FarmDyn

true

19.04.2018

Abstract

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.

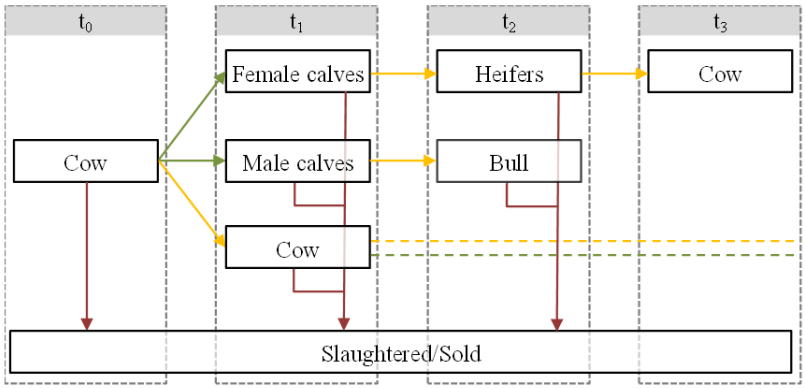
The framework is the outcome of several research activities. Its first version (named DAIRYDYN) was developed in the context of a research project financed by the German Science Foundation focusing in marginal abatement costs of dairy farms in. That project contributed the overall concept and the highly 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.

Contents

FarmDyn - A highly detailed template model for dynamic optimization of farms

Universität Bonn Institute for Food and Resource Economics Economic Modeling of Agricultural Systems Group

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 economically optimal production and investment decisions in detail at single farm scale. FARMDYN depicts various farm branches (arable cropping, pig fattening, piglet production, dairy, beef fattening, biogas plants). Its default layout maximizes the deterministic net present value 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 gases relevant for global warming. A state-of-the-art software implementation based on GAMS in combination with MIP industry solvers and a graphical user interface allows for efficient analysis. FARMDYN consists of several interacting modules (Figure 1).



Remark: Represents reproduction processes
Represents animal life development
Represents an animal being slaughtered or sold
Source: Own illustration

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. In a specific

cattle module, cow herds can be distinguished by genetic potential, including endogenous breeding towards higher milk yield. Furthermore, herds can be differentiated by animal types - such as cow, heifer, calf -, breeds, intensity levels (milk yield, daily weight gain) and feeding regimes. 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 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.

The **cropping module** optimizes the cropping pattern subject to land availability, reflecting yields, prices, machinery and fertilizing needs and other variable costs for a selectable longer list of arable crops. The crops can be differentiated by production system (plough, minimal tillage, no tillage, organic) and intensity level (normal and reduced fertilization 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 **labourmodule** (not shown in Figure 1) optimizes 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.

The **environmental accounting module** allows quantifying gas emissions of Ammonia (NH3), nitrous oxide (N2O), nitrogen oxides (NOx) and elemental nitrogen (N2). 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.

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 (2014) and FNR (2013). The equations within the template model related to the biogas module are presented in the following section.

Introduction

The dynamic single farm model documented in here is the outcome of several, partially on-going research activities. Its first version (named DAIRYDYN) was developed in the context of a research project financed by the German Science Foundation (DFG, Nr. HO3780/2-1) focusing on marginal abatement costs of dairy farms in comparison across different indicators for Green House Gases. Relating material and information on the project are available on the project related web-page: http://www.ilr.uni-bonn.de/agpo/rsrch/dfg-ghgabat/dfgabat_e.htm. That project contributed the overall concept and the highly detailed description of dairy farming and GHG accounting, while it comprises only a rudimentary module for arable cropping. It was – while improvements were going on – used for several peer reviewed papers (Lengers and Britz 2012, Lengers et al. 2013a, 2013b, Lengers et al. 2014) and conference contributions (Lengers and Britz 2011, Lengers et al. 2013c).

That version of the model was used by Garbert (2013) in her PhD thesis as the starting point to develop a first module for pig farming, however with 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 that came a 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. Activities in summer 2013 then contributed a soil pool approach for nutrient accounts, differentiated by month and soil depth layer while also introducing different soil types and three states of weather. In parallel, further information from

farm planning books was integrated (e.g. available field working days depending on soil type and climate zone) and more crops and thus machinery was added. The Graphical User Interface (GUI) and reporting parts were also enhanced. As the model now incorporated beside dairy production also other agricultural production activities, 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. Since 2014, sensitivity analysis with regard to farm endowment and prices was used to generate observations sets to estimate dual profit function which were then used in the Agent Based Model ABMSIM (http://www.ilr.uni-bonn.de/agpo/rsrch/abmsim/abmsim_e.htm). Around the same time, Till Kuhn improved substantially the nutrient flow and fertilizer management handling in the model. A project financed by the DFG (http://www.ilr.uni-bonn.de/agpo/rsrch/dfg-dairystruct/dfgdairystruct_e.htm) with a focus on Agent Based Modelling supports since 2015 that line of work. Since summer 2016, a project financed by the State of Nordrhine-Westfalia sponsors the combined application of crop-growth models, FARMDYN and ABMSIM, focusing on nutrient flows on-farm and between farms. In 2016, a stochastic programming extension with decision tress where all variables are stage contingent was developed. That extension can capture different type of risk behaviours and uses Mean Reverting Processes for the logs of prices in conjunction with a tree reduction algorithm.

That documentation is organized as follows. Following the introduction, we will discuss the core methodology with regard to the overall concept of the tool and the layout of the template model, with detail on the different modules, such as the herd, cropping, fertilization or investment module. The third section discusses the dynamic examination of the modelling approach, followed in section four by a discussion of the stochastic version and how different types of risk behaviour can be integrated. Section five describes the different GHG indicators and the calculation of Marginal Abatement Curves (MAC). The following sections present the coefficient generator, the technical implementation and the graphical user interface (GUI) which help the user to define experiments and visualize or analyze the results. For more information or access to unpublished technical papers of Britz and Lengers please feel free to contact:

Wolfgang Britz, Dr., Institute of Food and Resource Economics, University of Bonn, wolfgang.britz@ilr.uni-bonn.de

Basic methodology and tool concept

The core of the simulation framework consists of a detailed fully dynamic mixed integer optimization (MIP) model for one farm which can be extended to stochastic programming framework. The linear program maximizes utility – Net Present Value (NPV) of future farm profits, expected NPV or depicting

different types of risk behaviour – under constraints which describe (1) the production feasibility set of the farm with detailed bio-physical interactions, (2) maximal willingness to work of the family members for working on and off farms, (3) liquidity constraints, and (4) environmental restrictions.

Using MIP allows depicting the non-divisibility of investment and labour use decisions. An overview on mixed integer programming models and their theoretical concepts provide for instance Nemhauser and Wolsey (1999) or Pochet and Wolsey (2006). Non-linear relations such as yield-nutrient responses of crops are depicted by piece-wise linearization. The fully dynamic character optimizes farm management and investment decisions over a planning horizon. However, the model can also be simplified to a comparative-static one by assuming continuous re-investments, see section 3.1, or extended to a stochastic fully-dynamic one, in combination with different type of risk behaviour, see section 4.

FARMDYN presents a modular and extendable framework which allows simulating in detail changes of farm management and investment decisions under different boundary conditions such as prices or policy instruments e.g. relating to GHG abatement such as tradable permits or an emission tax, for a wide range of different farm systems found in Germany and beyond. It depicts the complex interplay of farm management and investment decisions - such as e.g. adjustments of herd sizes, milk yields, feeding practise, crop shares and intensity of crop production, manure treatment – in a highly detailed fully dynamic bio-economic simulation model, building on Mixed-Integer Programming.

In its default version, the model assumes a fully rational and fully informed farmer optimizing the net present value of the farm operation plus earnings from working off farm. A rich set of constraints describe the relations between the farmer's decision variables in financial and physical terms and his production possibility set arising e.g. from the firm's initial endowment of primary factors. These constraints also cover different relevant environmental externalities. Its dynamic approach over several years has clearly advantages in policy analysis as the adjustment path including (re-)investments can be depicted as it reflects sunk costs and other path dependencies.

The application of a mixed integer programming approach allows considering non-divisibility of labour use and investment decisions. Neglecting that aspect has at least two serious dis-advantages. Firstly, economies of scale are typically not correctly depicted as e.g. fractions of large-scale machinery or stables will be bought in a standard LP. That will tend to underestimate production costs and overestimate the flexibility in changing capital stock. Secondly, using fractions increases the production feasibility set which again will tend to increase profits and decrease costs.

Sensitivity analysis using Design of Experiments can be used to derive the simulation response under, for instance, changes in farm endowment or input and output prices.

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 describing their relations. 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] 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 farms 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 characterized 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 fertilizer-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 individual farms and the industry as a whole that optimal short and long term strategies might differ considerably.

[1] International Food Policy Research Institute

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.

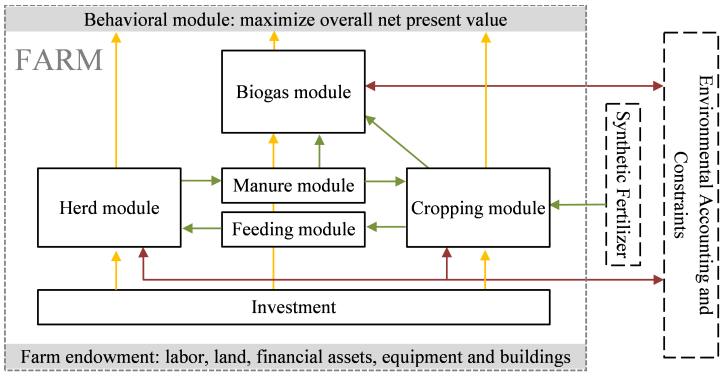


Figure 1: Overview of template model

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 maximizing 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. Female cows of milk breeds can be optionally differentiated by their genetic potential regarding the milk yield.

The model uses two different variables to describe herds: *v_herdStart* describes the number of animals by type which enter the 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 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.

```
*
*   ---- definition of standing herds
*
    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))
```

```
28                                          $ (p_mDist(t,m,t1,m1) le 0))

30                                        or
31                                         ( (abs(p_mDist(t,m,t1,m1)-12)
                                                le (p_prodLength(herds,
                                               breeds)-1))
32                                              $ (p_mDist(t,m,t1,m1)-12
                                                   le 0)) $
                                                   p_compStatHerd
33                                      )
34                                  $ sum(feedRegime,actHerds(herds,breeds,
                                        feedRegime,t1,m1)) $ isNodeBefore(
                                        nCur,nCur1)
35                          $$iftheni.sows "%farmBranchSows%" == "on"
36                                  $(not sameas(herds,"piglets"))
37                          $$endif.sows
38                                   ),
39                        v_herdStart(herds,breeds,t1,nCur1,m1)


42                        $$iftheni.ch %cowHerd%==true
43  *
44  *                     ---- minus, in case of cows, slaughtered before
        reaching the final age
45  *
46                              -sum( (slgtCows,cows) $ (sum(feedRegime,
                                actHerds(slgtCows,breeds,feedRegime,t1,m1)
                                )
47                                 $ sameas(cows,herds) $ (slgtCows.pos eq
                                     cows.pos)),
48                                 v_herdStart(slgtCows,breeds,t1,nCur1,
                                     m1))
49                        $$endif.ch
50                )
51  *
52  *           ---- 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
53  *

55                $$iftheni.sows "%farmBranchSows%" == "on"
56              + 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)))
57                                  $ (p_mDist(t,m,t1,m1) le 0) $
                                        isNodeBefore(nCur,nCur1)
58                                  $ sum(feedRegime,actHerds(herds,breeds,
                                        feedRegime,t1,m1))
59                                  $ (not sameas(herds,"sows"))
60                                  ${ ( sameas(t,t1) $ (not sameas(m  -
                                        p_prodLengthB(herds,breeds),m1)))
61                                    or ((not sameas(t,t1)) $ (sameas("Jan",
                                         m))$ (sameas( m + 11, m1)))}),

63                        v_herdStart(herds,"",t1,nCur1,m1))
64                $$endif.sows
65        ;
```

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.

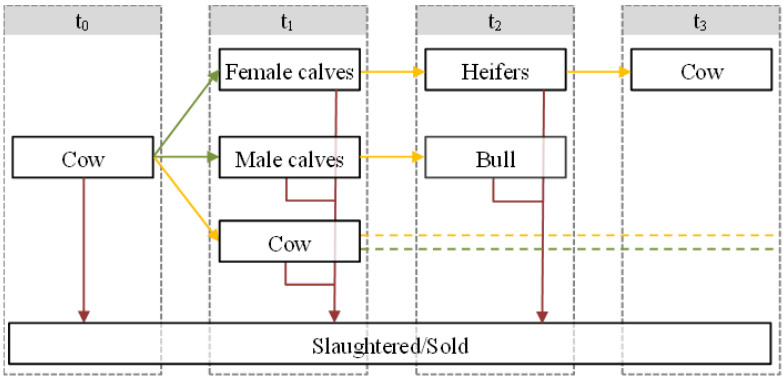
```
1  *
2  *   ---- general balance definition
3  *
4      herdsBal_(balHerds,breeds,tCur(t),nCur,m) $ (  sum(feedRegime,
           actherds(balHerds,breeds,feedRegime,t,m)) $ t_n(t,nCur)
5  *
6       $ (p_Year(t) le p_year("%lastYear%"))
7       $ (sum( (herds_from_herds(balHerds,herds,breeds),t1,m1)
8                       $ ( (-p_mDist(t,m,t1,m1) eq round(p_prodLengthB(
                            herds,breeds)/(12/card(herdM)))* (12/card(herdM)
                            ) )
9                               $  sum(feedRegime,actHerds(herds,breeds,
                                    feedRegime,t1,m1))),1)
10        $$iftheni.compStat "%dynamics%" == "comparative-static"
11          or (sum( (herds_from_herds(balHerds,herds,breeds),t1,m1)
12                      $ ( (-p_mDist(t,m,t1,m1)+12 eq round(p_prodLengthB(
                            herds,breeds)/(12/card(herdM)))* (12/card(herdM)
                            ) )
13 *                       and not (-p_mDist(t,m,t1,m1) eq round(
       p_prodLengthB(herds,breeds)/(12/card(herdM)))* (12/card(herdM)) )
14                              $  sum(feedRegime,actHerds(herds,breeds,
                                    feedRegime,t1,m1))),1))
15        $$endif.compStat
16                              or sum((bought_to_herds(herds,breeds,
                                    balherds),feedRegime) $ actherds(herds,
                                    breeds,feedRegime,t,m),1)
17                              or sum((sold_from_herds(herds,breeds,
                                    balherds),feedRegime) $ actherds(herds,
                                    breeds,feedRegime,t,m),1)  )
18                         ) ..
19 *
20 *      ---- herd starting at current time point
21 *
```

```
22            v_herdStart(balHerds,breeds,t,nCur,m)/p_herdYearScaler(
                  balHerds,breeds)

24  *
25  *       ---- plus herd starting at current time point which compete for
        the same input herds
26  *

28        + sum( herds1 $ [ (sum(herds_from_herds(herds1,herds,breeds)
29                                        $ herds_from_herds(balHerds,herds
                                              ,breeds),1)
30                        or sum(bought_to_herds(herds,breeds,herds1)
31                                $ bought_to_herds(herds,breeds,balherds),1)
                                  )
32                        $ (not sameas(balHerds,herds1)) $  sum(feedRegime,
                              actherds(herds1,breeds,feedRegime,t,m))],
33              v_herdStart(herds1,breeds,t,nCur,m)/p_herdYearScaler(herds1,
                    breeds))

35  *
36  *       ---- sold animals from the process (e.g. female calv one year old
        )
37  *
38          + sum( sold_from_herds(herds,breeds,balherds) $ sum(feedRegime,
                actherds(herds,breeds,feedRegime,t,m)),
39                                              v_herdStart(herds,breeds,t
                                                    ,nCur,m))


42            =e=
43  *
44  *       ---- equal to the starting herd of the process wich generates
        these
45  *            these herds

47        + sum( (herds_from_herds(balHerds,herds,breeds),t_n(t1,nCur1),m1)
48                      $ (( (-p_mDist(t,m,t1,m1)     eq round(p_prodLengthB(
                              herds,breeds)/(12/card(herdM)))* (12/card(herdM)
                              ) )
49                    $$iftheni.compStat "%dynamics%" == "comparative-static"
50                         or (-p_mDist(t,m,t1,m1)+12 eq round(p_prodLengthB(
                                herds,breeds)/(12/card(herdM)))* (12/card(
                                herdM)) )
51                    $$endif.compStat
52                           )     $  sum(feedRegime,actHerds(herds,breeds,
                                  feedRegime,t1,m1)) $ isNodeBefore(nCur,nCur1)
                                  ),
53                                        v_herdStart(herds,breeds,t1,nCur1,
                                              m1))

55  *
56  *       ---- bought to herd (e.g. heifers bought from market)
57  *
58        + sum( (bought_to_herds(herds,breeds,balherds))
59              $ sum(feedRegime,actherds(herds,breeds,feedRegime,t,m)),
                    v_herdStart(herds,breeds,t,nCur,m));
```

Cattle Module

!!! abstract Cow herds can be distinguished by genetic potential, including endogenous breeding towards higher milk yield. Herds can be differentiated by animal types - such as cow, heifer, calf -, breeds, intensity levels (milk yield, daily weight gain) 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 remonte. In Figure 3 the general concept of the cattle module and its decision points are illustrated.



Remark: represents reproduction processes
represents animal life development
: being slaughtered or sold
Source: Own illustration

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.

```
1  *
2  *   ---- definition of calves born
3  *
4      newCalves_("%basBreed%",t,nCur,m) $ ( sum( (calvs,feedRegime),
           actHerds(calvs,"%basBreed%",feedRegime,t,m))
5                          $ (p_Year(t) le p_year("%lastYear%")) $ t_n(t,
                               nCur)) ..
```

```
6  *
7  *      ---- new born calves (for females by genetic potential for milk
       yield) are born
8  *           from the current herd of cows
9  *
10                    v_herdStart("fCalvsRais","%basBreed%",t,nCur,m)  $
                          sum(feedRegime,actHerds("fCalvsRais","%basBreed
                          %",feedRegime,t,m))
11                  + v_herdStart("fCalvsSold","%basBreed%",t,nCur,m)  $
                        sum(feedRegime,actHerds("fCalvsSold","%basBreed%",
                        feedRegime,t,m))
12                  + v_herdStart("mCalvsSold","%basBreed%",t,nCur,m)  $
                        sum(feedRegime,actHerds("mCalvsSold","%basBreed%",
                        feedRegime,t,m))
13                  + v_herdStart("mCalvsRais","%basBreed%",t,nCur,m)  $
                        sum(feedRegime,actHerds("mCalvsRais","%basBreed%",
                        feedRegime,t,m))
14                $$iftheni.crossBreed "%crossBreeding%"=="true"
15                  + v_herdStart("fCalvsRais","%crossBreed%",t,nCur,m) $
                        sum(feedRegime,actHerds("fCalvsRais","%crossBreed%",
                        feedRegime,t,m))
16                  + v_herdStart("fCalvsSold","%crossBreed%",t,nCur,m) $
                        sum(feedRegime,actHerds("fCalvsSold","%crossBreed%",
                        feedRegime,t,m))
17                  + v_herdStart("mCalvsSold","%crossBreed%",t,nCur,m) $
                        sum(feedRegime,actHerds("mCalvsSold","%crossBreed%",
                        feedRegime,t,m))
18                  + v_herdStart("mCalvsRais","%crossBreed%",t,nCur,m) $
                        sum(feedRegime,actHerds("mCalvsRais","%crossBreed%",
                        feedRegime,t,m))
19                $$endif.crossBreed


22            =e= sum( (cows,t1,nCur1,m1,mDist) $ (sum(feedRegime,actHerds(
                  cows,"%basBreed%",feedRegime,t1,m1))
23                      $ (      (mDist.pos  eq -p_mDist(t,m,t1,m1))
24                             or (mDist.pos eq -p_mDist(t,m,t1,m1)+12) $
                                  p_compStatHerd)
25                           $ t_n(t1,nCur1)),
26                              v_herdStart(cows,"%basBreed%",t1,nCur1,m1) *
                                    p_calvCoeff(cows,"%basBreed%",mDist));
```

The calving coefficients are defined in the *Cows* tab of the Graphical User Interface (see Fig. @fig:calvingCoeff). 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 (MC). The values are stored in the parameter `p_calvAttr`.

The calving coefficient table in the Cows *tab of the GUI.*

The amount of living calves per year is then calculated from these values as follows (from *coeffgen\ini_herds.gms*)

In order to allow for an increase of the genetic yield potential of herd, two

mechanisms are available. If the farmer is allowed to buy heifers from the market, the bought heifers can have a higher milk yield than the replaced ones; the price for heifers depends on their milk yield potential. The second mechanism is to systematically breed towards higher milk yields. Breeding progress is restricted to about 200 kg per year and cow, which can be seen from the following equation.

```
*
*   --- maximum ammount of remonte of specific genetic potential
*
    remonteMax_(remonte,breeds,feedRegime,t,nCur,m) $ (actHerds(remonte,breeds,feedRegime,t,m) $ (not p_compStatHerd) $ tCur(t-2) $ t_n(t,nCur))  ..
*
*         --- heifers entering the herd
*
            sum(actHerds(remonte,breeds,feedRegime,t,m), v_herdStart(remonte,breeds,feedRegime,t,nCur,m))
*
*         --- plus heifers sold to market
*
          + sum(actHerds(heifsSold,breeds,feedRegime,t,m) $ (heifsSold.pos eq remonte.pos), v_herdSize(heifsSold,breeds,feedRegime,t,nCur,m))
            =l=
*
             sum(actHerds(cows,breeds,feedRegime,t-2,m1) $ (cows.pos eq remonte.pos), v_herdSize(cows,breeds,feedRegime,t,nCur,m1)
                                                  * p_calvCoeff(cows,breeds,"fcalvs",m1) * 0.66)
           + sum(actHerds(cows,breeds,feedRegime,t-2,m1) $ (cows.pos+1 eq remonte.pos), v_herdSize(cows,breeds,feedRegime,t,nCur,m1)
                                                  * p_calvCoeff(cows,breeds,"fcalvs",m1) * 0.33)
*
*         --- plus heifers bought from market
*
          + sum(actHerds(heifsBought,breeds,feedRegime,t,m) $ (heifsBought.pos eq remonte.pos), v_herdSize(heifsBought,breeds,feedRegime,t,nCur,m));
```

Figure 1:

Most equations - such as those relating to stable place needs - differentiate by genetic potential. Therefore, in the following equation the individual herds are aggregated to summary herds which are partly used in other equations where differentiation by genetic potential is not needed. Additionally, this provides a better overview on model results in the equation listing.

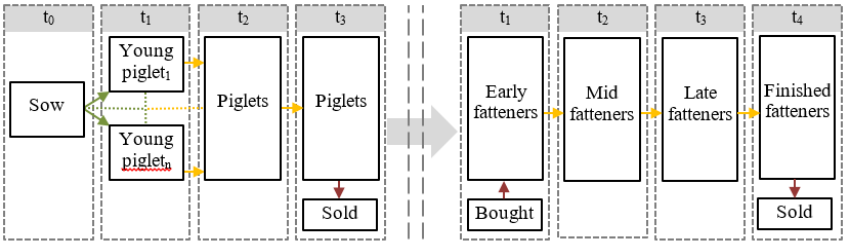
```
*
*   --- definition of summary herds (cows, heifs, female calves)
*       which enter e.g. labour need equations
*
    sumHerds_(sumHerds,breeds,feedRegime,t,nCur,m) $ (t_n(t,nCur) $  sum(sum_herds(sumHerds,possHerds) $ actHerds(possHerds,breeds,feedRegime,t,m),1)) ..

       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));
```

Figure 2:

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 piglets, which are separated from their mother after two weeks.

The pig module, similar to the cattle module, is closely linked with the general herd module. It distinguishes between a fattening branch and a piglet production branch with sows. The herd dynamics of the pig module are shown in Figure 4.



Remark: represents reproduction processes
represents animal life development
being bought or sold
Source: Own illustration

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 parameterization. After one month young piglets become piglets and remain 2 months within the herd before they are sold or transferred to the fattener branch. Labor 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 labor 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*. That allows to adapt feeding patterns, for instance to adjust nutrient output in response to legislatively given fertilizer 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 after a month transferred to the next fattening stage 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","",feedRegime,t,m)       = yes;
      actHerds("sows","",feedRegime,t,m)          = yes;
      actHerds("youngPiglets","",feedRegime,t,m)  = yes;
      actHerds("youngSows","",feedRegime,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","",feedRegime,t,m)        = yes;
       actHerds("earlyFattners","",feedRegime,t,m)   = yes;
       actHerds("midFattners","",feedRegime,t,m)     = yes;
       actHerds("lateFattners","",feedRegime,t,m)    = yes;
       actHerds("pigletsBought","",feedRegime,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.

??? note "Excurse - Computation of Output Coefficients for each Lactation Phase" 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.

```
1  ![](../media/image14.png)
2  :   Figure 6: Lactation curves of different yearly
3      milk yield potentials and average milk yield in different
4      lactation phases (30—70—100—105—60).
5      Remark: Calculation based on Huth (1995:pp.224—226)
6      Source: own illustration

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


14     |                  |  LC30    | LC100   |  LC200   | LC3005   | Dry |
15     |——————————————————|——————————|—————————|——————————|——————————|—————|
16     |Daily fraction    | 0.00356 |0.0043   | 0.00333 | 0.00233 |  0   |

18 :   Figure 7. Daily fraction of whole lactation milk
19     yield in different lactation phases
20     Remark: Own calculation based on Huth (1995, pp.224—226)

22 Following these outputs, e.g. on each of the first 30 days of
23 lactation, the cow produces 0.356% of the yearly milk yield (e.g. 28
24 kg per day for a cow which produces 8000 kg per year). In a next
25 step, these coefficients are used to calculate the sum of milk output
26 in each lactation phase to further calculate feed requirements
27 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.

The model differentiates for each herd between requirements for energy in NEL, raw protein and maximum dry matter. So far, for heifers and calves only one

```
p_mlkPerDay(cows,breeds,"LC30")  = 0.003555556 * sum(t $ (t.pos eq 1), p_OCoeff(Cows,"milk",t)*1000);
p_mlkPerDay(cows,breeds,"LC100") = 0.004333333 * sum(t $ (t.pos eq 1), p_OCoeff(Cows,"milk",t)*1000);
p_mlkPerDay(cows,breeds,"LC200") = 0.003333333 * sum(t $ (t.pos eq 1), p_OCoeff(Cows,"milk",t)*1000);
p_mlkPerDay(cows,breeds,"LC305") = 0.002333333 * sum(t $ (t.pos eq 1), p_OCoeff(Cows,"milk",t)*1000);
```

Figure 3:

feeding phase is depicted such that daily requirements during the production process are identical.

The distribution of the requirements for cows in specific lactation periods *p_reqsPhase*, over the months, *m*, depends on the monthly distribution of births, *p_birthDist*, as can be seen in the following equation.

```
*
* --- cows:
*
  p_reqsMonthly(cows,breeds,phase,m,reqsAll) = 0;
*
* --- last 2 months - 60 days
*
  p_reqsMonthly(cows,breeds,"dry",m,reqsAll)   = p_reqsPhase(cows,breeds,"dry",reqsAll) * (p_birthDist(m++1)*0.5+p_birthDist(m++2)*0.5);
*
* --- last 5.5 months (= 165 days / 30 ) - 105 days
*
  p_reqsMonthly(cows,breeds,"LC305",m,reqsAll) = p_reqsPhase(cows,breeds,"LC305",reqsAll)
                  * (   p_birthDist(m++3)/3.5
                      + p_birthDist(m++4)/3.5
                      + p_birthDist(m++5)/3.5
                      + p_birthDist(m++6) * 0.5/3.5);
*
* --- last 8.83 months (= 265 days / 30 ) - 100 days
*
  p_reqsMonthly(cows,breeds,"LC200",m,reqsAll) = p_reqsPhase(cows,breeds,"LC200",reqsAll)
                                               * (  p_birthDist(m++6) * 30/100 * 0.5
                                                  + p_birthDist(m++7) * 30/100
                                                  + p_birthDist(m++8) * 30/100
                                                  + p_birthDist(m++9) * 25/100);
*
* --- last 11.167 months (= 335 days / 30 )
*
  p_reqsMonthly(cows,breeds,"LC100",m,reqsAll) = p_reqsPhase(cows,breeds,"LC100",reqsAll)
                                               * (  p_birthDist(m++9)  * 5/70
                                                  + p_birthDist(m++10) * 30/70
                                                  + p_birthDist(m++11) * 30/70);
*
* --- last months (= 365 days / 30 )
*
  p_reqsMonthly(cows,breeds,"LC30",m,reqsAll)  = p_reqsPhase(cows,breeds,"LC30",reqsAll)  *  p_birthDist(m++12);

*
* --- scale requirements per month to fit total
*
  p_reqsMonthly(cows,breeds,phase,m,reqsAll) $ p_reqsPhase(cows,breeds,phase,reqsAll)
      = p_reqsMonthly(cows,breeds,phase,m,reqsAll)
          *  p_reqsPhase(cows,breeds,phase,reqsAll) /sum(m1,p_reqsMonthly(cows,breeds,phase,m1,reqsAll));
```

Figure 4:

In order to test different model configurations and to reduce the number of equations and variables in the model, the monthly requirements, *p_Monthly*, are aggregated to an intra-annual planning period, *intrYPer*, for which a different feed mix can be used for each type of herd, see the following equation.

```
p_reqs(herds,breeds,phase,intrYPer,reqsAll) = sum( intrYPer_m(intrYPer,m), p_reqsMonthly(herds,breeds,phase,m,reqsall));
```

Figure 5:

The requirements per planning period, *p_reqs*, enter the equation structure of the model. The equations are differentiated by herd, year, planning period and state-of-nature, and ensure the requirements are covered by an appropriate feed mix made out of different feeding stuff [2]. The composition of the feed mix is determined endogenously. In general, a herd consists of cows of different milk yield potentials, heifers and different types of calves. Total feed requirements for

[2] Grass and maize silage and grass from pasture, which are own produced, and three type of concentrates

a farm in the different intra-yearly planning periods depend on the distributions 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), state-of-nature and year, if the requirement phases are not defined for specific time spans after the herd start:

```
*
*   --- requirement constraints (per herd, year and SON)
*
    reqs_(possHerds,breeds,feedRegime,reqs,phase,intrYPer,t,nCur,s) $ sum( m_to_herdm(m,herdm)  $ ((not p_reqsPhaseLength(possHerds,breeds,phase))
                                                                      $ p_reqs(possHerds,breeds,phase,intrYPer,"DRMX")
                                                                      $ intrYPer_m(intrYPer,m)),
                                                                        actHerds(possHerds,breeds,feedRegime,t,herdm) $ tCur(t) $ t_n(t,nCur)) ..
*
*      --- herd size times requirements per head, minus year and SON specific reduction in milk yield

       sum((m_to_herdm(m,herdm)) $ (intrYPer_m(intrYPer,m) $ actHerds(possHerds,breeds,feedRegime,t,herdm)),
             v_herdSize(possHerds,breeds,feedRegime,t,nCur,herdm)* p_reqs(possHerds,breeds,phase,intrYPer,reqs))
      /sum((m_to_herdm(m,herdm)) $ (intrYPer_m(intrYPer,m) $ actHerds(possHerds,breeds,feedRegime,t,herdm)),1)

     - v_redMlk(possHerds,breeds,phase,intrYPer,t,nCur,s)*p_reqsRed(reqs) $ cows(possHerds)
*
*      --- must be covered by feeding times the content of the feed stuff
*
         =L= sum(feeds, v_feeding(possHerds,breeds,phase,intrYPer,feeds,t,nCur,s) * p_cont(feeds,reqs));
```

Figure 6:

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:

```
*
*   --- requirement constraints (per herd, year and SON)
*
    reqsPhase_(possHerds,breeds,reqs,phase,intrYPer,t,nCur,s) $ sum(intrYPer_m(intrYPer,m) $ (p_reqsPhaseLength(possHerds,breeds,phase)
                                                                $ p_reqsPhase(possHerds,breeds,phase,"DRMX")),
                                                                    sum(feedRegime, actHerds(possHerds,breeds,feedRegime,t,m))$ tCur(t)  $ t_n(t,nCur)) ..
*
*        --- herds which started in the months before the production length, in case for piglets a separate construct is used
*

        +  sum( (feedRegime,m,t1,nCur1,m1) $ ( intrYPer_m(intrYPer,m) $ tCur(t1) $ t_n(t1,nCur1) $ isNodeBefore(nCur,nCur1) $
                                 (   (abs(p_mDist(t,m,t1,m1))-p_reqsPhaseStart(possHerds,breeds,phase)    lt p_reqsPhaseLength(possHerds,breeds,phase))
                                  or (abs(p_mDist(t,m,t1,m1)-12)-p_reqsPhaseStart(possHerds,breeds,phase) lt p_reqsPhaseLength(possHerds,breeds,phase)) $ p_compstatHerd
                                 )
                               $ (    (p_mDist(t,m,t1,m1)+p_reqsPhaseStart(possHerds,breeds,phase)    le 0)
                                  or (p_mDist(t,m,t1,m1)+p_reqsPhaseStart(possHerds,breeds,phase)-12 le 0) $ p_compstatHerd  )
                               $actHerds(possHerds,breeds,feedRegime,t1,m1)
                               ),
*
*                    -- number of months that herd in tha requirement phase during that period
*                       multiplied with monthly requirements
*
                     v_herdStart(possHerds,breeds,feedRegime,t1,nCur1,m1) * p_reqsPhase(possHerds,breeds,phase,reqs))
*
*
*        --- minus, in case of cows, slaughtered before reaching the final age
             -sum( (actHerds(slgtCows,breeds,feedRegime,t,m),cows) $ ( cows(possHerds) $ (slgtCows.pos eq cows.pos)),
                      v_herdStart(slgtCows,breeds,feedRegime,t,nCur,m) * p_reqsPhase(possHerds,breeds,phase,reqs))

     - v_redMlk(possHerds,breeds,phase,intrYPer,t,nCur,s)*p_reqsRed(reqs) $ cows(possHerds)
*
*      --- must be covered by feeding times the content of the feed stuff
*
         =L= sum(feeds, v_feeding(possHerds,breeds,phase,intrYPer,feeds,t,nCur,s) * p_cont(feeds,reqs));
```

Figure 7:

This extension of the feeding module is used for the raising calves process. As a representative example the raising calves process for females calves, fCalvsRais, from birth to the 12th month. Three requirement phases are defined: *0_2*, *3_7* and *8_12*; the labels indicate the start and end month of each phase:

```
set phase "lactation phase / dry / general / months in year" / LC30,LC100,LC200,LC305,dry,GEN,0_2,3_7,8_12 /;
```

Figure 8:

The requirements are defined for each phase separately, as representative examples the first two phases are illustrated in the following.

The link between the variable *v_herdStart* and per phase requirements when using the *reqs1_* equation is shown in the listing below. The requirements for

```
*   -------------first 7 weeks - 49 days (weaning period):
*
*   --- milking powder in increasing quant.
*
  p_reqsPhase("fCalvsRais",breeds,"0_2","NEG")   $ (not meatBreeds(breeds)) = 11.79  * (2.+2.+5.+6.+7.+8.+8.)/7 * 61 * 0.3;
*
  p_reqsPhase("fCalvsRais",breeds,"0_2","CRPR")  $ (not meatBreeds(breeds)) = 0.35   * (2.+2.+5.+6.+7.+8.+8.)/7 * 61 * 0.3;
*
  p_reqsPhase("fCalvsRais",breeds,"0_2","DRMN")  $ (not meatBreeds(breeds)) = 0.93   * (2.+2.+5.+6.+7.+8.+8.)/7 * 61 * 0.3;
*
  p_reqsPhase("fCalvsRais",breeds,"0_2","DRMX") = -p_reqsPhase("fCalvsRais",breeds,"0_2","DRMN")*1.2;

  p_reqsPhaseLength("fCalvsRais",breeds,"0_2") = 2;

*
*
*       (see KTBL, page 547)
*
*   --- next 163 days: 80 -> 173kg
*
  p_avLivewgt("fCalvesRais",breeds)          = ( 90. + 173.) / 2. ;
  p_fattngDays("fCalvsRais",breeds)          = 152.;
  p_dailyWIncr("fCalvsRais",breeds,"3_7")    = (173.-90.)/p_fattngDays("fCalvsRais",breeds);

  p_reqsPhaseLength("fCalvsRais",breeds,"3_7")  = 5;
  p_reqsPhaseStart("fCalvsRais",breeds,"3_7")   = 2;
*
*  --- Calculation of Net energy for growth according to IPCC 2006 (Eq. 10.6)
*
  p_reqsPhase("fCalvsRais",breeds,"3_7","NEG") =
                        +  22.02 * (p_avLivewgt("fCalvesRais",breeds) / (0.8 * p_femwgt))**0.75
                                 * (p_dailyWIncr("fCalvsRais",breeds,"3_7")**(1.097)) * p_fattngDays("fCalvsRais",breeds);
*
* --- net energy for maintenance (in IPCC: NEm=cfi*(Weight)**0.75, IPCC 2006)
*
  p_reqsPhase("fCalvsRais",breeds,"3_7","NEM")       = ( p_avLiveWgt("fCalvesRais",breeds)**0.75 * 0.322) * p_fattngDays("fCalvsRais",breeds);

*
* --- net energy for activity (in IPCC: NEa, Ca: Stall/Pasture, shares from N2O Manure Management Miterra, IPCC 2006)
*                                                  Hälfte der Zeit sind die Kälber auf der Weide
  p_reqsPhase("fCalvsRais",breeds,"3_7","NEA")       =  ( p_avLiveWgt("fCalvesRais",breeds)**0.75 * 0.322) * 0.17 *
                                                        0.5 * p_fattngDays("fCalvsRais",breeds);
*
*
  p_reqsPhase("fCalvsRais",breeds,"3_7","CRPR") =
        + (   0.00039 * p_avLiveWgt("fCalvesRais",breeds) + 0.000628 * p_dailyWIncr("fCalvsRais",breeds,"3_7") * 1000. + 0.0193 ) * p_fattngDays
*
  p_reqsPhase("fCalvsRais",breeds,"3_7","DRMN") =
        + ( 0.0271 * p_avLiveWgt("fCalvesRais",breeds) - 0.433 ) * p_fattngDays("fCalvsRais",breeds) * 0.9;
*
  p_reqsPhase("fCalvsRais",breeds,"3_7","DRMX") = -p_reqsPhase("fCalvsRais",breeds,"3_7","DRMN");
*
```

Figure 9:

the first two months, *0_2*, are only entering the first intra-year feeding period (which covers the months January to April). The requirements for the next 5 months, *3_7*, are distributed with a weighting of 2:2:1 over the first three intra-year periods (two months in the period of January to April, JAN_APR, two months in the period of May to June, MAY_JUN, and one in the period of July to August, JUL_AUG). Similarly, the periods for the last five months, *8_12*, enter with a weighting of 1:2:2 over the last three feeding periods of that year.

