MIND STEP

MODELLING INDIVIDUAL DECISIONS TO SUPPORT THE EUROPEAN POLICIES RELATED TO AGRICULTURE

Deliverable 3.1: Specification of model requirements

Protocols for code and data

AUTHORS	Marc Müller (WR), David Schäfer (UBO), Wolfgang Britz (UBO), Paolo Sckokai (UCSC), Alain Carpentier (INRAE), Fabienne Femenia (INRAE), Frank Offermann (THUENEN), Scarlett Wang (WU), Frederic Ang (WU), Alfons Oude-Lansink (WU), Christoph Pahmeyer (UBO)
APPROVED BY WP MANAGER:	John Helming (WR)
DATE OF APPROVAL:	26.08.2021
APPROVED BY PROJECT COORDINATOR:	Hans van Meijl (WR)
DATE OF APPROVAL:	26.08.2021

CALL H2020-RUR-2018-2	Rural Renaissance
WORK PROGRAMME Topic RUR-04-2018	Analytical tools and models to support policies related to agriculture and food - RIA Research and Innovation action
PROJECT WEB SITE:	https://mind-step.eu

This document was produced under the terms and conditions of Grant Agreement No. 817566 for the European Commission. It does not necessary reflect the view of the European Union and in no way anticipates the Commission's future policy in this area.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

This page is left blank deliberately

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

TABLE OF CONTENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

LIST OF FIGURES

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

ACRONYMS/ ABBREVIATIONS

ABM	Agent Bases Modelling
API	Application Programming Interface
BEFM	Bio-Economic Farm Model
CAP	Common Agricultural Policy
CMP	Crop Management Practices
DEA	Data Envelopment Analysis
EC	European Commission
EU	European Union
FADN	Farm Accountancy Data Network
FarmAgriPoliS	is an interactive game which simulates the development of an agricultural region made up of farms
FARMDYN	a dynamic mixed integer bio-economic farm scale model
FFS	Farm Structure Survey
FLINT	Farm-level Indicators for New Topics in policy evaluation
FSU	Farm Structure Units
GHG	Green House Gasses
GLOBIOM	Global Biosphere Management Model
GUI	Graphical User Interface
HSU	Homogeneous Spatial Units
IDM	Individual Decision Making
IFM	Individual Farm Modelling
MAGNET	Modular Applied GeNeral Equilibrium Tool
MIND STEP	Modelling INdividual Decisions to Support The European Policies related to agriculture
RMI	Risk Management Instruments
SDG's	Sustainability Development Goals
SVN	Apache Subversion, a software versioning and revision control system
VCS:	Version control system

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

EXECUTIVE SUMMARY

The MIND STEP project aims at developing a modular and extendable model structure. This requires a clear definition of criteria that the involved models and tools should fulfil in order to fit into this modular structure. This deliverable 3.1 provides an outline on how such a modular approach to model integration can be realized in practice. The focus is on the interaction of methods and results developed in the tasks within the MIND STEP work package 3 (WP3), titled "Development of modular and customisable suit of models focussing on the IDM farming unit" and the relation to models operating at higher organizational scales, like agent-based models that simulate the interactions of groups of individual decision making (IDM) units or market-models targeting the responses of the whole farming sector towards changing economic conditions at national level. The tasks within WP3 apply rather heterogeneous methods, ranging from micro-econometric analyses to the development of a simulation model for individual farms. Integrating these approaches requires therefore a conceptual structure that defines potential interfaces between them. This deliverable starts therefore with an overview on how the principle of modular model development can be operationalized, building on a literature review and an in-depth survey of four applied farm-level models (Britz et al. 2021). Based on this, the implications for task 3.2 ("Develop an overarching model structure for modelling IDM farm units in the agricultural sector together with parallel working consortia") are derived in chapter 3. A major conceptual decision is to define a core simulation model to which the methods and results from the other tasks in WP3 can be added in a modular manner. From the literature review, it became clear that the most widely applied type of model for policy impact analysis in agriculture on farm-level belongs to the family of mathematical programming models (MP). Based on this, concepts how the more survey-based and econometric approaches in tasks 3.3, 3.4, and 3.5, can be used to inform such an MP are outlined in the respective chapters. The implications of such a modular design structure for models at higher organisational scales are discussed in chapter 4. Here, general guidelines for modular simulation model and econometric model connections to WP4 and WP5, the related requirements, and a feasible work-flow are derived.

A major challenge experienced in model development is the maintenance of models during and beyond a typical project cycle. Thus, quality management of models play a critical role to facilitate the collaboration between project members and to ensure that models are re-usable in future projects, which is addressed in chapter 5. The main purpose of this last chapter is to highlight important aspects of quality management within the MIND STEP project and to introduce the three appendices of this deliverable. As it is intended to serve also as a reference handbook for the researchers and model users involved in MIND STEP, these appendices provide guidelines for good coding practices (Appendix 1) as well as guidelines concerning documentation and transparency recommended by the Wageningen Modelling Group. Appendix 3 provides an example for continued testing and quality assurance as it is implemented at the FarmDyn modelling team at University of Bonn.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

1. INTRODUCTION

MIND STEP aims at developing a modular and extendable model structure. This requires a clear definition of criteria that models should fulfil in order to fit in this model structure (e.g. with respect to specification of input and output data, modelling concepts, definition of terms, level of detail etc.). These requirements are described here by a set of protocols for input-output relations with respect to the functionality of the individual tool as well as requirements concerning interfaces between tools in the MIND STEP model toolbox. The MIND STEP suite of models is intended to be a modular framework where functionality can be added with additional models and data. The system of interlinked models centred around individual decision-making units is pictured in Figure 1. The focus of work package 3 (WP3) is on the behaviour of the individual farmer. Work package 4 (WP4) takes these single farm models and combines them with regional level models, such as Agent Based Models (ABMs), considering the interaction between farms and other actors in the agricultural food chain and non-food chain actors. This task has strong links with data-related works in work package 2 (WP2) and IT-related activities in work package 7 (WP7) regarding the technical implementation of models and solutions for data exchange. A crucial input for this deliverable is the work by (Britz et al., 2021) who elaborated on conceptual aspects of modular model design.

