %timeWeight%)\*0.95];

v\_labOffB.prior(t,n) $ t\_n(t,n) = %priorOperator% (p\_priorMin \* 0.9 + %timeWeight%);

v\_labOff.prior(t,n,workType) $ t\_n(t,n) = %priorOperator% (p\_priorMin \* 0.8 + %timeWeight%);

For other investment decisions, the investment sum is used for priority ordering, e.g:

p\_rank(buildings) = p\_priceBuild(buildings,"%firstYear%") / ( p\_building(buildings,"lifeTime") + 15 $ (not p\_building(buildings,"lifeTime")));

The SOS1 variables should have all the same priorities. Therefore, no distinction is introduced for the *v\_workOff* and *v\_siCovComb* variables, with the exemption of the time dimension.

Generally, it is recommend using these priorities as they have proven to speed up the solution process.

## Reporting

As discussed in the following chapter, a GUI allows exploitation of model results, also comparing different model runs. That part requires that all results are stored in one multi-dimensional cube. Accordingly, after the model is solved, its variables are copied to a result parameter, as shown in the following example:

p\_res(%1,%2,"liquid","sum","",tCur) = sum(t\_n(tCur,nCur), p\_probn(nCur) \* v\_liquid.l(tCur,nCur));

p\_res(%1,%2,"liquid","sum","","mean") = sum(t\_n(tCur,nCur), p\_probn(nCur) \*v\_liquid.l(tCur,nCur))/p\_cardTCur;

## Systematic sensitivity analysis based on Design of Experiments

As discussed above, solution for one indicator and one GHG emission target might require between a few seconds to several minutes on a powerful multi-core machine. The derivation of the marginal abatement cost curves requires solving repeatedly model instances over a range of GHG emission targets, therefore it might require an hour or more to solve one specific farm configuration.

An application of the model to a larger sample of existing farms is consequently computationally impossible. This is why it was envisaged from the beginning to use sensitivity analysis to generate a sufficient number of instances to derive a meta-model in order to estimate abatement costs for larger population of farms, for example based on an appropriate regression model.. Meta modeling seems also a suitable tool to learn more about which farm attribute impact abatement costs and to which extend the occurring Marginal Abatement Costs (MACs) depend on the GHG calculation procedure of the different indicators.

For this four steps are required:

1. Setting up of appropriate sensitivity experiments which cover the distribution of farm attributes in an appropriate sample (such as the farm structure survey for North-Rhine-Westphalia). Consequently, this requires the use of an efficient and space filling random sampling design to lower the necessary sample size for the derivation of a meta-model. At the same time, it has to be ensured that the randomised factor level combinations are smoothly distributed over the range of factor level permutations. [[6]](#footnote-158)
2. Running the single farm model on these experiments and collecting key results.
3. Deriving a meta model from these experiments.

This section focuses mainly on technical aspects of this process.

The overall strategy consists of combining a Java based package for interface generation and result exploitation, which also comprises a machine learning package, with GAMS code. For the definition of representative sensitivity experiments a sampling routine, (lhs\_0.10) implemented in R (version 2.15.1) is combined with the GAMS code to generate sample farms under recognition of correlations between factors.

The GAMS code *(scen\_gen*) modifies settings entered via the interface (see next section) to define attributes for each experiment. A single farm run is then executed as a child process with these settings. The user is able to define upper and lower bounds for single factors to define a solution room in which factor levels can vary between scenarios for different production specific attributes of the farm (see next section). The interface also allows defining if correlations between selected variables should be recognized during the sample randomization procedure. Furthermore, depending on the number of draws and the complexity of the assumed correlation matrix a maximum number of sampling repetitions can be selected [[7]](#footnote-159).

Only the factors for which the selected maximum value differs from the minimum value are varied between model runs. Hence, the user is able to fix factor levels for single factors over all experiments by defining the minimum and maximum factor level. The upper and lower bounds of the variables define the solution space of possible factor level combinations of different factors. If the chosen minimum and maximum values are equal, the factor level of the specific attribute is holding constant during the scenario definitions. For the definition of wage rates and prices for concentrates the user is able to select constant differences to the full time wage rate or the concentrate type 1.

With increasing number of factors that can vary between scenarios and increasing possible factor levels per factor, the number of possible scenarios (factor level permutations) will increase exponentially (up to a few thousands). Hence, to create model outputs representative for all admissible scenarios, a large number of scenario runs would have to be processed to get reliable outputs for the derivation of a meta-model.

As this would cause long computing time also on a multi-core processor (several days), the numbers of scenario runs have to be restricted to a manageable number, while at the same time being representative for the real life distribution of farm attributes.

Therefore, the scenario definition is done by Latin Hypercube Sampling (LHS) to create an efficient sample with a small sample size (to lower computing time) while guaranteeing a space filling sample design over the full range of admissible scenarios (McKay et al. 1979, Iman and Conover 1980). This is done, using a bridge from GAMS to the statistical software R. Therefore the LHS package of R has to be installed for being able to create LHS samples for a defined number of draws *n* and factors *k* (in our case taking the command "*improvedLHS(n,k*)"). LHS sampling creates a sample matrix of size n\*k incorporating random values between 0 and 1, which are interpretable as percentages. These are drawn assuming an uniform distribution between 0 and 1. Further on, LHS sampling outputs ensure orthogonality of the output matrix and that factor level combinations evenly distributed over the possible permutation area.

The GAMS side of the technical implementation is shown in the following:

\* Use R to define the DOE  
\*  
\*------------------------------------------------------------------------------  
\*  
 file rIncFile / "%curDir%/rBridge/incFile.r" /;  
 put rIncFile;  
 $$setglobal outputFileD "%scrdir%/fromR"  
 $$setglobal inputFile "%scrdirR%/toR.gdx"  
  
 put ' plotFile <- "%resdirR%/scenGen/lhs\_%scenDes%.pdf"; '/;  
 put ' outputFile <- "%outputFile%"; '/;  
 put ' inputFile <- "%inputFile%"; '/;  
 put ' useCorr <- "%useCorr%"; '/;  
 put ' useColors <- "true"; '/;  
 put ' maxRunTime <- %maxRunTime%; '/;  
 putclose;

The maximal run time for finding a sample can be defined, *maxRunTime.* If correlations between variables are known and should be recognised within the sampling procedure, the command *useCorr* has to be set to *"true"*. Then the correlation matrix can be defined specifically.

\* --- set correlation matrix  
\*  
\* correlation coefficients are derived from data collections from AMI for prices and  
\* from LWK-NRW (Milchviehreport NRW, verschiedene Jahrg�nge, 2007 bis 2011)as well as  
\* a data collection of the LKV-NRW in 2012 for 5000 dairy farms in NRW. Correlation between  
\* nCows and CowsPerAK stem from the Forschungsdatenzentrum des Bundes und der L�nder after  
\* analysis on the "Landwirtschaftsz�hlung 2010", results were aligned with results derived  
\* from KTBL (2010,p.541).  
  
 table p\_cor1(\*,\*)  
  
 WinterCerePrice SummerCerePrice MaizCornPrice WinterRapePrice SummerBeansPrice SummerPeasPrice PotatoesPrice SugarBeetPrice  
 WinterCerePrice 0.8 0.7 0.5 0.5 0.5 0.5 0.5  
 SummerCerePrice 0.7 0.5 0.5 0.5 0.5 0.5  
 MaizCornPrice 0.5 0.5 0.5 0.5 0.5  
 WinterRapePrice 0.5 0.5 0.5 0.5  
 SummerBeansPrice 0.7 0.5 0.5  
 SummerPeasPrice 0.5 0.5  
 PotatoesPrice 0.5  
 SugarBeetPrice  
  
 ;

The names of the set of varying factors, the factor names, the scenario name, the desired number of draws and, if activated, also the correlation matrix are send to R. Then the R file "*rbridge\lhs.r*" is executed.

set factor\_name(\*,\*) / name.factors /;  
 set scen\_name(\*,\*) / name."%scendes%"/;  
  
 execute\_unload "%inputFile%" p\_n,factor\_name,scen\_name,factors,p\_cor;  
 $$setglobal rFile "%curDir%/rbridge/lhs.r"  
  
 $$if exist "%outputFileD%\_doe.gdx" execute "rm %outputFileD%\_doe.gdx"  
 $$batinclude 'util/title.gms' "'execute %rexe% %rFile%'";  
 $$if exist %rexe% execute "%rexe% %rFile% %curDir%/rBridge/incFile.r";  
  
$endif.onlyCollect  
  
\*  
\* --- read output from LHS sampling provided by R  
\*  
 parameter p\_doe(\*,\*);  
 execute\_load "%outputFile%\_doe" p\_doe;  
 if ( card(p\_doe) eq 0, abort "Error generating doe, no data found";);  
 display p\_doe;  
  
 parameter p\_testDoe "Check for mean of draws";  
 p\_testDoe(factors) = sum(draws, p\_doe(draws,factors))/card(draws);  
 display p\_testDoe;

The R-bridge is hence activated (R side). Therefore, several packages are installed in R from the R library to be able to do LHS sampling:

#install.packages("d:\r\R-2.15.1\library\mc2d\_0.1-13.zip",repos=NULL);  
#install.packages("d:\r\R-2.15.1\library\mvtnorm\_0.9-9992.zip",repos=NULL);  
#install.packages("d:\r\R-2.15.1\library\lhs\_0.10.zip",repos=NULL);  
#install.packages("d:\\temp\\gclus\_1.3.1.zip",repos=NULL);  
# install.packages("t:\\britz\\gdxrrw\_0.0-2.zip",repos=NULL);  
# install.packages("D:\\temp\\gdxrrw\_1.0.2.zip", repos = NULL, type="source");  
 library(lhs);  
 library(gdxrrw);  
 igdx("N:/soft/gams24.7new/24.7");  
 library(mc2d);  
 library(mvtnorm);  
 library(Matrix);  
 library(gclus);  
 library(reshape2);

*p\_n* denotes the number of draws defined via the GUI, which is equivalent to the number of scenarios resulting from the sampling routine. *Sys.getenv(....)* asks for commands or information given by the environment (for example if correlations have to be recognised or not).

#useCorr <- Sys.getenv("useCorr")  
#useColors <- Sys.getenv("useColors")  
#inputFile <- Sys.getenv("inputFile");  
#plotFile <- Sys.getenv("plotFile")  
#outputFile <- Sys.getenv("outputFile");  
#maxRunTime <- as.numeric(Sys.getenv("maxRunTime"))

We decided to use the "*improvedLHS"* type for randomisation [[8]](#footnote-160) which produces a sample matrix of *n* rows and *k* columns (n = number of draws, k = number of factors). This leads to a quite efficient sample generation in R:

out1 <- improvedLHS(n,k);

Usually, input variables for sensitivity analysis in computer models are assumed to be independent from each other (Iman et al., 1981a;b). Also LHS sampling was designed to create a sample of factor level combinations for different factors avoiding correlations between factors in random draws to ensure a space filling output. But, for our purposes, it is important to incorporate as much information about the multivariate input distribution as possible to get more realistic sample scenarios and exclude factor combinations that are rather impossible in reality. Hence, following Iman and Conover (1982:p.331-332) correlation structure information among input variables should be recognised within the sampling process, if available. Otherwise "the theoretical properties of the statistics formed from the output may no longer be valid." (Iman and Conover 1982:p.331)

To also incorporate information about dependencies between interesting variables during the sampling procedure we expanded the sampling method by an approach of Iman and Conover (1982) designing a swapping algorithm which shuffles observations for single factors between the draws to mimic given *k\*k* correlation matrix (therefore the R package *MC2d* including the routine *cornode* is necessary).

# --- load correlation matrix from GAMS  
#  
 t <- rgdx.param(inputFile,"p\_cor",names=c("f1","f2"),compress="true");  
 t  
 t<-acast(t, f1~f2, value.var="value")  
 t<-as.matrix(t);

To increase the possibility to randomise a sample which offers a correlation matrix of factors near the proposed one, the routine allows to repeat the random sampling of demanded *n* draws (yielding *n* experiment scenarios) for a maximal given computing time ("*maxRunTime*" e.g. 300 seconds.). The sample (incorporating *n* draws for *k* factors) with the smallest mean percentage deviation (*meanDev*) between given and randomised correlation matrix is then selected and send back to GAMS as the random sample representing the possible population. Alternatively, the repetition of *n* draws (*n* x *k* sampling matrix) will be stopped by a threshold value (*if meanDev < 1*) for the deviation between the assumed and the randomised sample correlation matrix.

bestFit <- 10;  
 iDraw <- 0;  
  
  
 while( runTime < maxRunTime ){  
  
 iDraw <- iDraw + 1;  
  
 if ( LHSType == "optimumLHS" ) {  
 out1 <- optimumLHS(n,k,2,0.01);  
  
 print("shit");  
  
 } else {  
 out1 <- improvedLHS(n,k);  
 }

# --- use cornode to apply Iman & conover 1982 to impose correlation  
# t on the LHS matrix out  
#  
 out1 <- cornode(out1,target=t)  
 c <- cor(out1);  
  
 fit = 0;  
 for ( i in 1:k )  
 for ( j in 1:k )  
 if ( fit < bestFit )  
  
 if ( fit < bestFit )  
 {  
 out <-out1;  
 bestFit <-fit;  
  
 meanDev = sqrt(fit/k)\*100;  
 };  
  
 if ( iDraw %% reportDraws == 0){  
 curTime <- as.numeric(Sys.time(),units="seconds");  
 runTime <- curTime - begTime;  
 print(paste(" draw :",iDraw," runTime ",round(runTime)," of ",maxRunTime,"seconds, mean sqrt of squared diff between given corr and best draw: ",round(meanDev,2),"%"));  
 }

For the case that the correlations between factors are given by the user, leading to an undefined correlation matrix, the program adjusts the correlation matrix to the nearest one possible:

# --- find nearest positive definite matrix  
#  
  
 t1<-nearPD(t);  
  
 t <- as.matrix(t1$mat);

As mentioned above the LHS sampling defines random value combinations between all factors in each single draw. Therefore, uniform distributed random values between 0 and 1 are drawn. The total set of draws defines one random sample of n single experiments (factor level combinations (in this stage of the sampling still between 0 and 1)). The routine implemented into the LHS-module now tries to find the best fitting sample which corresponds to the demanded correlation matrix most properly. Sampling outputs of the LHS draws show efficiency characteristics, also under recognition of correlations. This means that the mean of drawn random values is still 0.5 (as LHS draws lie between 0 and 1). And if the number of draws is large enough (greater than 30), quantiles as well as the mean of the distribution of LHS random values show that we are still consistent with the assumption of an uniform distribution function of the random draws (between 0 and 1), as necessary for efficient LHS outputs, also under recognition of factor correlations. The best fitting sample with the minimal average percentage deviation of correlations between defined and randomised correlation matrix is then selected and stored by the program and automatically printed as a PDF-document for visualisation. The PDF gives also information about average percentage bias of the randomised correlation matrix as well as the number of total draws which define the number of resulting sample experiments:

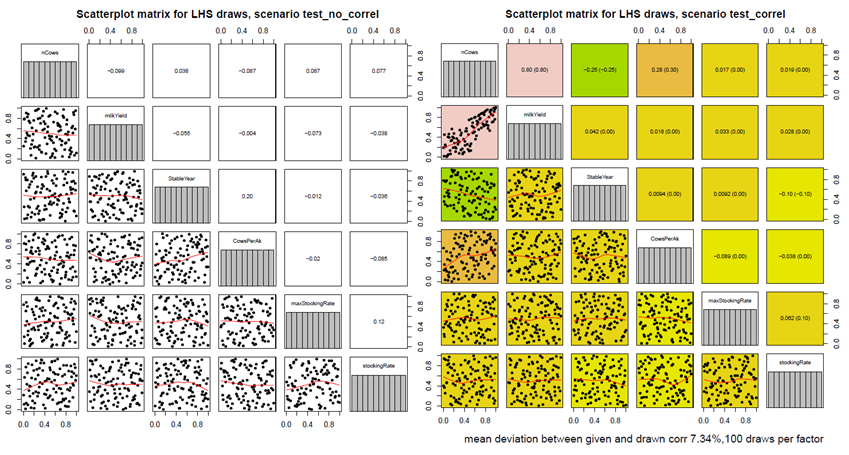


Figure 12: Scatter plot matrices for different LHS samples. With and without the recognition of factor correlations

On the left hand side one can see the scatter plot matrix without any correlations between factors. In contrast, a clear difference in sampled values is visualised by the right hand side matrix. For example, a correlation between *nCows* and *milkYield* was assumed to be 0.8. The best fitting matrix lead to the same correlation between these two factors. The correlation coefficients within brackets are the correlations predefined by the operator. The values in front of the brackets are the correlation coefficients fitted by the sampling matrix. The average mean percentage deviation of the randomised correlation matrix and the assumed correlation matrix is quantified by 7.34%, meaning, that on average, the randomised correlations deviate by 7.34% from the predefined ones. The distribution function in the diagonal shows, that the sampled values of each factor still ensure a uniform distribution.

The random values for the scenarios are transformed by GAMS to the real factor levels following the distribution functions of single variables. A uniform distribution of factor levels for the relevant variables is assumed. These are easily to define by the minimal value *a* and the maximum value *b*. A uniform distribution function can be defined by the following density function (left graph):

Values below *a*, or above *b* have a probability of 0. The antiderivative expresses the cumulative distribution function of the random variable whose values lie within the interval (right graph):

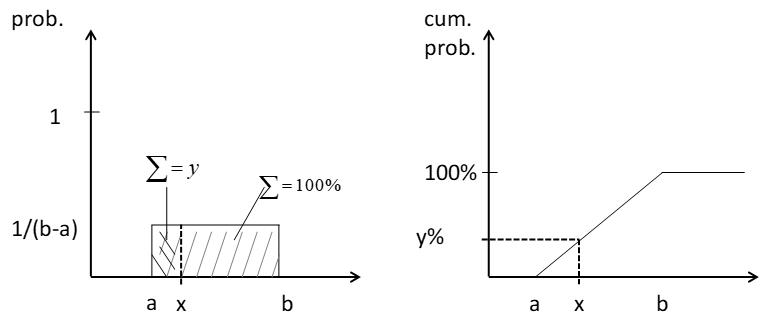


Figure 13: Density function and cumulative distribution function of an uniform distributed variable

From the left hand side density function one can easily derive the right hand side cumulative distribution function. The *y* value of the distribution function equals the integral below the density function (cumulative probabilities below x).

The given random values of the R-routine (*F(x)*) enable the allocation of corresponding factor levels (between *a* and *b*) to this random percentage values from the cumulative distribution function. A random cumulative probability value *y* corresponds to the factor level *x* which lies within the real value domain of the interesting factor. Hence this random sampling procedure produces random values by transforming uniform distributed random percentages (between 0 and 1) to factor levels, which are conform to the assumed distribution function of the variable. So far a uniform distribution function is assumed for the real factor levels (this can be adjusted to other functions if, for example, a known population has to be simulated).

For an assumed uniform distribution function of factor levels this is done following the formula:

The randomised value *y* is transformed to the factor level room concerning the given distribution function of the factor. Hence, for each single random draw a value is generated for the interesting variable corresponding to its assumed probability distribution. If a different distribution is assumed, formula 3 changes.

In GAMS code the formula (3) has to be applied for each factor to calculate the sample values whereat *p\_doe(draws,"factor")* is equivalent to F(x). The random percentage *p\_doe(\*,\*)* has to be multiplied by the difference between the possible maximum and minimum value of the factor *(%factorMax% - %factorMin%)*. Afterwards the min value *(%factorMin%)* has to be added to the product to yield the factor level *x* for the specific factor and scenario. This is illustrated for some parameters in the following.

\* --- result related declaration  
\*  
 PARAMETER p\_res(\*,\*,\*,\*,\*,\*)  
 p\_meta;  
 set resItems / mac,mean,cows,levl,margArab,margGras,margLand,herdRand,cropRand,ProfitDiff,manExportVol,profit/;  
  
  
 parameter p\_scenParam(draws,allFactors) "Numerical values for the scenario specific items";  
  
\*  
\* --- standard setting for aks  
\*  
 p\_scenParam(draws,"Aks") = %aks%;  
\*  
\* --- general mapping from DOE to factor ranges as defined on interface  
\*  
 p\_scenParam(draws,factors) = p\_doe(draws,factors) \* (p\_ranges(factors,"max")-p\_ranges(factors,"min"))+p\_ranges(factors,"min");  
  
$iftheni.obDist %useObsDistr% == true

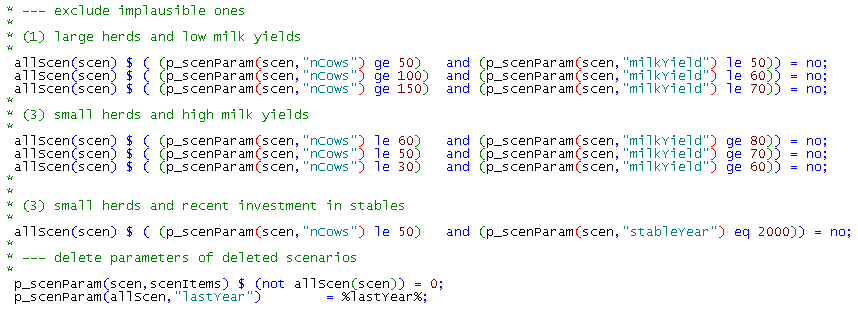
*p\_scenParam(draws,factor)* gives the scenario parameter one factor defined by the random values given by the LHS sampling routine. The combination of factor levels of the different factors for one single draw defines one single sensitivity scenario.

The set *scenItems* defines which settings are (possibly) defined specificly for each scenario:

file scenFile / "incgen/curScen.gms" /;

alias(scenItems,allFactors);

Nevertheless, correlations between factors are able to be recognised during the sample generation to avoid factor level combinations within scenarios that conflict with common statistical knowledge; the model code enables the user to specifically exclude factor level combinations that seem to be implausible. For example, high labour input per cow and low milk yield levels or high numbers of cows per farm and only very low yielding phenotypes.

 -> wird nicht mehr genutzt so [RW]

These scenario settings must be stored in a GAMS file which is then picked up by the child processes. In order to keep the system extendable, firstly, all settings inputted via the GUI are copied over to the specific scenario:

\* --- copy content of current scen file into new one  
\* via OS command  
\*  
 execute "cp %curDir%/incgen/expinc.gms %curDir%/incgen/curScen.gms"

Secondly, the modifications defining the specific sensitivity experiment, i.e. the scenario, are appended with GAMS file output commands (see *scenGen\gen\_inc\_file.gms*):

\* --- put statements will append to the new scen file  
\* and overwrite standard setting  
\*  
 put scenFile;

Finally, the content is copied to a specific scenario input file:

put\_utility batch 'shell' / "cp %curDir%/incgen/curScen.gms %curDir%/incgen/"scen.tl".gms";

The code to build and optimise the single farm model is realised in GAMS and uses CPLEX 12.6 in parallel mode as the MIP solver. Automatic tuning is used to let CPLEX use appropriate solver setting on the problem. The model instances are set up in order to avoid any conflicts with I/O operations to allow for parallel execution.

A single instance has a load of about 1.8 cores on average. In a multi-core machine it seems promising to execute several such processes in parallel. That is realised by a GAMS program which starts each model on its own set of input parameters:

\* --- execute exp\_starter as a seperate program, no wait, program will delete a flag at the end to signal that it is ready  
\*  
  
 put\_utility batch 'msglog' / '%GAMSPATH%/gams.exe %CURDIR%/exp\_starter.gms --scen='allScen.tl  
 ' --iScen='iLoop:0:0' -maxProcDir=255 -output='allScen.tl'.lst'  
 ' --seed=',uniform(0,1000):0:0,  
 ' -maxProcDir=255 -output='allScen.tl:0'.lst %gamsarg% lo=3'  
 ' --pgmName="'allScen.tl' (',iLoop:0:0,' of ',card(allScen):0:0,')"';

The name of the scenario, *allScen.tl* is passed as an argument to the process which will lead a specific include file comprises the definition of the scenario.

The GAMS process will use its own commando processors and run asynchronously to the GAMS thread which has started it. The calling mother process has to wait until all child processes have terminated. That is achieved by generating a child process specific flag file before starting the child process:

put\_utility batch 'shell' / ' %GAMSPATH%gbin/rm -f "../results/expFarms/res\_',scen.tl,'\_until\_' p\_scenParam(scen,"lastYear"):0:0,'.gdx"';  
 put\_utility batch 'shell' / ' %GAMSPATH%gbin/rm -f "%curdir%/incgen/'scen.tl'.gms"';  
 put\_utility batch 'shell' / ' %GAMSPATH%gbin/rm -f "%curdir%/'scen.tl'.lst"';

This flag file will be deleted by the child process when it finalises:

 -> finde ich so nicht [RW]

A simple DOS script waits until all flags are deleted:

set /a \_trys=0  
:again  
IF %\_Mode% EXIST %\_FlagFiles% (  
 set /a \_trys+=1  
 if %\_trys%.==%\_MaxTrys%. goto errorexit  
 sleep.exe %\_seconds%  
 goto again  
)

Using that set-up would spawn for each scenario a GAMS process which would then execute all in parallel. The mother process would wait until all child processes have deleted their flag files before collecting their results. As several dozen or even hundredth of scenarios might be executed, that might easily block the machine completely, e.g. by exceeding the available memory.

It is hence necessary to expand the set-up by a mechanism which ensures that only a pre-defined number of child processes is active in parallel. That is established by a second simple DOS script which waits until the number of flag files drops below a predefined threshold:

set /a \_trys=0  
:again  
  
set \_count=1  
  
for %%x in (%\_FlagFiles%) do set /a \_count+=1  
  
REM @echo %\_count% %\_nFiles% >> d:\temp\test.txt  
  
if %\_count% gtr %\_nFiles% (  
  
 set /a \_trys+=1  
  
 if %\_trys%.==%\_MaxTrys%. goto errorexit  
  
REM @echo %\_trys% %\_maxTrys% %\_seconds% >> d:\temp\test.txt  
  
  
 sleep.exe %\_seconds%  
  
 goto again  
  
)

Finally, the results from individual runs are collected and stored. A GAMS facility is used to define the name of a GDX file to read at run time:

put\_utilities batch 'gdxin' / ' ../results/expFarms/res\_',scen.tl,'\_until\_' p\_scenParam(scen,"lastYear"):0:0,'.gdx';

We now transformed all MAC estimates which are 0 due to an exit decision of a farm to be able to select these cases for our meta-modelling estimation (Heckman two-stage selection, described in the next technical documentation: "R routine to estimate Heckman two stage regression procedure on marginal abatement costs of dairy farms, based on large scale outputs of the model DAIRYDYN" by Britz and Lengers (2012)).

\* --- load the result  
\*  
 execute\_load p\_res;  
 p\_dummy = sleep(.01);  
\*  
\* --- filter out results of interest (so far only macs, avAcs and totACs)  
\*  
 $$ifi "%scentype%"=="MAC" $include 'scengen/scen\_load\_res\_mac.gms'  
  
 $$ifi "%scentype%"=="PROFITS" $include 'scengen/scen\_load\_res\_profits.gms'  
 $$ifi "%scentype%"=="Fertilizer directive" $include 'scengen/scen\_load\_res\_profits.gms'  
  
 );

Further on, the scenario specific settings which can be used as explanatory variables for later regressions are stored, see for example:

\* --- add scen variables to store explanatory vars  
\*  
 p\_meta(actInds,redLevl,scenItems,actInds1,scen)  
 $ sum(redlevl1, p\_res(actInds,redLevl1,"mac",actInds1,"mean")) = p\_scenParam(scen,scenItems);  
  
 p\_meta(actInds,redLevl,actInds,actInds1,scen)  
 $ sum(redlevl1, p\_res(actInds,redLevl1,"mac",actInds1,"mean")) = 1;  
  
 p\_meta(actInds,redLevl,"redLevl",actInds1,scen)  
 $ sum(redlevl1, p\_res(actInds,redLevl1,"mac",actInds1,"mean")) =  
 p\_res(actInds,redLevl,"redlevl",actInds1,"mean");