```
v_herdStart(fCalvsRais,HF,2011,JAN)
-----------------------------------------------------------------------
     1171.2523  reqs1_(fCalvsRais,HF,ENNE,0_2,JAN_APR,2011,s1)
     1103.5245  reqs1_(fCalvsRais,HF,ENNE,3_7,JAN_APR,2011,s1)
     1103.5245  reqs1_(fCalvsRais,HF,ENNE,3_7,MAY_JUN,2011,s1)
      551.7622  reqs1_(fCalvsRais,HF,ENNE,3_7,JUL_AUG,2011,s1)
      923.3353  reqs1_(fCalvsRais,HF,ENNE,8_12,JUL_AUG,2011,s1)
     1846.6707  reqs1_(fCalvsRais,HF,ENNE,8_12,SEP_OCT,2011,s1)
     1846.6707  reqs1_(fCalvsRais,HF,ENNE,8_12,NOV_DEC,2011,s1)
```

Figure 10:

If the herd starts one month later in February (see the following listing), the weights are shifted accordingly and one fifth of the requirements for the last five months, *8_12*, occurs in the first feeding period of the next year 2012.

The model allows to not fully exploit the genetic potential of cows, based on the endogenous variable *v_redMlk*. Lower utilization reduces requirements for a specific cow herd, in a specific lactation period, year and planning period by the amount of energy and protein requirement for a specific amount of milk and

```
v_herdStart(fCalvsRais,HF,2011,FEB)
                (.LO, .L, .UP, .M = 0, 0, 114.285714285714, 0)
---------------------------------------------------------------------------
     1171.2523  reqs1_(fCalvsRais,HF,ENNE,0_2,JAN_APR,2011,s1)
      551.7622  reqs1_(fCalvsRais,HF,ENNE,3_7,JAN_APR,2011,s1)
     1103.5245  reqs1_(fCalvsRais,HF,ENNE,3_7,MAY_JUN,2011,s1)
     1103.5245  reqs1_(fCalvsRais,HF,ENNE,3_7,JUL_AUG,2011,s1)
      923.3353  reqs1_(fCalvsRais,HF,ENNE,8_12,JAN_APR,2012,s1)
     1846.6707  reqs1_(fCalvsRais,HF,ENNE,8_12,SEP_OCT,2011,s1)
     1846.6707  reqs1_(fCalvsRais,HF,ENNE,8_12,NOV_DEC,2011,s1)
```

Figure 11:

reduces milk production of the farm accordingly.

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.

```
*
*   --- definition of feeduse from feeding coefficients, yearly
*
    feedUse_(feedsY,t,nCur,s) $ (tCur(t) $ t_n(t,nCur)) ..

       v_feedUse(feedsY,t,nCur,s)

           =e= sum( (possHerds,breeds,phase,intrYPer)
                      $ (p_reqs(possHerds,breeds,phase,intrYPer,"DRMX") $ sum((feedRegime,m),actHerds(possHerds,breeds,feedRegime,t,m))),
                                                  v_feeding(possHerds,breeds,phase,intrYPer,feedsY,t,nCur,s));
```

Figure 12:

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.

```
                                                  v_feeding(possHerds,breeds,phase,intrYPer,feedsY,t,nCur,s));
*
*   --- definition of feeduse from feeding coefficients, intra-year feeding period
*
    feedUseM_(feedsM,intrYper,t,nCur,s) $ (tCur(t)  $ t_n(t,nCur)) ..

       v_feedUseM(feedsM,intrYPer,t,nCur,s)

           =e= sum( (possHerds,breeds,phase)
                      $ (p_reqs(possHerds,breeds,phase,intrYPer,"DRMX") $ sum((feedRegime,m),actHerds(possHerds,breeds,feedRegime,t,m))),
                                                  v_feeding(possHerds,breeds,phase,intrYPer,feedsM,t,nCur,s));
```

Figure 13:

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, 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 to 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 following equations and table show a part of the feeding requirements definition as well as minimum and maximum amounts of certain feeding products.

```
* --- Norm feed requirements
        p_feedReqPig("earlyFattners","normFeed",feedAttr)   = (p_dailynutrientReqPhaseFeed("stg28_40","normFeed",feedAttr) * p_daysInMassPhase("stg28_40")
                                                             + p_dailynutrientReqPhaseFeed("stg40_118","normFeed",feedAttr)* (30- p_daysInMassPhase("stg28_40"));

        p_feedReqPig("midFattners","normFeed",feedAttr)     =    p_dailynutrientReqPhaseFeed("stg40_118","normFeed",feedAttr) * 30;

        p_feedReqPig("lateFattners","normFeed",feedAttr)    =    p_dailynutrientReqPhaseFeed("stg40_118","normFeed",feedAttr) * 30;

        p_feedReqPig("Fattners","normFeed",feedAttr)        =    p_dailynutrientReqPhaseFeed("stg40_118","normFeed",feedAttr)
                                                               * ( sum(massPhases, p_daysInMassPhase(massPhases) $ massPhases_feedRegime(massPhases,"normFeed")) - 90 );
```

Figure 14:

```
Table p_feedMinPig(herds,feedspig,feedRegime)   "feedmix requirements accounting for minerals, feed texture, digestability"
                                soybeanMeal.normFeed          soybeanOil.normFeed          minFu.normFeed
     earlyFattners                      0.20                         0.001                       0.015
     midFattners                        0.18                         0.001                       0.015
     lateFattners                       0.16                         0.001                       0.015
     fattners                           0.12                         0.001                       0.015

     sows                               0.01                         0.001                       0.01
     piglets                            0.19                         0.001                       0.01

;

* --- Maximum of feed requirements

     p_feedMaxPig(herds,cereFeedPig,feedRegime)  = 0.85;
     p_feedMaxPig(herds,"soybeanOil",feedRegime) = 0.03;
     p_feedMaxPig(herds,"minfu",feedRegime)      = 0.04;
     p_feedMaxPig(herds,"soybeanMeal",feedRegime)= 0.3;
```

Figure 15:

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.

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.

```
*
*   --- feeding requirements for energy, crude protein, lysin, phosphatefeed and mass
*
    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)
$iftheni "%farmBranchSows%" == "on"                   * 12/card(herdM)
$endif
                                    =L=
                              sum(feedspig , v_feedingPig(possherds,feedsPig,feedRegime,t,nCur,m) * p_feedAttrPig(feedsPig,feedAttr));

*
*  --- Either purchased or own produced product
*
    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));
```

Figure 16:

Cropping, Land and Land Use

!!! abstract The cropping module optimizes the cropping pattern subject to land availability, reflecting yields, prices, machinery and fertilizing needs and other variable costs for a selectable longer list of arable crops. The crops can be differentiated by production system (plough, minimal tillage, no tillage, organic) and intensity level (normal and reduced fertilization 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 and land (gras, arable) type and size.

Crop activities are differentiated by crop, *crops*, soil types, *soil,* management intensity, *intens*, and tillage type, *till*. 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

Crop activities are defined with a yearly resolution and can be adjusted to the state of nature in the partially stochastic version. The farmer is assumed to be able to adjust on a yearly basis its land use to a specific state of nature as long as the labour, machinery and further restrictions allow for it. Land is differentiated between arable and permanent grass land, *landType*, the latter is not suitable for arable cropping. 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, as well as 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.

```
*
*   --- land endowment definition per plot (initial size plus bought adjacent plots)
*
    totPlotLand_(plot,tCur(t),nCur) $ (p_plotSize(plot) $ t_n(t,nCur)) ..

       v_totPlotLand(plot,t,nCur)

            =L=
*
*            --- 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))
~            ;
```

Figure 17:

Total cropped land is defined by the land occupied by the different crops, *v_cropHa*. The *c_s_t_i* set defines the active possible combinations of crops, soil type, tillage type and management intensity.

```
*
*   --- total cropped land in each year and SON
*
    croppedLand_(landType,soil,tCur(t),nCur,s)  $ t_n(t,nCur) ..

       v_croppedLand(landType,soil,t,nCur,s)
          =L= 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,s));
```

Figure 18:

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:

```
*
*   --- land constraint: crop levels plus renting out cannot exceed available land per plot
*                        in each year and SON
*
    plotland_(plot,tCur(t),nCur,s) $ (p_plotSize(plot) $ t_n(t,nCur)) ..
*
         v_croppedPlotLand(plot,t,nCur,s)
*
$ifi %landLease% == true + v_rentOutPlot(plot,t,nCur)
*
            =L= v_totPlotLand(plot,t,nCur);
~
```

Figure 19:

Maximum rotational shares, *p_maxRotShare*, enter *cropRot_*, which is only active if no crop rotations are used (see chapter 2.3.2).

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:

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

```
*
*   --- crop rotation constraints: each crop can occupy a max. share on cropped land
*
    cropRot_(landType,curCrops(crops),plot,tCur(t),nCur,s)
        $ (  sum(c_s_t_i(crops,plot,till,intens)
                                $ (v_cropHa.up(crops,plot,till,intens,t,nCur,s) ne 0),1)
                        $ (sum(plot_soil(plot,soil), p_maxRotShare(Crops,soil)) ne 1)
                        $  crops_t_landType(crops,landType)
                        $ t_n(t,nCur)  ) ..

             sum( c_s_t_i(crops,plot,till,intens), v_cropHa(crops,plot,till,intens,t,nCur,s))
                  =l= v_croppedPlotLand(plot,t,nCur,s) * sum(plot_soil(plot,soil),p_maxRotShare(crops,soil));
```

Figure 20:

```
*
*   --- Maximum stocking rate allowed
*       (ensures that a certain ammount of land is present)
*
    luLand_(tCur(t),nCur) $ t_n(t,nCur)    ..

      sum( 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), 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));
```

Figure 21:

(shown in the following list, see *model\templ_decl.gms*), set *rot*, show the order of the crops in the rotations. Each line depict a sequence 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
```

Figure 22:

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).

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

In order to use the crop rotations in the model equations, three cross sets are

```
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
```

Figure 23:

```
set cropTypes_crops(cropTypes,crops) / winterCere.winterCere
                                       summerCere.(summerCere,maizCorn,maizCCM,WheatGPS)
                                       other.(winterrape,summerBeans,summerPeas)
                                       potatoes.potatoes
                                       sugarbeet.sugarbeet
                                       idle.idle
                                     /;
```

Figure 24:

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;
```

Figure 25:

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

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 four constraints, while the left hand side exhausts these hectares in the current, next and after next year based on the rotations grown in these years.

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

Labour

!!! abstract The labour module optimizes 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.

General Concept

The template differentiates between three type of labour on farm:

```
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;
```

Figure 26:

```
--- rotation crops in first year of rotation

rotHa0_(cropTypes,plot,tCur(t),nCur,s) $ ( (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,s) ne 0),1))) ..

    sum( (cropTypes_crops(cropTypes,crops),c_s_t_i(crops,plot,till,intens)), v_cropHa(crops,plot,till,intens,t,nCur,s))

        =E=   sum(cropType0_rot(cropTypes,curRot(rot)), v_rotHa(rot,plot,t,nCur,s));

--- rotation crops in second year of rotation

rotHa1_(cropTypes,plot,tCur(t),nCur,s) $ ((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,s) 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,s))

        =E=   sum((cropType1_rot(cropTypes,curRot(rot)),t_n(t+1,nCur1)), v_rotHa(rot,plot,t+1,nCur1,s));

--- rotation crops in third year of rotation

rotHa2_(cropTypes,plot,tCur(t),nCur,s) $ ((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,s) 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,s))

        =E=   sum((cropType2_rot(cropTypes,curRot(rot)),t_n(t+2,nCur1)), v_rotHa(rot,plot,t+2,nCur1,s));
```

Figure 27:

1. **General management and further activities for the whole farm,** *p_labManag("farm","const"*, which are needed as long as the farm is not given up ,$v\_hasFarm = 1$, *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, pig fattening, sows. The necessary working hours are broken down into a base need, *const* which is linked to having the 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.

```
*
*   ---- yearly labour restriction
*
    LabTot_(tCur(t),nCur,s) $ t_n(t,nCur) ..
        sum(m, v_labTot(t,nCur,m,s)) =L= p_yearlyLabH(t);
```

Figure 28:

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.

```
*
*   --- Akh per year: 52 weeks times 40 hours a week
*
    p_yearlyLabH(t)   =  %AkhFirst%   * min(1,%Aks%)
                         + %AkhSecond%  * min(1,%Aks%-1) $ (%Aks% > 1)
                         + %AkhFurther% * min(1,%Aks%-2) $ (%Aks% > 2);
```

Figure 29:

The maximal work hours per month is defined in the following statement, represented by the parameter *p_monthlyLabH*:

```
*  --- akh per month: much more then yearly sum to allow covering work peaks
*                    (e.g. at harvest time, up 12 days for each normal work days monday-friday)
*
*
   p_monthlyLabH(t,m) =  max(p_yearlyLabH(t) / 365 * p_daysPerMonth(m)*1.2,  %Aks% * 12 * p_daysPerMonth(m) * 5/7);
```

Figure 30:

The template considers 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.

```
*
*   --- labour use for crops and herds and off-farm, per month,
*
    labTotSM_(tCur(t),nCur,m,s) $ t_n(t,nCur)  ..
*
*       --- sum of work in hours in current month
*
       v_LabTot(t,nCur,m,s) =e=
*
*      --- labour use for crops and herds
*
           v_labCropSM(t,nCur,s,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,s)
*
*        --- labour use for biogas plant
*
$ifi %biogas%== true + sum((curBhkw(bhkw)), v_labBioGas(bhkw,t,nCur,m))
       ;
```

Figure 31:

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:

```
*
*   --- definition of branch size in ha or number of animals
*
    branchSize_(branches,tCur(t),nCur) $ (sum(branches_to_acts(branches,acts) ,1) $ t_n(t,nCur)) ..

       v_branchSize(branches,t,nCur) =E= sum((branches_to_acts(branches,curCrops(crops)),plot,till,intens,s)
                                          $ c_s_t_i(crops,plot,till,intens),
                                            v_cropHa(crops,plot,till,intens,t,nCur,s) * p_prob(s))

$iftheni %herd% == true
                                  + sum( (branches_to_acts(branches,possHerds),breeds,feedRegime,m) $ actHerds(possHerds,breeds,feedRegime,t,m),
                                        v_herdSize(possHerds,breeds,feedRegime,t,nCur,m)
                                          * 1/min(12,p_prodLength(possHerds,breeds))

$$iftheni.fat not "%farmBranchFattners%" == "on"
                                          * 12/card(herdM)
       $$endif.fat)
$endif
    ;
```

Where the cross-set, *branches_to_acts,* defines which activities count to a certain branch:

```
    set branches_to_acts(branches,acts) /
$ifi "%farmBranchArable%"    == "on"   cashCrops.(winterCere,winterBarley,summerCere,winterRape,summerBeans,summerPeas,
$ifi "%farmBranchArable%"    == "on"                MaizCorn,potatoes,sugarBeet,WheatGPS,MaizCCM)
$iftheni.dairy "%farmBranchDairy%"     == "on"
                                       dairy.cows
$ife  %nMotherCows%>0                  motherCows.motherCow
$endif.dairy
$ifi "%farmBranchSows%"      == "on"   sowPig.sows
$ifi "%farmBranchFattners%" == "on"    fatPig.fattners
                                         /;
```

Figure 32:

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

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

```
*
*   --- trigger for having branches (steers management labour need)
*
    hasBranch_(branches,tCur(t),nCur)  $ (sum(branches_to_acts(branches,acts) ,1) $ t_n(t,nCur))  ..
       v_branchSize(branches,t,nCur) =l= v_hasBranch(branches,t,nCur) * 10000;
```

Figure 33:

```
*
*   --- trigger for having farm (steer general management labour need)
*
    hasFarm_(branches,tCur(t),nCur) $ ((not sameas(branches,"farm")) $ t_n(t,nCur)) ..
       v_hasBranch(branches,t,nCur)  =l= v_hasFarm(t,nCur);
```

Figure 34:

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

```
*
*   ---- definition of on farm work
*
    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"));
```

Figure 35:

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 in addition by the time requirement per stable place, which is differentiated by stable type. This formulation allows labour saving scale effects related to the stable size:

A similar equation exists for crops, however, crop labour need is differentiated here by state of nature in the partial stochastic version. 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 and manure:

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

```
*
*   --- labour need of herds, per month
*
    labHerdM_(tCur(t),nCur,m) $ t_n(t,nCur) ..
*
*       --- labour for animal activities, expressed per animal and month
*
       v_labHerdM(t,nCur,m) =e= sum(actHerds(sumHerds,breeds,feedRegime,t,m1) $ m_to_herdm(m,m1),
                                    v_herdSize(sumHerds,breeds,feedRegime,t,nCur,m1)
                                        * p_herdLab(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) );
```

Figure 36:

```
*
*   --- labour need of crops, per state of nature of month
*
    labCropSM_(tCur(t),nCur,s,m) $ t_n(t,nCur) ..

       v_labCropSM(t,nCur,s,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,s) * p_cropLab(crops,till,intens,m))

*         --- labour need for application of N (fertilizer and manure N)

$ifthen i %herd% == true
       + sum((c_s_t_i(curCrops(crops),plot,till,intens),manApplicType,manType)
             $ ( v_manDist.up(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m) ne 0),
              v_manDist(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m) * p_manDistLab(ManApplicType))
$endif

       + sum((c_s_t_i(crops,plot,till,intens),syntFertilizer),
              v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m) * p_syntDistLab(syntFertilizer));
```

Figure 37:

operations, *labReqLevl*:

The number of field work 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:

Furthermore, the distribution of tractors is determined endogenously:

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 realized 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 per hour wage of full time work is higher than of working half time one. For half and full time work commuting time can be considered:

The set *workType* lists the possible combinations:

It is assumed that decisions about how much to work flexibly on an hourly basis

```
*
*   --- available field working days per half month, depending on soil type,
*       derived from tractor hours
*
    fieldWorkHours_(plot,labReqLevl,labPeriodSummed,tCur(t),nCur,s)
               $ (p_plotSize(plot) $ plot_landType(plot,"arab") $ t_n(t,nCur) ) ..

       v_fieldWorkHours(plot,labReqLevl,LabPeriodSummed,t,nCur,s)

         =e=
*
       sum(labPeriodSummed_ori(labPeriodSummed,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,s)
                * 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,s,m)
                   * p_machNeed(syntFertilizer,"plough","normal","tractor","hour") ) * sameas(labReqLevl,"rf3")
          ));
```

Figure 38:

```
*
*   --- assume that farm labor is allowed for up to 15 hours a day
*
    tracRestrFieldWorkHours_(plot,labReqLevl,labPeriodSummed,tCur(t),nCur,s)
       $ (p_plotSize(plot) $ plot_landType(plot,"arab") $ t_n(t,nCur)) ..

       v_fieldWorkHours(plot,labReqLevl,LabPeriodSummed,t,nCur,s)

        =L=
             sum(labPeriodSummed_ori(labPeriodSummed,LabPeriod),
               sum(plot_soil(plot,soil),
                     sum(curClimateZone, p_fieldWorkingDays(labReqLevl,labPeriod,curClimateZone,soil)) * 12)
                           * v_tracDist(plot,labPeriodSummed,t,nCur,s));
```

Figure 39:

```
*
*   --- tractor use distribution (in number of tractors)
*       in each labor period per soil
*
    tracDistribution_(labPeriodSummed,tCur(t),nCur,s) $ t_n(t,nCur) ..

       sum(plot $ p_plotSize(plot), v_tracDist(plot,labPeriodSummed,t,nCur,s)) =L= ceil(%Aks%);
```

Figure 40:

```
*
*   --- off farm work binary in yearly hours
*
    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));
```

Figure 41:

```
*
*  --- construct a sequence of half, full, half+full, 2 full, 2 full + half, 3 full ...
*
   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);
```

Figure 42:

are taken on a yearly basis (i.e. the same amount of hours are inputted in each month) and can be adjusted to the state of nature.

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

```
*
*   --- total off farm work (binary and flexigle)
*
    offFarmWorkTot_(tCur(t),nCur) $ t_n(t,nCur) ..
*
       v_labOffTot(t,nCur) =e=
       sum(s, v_labOffHourly(t,nCur,s) * p_prob(s)) * card(m)
     + v_labOffFixed(t,nCur) $ sum(workOpps(workType), v_labOff.up(t,nCur,workType));
```

Figure 43:

Stables

The template applies a vintage based model for different stable types in addition to other buildings and selected machinery, and a physical use based depreciation for the majority of the machinery park. Under the vintage based model stables, other buildings and machinery become unusable after a fixed number of years. In the case of physical depreciation machinery becomes inoperative when its maximum number of operating hours or another measurement of use (e.g. the amount handled) is reached. Investments in stable, buildings and machinery are implemented as binary variables. In order to keep the possible branching trees at an acceptable size, the re-investment points can be restricted to specific years. For longer planning horizon covering several decades, investment could e.g. only be allowed every fourth or fifth year.

The stable inventory, *v_stableInv,* for each type of stable, *stables,* is defined as can be seen in *stableInv_*. *p_iniStables* is the initial endowment of stables by the construction year, *p_lifeTimeS* is the maximal physical life time of the stables and *v_buyStables* are newly constructed stables.

```
*
*   --- inventory of stable according to age (initial and new investments)
*
    stableInv_(stables,t)   ..
         v_stableInv(stables,t) =E=
*
*         --- old stables according to building date and lifetime
*
            sum(tOld $ (     ((p_year(tOld) + p_lifeTimeS(stables)) ge p_year(t))
                        and ( p_year(tOld)                          le p_year(t))),
              p_iniStables(stables,tOld))
*
*         --- plus (old) investments
*
          + sum(t1   $ (     ((p_year(t1)  + p_lifeTimeS(stables) ) ge p_year(t))
                        and ( p_year(t1)                           le p_year(t))),
                              v_buyStables(stables,t1));
```

Figure 44:

For cow stables a differentiation is introduced 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, as defined by the investment horizon set *hor*:

```
set hor "Investment horizons"/
      short   "10 years lifetime"
      middle  "15 years lifetime"
      long    "30 years lifetime"
       /;
```

Figure 45:

A stable can only be used, if short and middle term maintenance investment is done.

The model distinguishes between several stable types for cattle, shown in the following list). They differ in capacity, cattle type, investment cost and labour need per stable place.

```
      set set_cowStables "Stable for dairy cows"  /
                    milk30
                    milk45
                    milk60
                    milk75
                    milk90
                    milk105
                    milk120
                    milk135
                    milk150
                    milk165
                    milk180
                    milk195
                    milk210
                    milk225
                    milk240
      /;
      set set_youngStables "Stable for cattle older than 6 months"
            / youngCattle15
              youngCattle30
              youngCattle45
              youngCattle60
              youngCattle75
              youngCattle90
              youngCattle120
              youngCattle150
              youngCattle180
           /;
      set set_calvStables  "Stable for calves up to 6 months"
           / calves15
             calves30
             calves45
             calves60
             calves75
             calves90
             calves120
             calves150
          /;
$$endif.dh
```

Figure 46:

For pigs the following stable sizes are available:

```
      $$iftheni.pg %pigHerd% == true
*
*          --- pig fattening
*
                    fat400
                    fat500
                    fat800
                   fat1000
                   fat1200
                   fat1500
                   fat1800
                   fat2000

*
*          --- sows and piglets
*
                   sows120
                   sows200
                   sows250
                   sows300
                   sows400
                   sows500
                   piglet500
                   piglet750
                   piglet1000
                   piglet1500
                   piglet2000
          $$endif.pg
```

Figure 47:

The used part of the stable inventory (a fractional variable) must cover the stable place needs for the herd:

```
*
*   --- stable places
*
    stables_(stableTypes,tFull(t),nCur,m)
          $ (sum(actHerds(sumHerds,breeds,feedRegime,t,m)
                  $ 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), 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),   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));
```

Figure 48:

The used part cannot exceed the current inventory (a binary variable):

```
*
*   --- part of stable used must be less then actual inventory
*
    stableUsed_(stables,hor,tFull(t),nCur) $ ( (      sum( t_n(t1,nCur1) $ isNodeBefore(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);
```

Figure 49:

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:

```
*
*   --- share of stable entering variable costs calculated at least 75%
*
    stableCostLo_(stables,hor,tFull(t),nCur)
       $ ((       sum( t_n(t1,nCur1) $ isNodeBefore(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;

*
*   --- share of stable entering variable costs calculated at least as high
*       as actually used part
*
    stableCostUp_(stables,tFull(t),nCur)
        $ (   (   sum( (hor,t_n(t1,nCur1)) $ isNodeBefore(nCur,nCur1), v_buyStables.up(stables,hor,t1,nCur1))
               or sum( (hor,tOld),  p_iniStables(stables,hor,tOld))) $ t_n(t,nCur))  ..

       v_stableShareCost(stables,t,nCur) =G= v_stableUsed(stables,t,nCur);
```

Figure 50:

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:

```
*
*   --- manure silo inventory (not binary), (deprecitation over time)
*
    siloInv_(silos,tCur(t),nCur)
          $ ( (   sum(t_n(t1,nCur1) $ isNodeBefore(nCur,nCur1), v_buySilos.up(silos,t1,nCur1))
                or sum(tOld, p_iniSilos(silos,tOld))) $ t_n(t,nCur) ) ..

       v_siloInv(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(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(silos,t1,nCur1));
```

Figure 51:

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

```
*
*   --- buildings and structures inventory
*
    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));
```

Figure 52:

The buildings included in the model are:

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

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

Farm Machinery

The model includes farm machineries in quite some detail:

For further information see Appendix A1.

```
set s_bunkerSilos /
                    bunkerSilo450
                    bunkerSilo900
                    bunkerSilo1620
                    bunkerSilo2640
                    bunkerSilo3630
                    bunkerSilo4620
                    bunkerSilo8580
                    bunkerSilo11870
                    bunkerSilo26550
                  /;

set buildings  / potaStore500t
                 set.s_bunkerSilos
                /;
```

Figure 53:

```
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
 ;
```

Figure 54:

```
*
*   --- buildings inventory (not binary)
*
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);
```

Figure 55:

```
set machType   /  tractor
                  tractorSmall
                  plough              "Pflug"
                  chiselPlough        "Schwergrubber"
                  sowMachine          "Sämaschine"
                  directSowMachine    "DirektSämaschine"
                  seedBedCombi        "Saatbeetkombination"
                  circHarrow          "Scheibenegge"
```

Figure 56: C:\Users\Kokemohr\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Neues Bild.png

Each machinery type is characterized by set of attributes *p_machAttr* (see *coeffgen\mach.gms*), see as an 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
*
ChiselPlough       5600                2600                         5.0
```

Figure 57:

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                 "Gülleausbringung"
                basFert                 "P und K Duengung, typischerweise Herbst"
                plow                    "Pflügen"
                chiselPlow              "Tiefengrubber"
                seedBedCombi            "Saatbettkombination"
                herb                    "Herbizidmaßnahme"
```

Figure 58: C:\Users\Kokemohr\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Neues Bild.png

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:

```
*  --- attributes for the operations
*
 table op_attr(operation,machVar,plotSize,opAttr)

                                        labTime         diesel      fixCost      varCost   nPers
    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
```

Figure 59:

Furthermore, the model considers the effect of different plot size and the mechanisation level:

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

That detailed information on farm operations determines

```
*
*   --- see page 250 KTBL 2010/2011 for winter cereals
*
*       Describe effect of plot size and mechanisation (= work width) on time, variable and fix
*       machinery costs and diesel.
*
table p_plotSizeEffect(crops,machVar,opAttr,plotSize)

                                1ha    2ha   5ha  20ha

    winterCere. 67kw .labTime  12.4   10.5   9.3   8.0
    winterCere. 67kw .diesel     90     83    78    73
    winterCere. 67kw .varCost   205    188   176   168
    winterCere. 67kw .fixCost   282    258   241   231

    winterCere.102kw .labTime  11.1    9.1   7.6   6.8
    winterCere.102kw .diesel     95     86    78    74
    winterCere.102kw .varCost   209    188   172   164
    winterCere.102kw .fixCost   315    284   262   249

    winterCere.200kw .labTime  11.9    8.6   6.3   4.9
    winterCere.200kw .diesel    118     99    84    75
    winterCere.200kw .varCost   240    201   173   157
    winterCere.200kw .fixCost   396    334   292   267
;

p_plotSizeEffect("winterCere",machVar,"nPers",plotSize) = 1;


*
* -- use mean of given crops if missing for one of the crops
*
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);
```

Figure 60:

```
table crop_op_per_till(crops,operation,labPeriod,till)
                                                          plough     minTill   noTill          eco

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       0.333                 0.333
```

Figure 61:

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

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

```
*
*  --- sum labour hour needs over operations for each month,
*      accounting for effect of mechanisation and plot size
*
   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")   );
```

Figure 62:

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)
*
   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
*
      - sum(s, v_machNeed(machType,machLifeUnit,t,nCur,s) * p_prob(s)) $ tCur(t)
*
*       --- minus operating hours of weighted average over normal planning period
*           if beyond the normal planning period
*
      - [sum( (t_n(t1,nCur1),s) $ ( (p_year(t1) lt p_year(t)) $ tCur(t1) $ isNodeBefore(nCur,nCur1)),
                    v_machNeed(machType,machLifeUnit,t1,nCur1,s)
                                                    * p_prob(s) * 1/(p_year(t)+5 - p_year(t1)) )
        /sum( (t1,s) $ ( (p_year(t1) lt p_year(t)) $ tCur(t1)),     p_prob(s) * 1/(p_year(t)+5 - p_year(t1)) )
         ]
                   $ ( (not tCur(t)) and p_prolongCalc)
   ;
```

Figure 63:

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 the maximum of the need in any state-of-nature in that year:

```
*
*   --- machinery needs of herds
*
    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), v_herdsSize(sumHerds,breeds,feedRegime,t,nCur,m)
               * p_machNeed(sumHerds,"plough","normal",machType,machLifeUnit)
                   * 1/min(12,p_prodLength(sumHerds,breeds)) * 12/card(herdM));

*
*   --- need of machineries per SON, the SON with the highest need
*       defines the machinery park investments
*
    machines_(curMachines(machType),machLifeUnit,tCur(t),nCur,s) $ (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,s)
            * 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,s,m)
               * p_machNeed(syntFertilizer,"plough","normal",machType,machLifeUnit))

$iftheni %herd%==true

*        ---- machine need for the application of N (manure/fertilizer)

     + sum((c_s_t_i(curCrops(crops),plot,till,intens),ManApplicType,manType,m)
              $ (v_manDist.up(crops,plot,till,intens,manApplicType,manType,t,nCur,s,m) ne 0),
                v_manDist(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m)
                  * p_machNeed(ManApplicType,"plough","normal",machType,machLifeUnit))
*
      + v_machNeedHerds(machType,machLifeUnit,t,nCur)
           $ sum(actHerds(sumHerds,breeds,feedRegime,t,m),
                p_machNeed(sumHerds,"plough","normal",machType,machLifeUnit))
$endif
*
*         --- total machinery need
*
              =L= v_machNeed(machType,machLifeUnit,t,nCur,s)
    ;
```

Figure 64:

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:

Those are linked to the existence of stables, i.e. stables cannot be used if machinery is not present:

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 is reflected as well.

The total investment sum *v_sumInv* in each year is defined by:

It can be financed either by equity or by credits, and enters accordingly the cash balance definition, *v_liquid*. The cash balance is the cash at the end of the last

```
*
*   --- inventory of mach according to simple lifetime in years
*
    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));
```

Figure 65:

```
*
*   --- specific machinery (front loader, shear grab, dung grab ... are linked to stables)
*
    machInvStable_(curMachines(machType),stables,tCur(t),nCur) $ ( (v_machInv.up(machType,"years",t,nCur) ne 0)
                                                                   $  (   sum( t_n(t1,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);
```

Figure 66:

```
*
*   --- sum of investments in current year
*
    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(silos, v_buySilos(silos,t,nCur)*p_priceSilo(silos,t))

*
*        --- new biogas plant bought
*
$iftheni %biogas%==true

                  + sum((curBhkw(bhkw), curEeg(eeg)),
                        v_buyBioGasPlant(bhkw,eeg,t,nCur,"ih20") $tCur(t)
                                              * p_priceBioGasPlant(bhkw,"ih20"))

                  + sum((curBhkw(bhkw), curEeg(eeg), ih),
                        v_buyBioGasPlantParts(bhkw,eeg,t,nCur,ih)
                                              * ( p_priceBioGasPlant(bhkw,ih) $ (not(ih20(ih)))
                                                + p_priceFlexBioGasPlant(bhkw,eeg,ih)$eegDM(eeg) ))
$endif;
```

Figure 67:

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*:

```
*
*   --- accumulated liquidity
*
    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))
*
*      -- expected cash flow of enterprise (revenues - cost), mean over SONs
*
          + v_netCashFlow(t,nCur)
*
*      -- new credits
*
$iftheni not "%dynamics%"=="comparative-static" + v_liquidation(nCur) $ sameas(t,"%lastYearCalc%")
          + sum(creditType, v_credits(creditType,t,nCur)) $ (p_year(t) lt p_year("%lastYear%"))
$endif
*
*      -- household expenditures
*
          - p_hcon(t)
*
*      -- investments
*
          - v_sumInv(t,nCur);
```

Figure 68:

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:

```
*
*   --- outstanding credits: past full amounts, paid back with specific payback time
*       (needs to be modified to account for credits with different conditions)
*
    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))));
```

Figure 69:

The net cash flow is defined as the sum of the gross margins in each SON, *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:

Revenues from liquidation are only assumed to take place in the last year (of the farm's life):

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 state-of-nature 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 fertilizer, 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,

```
*
*--- cash flow in current period
*
    netCashFlow_(tFull(t),nCur) $ t_n(t,nCur)  ..
*
      v_netCashFlow(t,nCur) =e=
*
*       --- expected gross margin from cropping and herds plus off farm income plus land rents
*
          sum(s, v_objeTS(t,nCur,s) * p_prob(s)) $ tCur(t)
*
*             --- very small costs to store manure, to avoid stochastic behavior of REFIND
*
$ifi %herd%==true  - sum((manStorage,m), v_volInStorageType(manStorage,t,nCur,m) * 0.01)
*
*       --- interest gained from accumulated liquidity
*           (last year's liuqidity minus new credits)
*
         + sum(t_n(t-1,nCur1) $ anc(nCur,nCur1),
               (v_liquid(t-1,nCur1)-sum(creditType, v_credits(creditType,t-1,nCur1)))
                    * p_interestGain/100)

$iftheni not "%dynamics%"=="comparative-static"
*
*       --- interest paid on outstanding credit sum
*
         - sum(creditType, v_sumCredits(creditType,t,nCur) * p_interest(creditType)/100)

*
*       --- re-payments on past credits
*
         - sum((creditType,t1,nCur1) $ ( (    ((p_year(t1)    + p_payBackTime(creditType))  ge p_year(t))
                                          $    ( p_year(t1)+1                               le p_year(t)))
                                           $ tFull(t1)  $ isNodeBefore(nCur,nCur1) $ t_n(t1,nCur1)  ),
                                               v_credits(creditType,t1,nCur1) * 1/p_payBackTime(creditType))
*
*       --- liquidation revenues of farm
*
         + v_liquidation(nCur) $ sameas(t,"%lastYearCalc%")
$endif
*
*       --- weighted average of past expected gross margins beyond full planning horizon
*           (take into only the last 40 years)
*

        + [   sum( (s,t1,nCur1) $ ( isNodeBefore(nCur,nCur1)
                                    $ (p_year(t) gt p_year(t1))
                                    $ (abs(p_year(t1)-p_year(t)) le 40)
                                    $ tCur(t1) $ t_n(t1,nCur1)),
*
*                 --- discounted past revenues minus costs
*
                  v_objeTS(t1,nCur1,s)  *  (1+p_discountRate/100)**(p_year(t)-p_year(t1))
*
*                 --- weighted with difference to current time time point
*
                        * p_prob(s) * 1/(p_year(t)+5 - p_year(t1)) )
            /sum( (s,t1) $ ( (p_year(t) gt p_year(t1)) $ (abs(p_year(t1)-p_year(t)) le 40) $ tCur(t1)),
                           p_prob(s) * 1/(p_year(t)+5 - p_year(t1)) )
          ]  $ ( (not tCur(t)) and p_prolongCalc)

            ;
```

Figure 70:

```
*
*   --- liquidation revenues of farm
*
    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 %dairyherd%==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),s),
                        sum(m_to_herdm("dec",herdm), v_herdSize("cows",curBreeds,feedRegime,"%lastYear%",nCur,herdm))
                                  * p_price("youngCow","%lastYearCalc%",s) * 0.6 * p_prob(s)) $ p_liquid
*
*               -- heifers at 90% of value of a young cow
*
              + sum( (actHerds("heifs",curBreeds,feedRegime,"%lastYear%",herdm),s),
                        sum(m_to_herdm("dec",herdm), v_herdSize("heifs",curBreeds,feedRegime,"%lastYear%",nCur,herdm))
                                  * p_price("youngCow","%lastYearCalc%",s) * 0.9 * p_prob(s)) $ p_liquid
*
*               -- raising cavles at 30% of value of a young cow
*
              + sum( (actHerds("fCalvsRais",curBreeds,feedRegime,"%lastYear%",herdm),s),
                        sum(m_to_herdm("dec",herdm), v_herdSize("fCalvsRais",curBreeds,feedRegime,"%lastYear%",nCur,herdm))
                                  * p_price("youngCow","%lastYearCalc%",s) * 0.3 * p_prob(s)) $ p_liquid
$endif.dh
       ;
```

Figure 71:

p_weekTime, is given. In addition, it is assumed that off-farm week covers 46 weeks each year, so that income is defined then from multiplying these two terms with hourly wage, *p_wage*.