The modular framework should be flexible and sustainable in use (keeping complexity within certain limits) and will allow further improvements as needs arise. Therefore, MIND STEP develops an overarching IDM model structure that re-uses and improves existing modules. For that purpose, IDM models like IFM-CAP (Louhichi et al., 2017), FarmDyn (Britz et al., 2016), AGRISPACE (Mittenzwei and Britz, 2018) and the ABM AgriPoliS (Sahrbacher et al., 2014; Happe et al., 2006) are available in the MIND STEP consortium and serve as useful starting points. AgriPoliS allows performing experiments with artificial economic agents interacting in a dynamic and spatially explicit manner, especially focussing on structural change and land markets. IFM-CAP is an EU-wide individual farm level model aiming to assess the impacts of CAP towards 2020 on farm economics and environmental effects.

FarmDyn provides a flexible, modular template to simulate economically optimal production and investment decisions in detail at individual farm level. The current version of FarmDyn (Rev. 2355, 16.06.2021) depicts various farm branches (arable cropping, pig fattening, piglet production, dairy, beef fattening, biogas plants). The behaviour module maximizes the net present value over a longer simulation horizon, taking into account detailed restrictions related to feeding, fertilisation, further biophysical and environmental constraints and farm endowment constraints: labour, land, financial assets, equipment and buildings. Integer variables depict indivisibilities in labour use and investment decisions. FarmDyn consistently combines production, input use, and environmental constraints. As such FarmDyn also acts as a test-bed for the integration of IDM data in current models like MAGNET (Woltjer and Kuiper, 2014). Given the different policies (CAP, climate change, trade, environment, energy, etc.) and policy issues (from local to global), different availability of data, different farm types, different regions with different socio-economic and environmental characteristics, IDM models in WP3 and WP4 focus on meaningful subjects, farm types and regions in the EU rather than the EU as a whole.

Policies related to agriculture increase their scope to incorporate for example objectives of the Paris climate agreement and the Sustainability Development Goals (SDGs). Modelling these policies require models that offer both the individual farm level decisions regarding adoption of new technologies, risk management, farm exit, etc. and the interactions between individual farms and with non-farmers e.g. in the value chain.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

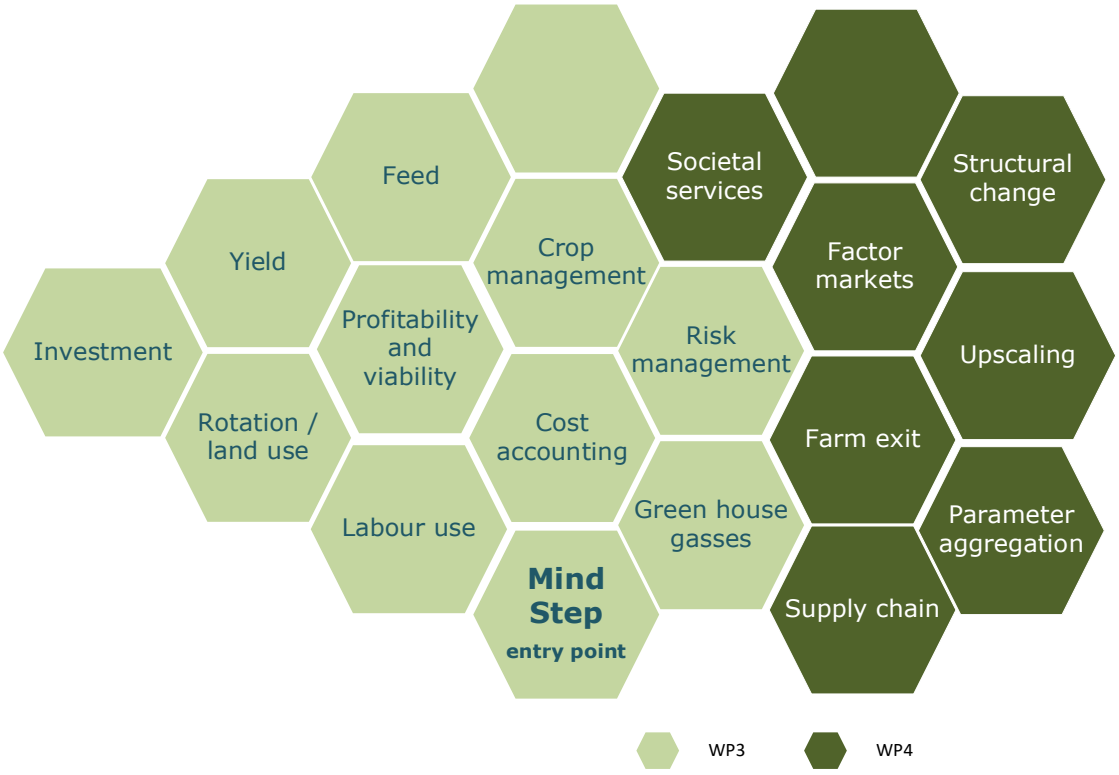


Figure 1 MIND STEP: system of interlinked models centred around individual decision making units

A prominent example is the future CAP beyond 2020, as it will include environmental and climate practices as new conditionality tools for obtaining farmers income support. Climate policies require IDM models with farm management mitigation options that are available to reduce greenhouse gas emissions and the trade-offs with other environmental and animal welfare policies, as well as income and risk (Spiegel et al., 2018). The modelling of GHG emissions and related mitigation options draws on the technology rich IDM model FarmDyn (Britz et al., 2016; Lengers et al., 2014). Besides improved modelling of legislation and mitigation options in Germany, the model will be extended to regions and farming systems in the Netherlands focussing on mitigation of methane and nitrous oxide emissions. The analysis of policies focussing on eco-system services and improving the conservation of the EUs wild flora and fauna via preservation of farm-genetic resources, biodiversity and habitat require multi-crop models.

Based on these general considerations regarding model coupling and modular design, the following sections are structured around the tools developed in the MIND STEP work packages and provide concepts on how these tools can be linked in a modular manner. To ensure that the modelling activities in MIND STEP are interchangeable between groups of developers, practical issues like good modelling practices, coding conventions, and quality management criteria discussed, and guidelines are provided in the appendices.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

2. MODULAR MODEL STRUCTURE

The implementation of an IDM involves data preparation, model set-up and parameterization, and reporting as depicted by Figure 2 Typical Model Workflow

, which are usually separated from model equations (separation of code from data). Particularly when relying on statistical sources, data preparation must deal with outliers or missing entries that can impair model execution, performance and more so plausibility of results. This underlines that the generation of the model database is an integral part of the model workflow, particularly because it is instrumental for the model set-up and parametrization in a subsequent step before the model itself is solved.