In a next step, the results are stored in a GDX container

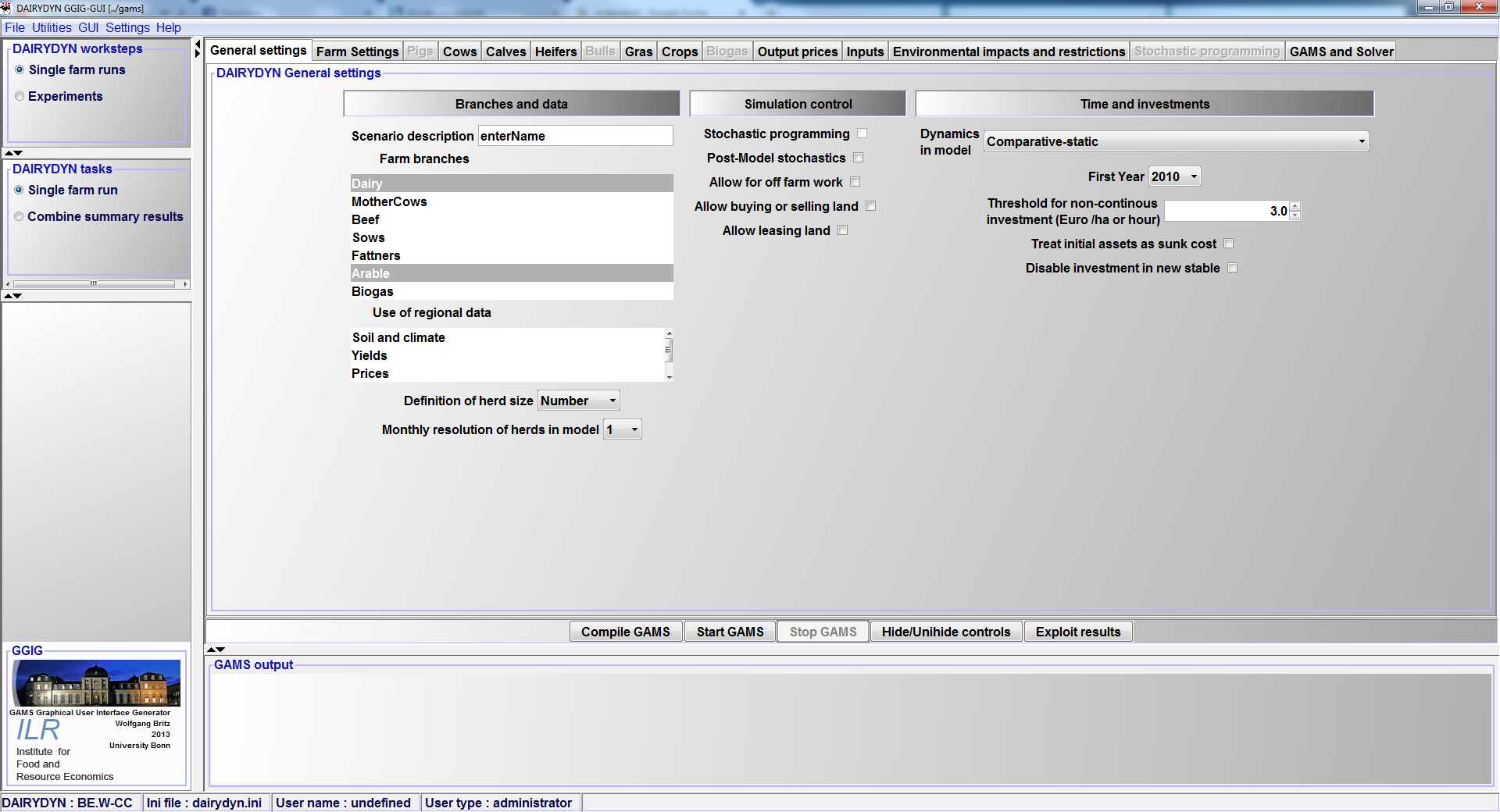
\* --- Store to disk  
\*  
 execute\_unload '../results/scenGen/meta\_%scenDes%.gdx' s\_meta,p\_meta=p\_res;

The major challenge consists in ensuring that the child processes do not execute write operation on shared files. In the given example, that relates to the GAMS listing, the GDX container with the results and the option files generated by the CPLEX tuning step. For the latter, two options are available: (1) set up child specific directory, copy the option files into it and use the *optdir* setting in GAMS, or (2) label the option files accordingly. That latter option was chosen which restricts the number of scenarios to 450:

\* --- opt3 file will by replaced by ###  
\* that allows for 300 parallel threads  
\*  
$iftheni.iScen not "%iScen%"==""  
  
$evalGlobal op3 round(%iScen%+100)  
$evalGlobal op4 round(%iScen%+400)  
$evalGlobal op5 round(%iScen%+700)  
  
  
$setglobal scenWithoutIncgen %scen%  
$set scen incgen/%scen%  
  
$else.iScen

In the case of normal single farm run, the standard option files will be used.

# The Graphical User Interface (GUI)



The Graphical User Interface (GUI) is based on GGIG (GAMS Graphical Interface Generator, Britz 2014). It serves two main purposes: to steer model runs and to exploit results from one or several runs. The creation of a visual user interface is also described as "visual debugging" (Grimm, 2002) to allow for an easy adjustment of parameters and quantitative and graphical examinations. With the help of only a few adjustments one can define single or multiple model farm runs for the interesting farm types with their specifications. Thereby, the former described coefficient generator helps to condense the necessary information for farm run definition by adjusting and calculating all production specific parameters to be consistent with the defined farm type (initial arable land, grass land, initial stables, initial manure storages, initial machine endowment...). After simulating the interesting experiments, the GUI enables the user to systematically analyse the simulated model variables and results.

A separate user handbook for the general use of the GUI is available at:

Britz, W. (2010), GGIG Graphical Interface Generator User Guide, Institute for Food and Resource Economics, University Bonn, 147 pages, <http://www.ilr.uni-bonn.de/agpo/staff/britz/GGIG_user_Guide.pdf>

## Model farm and scenario specifications

In the following, the different tabs of the GUI are shown and shortly described.

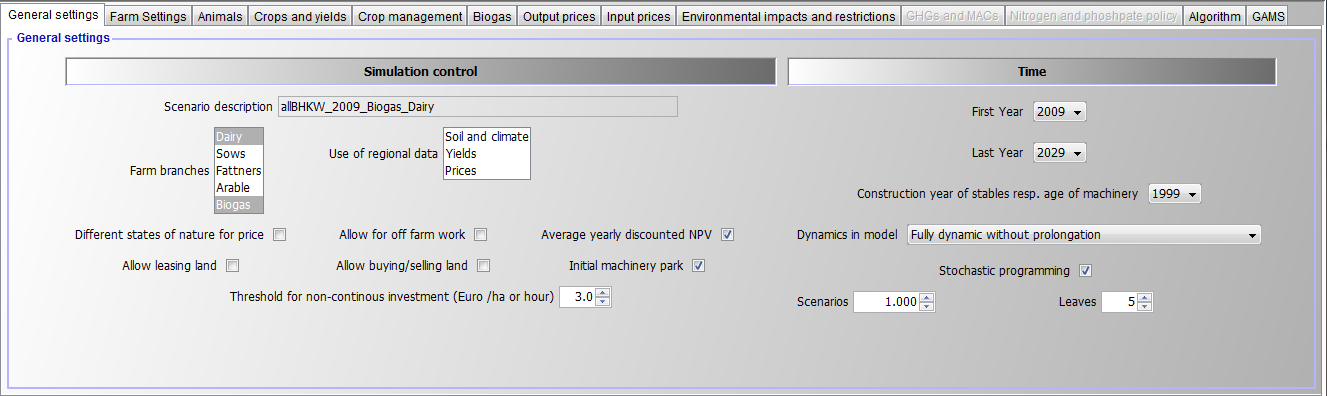
### Work step and task selection

In „Single farm runs" mode, all run specific settings (input and output prices, farm assets etc.) are set by the user in the interface. This is discussed in the following.

In the experiment mode the user instead defines ranges for selected settings which are varied based on stratified random sampling using Design of Experiments for a defined number of experiment. For each experiment, a single farm run is solved. These single farm runs are typically solved in parallel. After they are finalized, their results are combined into one result set.

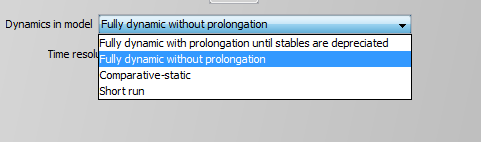
|  |  |
| --- | --- |
| Workstep and tasks: Single farm runs | Workstep and tasks: Experiments |
|  |  |

### General Settings

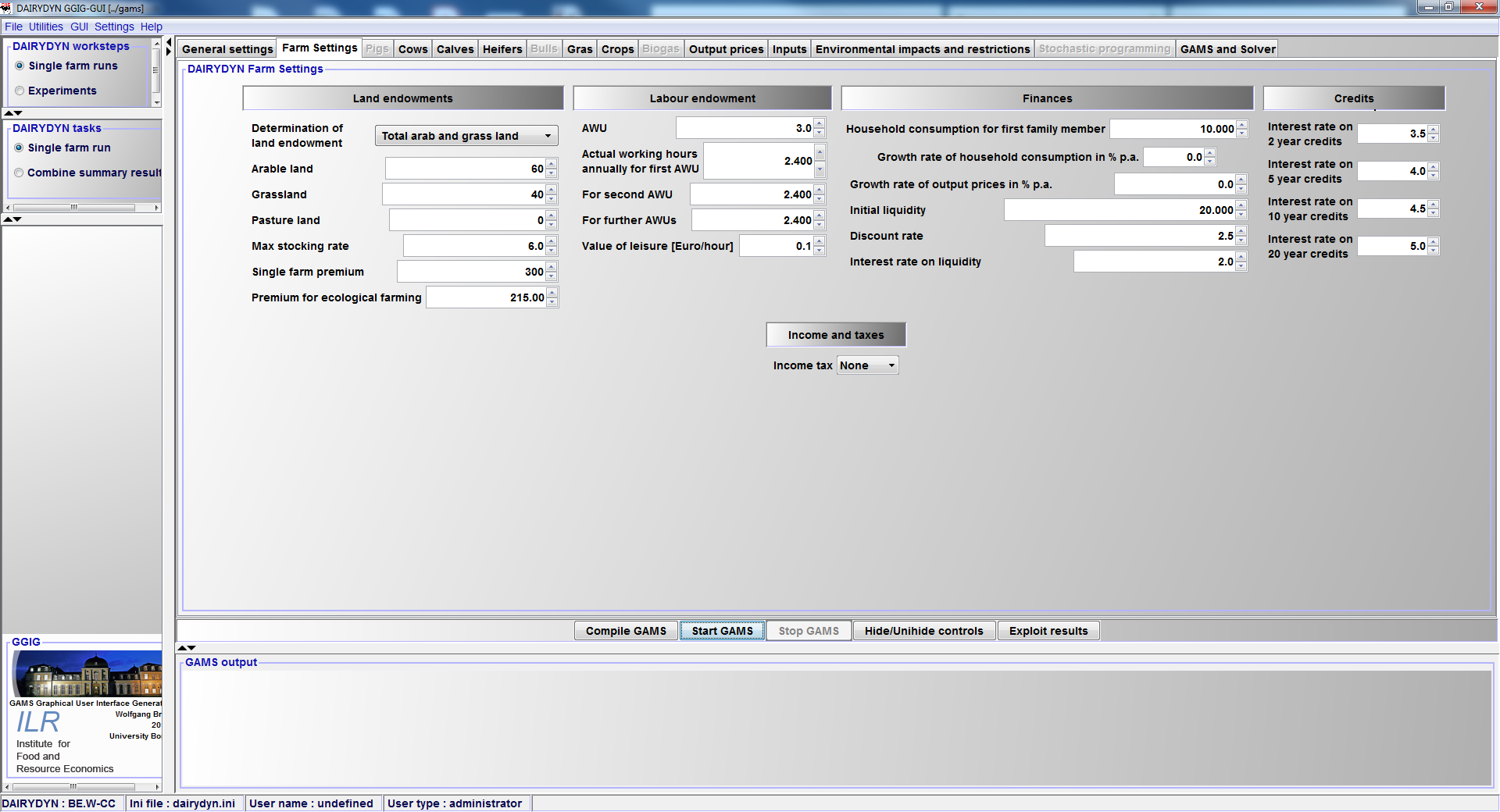


The General settings are sub divided into three components: i) Branches and data, where the user might name the scenario under which the results are stored, define the active farm branches, decide on the use of regional data and the timely resolution of the herd; ii) Simulation control, where the uses decides on the models stochastics, if off-farm work for the farm-employed family members is optional and if land is marketable; and time and iii) Time and investment, where the models dynamics, the simulation period, the construction year of the stables and age of machinery are chosen.

The model allows choosing between the following four mode to describe dynamics (or not):



### Farm Settings

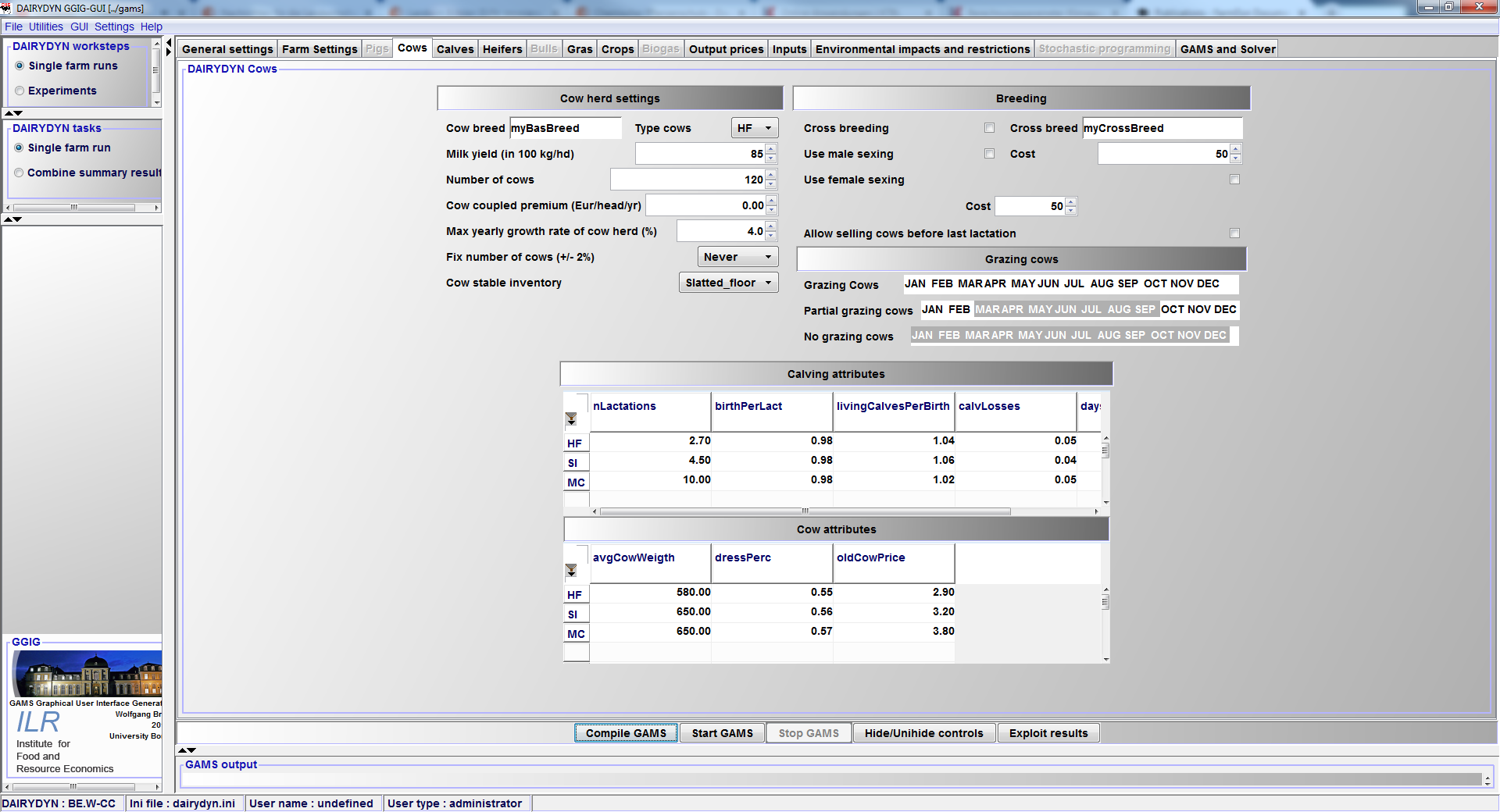


The farm settings panel carries general information about the farms land endowment, available premiums, labour endowment, the farms finances, the credit range and income taxes.