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 state of nature, *p_price*:

The sale quantity, *v_saleQuant*, plus feed use, *v_feedUse*, must exhaust the production quantity, *v_prods*:

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:

The variable *v_redMlk* allows the farmer to not fully use the genetic potential of the milk cow by adjusting the feed mix. This could be of relevance in the optimization process for instance if the yield potential of different herds are very high, but price combinations of in- and output lead to an economic optimal intensity level that is below the maximum milk yield potential. Without this variable, cows would have always to be milked at the maximum level.

```
*
*   --- revenues - costs for each state of nature from agriculture
*       (net of investments and costs/revenues related to capital/credits)
*
    objETS_(tCur(t),nCur,s)  $ t_n(t,nCur)   ..
*
      v_objETS(t,nCur,s)
                                =e=
*
*        --- income from selling products
*
             + v_salRev(t,nCur,s)
*
*        --- single farm premium
*
             + v_sfPrem(t,nCur,s)
*
*        --- revenues from selling biogas
*
$ifi %biogas%==true  + v_salRevBioGas(t,nCur)
*
*        --- income from renting out land
*
$ifi %landLease% == true + sum( plot, v_rentOutPlot(plot,t,nCur) * p_plotSize(plot) * p_landRent(plot,t))
*
*        --- off farm income, flexible on a hourly basis
*
             + v_labOffHourly(t,nCur,s) *  p_wage("hourly",t,s) * card(m)
$iftheni.sp %stochProg%==true
     * ( 1 + (p_randVar("priceInputs",nCur)-1) $ randProbs("hourly") )
$endif.sp
*
*        --- half/full time work off the farm, decided upon for several years (p_decPeriodInv)
*            depending on length of planning horizon
*
             + sum( workOpps(workType),
                    v_labOff(t,nCur,workType) * p_workTime(workType) * 46 * p_wage(workType,t,s)
$iftheni.sp %stochProg%==true
     * ( 1 + (p_randVar("priceInputs",nCur)-1) $ randProbs(workType) )
$endif.sp
             )
*
*        --- variable cost (not explicitly covered by constraints) minus premiums
*
             - v_varCost(t,nCur,s);
```

Figure 72:

```
        *
        *   --- revenue from sales of animal and crop products in average over states of nature
        *       (SON specific price times SON specific production quantities)
        *
        salRev_(tCur(t),nCur,s) $ t_n(t,nCur)    ..
        *
            v_salRev(t,nCur,s)  =e= sum(  curProds(prodsYearly),
                                          p_price(prodsYearly,t,s)
        $iftheni.sp %stochProg%==true
                * ( 1 + (p_randVar("priceOutputs",nCur)-1) $ randProbs(prodsYearly) )
        $endif.sp
                                       *  v_saleQuant(prodsYearly,t,nCur,s));
```

Figure 73:

```
*
*   --- sale quantity and feed use as well as biogas use is equal to the total amount of produce
*
    saleQuant_(curProds(prodsYearly),tCur(t),nCur,s) $ (p_price(prodsYearly,t,s) $ t_n(t,nCur)) ..

       v_saleQuant(prodsYearly,t,nCur,s)

$iftheni.dh %dairyHerd%==true
     + sum( sameas(prodsYearly,feedsY), v_feedUse(feedsY,t,nCur,s))
$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
        =e= v_prods(prodsYearly,t,nCur,s);
```

Figure 74:

```
*
*   --- quantities of outputs (for those with a yearly resolution)
*
    prods_(prodsYearly,tCur(t),nCur,s) $ (sum(sameas(prodsYearly,curProds),1) $ t_n(t,nCur))   ..

       v_prods(prodsYearly,t,nCur,s)
         =e=
*
*        --- crop output
*
         sum( c_s_t_i(crops,plot,till,intens), v_cropHa(crops,plot,till,intens,t,nCur,s)
             * sum(plot_soil(plot,soil),p_OCoeffC(crops,soil,till,intens,prodsYearly,t)))
*
$iftheni.herd %herd% == true
*
*        --- animal output
*
     +  sum((actherds(possHerds,breeds,feedRegime,t,m)) $ (sum(breed_prods_prods(breeds,prods,prodsYearly),
                                                              p_OCoeff(possHerds,prods,t))
                                                          or p_OCoeff(possHerds,prodsYearly,t)),
*
*            -- herd size in different month times output yearly coefficient (milk, young animals ..)
*
                           (    v_herdSize(possHerds,breeds,feedRegime,t,nCur,m)
                                  * ( 1/min(12,p_prodLength(possHerds,breeds)) * 12/card(herdM))
                                      $ ( (p_prodLength(possHerds,breeds) gt 1)
$ifi "%farmBranchFattners%" == "on"       or ((p_prodLength(possHerds,breeds) le 1 $ actHerds("Fattners","",feedRegime,tCur,m)))
$ifi "%farmBranchSows%" == "on"           and (p_prodLength(possHerds,breeds) gt 2)
                                     )
                              + v_herdStart(possHerds,breeds,feedRegime,t,nCur,m)
                                          $ ( (p_prodLength(possHerds,breeds) le 1)

$ifi "%farmBranchFattners%" == "on"          and (not (actHerds("Fattners","",feedRegime,tCur,m)))
$ifi "%farmBranchSows%" == "on"              or (p_prodLength(possHerds,breeds) le 2)
                                            )
                           )

                  *(
                      sum(breed_prods_prods(breeds,prods,prodsYearly),
                         p_OCoeff(possHerds,prods,t))
                       + p_OCoeff(possHerds,prodsYearly,t)
                   )


$iftheni.dairyHerd %dairyHerd% == true
*            -- reduction of milk yields for specific cows in specific phases
*               (for high yielding phases in the lactation)
*
           - sum( (phase,intrYPer),
                  v_redMlk(possHerds,breeds,phase,intrYPer,t,nCur,s)) $ sameas(prodsYearly,"milk")
$endif.dairyHerd

         )
$endif.herd;
```

Figure 75:

Manure

!!! abstract Manure excretion from animals is calculated 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 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. It is computed by summing the monthly manure for the herd with considering the amount excreted while grazing, shown in the following equation:

Furthermore, the monthly excretion of nutrients, NTAN, Norg and P 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

```
*
*   --- definition of manure excreted in m3 per month
*

     manQuantM_(tCur(t),nCur,m) $ t_n(t,nCur) ..


           v_manQuantM(t,nCur,m) =e=
              sum(actherds(herds,breeds,feedRegime,t,m1),
                  p_manQuantMonth(herds)
                   * sum(m_to_herdm(m,m1), v_herdSize(herds,breeds,feedRegime,t,nCur,m1)))

$iftheni.d %dairyherd%==true

*
*  --- substract manure excreted while grazing
*
         - sum( (possHerds,breeds,crops,plot,till,intens,s)
*                $ ( sum(m $ actHerds(possHerds,breeds,t,m),1)
                   $ (sum( (phase,intryPer),  p_reqs(possHerds,breeds,phase,intrYPer,"DRMX"))
                     $ sum( (c_s_t_i(crops,plot,till,intens),plot_soil(plot,soil)),
                        p_oCoeffC(crops,soil,till,intens,"grasPast",t)),
                         v_grazingHaDays(possHerds,breeds,crops,plot,till,intens,t,nCur,s)
                                 * p_manQuantMonth(possHerds)  * 12/365)
$endif.d
                                                              ;
```

Figure 76:

parameters can be found in *coeffgen\manure.gms* (not shown here). For dairy cows, excretion on pasture is subtracted.

```
* --- Calculation of NTan, NOrg and P excreted by herd; assumption that mother cows are outside whole year

nut2ManureM_(nut2,tCur(t),nCur,m) $ t_n(t,nCur) ..

    v_nut2ManureM(nut2,t,nCur,m) =e=
          sum(actherds(possHerds,breeds,feedRegime,t,m1),
                  p_nut2ManMonth(possHerds,feedRegime,nut2)
                   * sum(m_to_herdm(m,m1), v_herdSize(possHerds,breeds,feedRegime,t,nCur,m1)))


$iftheni.d %dairyherd%==true

*
*        --- substract estimated output during days where herds are grazing
*

                  - v_nut2ManurePast(nut2,t,nCur,m)
$endif.d

;
```

Figure 77:

Biogas production involves the production of digestates. Four sources can be differentiated depending on the origin of the feed crop: use of manure produced on farm, manure imported to the farm, crops grown on farm and crops imported on farm. Manure produced on farm is treated like 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.

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 correspond-

```
*
*   --- Volume of produced digestate
*

     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));
*
*   --- Volume of digested manure which was purchased. Own manure is not accounted
*       in the volume flow but is directly flowing into the silos
*

     biogasVolManDigestate_(tCur(t),nCur,m) $ t_n(t,nCur) ..

        v_volDigMan(t,nCur,m) =E= sum( (curBhkw(bhkw), curEeg(eeg), maM) ,
                                        v_purchManure(bhkw,eeg,maM,t,nCur,m) $ selPurchInputs(maM)   * p_fugMan);
```

Figure 78:

ing 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.

```
*
*  --- Nutrient input from crops - Summarized for the whole year and then divided by all month
*      in order to account for unrealistic fluctuating feeding patterns
*       (DS 02/09/2015)

    nutCropBiogasY_(nut2,tCur(t),nCur) $ t_n(t,nCur) ..

        v_nutCropBiogasY(nut2,t,nCur) =E=
             sum( ( crM(biogasFeedM),m,curBhkw(bhkw), curEeg(eeg) ),
                                        v_usedCropBiogas(bhkw,eeg,crM,t,nCur,m)
                                             * p_nutDigCrop(nut2,crM));

    nutCropBiogasM_(nut2,tCur(t),nCur,m) $ t_n(t,nCur)..

        v_nutCropBiogasM(nut2,t,nCur,m) =E=  v_nutCropBiogasY(nut2,t,nCur) / card(m);

*
* --- Nutrients from purchasing manure
*
       nut2ManurePurch_(nut2,maM,tCur(t),nCur,m) $ t_n(t,nCur)..

         v_nut2ManurePurch(nut2,maM,t,nCur,m)
            =E=    sum ( (curBhkw(bhkw), curEeg(eeg)),
                        v_purchManure(bhkw,eeg,maM,t,nCur,m) * p_nut2manPurch(nut2,maM)  )    ;

*
*        steady nutrient pool for biogas plant [NOT USED]
```

Figure 79:

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 *manure_module.gms* is activated when fattners, 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.

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 subtracted. When environmental accounting is switched off, standard loss factors are subtracted directly in the equation.

```
*
*   --- Defines manure amount in m3 per month in each storage type
*
    volInStorage_(tCur(t),nCur,m) $ t_n(t,nCur) ..

       v_volInStorage(t,nCur,m) =e= [sum(t_n(t-1,nCur1) $ anc(nCur,nCur1),  v_volInStorage(t-1,nCur1,"Dec")) $ (sameas(m,"Jan") $ tCur(t-1))
                                  + v_volInStorage(t,nCur,m-1)      $ (not sameas(m,"Jan"))]

$iftheni.herd %herd% == true
*
*                                 --- m3 excreted per year divied by # of month: monthly inflow
*
                                  + v_manQuantM(t,nCur,m)

$endif.herd

*
$iftheni.b %biogas% == true
*                                 --- m3 coming from biogas plant s energy crops and purchased manure

                                  --- Diogas digestate based on energy crops

                                  +  sum(crm(biogasfeedM), v_volDigCrop(crM,t,nCur,m))

*                                 --- Biogas digestate based on manure

                                  +  v_volDigMan(t,nCur,m)
$endif.b
*
*
*                                 --- m3 taken out of storage type for application to crops

                                  - v_volManApplied(t,nCur,m)

$iftheni.ExMan %AllowManureExport%==true
*                                 --- m3 exported from farm

                                  - sum (curManType, v_manExport(curManType,t,nCur,m))

$endif.ExMan

$iftheni.emissionRight not "%emissionRight%"==0
*                                 --- m3 exported through manure emission rights

                                  - sum (curManType, v_manExportMER(curManType,t,nCur,m))

$endif.emissionRight
                           ;
```

Figure 80:

```
*
* --- NTAN, NORG and P in storage
*
     nutPoolInStorage_(nut2,tCur(t),nCur,m) $ t_n(t,nCur) ..

           v_nutPoolInStorage(nut2,t,nCur,m)

             =e=  [sum(t_n(t-1,nCur1) $ anc(nCur,nCur1),    v_nutPoolInStorage(nut2,t-1,nCur1,"Dec")) $ (sameas(m,"Jan") $ tCur(t-1))
                                                          + v_nutPoolInStorage(nut2,t,nCur,m-1)       $ (not sameas(m,"Jan"))]

$iftheni.herd %herd% == true
                + v_nut2ManureM(nut2,t,nCur,m)
$endif.herd
$iftheni.biogas %biogas% == true
                +  sum( (curBhkw(bhkw),curEeg(eeg),maM),   v_nut2ManurePurch(nut2,maM,t,nCur,m)  )
                +  v_nutCropBiogasM(nut2,t,nCur,m)
$endif.biogas

$iftheni.envAcc %envAcc% == true
* --- When environmental accounting is switched on, storage losses are calculated in envir_acc_module.gms
               - v_nutLossInStorage(nut2,t,nCur,m)
$else.envAcc
* --- When environmental accounting is switched off, only standard losses for NH3 are substracted from NTAN
               $$iftheni.herd %herd% ==true
                        - v_nut2ManureM(nut2,t,nCur,m) * (1 -  p_nutEffectivDueVal)   $ sameas(nut2,"NTAN")
               $$endif.herd
               $$iftheni.biogas %biogas% == true
                       -  sum( (curBhkw(bhkw),curEeg(eeg),maM),   v_nut2ManurePurch(nut2,maM,t,nCur,m)
                           * (1 -  p_nutEffectivDueValBiogasPurchMan(maM)  )) $ sameas(nut2,"NTAN")
                       -  v_nutCropBiogasM(nut2,t,nCur,m)  * (1 -  p_nutEffectivDueValBiogasPlantDig)   $ sameas(nut2,"NTAN")
               $$endif.biogas
$endif.envAcc
* --- Nutrients applied
                - v_nut2ManApplied(nut2,t,nCur,m)
* --- Nutrients exported from farm
$iftheni.ExMan %AllowManureExport%==true
                - v_nut2export(nut2,t,nCur,m)
$endif.ExMan

$iftheni.emissionRight not "%emissionRight%"==0
* --- Nutrient exported via manure emission rights
               -  v_nut2ExportMER(nut2,t,nCur,m)
$endif.emissionRight
 ;
```

Figure 81:

When environmental accounting is switched on, losses are calculated in the equation *nutLossInStorage_*, using emission factors from the environmental impact accounting module (see chapter 2.12).

```
*
* --- Calculation of nutrient losses from storage, when environmental impact accounting is switched on
*
$iftheni.envAcc %envAcc% == true
    nutLossInStorage_(nut2,tCur(t),nCur,m) $ t_n(t,nCur) ..
       v_nutLossInStorage(nut2,t,nCur,m) =E=
* --- NH3 losses in stable and storage, only related to N TAN
                  + v_emStabSto("NH3",t,nCur,m) $ sameas(nut2,"NTAN")
            $$iftheni.herd %herd% == true
* --- N2O, N2 and NO losses in stable and storage, related to NTAN and Norg
                  + v_nut2ManureM("NTAN",t,nCur,m) * ( p_EFStaSto("N2O") +  p_EFStaSto("NOx") + p_EFStaSto("N2") )   $ sameas(nut2,"NTAN")
                  + v_nut2ManureM("NOrg",t,nCur,m) * ( p_EFStaSto("N2O") +  p_EFStaSto("NOx") + p_EFStaSto("N2") )   $ sameas(nut2,"NOrg")
            $$endif.herd
* --- N2O, N2 and NO losses from storage from digestate, related to NTAN and Norg
            $$iftheni.biogas %biogas% == true
                  + [ ( v_nutCropBiogasM("NTAN",t,nCur,m)   + sum (maM, v_nut2ManurePurch("NTAN",maM,t,nCur,m)  ) )
                                    * p_EFStaSto("N2O")   * p_EFStaSto("NOx")  * p_EFStaSto("N2") ]  $ sameas(nut2,"NTAN")
                  + [ ( v_nutCropBiogasM("NOrg",t,nCur,m)   + sum (maM, v_nut2ManurePurch("NOrg",maM,t,nCur,m) ) )
                                    * p_EFStaSto("N2O")   * p_EFStaSto("NOx")  * p_EFStaSto("N2") ]  $ sameas(nut2,"NOrg")
             $$endif.biogas
       ;
$endif.envAcc
```

Figure 82:

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.

```
*
*    --- Total manure storage capacity of overall farm in m^3
*
     totalManStorCap_(tCur(T),nCur) $ t_n(t,nCur) ..
        v_TotalManStorCap(t,nCur) =e=


$iftheni.herd %herd%  == true
                              v_SubManStorCap(t,nCur)
                              + v_SiloManStorCap(t,nCur)
$endif.herd

$ifi %biogas% == true         + v_siloBiogasStorCap(t,nCur)
     ;
```

Figure 83:

The storage capacity of silos *v_SiloManStorCap* is derived by multiplying the silo inventory with parameters characterizing the corresponding storage capacity.

The subfloor storage capacity of stables *v_SubManStorCap* is calculated in the

```
*
*   --- Storage capacity for manure in outdoor silo systems
*
    siloManStorCap_(tCur(t),nCur) $ t_n(t,nCur) ..

       v_SiloManStorCap(t,nCur) =e= sum(silos $ (      sum(t_n(t1,nCur1) $ isNodeBefore(nCur,nCur1), v_buySilos.up(silos,t1,nCur1))
                                                  or (sum(tOld, p_iniSilos(silos,tOld)))),
                                     v_SiloInv(silos,t,nCur)*p_ManStorCapSi(silos))    ;
```

Figure 84:

general_herd_module.gms. The stable inventory is multiplied with parameters characterizing 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 code, *p_ManStorCapSi*, from 500 to 4000 m³.

```
*
*   --- Storage capacity for manure subfloor in stable systems
*
    subManStorCap_(tCur(t),nCur) $ t_n(t,nCur) ..

       v_SubManStorCap(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(stables));
*
```

Figure 85:

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*.

```
*
*   ---  Silo inventory is pegged to the biogas plant investment
*
    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));
```

Figure 86:

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

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.

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. The necessary silo capacity can be set (1) to 6 month, i.e. 50 % of annual manure excretion, or to (2) amount of month when manure

```
*
*   --- defines how manure amount is distributed to the single storage types
*
    storageDistr_(tCur(t),nCur,m) $ t_n(t,nCur) ..

       v_volInStorage(t,nCur,m) =e= sum (manStorage,v_volInStorageType(ManStorage,t,nCur,m));
```

Figure 87:

```
*
*   --- Declaration that the model knows what coverage type is on the manure silos
*
    siloCoverInv_(silos,tCur(t),nCur)
       $ ( (    sum(t_n(t1,nCur1) $ isNodeBefore(nCur,nCur1), v_buySilos.up(silos,t1,nCur1))
              or sum(tOld, p_iniSilos(silos,tOld))) $ t_n(t,nCur)) ..

       v_siloInv(silos,t,nCur) =e=  sum(siloCover, v_siCovComb(silos,t,nCur,siloCover));
```

Figure 88:

application is forbidden in winter. For (2), there is a differentiation for arable land and grassland.

```
*
*  --- Different ways to define required manure storage capacity, chosen via interface
*
$iftheni.sixM %manureStorageRequired% == sixMonth
*  --- Manure storage capacity must cover at least 50% of annual manure quantity excreted following: JGA AnlagenVO
    manStorCapNeed_(tCur(t),nCur) $ t_n(t,nCur) ..
       v_ManStorCapNeed(t,nCur) =e= 0.5 * (
$ifi %herd% == true                  v_manQuant(t,nCur)
*  --- 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))
                                     );
$endif.sixM
$iftheni.blP %manureStorageRequired% ==  blockingPeriod
*  --- Manure storage capacity equals month of periods when fertilizer application is forbidden,
*      related to arable land when arable/arable + grass is avaialbe, otherwise to grass land [TK 16/06/15]
$iftheni.arab %Arable% == true
    manStorCapNeed_(tCur(t),nCur) $ t_n(t,nCur) ..
      v_ManStorCapNeed(t,nCur)   =e= card(monthApplicationForbiddenArab)/12 *  (
$ifi %herd% == true                    v_manQuant(t,nCur)
*  --- 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))
                                       );
$else.arab
       v_ManStorCapNeed(t,nCur)   =e= card(monthApplicationForbiddenGrass)/12 * (
$ifi %herd% == true                  v_manQuant(t,nCur)
*  --- 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,crm,nCur,m) + v_volDigMan(t,nCur,m))
                                     );
$endif.arab
$endif.blP
```

Figure 89:

The total manure storage capacity *v_TotalManStorCap* must be greater than the required storage capacity *v_ManStorCapNeed*.

Besides requirement with regard to 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 storage can be

```
*
*   --- Total manure storage capacity has to be greater than required storage capactiy ManStorCapNeed(t)
*
    manStorCap_(tCur(t),nCur) $ t_n(t,nCur) ..
       v_TotalManStorCap(t,nCur) =g= v_ManStorCapNeed(t,nCur);
```

Figure 90:

emptied when relation between nutrients and volume changes due to nutrient losses during storage (see chapter 2.9.3).

```
*          ..                                ..
*   --- Each year in may the manure storage has to be emptied
*
    emptyStorageVol_(tCur(t),nCur,m) $(sameas(m,"apr") $ t_n(t,nCur)) ..
             v_volInStorage(t,nCur,m) =L= 0;

    emptyStorageNut_(nut2,tCur(t),nCur,m) $(sameas(m,"apr") $ t_n(t,nCur)) ..

            v_nutPoolInStorage(nut2,t,nCur,m) =L= 0;
```

Figure 91:

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.

```
*
*   --- Specific equations for last year: not more than 4/12 of total excreted manure
*       in last year in storage. Currently only for dairy herd, as the ratios for nutritions
*       forbid the appropriate emptying of the manure storage
*
    maxManVolStorLastMonth_("%lastYear%",nCur,"Dec") $ t_n("%lastYear%",nCur)  ..
                        (
$ifi %herd% == true         v_manQuant("%lastYearCalc%",nCur)
$ifi %biogas% == true     + sum((crm(biogasFeedM),m), v_voldigCrop(crM,"%lastYearCalc%",nCur,m)+ v_voldigMan("%lastYearCalc%",nCur,m) )
                         ) * 8/12
                       =G=  v_volInStorage("%lastYear%",nCur,"Dec");

    maxManNutStorLastMonth_(nut2,"%lastYear%",nCur,"Dec") $ t_n("%lastYear%",nCur) ..
                        (
$ifi %herd% ==true          sum(m, v_nut2ManureM(nut2,"%lastYear%",nCur,m))
$ifi %biogas% == true     + sum((maM,m), + v_nut2ManurePurch(nut2,maM,"%lastYearCalc%",nCur,m))
                         ) * 8/12
                      =G=  v_nutPoolInStorage(nut2,"%lastYear%",nCur,"Dec");
```

Figure 92:

Manure Application

Different application procedures for manure N are implemented, *ManApplicType*, including broad spread, drag hose spreader and injection of manure. The core variable is *v_mandist* that represents the amount of manure in 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. Relevant parameters are calculated in *coeffgen\manure.gms*. There are two approaches to calculate the parameter, (1) environmental accounting not activated and (2) environmental accounting activated.

For both systems, the amount of different nutrients per cubic meter is calculated without losses, *p_nut2inManNoLoss*. In the following equation, *p_nut2inMan* is calculated when the environmental accounting is not active. In this case, only a fixed factor for NH3 emissions is subtracted from NTAN contained in the manure. The parameter is calculated for the types of manure from different herds, *mantype*. The herds and types of manure are liked via the cross set *herds_mantype*. This calculation implies that there is one type of manure for every herd activated in FARMDYN.

```
p_nut2inMan("NTAN",manType)          =  sum(herds_manType(herds,manType), p_nut2inManNoLoss("NTAN","normFeed",manType) *  p_nutEffectivDueVal  ) ;
p_nut2inMan("NOrg",manType)          =  sum(herds_manType(herds,manType),  p_nut2inManNoLoss("NOrg","normFeed",manType) ) ;
p_nut2inMan("P",manType)             =  sum(herds_manType(herds,manType), p_nut2inManNoLoss("P","normFeed",manType) ) ;
```

Figure 93:

If environmental accounting is active, the calculation of *p_nut2inMan* differs. It is taken into account, the storage time of manure varies and, therefore, losses during storage vary. 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.

```
* --- Calculation of NTAN, Norg and P with MAXIMUM losses and MINIMUM nutrient content
   p_nut2inMan("NTAN",manType)  $ manuretype_nutConMan("low",mantype)
                                =  p_nut2inManNoLoss("NTAN","normFeed",mantype) * (1 - p_lossFactorSto("low",mantype,"NTAN")  );
   p_nut2inMan("NOrg",manType)  $ manuretype_nutConMan("low",mantype)
                                =  p_nut2inManNoLoss("NOrg","normFeed",mantype) * (1 -  p_lossFactorSto("low",mantype,"Norg") );
   p_nut2inMan("P",manType)     $ manuretype_nutConMan("low",mantype)
                                =  p_nut2inManNoLoss("P","normFeed",mantype);
* --- Calculation of NTAN, Norg snd P with MINIMUM losses and MAXIMUM nutrient content
   p_nut2inMan("NTAN",manType) $ manuretype_nutConMan("high",mantype)
                                =  p_nut2inManNoLoss("NTAN","normFeed",manType) * (1 - p_lossFactorSto("high",mantype,"NTAN")  );
   p_nut2inMan("NOrg",manType) $ manuretype_nutConMan("high",mantype)
                                =  p_nut2inManNoLoss("NOrg","normFeed",manType) * (1 -  p_lossFactorSto("high",mantype,"Norg") );
   p_nut2inMan("P",manType)    $ manuretype_nutConMan("high",mantype)
                                =  p_nut2inManNoLoss("P","normFeed",manType);
```

Figure 94:

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

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 month,

```
nut2ManApplied_(nut2,tCur(t),nCur,m) $ ((v_volManApplied.up(t,nCur,m) ne 0) $ t_n(t,nCur)) ..

    v_nut2ManApplied(nut2,t,nCur,m) =e= sum( (c_s_t_i(curCrops(crops),plot,till,intens),ManApplicType,manType,s)
                                        $ (v_manDist.up(crops,plot,till,intens,manApplicType,manType,t,nCur,s,m) ne 0),

                                          v_manDist(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m)
                                              * p_nut2inMan(nut2,manType) * p_prob(s));
```

Figure 95:

```
*
*   --- manure application in m3  to crops
*
    volManApplied_(tCur(t),nCur,m) $ ((v_volManApplied.up(t,nCur,m) ne 0) $ t_n(t,nCur)) ..

       v_volManApplied(t,nCur,m) =e= sum( (c_s_t_i(curCrops(crops),plot,till,intens),ManApplicType,manType,s)
                                        $ (v_manDist.up(crops,plot,till,intens,manApplicType,manType,t,nCur,s,m) ne 0),
                                          v_manDist(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m) * p_prob(s));
```

Figure 96:

e.g. in maize at certain height of growth.

```
* --- no application to cereals between May and August

  set doNotApplyManure(crops,m) / summerCere    .(May,Jun,Jul,Aug)
                                  winterCere    .(May,Jun)
                                  winterRape    .(May,Jun,Jul,Aug)
                                  potatoes      .(May,Jun,Jul,Aug)

                                  maizSil.(Apr,May,Jun,Jul,Aug,Sep)

$ifi %dairyherd% == true          past33.(Mar,Apr,Jun,Jul,Oct)

                                /;


   v_manDist.up(crops,plot,till,intens, manApplicType,manType,t,n,s,m) $ (t_n(t,n) $ doNotApplyManure(crops,m) $ c_s_t_i(crops,plot,till,intens)) = 0;

* --- on certain crops/in certain month the following manure application techniques cannot be used

   v_manDist.up(arableCrops(crops),plot,till,intens, "applTShoe",manType,t,n,s,m) $ (t_n(t,n) $ c_s_t_i(crops,plot,till,intens)) = 0 ;
$ifi %dairyherd% == true   v_manDist.up(grassCrops(crops),plot,till,intens, "applTShoe",manType,t,n,s,m) $ (t_n(t,n) $ c_s_t_i(crops,plot,till,intens)) = 0 ;

* --- application of manure per injection not possible for maize silage

   v_manDist.up("maizSil",plot,till,intens,"applInjec",manType,t,n,s,m)
      $ (t_n(t,n) $  (not (sameas(m,"Mar") or sameas(m,"Oct"))) $ c_s_t_i("maizSil",plot,till,intens)) = 0;

* --- no manure on idling land

   v_manDist.up("idle",plot,till,intens, manApplicType,manType,t,n,s,m) $ (t_n(t,n) $ c_s_t_i("idle",plot,till,intens)) = 0;
$ifi %dairyherd% == true   v_manDist.up("idlegras",plot,till,intens, manApplicType,manType,t,n,s,m) $ (t_n(t,n) $ c_s_t_i("idlegras",plot,till,intens)) = 0;
```

Figure 97:

There are restrictions with the timing and the quantity of applied manure coming from the German fertilizer directive. Generally, the application of manure is forbidden during winter. Depending on settings in the GUI, manure application can be forbidden only three month during winter or more restrictive regulations apply.

Furthermore, there is the option to ban certain manure application techniques to represent technical requirements given by the German fertilizer directive. These requirements can be activated in the GUI. The sets *tNotLowAppA(t)* and *tLowAppA(t)* represents the years with certain technical requirements.

In the German fertilizer directive, the total amount of Nitrogen from manure is restricted to 170 kilogram N per ha and year in farm average. For grassland, there is the possibility to apply 230 kilogram. The restrictions can be switched on or off in the GUI. In the following equations, *v_DueVOrgN* represent the amount of organic nutrients excreted by animals minus nutrient exports from the farm. *v_nutExcrDueV* is calculated in the *general_herd_module.gms and v_nutBiogasDueV* in the *biogas_module.gms* (equations not shown here).

```
$iftheni.appS %blockingPeriodManureAppl% == short

*
* --- manure application is only forbidden in winter month, enables autumn application, equal to old iimplementation [TK 15/06/15]
*

  set monthApplicationForbidden(m) / Nov, Dec, Jan/;

   v_volManApplied.up(t,n,monthApplicationForbidden) $ t_n(t,n) = 0;
   v_nut2ManApplied.up(nut2,t,n,monthApplicationForbidden) $ t_n(t,n) = 0;
   v_manDist.up(crops,plot,till,intens,manApplicType,manType,t,n,s,monthApplicationForbidden) $ (t_n(t,n) $ c_s_t_i(crops,plot,till,intens)) = 0;

$endif.appS


$iftheni.appL %blockingPeriodManureAppl% == long

*
* --- manure application is restricted in autumn [TK 15/06/15]
*

   set monthApplicationForbidden(m) /Nov,Dec,Jan/;

   v_volManApplied.up(t,n,monthApplicationForbidden) $ t_n(t,n)  = 0;
   v_nut2ManApplied.up(nut2,t,n,monthApplicationForbidden) $ t_n(t,n) = 0;
   v_manDist.up(crops,plot,till,intens,manApplicType,manType,t,n,s,monthApplicationForbidden)
       $ (t_n(t,n) $  c_s_t_i(crops,plot,till,intens)) = 0;

* --- restriction on arable land; manure application is also restricted after the harvest of main crop

   v_manDist.up(arabCrops(crops),plot,till,intens,manApplicType,manType,t,n,s,monthApplicationForbiddenArab)
    $ (t_n(t,n) $ c_s_t_i(crops,plot,till,intens)) = 0;

* --- restriction on grasland

   v_manDist.up(arabCrops(crops),plot,till,intens,manApplicType,manType,t,n,s,monthApplicationForbiddenGrass)
    $ (t_n(t,n) $ c_s_t_i(crops,plot,till,intens)) = 0;


$endif.appL
```

Figure 98:

```
$iftheni.duev %duev%==true
$iftheni.appl %LowNH3Applobligatory% == true

   v_manDist.up(arabCrops(crops),plot,till,intens,"applInjec",manType,tNotLowAppA(t),nCur,s,m) $ t_n(t,Ncur) = 0   ;
   v_manDist.up(arabCrops(crops),plot,till,intens,"applTailh",manType,tNotLowAppA(t),nCur,s,m) $ t_n(t,Ncur) = 0   ;
   v_manDist.up(arabCrops(crops),plot,till,intens,"applTShoe",manType,tNotLowAppA(t),nCur,s,m) $ t_n(t,Ncur) = 0   ;


   $$iftheni.dh %dairyherd%==true

   v_manDist.up(grassCrops(crops),plot,till,intens,"applInjec",manType,tNotLowAppG(t),nCur,s,m) $ t_n(t,Ncur) = 0   ;
   v_manDist.up(grassCrops(crops),plot,till,intens,"applTailh",manType,tNotLowAppG(t),nCur,s,m) $ t_n(t,Ncur) = 0   ;
   v_manDist.up(grassCrops(crops),plot,till,intens,"applTShoe",manType,tNotLowAppG(t),nCur,s,m) $ t_n(t,Ncur) = 0   ;

   $$endif.dh

*   --- use of broadcast spreader is forbidden on grassland and arable land, respectively starting in certain year

   v_manDist.up(arabCrops(crops),plot,till,intens,"applSpread",manType,tLowAppA(t),nCur,s,m) $ t_n(t,Ncur) = 0 ;

   $$iftheni.dh %dairyherd%==true

   v_manDist.up(grassCrops(crops),plot,till,intens,"applSpread",manType,tLowAppG(t),nCur,s,m) $ t_n(t,Ncur) = 0 ;

   $$endif.dh

*  --- Force use of broadcast spreader

$iftheni.onlyB %onlyBroadcast% == true

   v_manDist.up(arabCrops(crops),plot,till,intens,"applInjec",manType,tCur(t),nCur,s,m)  $ t_n(t,Ncur) = 0   ;
   v_manDist.up(arabCrops(crops),plot,till,intens,"applTailh",manType,tCur(t),nCur,s,m)  $ t_n(t,Ncur) = 0   ;
   v_manDist.up(arabCrops(crops),plot,till,intens,"applTShoe",manType,tCur(t),nCur,s,m)  $ t_n(t,Ncur) = 0   ;
   v_manDist.up(grassCrops(crops),plot,till,intens,"applInjec",manType,tCur(t),nCur,s,m) $ t_n(t,Ncur) = 0   ;
   v_manDist.up(grassCrops(crops),plot,till,intens,"applTailh",manType,tCur(t),nCur,s,m) $ t_n(t,Ncur) = 0   ;
   v_manDist.up(grassCrops(crops),plot,till,intens,"applTShoe",manType,tCur(t),nCur,s,m) $ t_n(t,Ncur) = 0   ;

$endif.onlyB

$endif.appl
$endif.duev
```

Figure 99:

```
*
*   --- Calculation of organic N that is limited by Duev 2006 (Ausbringungsobergrenze)[TK 21.11.13]
*
    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
*   --- nutrients exported from farm
    $$iftheni.ExMan %AllowManureExport%==true
          -  sum( (m,nut2) $(not sameas (nut2,"P")), v_nut2export(nut2,t,nCur,m) )
    $$endif.ExMan
    $$iftheni.emissionRight not "%emissionRight%"==0

          -  sum( (m,nut2) $(not sameas (nut2,"P")), v_nut2exportMER(nut2,t,nCur,m) )
    $$endif.emissionRight
*  --- nutrients coming from biogas plant (including energy crops and purchased manure)
    $$iftheni.b %biogas% == true
          + v_nutBiogasDuev("N",t,nCur)   *  p_nutEffectivDueValBiogas
    $$endif.b
;
```

Figure 100:

The applied N is not allowed to exceed the given threshold, *p_nutManApplLimit*, as stated in the following equation. In current legislation, organic N from digestates of plant origin is excluded in the calculation of this threshold. Note that the organic N application according to the fertilizer directive is always calculated in FARMDYN. The restrictive threshold can be switched on and off in the GUI.

```
*
*    --- Restriction that applied organic N per ha does not exceed limits given by DüV
*
    DuevOrgNLimit_ (tCur(t),nCur) $ t_n(t,nCur) ..
       v_DueVOrgN (t,nCur)
*  --- option to include digestates from plant origin
 $$iftheni.bio %biogas% == true
     $$iftheni.dig %includeDigestatesPlantOrigin% == false

          -       sum( (m,nut2) $ (not sameas (nut2,"P")),
                        v_nutCropBiogasM(nut2,t,nCur,m)  )  *  p_nutEffectivDueValBiogas

     $$endif.dig
  $$endif.bio
            =L=
             sum((c_s_t_i(curCrops(crops),plot,till,intens),s), p_nutManApplLimit(crops,t)
                    * v_cropHa(crops,plot,till,intens,t,nCur,s) * p_prob(s)) ;
```

Figure 101:

Synthetic Fertilizers

To meet the N and P demand of crops, synthetic fertilizer can be applied besides manure, *v_syntDist*. Synthetic fertilizer 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 fertilizer are applied for certain crops. It represents the limitation meeting the plant need with nutrients from manure, e.g. fertilizing short before harvest for baking wheat cannot be done with manure.

```
*
*   --- minimum share of mineral fertilizer restiction
*
    nMinMin_(c_s_t_i(curCrops(crops),plot,till,intens),nut,tCur(t),nCur,s)
        $ (  (v_cropHa.up(crops,plot,till,intens,t,nCur,s) ne 0)
                $ p_minShareMin(crops,nut) $ t_n(t,nCur) ) ..

       sum ((syntFertilizer,m),
                      v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,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,s) * p_minShareMin(crops,nut);
```

Figure 102:

Plant Nutrition

The equations related to the 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. Needed nutrients are provided by manure and synthetic fertilizer. There are two approaches to model the nutrient need and supply of crops, a fixed factor approach and detailed nutrient fate model.

Calculation of plant need

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

1. A **fixed factor approach** with yearly nutrient balances per crop
    a. Using N response curves
    b. Using planning data
2. A detailed **flow model** with a monthly resolution by soil depth (deprecated).

c. Using N response curves

p_nutNeed is the nutrient need for different crops and 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 needs to be calculated based on N response curves. All relevant calculation can be found in *coeffgen\cropping.gms*.

N response curves

The yield level of different crops is chosen in the GUI. The following equations show, at 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))
              =  p_cropYieldInt("winterCere","yield")   * (1.00 + p_cropYieldInt("winterCere","GrowthRateY")/100) **t.pos;
```

Figure 103:

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

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

Figure 104:

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 fertilizer 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 fertilizer application.