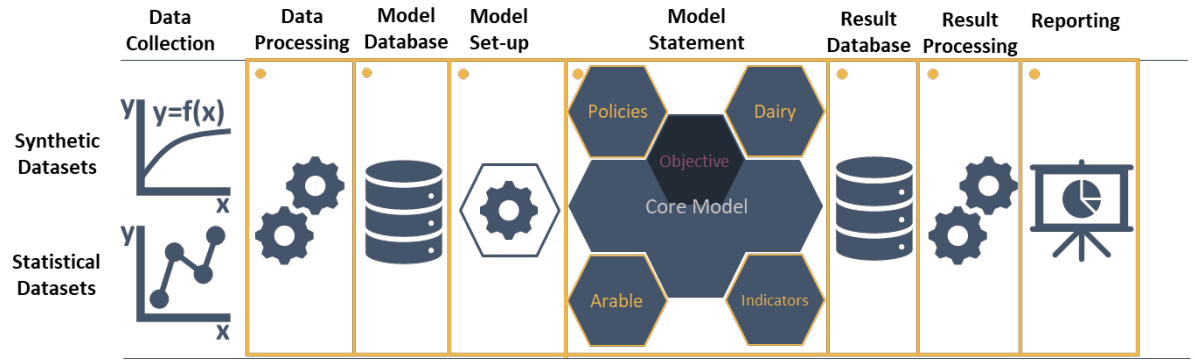


Figure 2 Typical Model Workflow

Restricting data preparation, parameterization, model solving and reporting to the currently needed farm branches, farming systems or relevant policies greatly eases model application. A block of equations and variables with the related code-blocks for data preparation and reporting, for instance for dairy farming, can be jointly understood as a module if it can be switched off without impairing the use of the core model and other modules. The activation of modules can be data or user-driven. Such a modular design is defined by Russell (2012) as:

"Modularity describes specific relationships between a whole system and its particular components. A modular system consists of smaller parts (modules) that fit together within a predefined system of architecture. Modules feature standardized interfaces, which facilitate their integration with the overarching system architecture. A key feature of each module is that it should encapsulate (or "black box") its messy internal details […] to display only a consistent interface. The designers of modular systems are therefore able to swap modules in a 'plug-and-play' manner, which increases the system's flexibility." (Russell, 2012)

Flexibility in configuring the IDM as the system discussed in here is required for a generic model. A modeller may not be interested in applying all aspects of a generic model for a given use case. Instead, modules directly relevant for the research question will be activated and others switched off, for instance by including a specific set of policies or an alternative objective function. Analysing policy effects on a potential farm exit might require a long-time horizon and the activation of modules relating to on- versus off-farm labour, equity use, and farm succession aspects. In contrast, for such an application, a monthly time scale related to detailed dis-aggregation of field operations might be switched off, but might be required to assess agri-environmental measures. Such flexibility in model set-up keeps each instance of the model at manageable size and facilitates the parameterization from a case-study specific database.

Software engineering embraced modularization from the beginning and continues to conduct extensive research in this field (van der Hoek and Lopez, 2011). Quite early Parnas (1972) established the fundamental principle of reducing the information that a module opens for access, termed "information hiding". The related principle of "low coupling and high cohesion" by Stevens, Myers,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

and Constantine (1974) advocate for low dependence between modules (coupling) and strong dependence between elements inside a module (cohesion). “Separation of concerns” as a further principle decomposes a computer program such that each module addresses different aspects of the problem at hand (Dijkstra, 1982). Similarly, Lieberherr and Holland (1989) propose the “Law of Demeter”, as a special case of loose coupling, which emphasizes that modules should be separated from each other as much as possible. In the context of BEMFs, these principles lead to the following general advantages:

- *Transparency*: the model can be reviewed module by module, facilitating overall comprehension and quality control.
- *Maintainability*: Code and data base updates of a module do not affect others.
- *Extensibility*: Modules can be extended or added to the core model without affecting other.
- *Distributed development*: Modellers focus on specific modules which eases coordination of coding efforts.

While desirable, achieving modularization for an IDM is challenging. **Cross cutting aspects/concerns** limit the extent of low coupling, for instance, most modules calculating indicators need information on all crop and animal activities. Conceptually, **there is an unlimited set of possible modularizations.** For instance, the yellow hexagons in Figure 2 (or sub-divisions thereof) could be grouped into a large number of functional units, depending on pragmatic and conceptual considerations. Different viewpoints might suggest different organizations into modules, such as which data sources feed into which equations, domain knowledge of coders responsible for specific aspects, or the need to reflect regional detail in the equation structure, for instance related to policy implementation. Deciding on the number of modules and their delineation is hence a core design challenge.

A recent study by (Britz et al., 2021) provides an overview on the modular structure of four applied IDMs, which show different degrees of modularization. Modules for “database generation” and “model statement” are distinguished in all models, see Figure 2, to separate code and data. The database generation is usually only performed once for each case-study as this involves time consuming data work and possibly fairly complex statistical methods. A complete separation of code and data is still not fully implemented in any of the reviewed models, as numbers or references to specific list elements might still appear in equations, such as “y = 3*x” instead of “a = 3” and “y = a*x”, or “x[“wheat”] = y” instead of “a = [“wheat”]” and “x[a] = y”. This is less the result of a design decision but often rather due to time shortage in project-based development, where ad-hoc changes of the model code were implemented and not revised at later stages, so that such blocks of code persist. Still, such observations are the exception rather than the rule, and the models follow in general the principle to separate code from data. The possibility to parameterize a template model flexibly for new use cases, as long as the database fulfils certain minimum requirements, is a critical feature for a generic IDM. However, this also implies that the same equations and variables will be used for all use cases, at least at the level of the core model. While this may not be problematic in some instances (accounting identities, bio-physical relations), the representation of policies or the calculation of environmental indicators may require further adjustments of the model code, and are hence better placed within modules.