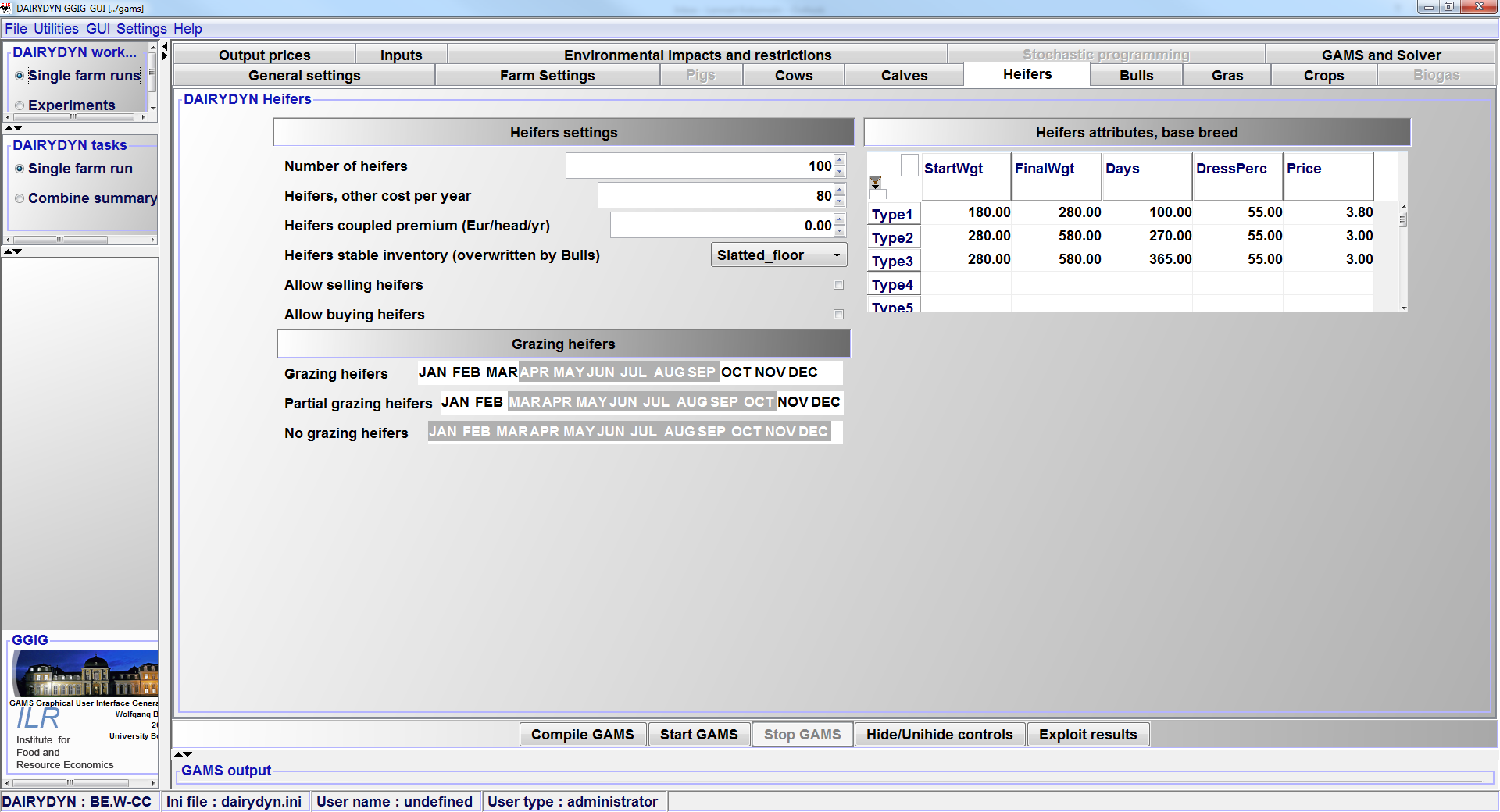
### Animals

Depending on the chosen farm branches the user has different tabs to further specify his simulated farm. For pigs and sows only one tab exists where the user can specify the initial herdsizes. For cattle several tabs and options exist. In general the initial herdsizes are used to determine the initial stable endowment and manure storage capacities on farm.

In the tab Cows the herd size, the milk yield, premiums, herd dynamics, manure management in stable, reproductive management and grazing period are to be specified by the user. The grazing period is distinguished into grazing, meaning animals are outside day and night, partial grazing, meaning animals are kept in stable over night and no grazing. Besides milk yield, further attributes of the specified cow breed are variable in the calving attributes and cow attributes tables.

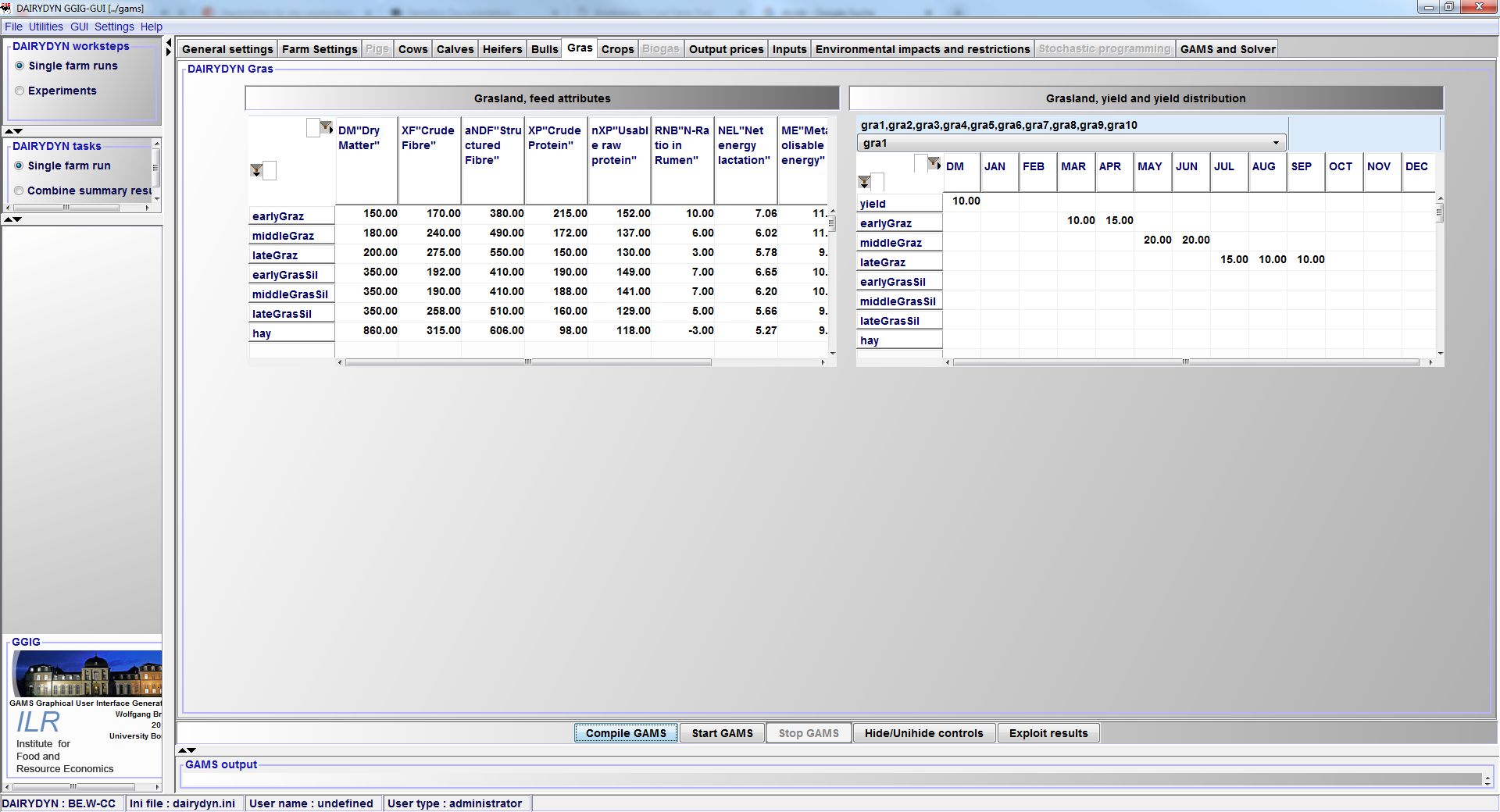


The other cattle related tabs, namely calves, heifers and bulls, allow for specification of the length of different growth periods and the respective start and end-weight, prices and the dress percentage of the potentially slaughtered animals. Similar to the cows, the grazing period and manure management system have to be specified. The tab Heifers is depicted below:

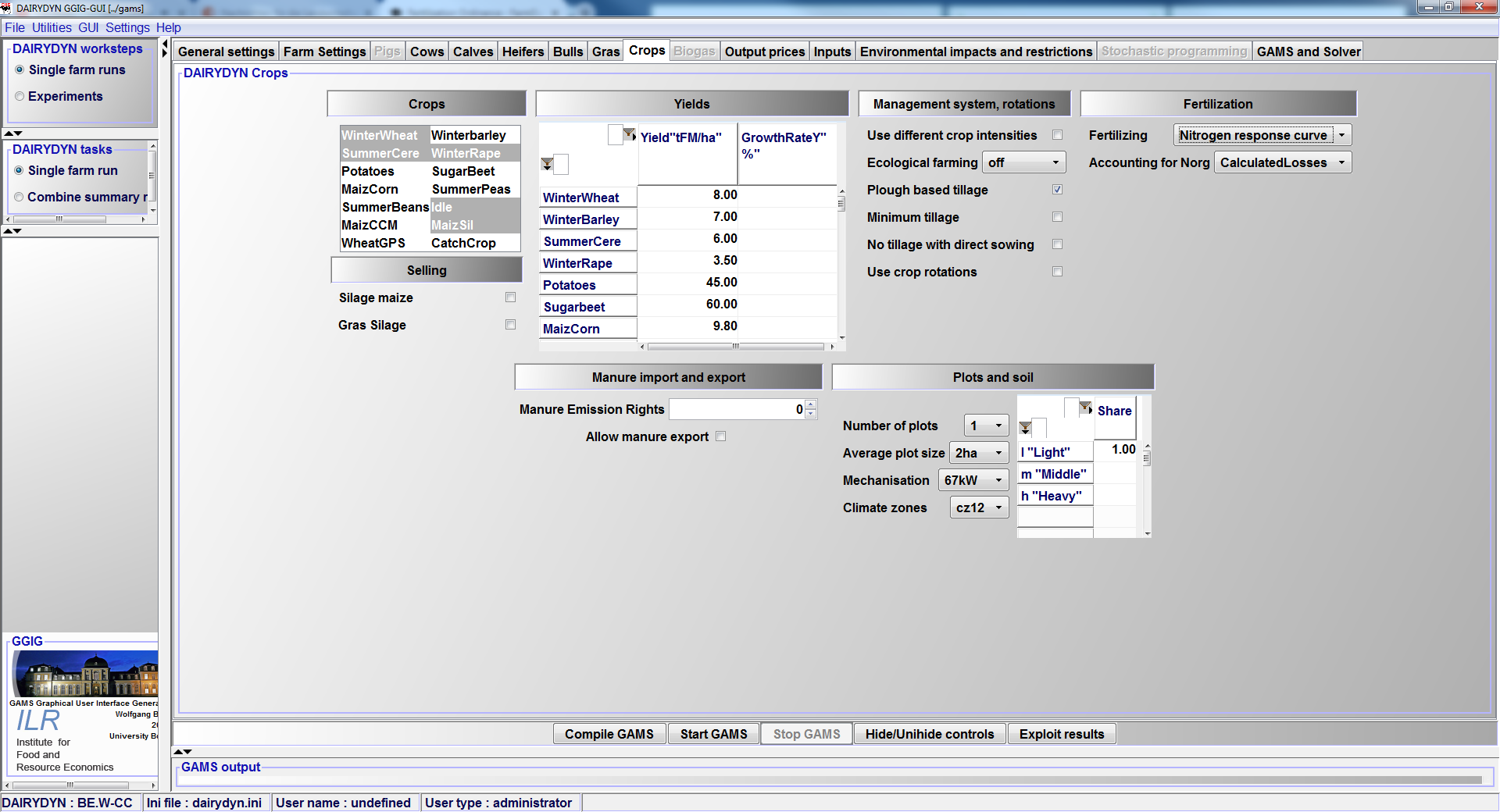


### Cropping

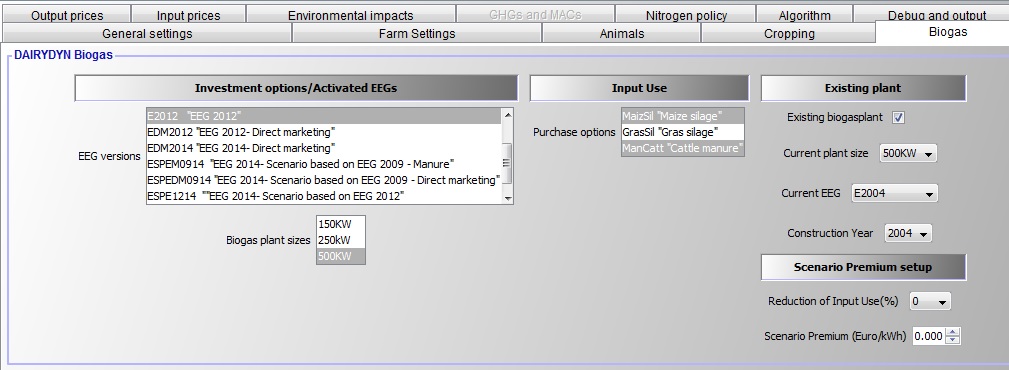
The tabs "Crops and Gras" comprise the selection of the crops and the yields and their growth rates. The tab Gras comprises two tables: the feed attributes table and the yield and yield distribution table. In the feed attribute table, the user might change the different ingredients of the harvested products. Products are silage gras, grazed gras and hay. The products are further divided by the time of harvest: early, middle and late. The yield table specifies the month of harvest of the respective products, the total yield in DM and the distribution of the yield over the harvest month. Using these information up to 10 different grassland usage options are derivable with differing management, yields and harvested products.