```
* ---  definition on different crop intensities based on N fertilizer level (100% to 20%)

    p_oCoeffc(arabCrops,soil,till,"fert80p",prods,t)    = p_oCoeffc(arabCrops,soil,till,"normal",prods,t) * 0.96;
    p_oCoeffc(arabCrops,soil,till,"fert60p",prods,t)    = p_oCoeffc(arabCrops,soil,till,"normal",prods,t) * 0.90;
    p_oCoeffc(arabCrops,soil,till,"fert40p",prods,t)    = p_oCoeffc(arabCrops,soil,till,"normal",prods,t) * 0.82;
    p_oCoeffc(arabCrops,soil,till,"fert20p",prods,t)    = p_oCoeffc(arabCrops,soil,till,"normal",prods,t) * 0.73;

    p_oCoeffc(arabCrops,soil,till,"fert80p",prods,t)    = p_oCoeffc(arabCrops,soil,till,"normal",prods,t) * 0.95;
    p_oCoeffc(arabCrops,soil,till,"fert60p",prods,t)    = p_oCoeffc(arabCrops,soil,till,"normal",prods,t) * 0.85;
    p_oCoeffc(arabCrops,soil,till,"fert40p",prods,t)    = p_oCoeffc(arabCrops,soil,till,"normal",prods,t) * 0.71;
    p_oCoeffc(arabCrops,soil,till,"fert20p",prods,t)    = p_oCoeffc(arabCrops,soil,till,"normal",prods,t) * 0.53;
```

Figure 105:

The output coefficients, *p_OCoeffC*, represents the yields per hectar. They are 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 fertilizer directive (DüV 2007, Appendix 1).

```
*
*  --- nutrient need, taking into that output coefficient are measured in t and not dt, therefore * 10
*
 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));
```

Figure 106:

For different intensities, the corresponding amount of nutrient applied has to be determined to fulfil the need *p_nutNeed*.

The parameter *p_basNut* defines the amount of nutrients coming from other sources than directly applied fertilizer. 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 (mineralisation, atmospheric deposition):

```
*
*  -- calculate maximal "background" deliveries over different fertilizing intensities, taking into account
*     realized yield under different intensities and the applied fertilizer (at given loss rates)
*
 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")) );
```

Figure 107:

The amount of nutrient 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 * 2;
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 * 3;

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 * 4;

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 * 5;
```

Figure 108:

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 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 fertilizer management.

```
*
*  --- calculation of unavoidable losses as a share of nutrient need of plant that has to be applied over plant removal (Mehrbedarfskoeffizient)
*
   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),
            (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);
```

Figure 109:

Planning Data

The nutrient need can also derived from planning data from the revised Fertilizer directive (BMEL 2015). The proposed directive includes compulsory fertilizer planning to increase N use efficiency on farms. This measure is included in FARMDYN. When fertilizer 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("winterCere",soil,till,intens,"winterCere",t)
              $ sum(soil_plot(soil,plot),c_s_t_i("winterCere",plot,till,intens))       =  8   ;
```

Figure 110:

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

```
p_nutNeed("winterCere",soil,till,intens,"N",t)
          $ sum(soil_plot(soil,plot), c_s_t_i("winterCere",plot,till,intens))   =   250 ;
```

Figure 111:

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

The directive prescribes that nutrients delivered from soil and air have to be taken into account. This reduces the amount of fertilizer that needs to be applied. The parameter, *p_basNut*, enters the Standard Nutrient Fate Model (see chapter 2.11.2). We assume a fixed amount of 30 kg N per hectar and year for every crop.

The directive does not allow applying more nutrients than required following the planning data. Unavoidable losses are already reflected in *p_nutNeed*. In the GUI, a factor for *over fertilization* can be activated. In this case, the N need for plant increases. This feature can be used to assess the impact of inefficient N management of farms.

Furthermore, the directive prescribes the share of N from organic sources that has to be accounted for in the fertilizing management. The requirements of the

```
*  --- P need is calculated based on P removal by 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));
```

Figure 112:

```
p_basNut(crops,soil,till,"P",t)   = 0;
p_basNut(crops,soil,till,"N",t)   = 30;
```

Figure 113:

directive can be activated in the GUI and enter the Standard Nutrient Fate Model (see chapter 2.11.2).

Standard Nutrient Fate Model

The standard nutrient fate model defines the necessary fertilizer applications based on yearly nutrient balances for each crop category, *NutBalCrop_*. In the equation below, the left hand side defines the nutrient need plus the application of manure over plant need. The right hand side captures the net deliveries from mineral and manure application plus deliveries from soil and air.

FARMDYN allows different ways to account for N from manure. Organic N can be accounted for based on (1) requirements from the Fertilizer Directive, (2) a given factor by the interface and (3) exogenous calculated losses. Losses are calculated using the environmental accounting module (see chapter 2.12). If the environmental accounting module is switched off, calculated losses are derived using standard loss factors from the Fertilizer directive. Different elements of the equation *NutBalCrop_* are explained below.

The crop need is derived from *p_nutneed*. In the case of using N response functions, the needed nutrients increase by unavoidable losses, *p_nutLossUnavoidable*. In the case of using planning data, unavoidable losses are already included in *p_nutNeed* and, therefore, *p_nutLossUnavoidable* is set to 0 (see chapter 2.11.1.2).

When activated in the GUI, more organic N and P than needed for plant nutrition can be applied.

```
*  --- Definition of the application of nutrients over plant need: assumed to be 0 for P, defined for N over interface in %
    p_nutLossUnavoidable(soil,till,intens,"P")   = 0 ;
    p_nutLossUnavoidable(soil,till,intens,"N")   = %NOverNeedValue% ;
```

Figure 114:

```
*
*   --- definition of N and P balance for crops nutrition [TK 03.02.15 revised]
*

   NutBalCrop_(c_s_t_i(curCrops(crops),plot,till,intens),nut,tCur(t),nCur,s)
       $ ((v_cropHa.up(crops,plot,till,intens,t,nCur,s) ne 0) $ t_n(t,nCur)

$iftheni.n "%nitrogen%"=="false"
                                                  $  (( 1 eq 2) or (not sameas(nut,"N")))
$endif.n
$iftheni.p "%phosphate%"=="false"
                                                   $  (( 1 eq 2) or (not sameas(nut,"P")))
$endif.p
                                                                    ) ..
```

Figure 115:

```
*               ---  crop need based on plant uptake and calculated further need (Mehrbedarfskoeffizient)

                sum(plot_soil(plot,soil),
                         p_nutNeed(crops,soil,till,intens,nut,t) * v_cropHa(crops,plot,till,intens,t,nCur,s)
                                 * (1 + p_nutLossUnavoidable(soil,till,intens,nut)))
```

Figure 116:

The plant need (including over application of manure) has to equal the offered nutrients. For pasture, there is a special accounting needed since loss factors differ from stable. Furthermore, nutrients excreted during grazing are only available on the pasture. As stated above, there are different ways to account for N from manure.

The following elements determine the amount of N and P entering the nutrient balance with applied manure. Again, different ways to account for organic N are represented in the equation.

Losses can be calculated representing exogenous estimated N emissions. The reader should note that nutrients applied from manure are net of losses during storage.

The amount of organic N accounted for in fertilizing can be chosen in the GUI.

Furthermore, the requirements of the fertilizer planning from the German fertilizer directive can be followed.

For P from manure, no losses are taken into account.

Nutrients from mineral fertilizer application are added. No losses are taken into account since they are already included in the calculation of *p_nutLossUnavoidable* (see chapter 2.11.1).

Delivery of nutrients from soil and air are included.

```
$iftheni.man %manure% == true
*               ---  application over plant need of organic fertilizer is possible
                + v_nutOrganicOverNeed(crops,plot,till,intens,nut,t,nCur,s)
$endif.man
```

Figure 117:

```
                =e=
$ifthen.dh %dairyherd% == true
*
*   --- manure excreted during grazing on pasture: N , different calculation of losses [TK 01.03.16 revised]
*
                  sum( (nut2,m) $ (sameas(nut,"N") and (sameas(nut2,"norg") or sameas(nut2,"ntan")) ),
                  v_nut2ManurePast(nut2,t,nCur,m)
$iftheni.NorgAcc %NorgAccounting% == Interface
                     *   %NOrgAccountedInt%  )  $   sameas (crops,"past33")
$endif.NorgAcc
$iftheni.NorgAcc %NorgAccounting% == PlanningDueV16
                     *  0.8  )  $ sameas (crops,"past33")
$endif.NorgAcc
$iftheni.NorgAcc %NorgAccounting% == CalculatedLosses
                    )  $  sameas (crops,"past33")
$iftheni.envAcc %envAcc% == true
              -   sum ( (NiEmissions(emissions),m), v_emissions("past",emissions,t,nCur,m) ) $  sameas (crops,"past33")
$else.envAcc
            *  p_nutEffectivPastDueVNv   $  sameas (crops,"past33")
$endif.envAcc
$endif.NorgAcc
*   --- manure excreted during grazing pasture: P [TK 01.03.16 revised]
                  +  sum( m, v_nut2ManurePast("P",t,nCur,m) )    $ (sameas(nut,"P") and sameas (crops,"past33"))
$endif.dh
```

Figure 118:

```
$iftheni.man %manure% == true
*
*               -- application of N and P with organic fertilizer [TK 09.02.15 revised]
*
*   -- accounted for all N that is not lossed during application
$iftheni.NorgAcc %NorgAccounting% == CalculatedLosses
                  + sum( (ManApplicType,curManType,m)
                       $ (v_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,s,m) ne 0),
                          v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                              * p_nut2inMan("NORG",curManType)    )   $ sameas (nut, "N")
*                 -- NTAN applied minus losses with application
                  + sum( (ManApplicType,curManType,m)
                      $ (v_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,s,m) ne 0),
                         v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                             * p_nut2inMan("NTAN",curManType)
                                * (1 - p_ApplicationLossShareManure(crops,ManApplicType,m)) )  $ sameas (nut, "N")
$endif.NorgAcc
```

Figure 119:

```
*  -- accounting for Norg depends on percentage given in the interface
$iftheni.NorgAcc %NorgAccounting% == Interface
                  + sum( (ManApplicType,nut2,curManType,m)
                      $ ( ( not sameas (nut2,"P"))
                        $ (v_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,s,m) ne 0) ),
                        v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                            * p_nut2inMan(nut2,curManType) * %NOrgAccountedInt%  )  $ sameas (nut, "N")
$endif.NorgAcc
```

Figure 120:

```
*  -- accounting for Norg if farmers stick to the fertilizer planning of DueV 2016
$iftheni.NorgAcc %NorgAccounting% == PlanningDueV16
*                -- share of total N organic has to be accounted for in year of application
                 + sum( (ManApplicType,nut2,curManType,m)
                        $ ( ( not sameas (nut2,"P"))
                        $ (v_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,s,m) ne 0) ),
                     v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                          * p_nut2inMan(nut2,curManType) * 0.6  )  $ sameas (nut, "N")
*                -- Nmin in spring has to be taken into account
                 +   sum (plot_soil(plot,soil),p_basNut(crops,soil,till,nut,t))     $ sameas (nut, "N")
*                -- 10 % of total N organic of previous year are accounted
                 + sum( (ManApplicType,nut2,curManType,m,t_n(t-1,nCur1))
                        $ ( anc(nCur,nCur1) $ ( not sameas (nut2,"P")) $
                           (v_manDist.up(crops,plot,till,intens,manApplicType,curManType,t-1,nCur1,s,m) ne 0)  ),
                     v_manDist(crops,plot,till,intens,ManApplicType,curManType,t-1,nCur1,s,m)
                          * p_nut2inMan(nut2,curManType) * 0.1  )  $ sameas (nut, "N")

$endif.NorgAcc
```

Figure 121:

```
*                -- P from manure application, no losses taken into account
                  + sum( (ManApplicType,curManType,m)
                     $ (v_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,s,m) ne 0),
                        v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                          * p_nut2inMan("P",curManType)     )     $ sameas (nut, "P")
$endif.man
```

Figure 122:

```
*                -- mineral N application
                 + sum ((syntFertilizer,m),
                       v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m)
                                                * p_nutInSynt(syntFertilizer,nut) )
```

Figure 123:

```
*                -- N and P delivered from soil and air (mineralisation, deposition)
                 + sum(plot_soil(plot,soil),p_basNut(crops,soil,till,nut,t))*v_cropHa(crops,plot,till,intens,t,s)
 ;
```

Figure 124:

In the equation *nutSurplusMax_*, the application of nutrients from manure over plant need is restricted for each crop type:

```
*
*   --- nutrient surplus max over per ha restriction
*
    nutSurplusMax_( c_s_t_i(curCrops(crops),plot,till,intens),nut,tCur(t),nCur,s)
        $ ( (v_cropHa.up(crops,plot,till,intens,t,nCur,s) ne 0) $ t_n(t,nCur)) ..

       p_nutSurplusMax(crops,plot,till,intens,nut,t) * v_cropHa(crops,plot,till,intens,t,nCur,s)
             =G=  v_nutOrganicOverNeed(crops,plot,till,intens,nut,t,nCur,s);
```

Figure 125:

p_nutSurplusMax is calculated in *coeffgen\cropping.gms*.

```
p_nutSurplusMax(crops,plot,till,intens,nut,t) $ c_s_t_i(crops,plot,till,intens)
                  = min(200,sum(plot_soil(plot,soil),p_nutNeed(crops,soil,till,intens,nut,t) * 0.5));
```

Figure 126:

In the standard nutrient fate model the reductions in soil nutrient can be managed by:

1. reducing unnecessary manure applications which decrease *v_nutSurplusField*
2. lowering cropping intensity (when nutrient need is derived using N response curves). It reduces not only the overall nutrient needs and therefore the losses, but also reduces the loss rates per kg of synthetic fertilizer
3. switching between mineral and organic fertilization
4. changing the cropping pattern

Deprecated: Detailed Nutrient Fate Model by Crop, Month, Soil Depth and Plot

The detailed soil accounting module considers nutrient flows both from month to month and between different soil layers (top, middle, deep). It replaces the equations used in the standard nutrient fate model shown in the section above. The central equation is the following:

The detailed nutrient fate model considers as input flows:

- Application of organic and mineral fertilizers net of NH3 and other gas losses from application, they are brought to the top layer,
- atmospheric deposition (to the top layer),
- net mineralisation and
- nutrient leaching from the layer above.

The considered output flows are:

```
*
*    --- N,P pool in soil, certain depth, for certain crops
*        (works in that form for mono-culture)
*
    nutInSoilDepth_(curCrops(crops),plot,till,intens,nut,depth,tCur(t),nCur,m,s,w)
        $ (c_s_t_i(crops,plot,till,intens) $ p_plotsize(plot) $ t_n(t,nCur)) ..

       v_nutInSoilDepth(crops,plot,till,intens,depth,nut,t,nCur,m,s,w) =e= [
*
*               --- crop in previous year on same area(???)
*
                (v_nutInSoilDepthStart(crops,plot,till,intens,depth,nut,t,nCur,s))    $ (sameas(m,"Jan") $ tCur(t-1))
                + v_cropHa(crops,plot,till,intens,t,nCur,s)*20                        $ (sameas(m,"Jan") $ (not tCur(t-1)))
                + v_nutInSoilDepth(crops,plot,till,intens,depth,nut,t,nCur,m-1,s,w)   $ (not sameas(m,"Jan"))]
*
*               --- atmospheric deposition and mineralization
*
                + v_cropHa(crops,plot,till,intens,t,nCur,s) * (
*
*               --- atmospheric deposition by month and per ha
*
                     p_atmDep(nut,m,w) $ sameas(depth,"top")
*
*               --- mineralization per ha in specific soil strata by month and weather
*
                    + sum(plot_soil(plot,soil),p_mineral(soil,depth,nut,m,w))                  )
*
*               --- appplication of manure and synthetic fertilizer
*
$iftheni.h %herd%==true
                + sum( (ManApplicType,manType)
                         $ (v_manDist.up(crops,plot,till,intens,manApplicType,manType,t,nCur,s,m) ne 0),
                           v_manDist(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m)
                             * p_nutInMan(manType,nut)
                                * (1-p_nutManApplicTypeLossShare(ManApplicType,nut,m,w))) $ sameas(depth,"top")
$endif.h
*
*                 --- mineral N application times unity minus specific loss rate for mineral
*                     fertilizer type of NH3
*
                 + sum ( syntFertilizer,
                         v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m)
                             * p_nutInSynt(syntFertilizer,nut)
                                 * (1-sum(plot_soil(plot,soil),p_syntAppLossShare(syntFertilizer,soil,nut,m,w))))
                                                                                                  $ sameas(depth,"top")
*
*                   --- nutrient uptake by crops by soil level
*
                    - v_cropHa(crops,plot,till,intens,t,nCur,s)
                           * sum(plot_soil(plot,soil),
                              p_nutNeed(crops,soil,till,intens,nut,t))
*
*                          --- distribution over months and weather
*
                             * p_nutNeedDist(Crops,intens,w,m)
                               * p_uptakeDistrDepth(crops,depth,m)
*
*                   --- nutrient leaching to lower soil levels
*
                    - v_nutLeaching(crops,plot,till,intens,nut,depth,t,nCur,m,s,w)
*
*                   --- nutrient leaching from soil level above
*
                    + v_nutLeaching(crops,plot,till,intens,nut,depth-1,t,nCur,m,s,w);
```

Figure 127:

- Uptake by crops and
- leaching to the layer below.

The difference between the variables updates next month's stock based on current month's stock. Monthly leaching to the next deeper soil layer, *v_nutLeaching,* is determined as a fraction of plant available nutrients (starting stock plus inflows):

```
   nutLeaching_(curCrops(crops),plot,till,intens,nut,depth,tCur(t),nCur,m,s,w)
                $ (c_s_t_i(crops,plot,till,intens) $ p_plotSize(plot) $ t_n(t,nCur)  ) ..
         v_nutLeaching(crops,plot,till,intens,nut,depth,t,nCur,m,s,w)
            =e=  sum(plot_soil(plot,soil),
                     (
                      [   (v_nutInSoilDepthStart(crops,plot,till,intens,depth,nut,t,nCur,s))   $ (sameas(m,"Jan") $ tCur(t-1))
                        + v_cropHa(crops,plot,till,intens,t,nCur,s)*20  $ (sameas(m,"Jan") $ (not tCur(t-1)))
                        + v_nutInSoilDepth(crops,plot,till,intens,depth,nut,t,nCur,m-1,s,w)   $ (not sameas(m,"Jan"))]
*
*                           --- atmospheric deposition and mineralization
*
                          + v_cropHa(crops,plot,till,intens,t,nCur,s) * (
*
*                               --- atmospheric deposition by month and per ha
*
                              p_atmDep(nut,m,w) $ sameas(depth,"top")
*
*                               --- mineralization per ha in specific plot strata by month and weather
*
                             + p_mineral(soil,depth,nut,m,w)
                            )
*
*                           --- appplication of manure and synthetic fertilizer
*
$iftheni.h %herd%==true
                          + sum(  (ManApplicType,manType)
                                $ (v_manDist.up(crops,plot,till,intens,manApplicType,manType,t,nCur,s,m) ne 0),
                                  v_manDist(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m)
                                    * p_nutInMan(manType,nut)
                                      * (1-p_nutManApplicTypeLossShare(ManApplicType,nut,m,w)))
                                                                                $ sameas(depth,"top")
$endif.h
*
*                           --- mineral N application times unity minus specific loss rate for mineral
*                               fertilizer type of NH3
*
                          + sum ( syntFertilizer,
                                  v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m)
                                    * p_nutInSynt(syntFertilizer,nut)
                                      * (1-p_syntAppLossShare(syntFertilizer,soil,nut,m,w)))
                                                                                $ sameas(depth,"top")
*
*                           --- nutrient leaching from plot level above
*
                          + v_nutLeaching(crops,plot,till,intens,nut,depth-1,t,nCur,m,s,w)
                     )
                   * p_leachFactor(soil,m,w));
```

Figure 128:

The leaching losses below the root zone in combination with ammonia, other gas losses from mineral and organic fertilizer applications define the total nutrient losses at farm level in each month:

```
*
*   --- define different nutrient losses (NLeach,NH3,PLeach), enter also the GHG accounting
*
    NutLossApplic_(nutLosses,tCur(t),nCur,m) $ t_n(t,nCur) ..
       v_nutLossApplic(nutLosses,t,nCur,m) =e=
                  sum((c_s_t_i(curCrops(crops),plot,till,intens),nutLosses_nut(nutLosses,nut),s,w) $ p_plotSize(plot),
                      [
*
*                       --- nutrient leaching below root zone
*
                        v_nutLeaching(crops,plot,till,intens,nut,"deep",t,nCur,m,s,w) $ sameas(nutLosses,"nLeach")
*
*                       --- NH3 losses from manure
*
$iftheni.h %herd%==true
                      + sum( (ManApplicType,manType) $ (v_manDist.up(crops,plot,till,intens,manApplicType,manType,t,nCur,s,m) ne 0),
                           v_manDist(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m)
                             * p_nutInMan(manType,nut)
                               * p_nutManApplicTypeLossShare(ManApplicType,nut,m,w)) $ sameas(nutLosses,"nH3Man")
$endif.h
*                         --- mineral N application times unity minus specific loss rate for mineral fertilizer type of NH3
*
                        + sum ( syntFertilizer,
                                v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m) * p_nutInSynt(syntFertilizer,nut)
                                  * sum(plot_soil(plot,soil),p_syntAppLossShare(syntFertilizer,soil,nut,m,w))) $ sameas(nutLosses,"nH3Min")
                      ]  *  1/card(w)  * p_prob(s)     );
```

Figure 129:

The approach requires defining the nutrient needs of each crop per month, which is currently estimated:

```
*
*  --- nutrient need distribution is a rough guess, not really source
*
 table p_nutNeedDist(crops,intens,w,m) "Distribution of crop needs over months"

                                  JAN  FEB   MAR   APR MAY  JUN  JUL  AUG  SEP   OCT  NOV   DEC
 winterCere.normal.norm                0.05  0.1   0.2 0.2  0.3  0.1  0.05
 summerCere.normal.norm                      0.05  0.2 0.25 0.3  0.1
 winterRape .normal.norm               0.05  0.1   0.2 0.25 0.3  0.1       0.05
 potatoes.normal.norm
 maizCorn.normal.norm                                0.1 0.20 0.2  0.25 0.25 0.2   0.05
 maizCCM.normal.norm                                 0.1 0.20 0.2  0.25 0.25 0.2   0.05
 sugarbeet.normal.norm                               0.1 0.20 0.2  0.25 0.25 0.2   0.05
 summerBeans.normal.norm                 0.15  0.3 0.30 0.25 0.1
 summerPeas.normal.norm                  0.15  0.3 0.30 0.25 0.1
 wheatGPS.normal.norm             0.05  0.1  0.35  0.2 0.3


$iftheni.dairyHerd %dairyHerd% == true

 maizSil     .normal.norm                            0.1 0.20 0.2  0.25 0.25 0.2   0.05
 past33      .normal.norm          0.02 0.05  0.1   0.1 0.20 0.25 0.25 0.20 0.15  0.10 0.02  0.02
 gras20      .normal.norm          0.02 0.05  0.1   0.1 0.20 0.25 0.25 0.20 0.15  0.10 0.02  0.02
 gras29      .normal.norm          0.02 0.05  0.1   0.1 0.20 0.25 0.25 0.20 0.15  0.10 0.02  0.02
 gras34      .normal.norm          0.02 0.05  0.1   0.1 0.20 0.25 0.25 0.20 0.15  0.10 0.02  0.02


$endif.dairyHerd
;
*
* --- use same distribution for all intensities
*
 p_nutNeedDist(Crops,intens,"norm",m) = p_nutNeedDist(Crops,"normal","norm",m);
*
* --- fall back if not data are given: equal distribution across the year
*
*
 p_nutNeedDist(Crops,intens,"norm",m) $ (not sum(m1, p_nutNeedDist(Crops,intens,"norm",m1))) = 1/card(m);
*
* --- scale distribution to unity
*
 p_nutNeedDist(Crops,intens,"norm",m) $ p_nutNeedDist(Crops,intens,"norm",m)
    = p_nutNeedDist(Crops,intens,"norm",m) * 1/sum(m1, p_nutNeedDist(Crops,intens,"norm",m1));

*
*  --- nutrient need, taking into that output coefficient are measured in t and not dt, therefore * 10
*
 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));
```

Figure 130:

Similarly, the nutrient uptake by the crop from different soil layers is determined:

A weakness of this approach is how changes of cropping patterns are handled between years. It would be favourable to define the transition of nutrient pools from year to year based on a "crop after crop" variable in hectares for each soil type. However, this leads to quadratic constraints which failed to be solved by the industry QIP solvers [3]. Instead, the pool is simply redistributed across crops and a maximum content of 50 kg of nutrient per soil depth layer is fixed:

If the user switches on crop rotations a further restriction is added:

Nutrient Balance According to the Fertilizer directive

The German fertilizer directive requires that farms calculate a nutrient balance on an annual basis (DüV 2007). It combines nutrients input via manure and synthetic fertilizer with nutrients removal via the harvested crops. The surplus, i.e. the balance, is not allowed to exceed a certain threshold. In FARMDYN, the nutrient balance is always calculated. The threshold can be switched on and off in the GUI. Relevant equations can be found in *model\templ.gms*.

Nutrient removal via harvested product is calculated.

[3] QIP solvers do not allow for equality conditions which are by definition non-convex

```
*
*    --- nutrient update need by soil type is a pure guess
*
     p_uptakeDistrDepth(depthCrops,depth,m) = 0;

     p_uptakeDistrDepth(depthCrops,"top","FEB")    = 1;
     p_uptakeDistrDepth(depthCrops,"top","SEP")    = 1;
     p_uptakeDistrDepth(depthCrops,"top","OCT")    = 1;

     p_uptakeDistrDepth(depthCrops,"top","MAR")     = 0.9;
     p_uptakeDistrDepth(depthCrops,"medium","MAR")  = 0.1;

     p_uptakeDistrDepth(depthCrops,"top","APR")    = 0.80;
     p_uptakeDistrDepth(depthCrops,"medium","APR") = 0.15;
     p_uptakeDistrDepth(depthCrops,"deep","APR")   = 0.05;

     p_uptakeDistrDepth(depthCrops,"top","MAY")    = 0.70;
     p_uptakeDistrDepth(depthCrops,"medium","MAY") = 0.20;
     p_uptakeDistrDepth(depthCrops,"deep","MAY")   = 0.10;

     p_uptakeDistrDepth(depthCrops,"top","JUN")    = 0.65;
     p_uptakeDistrDepth(depthCrops,"medium","JUN") = 0.25;
     p_uptakeDistrDepth(depthCrops,"deep","JUN")   = 0.10;

     p_uptakeDistrDepth(depthCrops,"top","JUL")    = 0.60;
     p_uptakeDistrDepth(depthCrops,"medium","JUL") = 0.30;
     p_uptakeDistrDepth(depthCrops,"deep","JUL")   = 0.10;

     p_uptakeDistrDepth(depthCrops,"top","AUG")    = 0.60;
     p_uptakeDistrDepth(depthCrops,"medium","AUG") = 0.30;
     p_uptakeDistrDepth(depthCrops,"deep","AUG")   = 0.10;

     p_uptakeDistrDepth(grasCrops,"top",m)    = 0.40;
     p_uptakeDistrDepth(grasCrops,"medium",m) = 0.40;
     p_uptakeDistrDepth(grasCrops,"deep",m)   = 0.20;

     p_uptakeDistrDepth(depthCrops,depth,m) $ p_uptakeDistrDepth(depthCrops,depth,m)
       = p_uptakeDistrDepth(depthCrops,depth,m) * 1 / sum(depth1, p_uptakeDistrDepth(depthCrops,depth1,m));
```

Figure 131:

```
    nPool_(plot,depth,nut,tCur(t),nCur,s) $ (tCur(t-1) $ p_plotSize(plot) $ t_n(t,nCur)) ..

       sum( c_s_t_i(curCrops(crops),plot,till,intens),
            v_nutInSoilDepthStart(crops,plot,till,intens,depth,nut,t,nCur,s))

         =e= sum( (t_n(t-1,nCur1),c_s_t_i(curCrops(crops1),plot,till,intens),w) $ anc(nCur,nCur1),
              v_nutInSoilDepth(crops1,plot,till,intens,depth,nut,t-1,nCur1,"Dec",s,w)/card(w)) $ tCur(t-1);

*
*   --- max nutrient concentration at start of year
*
    nPoolmax_(plot,curCrops(crops),till,intens,depth,nut,tCur(t),nCur,s)
        $ (c_s_t_i(crops,plot,till,intens) $ p_plotSize(plot) $ t_n(t,nCur) ) ..

       v_nutInSoilDepthStart(crops,plot,till,intens,depth,nut,t,nCur,s)
                  =L= v_cropHa(crops,plot,till,intens,t,nCur,s) * 50;
```

Figure 132:

```
*
*   --- shift NPool from one to the next year
*
$iftheni.cropRot %cropRotations% == true

    nPoolRot_(plot,cropTypes,depth,nut,tCur(t),nCur,s)

          $ (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,s) ne 0),1)
                       $ tCur(t-1) $ t_n(t,nCur)) ..

*
*      --- that is implicitly a loop over all rotation which have cropTypes in the current year
*
       sum( (cropTypes_crops(cropTypes,crops),c_s_t_i(curCrops(crops),plot,till,intens)),
            v_nutInSoilDepthStart(crops,plot,till,intens,depth,nut,t,nCur,s))

         =L=
*
*         --- take the rotations which have the cropTypes in the current years and cropTypes1
*             in the last year
*
          sum((cropType0_rot(cropTypes,curRot(rot)),cropType1_rot(cropTypes1,rot),
                cropTypes_crops(cropTypes1,crops1),
                c_s_t_i(curCrops(crops1),plot,till,intens),t_n(t-1,nCur1),w) $ anc(nCur,nCur1),
                   v_nutInSoilDepth(crops1,plot,till,intens,depth,nut,t-1,nCur1,"Dec",s,w)/card(w)) $ tCur(t-1);
```

Figure 133:

```
*
*   --- Calcution of nutrient removal by crop output for calculation of nutrient balance according to DüV [TK 19.12.13]
*       Crop output (p_nutContent is per 100 kg, output coefficients are in tons)
*

    nutRemovalDuev_(nut,tCur(t),nCur) $ t_n(t,nCur) ..

       v_nutRemovalDuev(nut,t,nCur)
            =e=

               sum( (c_s_t_i(crops,plot,till,intens),s), v_cropHa(crops,plot,till,intens,t,nCur,s)
                           * sum( (plot_soil(plot,soil),curProds), p_OCoeffC(crops,soil,till,intens,curProds,t)
                                 * p_nutContent(crops,curProds,nut)*10 * p_prob(s))   );
```

Nutrient input via synthetic fertilizer is calculated.

```
*
*   --- Calculation of annual mineral fertilizer applied entering nutbalduev (in kg P and N)
*

   synthAppliedDueV_(nut,t,nCur)  $ t_n(t,nCur)..

           v_synthAppliedDueV(nut,t,nCur)    =e=

                                sum( (c_s_t_i(crops,plot,till,intens),syntFertilizer,s,m),
                                   v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m)
                                             * p_nutInSynt(syntFertilizer,nut) * p_prob(s))      ;
```

Figure 134:

In the equation *nutBalDuev_*, nutrient input and output are combined. Input from organic sources, *v_nutExcrDuev* and *v_nutBiogasDuev,* are calculated in the *manure_module.gms* and the *biogas_module.gms* (equations not shown here). Furthermore, nutrients export via manure export is taken into account. *v_surplusDueV* is the surplus of the nutrient balance.

The surplus, *v_surplusDueV,* is not allowed to exceed a certain threshold given by the GUI.

Environmental Accounting Module

!!! abstract The environmental accounting module *env_acc_module.gms* allows quantifying Ammonia (NH3), nitrous oxide (N2O), nitrogen oxides (NOx), elemental nitrogen (N2). 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.

Gaseous emissions

All relevant calculations are listed in *model\env_acc_module.gms.* Emissions of NH3, N2O, NOx and N2 are calculated, following calculation schemes and emission factors from relevant guidelines (IPCC 2006, Rösemann et al. 2015, EMEP 2013). We apply a mass balance approach to quantify different N emissions, subtracting emission factors from different N pools along the nutrient flow of the farm. Emissions factors are listed in *coeffgen\env_acc.gms.*

The sets *emissions* and *sources* contain relevant emissions and their sources. The cross set *source_emissions* links emissions to relevant sources (Methane (CH4)

```
*
*   --- Calculation of nutrient balance according to DüV 2006 § 5(1) [TK 18.10.13]
*
    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)
*  --- Crop output (nutrient removal)
        -   v_NutRemovalDuev(nut,t,nCur)
$iftheni.h %herd% == true
*   --- Nutrients exported from farm
      $$iftheni.ExMan %AllowManureExport%==true
        -  sum( (m,nut2) $(not sameas (nut2,"P")), v_nut2export(nut2,t,nCur,m) )  $ sameas (nut,"N")
        -  sum(m, v_nut2export("P",t,nCur,m) )                                    $ sameas (nut,"P")
      $$endif.ExMan
      $$iftheni.emissionRight not "%emissionRight%"==0
        -  sum( (m,nut2) $(not sameas (nut2,"P")), v_nut2exportMER(nut2,t,nCur,m) )  $ sameas (nut,"N")
        -  sum(m, v_nut2exportMER("P",t,nCur,m) )                                    $ sameas (nut,"P")
      $$endif.emissionRight
$endif.h
        =e=
            v_surplusDueV(t,nCur,nut)      ;
```

Figure 135:

```
$ifthen.duev %duev% == true
*
*   ---- Restriction of surplus according to DueV 2006 [TK 18.10.13]
*
    nutSurplusDueVRestr_ (tCur(t),nCur,nut)   $ (p_surPlusDuevMax(t,nut) $ t_n(t,nCur))  ..
      v_surplusDueV(t,nCur,nut)
        =L=
          p_surplusDueVMax(t,nut) *  sum(plot,v_totPlotLand(plot,t,nCur));

$endif.duev
```

Figure 136:

is also included but still on the development stage and therefore not described here).

```
$iftheni.env %envAcc%==true

   set emissions / NH3,N2O,NOx,N2,N2Oind,NSoilSurplus,PsoilSurplus
$ifi %methane% == true    ,CH4
/ ;
```

Figure 137:

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

   set source_emissions(source,emissions) /
$iftheni.met %methane% == true
                                          staSto.(NH3,N2O,NOx,N2,N2Oind,CH4)
$else.met
                                          staSto.(NH3,N2O,NOx,N2,N2Oind)
$endif.met
                                          past.(NH3,N2O,NOx,N2,N2Oind)
                                          manAppl.(NH3,N2O,NOx,N2,N2Oind)
                                          minAppl.(NH3,N2O,NOx,N2,N2Oind)
                                          field.(NSoilSurplus,PsoilSurplus)
$ifi %methane% == true                    entFerm.CH4
                                          / ;

$endif.env
```

Figure 138:

All gaseous emissions are calculated in the equation *emissions_* on a monthly basis. By the use of *sameas* statements, only relevant sources and emissions are activated in the corresponding part of the equation.

```
v_emissions(source,emissions,t,nCur,m)

            =E=
```

Figure 139:

Calculation of emissions from stable and manure storage.

Calculation of emissions from manure excreted on pasture (only relevant for dairy).

Calculation of emissions from manure application, application techniques have different emission factors.

Calculation of emissions from mineral fertilizer application.

```
$iftheni.h %herd% == true
* --- Calculation of NH3, N2O, NOx, N2, N2Oind from stable and storage (staSto)
            [ v_nut2ManureM("NTAN",t,nCur,m)       * p_EFSta("NH3") ]    $ ( sameas (emissions,"NH3") $ sameas (source,"staSto") )
          +  [  v_nutPoolInStorage("NTAN",t,nCur,m) * p_EFSto("NH3") ]    $ ( sameas (emissions,"NH3") $ sameas (source,"staSto") )
          +  [  ( v_nut2ManureM("NTAN",t,nCur,m) +  v_nut2ManureM("NOrg",t,nCur,m) )  * p_EFStaSto("N2O") ]   $ ( sameas (emissions,"N2O") $ sameas (source,"staSto") )
          +  [  ( v_nut2ManureM("NTAN",t,nCur,m) +  v_nut2ManureM("NOrg",t,nCur,m) )  * p_EFStaSto("NOx") ]   $ ( sameas (emissions,"NOx") $ sameas (source,"staSto") )
          +  [  ( v_nut2ManureM("NTAN",t,nCur,m) +  v_nut2ManureM("NOrg",t,nCur,m) )  * p_EFStaSto("N2") ]   $ ( sameas (emissions,"N2") $ sameas (source,"staSto") )
          +  [  ( v_nut2ManureM("NTAN",t,nCur,m) +  v_nut2ManureM("NOrg",t,nCur,m) )  * p_EFStaSto("N2Oind")] $ ( sameas (emissions,"N2Oind") $ sameas (source,"staSto") )
$endif.h
```

Figure 140:

```
* --- Calculation of NH3, N2O, NO and N2 losses from manure excretion on pasture for dairy cows
$iftheni.dh %dairyherd% == true

          +    [ v_nut2ManurePast("NTAN",t,nCur,m) * p_EFpasture("NH3") ]  $ ( sameas (emissions,"NH3") $ sameas (source,"past") )
          +   [ ( v_nut2ManurePast("NTAN",t,nCur,m) +  v_nut2ManurePast("NOrg",t,nCur,m) ) *  p_EFpasture("N2O") ] $ ( sameas (emissions,"N2O") $ sameas (source,"past") )
          +   [ ( v_nut2ManurePast("NTAN",t,nCur,m) +  v_nut2ManurePast("NOrg",t,nCur,m) ) *  p_EFpasture("NOx") ] $ ( sameas (emissions,"NOx") $ sameas (source,"past") )
          +   [ ( v_nut2ManurePast("NTAN",t,nCur,m) +  v_nut2ManurePast("NOrg",t,nCur,m) ) *  p_EFpasture("N2") ] $ ( sameas (emissions,"N2") $ sameas (source,"past") )
$endif.dh
```

Figure 141:

```
* --- Calculation of NH3, N2O, NOx, N2 and N2Oind from manure application
*     NH3 losses depending on technology, source Rösemann et al. 2015, pp. 316-318,342

          +   [sum( (c_s_t_i(curCrops(crops),plot,till,intens),ManApplicType,curManType,s),
                v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                                    * p_nut2inMan("NTAN",curManType)
                                    * p_ApplicationLossShareManure(crops,ManApplicType,m)  )  ]      $ ( sameas (emissions,"NH3") $ sameas (source,"manAppl") )
          +  [ sum(nut2 $ (not sameas (nut2,"P")), v_nut2ManApplied(nut2,t,nCur,m))  *   p_EFApplManN2O ]   $ ( sameas (emissions,"N2O") $ sameas (source,"manAppl") )
          +  [ sum(nut2 $ (not sameas (nut2,"P")), v_nut2ManApplied(nut2,t,nCur,m)) * 0.012 ]    $ ( sameas (emissions,"NOx") $ sameas (source,"manAppl") )
          +  [ sum(nut2 $ (not sameas (nut2,"P")), v_nut2ManApplied(nut2,t,nCur,m)) * 0.07 ]     $ ( sameas (emissions,"N2") $ sameas (source,"manAppl") )

          + [[   sum( (c_s_t_i(curCrops(crops),plot,till,intens),ManApplicType,curManType,s),
                       v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                                    * p_nut2inMan("NTAN",curManType)
                                    * p_ApplicationLossShareManure(crops,ManApplicType,m) * p_prob(s) )

            +  ( sum(nut2 $ (not sameas (nut2,"P")), v_nut2ManApplied(nut2,t,nCur,m)) * 0.012 )
                  ] * 0.01 ]     $  ( sameas (emissions,"N2Oind") $ sameas (source,"manAppl") )
```

Figure 142:

```
* --- Calculation of NH3, N2O, NOx, N2 and N2Oind from mineral fertilizer application
*     Based on Rösemann et al. 2015, pp. 316-317

            +   [    sum( (c_s_t_i(curCrops(crops),plot,till,intens),syntFertilizer,s),
                          v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m)    * p_nutInSynt(syntFertilizer,"N") * p_prob(s)  )
                    *  0.037 ]   $ ( sameas (emissions,"NH3") $ sameas (source,"minAppl") )

            +   [    sum( (c_s_t_i(curCrops(crops),plot,till,intens),syntFertilizer,s),
                          v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m)   * p_nutInSynt(syntFertilizer,"N")  * p_prob(s) )
                    * 0.01   ]   $ ( sameas (emissions,"N2O") $ sameas (source,"minAppl") )

            +   [    sum( (c_s_t_i(curCrops(crops),plot,till,intens),syntFertilizer,s),
                          v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m)    * p_nutInSynt(syntFertilizer,"N") * p_prob(s)  )
                    * 0.012  ]   $ ( sameas (emissions,"NOx") $ sameas (source,"minAppl") )

            +   [    sum( (c_s_t_i(curCrops(crops),plot,till,intens),syntFertilizer,s),
                          v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m)   * p_nutInSynt(syntFertilizer,"N") * p_prob(s)   )
                    *  0.07  ] $ ( sameas (emissions,"N2") $ sameas (source,"minAppl") )

            +   [    sum( (c_s_t_i(curCrops(crops),plot,till,intens),syntFertilizer,s),
                          v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m)    * p_nutInSynt(syntFertilizer,"N") * p_prob(s)  )
                          * ( 0.037 + 0.012)  ]  $ ( sameas (emissions,"N2Oind") $ sameas (source,"minAppl") )


                 ;
```

Figure 143:

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 most relevant environmental threats of farming systems. Since it is highly depending on environmental and geographical conditions, fixed emissions 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. This part of the environmental accounting will be replaced by results from crop models as part as of an ongoing research project (http://www.ilr.uni-bonn.de/pe/research/project/pro/pro16_e.html).