Such a modularization of equation blocks that are used in the model statement is particularly observable in FarmDyn, which is structured along functional units of code which can be arranged rather flexibly into a customized MP model for each farm instance. At the top-level, farm branches can be selected to add related blocks of equations to the core model. For instance, adding the dairy farming branch will integrate blocks of equations that govern herd composition, feeding requirements, and manure management. These modular blocks themselves can be replaced by alternative implementations as long as the input-output relations defining their interfaces with core model and related modules are maintained. The main challenge in this replacement of alternative implementations is the interpretation of the input/output in relation to the interpretation of these

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

inputs/outputs for other modules and the effect that has on (implicit) assumptions in the structure of these modules. Swapping modules also implies that the context of the data-input and output is consistent between them.

Exchangeable policy modules are also an important part of a generic IDM to reflect case study specific implementation of measures. Policy modules can restrict the solution space and/or define subsidies as part of the objective, potentially depending on farm management choice in case of opt-in measures. This requires a generic approach to handle subsidies in the objective function. Policy modules might introduce constraints which restrict environmental indicators such as a soil-nutrient-balance as defined in the country- or region-specific regulations. These definitions of indicators might deviate to what the scientific state-of-the-art suggests. Indicators derived from legislation should hence be coded in the related policy module and kept separated from indicator modules that serve dominantly reporting purposes. If indicators enter the objective function, a modular choice of indicators requires a generic approach to handle varying lists of indicators.

The objective function can be regarded as a module itself. A purely profit-maximizing approach has been observed by Janssen and van Ittersum (2007) and Reidsma et al. (2018) for the majority of the reviewed models. Two of the four models we reviewed permit at least the inclusion of a farmer's risk preference, either by weighing the expected profit against its variance in a comparative-static setting in IFM-CAP, or on demand in FarmDyn where different risk behavioural models can be used in a stochastic-dynamic programming framework.

The observed models all comprise code for calibration against the observed farm data as an important feature of a generic model. IFM-CAP, FSSIM and CAPRI-FT draw on Positive Mathematical Programming (PMP) which requires at least a non-linear objective function. A larger body of literature on PMP suggests two important observations. First, PMP requires some econometric evidence on how input and output quantities react to changes in prices and, second, it can be applied also to models with a quite limited set of constraints. Calibration of a modular system is challenging as a re-configuration by adding or replacing modules will likely impact the allocative response of the model or can even require a re-calibration. There are now also automated approaches to calibrate linear and mixed integer programs (Britz, 2020) which are for instance applied in FarmDyn.

Modularization mainly aims at, first, easier adjustments to different use cases, such as covering regional policies, and second, at model extensions, for instance, by integrating new indicators. According to the principle of information hiding, a module is defined by its task, such as determining feeding amounts at given herd sizes and component prices, but not by how the task is achieved or coded in detail. Accordingly, the module's definition includes the list of well specified inputs required from other modules and core model, and the minimum set of well specified outputs to be generated for others. This requires a clear and technically detailed documentation (symbol name, units, dimensions etc.) of the variables supposed to be defined endogenously by a module and of all variables exposed to other modules and the core model. According to the low coupling principle (Stevens et al., 1974), modules should interact only through these defined interfaces. Thus, a module should bundle as many functionalities pertaining to its task as possible (high cohesion). This includes not only the equations that a module contributes to the overall model, but also its parameterization and reporting.

Accordingly, a module of a generic IDM should be broken in three code blocks: (1) its data preparation – to separate data from model code and avoid time consuming data preparation for each model run, (2) its equations which feed into the overall MP model statement, and (3) its reporting part. Its equations and related variables are at its core by providing the link to equations and variables of other modules. The equations also mirror how the task is performed in detail and therefore constitute its unique core. But a module might feature multiple implementations for data preparation to work with differently structured databases, and for reporting, for example, to provide rough overviews or

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

detailed debugging reports, or to output different formats, such as spreadsheets or interactive web-pages.

With regard to the required data, a distinction between native and contributed modules is useful here (Figure 3 Modular Setup). By definition, a native module (the hexagons labelled: “Module” in Figure 3) can be always fully parameterized from the general model database, while a contributed equation module offering additional functionalities (the purple block of hexagons in Figure 3) might require additional data which it must provide by own code for data preparation. The same holds for the reporting step.

Ideally, the general model database could serve any case-study using native modules only. Yet, EU wide data bases such as FADN cannot provide the farm management detail required for a technologically rich generic IDM. As a compromise, contributed modules should provide sensible default values in case the required information cannot be obtained from the data-processing steps (the purple database symbol labelled: “0” in Figure 3 Modular Setup

). A case-study application can then code its own data driver to use a specific data base which replaces default values.

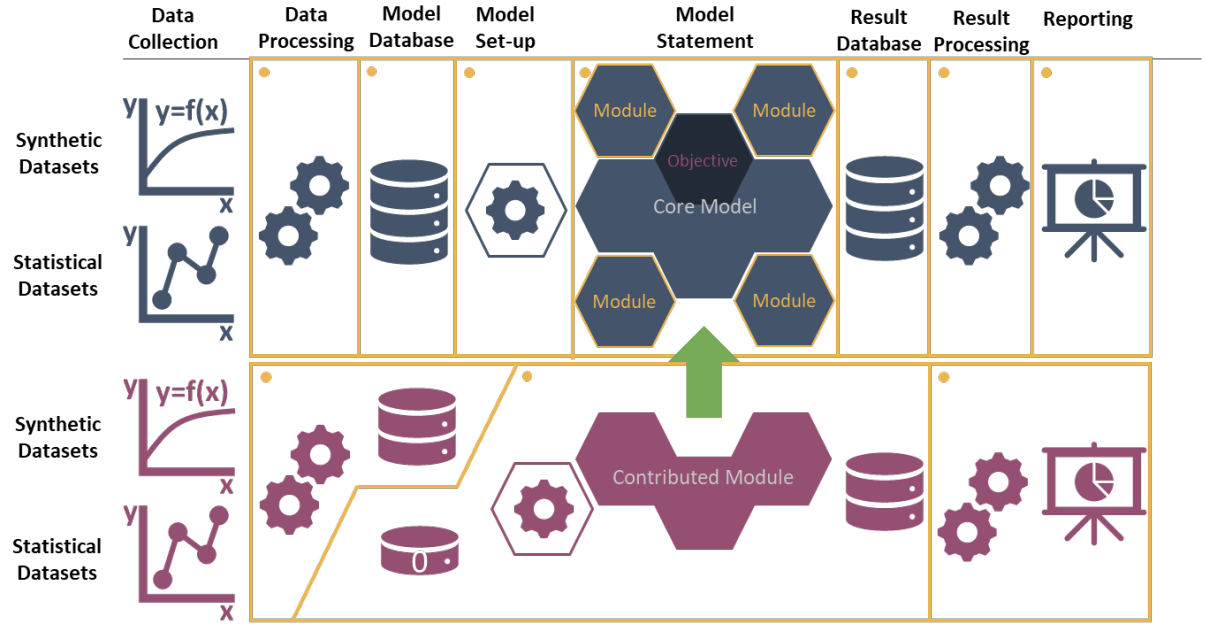


Figure 3 Modular Setup

In summary, the most crucial aspects for design and integration of modules in such a setting are the clear definition of obligatory inputs and outputs (interfaces) and ensuring that the equations in the module can be executed by providing default values for all parameters. This also implies that the technical documentation of core model and modules, and the development of protocols for contributor should receive particular attention from the very beginning if model development and maintenance is to be distributed across multiple teams with high staff turnover rates.