The tab crops lets the user choose the active arable crops, sell options for forage crops, yields, manure import and export, management and tillage system (ecological/conventional ploughing/direct tillage), the usage of crop rotations and the fertilization planning. Moreover, the average plot size, mechanisation level and the climate zone as well as the distribution of the soil type can be chosen.



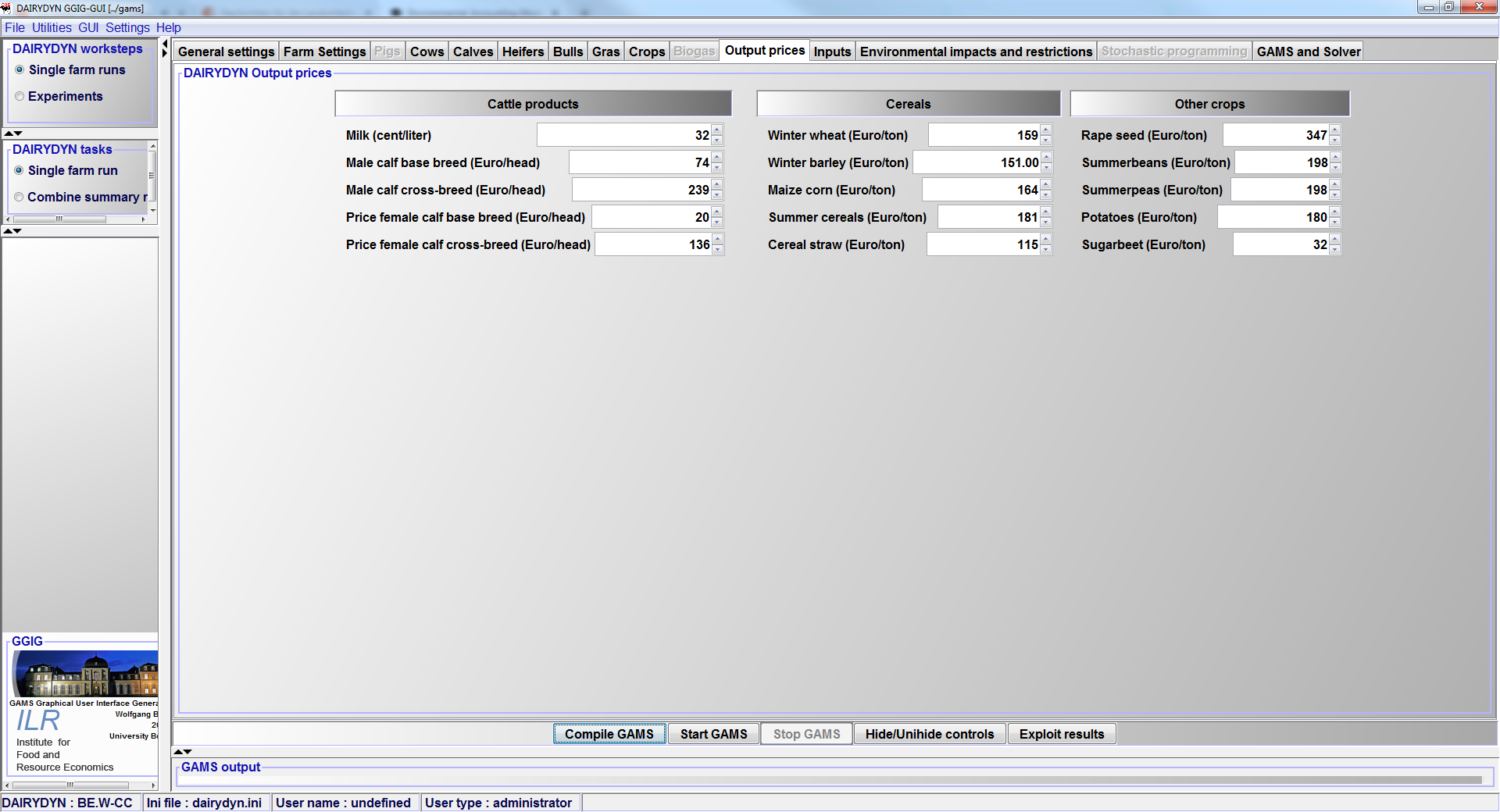
### Biogas



The biogas tab includes different Renewable Energy Act (EEGs) choices for the investment options as well as available biogas plant sizes. Moreover, it provides the option to select from potential inputs. Additionally, one can set up an existing biogas plant with the options to choose the size, the valid EEG and the construction year. However, in order to use this function the plant size and EEG has to be activated in the "Investment options" panel. Lastly, some options for scenario premiums are included.

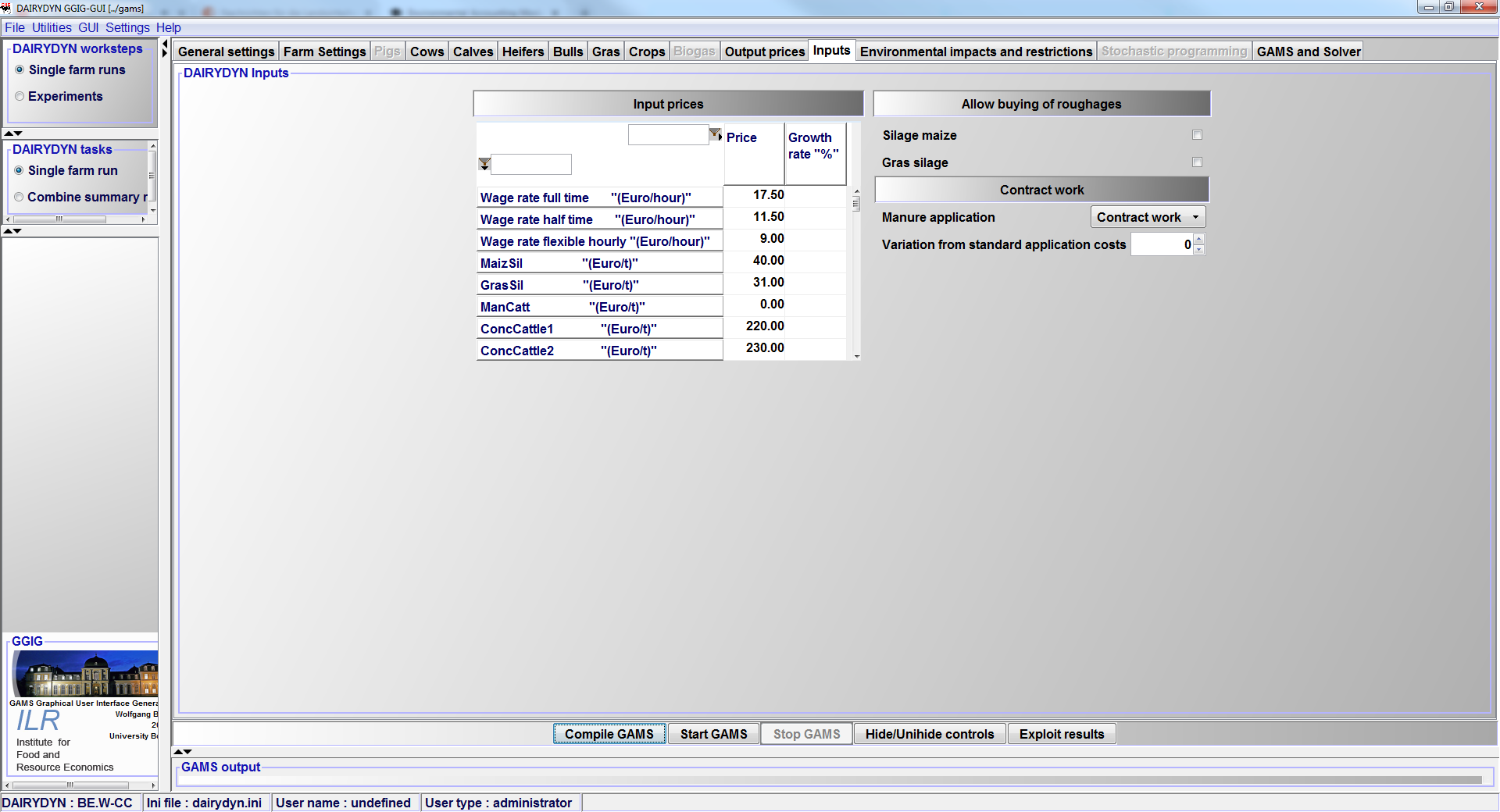
### Output Prices

That panel *Output prices* allows to set the price of the outputs present in the model.

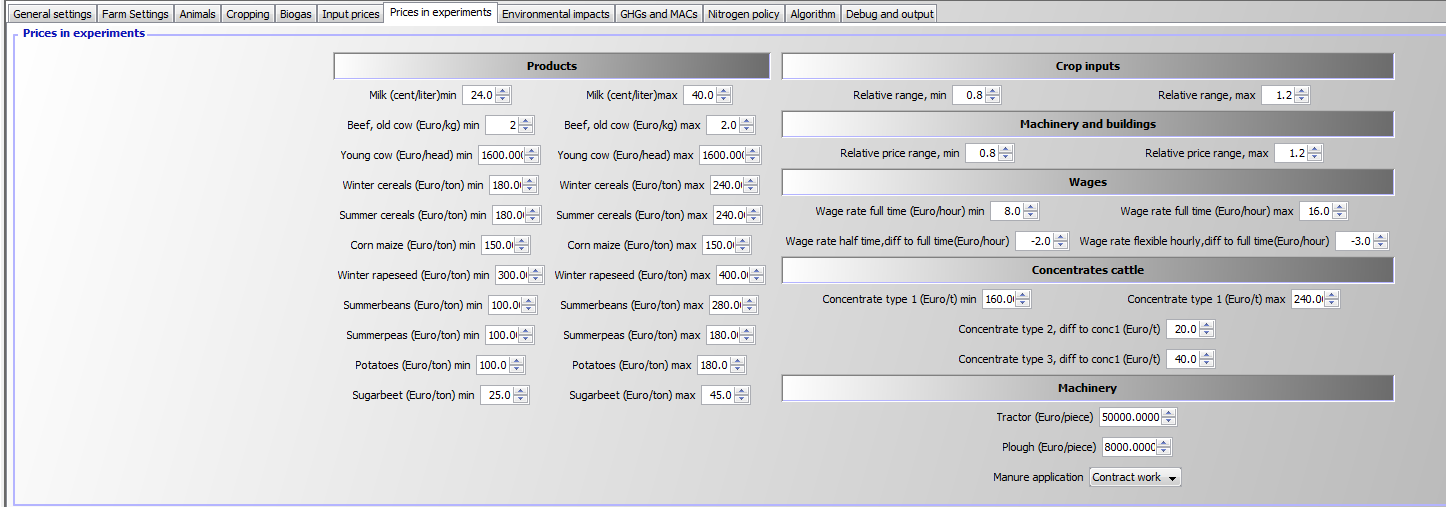


### Input Prices

That table *Input prices* allows to set the price of the outputs present in the model.