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

```
*
* --- Calculation of soilsurface balance that indicates potential N for leaching and P for erosion; not plot or crop specific
*
   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,curManType,s,m)
                     $ (v_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,s,m) ne 0),
                   v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                      * p_nut2inMan("NORG",curManType) * p_prob(s)    )    $ sameas (nut, "N")

                + sum( (c_s_t_i(curCrops(crops),plot,till,intens),ManApplicType,curManType,s,m)
                     $ (v_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,s,m) ne 0),
                   v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                      * p_nut2inMan("NTAN",curManType) * p_prob(s)    )  $ sameas (nut, "N")

                + sum( (c_s_t_i(curCrops(crops),plot,till,intens),ManApplicType,curManType,s,m)
                     $ (v_manDist.up(crops,plot,till,intens,manApplicType,curManType,t,nCur,s,m) ne 0),
                   v_manDist(crops,plot,till,intens,ManApplicType,curManType,t,nCur,s,m)
                      * p_nut2inMan("P",curManType) * p_prob(s)    )    $ sameas (nut, "P")

                -  sum( (NiEmissions(Emissions),m), v_emissions("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,s,m),
                        v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m) * p_nutInSynt(syntFertilizer,nut) * p_prob(s)  )

                -  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),s),
                        p_nutNeed(crops,soil,till,intens,nut,t) * v_cropHa(crops,plot,till,intens,t,nCur,s)  * p_prob(s) )

                +    sum( (plot_soil(plot,soil),c_s_t_i(curCrops(crops),plot,till,intens),s),
                        p_basNut(crops,soil,till,nut,t) *  v_cropHa(crops,plot,till,intens,t,nCur,s) * p_prob(s) )


             ;
```

Figure 144:

Note, that the calculated surplus differs from the surplus calculated for the threshold under the fertilizer 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 fertilizer 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 (2014) 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 at the one hand the revenues stemming from the heat and electricity production of the biogas plant, and at 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))) ;

display p_priceElec;
p_priceElec(bhkw,"E2004",tCur(t)) =   ( p_priceElecBase("150kW","E2004") * (150/p_powRate(bhkw,"E2004"))
                                    + p_priceElecBase(bhkw,"E2004") * (p_powRate(bhkw,"E2004") - 150)/p_powRate(bhkw,"E2004"))
                                        + p_priceElecE2004(bhkw,"E2004") * (150/p_powRate(bhkw,"E2004"))
                                        + p_priceElecE2004(bhkw,"E2004") * (p_powRate(bhkw,"E2004") - 150)/p_powRate(bhkw,"E2004")
```

Figure 145:

p_priceElecBase, used to calculate the guaranteed feed-in tariff differentiated by size, includes the base rate and additional bonuses [4] 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.

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.*

[4] For the EEG 2004: NawaRo-Bonus, KWK-Bonus; For the EEG 2009: Nawaro-Bonus, KWK-Bonus **or** NawaRo-Bonus, KWK-Bonus and Manure-Bonus

Furthermore, the revenue stemming from heat is accounted for and it 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 *\coeff-gen\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"
                                 /;
```

Figure 146:

```
*
*   --- Yearly Revenue of BHKW
*
    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)));
```

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"
                                           /
```

Figure 147:

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.

```
*
*   --- never more then one plant operational at any time
*
    invBioGasTot_(tCur(t),nCur) $ t_n(t,nCur) ..

       sum( (curBhkw(bhkw),curEeg(eeg)), v_invBioGas(bhkw,eeg,t,nCur)) =L=  1;
*
*   --- Inventory of biogas plants on farm: past investments, account for max life time and
*       for plant parts, the latter differentiated by investment horizon (IH)

    invBioGas_(curBhkw(bhkw),curEeg(eeg),tFull(t),nCur,ih) $ (ih2O(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,t1,n1,ih) )

                 + 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) );
*
*   --- inventory of parts restricts used of existing plant
*
    invBioGasTotParts_(curBhkw(bhkw),tCur(t),nCur) $ t_n(t,nCur)  ..

       sum ( curEeg(eeg), v_invBioGasParts(bhkw,eeg,t,nCur)) =G= sum(curEeg(eeg), v_invBioGas(bhkw,eeg,t,nCur));
```

Figure 148:

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.

```
*
*   --- Providing the possibility to switch an existing plant built under a past eeg
*       to a new EEG, technically, the switch re-assignes the plant to the new egg in the rest of the model
*       Note: the diagonal element eeg,eeg1 simply use the inventories without switch
*
    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));
*
*   --- actual biogas plant use, restricted to "switched" inventory
*
    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));
```

Figure 149:

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, as it is assumed that the biogas plant is not operating for 9% of the available time due to maintenance, etc.

```
*
*--- Fixing the maximum electricty produced by the biogas plant differentiated by plant size
*    as well as accounting for transformation losses

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);
```

Figure 150:

The production process of electricity, *v_prodElec,* is constructed in a two-stage procedure. First, biogas [5], *v_methCrop/Manure,* is produced in the fermenter as the product of crops and manure, *v_usedCrop/Man,* 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*.

```
*       I. Stage
*
*   --- methane from crops (relates to needed substrates to meet methane requirement)
*
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) );

*   --- methane from manure per year and months

methManure_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ t_n(t,nCur) ..

       v_methManure(bhkw,eeg,t,nCur,m)
          =e= sum(maM,     v_usedManBiogas(bhkw,eeg,maM,t,nCur,m) * p_manure(maM) );


*       II. Stage
*
*--- Calculation of electricity output in kWh of biogas plant
*
kWel_(curBhkw(bhkw),curEeg(eeg),tCur(t),nCur,m) $ t_n(t,nCur) ..

       v_prodElec(bhkw,eeg,t,nCur,m)
         =e= (v_prodElecCrop(bhkw,eeg,t,nCur,m) + v_prodElecManure(bhkw,eeg,t,nCur,m));

*--- Electricity output generated by crops

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;

*--- Electricity output generated by manure

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;
```

Figure 151:

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.

[5] 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.

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 percent maize silage and 30 percent manure. Hence, the input quantity derived from crops, *v_usedCropBiogas,* and manure, *v_usedManBiogas,* is bound by the fermenter size, *v_totVolFermMonthly.*

```
*
*   --- Fermenter input cannot exceed fermenter capacity of existing bio-gas plants :
*       Link made to size of biogas - plant
*
    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);

*
*   --- restriction on total volume depending on substrate input composition
*
    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(maM, v_usedManBiogas(bhkw,eeg,maM,t,nCur,m) );
```

Figure 152:

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.

```
*
*   --- Losses for purchased crops and on-farm produced crops
*
    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);

*
*   --- Aggregation of purchased manure and manure stemming from on farm
*
    manureTot_(curBhkw(bhkw), curEeg(eeg),maM,tCur(t),nCur,m) $ t_n(t,nCur) ..

       v_usedManBiogas(bhkw,eeg,maM,t,nCur,m)
          =e=
             v_purchManure(bhkw,eeg,maM,t,nCur,m) $ selPurchInputs(maM)
$ifi %herd%==true            + v_volManBiogas(bhkw,eeg,maM,t,nCur,m)
    ;

*
*   --- link manure quantity on farm to volume in fermenter
*
$iftheni.h %herd%==true

    volManBioGas_(tCur(t),nCur) $ t_n(t,nCur)..

       v_manQuant(t,nCur)
          =G= sum( (curbhkw(bhkw),curEeg(eeg),maM,m), v_volManBiogas(bhkw,eeg,maM,t,nCur,m)) ;

$endif.h
```

Figure 153:

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 $\frac{\text{kg oDM}}{m^3 \cdot d}$ [6] and is converted into a monthly limit.

[6] oDM = organic dry matter; m3 = cubic meter; d = day

```
*
*   --- Implementing the digestion load as technological constraint with regard
*       to maximal organic dry matter in the fermenter

    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))  ;
*
*    --- share of fermented volume used, in organic dry matter
*
    digLoad_(curBhkw(bhkw),tCur(t),nCur,m) $ t_n(t,nCur) ..

       v_digLoad(bhkw,t,nCur,m) =e= (   sum ( (curEeg(eeg),crM(biogasFeedM)),
                                        v_usedCropBiogas(bhkw,eeg,crM,t,nCur,m)
                                           * p_dryMatterCrop(crM)* p_orgDryMatterCrop(crM))

                                + sum( (curEeg(eeg),maM),
                                        v_usedManBiogas(bhkw,eeg,maM,t,nCur,m)
                                           * p_dryMatterManure(maM)*p_orgDryMatterManure(maM) ) )
                                     / p_volFerm(bhkw);
```

Figure 154:

The data used for the fermenter technology can be seen in *\coeff-gen\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 percent of his input quantity is manure based, as can be seen in the following code.

```
*
*   --- EEG 2009:  restricitions for manure boni
*
    manureRes_(curBhkw(bhkw),eegMan(eeg),tCur(t),nCur,m) $ t_n(t,nCur) ..

       sum(maM,  v_usedManBiogas(bhkw,eeg,maM,t,nCur,m)) =g= v_totVolFermMonthly(bhkw,eeg,t,nCur,m)*0.3 ;
```

Figure 155:

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 has to ensure that not more than 60 percent of the used fermenter volume, *v_totVolFermMonthly,* is used for maize. Second, under the assumption that the operator uses 25 percent of the heat emitted by the combustion engine for the fermenter itself, he has to sell at least 35 percent of the generated heat externally;

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

```
*
*   --- EEG 2012 - Restrictions
*       (a) Mass restriction of 60% corn products in the fermenter at all times
*           - All boni are dependent on this restriction

    maizeRes_(curBhkw(bhkw),eegDif(eeg),biogasFeedM,tCur(t),nCur,m) $ (eegDif(eeg) $ t_n(t,nCur)) ..
       v_usedCropBiogas(bhkw,eeg,"maizSil",t,nCur,m) =l=  0.6 * v_totVolFermMonthly(bhkw,eeg,t,nCur,m);

*
*   --- EEG 2012 - Restrictions
*       b) Minimum amount of heat which has to be used/sold - All boni are dependent on the fact if that requirement
*          is fullfiled - level of sold v_prodheat has to be determined in the beginning
*            -> parameter for share of prodheat, which can be chosen. Otherwise the farmer would always opt
*               for the highest value of the parameter! Parameter set has to include the dependent eeg!

    heatRes_(curBhkw(bhkw),eegDif(eeg),tCur(t),nCur,m) $ (eegDif(eeg) $ t_n(t,nCur)) ..
       v_sellHeat(eeg,t,nCur) =g= p_minHeatSold * v_prodHeat(eeg,t,nCur);
```

Figure 156:

Dynamic Character of FARMDYN

The Fully Dynamic Version

As described in earlier sections, the model template optimizes the farm production process over time in a fully dynamic setting. Connecting different modules over time (t_1-t_n) allows for a reproduction of 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.

Concerning fodder composition, decision points in each year are every three month. 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 application of manure or synthetic fertilizers, 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 behavior: deterministic versus stochastic model versions

!!! danger “Prototype feature” This feature has’nt been thoroughly tested and should be used with caution.

!!!abstract The default layout of the model maximizes the net present value 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 maximization of the expected net present value, different types of risk behavior 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, three set-ups are possible:

1. A deterministic version
2. A partial stochastic programming version where only some variables (crop acreages, feed mix, manure handing and fertilization) are state contingent. In that version, in each year, several price levels for input and outputs can be considered and the state contingent variables adjusted accordingly, but conceptually, each year starts again with a given mean expected price level as all other variables are not state contingent.
3. 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 States-of-Natures, 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 states-of-natures 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 state-of-nature.

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 2^10 ~ 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 behavioral models to maximize 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 behavior section below.

Partly stochastic version where long term variables are not state contingent

States of Nature (SON) in the partly stochastic version

In the partly stochastic version, only some variables are assumed to be state contingent and there is no complex tree structure. Specifically, we assume that SONs relate to the farm's market environment, specifically to the input and output prices, wages and interest rates faced by the firm. Crop yield variability is not considered. Each SON defines price levels for all input and output prices simultaneously, i.e. one vector of input and output prices. The cropping pattern, feeding, off-farm labour on an hourly basis and further farm activities are state contingent with regard to these SONs. However, decisions with a long-term character (full or half time work off farm, investment decisions, herd size, renting out of land) are not state contingent. The differentiation between SON specific decisions (cropping, feeding) and annual decisions in the otherwise deterministic version of the model is depicted in the following Figure:

7. Figure 6: Systematic view on the model approach

Source: Own illustration

The first two blocks of variables shown above labelled investments and herds are identical for all state of nature as indicated with the blue brackets below, whereas cropping and feeding and some other farm management decisions are state contingent and are adjusted to the State-of-the-nature. That implies that for instance the unique, i.e. not state contingent, investment decisions in machinery must ensure that the maximum occurring depreciation under any state of nature can be realized.

Objective Function in the deterministic and partly stochastic version

In the deterministic version of the model, we consider a maximization of net present value of profit under a discount rate. The partly stochastic version assumes always a risk neutral profit maximizing farmer, and hence maximizes the expected net present values of profits from SONs. 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 net present value; therefore, a definition close to the flow-to-equity approach is used:

```
*
*   --- net present value of cash balance in average per year
*       over the simulation horizon
*
    OBJE_           ..
*
       v_obje =L=
$iftheni.compStat not "%dynamics%"=="comparative-static"
*
*        --- accumulated liquidity
*           + household withdrawals
*           + revenues-costs of
*            liquidating farm in last year, mean over final leaves
*            (= equal to simulated value for determinstic version)
*
                v_objeMean

$else.compStat

               sum( t_n(tCur,nCur), (v_liquid(tCur,nCur)+ p_hcon(tCur))*p_probN(nCur))
*
*       --- penalty in case GHG limits becomes infeasible
*
        $$ifi %GHGs%==true   - sum( (actInd(ind),nCur) $ ( (nCur.pos eq 1) $ v_GhgEmissions.up(ind,"Mean",nCur)),
        v_slackGHG(ind,nCur) * 1.E+3 *p_probN(nCur))

$endif.compStat

*
*       --- 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;
```

Figure 157:

Further on, fully dynamic optimization assumes that the decision maker is fully informed about the future states of nature such that the economically optimal farm plan over the chosen planning horizon is simulated, potentially under different future SON (best-practice simulations).

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 state of nature 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 and partly stochastic 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 maximizes the probability weighted average of the final liquidity for each final leave in the decision tree:

```
    OBJE_           ..
*
        v_obje =L=
$ifthen1.compStat not "%dynamics%"=="comparative-static"
*
*         --- accumulated liquidity
*             + household withdrawals
*             + revenues-costs of
*               liquidating farm in last year, mean over final leaves
*               (= equal to simulated value for determinstic version)
*
                v_objeMean
```

Figure 158:

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

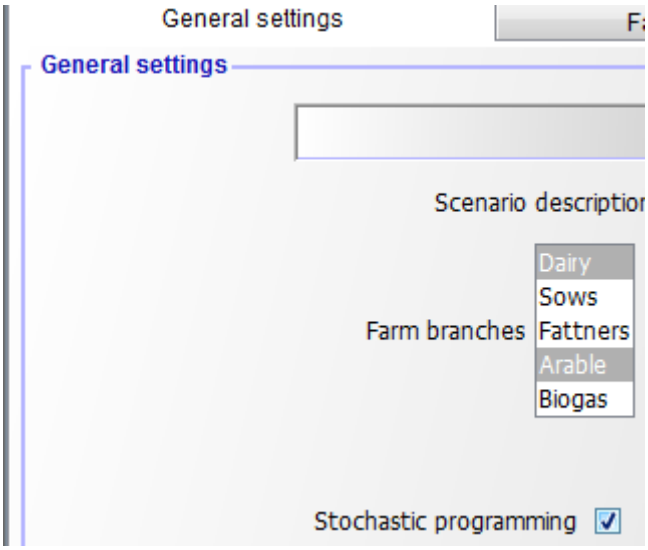


Figure 159:

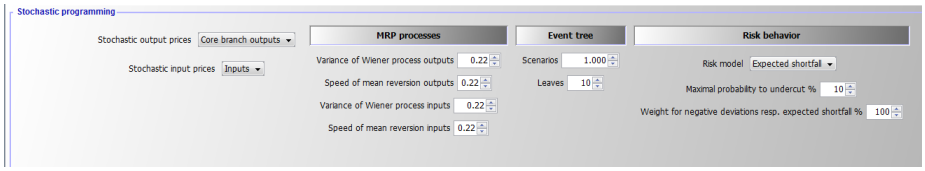


Figure 160:

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" / " ",nonsense /;
   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;
```

Figure 161:

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:

```
---- VAR v_objeTS  Gross margin each state of nature

                LOWER          LEVEL          UPPER       MARGINAL

2010..s1        -INF      153800.7949        +INF              .
2011..s1        -INF      149673.2763        +INF              .
```

Figure 162:

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

Generating Random Variable(s) and the decision tree

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

```
$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
```

Figure 163:

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 graphical user interface 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"

 set dummy / price /;
 execute_loadpoint "%scrdir%\\mrp.gdx" p_randVar,tn,anc;
 p_randVar("priceInputs",n) = p_randVar("Price",n);

$endif.stochPrices
```

Figure 164:

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%.

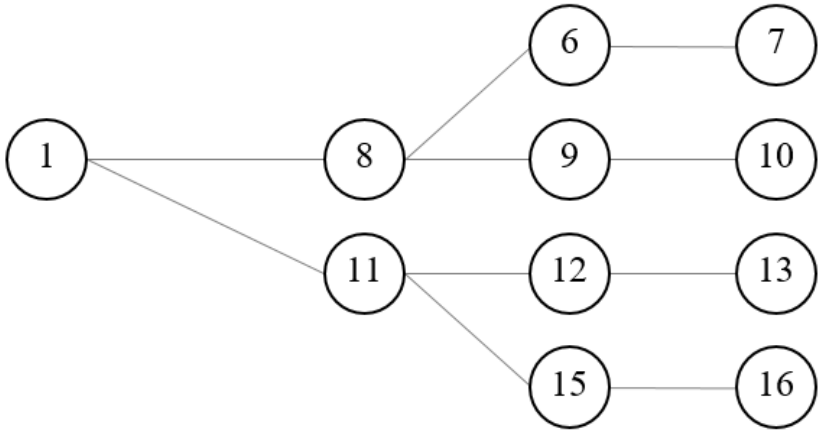


Figure 8: Example of an input decision tree organized as a fan. Source: Own illustration

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. That leads to the curse of dimensionality, as the number of variables and equations in the model increases quadratic in 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* 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).

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;
```

Figure 165:

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

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:

```
*
*   --- 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);
```

Figure 166:

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

Figure 167:

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.:

```
*
*   --- if the farm is there in t, it muse have been there in t-1
*
    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));
```

Figure 168:

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

+| > **Important Aspects to remember!** | | | | 1. Even if the program scales the drawn price changes such that their | | mean is equal to unity, this does not guarantee that the model, even | | without stage contingency, would perfectly replicate the deterministic | | version as the timing of the changes is also relevant (discounting, | | dynamic effects on liquidity etc.). | | | | 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 maximization. | | | | 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

```
*         --- steady state: starting herds before the first fully simulated year are equal to that one
*
      v_herdStart(herds,breeds,feedRegime,tBefore,nCur1,m) =e= sum(t_n("%firstYear%",nCur) $ isNodeBefore(nCur,nCur1),
                                                                 v_herdStart(herds,breeds,feedRegime,"%firstYear%",nCur,m));
```

Figure 169:

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):

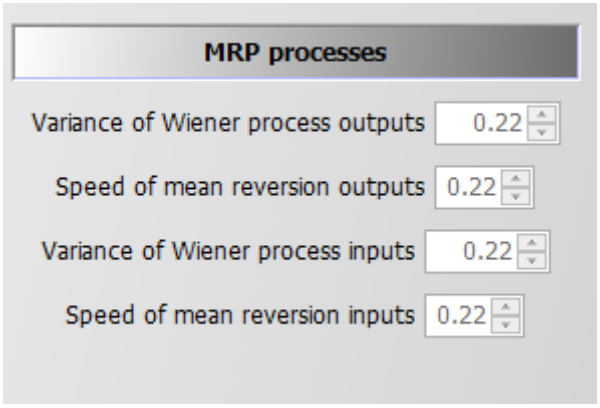


Figure 170:

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 a parameter in GAMS and not declared as a variable. 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):

```
*
*   --- revenue from sales of animal and crop products in average over states of nature
*       (SON specific price times SON specific production quantities)
*
    salRev_(tCur(t),nCur,s) $ t_n(t,nCur)    ..
*
       v_salRev(t,nCur,s)  =e= sum(  curProds(prodsYearly),
                                     p_price(prodsYearly,t,s)
$iftheni.sp %stochProg%==true
      * ( 1 + (p_randVar("priceOutputs",nCur)-1) $ randProbs(prodsYearly) )
$endif.sp
                                   *  v_saleQuant(prodsYearly,t,nCur,s));
```

Figure 171:

```
*
*   --- costs of buying inputs
*       (SON specific price times SON specific production quantities)
*
    buyCost_(curInputs(inputs),tCur(t),nCur,s) $ (t_n(t,nCur) $ p_inputprice(inputs,t,s))   ..

       v_buyCost(inputs,t,nCur,s) =e= p_inputprice(inputs,t,s)
$iftheni.sp %stochProg%==true
      * ( 1 + (p_randVar("priceInputs",nCur)-1) $ randProbs(inputs) )
$endif.sp
                                   * v_buy(inputs,t,nCur,s);
```

Figure 172:

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

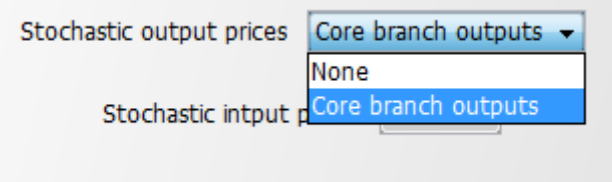


Figure 173:

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

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

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 Behavior

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

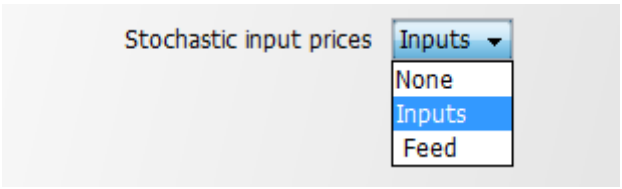


Figure 174:

```
$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;

$endif.stochPrices
```

Figure 175:

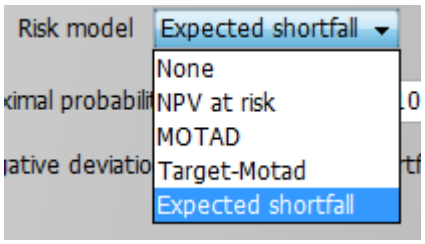


Figure 176:

run 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 maximizes a linear combination of the expected NPV and the expected mean negative deviation from the NPV.

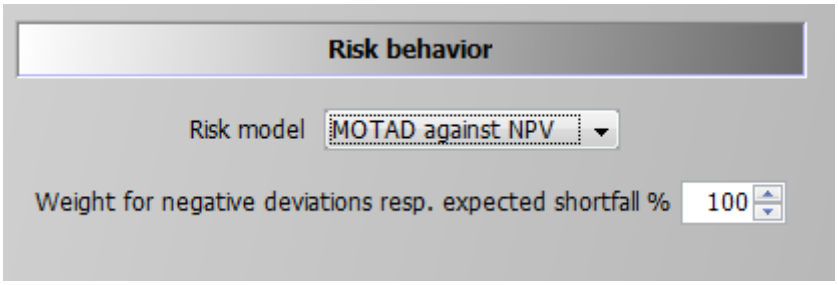


Figure 177:

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:

```
*
*    --- negative deviaton from NPV
*
   negDevNPV_(nCur) $ t_n("%lastYearCalc%",nCur) ..
         v_objeN(nCur) + v_negDevNPV(nCur)  =G= v_objeMean;
```

Figure 178:

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

```
*
*    --- Expected mean deviation
*
   expNegDevNPV_ ..
         v_expNegDevNPV =E= sum(nCur $ t_n("%lastYearCalc%",nCur), v_negDevNPV(nCur)*p_probN(nCur));
```

Figure 179:

determined in our stage contingent approach endogenously, symmetry makes limited sense. The expected mean deviation is calculated as:

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)
```

Figure 180:

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

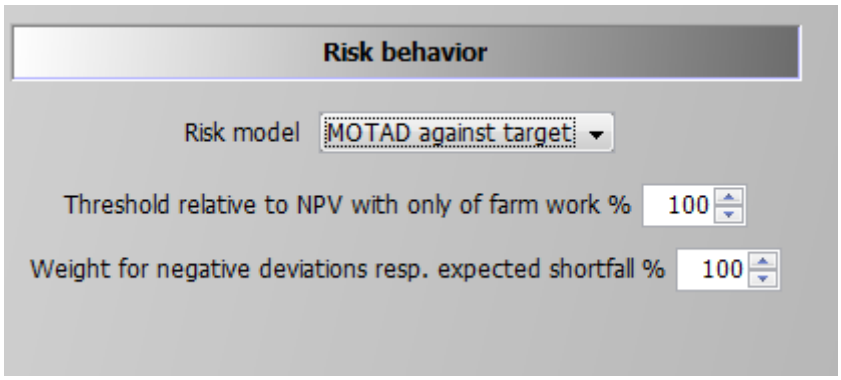


Figure 181:

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 graphical user interface. This effects the following equation:

```
*
*    --- negative deviaton from given limit
*
    shortFall_(nCur) $ (t_n("%lastYearCalc%",nCur) $ (p_npvAtRiskLim gt 1))  ..
          v_objeN(nCur) + v_shortFall(nCur) =G= p_npvAtRiskLim;
```

Figure 182:

Using this information the expected shortfall is defined:

```
*
*    --- Expected shortFall
*
    expShortFall_ $ ( p_expShortFall or p_maxShortFall ) ..
          v_expShortFall =E= sum(nCur $ t_n("%lastYearCalc%",nCur), v_shortFall(nCur)*p_probN(nCur));
```

Figure 183:

The expected shortfall enters then the objective function:

```
$ifi %stochProg%==true - v_expShortFall * p_negDevPen  $ (not p_expShortFall)
```

Figure 184:

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:

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:

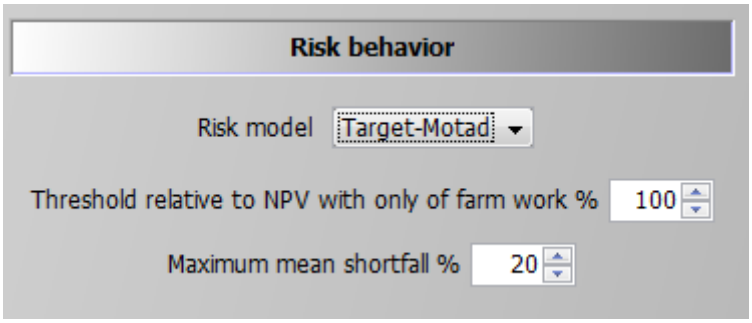


Figure 185:

```
*
*     --- maximum mean short fall definition
*
    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;
```

Figure 186:

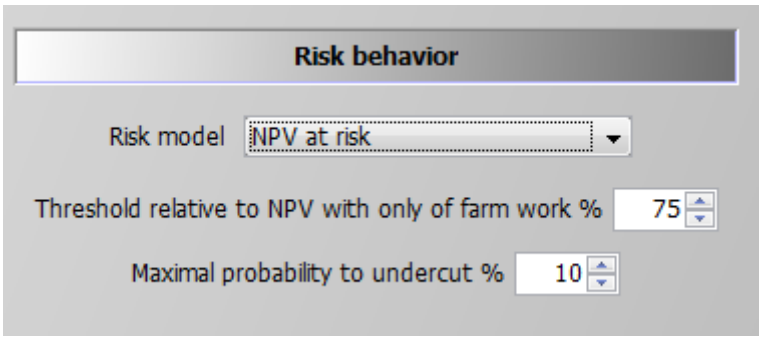


Figure 187:

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 state of nature, 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:

```
*
*    --- binary trigger for npv of each final leaf to undercut pre-defined lower limit
*
   npvAtRisk_(nCur) $ (t_n("%lastYearCalc%",nCur) $ (p_npvAtRiskLim gt 1))  ..
        v_objeN(nCur) =G= p_npvAtRiskLim - v_npvAtRisk(nCur) * p_npvAtRiskLim;
```

Figure 188:

If v_*npvAtRisk* is zero, the objective value (LHS) fo r 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:

```
*
*    --- sum of probabilities of leaves under cutting pre-defined lower limit of NPV
*        is not allowed to exceed a certain threshold probability p_maxProb
*
   maxProbNpvAtRisk_ $ ( ( (p_npvAtRiskLim gt 1) or p_expShortFall) $ p_npvAtRiskmaxProb) ..
      sum(t_n("%lastYearCalc%",nCur), v_npvAtRisk(nCur) * p_probN(nCur)) =L= p_npvAtRiskmaxProb;
```

Figure 189:

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 maximizes 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

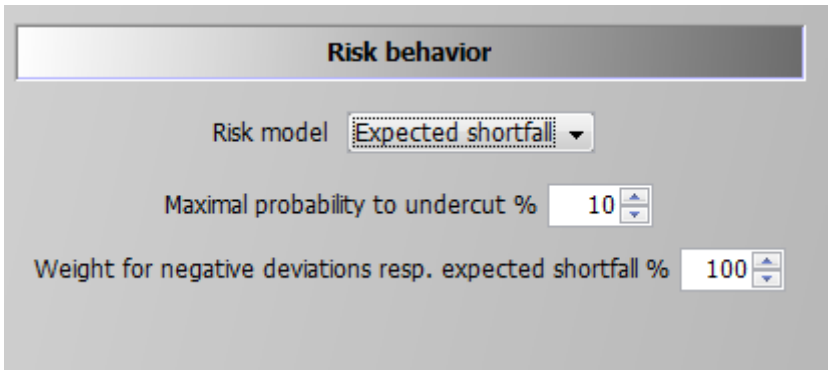


Figure 190:

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):

```
*
*    --- sum of probabilities of leaves under cutting pre-defined lower limit of NPV
*        is not allowed to exceed a certain threshold probability p_maxProb
*
   maxProbNpvAtRisk_ $ ( ( (p_npvAtRiskLim gt 1) or p_expShortFall) $ p_npvAtRiskmaxProb) ..
      sum(t_n("%lastYearCalc%",nCur), v_npvAtRisk(nCur) * p_probN(nCur)) =L= p_npvAtRiskmaxProb;
```

Figure 191:

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 size must be zero as well. On the other hand, if *v_npvAtRisk* is unity, i.e. the final leaves's 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.