This already underlines that modularity comes along with challenges for the computational implementation. The example models above comprise ten- or hundred thousands of code lines of which larger sections relate to data processing and reporting. The restricted language features of an AML like GAMS, used for all four models reviewed, eases learning but challenge code development and maintenance for such large-scale projects, especially if multiple developers are involved. In particular cases where different variables and parameters are named similarly but referring to different contents (namespace conflicts) are difficult to handle in programming languages that do not

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

allow for the distinction of namespaces. This requires additional synchronization efforts, for instance by establishing coding protocols within the user community and by emphasizing the need for good code documentation. In addition, there should also be a documentation which modules cannot be combined, e.g. in the form of an exclusion/inclusion matrix.

Modularity also needs to reflect user-model interaction. Three models reviewed in detail (CAPRI-FT, IFM-CAP, FarmDyn) feature a GUI, all realized in GGIG, to facilitate, for instance, choosing the included modules or the data base to use. An important question is to which extent a specific model configuration (farm branches, activities covered, specific policy implementation etc.) is driven by the data base or defined by user interactions. Second, to what extent should the user be able to provide (or overwrite) via the GUI data otherwise read from the model data-base, such as, e.g. run specific prices, yields or values of policy measures. Third, should the GUI also cover such functionalities for contributed modules? If yes, how is this technically achieved and institutionally organized? And finally, which alternative ways does a user have to interact with the model itself?

3. MODULAR ALIGNMENT OF TASKS IN WP3

3.1. Overarching model structure: Core model and modules

Several reviews of IDMs (Britz et al., 2021; Janssen and van Ittersum, 2007; Reidsma et al., 2018) found that the majority of applied IDMs are essentially mathematical programming models (MP) at farm level. In general, such models optimize an objective function of decision variables, subject to constraints, where both, objective function and constraints can be linear or non-linear. In particular models to investigate investment decisions regarding farm machinery or buildings may require that some decision variables can only have integer values, in which case the models become “mixed-integer programming models” (MIP). The optimization of the farm plan is then done using numerical algorithms, or solvers. The choice for one particular model type and the appropriate solver has implications for the flexibility of the model regarding the integration of additional model features. For instance, a typical solver for MIPs may not be capable of solving a model with non-linear constraints or objective functions.

Two IDMs currently available within the MIND-STEP consortium, which were also reviewed by (Britz et al., 2021), are IFM-CAP and FarmDyn. IFM-CAP features a quadratic objective function and linear constraints, whereas FarmDyn is completely linear, but includes integer variables. The decision to choose core functionalities of either FarmDyn or IFM-CAP as the core model for the overarching structure – or a hybrid version of both – has a range of implications for the interactions with the other research activities within work package 3 and between work packages 3, 4, and 5.

Within WP3, the overall set-up is that task 3.2 aims at defining a core model and structures for exchange of information with the other tasks. Tasks 3.3, 3.4. and 3.5. will then be linked to the core model and either contribute to it by providing modules with additional functionality, or by making use of the core model, e.g. by building on simulation results. Figure 4 summarizes the intended interactions within WP3. Task 3.3. will investigate behavioural aspects of GHG mitigation strategies at farm-level, and thus improve the representation of farmer’s behaviour in the core model. Subsequently, the improved core model will be used within Task 3.3 to derive strategies for farm-level decisions to mitigate GHG emissions under a range of policy options. Task 3.4. will add improved representation of crop-management options, and therefore enrich the representation of farming technologies. Finally, Task 3.5. investigates risk-management behaviour and risk-management instruments for farmers. While the former refers to an improvement of the objective function, the latter increases the number of decision variables by adding the usage of RMIs.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

Colours: Blue: Core model

Purple: Informs the core model or adds functionality

Green: Uses results from the simulation model

Figure 4 Modular interactions between tasks in WP3

Figure 4 depicts the interactions between the different tasks within WP3, structured around the concept of core model and modules. The technical implementation of these interactions critically depends on the formal aspects of the core model.

3.2. Features of the core model

MIND STEP will develop an overarching model structure for IDM farming units. Key functionalities will be implemented, which are common to any farm type and instance of the system analysed, including for example endogenous adjustment of yield, selected variable inputs, land and labour use, investment and farm viability. Starting point of this Task are IDM models available inside the MIND STEP consortium e.g. IFM-CAP, FES and FarmDyn. Other IDM models outside the consortium are considered as well. The functionality of the models is reviewed and existing modules are re-used as far as possible. We will contact other consortia financed under the topic to circumvent possible double work and to share ideas and approaches with them. The implementation of the key functionalities will follow the protocol and specifications for models as defined in Task 3.1. Output of this Task is an overarching IDM model that fits to the purposes of MIND STEP as the basic structure where additional models (to be developed in WP3 and WP4) can be added to extend the MIND STEP functionality in a consistent and modular way.

While the structure of the core model is not yet (April 2021) fully determined, it has become clear from previous surveys and the review by (Britz et al., 2021) that it will take the form of an MP model, as this provides the widest range of possible application areas. In the simplest case of a linear MP model, the general structure will be as depicted in Figure 5: The farmer's decision variables or activity levels are represented here by an n*1 vector named x. This includes all production activities for crops and animals, but also buying and selling of inputs and products. Furthermore, lease of additional farmland, purchase of crop insurances, off-farm labour, and so on, can be included in x. Associated with this is an n*1 vector of cost and prices named c. Each production activity requires some resources like land or labour, selling products requires that they have been produced in the first place, and so on. The relations between the demand of n activity levels for k resources is summarized in the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

coefficient matrix A of size k*n. Finally, the farm is supposed to be endowed with k resources, represented by a 1*k vector r. Formally, the problem is to maximize the scalar product of c and x, such that the resource requirements do not exceed the farm's endowments, and such that activity levels are larger or equal to zero (see also Hazell and Norton, 1986).