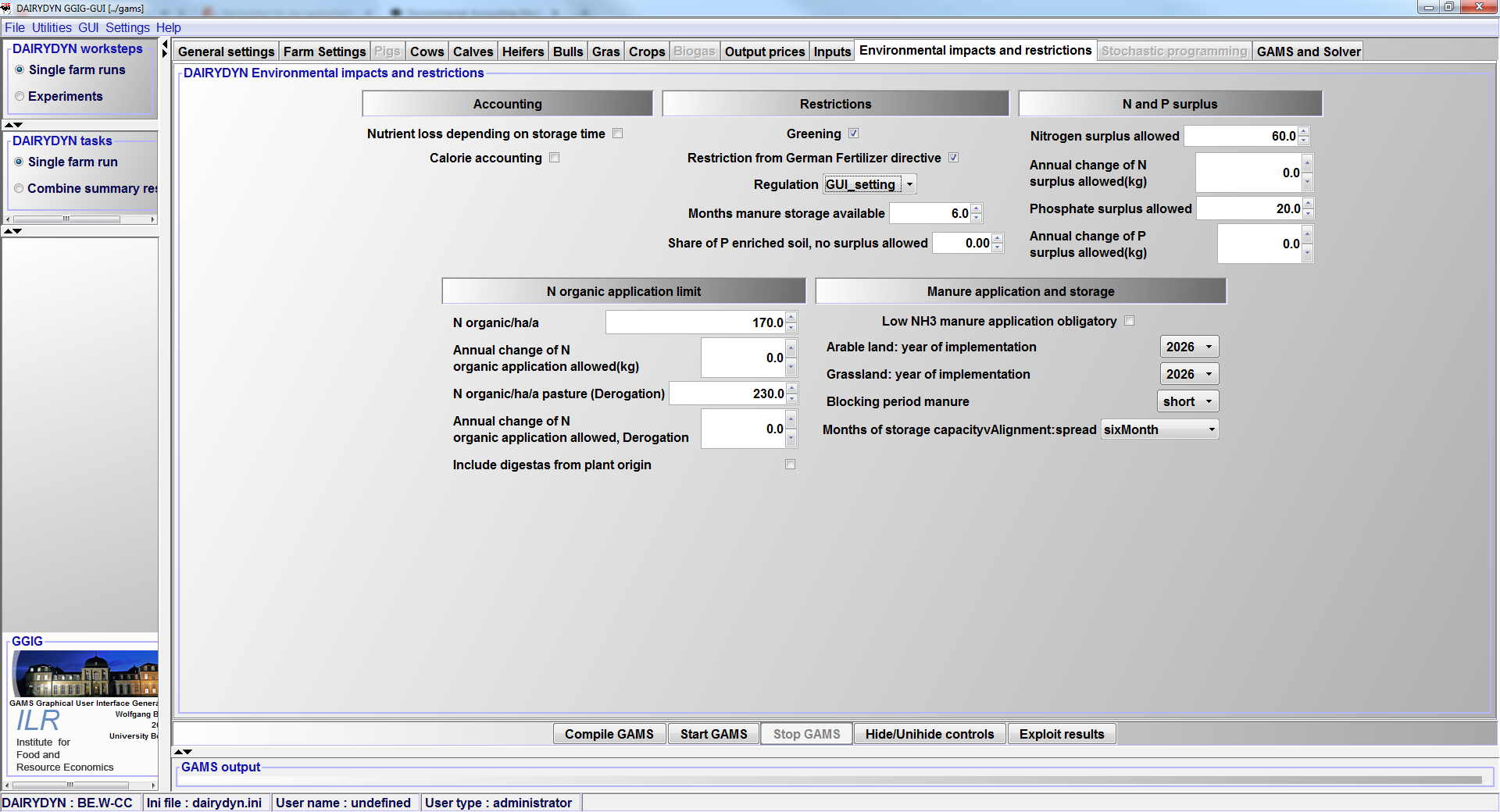


### Prices in Experiments



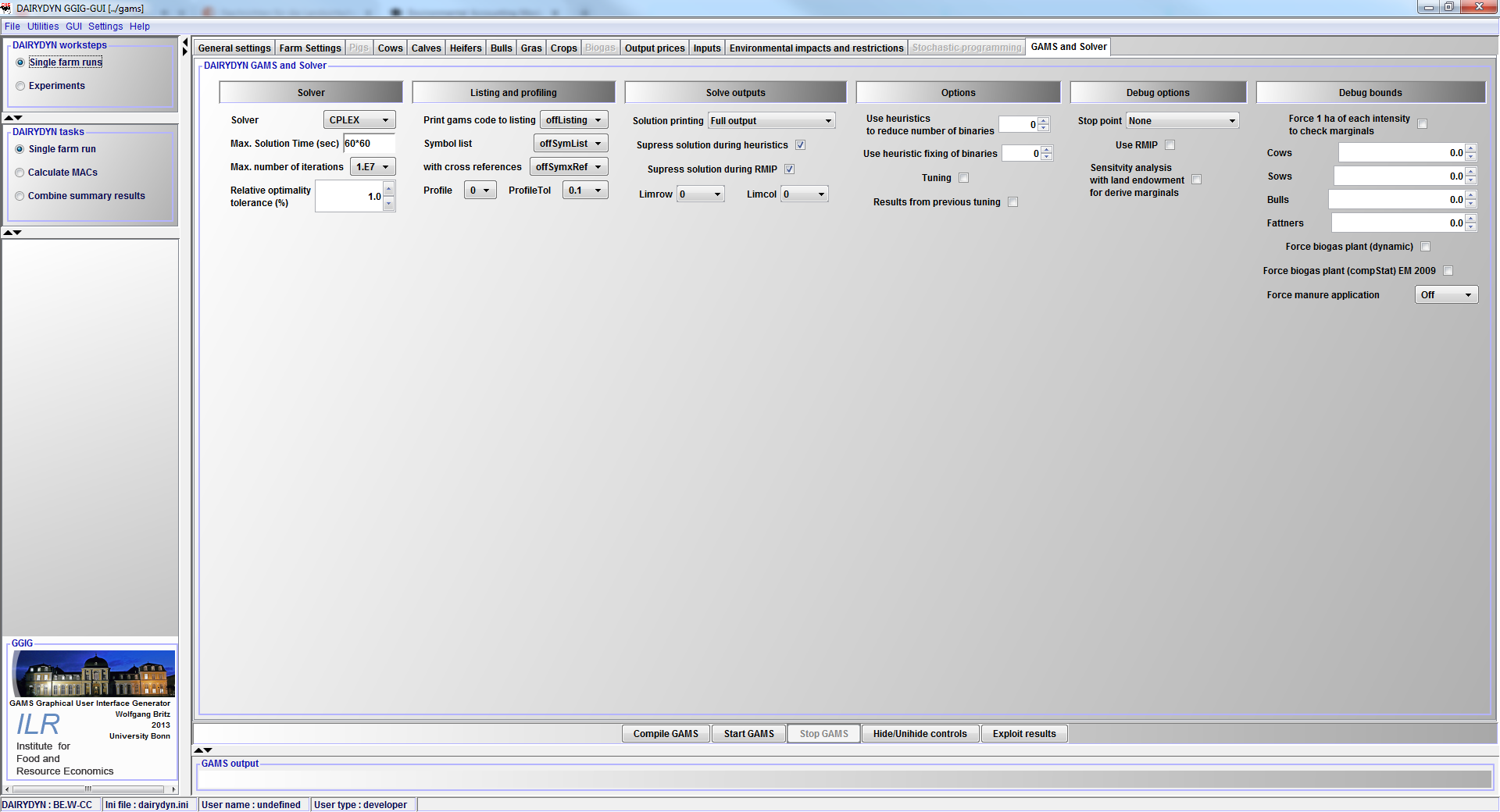
### Environmental Impacts and restrictions

Environmental impacts are always calculated, as described in the Environmental accounting module. At the moment of writing the farm can be restricted by the German FO and the Greening obligations. The fertiliser restrictions are further subdivided into the Ordinance of 2007 (FD\_2007), the latest revision of 2017 (FD\_2017) or the user is able to set limits and thresholds by her- or himself (GUI\_setting).



### GAMS and Solver

The last tab of the GUI which is shown here defines the chosen solver to optimise the fully dynamic MIP problem and further precision adjustments. Furthermore listing options for the resulting listing files and debug options are offered depending on the user type In general it is recommended to use CPLEX as the MIP solver, see section 7.2.



## Visualising and analysing results

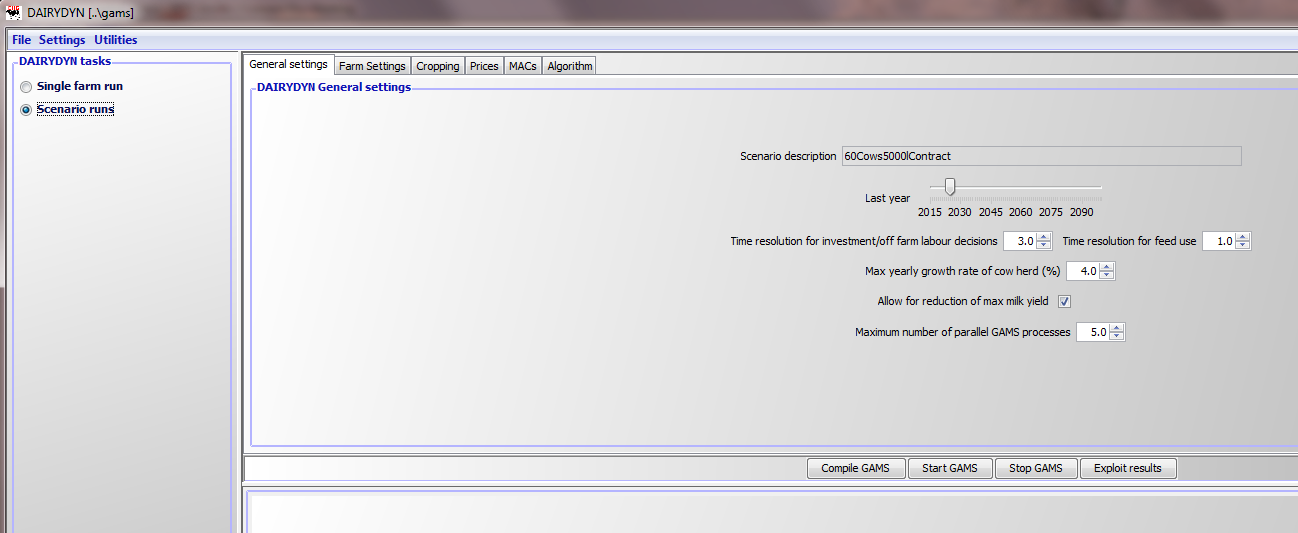
After a successful simulation (statement "normal completion" in the simulation window) the user can use the model interface to view results, plot them in tables or in graphs in order to make further analysis and interpretations. The analysis is based on a set of pre-defined reports, grouped by themes. Currently, the following reports are available:

* Model attributes
* Herd summary, mean
* Land use, mean
* Crops, mean
* Crops, intensities
* Crops, tillage
* Crop costs, mean
* Crop intensities, mean
* Tillage type, mean
* Cows, mean
* Cows, by yield
* Feeding overview, mean
* Production and related revenues, mean
* Feeding cows, mean
* Herd summary, time series
* Land use, time series
* Crops, time series
* Crops, tillage, time series
* Crops, intensities, time series
* Crop intensities, time series
* Tillage type, time series
* Cows, time series
* Cows, by yield, time series
* Feeding overview, time series
* Production and related revenues, time series
* Feeding cows, time series
* Stables overview, mean
* Stables, mean
* Machines, mean
* Stables overview, time series
* Stables, time series
* Machines, time series
* Overview work, mean
* Off-Farm, mean
* Overview work, time series
* Off-Farm, time series
* Manure, mean
* Manure, time series
* N Crops, total, mean
* P2L5 Crops, total, mean
* N total, mean
* P2O5 total, mean
* Storages, per month, mean
* Storages, total, mean
* N Crops, total, time series
* N Crops, per ha, mean
* N Crops, per ha, time series
* P2O5 crops, total, mean
* P2O5 Crops, per ha, time series
* N in different soil depths
* N in soil, by weather and month
* N in soil, by weather and soil depth
* N balance, per month, mean
* P2O5 balance, per month, mean
* N balance, per month, per ha
* N balance, per month, time series
* N balance, yearly, time series
* N balance per ha, yearly, time series
* P2O5 balance, per month, time series
* P2O5 balance, yearly, time series
* Overview GHGs
* GHGs by source
* MACs and GHGs
* Revenues, mean
* Revenues per ha, mean
* Costs, mean
* Costs per ha, mean
* Cash balance, mean
* Cash balance per ha, mean
* Revenues minus costs per SON, mean
* Crop costs, mean
* Revenues, time series
* Cash balance, time series
* MACs

## Using the exploitation tools for meta-modeling

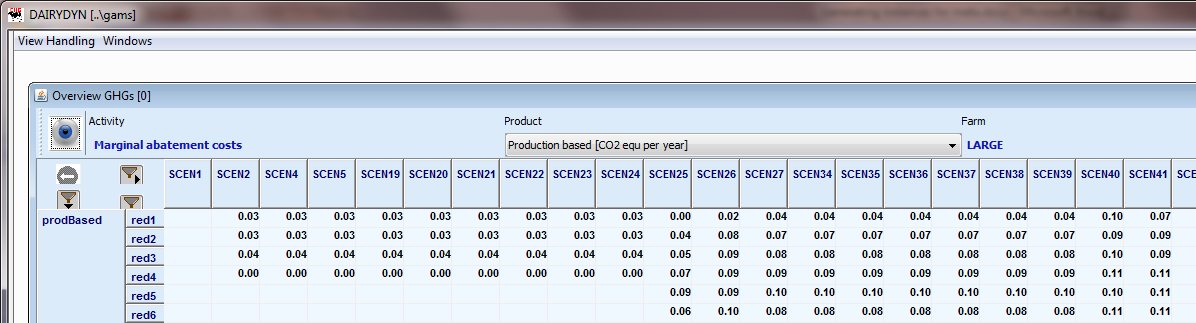
!!! info The model code comprises post-solution processing with various aggregations such as by branch, over time and calculations of indicators which can be exploited by a viewer in tables and graphs and compared across scenarios.

In the interface, the "exploit results" button will open a selection dialog to choose results from parallel runs:



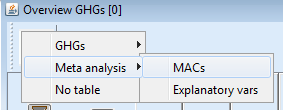


When *show results* are clicked the interface presents the results in tables:



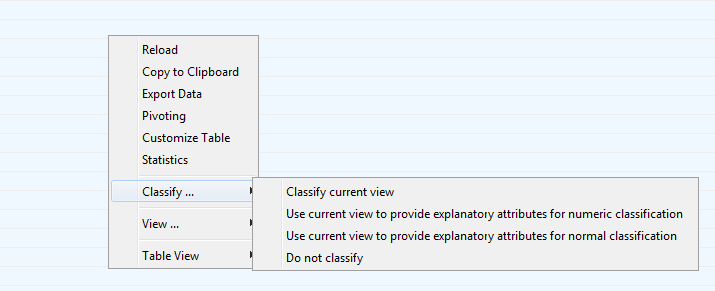
There are views necessary to use the machine learning package:

1. A view with the variable to estimate, in our case provide by the table *Meta analysis, MACs*

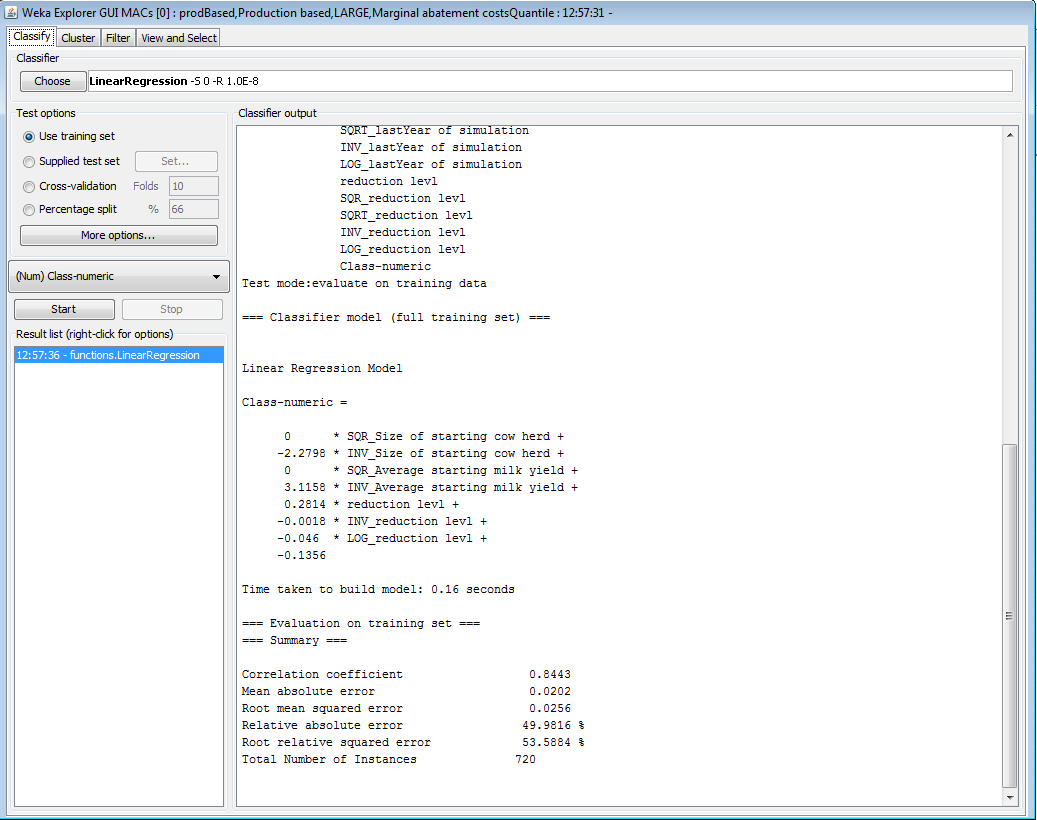


1. A view with the explanatory attributes, in our chase provided by the table *Meta analysis, explanatory vars*.

In the first view, a click in the table will open a pop-up menu from which "Classify, Classify current view" should be chosen:



Similarly, in the second view, "Classify, Use current view to provide explanatory attributes for numeric classification should be chosen". The WEKA Explorer can then be used to apply different algorithms from machine learning to the instance, the screen shot below shows the application of the multiple regression model with automatic variable transformation (automatically builds logs, square roots and inverses of the variable values) and variable selection [[9]](#footnote-202):



Additionally, the user has the possibility to view Figures, histograms or graphs of the interesting output values for graphical visualisation. Statistical characteristics like minimum, maximum or median values are automatically generated as well as mean value of the selected results and the standard deviation.