```
*
*   --- Expected shortFall is zero if the related trigger v_npvAtRisk is not set
*
   shortFallTrigger1_(nCur) $ ( t_n("%lastYearCalc%",nCur) $ p_expShortFall ) ..
         v_shortFall(nCur) =L= 1.E+7 * v_npvAtRisk(nCur);
```

Figure 192:

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.

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:

```
*
*    --- all other cases must exceed the best short fall
*
   shortFallBound_(nCur)  $ ( t_n("%lastYearCalc%",nCur) $ p_expShortFall  ) ..
         v_objeN(nCur)     =G= v_bestShortFall+1 - v_npvAtRisk(nCur) * 1.E+7;
```

Figure 193:

```
*
*   --- the shortFall (= NPV) is equal to the objective value for those below 5%
*
   shortFallTrigger2_(nCur) $ ( t_n("%lastYearCalc%",nCur) $ p_expShortFall ) ..
         v_slackNPV(nCur)  =L= 1.E+7 * (1-v_npvAtRisk(nCur));
*
*    --- Expected shortFall is bounded by objective value at that leaf
*        (and exactly equal to it)
*
   shortFallTrigger3_(nCur) $ ( t_n("%lastYearCalc%",nCur) $ p_expShortFall ) ..
         v_shortFall(nCur) + v_slackNPV(nCur) =E= v_objeN(nCur);
```

Figure 194:

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:

```
*
*    --- Expected shortFall
*
   expShortFall_ $ ( p_expShortFall or p_maxShortFall ) ..
         v_expShortFall =E= sum(nCur $ t_n("%lastYearCalc%",nCur), v_shortFall(nCur)*p_probN(nCur));
```

Figure 195:

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.

GHG Accounting and derivation of Marginal Abatement cost

!!! danger “Deprecated” This part of the model is deprecated!

The original version with its focus on milk and GHGs was developed as milk accounts for about one sixth of agricultural revenues in the EU, being economically the most important single agricultural product. Dairy farms also occupy an important share of the EU’s agricultural area. They are accordingly also important sources for environmental externalities such nutrient surpluses, ammonia and greenhouse gas (GHG) emissions or bio-diversity, but also contribute to the livelihood of rural areas. With regard to GHGs, dairy farming accounts for a great percentage of the worlds GHG emissions of CO2, N2O and CH4 (FAO 2006, 2009), and is hence the most important single farm system with regard to GHG

```
$ifi %stochProg%==true + v_expShortFall * p_negDevPen  $ p_expShortFall;
```

Figure 196:

emissions. Given the envisaged rather dramatic reduction of GHGs, postulated by the recent climate agreement, it is therefore highly probable that agriculture, and consequently dairy farms, will be integrated in GHG abatement efforts. Any related policy instruments, be it a standard, a tax or tradable emission right, will require an indicator to define GHG emissions at farm level. Such an indicator sets up an accounting system, similar to tax accounting rules, which defines the amount of GHG emissions from observable attributes of the farm such as the herd size, milk yield, stable system, cropping pattern, soil type or climate. The interplay of the specific GHG accounting system and the policy instrument will determine how the farm will react to the policy instruments and thus impact its abatement costs, but also the measurement and control costs of society for implementing the policy. The objective of the paper is to describe the core of a tool to support the design of efficient indicator by determining private and social costs of GHG abatement under different GHG indicators. It is based on a highly detailed farm specific model, able to derive abatement and marginal abatement cost curves with relation to different farm characteristics, a highly detailed list of GHG abatement options and for different designed emission indicators.

The calculation of the farm specific greenhouse gas emissions is defined by different specified GHG indicators (GHG calculation schemes). The indicators differ in degree of aggregation and feasibility of required production data. In the next sections the derivation procedures are described for all indicators and divided concerning gas types.

Short conceptual explanation of the implemented indicator schemes:

The different indicators are mainly based on the IPCC (2006) guidelines which comprise so-called tiers of increasing complexity to calculate GHG emission. Tier 1 provides the simplest approach to account emissions using default parameters. Tier 1 is used as far as possible to define our simplest indicator, *actBased*, where emission factors are linked to herds and crop hectares. The exemptions from the IPCC methodology are manure management and fertilization, for those IPCC links emission factors to organic and synthetic fertilizer amounts. Whereas average excretion and fertilizer application rates to derive per animal and per hectare emission factor coefficients are assumed in the model.

A somewhat more complex indicator called *prodBased* links emission factors to production quantities of milk and crop outputs. In general, at the assumed average yields, *actBased* and *prodBased* yield the same overall emissions. Compared to the activity based indicator, farmers have somewhat more flexibility as they for example might switch between different grass land management intensities to abate emissions.

A more complex and also presumably very accurate indicator is called *NBased*. Values for enteric fermentation are calculated from the requirement functions, for energy based on IPCC guidelines, which also drive the feed mix. For manure management, emissions are linked to the amount of manure N in specific storage types in each month. For fertilization emission factors are linked to distributed nitrogen and are differentiated by application techniques. The indicator thus gives the farmer the chance to abate nitrogen losses by changing storage types, storage periods or fertilization application technique, beside changes in herd sizes, herd structure and cropping pattern.

An intermediate indicator linked to production quantities is called *genProdBased*. The emission factors are linked mainly to output quantities and as far as possible derived from the indicator NBased. The differences stem from the calculation of emissions from enteric fermentation and manure management: the Tier 1 default values are either directly replaced by Tier 3 values (enteric fermentation) or Tier 3 values for manure management are used in conjunction of assumed shares for manure storage and application. Specifically, the indicator introduces milk yield dependent emission factors. This reflects that higher milk yields reduce per litre emissions by distributing maintenance and activity need per cow over a larger milk quantity.

The most complex and accurate indicator scheme is the reference indicator called *refInd*. It is an enhancement of the *NBased* indicator. The difference is that real feed intake of different feed compounds is recognized in order to implement the impact of feed digestibility on emissions from enteric fermentation. Moreover, the addition of fats and oils to feed is recognized to increase digestibility and lower methane from ruminant processes.

The different indicator calculation schemes (except of *NBased* and *refInd*) are listed in a sub-module named *indicators*. In this module some basic parameters and scale definitions for later emission calculation formulas are explained. The information is derived from IPCC (2006) (chapter 10 and 11), Dämmgen (2009), Velthof and Oenema (1997) and the RAINS model definition (for changes in NOx and NH3 fluxes concerning manure application type) (Alcamo et al., 1990).

The parameters and scalars shown above are described by the statements within the model code excerpt. These are basic emission factors which are later linked to the indicator schemes representing source specific emission conversion factors or compound rates. Concerning the specifications when running the model made by the user, different ways of GHG calculation are active, depending on the chosen indicator, *ind*. Hence, the resulting emissions by each simulated farm per year, *v_GHGEmissions(ind,t),* are either calculated on per head or hectare basis, per production quantity or on highly disaggregated way using the NBased or refInd indicator. The equation *GHGEmissions_(*,*,*,*)* facilitates an indicator and gas and source specific quantification of emissions according to the chosen indicator scheme, *v_GHGs(ind,emitters,gases,t)*. Therefore, *p_GHGEmissions(*,*,ind,emitters,gases)* are per head or hectare or per production unit specific emission parameters inserted for calculation the indicators

```
*  Parameter and scale definition for calculation of indicators:
*        Parameters for CH4 following IPCC 2006 (Tier1 and Tier2)

 scalar p_ASH "ash content of manure, normally 0.08 for cattle"
       /0.08/;

$onempty

 parameters p_Bo(herds) "maximum methane producing capacity for manure produced by livestock in m³CH4/kg"
          /
$iftheni %dairyHerd% == true
           cows       0.24
           heifs      0.18
           fcalvsRais 0.18
           fcalvsSold 0.18
           mcalvs     0.18
$endif
             /;

 parameters p_MCF(manStorage) " methane conversion factors for each manure management system, 0.17  means 17%"
          /storsub   0.17
           stornocov 0.17
           storstraw 0.10
           storfoil  0.0  /;

 parameters p_Ym(*) " methane conversion factor, table 10.12 in IPCC, 2006)"
          /
$iftheni %dairyHerd% == true
           cows   6.5
           heifs  6.5
           calvs  6.5
$endif
            / ;


*        Parameters for N2O following IPCC 2006 (Tier1 and Tier2)
*
* --- Table 10.21
*
 parameters p_EF3s(manStorage) "emission factor for direct N2O emissions from manure management system, kg N2O-N/kgN in manure system"
          / storsub   0.002
            stornocov 0.000
            storstraw 0.005
            storfoil  0.0   /;
```

Figure 197:

```
        scalar p_avDmMan "average dry matter content of cattle manure, 0.11 means 11%"
               /0.11/;

        parameter p_backCH4soil(crops) "background emission kg CH4(negative) from one ha of specific crop category"

               /winterCere    -1.5
                summerCere    -1.5
                winterRape    -1.5
                potatoes      -1.5
                idle          -1.5

       $iftheni %dairyHerd% == true
                maizsil       -1.5
                past33        -2.5
                gras20        -2.5
                gras29        -2.5
                gras34        -2.5
                idleGras      -1.5
       $endif
                /;

        scalar p_backN2Osoil "background emission kg N2O-N per ha. same for all crops and soils"
               /0.9/;

         parameter p_assumedMinShare(crops) /

       $iftheni %herd% == true
            winterCere    0.48
            summerCere    0.48
            winterRape    0.67
            potatoes      0.48

       $else
            winterCere    1.00
            summerCere    1.00
            winterRape    1.00
            potatoes      1.00
       $endif

       $iftheni %dairyHerd% == true
            MaizSil 0.25
            gras20  0.05
            gras29  0.25
            gras34  0.33
            past33  0.33
       $endif
         /;
```

Figure 198:

actBased, *prodBased* and *genProdBased*. Calculation procedures for the *NBased* and *refInd* indicator schemes are explained later.

```
*
*   --- GHG Emissions by indicators
*       indicators available are: actbased, prodbased, genProdBased, N-Based, refInd
*
    GHGEmissionsInd_(actInd(ind),tCur(t),nCur) $ t_n(t,nCur) ..

       v_GHGEmissions(ind,t,nCur) =e= sum( (emittors,gases), v_GHGs(ind,emittors,gases,t,nCur));

*
*   --- GHG Emissions by emittor and gas
*
    GHGemissions_(actInd(ind),emittors,gases,tCur(t),nCur) $ t_n(t,nCur) ..

       v_GHGs(ind,emittors,gases,t,nCur)=e=
*
*    --- accounting of emission amounts calculated by different indicators.
*        basic formula which depicts emission values for specific indicator schemes

*          --- per head and per ha

$ifthenI %dairyHerd%==true

      [    sum((actHerds(heifs,breeds,feedRegime,t,m)),
                v_herdSize(heifs,breeds,feedRegime,t,nCur,m)        * p_GHGEmissions(heifs,"perHead",ind,emittors,gases) )

         + sum((actHerds(cows,breeds,feedRegime,t,m)),
                v_herdSize(cows,breeds,feedRegime,t,nCur,m)         * p_GHGEmissions(cows,"perHead",ind,emittors,gases))

         + sum((actHerds("mCalvsRais",breeds,feedRegime,t,m)),
                v_herdSize("mCalvsRais",breeds,feedRegime,t,nCur,m) * p_GHGEmissions("mCalvsRais","perHead",ind,emittors,gases))

         + sum((actHerds("mCalvsRaisH",breeds,feedRegime,t,m)),
                v_herdSize("mCalvsRaisH",breeds,feedRegime,t,nCur,m) * p_GHGEmissions("mCalvsRaisH","perHead",ind,emittors,gases))

         + sum((actHerds("mCalvsSold",breeds,feedRegime,t,m)),
                v_herdSize("mCalvsSold",breeds,feedRegime,t,nCur,m) * p_GHGEmissions("mCalvsSold","perHead",ind,emittors,gases))
      ] / card(herdm)
$endif

         + sum( (c_s_t_i(curCrops(crops),plot,till,intens),s),
                   v_cropHa(crops,plot,till,intens,t,nCur,s) * p_GHGEmissions(crops,"perHa",ind,emittors,gases) * p_prob(s))
*
*          --- basis are production quantities
*
         + sum( (prods,s),              v_prods(prods,t,nCur,s) * p_GHGEmissions(prods,"perProdQuant",ind,emittors,gases) * p_prob(s))
```

Figure 199:

For each indicator scheme different parameters per head, hectare or per production quantity are defined in the indicator module and will be implemented into the calculation formula in the previous equation following the chosen indicator in the model run. Hence, if *NBased* is chosen, only the last summand is active, leaving the lines at the beginning of the equation disregarded as *NBased* scheme does not hold any emission parameters per hectare, head or per production quantity.

The detailed calculation schemes for the different chosen indicators are illustrated in the following subchapters.

GHG indicators

Conceptually, FARMDYN is a microeconomic supply side model for "bottom-up" analysis based on a programming approach, i.e. constrained optimization. A bottom-up approach principally connects sub-models or modules of a more complex system to create a total simulation model, which increases the complexity but hopefully also the realism (Davis, 1993). On farm bio-economic processes are described in a highly disaggregated way. Whereas so-called engineering models also optimise farm level production systems, in that type of model possible changes in management or for instance GHG mitigation options are predefined (different feed rations, defined N intensities...), implemented separately and ordered concerning their derived single measure mitigation costs to explore the

MAC curve. Contrary to that, the LP-approach of our Supply Side model enables to solve for optimal adjustments of production processes by continuous variation of decision variables, such that the optimal combination of mitigation measures is derived. The same argument holds for analyzing shocks in prices or other type of policy instruments. With regard to GHGs, a further advantage of the approach relates to interaction effects between measures with regard to externalities as different gases and emission sources along with their interactions are depicted (see e.g. Vermont and DeCara, 2010). Market prices are exogenous in supply side models such that market feedback is neglected, contrary to so called equilibrium models which target regional or sector wide analyses (such as e.g. the ASMGHG model used by Schneider and McCarl 2006).

prodBased Indicator

As first indicator the production based, *prodBased*, is described as other more general indicators use some calculations build for it, but those more simple indicators use average data.

The different GHG emission parameters, *p_GHGEmissions,* per production quantity are implemented in the submodule *indicators*. For *idle* only the background emissions per hectare are recognized. Emission parameters for all other crops are derived by using the equations [7] shown below, calculating N inputs from yield level and average N content of crops, *p_nContent*, and multiplying this by the above shown standard emission factors for direct and indirect GHG emissions. Additionally, background emissions for arable soils are considered *p_backCH4Soil(crops), p_backN2OSoil(crops).* To obtain per unit of product emission parameters, *p_GHGEmissions(prods,…),* the calculated total GHG amount per hectare is divided by the output quantity per hectare, *p_OCoeff(crops,prods,t).*

```
p_GHGEmissions(idle,"perna","prodBased",emittors,gases)   =  ((- 1.5*21) $(sameas (emittors,"backSoil") and sameas (gases,"CH4")))
                                                              + ((p_backN2OSoil * 44/28 * 310)$(sameas (emittors,"backSoil") and sameas(gases,"N2O"))) ;

p_GHGEmissions(prods,"perProdQuant","prodBased",emittors,gases)  $ sum( crops $ (      p_nutContent(crops,prods,"N")
                                                                                    and sum( (soil,till,intens,t) $ p_OCoeffC(crops,soil,till,intens,prods,t),1)), 1)

                                 =  sum( crops $ p_nutContent(crops,prods,"N"),
                                    [[  p_nutContent(crops,prods,"N")*10 * (p_assumedMinShare(crops)  * ( 1 / (1-0.1))   $ (sameas (emittors,"synthApplic") and  sameas (gases,"N2O"))
                                                                           +  (1-p_assumedMinShare(crops)) * ( 1 / (1-0.4))   $ (sameas (emittors,"manApplic")  and  sameas (gases,"N2O"))
                                                                                                                              )

                                      --- direct N2O emissions from N application
                                          * (  p_EF1  $ (not sameas(crops,"past33"))
                                      --- direct N2O emissions from N application
                                          + p_EF3prp("past33") $ sameas(crops,"past33")
                                      --- return via atmospheric deposition
                                          + (   p_fracGasF * p_assumedMinShare(crops)                          $ sameas (emittors,"SynthApplic")
                                          + p_fracGasM("ApplSpread") * (1-p_assumedMinShare(crops))   $ sameas (emittors,"manApplic")
                                          ) * p_ef4
                                      --- leaching
                                          + p_fracLeach * p_ef5 )] * 44/28 * 310 ]
                                      --- background emission
                                          + ( p_backN2OSoil / (sum( (soil,till,intens,t), p_OCoeffC(crops,soil,till,intens,prods,t))/
                                                                sum( (soil,till,intens,t) $ p_OCoeffC(crops,soil,till,intens,prods,t),1))

                                           * 44/28 * 310) $ (sameas (emittors,"backSoil") and sameas (gases,"N2O"))

                                        + ( p_backCH4Soil(crops) / (sum( (soil,till,intens,t), p_OCoeffC(crops,soil,till,intens,prods,t))
                                                                                              / sum( (soil,till,intens,t) $ p_OCoeffC(crops,soil,till,intens,prods,t),1))
                                                                    * 21 )      $ (sameas (emittors,"backSoil") and sameas (gases,"CH4"))   )

                                       /sum( crops  $ p_nutContent(crops,prods,"N"), 1);
```

Figure 200:

For dairy cows the GHG emission factor per kg of milk, *p_GHGEmissions("milk",…)*is

[7] Based on IPCC (2006)

derived by taking default emission factors from IPCC (2006) from enteric fermentation (117 kg CH4) and manure management (21 kg CH4, 1.4 kg N2O) per animal and year and dividing the resulting CO2-equ. by an average yearly milk yield per cow (6000 kg).

For emission from heifers and calves, per head default emission factors from IPPC (2006) are taken. For emission parameter calculation of male calves and sold female calves, the residence time on farm is recognized (14 days on average). To calculate the default values of N2O emissions from herds, calculation functions 10.25, 10.26 and 10.30 of IPCC (2006) are filled with average weights (e.g. cow: 650kg) and excretion rates of the different herd categories taken from KTBL (2010).

```
* --- derived from calculations for 6000 litre cow derived from IPCC (Table 10.11) default Tier1 values.

  p_GHGEMissions("milk","perProdQuant","prodBased",emittors,gases)    = [(117*21) $( sameas(emittors,"entFerm") and sameas(gases,"CH4") )
                                                                       + (21*21)  $( sameas(emittors,"manStorage") and sameas(gases,"CH4"))
                                                                       + (1.4  * 310) $(sameas(emittors,"manStorage") and sameas (gases,"N2O"))
                                                                         ]/6000 * 1000;
*
* --- prodBased indicators for herds with the exemption of cows are calculated according to
*     the IPCC 2006 Tier1 approach using default values. heifer of average weight 350kg and calve 85kg (following KTBL2010)
*
*          (1) Methan from enteric fermentation and manure storage (*21)
*          (2) N2O from manure storage   (*310)
*
  p_GHGEmissions("heifs","perHead","prodBased",emittors,gases)        = (  57*21)         $ (sameas(emittors,"entFerm")    and sameas(gases,"CH4"))
                                                                        + (6*21)         $  (sameas(emittors,"manStorage") and sameas(gases,"CH4"))
                                                                        + (0.52 * 310)  $  (sameas(emittors,"manStorage") and sameas (gases,"N2O"));

  p_GHGEmissions("fCalvsRais","perHead","prodBased",emittors,gases)   = (57*21)$ (sameas(emittors,"entFerm") and sameas(gases,"CH4"))
                                                                        +(6*21) $ (sameas(emittors,"manStorage") and sameas(gases,"CH4"))
                                                                        +(0.13 * 310)$(sameas(emittors,"manStorage") and sameas (gases,"N2O"));
*
*         for calves, two weeks on farm (14/365)
*
  p_GHGEmissions("mCalvs","perHead","prodBased",emittors,gases)       = [  (57*21)      $ ( sameas(emittors,"entFerm")    and sameas(gases,"CH4"))
                                                                         + (6*21)       $ ( sameas(emittors,"manStorage") and sameas(gases,"CH4") )
                                                                         + (0.13 * 310) $ (sameas(emittors,"manStorage")  and sameas (gases,"N2O"))
                                                                         ]/365 * 14;

  p_GHGEmissions("fCalvsSold","perHead","prodBased",emittors,gases)   =  [(57*21)$ (sameas(emittors,"entFerm") and sameas(gases,"CH4"))
                                                                         +(6*21) $ (sameas(emittors,"manStorage") and sameas(gases,"CH4"))
                                                                         +(0.13 * 310)$(sameas(emittors,"manStorage") and sameas (gases,"N2O"))
                                                                         ]/365 * 14;
```

Figure 201:

No differentiation in GHG emission rates of the farm are made concerning storage and application techniques of manure and synthetic fertilizers. Moreover, storage time of manure as well as differences in emissions from applied fertilizer and manure N are not recognized by this indicator.

actBased Indicator

The *actBased* indicator denotes the simplest emission indicator implemented in the model approach of FARMDYN. It just multiplies activity data (hectare or heads) with specific default emission factors which are taken from IPCC (2006) on Tier 1 level. For example 117 kg CH4 per cow and year from enteric fermentation and 21 kg CH4 per cow from manure management. For N2O emissions per cow and year the default value of 1.4 kg is taken (see section *prodBased* indicator). For calves and heifers the default emission indicators are the same as for *prodBased*. Emission parameters from cropping activities are computed by the derivation scheme of the *prodBased* indicator taking average yield levels and fertilizer application rates.

Furthermore, no differentiation concerning emission rates of different intensity levels for grassland (different pasture types) is made.

```
* ------------------------------------------------------------------------------------------------------
*
*     (2) Act based, Tier 1
*
* ------------------------------------------------------------------------------------------------------

*
* --- copy per head coefficients
*
  p_GHGEmissions(herds,"perHead","actBased",emittors,gases) = p_GHGEmissions(herds,"perHead","prodBased",emittors,gases) $ (not sameas (herds,"cows"));
*
* --- for cows, use IPCC default values Tier 1
*
  p_GHGEmissions(cows,"perHead","actBased",emittors,gases)         = (117*21) $ (sameas(emittors,"entFerm") and sameas(gases,"CH4"))
                                                                       + (21*21)  $ (sameas(emittors,"manStorage") and sameas(gases,"CH4"))
                                                                       + (1.4  * 310) $(sameas(emittors,"manStorage") and sameas (gases,"N2O"));

$endif
*
* --- for idle, land continue to use per ha
*


  p_GHGEmissions(idle,"perHa","actBased",emittors,gases)   = p_GHGEmissions(idle,"perHa","prodBased",emittors,gases);

  p_GHGEmissions(crops,"perHa","actBased",emittors,gases) $ sum( (prods,soil,till,intens,t), p_oCoeffC(crops,soil,till,intens,prods,t))
    =  sum( (prods,soil,till,intens,t), p_GHGEmissions(prods,"perProduant","prodBased",emittors,gases) * p_oCoeffC(crops,soil,till,intens,prods,t))
     / sum( (prods,soil,till,intens,t) $ p_oCoeffC(crops,soil,till,intens,prods,t),1);

$iftheni %dairyHerd% == true

  p_GHGEmissions("gras20","perHa","actBased",emittors,gases) = p_GHGEmissions("gras29","perha","actBased",emittors,gases);
  p_GHGEmissions("gras34","perha","actBased",emittors,gases) = p_GHGEmissions("gras29","perha","actBased",emittors,gases);

$endif
```

Figure 202:

genProdBased Indicator

The indicator scheme called *genProdBased* also accounts for emissions of cropping activities. It is based on emission parameters per unit of product, taking the same equations as for the *prodBased* indicator.

For the calculation of emissions due to heifers and calves, the Tier 2 approach from IPCC (2006) is used, taking average weights (350kg for heifers and 90kg for calves), gross energy (GE) demands following KTBL (2010) and GE demands from Kirchgessner (2004). [8] The gross energy demands of cow species are given by requirement functions implemented in the sub-module *requ.gms,* considering functions for energy need for maintenance, activity, gross and lactation. GE demands used are calculated following equation from IPCC (2006) and taking a default digestibility of feed from IPCC of 60%.

```
*
* ------------------------------------------------------------------------------------------------------
*
*     (3) genProdBased, Mix of act based and prod based, differentiated for cows and heifers
*
* ------------------------------------------------------------------------------------------------------

$iftheni %dairyHerd% == true

* --- list of specific emission factors for cows differentiated by milk yield
*     for the genProdBased indicator

 parameter p_GHGEmissionsCowsYield(cows,emittors,gases) "emission parameters per kg milk for different milk yield levels devided to source and gastype" ;



         p_GHGEmissionsCowsYield(cows,emittors,gases)  =
*                                                  ---CH4
*                                        emissions from manure management following
*                                        equ. 10.24, 10.23 and 10.22 with dry matter content of volatile solids of 11% and a feed
*                                         digestibility of 60% as standard value as denoted by IPCC(2006) methodology
*                                        storage time of manure is meant to be half a year

                                         [[(((sum( (phase,breeds), p_grossEnergyPhase(cows,"HF",phase))
                                         *(1-0.6+0.04))
                                         * ((1-0.08)/18.45)
                                         )* p_Bo("cows")*0.67*p_MCF("storsub")/2)$(sameas(emittors,"manStorage")and sameas(gases,"CH4"))

*                                        emissions from enteric fermentation following equation 10.21 and 10.19

                                         + ((sum(phase, p_grossEnergyPhase(cows,"HF",phase))*p_ym("cows")/100)/55.65)
                                                                $(sameas(emittors,"entFerm")and sameas(gases,"CH4"))
                                           ]*21
*                                                  ---N2O
*                                        following equations from IPCC(2006) section 10.5
*                                        here the deviation by 2 for half year storage  (can be changed if emissin value do not fit any more)

                                         +  ((((((sum(phase, p_grossEnergyPhase(cows,"HF",phase))*(1-0.6+0.04))
                                         * ((1-0.08)/18.45))/11*100)/1000*4.7 )
                                         *(0.002+0.35*0.01+0.01*0.008))/2)
                                         *44/28*310)  $(sameas(emittors,"manStorage")and sameas(gases,"N2O"))

                                        ]/((cows.pos * 0.2 + 3.8)*1000);
```

Figure 203:

In case of emission calculations from dairy cows genetic potential is considered.

[8] This results in an average per head and year emissions of 1962 kg CH4-equ. for heifers and 504 kg CH4-equ. for calves.

Therefore, the gross energy demand of each different genetic yield potential is separately considered. Taking equations from IPCC Tier 2 approach leads to different per kg milk GE demand due to maintenance, lactation, gross and activity. The calculated emissions per cow are then divided by the milk yield potential, leading to a decline in GHG per kg milk for higher yield levels. For the calculation of emissions by dairy cows, a table with intensity dependent emission factors per kg milk is inserted into the module, displaying the parameter *p_GHGEmissionsCowsYield(cows,emittors,gases)*. The parameter value of whole emissions per kg milk diminishes from 0.74 CO2eq/kg for a cow with 4000 kg milk yield per year to 0.46 CO2eq/kg for a cow with 10000 kg. This decline in emissions per kg of milk with increasing milk yield is not linear as illustrated in the following Figure 10.

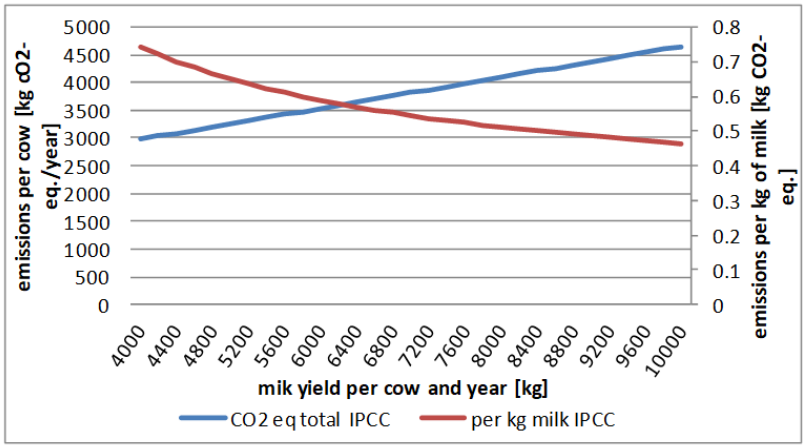


Figure 10: GHG emissions per cow and per kg of milk depending on milk yield potential. Source: Own illustration based on Kirchgessner (2004) and IPCC (2006)

In contrast to the simple *prodBased* indicator, the emissions from herds are now based on real, output depending energy requirements. Thus, it is resulting in different output level dependent emission factors per kg of milk and not a fix emission factor per kg of product disregarding the efficiency effects shown in Figure 10.

NBased Indicator

With difference to the other indicator definitions, the calculation formulas of the *NBased* indicator are not enclosed in the indicator module, but described in the basic template module. This is necessary because as it is a highly detailed and disaggregated indicator scheme, it calls for many model variables which differ between simulation steps.

As the *NBased* indicator up to now presents the reference indicator, the emissions which are emitted by each source of the production process (enteric fermentation, soils...) are calculated by this indicator in each simulation step. Hence, the GHG calculation is fragmented concerning the emission origins to credit the emissions to the single *emitters*. Summing up the source specific emissions lead to the overall GHGs, *v_GHGs(emitters,gases,t).*

CH4 accounting

The CH4 emissions by enteric fermentation are calculated using equation 10.21 of IPCC (2006) guidelines, which takes the gross energy intake of each livestock category, *p_grossEnergyPhase(...),* as basic variable to multiply it with the level, *v_herdsize,* of each category and a specific methane conversion factor, *p_YM.* By multiplying it with factor 21 CH4 emission are converted to CO2-equivalents.

For the accounting of methane from manure storage, the manure amount, storage period and differences in methane conversion factors, *p_MCF(manStorage),* between different manure storage types are recognized. This is done following equation 10.23 of IPCC (2006). The monthly manure amount in each storage type, *v_manInStorageType(...),* is multiplied by specific dry matter content, *p_avDmMan*, and by a methane emission factor (0.21 as a mix for cows and heifers) to obtain the m^3 of CH4 due to manure. In a next step multiplying it with 0.67 converts m^3 to kg methane. Because manure is quantified on monthly basis and CH4 emissions are also implemented on monthly basis, the methane conversion factors, *p_MCF(manStorage),* have to be divided by 5.66 to not overestimate monthly methane emissions from manure. [9]

The background emissions of methane from soils are derived by multiplying a crop specific emission parameter, *p_backCH4soil,* (taken from Dämmgen, 2009:p.315) with the activity level, *v_cropHa(crops,t,s).*

With the statement *$(sameas (emitters,"entFerm"))* the calculated CH4 emissions are assigned to the source enteric fermentation. The same is done for all other gases (N2O, CO2) and sources *(manStorage, backSoil, manApplic, syntApplic)* to be able to analyse source specific emission developments during the simulation runs.

N2O accounting

For N2O emissions from manure storage a differentiation between direct and indirect N losses is made. N amount in specific storage type, *v_NInStorageType(manStorage,t,m),* is multiplied by an emission factor for direct N2O-N flux from storage systems, *p_EF3s,* to get direct nitrous oxide emissions from storage systems in each month. Furthermore, indirect nitrous oxide emissions are considered following IPCC (2006) equations 10.27 and 10.29 to account for outgassing of NOx and NH3, *p_FracGasMS(manStorage),* and leaching *,p_FracLeachMS.*

[9] Derived by Table 10.17 (continued) of IPCC (2006). Yearly MCF =17, monthly MCF=3, 17/3=5.66

```
*            (1)  CH4 accounting

$ifthenI %dairyHerd%==true
        + {
*
*               --- CH4: enteric fermentation: methane output linked to energy need (IPCC, 2006, eq. 10.21)
*
                   sum( (actHerds(cows,breeds,feedRegime,t,m),phase),
                         p_grossEnergyPhase(cows,breeds,phase) * v_herdsize(cows,breeds,feedRegime,t,nCur,m)  * p_ym("cows")  / (100 * 55.65))

                +  sum(actHerds(heifs,breeds,feedRegime,t,m),
                         p_grossEnergyPhase(heifs,breeds,"gen")        * v_herdsize(heifs,breeds,feedRegime,t,nCur,m)       * p_YM("heifs") / (100 * 55.65))

                +  sum((actHerds(fcalvsrais,breeds,feedRegime,t,m),phase),
                         p_grossEnergyPhase("fcalvsrais",breeds,phase) * v_herdsize(fCalvsRais,breeds,feedRegime,t,nCur,m)  * p_YM("calvs") / (100 * 55.65))

                +  sum((actHerds("mcalvsRais",breeds,feedRegime,t,m),phase),
                         p_grossEnergyPhase("mCalvsrais",breeds,phase) * v_herdsize("mCalvsRais",breeds,feedRegime,t,nCur,m) * p_YM("calvs") / (100 * 55.65))

                +  sum((actHerds("mcalvsRaisH",breeds,feedRegime,t,m),phase),
                         p_grossEnergyPhase("mCalvsraisH",breeds,phase)* v_herdsize("mCalvsRaisH",breeds,feedRegime,t,nCur,m) * p_YM("calvs") / (100 * 55.65))

                +  sum((actHerds("mcalvsSold",breeds,feedRegime,t,m),phase),
                         p_grossEnergyPhase("mCalvsSold",breeds,phase) * v_herdsize("mCalvsSold",breeds,feedRegime,t,nCur,m) * p_YM("calvs") / (100 * 55.65))

                +  sum(actHerds(fCalvsSold,breeds,feedRegime,t,m),
                         p_grossEnergyPhase("fcalvsSold",breeds,"gen") * v_herdsize(fCalvsSold,breeds,feedRegime,t,nCur,m)  * p_YM("calvs") / (100 * 55.65))

              } * 1/card(herdM) * 21 $ (sameas(emittors,"entFerm") $ sameas(gases,"CH4") $ sameas(ind,"NBased"))
$endif

$ifthenI %dairyHerd%==true
*
*       --- indicator 5 (named refInd), as close as possible to IPCC 2006 equations and expanded by recognition
*           of digestibility and fat or oil additives to the ration
          + {
*
*               --- CH4: enteric fermentation: methane output linked usage of different feeding components to account for digestibility
*
                   sum( (possHerds,breeds,phase,intrYPer,feeds,s) $ (p_reqs(possHerds,breeds,phase,intrYPer,"enne")
                                                                     $ sum((feedRegime,m),actHerds(possHerds,breeds,feedRegime,t,m))),
                                v_feeding(possHerds,breeds,phase,intrYPer,feeds,t,nCur,s)  * p_feedsEmission(possHerds,feeds)
                    * p_prob(s))
             } * 21 $ (sameas(emittors,"entFerm") $ sameas(gases,"CH4") $ sameas(ind,"refInd"))

$endif

$ifthenI %herd%==true

*               ---  CH4 manure storage  (following IPCC, 2006, eq. 10.23):
*                    emissions related to manure quantity in storage in m3 converted to kg, multiplied with dry mater content,
*                    BO: 0,21 methane emission per kg manure as mix of cows and heifers and 0.67 parameter to cubic metre methane to kg methane
*
                +  sum( (manStorage,m),
                          v_volInStorageType(manStorage,t,nCur,m) * p_nutPerQubic("N")* 1000
                                * p_avDMMan * 0.21 * 0.67 * p_MCF(manStorage) /5.66                          )
                     * 21 $ (sameas(emittors,"manStorage") $ sameas(gases,"CH4") $ (sameas (ind,"refInd") or sameas(ind,"NBased")) )
$endif

*               ---   CH4 background emissions from soils (negative) following Dämmgen(2009,S.315)
                +  sum( (c_s_t_i(curCrops(crops),plot,till,intens),s),
                          v_cropHa(crops,plot,till,intens,t,nCur,s) * p_backCH4soil(crops) * p_prob(s)                                  )
                     * 21 $ (sameas(emittors,"backSoil") $ sameas(gases,"CH4") $ (sameas (ind,"refInd") or sameas(ind,"NBased")) )
```

Figure 204:

```
*
*               --- N2O manure storage:
*                   calculated on basis of N amounts in specific manure storages following IPCC(2006)

              + sum( (manStorage,m),

*                        --- direct N2O-N kg from storage (eq.10.25)

                          (v_volInStorageType(manStorage,t,nCur,m)/2 * p_nutPerQubic("N"))  * (p_EF3s(manStorage)/6)

*                        ---  indirect N2O-N kg from outgassing  (eq.10.27)

                         + (v_volInStorageType(manStorage,t,nCur,m)/2 * p_nutPerQubic("N")) * (p_FracGasMS(manStorage)/6)*p_EF4

*                        ---  indirect N2O-N kg from leaching    (eq.10.29)

                         + (v_volInStorageType(manStorage,t,nCur,m)/2 * p_nutPerQubic("N")) * (p_FracLeachMS/6) * p_EF5

                    )

                  * 44/28 * 310  $ (sameas(emittors,"manStorage") $ sameas(gases,"N2O") $ (sameas (ind,"refInd") or sameas(ind,"NBased")))
```

Figure 205:

The next excerpt of the model code represents the detailed N2O derivation concerning manure and synthetic fertilizer application on agricultural soils. Direct nitrous oxide emissions from soils follow the equation 11.1 and relating auxiliary calculations from IPCC (2006). The nitrous oxide amounts produced by cropland, *N2O_Inputs*, depends on the applied manure N amounts to the specific crops, on the application technique, *v_nManDist(Crops,ManApplicType,t,s,m),* and the N amount applied by synthetic fertilizer, *v_nSyntDist(Crops,syntNFertilizer,t,s,m).* Applied manure N is multiplied by an application specific increase factor, *p_n2OIncreaseFact,* which is higher for non-surface application techniques (see RAINS model). The sum of both is then multiplied by the N input dependent conversion factor for croplands, *p_EF1.* The same is done for calculation direct N2O emissions from N deposited to pasture, *past33*, using a pasture specific conversion parameter, *p_EF3prp("past33").*

The calculation of indirect N2O emissions from atmospheric deposition, leaching and runoff is corresponding to equations 11.9 and 11.10 from IPCC (2006) guidelines.

```
*               --- N2O from manure and synthetic fertilizer application (for NBased and Refind)
                + {
                  sum( (c_s_t_i(curCrops(crops),plot,till,intens),s,m),
                     [

$ifthenI %herd%==true
                        sum( (ManApplicType,manType) $ (v_manDist.up(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m) ne 0),
                            v_manDist(crops,plot,till,intens,ManApplicType,manType,t,nCur,s,m) * p_nutInMan(manType,"N")
                              * [
                                  + p_n2OIncreaseFact(ManApplicType)
                                        * (
*                                       --- N2O_Inputs kg N2O-N from overall N inputs (organic and synthetic)
*                                           on non-pasture
                                              p_EF1  $ (not sameas (crops,"past33"))

*                                       --- N2O_prp kg N2O-N from N deposition on pasture
                                            + p_EF3prp("past33") $ sameas(crops,"past33")
                                        )   ]

                          )  $ sameas(emittors,"manApplic")
*
*               --- N2O_atd kg N2O-Nfrom atmospheric desposition (eq. 11.9)
                    + v_nutLossApplic("NH3Man",t,nCur,m)/card(s)    * p_EF4 $ sameas(emittors,"NH3Man")

$endif
                     + sum( syntFertilizer,
                           v_syntDist(crops,plot,till,intens,syntFertilizer,t,nCur,s,m) * p_nutInSynt(syntFertilizer,"N")
                             * [
*                                --- N2O_Inputs kg N2O-N from overall N inputs (organic and synthetic)
*                                   on non-pasture
                                  + p_EF1 $ (not sameas (crops,"past33"))

*                                 --- N2O_prp kg N2O-N from N deposition on pasture
                                  + p_EF3prp("past33") $ sameas(crops,"past33")
                               ]
                          ) $ sameas(emittors,"synthApplic")

*               --- N2O background emissions per year from plots
*                  following Velthof and Oenema(1997) (IPCC-Werte are judged too high)
                       + v_cropHa(crops,plot,till,intens,t,nCur,s)/card(m) * p_backN2OSoil  $ sameas(emittors,"backSoil")
                  ] * p_prob(s)                )
              + sum(m,
*
*                   --- N2O_atd kg N2O-Nfrom atmospheric desposition (eq. 11.9)
                    + v_nutLossApplic("NH3Min",t,nCur,m)    * p_EF4 $ sameas(emittors,"NH3Min")
*
*                   --- N2O_L kg N2O-N from leaching and runoff (eq. 11.10)
                    + v_nutLossApplic("NLeach",t,nCur,m)* p_EF5 $ sameas(emittors,"NLeach")          )
             )  * 44/28 * 310  $ (sameas(gases,"N2O") $ (sameas (ind,"refInd") or sameas(ind,"NBased")))     ;
```

Figure 206:

The emission factor for background emissions of N2O from soils are not taken from IPCC (2006) methodology because these are valued to high, as they are from a study based on peat soils with very high volatilization rates. For this reason, data for background emission factors per ha of land, *p_backN2Osoil,* from Velthof and Oenema (1997:p.351) is used. They investigated an emission factor of 0.9 kg N2O-N per ha, as it was already shown in the declaration of basic emission parameters.

```
--- N2O background emissions per year from soils
    following Velthof and Oenema(1997) (IPCC-Werte are judged too high)

+ sum((crops,s), v_cropHa(crops,t,s)* p_backN2Osoil *p_prob(s)) * 44/28 * 310  $ (sameas(emittors,"backSoil") and sameas(gases,"N2O"))
```

Figure 207:

The parameters 44/28 and 310 which are multiplied with the calculated N2O-N amounts by each source are on the one hand the conversion factor from N2O-N to N2O (44/28) and the global warming potential (310) of nitrous oxide to calculate the corresponding CO2-equ.

refInd Indicator

As the *NBased* indicator, the *refInd* Indicator calculations are not enclosed in the indicator module, but described in the basic template module. This is necessary because it is the most detailed and disaggregated indicator scheme, it calls for many model variables which differ between simulation steps.

The reference indicator is the most disaggregated emission accounting scheme implemented in the model approach. It mainly bases on the calculation mechanisms of the former explained *NBased* indicator. Enhancements are made concerning the consideration of differences in feed compound digestibility, the addition of fat or oils as well as the impact on methane emissions from ruminant fermentation. The calculation of emissions from soils, manure and fertilizer application and manure storage are equal to the accounting procedure of the *NBased* indicator.

Methane calculations only differ between animals due to different enteric fermentation. The CH4 emissions from intestine processes are calculated by the following:

```
*           (1)  CH4 accounting

$ifthenI %dairyHerd%==true
        +  {
*
*               --- CH4: enteric fermentation: methane output linked to energy need (IPCC, 2006, eq. 10.21)
*
                   sum( (actHerds(cows,breeds,feedRegime,t,m),phase),
                        p_grossEnergyPhase(cows,breeds,phase) * v_herdsize(cows,breeds,feedRegime,t,nCur,m)  * p_Ym("cows")  / (100 * 55.65))

                +  sum(actHerds(heifs,breeds,feedRegime,t,m),
                        p_grossEnergyPhase(heifs,breeds,"gen")        * v_herdsize(heifs,breeds,feedRegime,t,nCur,m)       * p_Ym("heifs") / (100 * 55.65))

                +  sum((actHerds(fcalvsrais,breeds,feedRegime,t,m),phase),
                        p_grossEnergyPhase("fcalvsrais",breeds,phase) * v_herdsize(fCalvsRais,breeds,feedRegime,t,nCur,m)  * p_Ym("calvs") / (100 * 55.65))

                +  sum((actHerds("mcalvsRais",breeds,feedRegime,t,m),phase),
                        p_grossEnergyPhase("mCalvsrais",breeds,phase) * v_herdsize("mCalvsRais",breeds,feedRegime,t,nCur,m) * p_Ym("calvs") / (100 * 55.65))

                +  sum((actHerds("mcalvsRaisH",breeds,feedRegime,t,m),phase),
                        p_grossEnergyPhase("mCalvsraisH",breeds,phase)* v_herdsize("mCalvsRaisH",breeds,feedRegime,t,nCur,m) * p_Ym("calvs") / (100 * 55.65))

                +  sum((actHerds("mcalvsSold",breeds,feedRegime,t,m),phase),
                        p_grossEnergyPhase("mCalvsSold",breeds,phase) * v_herdsize("mCalvsSold",breeds,feedRegime,t,nCur,m) * p_Ym("calvs") / (100 * 55.65))

                +  sum(actHerds(fcalvsSold,breeds,feedRegime,t,m),
                        p_grossEnergyPhase("fcalvsSold",breeds,"gen") * v_herdsize(fCalvsSold,breeds,feedRegime,t,nCur,m)  * p_Ym("calvs") / (100 * 55.65))

             } * 1/card(herdW) * 21 $ (sameas(emittors,"entFerm") $ sameas(gases,"CH4") $ sameas(ind,"NBased"))
$endif
```

Figure 208:

The feed use for each animal, *v_feeding(herds,phase,...)*, is multiplied by a specific emission parameter per kg of feed compound, *p_feedsEmission(herds,feeds), which depends* on different ingredients of the component and the animal category.

Therefore emission parameters for different feed ingredients are calculated following the below stated routine. Per kg feed emission parameters are derived from IPCC (2006) equations based on GE of specific feedstuff, *p_feedAttr(feeds,"GE")*.