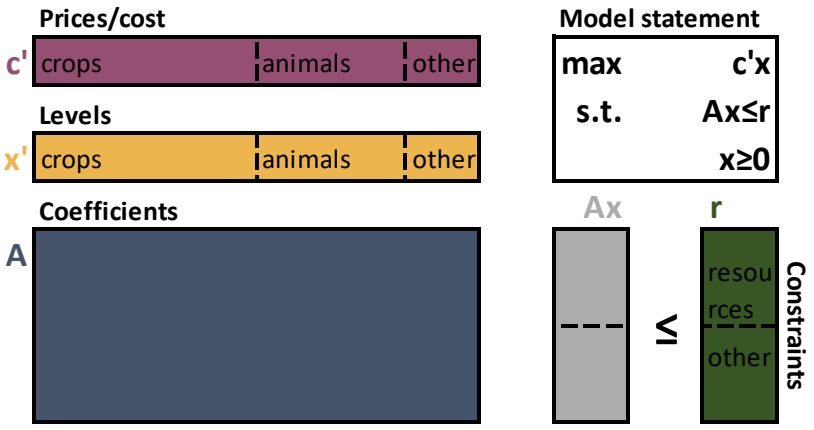


Figure 5 Typical Linear Programming Model

For pragmatic reasons, it is conceivable that the envisaged core model will consist of a minimalistic set of crop and animal production activities, as well as minimalistic set of constraints. Increasing the model's analytical capabilities will then take place by adding specific modules.

The extension of the core model by adding modules may take several forms. A typical case is extending the number of activities, for instance by accounting for different intensity levels in crop production. In this case, crops could be produced with higher or lower amounts of fertilizer and instead of one activity for each crop, the model would include now several (e.g. wheat with low fertilization, wheat with high fertilization). This increase in the number of decision variables is depicted by the lighter-shaded areas in Figure 6. As a consequence, also the other parts of the model need to be expanded: output prices may be the same in this case, but production cost in the vector c will change, and also the yields per hectare, which have an impact on the farm's potential to sell products. It is also conceivable that the number of constraints is extended, e.g. by adding new crop operations like minimum tillage, such that new machinery is required. A third possibility to enrich the core model is by including new terms to the objective function (vector d in Figure 6), which capture e.g. the yield variance of alternative cropping activities and the farmer's attitude towards the implied risks. These considerations imply that the development of modules must take place in awareness of the core model and its requirements. Furthermore, it is advisable to also provide a clear delineation of what the core model is not going to accommodate (e.g., non-linear constraints).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

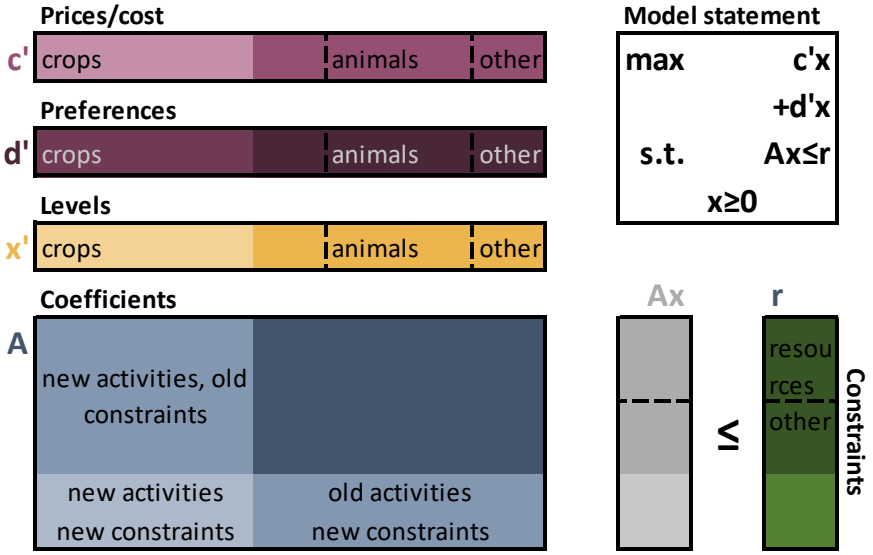


Figure 6 Modular extended core model

3.3. GHG mitigation options and farmers' choices

The aim of Task 3.3 in WP3 is to illustrate the extendibility of the MIND STEP model structure by adding a model to analyse mitigation strategies to climate change. For broad applications of the model to different regions and sectors, this requires geo-referenced farms linked to financial-economic and biophysical datasets. Because of data intensity the model will be first applied to regions in Germany and the Netherlands focussing on methane and nitrous oxide emissions on livestock and arable farms. The modelling starts from the technology rich IDM model FarmDyn. Based on the stage model of self-regulated behavioural change (Figure 7), MIND STEP designs and implements experimental procedures to measure farmers' socio-psychological factors, adoption intentions and stage membership in taking up mitigation measures, to more realistically model behaviour regarding adoption of new technologies and production and environmental impact of policies. Besides traditional agricultural statistics also big data will be considered (FADN, FLINT, individual census data, IACS, plot data, satellite data and bio-physical data will be processed and linked). Field interviews with a selected sample of Dutch dairy farms will be conducted to generate insights in farmers' willingness to participate in GHG reducing measures. Survey and spatial data provide detailed information on the individual farmers and agro-ecological conditions in local environments. Combining behavioural and biophysical characteristics feeds into the calibration and econometric parametrisation of the model to be developed in this Task 3.3. In practice, this implies that the existing FarmDyn model will be augmented by non-economic factors.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