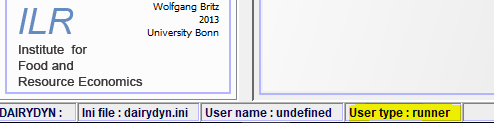
## Working with Projects

When applying FarmDyn to a set of different case studies, keeping track of the individual case study settings in the .ini files may become tedious.

In order to ease the handling, you can define so called projects. These projects can be managed from the user interface. The concept of projects is similar for any model using [GGIG](http://www.ilr.uni-bonn.de/em/staff/britz/ggig_e.htm) (FarmDyn, Capri etc.), so one can also refer to the official [user guide](http://www.ilr.uni-bonn.de/em/rsrch/ggig/GGIG_user_Guide.pdf).

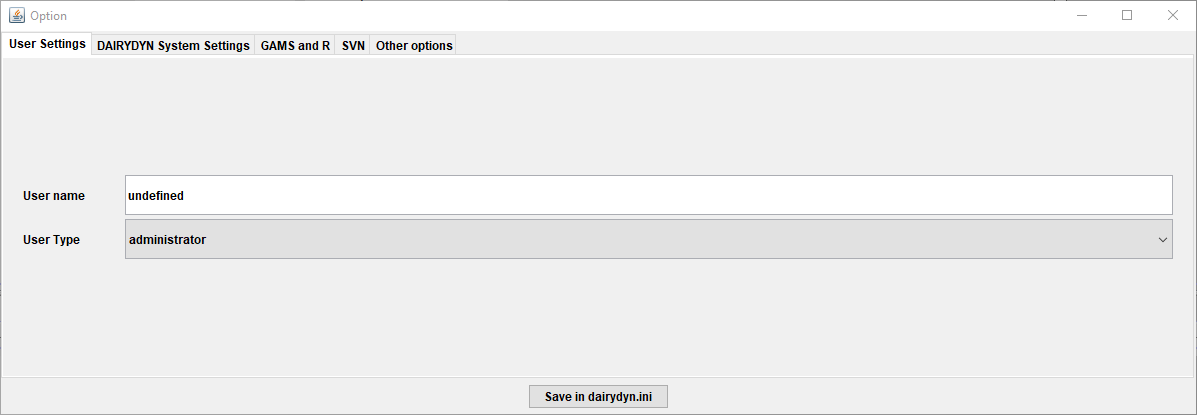
### Setting up a new project

In order to create a new project, you need to be logged in with the user type **Administrator**. You can see your current user type in the status bar at the bottom left of the GUI window.



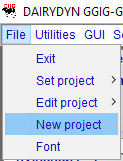
The current user type is displayed in the status bar

You can change your user type by clicking on **Settings** -> **Edit Settings**, where you will be presented with the following window:



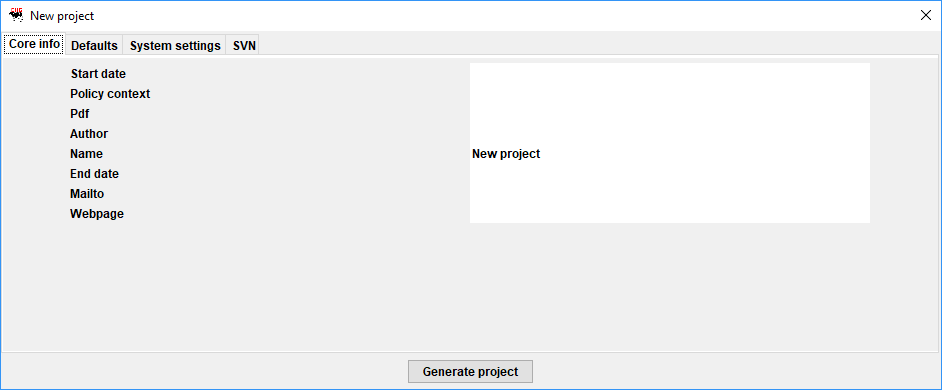
The user type can be changed in the "Edit settings" menu item under the "Settings" menu

Choosing **File** -> **New Project**



Creating a new project

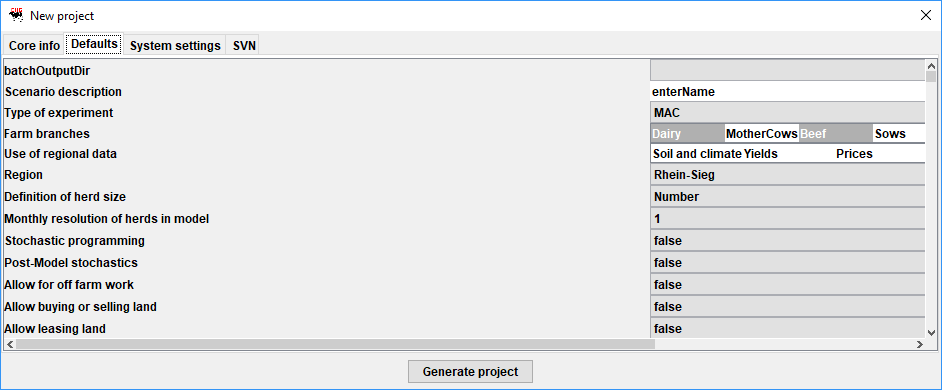
then opens the project dialog box:



Creating a new project

The dialog comprises four tabs. The first tab “Core info” shown above comprises information on the project itself. The entries “Pdf”, “Mailto” and “Webpage” generate new items in the top menu bar and provide information on the application context to the user.

The second tab “Defaults” comprises a list of GUI controls of the tool with their title and its default setting:



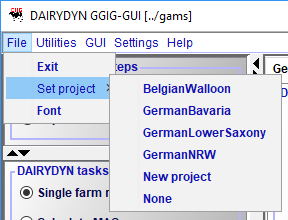
The left hand side shows the title of the control, the right hand comprises a text field with the default settings as stored in the XML-definition file for the GUI. That general default can be overwritten with a project specific one. Only these differences are stored.

Similarly, the two remaining tables show under “System settings” the location of system directories, and under the SVN tab optional project specific SVN settings.

Once you press the **Generate project** button, a new file called project\_Your Project Name\_default.ini, where Your Project Name will be replaced by the name you entered for the project, will be created in your FarmDyn GUIfolder.

### Switching projects

You can switch between projects by choosing **File** -> **Set Project** -> **Project Name**, where **Project Name** refers to the name of the project you want to switch to.



The currently selected project will be displayed in the status bar at the bottom left corner of the user-interface window.



### Editing projects

Make sure your current user type is set to **Administrator** first (see [here](#setting-up-a-new-project) for details).

You can edit an existing project by choosing **File** -> **Edit Project** -> **Project Name**, where **Project Name** refers to the name of the project you want to edit.



The project dialog as described under [*Setting up a new Project*](#setting-up-a-new-project) will be opened where the necessary changes can be done.

When you are done changing the project's default settings, make sure to press the **Update project** Button. Notice that the dialog window will not close automatically after the changes were saved.

### Important information on the project workflow

Please notice that you **cannot** change the projects default settings from the GUI tabs. The idea is as follows: In the project settings, the default values displayed in the GUI tabs are stored. However, changing these settings in the user-interface will not affect the project defaults. This way you can do sensitivity analysis or test certain settings without losing your default project values or breaking any important project specifics. If the tests turn out to be working fine, you may update the project defaults as described [*above*](#editing-projects).

# References

Alcamo J., Shaw R. & L. Hordijk (1990): The RAINS Model of Acidification. Science and Strategies in Europe. Kluwer Academic Publishers, Dordrecht, Netherlands

Britz, W. & B. Lengers (2012): R routine to estimate Heckman two stage regression procedure on marginal abatement costs of dairy farms, based on large scale outputs of the model DAIRYDYN. Technical Paper for modelling documentation, University of Bonn

Britz, W. & H.P. Witzke (2008): CAPRI model documentation 2008: Version 2. <http://www.capri-model.org/docs/capri_documentation.pdf>

Britz, W. (2010), GGIG Graphical Interface Generator User Guide, Institute for Food and Resource Economics, University Bonn, 147 pages

Britz, W. (2011): Algorithms from Machine Learning -- interesting for CAPRI? Technical discussion paper, http://www.capri-model.org/dokuwiki/..%5Cdocs%5CMachine\_learning\_in\_GUI.pdf (stand 17.11.2011)

Britz, W. (2014). A New Graphical User Interface Generator for Economic Models and its Comparison to Existing Approaches, German Journal of Agricultural Economics 63(4): 271-285

Dämmgen, U. (2009) Calculations of emission from German agriculture -- *National Emission Inventory Report (NIR) 2009 for 2007*. Landbauforschung vTI Agriculture and Forestry Research, Sonderheft 324

Davis, P.K. (1993): An introduction to variable-resolution modelling and cross-resolution model connection. RAND, National Defense Research Institute, Santa Monica, USA. <http://www.rand.org/content/dam/rand/pubs/reports/2009/R4252.pdf> (stand 13.08.2012)

DüV (2007): Verordnung über die Anwendung von Düngemitteln, Bodenhilfsstoffen, Kultursubstraten und Pflanzenhilfsmitteln nach den Grundsätzen der guten fachlichen Praxis beim Düngen. „Düngeverordnung in der Fassung der Bekanntmachung vom 27. Februar 2007 (BGBl. I S. 221), die zuletzt durch Artikel 18 des Gesetzes vom 31. Juli 2009 (BGBl. I S. 2585) geändert worden ist"

EMEP, EEA (2013): "EEA air pollutant emission inventory guidebook 2013." European Environment Agency, Copenhagen.

EMEP (2016): EEA, EMEP. "EEA air pollutant emission inventory guidebook 2016 (EEA Report No 21/2016)." European Environment Agency(2016).

FAO (2006): livestock´s long shadow, environmental issues and options. Food and Agricultural Organization, Rome Italy

FAO (2009): the state of Food and Agriculture -- livestock in the balance. Food and Agricultural Organization, Rome Italy

FNR (2013). Leitfaden Biogas. 6. Auflage, Fachagentur für Nachwachsende Rohstoffe e.V., pp. 1-244, Gülzow

Garbert, J. 2013. Ökonomische Auswirkung von Politiken zur Umsetzung der Wasserrahmenrichtlinie auf die Schweinehaltung im Münsterland. Dissertation, University of Bonn, Germany.

Grimm, V. (2002): Ten years of individual-based modelling in ecology: what have we learned, and what could we learn in the future. Ecological Modelling 115:129-148

Haenel, H.-D., et al (2018): Calculations of gaseous and particulate emissions from German agriculture 1990-2016: Report on methods and data (RMD) Submission 2018. No. 57. Thünen Report

Haenel, H.-D., Rösemann, C., Dämmgen, U., Poddey, E., Freibauer, A., Döhler, H., Eurich-Menden, B., Wulf, S., Dieterle, M., Osterburg, B. (2012): Calculations of gaseous and particulate emissions from German agriculture 1990–2010. Report on methods and data (RMD). Submission 2012. Landbauforsch, Special Issue 356, 394 pp

Hazell, P. (1971): "A linear alternative to quadratic and semivariance programming for farm planning under uncertainty." American Journal of Agricultural Economics 53.1: 53-62

Heckman, J.J. (1979): Sample Selection Bias as Specification Error. *Econometrica* 47:153-161.

Heitsch, H. et al. (2008): "Generierung von Szenariobäumen und Szenarioreduktion für stochastische Optimierungsprobleme in der Energiewirtschaft." WAGNER, H.-J.(Hrsg.): Innovative Modellierung und Optimierung. LIT Verlag.

Heitsch, H. & Römisch, W. (2009): "Scenario tree reduction for multistage stochastic programs." Computational Management Science 6.2: 117-133.

Hertel, T.W. (1997): Global Trade Analysis -- Modeling and Applications. Cambridge University Press

Huth, F.W. (1995): Die Laktation des Rindes -- Analyse, Einfluß, Korrektur. Eugen Ulmer GmbH & Co. Stuttgart

Iman, R.L. & Conover, W.J. (1980): Small sample sensitivity analysis techniques for computer models, with an application to risk assessment (with discussion): *Communication in Statistics* 9:1749-1842

Iman, R.L. & Conover, W.J. (1982): A distribution-free approach to inducing rank correlation among input variables. *Communications in Statistics -- Simulation and Computation* 11(3): 311-334

Iman, R.L., (2008): *Latin Hypercube Sampling*. Encyclopedia of Quantitative Risk Analysis and Assessment. DOI: 10.1002/9780470061596.risk0299

Iman, R.L., Helton, J.C. & Campbell, J.E. (1981a): An Approach to Sensitivity Analysis of Computer Models: Part I -- Introduction, Input Variables Selection and Preliminary Variable Assessment*. Journal of Quality Technology* 13(3): 174-183

Iman, R.L., Helton, J.C. & Campbell, J.E. (1981b): An Approach to Sensitivity Analysis of Computer Models: Part II -- Ranking of Input Variables, Response Surface Validation, Distribution Effect and Technique Synopsis. *Journal of Quality Technology* 13(4): 232-240.

IPCC (2006) *Guidelines for National Greenhouse Gas Inventories*, Volume 4, Agriculture,Forestry and other Land use. <http://www.ipcc-nggip.iges.or.jp/public/2006gl/vol4.html>

Kesicki, F. & N. Strachan (2011): Marginal abatement cost (MAC) curves: confronting theory and practice. Environmental Science and Policy 14: 1194-1204.

Kirchgessner, M. (2004): Tierernährung. 11. Neu überarbeitete Auflage. DLG-Verlags-GmbH, Frankfurt am Main

KTBL (2010): Betriebsplanung Landwirtschaft 2010/2011 -- KTBL Datensammlung. 22. Auflage, Kuratorium für Technik und Bauwesen in der Landwirtschaft e.V., Darmstadt

KTBL (2013): Faustzahlen Biogas. 3. Ausgabe, Kuratorium für Technik und Bauwesen in der Landwirtschaft e.V., Darmstadt

Lengers, B. and Britz, W. (2012). The choice of emission indicators in environmental policy design: an analysis of GHG abatement in different dairy farms based on a bio-economic model approach. Review of Agricultural and Environmental Studies, 93(2), 117-144.