```
    (5) refInd, recognition of digestibility of feed and addition of fats and oils

-------------------------------------------------------------------------------------------------------------------


--- calculation of emission parameters per kg of feed
    calculated by usage of IPCC equations for Energyintake if animals are feedet only with single feeds



  parameter p_feedsEmission(herds,feeds)     "kg methane emissions of methane per kg of feed";

  p_feedsEmission(herds,feeds)
     = p_feedAttr(feeds,"GE") * (6.5/100)  / 55.65 * 1000;


addition of fats and oils is implemented via percentage reduction of whole emissions per kg iserted in ration
  recognition of maximum addtiont of fats/oils of 8% of whole ration dry matter
 --------> da müssen noch die richtigen werte rein.
 parameter    p_oilemissred "emission reduction in percent per kg of oil in ration" /0.02/;
 parameter    p_fatemissred "emission reductin in percent per kg of fat in ration"  /0.019/;
```

Figure 209:

Source Specific Accounting of Emissions

To enable the model user to directly explore emission amounts and gas types allocated to different production process (enteric fermentation, manure CH4, CH4,~ and N2O from fertilizer or manure application and background emissions from soils) specific accounting functions are used. By using emission indicators they assign emissions to source and gas type. With this a detailed source specific emission analysis is possible. Consequently, farm management changes can be analysed also with respect to GHG mitigation efforts on the whole or part of the farm.

So far land use change, afforestation or change of tillage practices is not implemented in the overall model approach. This is why no CO2 accounting is implemented into the indicator schemes. For a higher resolution of land use practices and tillage procedures this has to be expanded due to potentials of carbon sequestration or release from soils. Additionally, intensity and production depending CO2 emissions from fuel use could be implemented.

Overview on calculation of Marginal Abatement Costs (MAC)

Marginal abatement costs (MAC) are defined in the model as the marginal loss of profits of a farm due to a marginal reduction of an emission amount (Decara & Jayet, 2001). In the model GHG emissions are measured in CO2 equivalents.

For our highly detailed template model, no closed form representation of the abatement costs exist, thus, the MAC can only be simulated parametrically. Specifically, in a first step the MAC is derived by introducing a step-wise reduced constraint on maximal GHG emissions. In the second step resulting changes in GHG emissions are related to loss of profits. [10] . As already mentioned in the introduction the resulting MAC curve is depending on the indicator used to calculate the emissions, the abatement strategies available to the firm and, further firm attributes such as market environment. With regard to the adoption of the farm to new emission constraints, the model template allows to analyse the amount and structure of chosen emission abatement options for each reduction level depending on farm type and chosen emission indicator. This helps the structural analyses of the level of abatement cost depending on the effective abatement strategies which are biased by the specification of the emission indicator.

Under a given indicator a stepwise reduction of the emission constraint leads to a stepwise reduction in farm profits. Relating the change in emissions to the changes in profits allows calculating the total and marginal abatement cost. Henceforth ε_{0j} stands for emissions emitted measured with indicator j under the profit maximal farm plan without any emission target and the zero characterizes the reduction level. To derive marginal abatement cost curves n reduction steps, each with the same reduction relative to the base ε_{0j}, are done, which leads to the objective values from π_{0j} to π_{0j}. Moreover, α_i denotes the relative reduction in step i compared to the baseline emissions. The maximal profit under reduction level of step i and indicator j is restricted by accounted emissions for the k decision variables according to:

$$\sum_k ef_{j,k} x_k \leq (1 - \alpha_i)\, \varepsilon_{0j}$$

In this equation ef_{jk} represents the emission factor attached to decision variable k under indicator$j$, i.e. the CO2 equivalent emission accounted per unit of variable$j$. The difference between π_{0j} - the profit without GHG restrictions - and π_{0j} measures the profit foregone due to the specific emission ceiling for the combination of reduction level of step i and indicator j. Therefore, the total *abatement costs* (AC) for the abatement of $\alpha_i \varepsilon_{0j}$ emissions are defined as:

$$AC_{i,j} = \pi_{0j} - \pi_{0j}$$

The change in profits from step $i-1$ to i is divided by the emission reduction from step to step to derive *marginal abatement costs*:

$$MAC_{i,j} = \frac{\pi_{i-1} - \pi_{i,j}}{\varepsilon i - 1 - \varepsilon i, j}$$

[10] Loss of profits are calculated by comparing new profits with those of the baseline scenario. These monetary losses occur due to higher emission constraints.

where

MAC_{ij} = marginal abatement costs for the reduction step from $(i-1)$ to i, using the indicator j
π_{ij} = value of objective function in simulation step i, using the indicator j
ε_{ij} = emission amount in simulation step i, calculated with the indicator j
i = step of simulation
α_i = amount of total percentage reduction of emission in step i compared to baseline
ε_{0j} = baseline emission of the farm without emission restriction
ef_{jk} = total amount of GHG emissions related to one unit of activity k

A stepwise reduction of the emission constraint leads to a sequence of changes in the farm program and related loss of profits. Using this information a farm specific MAC curves can be generated, which plot changes in profits against the GHG reduction.

Normalization of MACs

As shown above, the GHG calculation schemes have significant differences in detail of emission accounting. Consequently, the marginal abatement costs of GHG mitigation will be quite different. Furthermore, derived emission amounts for the very same farm production portfolio will not be the same for the different indicators. Because of this the directly calculated MACs are as well not comparable between the indicators as the MACs depend on indicator specific accounting rules.

For this reason MACs need to be normalized. In order to do this reference indicator is used. This indicator relates the mitigation costs induced by the single indicators to the *real* abated emission amounts calculated with the reference indicator. This enables the model user for example to make statements concerning cost efficiency in GHG abatement of different indicator schemes.

When comparing different emission indicators we face the problem that the MACs of each indicator relate to its specific GHG accounting rules. From a policy and farm perspective it is essential to know how much GHG is physically released from the farm to correctly assess costs and benefits from mitigation strategies. Thus, it is important not to use the probably rather biased ones which are accounted by a specific indicator. In an ideal world it would be possible to derive the *real* GHG emissions from the farm program. As this is not possible, a so-called reference indicator is constructed. It uses the best available scientific knowledge to derive from the farm program, i.e. based on all available decision variables, a total GHG emission estimate from the farm. The underlying calculation could be highly non-linear and complex and need not necessarily be integrated in the model template itself. Equally, it does not matter if it could be implemented in reality on a dairy farm given its measurement costs. It simply serves as a yard stick to normalize GHG emissions from different, simpler, but

more realistic and applicable indicators. Relating profit losses under different indicators and indicator-specific GHG emission targets to the GHGs abatement under the reference indicator r at the simulated farm program allows deriving normalized, comparable marginal abatement cost curves:

$$MAC_{i,j}^{norm} = \frac{\pi_{i-1,j} - \pi_{i,j}}{\varepsilon i - 1, r - \varepsilon i, r}$$

where

$\varepsilon_{i,j}$ = emission amount in simulation step i, calculated with reference indicator r
r = index for reference indicator
j = index for other specific indicator

Normalized MACs show under which indicator the highest efficiency is obtained, meaning that *real* abated emissions of the optimized production portfolios are calculated and related to the abatement costs.

The two calculations of MAC curves (normalized to reference indicator and not normalized) make it possible for the user to compare two different impacts of an emission abatement scheme. The not normalized calculated abatement cost curves will show the abatement reactions and the associated costs on farm level. This will show the charging of costs that will be induced to the different farm types through a crediting scheme because the not normalized MAC curves are the ones who drive the on farm decisions in abatement strategies. On the other hand, one can evaluate the cost efficiency of different emission indicators by the normalized MACs, because the calculated abatement amounts by a specific indicator can show great divergences to the real abatement efforts of the farm. The second task enables the model to evaluate different emission indicators concerning their real mitigation effect.

The Coefficient Generator

Concept and File Structure

The coefficient generator comprises a number of small modules, realized 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 realized 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 fertilizer 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:** Initializes 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: i**ncludes all data regarding the technical aspects of the biogas fermenter, the different inputs and their related biogas yields.
- **Fertilizing**: defines coefficients for various application techniques for organic and synthetic fertilizers.
- **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 fattners and sows**.**
- **Prices**: different default values are defined if prices for variables are not defined by the graphical user interface
- **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:

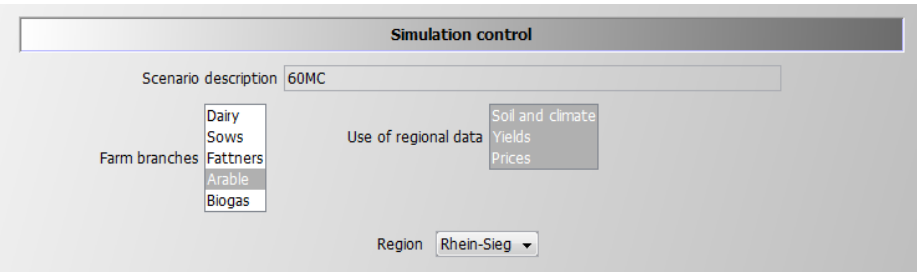


Figure 210:

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

```
<control>
  <Type>singlelist</Type>
  <Title>Region</Title>
  <Value>Rhein-Sieg</Value>
  <options>Rhein-Sieg,Steinfurt</options>
  <gamsName>region</gamsName>
  <tasks>all</tasks>
  <dependsOn>Use of regional data:Soil and climate,Yields,Prices</dependsOn>
</control>
```

Figure 211:

- *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;
$ifi "%useRegionalDataSoilAndClimate%"=="ON" $include 'regionalData\Climate_soil.gms'
```

Figure 212:

The file with the regional data, *regionalData\Climate_soil.gms*, comprises a cross-set between the regions and the climate zone. The current climate zone is only overwritten if an element in the cross-set for that region is found:

```
 set reg_climateZone(*,climateZone)
 /  "Rhein-Sieg".cz8 /;

curClimateZone(climateZone) $ reg_climateZone("%region%",climateZone)
   = reg_climateZone("%region%",climateZone);
```

Figure 213:

Similarly, soil shares entered via the interface or a batch file are only overwritten if for it at least one of the soil types data are entered:

Yield Data

Handling of the 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

```
table p_soilShareReg(*,soil)
                    l       m       h
"Rhein-Sieg"        0.8     0.2
;

p_soilShare(soil,"share") $ sum(soil1, p_soilShareReg("%region%",soil1)) = p_soilShareReg("%region%",soil);
```

Figure 214:

```
$if "%useregionaldataYields%"=="ON" $include 'regionalData\yields.gms'
```

Figure 215:

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.

```
TABLE p_cropYieldReg(*,*)

                  "Rhein-Sieg"
'WinterCere'       9.00
'SummerCere'       6.00
'WinterRape'       3.50
'Potatoes'        45.00
'Sugarbeet'       60.00
'MaizCorn'         9.80
'MaizCCM'         14.00
'Summerpeas'       3.50
'Summerbeans'      4.00
'Winterbarley'     7.00
'WheatGPS'        40.00
'MaizSil'         48.00
'gras20'          16.60
'gras29'          24.50
'gras34'          30.00
'past33'          45.00
;

p_cropYieldInt(acts,"yield") $ p_cropYieldReg(acts,"%region%") = p_cropYieldReg(acts,"%region%");
```

Figure 216:

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*:

Technical Realization

!!!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

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

Figure 217:

```
TABLE p_outputPricesReg(*,*)
                                    "Rhein-Sieg"
       WinterCere                            140
       WinterBarley                          140
       MaizCorn                              140
       SummerCere                            150
       WinterRape                            320
       SummerBeans                           140
       SummerPeas                            140
       Potatoes                              200
       SugarBeet                              35
       Pig                                  1.55
       PigletsSold                         57.10
       oldSow                             220.10
       milk                                30.00
       mCalv_HF                           100.00
       fCalv_HF                            40.00
       youngBull_HF                       550.00
       mCalv_SI                           210.00
       fCalv_SI                           120.00
       youngBull_SI                       800.00
       youngBullH_SI                     1100.00
;
```

Figure 218:

```
   set inputPrices / set.inputs,'Wage rate full time','Wage rate half time','Wage rate flexible hourly'/;

TABLE p_inputPricesReg(*,*)
                                    "Rhein-Sieg"
  'Wage rate full time'             12.00
  'Wage rate half time'             10.00
  'Wage rate flexible hourly'        6.00
  'ManCatt'
  'MaizSil'                         32.00
  'GrasSil'                         28.00
  'ConcCattle1'                    220.00
  'ConcCattle2'                    200.00
  'ConcCattle3'                    160.00
  'OilsForFeed'                   1150.00
  'Piglet'                          57.10
  'WinterCere'                     204.00
  'SummerCere'                     204.00
  'MaizCCM'                        110.00
  'WinterBarley'                   191.00
  'SoyBeanMeal'                    400.00
  'SoybeanOil'                    1150.00
  'MinFu'                          400.00
  'Diesel'                           0.70
  'ASS'                              0.31
  'AHL'                              0.05
  'seed'                             1.00
  'KAS'                              0.31
  'PK_18_10'                         0.10
  'KaliMag'                          0.34
  'Lime'                            59.00
  'Herb'                             1.00
  'Fung'                             1.00
  'Insect'                           1.00
  'growthContr'                      1.00
  'water'                            2.50
  'hailIns'                          9.34
;

   p_inputPrices(inputPrices,"price") $ p_inputPricesReg(inputPrices,"%region%") = p_inputPricesReg(inputPrices,"%region%");
```

Figure 219:

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

*
*  --- define input and output prices
*

   p_price(inputs,t,s) $ (not p_inputPrices(inputs,"price")) = p_price(inputs,"%firstYear%","s1") * ([1+%outputPriceGrowthRate%/100]**t.pos);
   p_inputprice(inputs,t,s) $ p_inputPrices(inputs,"price")      = p_inputPrices(inputs,"price") * ([1.+p_inputPrices(inputs,"growth rate")/100]**t.pos);

   p_outputPrices(prods) $ (not p_outputPrices(prods)) = p_inputPrices(prods,"price");
   p_price(prods,t,s)  = p_outputPrices(prods) * ([1+%outputPriceGrowthRate%/100]**t.pos);
```

Figure 220:

carefully chosen compilation and exploitation tests is used to check the code. The code is steered by a Graphical User Interface based on GGIG (ref., Java code) which also support result exploitation.

Overview of the Technical Realization

The model template and the coefficient generator are realized 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.

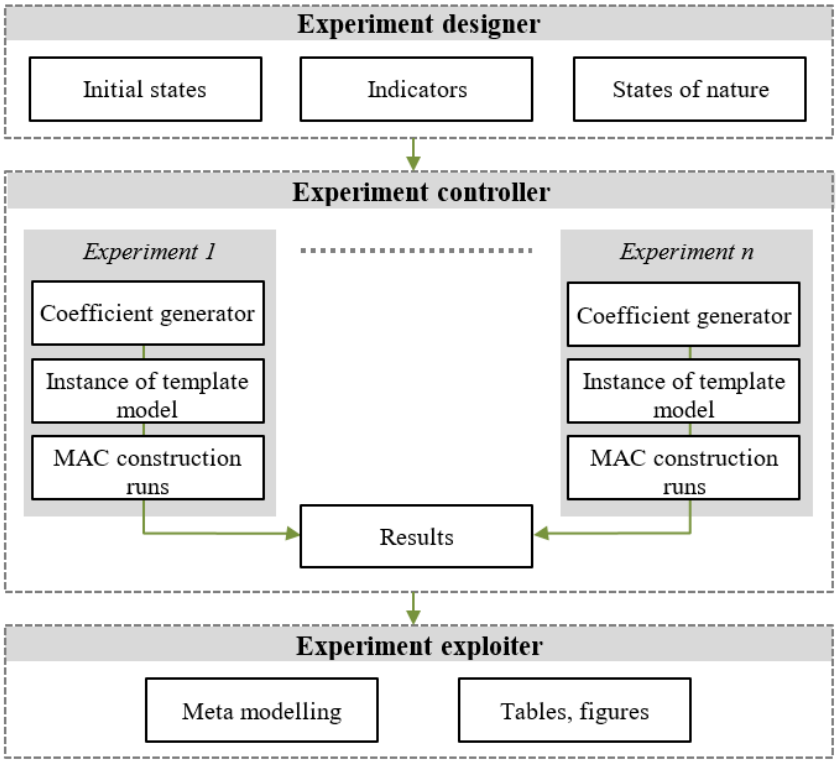


Figure 11: Overview of technical realization. Source: Own illustration

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 thousands 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 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 state-of-nature, and afterwards, the state contingent variables are copied to all other states-of-natures, 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 is 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 (*exp_starter.gms*) is shown below:

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

```
*
*    ---- option to allow for continous re-investment if investment costs per ha / hour / year
*         for a certain machine are below a pre-defined threshold: reduces number of binaries
*         can dramatically speed up solution of MIP version of model
*
$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;
```

Figure 221:

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 then 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
*
  parameter p_maxUsedStableSize(stableTypes);
  p_maxUsedStableSize(stableTypes)
     = smax(t1, sum(stables1 $ p_stableSize(stables1,stableTypes), v_stableUsed.l(stables1,t1)*p_stableSize(stables1,stableTypes)));

  p_maxUsedStableSize(stableTypes)
     = smin(stables1 $ (p_stableSize(stables1,stableTypes) ge p_maxUsedStableSize(stableTypes)*0.95),p_stableSize(stables1,stableTypes));

  v_stableInv.up(stables,hor,t)   $ sum(stableTypes $ (p_stableSize(stables,stableTypes) gt p_maxUsedStableSize(stableTypes)),1 ) = 0;
  v_buyStables.up(stables,hor,t)  $ sum(stableTypes $ (p_stableSize(stables,stableTypes) gt p_maxUsedStableSize(stableTypes)),1 ) = 0;
  v_stableUsed.up(stables,t)      $ sum(stableTypes $ (p_stableSize(stables,stableTypes) gt p_maxUsedStableSize(stableTypes)),1 ) = 0;

  v_stableInv.up(stables,hor,t)   $ sum(stableTypes $ ((p_stableSize(stables,stableTypes) lt p_maxUsedStableSize(stableTypes)*2/3) $ p_stableSize(stables,stableTypes)),1 ) = 0;
  v_buyStables.up(stables,hor,t)  $ sum(stableTypes $ ((p_stableSize(stables,stableTypes) lt p_maxUsedStableSize(stableTypes)*2/3) $ p_stableSize(stables,stableTypes)),1 ) = 0;
  v_stableUsed.up(stables,t)      $ sum(stableTypes $ ((p_stableSize(stables,stableTypes) lt p_maxUsedStableSize(stableTypes)*2/3) $ p_stableSize(stables,stableTypes)),1 ) = 0;

  v_buyStables.up(stables,hor,t) $ (not sum( (stables1,stableTypes,t1) $ (p_stableSize(stables,stableTypes) $ p_stableSize(stables1,stableTypes)),
          v_buyStables.l(stables1,hor,t1))) = 0;
```

Figure 222:

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 define logical ordering for the integer decisions. These equations are not necessarily truly restricting the solution space, they only re-inforce existing relations between variables. The additional equations often reduce the overall solution time by improving the branching more then 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 that a farm respectively a cow herd in t+1 implies that a farm respectively a cow herd in the previous year existed:

```
*
*   --- if no herd in t-1, then also no herd in the current year
*
    hasHerdOrder_(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));
```

Figure 223:

```
*
*   --- if the farm is there in t, it muse have been there in t-1
*
    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));
```

Figure 224:

The second one implies that working off-farm in an year t implies also working off-farm afterwards:

```
workOrder_(tCur(t)) $ tCur(t-1) ..
    v_workOffB(t) =G= v_workOffB(t-1);
```

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:

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:

```
*
*   --- convex combination for labour (only one type allowed)
*
    convLab_(tCur(t),nCur) $ (sum(workOpps(workType), v_labOff.up(t,nCur,workType)) $ t_n(t,nCur) ) ..
       sum(workOpps(workType), v_labOff(t,nCur,workType)) =E= v_labOffB(t,nCur);
```

Figure 225:

```
*
*    --- if the farm is there in t, it muse have been there in t-1
*
     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));
```

Figure 226:

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.

```
    hasAlwaysHerd_(breeds,t,m) $ actHerds("cows",breeds,t,m) ..
        v_hasAlwaysHerd =L= v_HasBranch("dairy",t);

    hasAlwaysCows_(breeds,tCur(t),m) $ actHerds("cows",breeds,tCur,m) ..
        v_hasAlwaysHerd =L= v_herdSize("cows",breeds,t,m);

    hasAlwaysLast_ $ sum( (breeds,m), actHerds("cows",breeds,"%lastYear%",m)) ..
        v_hasAlwaysHerd =E= v_HasBranch("dairy","%lastYear%");
```

Figure 227:

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:

```
*
*   --- restriction might help solver in branching algorithm, buy only a machine if there is
*       need for it
*
    machBuy_(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= (
                    sum(s, v_machNeed(machType,machLifeUnit,t,nCur,s) * p_prob(s)) $ tCur(t)
*
*         --- minus operating hours of weighted average over normal planning period
*             if beyond the normal planning period
*
                  + [sum( (t_n(t1,nCur1),s) $ ( (p_year(t1) lt p_year(t)) $ tCur(t1) $ isNodeBefore(nCur,nCur1)),
                                                 v_machNeed(machType,machLifeUnit,t1,nCur1,s)
                                                                * p_prob(s) * 1/(p_year(t)+5 - p_year(t1)) )
                    /sum( (t1,s) $ ( (p_year(t1) lt p_year(t)) $ tCur(t1)),     p_prob(s) * 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:

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

```
*   --- binary restriction (migh help solver): don t buy stable if the farm has no herd
*
    stableBuy_(stables,hor,t,nCur) $ ( (v_buyStables.up(stables,hor,t,nCur) gt 0) $ tCur(t) $ 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));
*
*   --- binary restriction (migh help solver): don t keep stables in use if the farm has no herd
*
    stableInvb_(stables,hor,tCur(t),nCur)
        $ ( (    (sum( t_n(t1,nCur1) $ isNodeBefore(nCur,nCur1), v_buyStables.up(stables,hor,t1,nCur1)) gt 0)
              or (sum( tOld, p_iniStables(stables,hor,tOld))))
                          $ v_hasHerd.up(t,nCur) $ 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));
```

Figure 228:

```
*
*   --- Silo Buy binary restriction (migh help solver): don t buy stable if the farm has no herd
*
    siloBuy_(silos,t,nCur) $ ( (v_buySilos.up(silos,t,nCur) gt 0) $ tCur(t) $ t_n(t,nCur)) ..

       v_buySilos(silos,t,nCur) =L= sum(animBranches,   v_HasBranch(animBranches,t,nCur));
```

Figure 229:

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)                             = %priorOperator%  (p_priorMax*6  +  %timeWeight%);
v_hasBranch.prior(branches,t)                = %priorOperator%  (p_priorMax*3  +  %timeWeight%);
v_hasBranch.prior("farm",t)                    = %priorOperator%  (p_priorMax*5  +  %timeWeight%);
```

Figure 230:

Generally, early years are given precedence:

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:

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

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

```
$setglobal timeWeight (card(t)-ord(t)+1)/card(t)  * p_priorMax * 10
```

Figure 231:

```
*
*   -- priorities for stable: try large stable first, and give precedence for cow over young over calv stable
*
    parameter p_priorStables(stables);

    p_priorStables(stables)
       = 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;
```

Figure 232:

```
v_buyStables.prior(stables,hor,t)              = %priorOperator%  (p_priorStables(stables)*hor.pos + %timeweight%);
v_stableInv.prior(stables,hor,t)               = %priorOperator%  [(p_priorStables(stables)*hor.pos + %timeweight%)*0.95];

v_laboffB.prior(t)                             = %priorOperator%  (p_priorMin * 0.9 +  %timeweight%);
v_laboff.prior(t,workType)                     = %priorOperator%  (p_priorMin * 0.8 +  %timeweight%);
```

Figure 233:

```
* *
*   -- for buildings and machinery, use investment price to define prorities
* *
    parameter p_rank;

    p_rank(buildings) = p_priceBuild(buildings,"%firstYear%");
    p_rank(machType)  = p_priceMach(machType,"%firstYear%");
    p_rank(silos)     = p_priceSilo(silos,"%firstYear%");

    set ranks(*); ranks(buildings) = YES; ranks(machType) = YES; ranks(silos) = yes;

$LIBINCLUDE rank p_rank ranks p_rank

    v_buySilos.prior(silos,t)              = %priorOperator%  (p_priorMin * 0.5 + p_rank(silos)/card(ranks)      * 0.19 +  %timeWeight%);
    v_buyBuildings.prior(buildings,t)      = %priorOperator%  (p_priorMin * 0.5 + p_rank(buildings)/card(ranks)  * 0.19 +  %timeWeight%);
*   v_buildingsInv.prior(buildings,t)      = %priorOperator%  [(p_priorMin * 0.5 + p_rank(buildings)/card(ranks) * 0.19   %timeWeight%)*1.05];
    v_buyMach.prior(machType,t)            = %priorOperator%  (p_priorMin * 0.5 + p_rank(machType) /card(ranks)  * 0.19 +  %timeWeight%);

    v_siCovComb.prior(silos,t,manStorage)  = %priorOperator%  (p_priorMin * 0.2 +  %timeWeight%);
```

Figure 234:

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:

```
*
*  --- financial results
*
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;
p_res(%1,%2,"hcon","sum","",tCur)        = p_hcon(tCur);
p_res(%1,%2,"hcon","sum","","mean")      = sum(tCur,p_hcon(tCur))/Card(tCur);
```

Figure 235:

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 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 randomized factor level combinations are smoothly distributed over the range of factor level permutations. [11]

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 [12].

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.

[11] 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).

[12] This is necessary to restrict sampling time but also guarantee 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

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
*
*-------------------------------------------------------------------------------
*

$setglobal r "%curdir%\..\r-2.15.1\bin\rscript.exe"

 file rIncFile / "%curDir%\rBridge\incFile.r" /;
 put rIncFile;
$setglobal outputFile   "%scrdirR%/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;
```

Figure 236:

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

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.

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:

```
*
* --- set correlation matrix
*
*     correlation coefficients are derived from data collections from AMI for prices and
*     from LWK-NRW (Milchviehreport NRW, verschiedene Jahrgänge, 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 Länder after
*     analysis on the "Landwirtschaftszählung 2010", results were aligned with results derived
*     from KTBL (2010,p.541).


*p_cor(factors,factors1) = uniform(-0.5,0.5);
*p_cor(factors,factors1)   = (p_cor(factors,factors1) + p_cor(factors1,factors))/2;

  p_cor("nCows","milkYield")              =  0.24;
  p_cor("nCows","cowsPerAk")              =  0.45;
  p_cor("nCows","stableYear")             = -0.1;
* p_cor("maxcowsPerHa","stockingRate") =  0;
* p_cor("stockingRate","stableYear")      = 0;
  p_cor("cowsPerAk","milkYield")          =  0.18;
  p_cor("milkPrice","conc1Price")         = 0.76;
* p_cor("milkPrice","winterCerePrice")           = 0;
  p_cor("winterCerePrice","conc1Price")           = 0.70;
  p_cor("summerCerePrice","conc1Price")           = 0.70;
  p_cor("summerCerePrice","winterCerePrice")      = 0.80;
*  p_cor("winterBarleyPrice","winterCerePrice")    = 0.80;
  p_cor("maizCornPrice","winterCerePrice")        = 0.70;
  p_cor("summerBeansPrice","winterCerePrice")     = 0.50;
  p_cor("summerBeansPrice","summerPeasPrice")     = 0.50;

  p_cor(allFactors,allFactors1) $ ( (not factors(allFactors)) or (not factors(allFactors1))) = 0;
  p_cor(factors,factors)  = 1;
*
* --- ensure symmetry
*
 p_cor(factors,factors1) $ ( not p_cor(factors,factors1))
     = p_cor(factors1,factors);
```

Figure 237:

```
 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 not exist %rexe% abort "R script.exe not found at " %rexe%;

$batinclude 'util\title.gms' "'execute %rexe% %rFile%'";
$if exist %rexe% execute "%rexe% %rFile% %curDir%\rBridge\incFile.r";

$endif.onlyCollect


 parameter p_doe(*,*);


 execute_load "%outputFile%_doe" p_doe;

 display p_doe;
 display "%outputFile%_doe";

 parameter p_testDoe;
 p_testDoe(factors) = sum(draws, p_doe(draws,factors))/card(draws);
 display p_testDoe;
```

Figure 238:

```
#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);
 library(lhs);
 library(gdxrrw);
 library(mc2d);
 library(mvtnorm);
 library(Matrix);
 library(gclus);
```

Figure 239:

p_n denotes the number of draws defined via the graphical user interface, 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 recognized or not).

```
 useCorr   <- Sys.getenv("useCorr")
 useColors <- Sys.getenv("useColors")
#
# --- determine GDX file with input data
#
 inputFile   = Sys.getenv("inputFile");
 if (!file.exists(inputFile)) stop(paste("inputFile ",inputFile," file doesn't exist! Check the code!"));
#
# --- number of draws as requested by GAMS
#
 n = rgdx.scalar(inputFile,"p_n");
#
# --- name of the set which comprises the scenario name
#
 scen_name = rgdx.set(inputFile,"scen_name",compress="true");
 scenName  <- levels(scen_name[scen_name$i == 'name',2])
#
# --- name of the set which comprises the factors
#
 factor_name = rgdx.set(inputFile,"factor_name",compress="true");
 set_to_load <- levels(factor_name[factor_name$i == 'name',2])
#
# --- load that set
#
 factors = rgdx.set(inputFile,set_to_load,compress="true");
#
# --- the number of factors is equal to the number of set elements
#
 LHSType <- "improvedLHS";
 k = nrow(factors);
```

Figure 240:

We decided to use the "*improvedLHS"* type for randomization [13] 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);
```

Figure 241:

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

[13] 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).

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 recognized 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<-acast(t, f1~f2, value.var="value")
    t<-as.matrix(t);
```

Figure 242:

To increase the possibility to randomize 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 randomized 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 randomized sample correlation matrix.

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

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

```
    bestFit <- 1;
   maxRunTime <- as.numeric(Sys.getenv("maxRunTime"))
    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 )
                      fit = fit + (c[i,j]-t[i,j])*(c[i,j]-t[i,j]);

       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 g
       }

       if ( meanDev < 1 ){
            print(paste(" draw :",iDraw," runTime ",round(runTime)," of ",maxRunTime,"seconds, mean sqrt of squared diff between
            runTime <- maxRunTime;
       }

    }
 }
```

Figure 243:

```
#
#    --- find nearest positive definite matrix
#

    t1<-nearPD(t);

    t <- as.matrix(t1$mat);
```

Figure 244:

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 randomized correlation matrix is then selected and stored by the program and automatically printed as a PDF-document for visualization. The PDF gives also information about average percentage bias of the randomized correlation matrix as well as the number of total draws which define the number of resulting sample experiments:

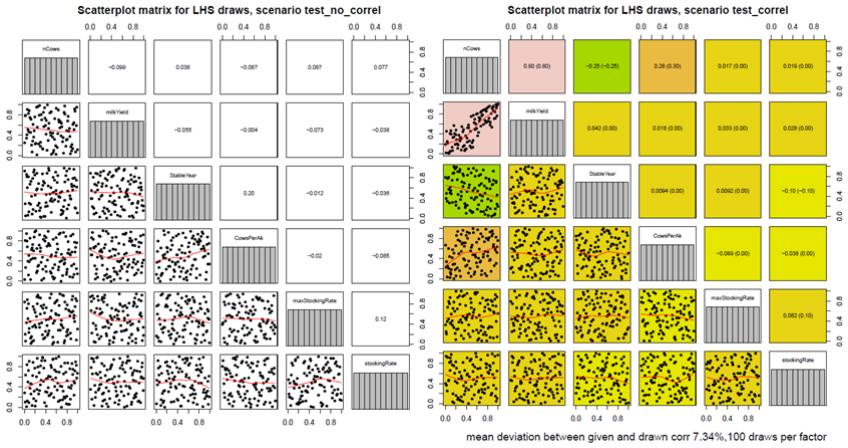


Figure 12: Scatterplot matrices for different LHS samples. With and without the recognition of factor correlations

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 visualized 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 randomized correlation matrix and the assumed correlation matrix is quantified by 7.34%, meaning, that on average, the randomized 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):

(1) $f(x) = \frac{1}{b-a}, a \leq b$

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 $[0; 1]$ (right graph):

(2) $f(x) = \frac{x-a}{b-a}, a \leq x \leq b$

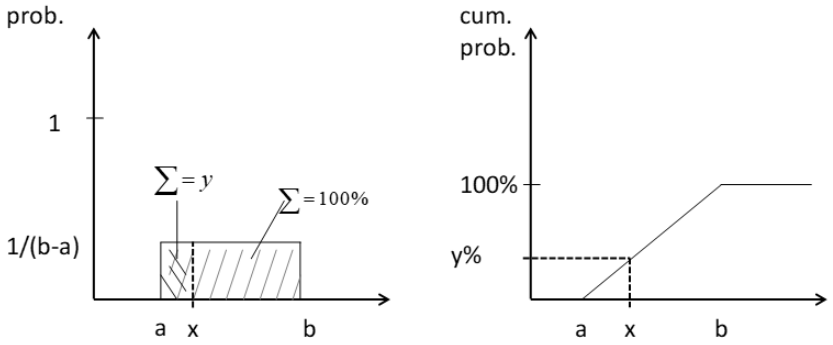


Figure 13: Density function and cumulative distribution function of an uniform distributed variable

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 $\int_a^x f(x)$ 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:

(3) $f(x) \cdot (b - a) + a = x$

The randomized 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 /;

 parameter p_scenParam(draws,scenItems) "Numerical values for the scenario specific items";

* ------------------------------------------------------------------------------------------------------------------------
*
*     Settings for dairy farms
*
* ------------------------------------------------------------------------------------------------------------------------
$iftheni.task1 "%task%" == "Experiments dairy"
*
* --- Emission Rights
*
      p_scenParam(draws,"emissionRight")   = p_doe(draws,"emissionRight") * (%emissionRightMax% - %emissionRightMin%) + %emissionRightMin%;
*
* --- Number of cows
*
      p_scenParam(draws,"nCows")           =  p_doe(draws,"nCows") * (%nCowsMax% - %nCowsMin%) + %nCowsMin%;
*
* --- Milk yield
*
      p_scenParam(draws,"milkYield")        = p_doe(draws,"milkYield") * (%milkYieldMax% - %milkYieldMin%) + %milkYieldMin%;
*
* --- milk yields go in 200 kg steps and are shown as 50,52,54 ... round accordingly
*
      p_scenParam(draws,"milkYield")        = round( p_scenParam(draws,"milkYield")/2)*2;

*
* --- Aks: derived from number of cows and Cows per AK
*
     p_scenParam(draws,"Aks")  = p_scenParam(draws,"nCows")
                               /[p_doe(draws,"CowsPerAk") * (%CowsPerAkMax% - %CowsPerAkMin%) + %CowsPerAkMin%];

*
* --- Aks in 0.5 steps, round accordingly
*
* p_scenParam(draws,"Aks")         = round( p_scenParam(draws,"Aks")*2) / 2;
*
* --- Max stocking rate
*
      p_scenParam(draws,"maxStockingRate") = p_doe(draws,"maxStockingRate") * (%maxStockingRateMax% - %maxStockingRateMin%) + %maxStockingRat

*
* --- Starting stocking rate, determines gras land endowment
*
      p_scenParam(draws,"cowsPerHaGras")    = p_doe(draws,"cowsPerHaGras") * (%cowsPerHaGrasMax% - %cowsPerHaGrasMin%) + %cowsPerHaGrasMin%;
*
* --- Starting stocking rate, determines arable land endowment
*
      p_scenParam(draws,"cowsPerHaArab")    = p_doe(draws,"cowsPerHaArab") * (%cowsPerHaArabMax% - %cowsPerHaArabMin%) + %cowsPerHaArabMin%;
```

Figure 245:

p_scenParam(draws,factor) gives the scenario parameter of 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 specific for each scenario:

Nevertheless correlations between factors are able to be recognized 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 labor input per cow and low milk yield levels or high numbers of cows per farm and only very low yielding phenotypes.