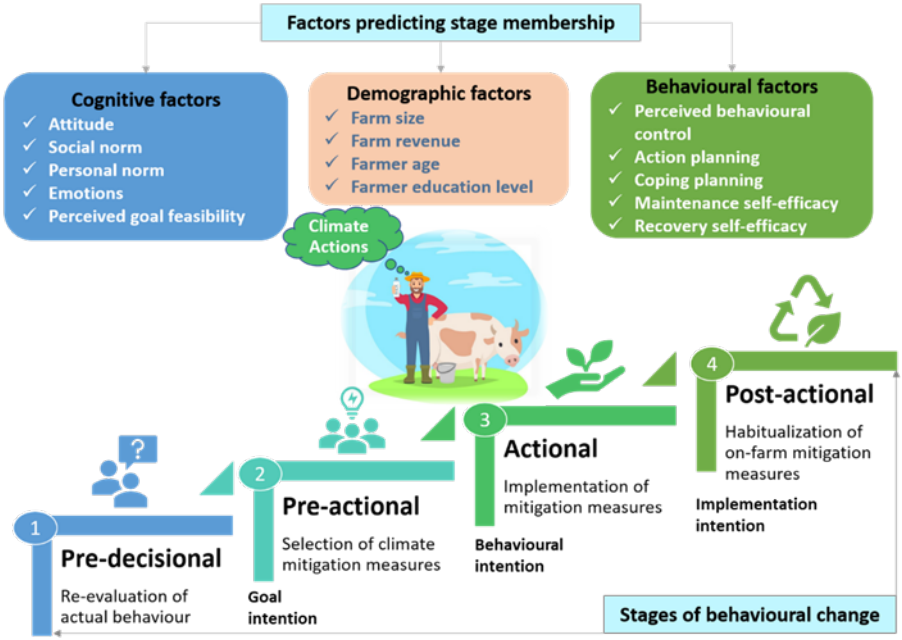


Figure 7 Self-regulated model of behavioural change (adapted from Bamberg, 2013)

Figure 8 depicts the interactions between statistical data, survey-based behavioural information, and the FarmDyn model. A data envelopment model for circular dairy farming will be developed and it will provide empirical support for the FarmDyn model. Using the dual approach, Data Envelopment Analysis (DEA) allows us to compute the shadow price of GHG emissions, that is, the farmers' willingness to pay to give up one unit of GHG emission. This will be used to calibrate d' in Figure 6 and thus feeds into the objective function in Figure 8 . Additionally, farmers' revealed preferences for engaging in GHG reducing activities will be included in the objective function (Figure 6 and Figure 8).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

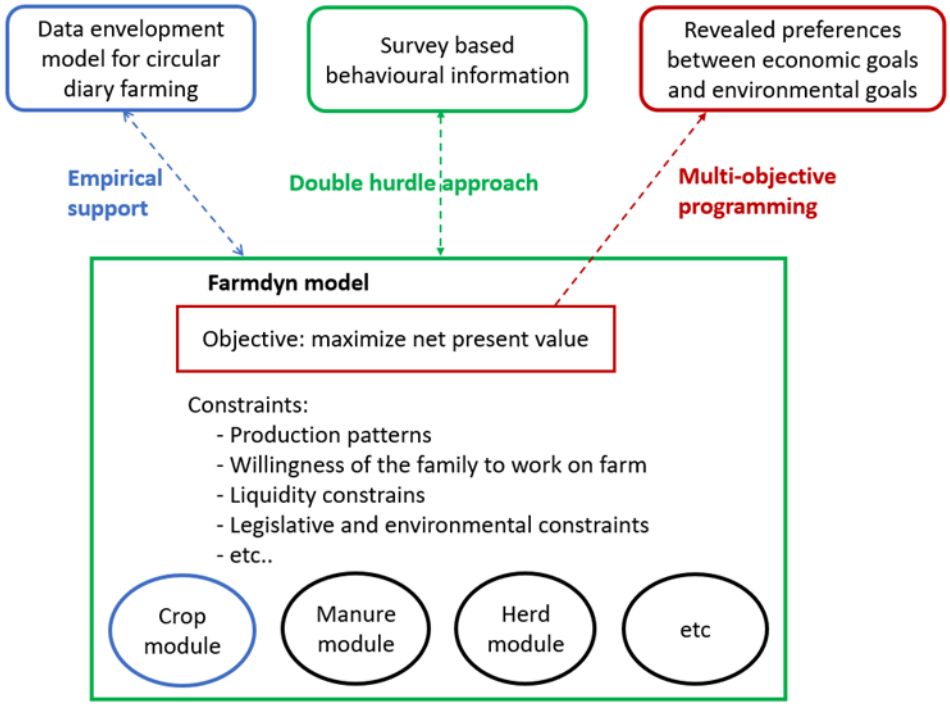


Figure 8 Three pathways to improve the FarmDyn model

A 'double hurdle approach' that combines results from survey and FarmDyn model will be implemented to model the adoption behaviour more realistically. The first hurdle comes from the predicted stage membership based on the self-regulated stage model. Stage membership is estimated by socio-psychological factors in the stage model (Figure 9). Only when the predicted membership is at actional stage, the second hurdle will be encountered which is the NPV based core FarmDyn model (Figure 9). In this way, it will likely provide a richer adoption prediction, as this 'double hurdle approach' includes not only the social-psychological factors reflected in self-regulated stage model, but also the economic factors (NPV) in FarmDyn model. Another important additional feature provided by this task is the identification of farmer intention to participate in GHG reduction. This will in effect restrict the number of additional management option. available in the farm model.