Lofgren, H., Harris, R.L. & S. Robinson (2002): A standard computable general equilibrium (CGE) model in GAMS. Microcomputers in Policy Research 5. International Food Policy Research Institute, Washington, D.C., http://www.ifpri.org/sites/default/files/publications/mc5.pdf

Louhichi, K., Janssen, S., Kanallopoulos, A., Li, H., Borowski, N., Flichman, G., Hengsdijk, H., Zander, P., Fonseca, M.B., Stokstad, G., Athanasiadis, I.N., Rizzoli, A.E., Huber, D., Heckelei, T. & M. Van Ittersum] (2010): A Generic Farming System Simulator, in: Environmental and Agricultural Modelling -- Integrated Approaches for Policy Impact Assessment, Springer

McKay, M.D., Conover, W.J. & Beckman, R.J. (1979): A comparison of three methods for selecting values of input variables in the analysis of output from computer code. *Technometrics* 21(2):239-245.

Park, J.S. (1994): Optimal Latin-hypercube designs for computer experiments. *Journal of Statistical Planning and Inference* 39:95-111

ReCiPe (2016): Huijbregts, M. A. J., Steinmann, Z. J. N., Elshout, P. M. F., Stam, G., Verones, F., Vieira, M. D. M., ... & van Zelm, R. (2016). A harmonized life cycle impact assessment method at midpoint and endpoint level Report I: Characterization.

Remble, A., W. Britz, and R. Keeney (2013): Farm Level Tradeoffs in the Regulation of Greenhouse Gas Emissions. selected paper presented at the Agricultural and Applied Economics Association, 2013 Annual Meeting, August 4-6, 2013, Washington, D.C (USA).

Rosemann (2015): Rösemann C, Haenel H-D, Dämmgen U, Freibauer A, Wulf S, EurichMenden B, Döhler H, Schreiner C, Bauer B, Osterburg B (2015) Calculations of gaseous and particulate emissions from German agriculture 1990 – 2013 : Report on methods and data (RMD) Submission 2015. Braunschweig: Johann Heinrich von Thünen-Institut, 372 p, Thünen Rep 27

R-software (2012): The R project for statistical computing. Version R-2.15.1. <http://www.r-project.org/index.html> (stand 16.10.2012)

Schneider, U.A. & B.A. McCarl (2006): Appraising agricultural greenhouse gas mitigation potentials: effects of alternative assumptions. Agricultural Economics 35:277--287[Velthof, G.L. & O. Oenema] (1997) Nitrous oxide emissions from dairy farming systems in the Netherlands. *Netherlands Journal of Agricultural Science* 45:347-360

Stehfest, E. & Bouwman, L. (2006): "N 2 O and NO emission from agricultural fields and soils under natural vegetation: summarizing available measurement data and modeling of global annual emissions." Nutrient Cycling in Agroecosystems74.3: 207-228.

Vermont, B. & S. DeCara (2010): How costly is mitigation of non-CO2 greenhouse gas emissions from agriculture? A meta-analysis. doi:10.1016/j.ecolecon.2010.02.020

# Appendix

Table 1. Machinery included in the set *machType*

|  |  |  |  |
| --- | --- | --- | --- |
| Machine | Naming in FarmDyn | Definition | Reference in KTBL 14/15 |
| Tractor large | *tractor* | Traktor, 67kW | KTBL 12/13 p. 65 |
| Tractor small | *tractorSmall* | Traktor, 54kW | KTBL 12/13 p. 65 |
| Plough | *plough* | 4 Schar Anbaudrehpflug | p. 82 |
| Chiselplough | *chiselPlough* | Schwergrubber 2,5m | p. 84 |
| Seed drill | *sowMachine* | Mechanischesämaschine, 3m | p. 97 |
| Direct seed drill | *directSowMachine* | Direktsämaschine | KTBL 12/13 p.98 |
| Seed bed combination | *seedBedCombi* | Saatbettkombination, 4,5m | p. 85 |
| Disc harrow | *circHarrow* | Scheibenegge, 3m | p. 86 |
| Spring tine harrow | *springTineHarrow* | Federzinkenegge, 4,5m | p. 85 |
| Tined Weeder | *fingerHarrow* | Hackstriegel, 4,5m | p. 100 |
| Combine harvester | *combine* | Schüttelmähdrescher, 150kW | p. 110 |
| Cutterbar Cereals | *cuttingUnitCere* | Getreideschneidwerk, 4,5m | p. 111 |
| Cutterbar addition for Rapeseed | *cuttingAddRape* | Zusatzausrüstung für Rapsernte, 4,5m | p. 111 |
| Maize picker | *cuttingUnitMaiz* | Maispflückeinrichtung für Mähdrescher, 4 reihig | p. 111 |
| Rotary Harrow | *rotaryHarrow* | Kreiselegge | p. 100 |
| Mulcher | *mulcher* | Mulcher | KTBL 12/13 p.103 |
| Potato Planter | *potatoPlanter* | Kartoffellegegerät, 4 reihig | KTBL 12/13 p.100 |
| Potato harvester | *potatoLifter* | Kartoffelroder, angehängt 1 reihig | p. 112 |
| Hoe | *hoe* | Hackmaschine, 5 reihig | p. 100 |
| Ridger | *ridger* | Kartoffeldammhäufler, 4 reihig | p. 100 |
| Haulm topper | *haulmCutter* | Kartoffelkrautschläger | KTBL 12/13 p.113 |
| Fork lift truck | *forkLiftTruck* | Gabelstapler | KTBL 12/13 p.68 |
| Three-way tipper | *threewayTrippingTrailer* | Dreiseitenkippanhänger | p. 78 |
| Sprayer | *Sprayer* | Anbauspritze, 15m 1m3 | p. 101+102 |
| Single grain sowing machine | *singleSeeder* | Einzelkornsähgerät | p. 98 |
| Beet harvester | *beetHarvester* | Zuckerrüben-Köpfrode-kombination, 2 reihig | p. 113 |
| Fertilizer spreader small | *fertSpreaderSmall* | Düngerstreuer 0,8 m3 | KTBL 12/13 p.92 |
| Fertilizer spreader large | *fertSpreaderLarge* | Kalkstreuer, angehängt 4,0 m3 | KTBL 12/13 p.92 |
| Forage harvester | *chopper* | Feldhäcksler, Lohnunternehmerleistung | KTBL 12/13 p.109 |
| Corn header | *cornHeader* | Maisgebiss für Häcksler | KTBL 12/13 p.110 |
| Mower conditioner | *mowerConditioner* | Heckscheibenmähwerk mit Aufbereiter, 2,4m | p. 103 |
| Grass reseeding combination | *grasReseedingUnit* | Grasnachsämaschine, 2,5m | p. 98 |
| Tedder | *rotaryTedder* | Kreiselzettwender, 4,5m | p. 105 |
| Rake | *rake* | Einkreiselschwader, 3,5m | p. 105 |
| Roller | *roller* | Walze, 3m | p. 88 |
| Silage trailer | *silageTrailer* | Silagetransport durch Lohnunternehmen |  |
|  | *closeSilo* |  |  |
| Slurry tanker Mainbarrel | *Mainbarrel* | Vakuumtankwagen | p. 95 |
| Draghose | *draghose* | Schleppschlauch | p. 96 |
| Injector | *injector* | Gülleinjektor | p. 96 |
| Trailing shoe | *trailingshoe* | Schleppschuh | p. 96 |
| Front loader | *frontloader* | Frontlader, für 67kW | p. 71 |
| Shear grab | *shearGrab* | Schneidzange | p. 76 |
| Dung grab | *dungGrab* | Dungzange | p. 74 |
| Silo block cutter | *siloBlockCutter* | Siloblockschneider | p. 119 |
| Mixer-wagon small | *fodderMixingVeh8* | Futtermischwagen 8m3 horizontale Schnecke, mit Befüllschild | p. 121 |
| Mixer-wagon medium | *fodderMixingVeh10* | Futtermischwagen 10m3 vertikale Schnecke mit Befüllschild | p. 121 |
| Mixer-wagon large | *fodderMixingVeh16* | Futtermischwagen 16 m3 2vertikale Schnecken, mit Befüllschild | p. 121 |

Table 2. Field operations from the set *operation*

|  |  |  |
| --- | --- | --- |
| Naming in FarmDyn | Included machines | Definition |
| *Soilsample* |  | Bodenproben ziehen |
| *manDist* | tractorSmall | Gülleausbringung |
| *basFert* | tractorSmall; fertSpreaderSmall | P und K Düngung im typischerweise Herbst |
| *plow* | plough | Pflügen |
| *chiselPlow* | chiselPlough | Tiefengrubber |
| *seedBedCombi* | seedBedCombi | Saatbettkombination (Saatbettbereitung) |
| *herb* | tractorSmall; sprayer | Herbizidmaßnahme |
| *sowMachine* | tractorSmall; sowMachine | Saemaschine(Säen nach Bodernbearbeitung) |
| *directSowmachine* | directSowMachine | Direktsaatmaschine (Direktsaat) |
| *circHarrowSow* | sowMachine; circHarrow | Kreiselegge und Drillmaschine Kombination |
| *springtineHarrow* | springtineHarrow | Federzinkenegge |
| *weedvaluation* |  | Unkrautbonitur |
| *weederLight* | tractorSmall; fingerHarrow | Striegeln |
| *weederIntense* | hoe | Hacken |
| *Plantvaluation* |  | Bestandsbonitur |
| *NFert320* | tractorSmall; fertSpreaderSmall |  |
| *NFert160* | tractorSmall; fertSpreaderSmall |  |
| *combineCere* | combine; cuttingUnitCere | Mähdrusch, Getreide |
| *combineRape* | combine; cuttingUnitCere; cuttingAddRape | Mähdrusch, Raps |
| *combineMaiz* | combine; cuttingUnitMaiz | Mähdrusch, Mais |
| *cornTransport* | tractorSmall; threeWayTrippingTrailer | Getreidetransport |
| *Store\_n\_dry\_8* |  |  |
| *Store\_n\_dry\_4* |  |  |
| *Store\_n\_dry\_beans* |  |  |
| *Store\_n\_dry\_rape* |  |  |
| *Store\_n\_dry\_corn* |  |  |
| *lime\_fert* | fertSpreaderLarge | Kalkung |
| *stubble\_shallow* | chiselPlough | Stoppelbearbeitung, flach |
| *stubble\_deep* | chiselPlough | Stoppelbearbeitung, tief |
| *rotaryHarrow* | tractorSmall; rotaryHarrow | Kreiselegge |
| *NminTesting* |  | Nmin Probennahme |
| *mulcher* | tractorSmall; mulcher | Mulcher |
| *Chitting* |  | Vorkeimen |
| *solidManDist* |  | Miststreuen |
| *seedPotatoTransp* | tractorSmall; threeWayTrippingTrailer, forkLiftTruck | Pflanzkartoffeltransport |
| *potatoLaying* | potatoPlanter | Kartoffellegen |
| *rakingHoeing* |  | Hacken, Striegeln |
| *earthingUp* | tractorSmall; ridger | Häufeln |
| *knockOffHaulm* | tractorSmall; haulmCutter | Kartoffelkraut schlagen |
| *killingHaulm* |  | Krautabtöten |
| *potatoHarvest* | potatoLifter | Kartoffeln roden |
| *potatoTransport* | tractorSmall; threeWayTrippingTrailer | Kartoffeln zum Lager transportieren |
| *potatoStoring* |  | Kartoffeln lagern |
| *singleSeeder* | tractorSmall; singleseeder | Einzelkornlegegerät für Zuckerrüben/Mais |
| *weederHand* |  | Von Hand hacken |
| *uprootBeets* | beetHarvester | Zuckerrüben roden |
| *DiAmmonium* | tractorSmall; fertSpreaderSmall | Diammonphosphat streuen |
| *grinding* |  | KornMahlen |
| *disposal* |  | Erntegut festfahren |
| *coveringSilo* |  | Silo reinigen und mit Folie verschliessen, Mais |
| *chopper* | chopper | Häckseln |
| *grasReSeeding* | grasReseedingUnit | Gras nachsäen |
| *roller* | tractorSmall; roller | Walzen |
| *mowing* | tractorSmall; mowerConditioner | Mähen mit Aufbereiten |
| *raking* | tractorSmall; rake | Schwaden |
| *tedding* | tractorSmall; rotaryTedder | Wenden |
| *silageTrailer* |  | Anwelkgut bergen mit Ladewagen bergen |
| *closeSilo* |  | Silo reinigen und mit Folie verschliessen |

1. The regression models originate from the ["Zifo2" Target-value fodder optimization program](http://www.zifo-bayern.de/) of the LfL Bayern. [↑](#footnote-ref-27)
2. The regression models originate from the ["Zifo2" Target-value fodder optimization program](http://www.zifo-bayern.de/) of the LfL Bayern. [↑](#footnote-ref-44)
3. For the EEG 2004: NawaRo-Bonus, KWK-Bonus; For the EEG 2009: Nawaro-Bonus, KWK-Bonus **or** NawaRo-Bonus, KWK-Bonus and Manure-Bonus [↑](#footnote-ref-98)
4. Biogas is a mixture of methane (CH4), carbon dioxide (CO2), water vapor (H2O) and other minor gases. The gas component containing the energy content of biogas is methane. Thus, the code with respect to production refers to the methane production rather than the production of biogas. [↑](#footnote-ref-101)
5. oDM = organic dry matter; m3 = cubic meter; d = day [↑](#footnote-ref-102)
6. This assures also under the restricted number of random values for each factor the components are still represented in a fully stratified manner over the entire range, each variable has the opportunity to show up as important, if it indeed is important (Iman, 2008). [↑](#footnote-ref-158)
7. This is necessary to restrict sampling time but also guarantees to find a random sample that appropriately implies the correlation structure as proposed by the user (more detailed explanation of this later in this paper) [↑](#footnote-ref-159)
8. Another possible routine for LHS sampling is "optimumLHS(\*\*\*)". But during our test runs it did not lead to more smooth space filling random draws, but increased the runtime of the sampling process. For optimal-LHS see also Park (1994). [↑](#footnote-ref-160)
9. The problem here arises, that the simple WEKA regression routines are not prepared for two stage regressions (like e.g. Heckman two stage regression (Heckman, 1979)) like in some cases demanded for our data set as farms which have already exited are in the regression relevant dataset. This may cause a sample selection bias and lead to potentially small explanatory character of the estimated linear regression model. Therefore, generated data should be analysed by a routine, written in R (also designed by Britz and Lengers in (2012) which is available on enquiry from the responsible authors. [↑](#footnote-ref-202)