These scenario settings must be stored in a GAMS file which is then picked

```
*-------------------------------------------------------------------------------
*
* Define scenarios to run
*
*-------------------------------------------------------------------------------
*
*
* -- include file to generate, will be loaded by exp_starter
*
 file scenFile / incgen\curScen.gms /;

 set scenItems "Items which differ betwens scenarios, are stored via SETGLOBAL in include file"
 /
*
*  --- for all type of experiments (independent of farm specialisation)
*
   stableYear, wageRateFull, wageRateHalf, wageRateHourly, lastYear, aks

   winterCerePrice, summerCerePrice, winterRapePrice, maizCornPrice, summerBeansPrice,
   summerPeasPrice, potatoesPrice, sugarBeetPrice, cropInputsPrice, invPrice
*
*  --- for dairy farms
*
   nCows, milkYield, maxStockingRate, cowsPerHaGras, cowsPerHaArab, arabShare, milkPrice,
   beefPriceOld, mCalv_HF_Price, mCalv_SI_Price, fCalv_HF_Price, fCalv_SI_Price,youngBull_HF_Price,
   youngBull_SI_Price, youngCowPrice, conc1Price, conc2Price, conc3Price, emissionRight
*
*  --- for arable farms
*
   arabLand
    /;
```

Figure 246:

```
* --- exclude implausible ones
*
* (1) large herds and low milk yields
*
 allScen(scen) $ ( (p_scenParam(scen,"nCows") ge 50)   and (p_scenParam(scen,"milkYield") le 50)) = no;
 allScen(scen) $ ( (p_scenParam(scen,"nCows") ge 100)  and (p_scenParam(scen,"milkYield") le 60)) = no;
 allScen(scen) $ ( (p_scenParam(scen,"nCows") ge 150)  and (p_scenParam(scen,"milkYield") le 70)) = no;
*
* (3) small herds and high milk yields
*
 allScen(scen) $ ( (p_scenParam(scen,"nCows") le 60)   and (p_scenParam(scen,"milkYield") ge 80)) = no;
 allScen(scen) $ ( (p_scenParam(scen,"nCows") le 50)   and (p_scenParam(scen,"milkYield") ge 70)) = no;
 allScen(scen) $ ( (p_scenParam(scen,"nCows") le 30)   and (p_scenParam(scen,"milkYield") ge 60)) = no;
*
*
* (3) small herds and recent investment in stables
*
 allScen(scen) $ ( (p_scenParam(scen,"nCows") le 50)   and (p_scenParam(scen,"stableYear") eq 2000)) = no;
*
* --- delete parameters of deleted scenarios
*
 p_scenParam(scen,scenItems) $ (not allScen(scen)) = 0;
 p_scenParam(allScen,"lastYear")        = %lastYear%;
```

Figure 247:

up by the child processes. In order to keep the system extendable, firstly, all settings inputted via the Graphical User Interface are copied over to the specific scenario:

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

Figure 248:

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;
    scenFile.ap = 1;
    scenFile.lw = 30;

$iftheni.scenType "%scenType%"=="Profits"

    put "$SETGLOBAL mode Single Farm Run" /;

$else.scenType

    put "$SETGLOBAL mode Calculate MACs" /;

$endif.scenType
*
*   --- send scen specific parameters to include file
*
    Loop(scenItems $ p_scenParam(scen,scenItems),

           put "$SETGLOBAL ",scenItems.tl," ",p_scenParam(scen,scenItems) /;
    );

    put "$SETGLOBAL scenDes curScen " /;

    putClose scenFile;
```

Figure 249:

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

```
$iftheni %1==YES
   put_utility batch 'shell' / "type %curDir%\incgen\curScen.gms > %curDir%\incgen\"scen.tl".gms";
$endif
```

Figure 250:

The code to build and optimize the single farm model itself is realized 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 such as to avoid any conflicts with I/O operations to allow for parallel execution.

A single instance has a typical load of about 1.8 cores in average. In a multi-core machine it seems promising to execute several such processes in parallel. That is realized 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'  / 'start /B /NORMAL %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,' -scriptexit=%curDir%\flags\'allScen.tl'.cmd'
                        ' -maxProcDir=255 -output='allScen.tl'.lst %gamsarg% lo=3 --pgmName="'allScen.tl'
                       (',iLoop:0:0,' of ',card(allScen):0:0,')"';
```

Figure 251:

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:

```
*
*     --- generate a flag file which will be deleted by the GAMS process upon completion,
*         it steers the further execution of the loop
*
      put_utility batch 'shell'   / "echo test > %curDir%\flags\"allScen.tl".flag";
```

Figure 252:

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

```
execute 'if exist  %curDir%\flags\%scen%.flag del %curDir%\flags\%scen%.flag';
```

Figure 253:

A simple DOS script waits until all flags are deleted:

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

Figure 254:

```
set /a _trys=0
:again

set _count=1

for %%x in (%_FlagFiles%) do set /a _count+=1

if %_count% gtr %_nFiles% (

  set /a _trys+=1

  if %_trys%.==%_MaxTrys%. goto errorexit

  sleep.exe %_seconds%

  goto again

)
```

Figure 255:

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:

```
*
*     --- set name of result file (comprises last year)
*
      put_utilities batch 'gdxin' / ' ..\results\expFarms\res_',scen.tl,'_until_' p_scenParam(scen,"lastYear"):0:0,'.gdx';
```

Figure 256:

And load from there the results of interest:

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;


*     --- filter out results of interest (so far only macs, avACs and totACs)
      p_meta(actInds,redLevl,"mac",actinds1,scen) = p_res(actInds,redLevl,"mac",actInds1,"mean");

*
*     --- set MACS to -1000 (= Farm exit indicator) if for all reduction levels higher than the current one
*         no cow herd is found after the 3th year
*
     option kill=years;
      years(allYears) $ (calYea(allYears) le p_scenParam(scen,"lastYear") ) = yes;

*  alt
*     p_meta(actInds,redLevl,"mac",actinds1,scen) $ (sum((redLevl1,years)  $  (    (redLevl1.pos gt  redLevl.pos)
*                                                                              and (years.pos gt 3) ), p_res(actInds,redLevl1,"cows","Levl",years))
*         = -1000;

      p_meta(actInds,redLevl,"mac",actinds1,scen) $ ((redLevl.pos ne card(redLevl))
                                                     and  (sum( (redLevl1,years) $  (      (redLevl1.pos gt redLevl.pos)
                                                     and (years.pos gt 3) ), p_res(actInds,redLevl1,"cows","Levl",years)) eq 0) )

          = -1000;

*
*     --- exemption for last reduction level: set to -1000 if previous reduction level was -1000
*
      p_meta(actInds,redLevl,"mac",actinds1,scen) $  (    (redLevl.pos eq card(redLevl))
                                                     and  (sum(redLevl1 $ ( (redLevl1.pos eq redLevl.pos-1)
                                                     and (p_meta(actInds,redLevl1,"mac",actinds1,scen) eq -1000)),1)eq 1))

          = -1000;
```

Figure 257:

```
*     ---   average Abatement costs (avAC)

      p_meta(actInds,redLevl,"avAC",actinds1,scen) $ (p_meta(actInds,redLevl,"mac",actinds1,scen) ne -1000)
        = p_res(actInds,redLevl,"avAC",actInds1,"mean");

*     ---   total Abatement costs (totAC)

      p_meta(actInds,redLevl,"totAC",actinds1,scen) $ (p_meta(actInds,redLevl,"mac",actinds1,scen) ne -1000)
         = p_res(actInds,redLevl,"totAC",actInds1,"mean");
```

Figure 258:

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

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

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

```
*
*     --- 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");
*     --- for avAc
            p_meta(actInds,redLevl,scenItems,actInds1,scen)
       $ sum(redlevl1, p_res(actInds,redLevl1,"avAC",actInds1,"mean")) = p_scenParam(scen,scenItems);

      p_meta(actInds,redLevl,"redLevl",actInds1,scen)
       $ sum(redlevl1, p_res(actInds,redLevl1,"avAC",actInds1,"mean")) =
                                                                      p_res(actInds,redLevl,"redlevl",actInds1,"mean");
*     --- for totAC
            p_meta(actInds,redLevl,scenItems,actInds1,scen)
       $ sum(redlevl1, p_res(actInds,redLevl1,"totAC",actInds1,"mean")) = p_scenParam(scen,scenItems);

      p_meta(actInds,redLevl,"redLevl",actInds1,scen)
       $ sum(redlevl1, p_res(actInds,redLevl1,"totAC",actInds1,"mean")) =
```

Figure 259:

```
*
*      --- Store to disk (so that also intermediate results can be inspected)
*
           execute_unload '..\results\scenGen\meta_%scenDes%.gdx' s_meta,p_meta=p_res;
```

Figure 260:

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:

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

Graphical User Interface

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-

```
*
* --- 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
*
$setglobal op3 300
$setglobal op4 400
$setglobal op5 500
*
$endif.iScen
```

Figure 261:

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.

Workstep 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 define 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.

General Settings

The first tab in *General settings* defines the name of the scenario under which the results are stored. Further on, the user chooses general farm characteristics and options for the run: (1) the active farm branches, (2) if different states of nature

Figure 262:

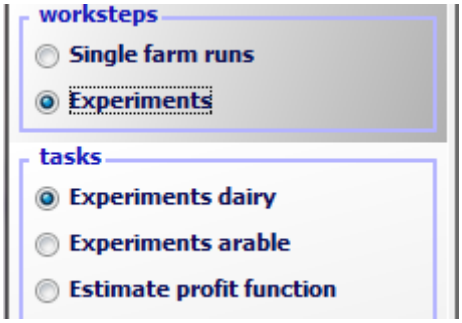


Figure 263:

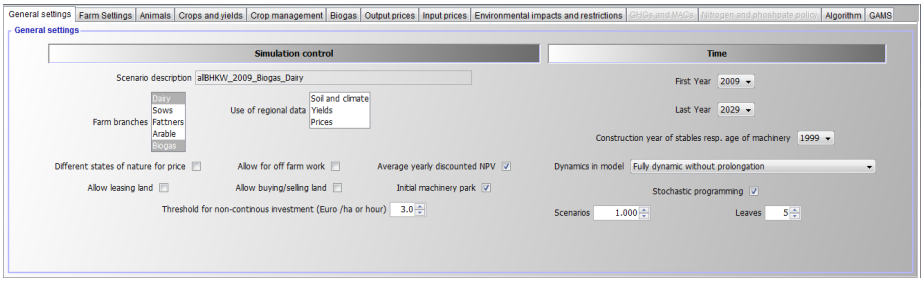


Figure 264:

(with regard to prices) are used, (3) if land can be leased or bought. Furthermore, the threshold to consider machinery investments as binary variables can be set.

The options under *Time* determine the last simulation year, if information from this simulation period is used to estimate economic returns until stables are depreciated and the time resolution for investment / farm labour and feeding decisions.

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

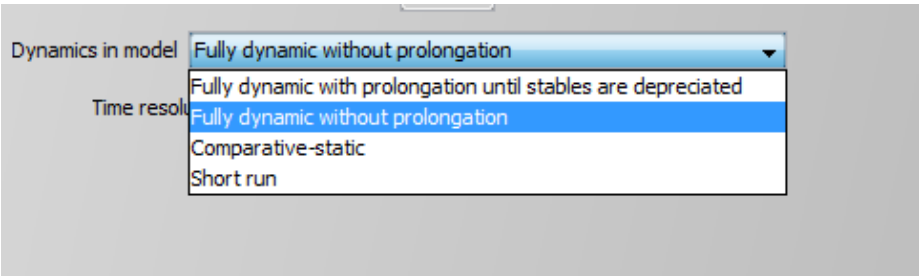


Figure 265:

The stochastic programming extension can only be switched on *Fully dynamic without prolongation mode.* For details on these settings, see section 3.3 above:

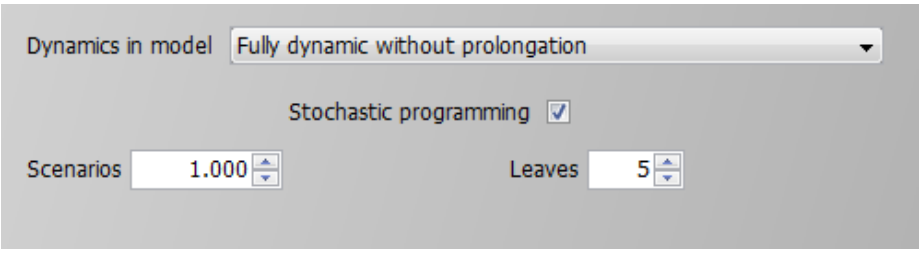


Figure 266:

In experiment mode, upper and lower ranges for certain settings can be set, as well as the number of experiments and some further algorithmic details:

Farm Settings

The farm settings panel carries general information about the farm-household and general price increases as shown in the following:

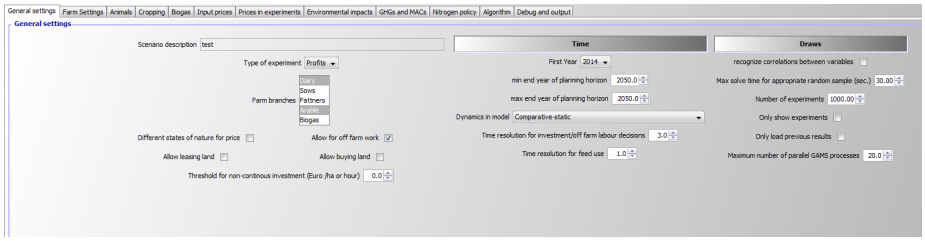


Figure 267:

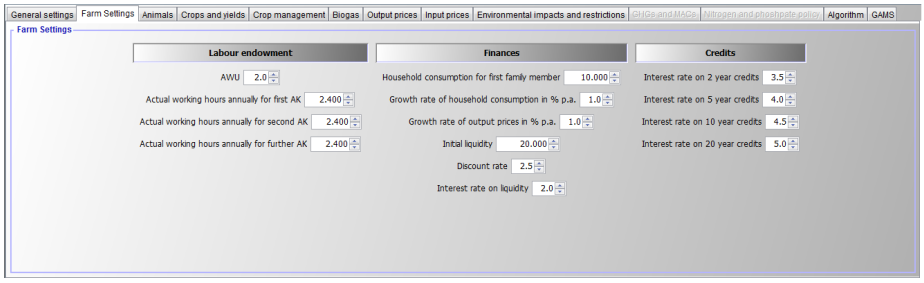


Figure 268:

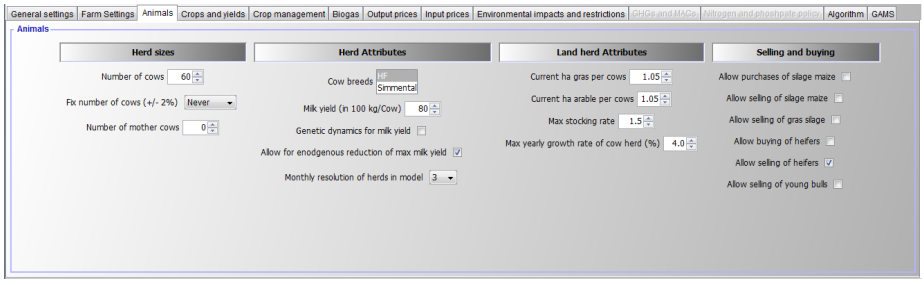


Figure 269:

Animals

The tab “animals” allows setting the initial herd sizes and other attributes related to animal husbandry. The herd size will be used to derive the initial stable and silo inventory, the current stocking rate the land endowment in case of a dairy farm. Note, the herd size which can be set depend on the active farm branches.

In the experiment mode, ranges can be defined:

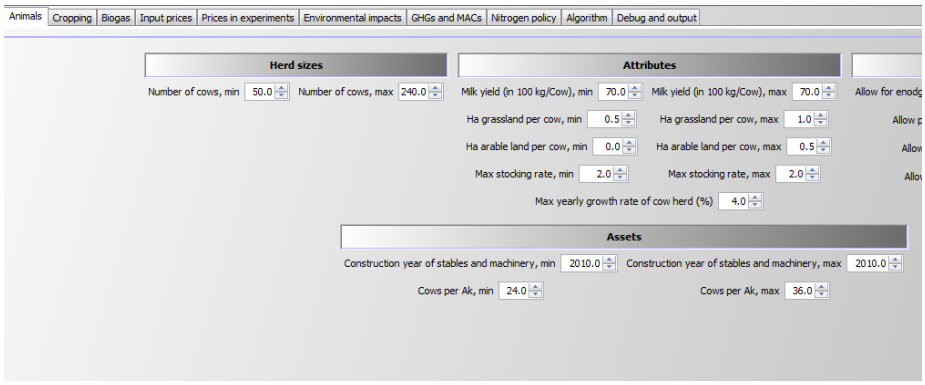


Figure 270:

Cropping

The tab “Crops and yields” comprises the selection of the crops and the yields and their growth rates (if arable farming is active).

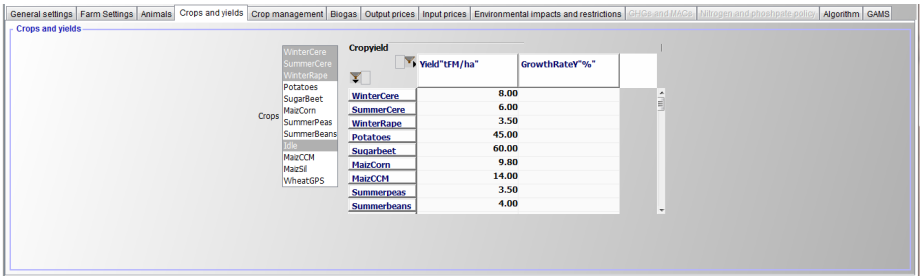


Figure 271:

A further tab “Crop management” comprises the steering options related to land use: if different tillage type, cropping intensities and crop rotations are used. Moreover, the average plot size, mechanization level and the climate zone as well as the distribution of the soil type can be chosen.

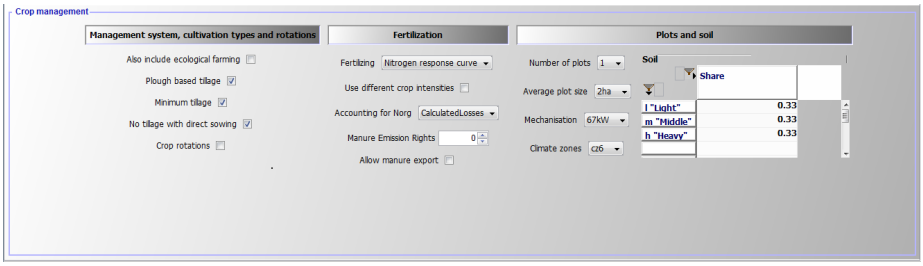


Figure 272:

Biogas

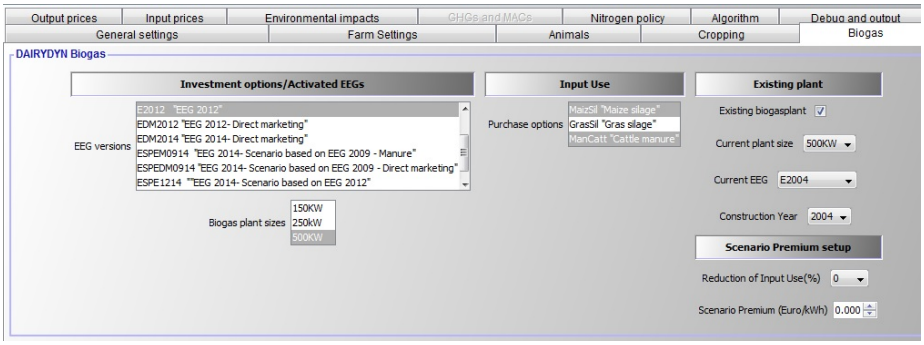


Figure 273: GUI - Biogas.jpg

The biogas tab includes different Renewable Energy Acts (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 as well as 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.

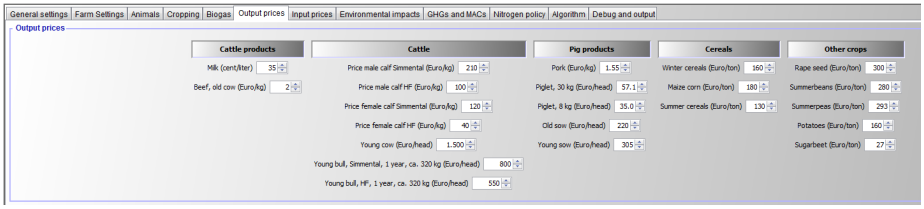


Figure 274:

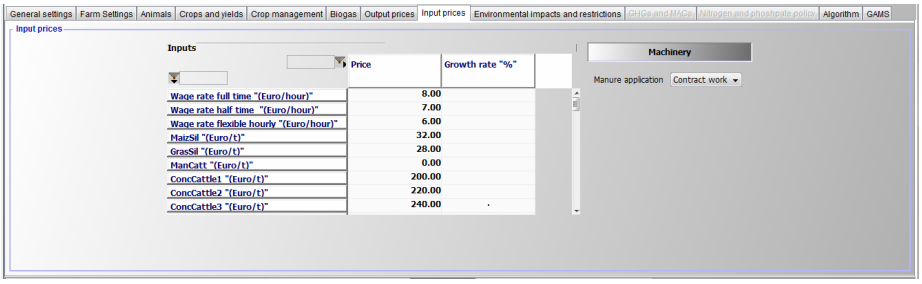


Figure 275:

Prices in Experiments

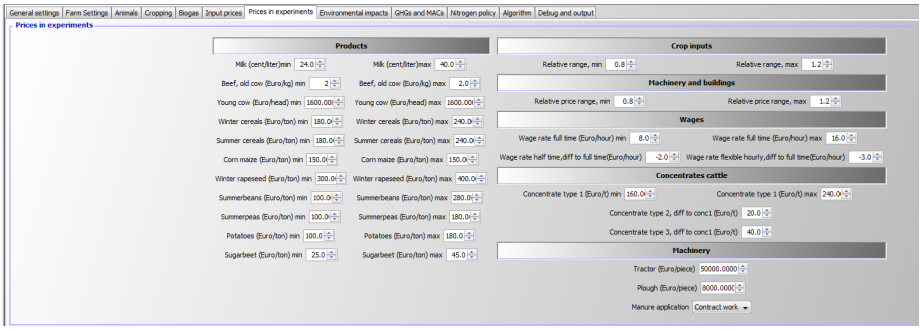


Figure 276:

Environmental Impacts

MACs

As the aim of the overall constructed model is to compare different designed emission indicator schemes, the GUI sheet “MACs” offers an assortment of the available indicators, usable to force abatement ceilings by the simulated farm

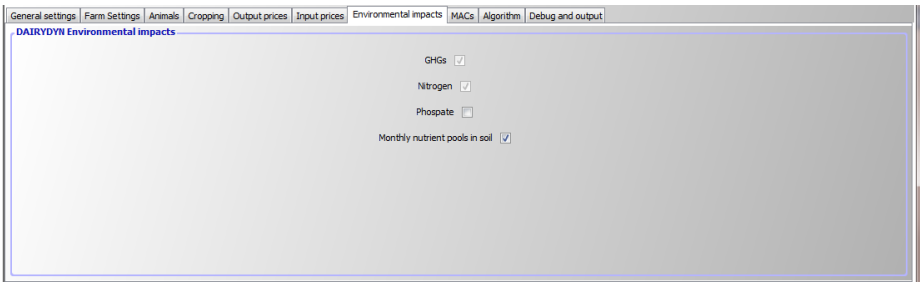


Figure 277:

to create marginal abatement cost curves. In addition the number of reduction emission reduction steps within the simulation runs and the percentage reduction per step (compared to baseline emissions) has to be defined to define the maximal emission reduction (GHG reduction steps time reduction per step).

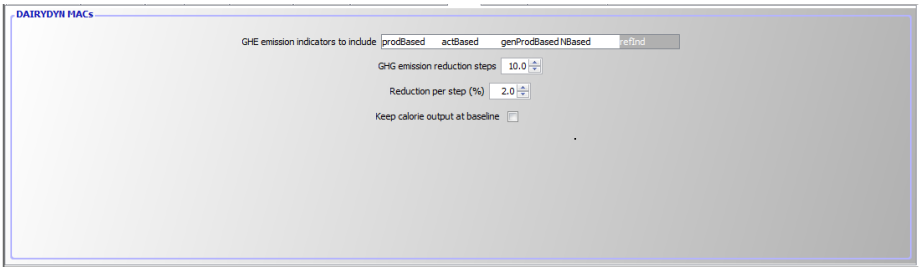


Figure 278:

Algorithm and Debugging Options

Figure 279:

The last tab of the GUI which is shown here defines the chosen solver to optimize the fully dynamic MIP problem and further precision adjustments.

In general it is recommended to use CPLEX as the MIP solver, see section 7.2.

GAMS

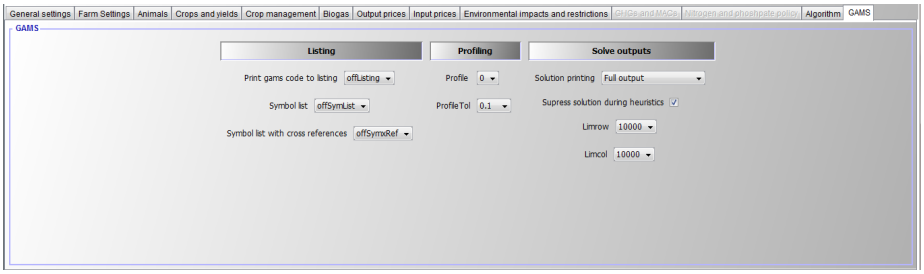


Figure 280:

Visualizing 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
- OffFarm, mean
- Overview work, time series
- OffFarm, 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:

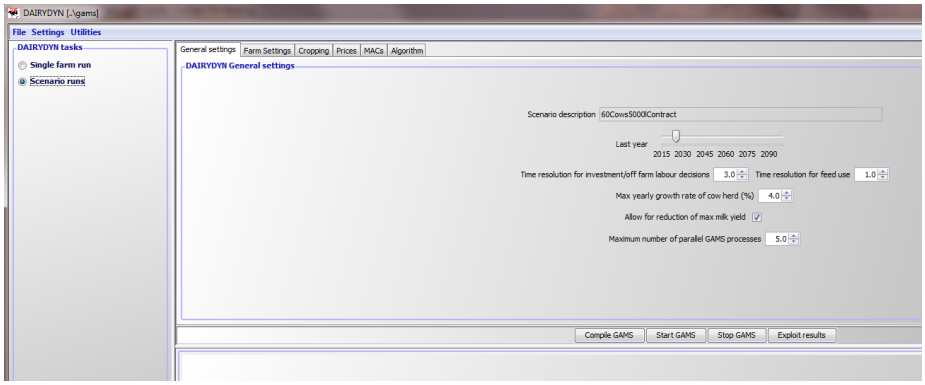


Figure 281:

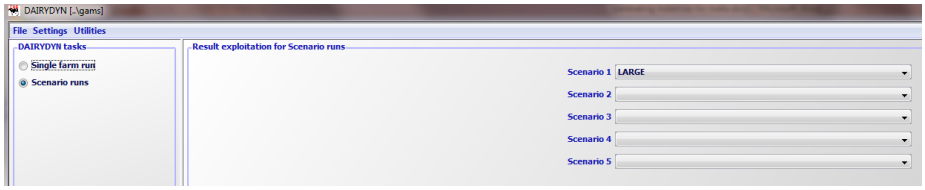


Figure 282:

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

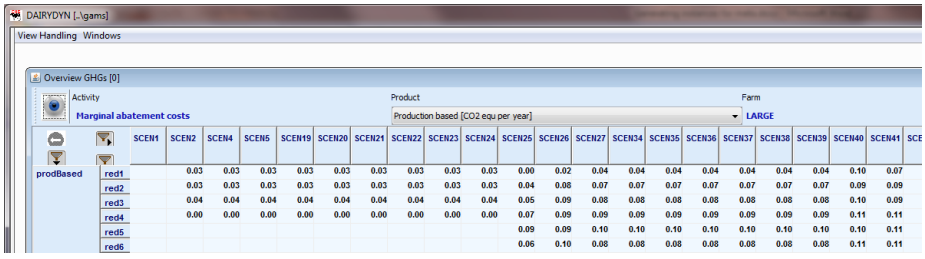


Figure 283:

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*

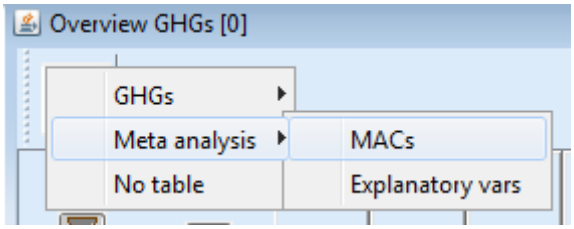


Figure 284:

2. 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:

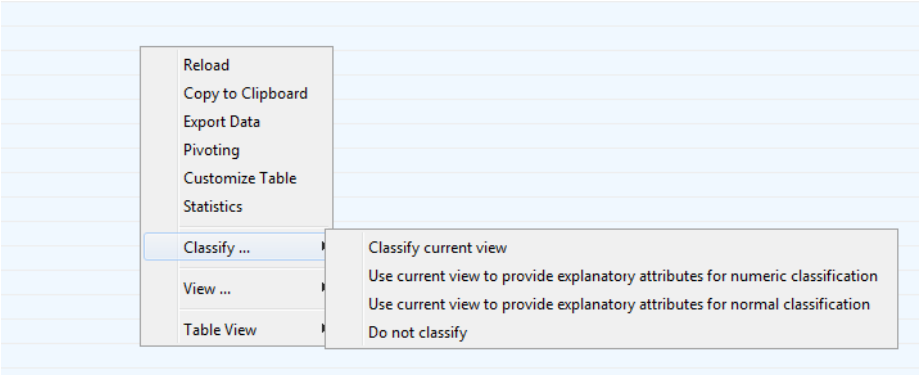


Figure 285:

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 [14]:

[14]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 analyzed by a routine, written in R (also designed by Britz and Lengers in 2012 (2012) which is available on enquiry from the responsible authors.

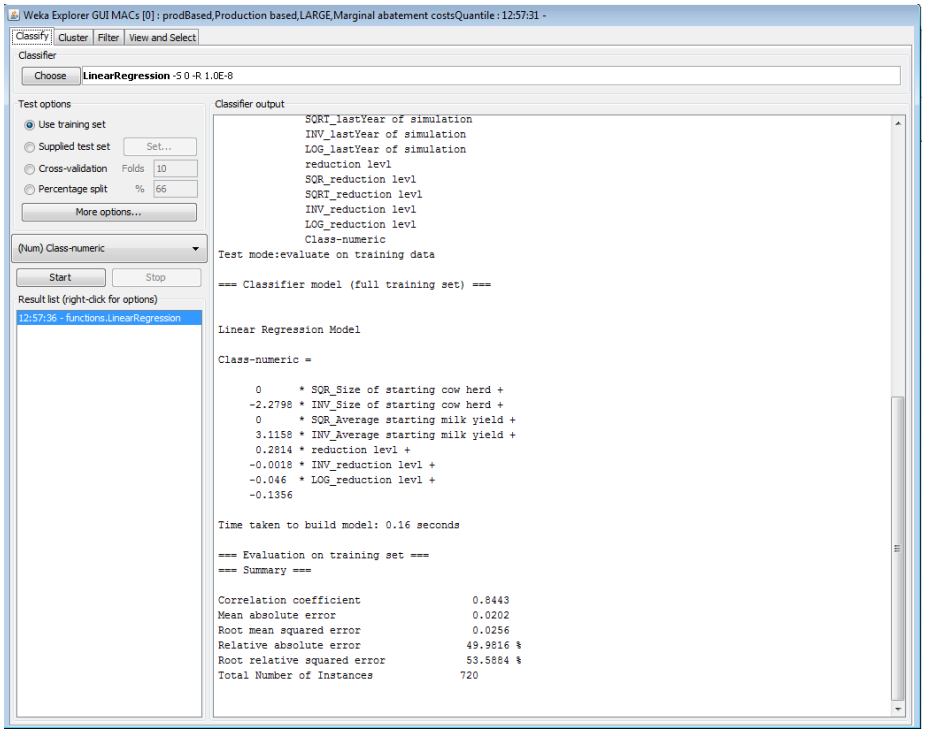

Weka Explorer GUI MACs [0] : prodBased,Production based,LARGE,Marginal abatement costsQuantile : 12:57:31 -
Classify
Cluster
Filter
View and Select
Classifier
Choose
LinearRegression -S 0 -R 1.0E-8
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Num) Class-numeric
Start
Stop
Result list (right-click for options)
12:57:36 - functions.LinearRegression
Classifier output
SQRT_lastYear of simulation
INV_lastYear of simulation
LOG_lastYear of simulation
reduction levl
SQR_reduction levl
SQRT_reduction levl
INV_reduction levl
LOG_reduction levl
Class-numeric
Test mode:evaluate on training data
=== Classifier model (full training set) ===
Linear Regression Model
Class-numeric =
0 * SQR_Size of starting cow herd +
-2.2798 * INV_Size of starting cow herd +
0 * SQR_Average starting milk yield +
3.1158 * INV_Average starting milk yield +
0.2814 * reduction levl +
-0.0018 * INV_reduction levl +
-0.046 * LOG_reduction levl +
-0.1356
Time taken to build model: 0.16 seconds
=== Evaluation on training set ===
=== Summary ===
Correlation coefficient 0.8443
Mean absolute error 0.0202
Root mean squared error 0.0256
Relative absolute error 49.9816 %
Root relative squared error 53.5884 %
Total Number of Instances 720

Figure 286:

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

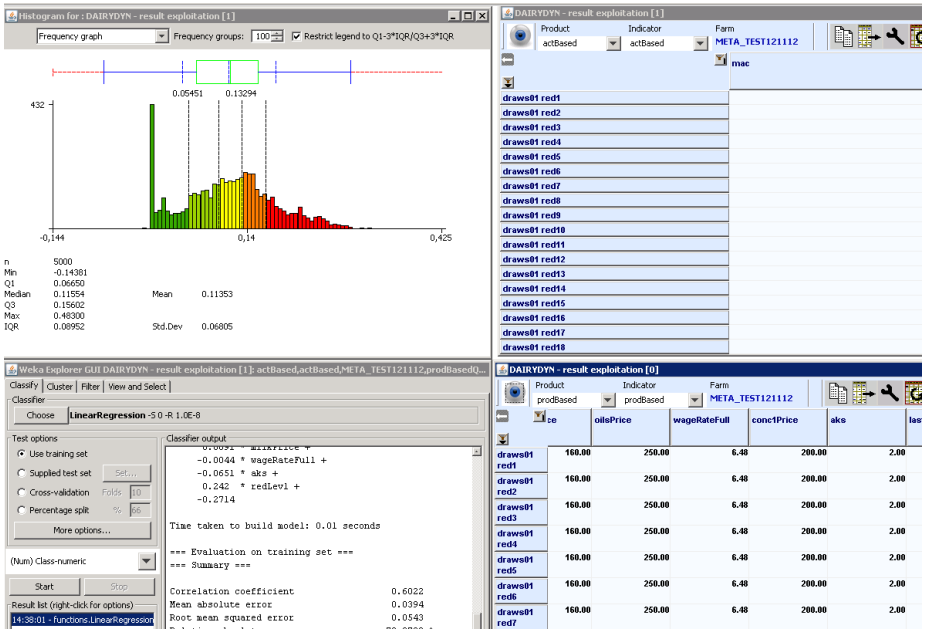


Figure 287:

References

Alcamo J., Shaw R. & L. Hordijk (1990): The RAINS Model of Acidification. Science and Strategies in Europe. Kluwer Academic Publishers, Dordrecht, Netherlands

Baker, E., Clarke, L.E. & E. Shittu (2008): Technical change and the marginal cost of abatement. Energy Economics 30:2799-2816.

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.pd (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

Clarke, L.E., Weyant, J.P. & Edmonds, J.A. (2008): On the sources of technological change: what do the models assume? Energy Economics 30:409-424.

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)

DeCara, S. & P.A. Jayet (2001): Agriculture and Climate Change in the European Union: Greenhouse Gas Emissions and Abatement Costs. AAEA Annual Meeting – August 4-8 2001, ChicagoFAO (2006): livestock´s long shadow – environmental issues and options. Food and Agricultural Organization, Rome, Italy

DüV (2009): 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"

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.

Gillingham, K., Newell, R.G. & W.A. Pizer (2008): Modeling endogenous technological change for climate policy analysis. Energy Economics 30:2734-2753.

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

Heckman, J.J. (1979): Sample Selection Bias as Specification Error. *Econometrica* 47:153-161.

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. & W. Britz (2011): Farm specific marginal abatement costs for dairy GHG emissions which base upon different emission indicators – a bio-economic model approach. Vorgestelltes Paper auf dem EAAE PhD Workshop 2011, Nitra, Slowakei, April 27-29.

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.

Lengers, B., Britz, W. & K. Holm-Müller (2013c): What drives marginal abatement costs of greenhouse gases on dairy farms – a meta-modelling approach. Paper presented at the 2012 AURÖ-Workshop, Frankfurt (Oder), Germany, February 18-19.

Lengers, B., Britz, W. & K. Holm-Müller (2014): What drives marginal abatement costs of greenhouse gases on dairy farms? A meta-modeling approach. *Journal of Agricultural Economics,* 65:2

Lengers, B., Britz, W. and Holm-Müller, K. (2013a): Comparison of GHG-emission indicators for dairy farms with respect to induced abatement costs, accuracy and feasibility, *Applied Economic Perspectives and Policy*, 35(3):451-475.

Lengers, B., Schiefler, I. & W. Büscher (2013b): A comparison of emission calculations using different modeled indicators with 1-year online measurements. *Environmental Monitoring and Assessment*, 185:9751-9762.

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.

Nemhauser, G. & J. Wolsey (1999): Integer and Combinatorial Optimization. A Wiley-Interscience Publication, New York

Park, J.S. (1994): Optimal Latin-hypercube designs for computer experiments. *Journal of Statistical Planning and Inference* 39:95-111

Pochet, Y. & L.A. Wolsey (2006): Production Planning by Mixed Integer Programming. Springer Series in Operation Research and Financial Engineering,

Springer Science + Business Media, USA

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).

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

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	*cuttingUnitCereal*	Getreideschneidwerk, 4,5m	p. 111

Machine	Naming in FARMDYN	Definition	Reference in KTBL 14/15
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

Machine	Naming in FARMDYN	Definition	Reference in KTBL 14/15
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* *closeSilo*	Silagetransport durch Lohnunternehmen	
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	*fodderMixingVert8*	Futtermischwagen 8m3 horizontale Schnecke, mit Befüllschild	p. 121
Mixer-wagon medium	*fodderMixingVert10*	Futtermischwagen 10m3 vertikale Schnecke mit Befüllschild	p. 121
Mixer-wagon large	*fodderMixingVert16*	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

Naming in FARMDYN	Included machines	Definition
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

Naming in FARMDYN	Included machines	Definition
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