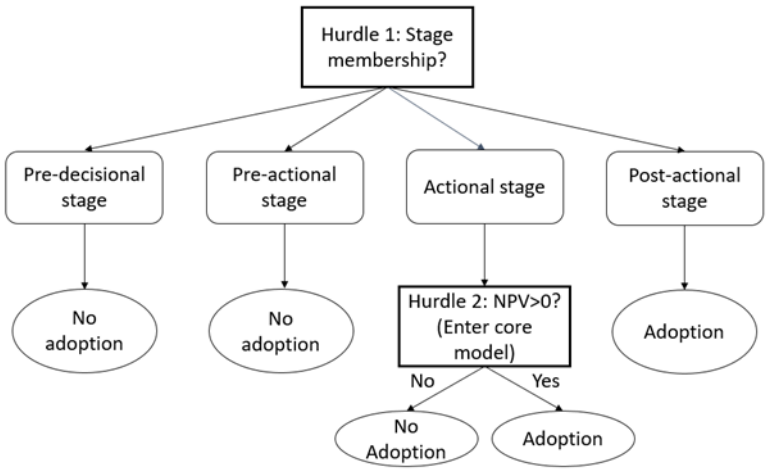


Figure 9 Double hurdle approach

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

3.4. Crop management choices

The main aim of the micro-econometric analyses conducted in Task 3.4 is to make the most of the information content of FADN datasets, possibly completed with soil and weather data, for feeding MP farm models. Tools developed in this task are designed to process FADN datasets for delivering (*a*) parameters to be directly used in MP farm models (*e.g.*, farm specific chemical input uses estimated at the crop level) and (*b*) items to be used for calibrating parameters of MP farm models (*e.g.*, farm specific crop acreage elasticities). The possible extensions of the core model (Figure 5) are illustrated in Figure 10: Farm-specific crop management options or intensity levels can be derived from observations to expand the number of cropping activities available to the MP. This leads in particular to a larger matrix of technical coefficients (A) and more refined information about the cost (c) associated with these new production activities. These changes are depicted by the lighter shaded areas in Figure 10. The mentioned crop acreage elasticities can be, for instance, be used by calibration techniques that involve adding a quadratic cost-term to the objective function (Q), where the diagonal elements of Q are typically derived from the response elasticities. Such an approach requires that the objective function itself is also a module that can be swapped if necessary (Figure 3).

The following sections provide an overview on the methodological approaches and some preliminary results from these research activities.

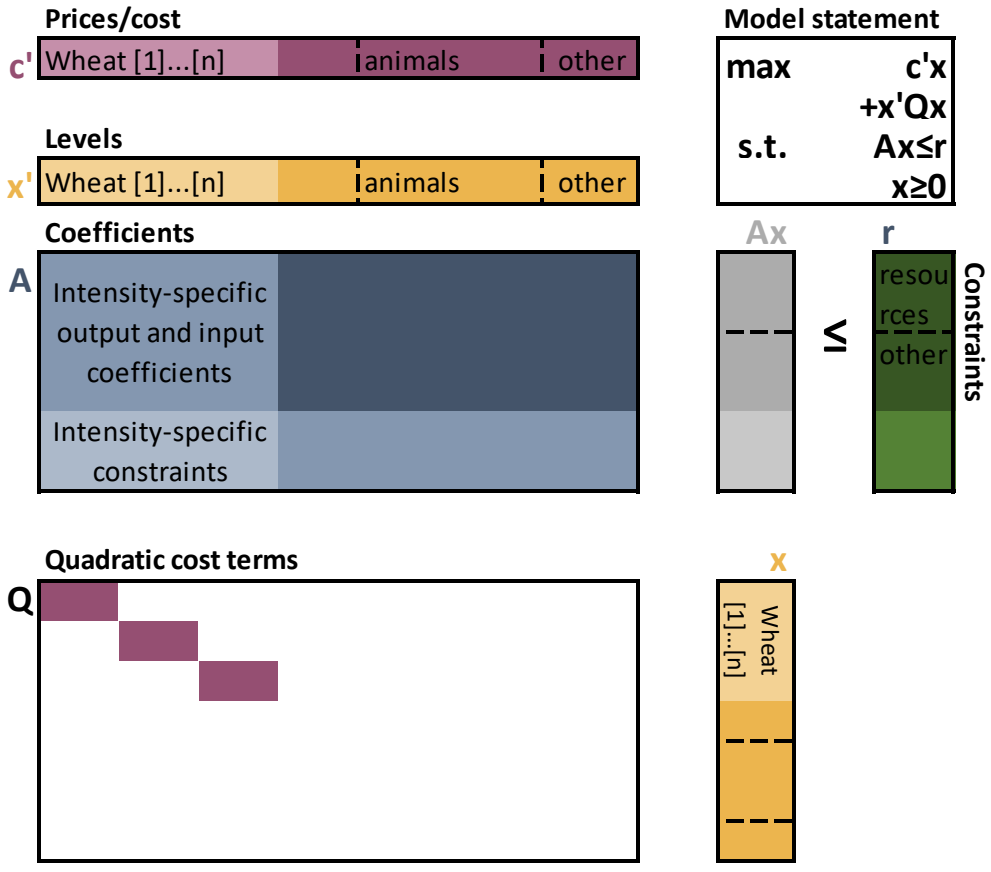


Figure 10 Extensions of the core model by using results from micro-econometric analyses

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

3.4.1. Random parameter micro-econometric multi-crop models, and farm specific crop acreage elasticities

The micro-econometric multi-crop models and estimation approaches considered in Task 3.4 are designed for two purposes: for being used as simple simulation models or for providing behavioral parameters for building other – more complex – simulation models (*e.g.*, IFM-CAP, GLOBIOM, MAGNET) as investigated in WP5. These micro-econometric modelling and estimation tools build on those previously proposed and developed by members of the MIND STEP consortium. As such, these estimation models are data preparation modules within the overarching IDM structure rather than run-time modules that form part of the model equations.

The considered micro-econometric multi-crop models are characterized by three main features. First, they rely on simple (yet theoretically consistent) functional forms of yield supply, input demand and crop acreage choice models, which make them empirically tractable (Carpentier and Letort 2012, 2014). Regarding acreage choices their basic structure is similar that of PMP models (*e.g.*, Heckelei et al 2014, Mérel and Howitt 2014, Britz and Arata 2019) and of a few multi-crop econometric models (*e.g.*, Chavas and Holt 1990, Heckelei and Wolff 2003). Second, these models feature farm specific random parameters, which make them especially relevant for capturing the effects of unobserved heterogeneous factors on farmers' production choices. The main purpose of the estimation of these models is to estimate the probability distribution of the random parameters in the population from which the sample is drawn. Third, these models explicitly account for crop production choices, and thus for null acreage choices, in a way that is fully consistent from a micro-economic viewpoint (Koutchadé *et al* 2020). These models can be estimated based on panel datasets reporting cost accounting data (Koutchadé *et al* 2018, 2020) as well as on standard accountancy panel datasets, such as typical FADN datasets (Carpentier *et al* 2014).

Once estimated, these random parameter micro-econometric multi-crop models enable the calibration of technical and behavioral parameters at the farm level based on a well-defined statistical background. For instance, farm specific crop acreage elasticities with respect to netput prices or crop returns can easily be derived from the estimated models (Carpentier *et al* 2014, Koutchadé *et al* 2018, 2020). Such elasticities can then be employed for calibrating farm specific parameters for MP farm models, including parameters involved in the PMP term of quadratic MP models or parameters involved in the constraint set of LP models (*e.g.*, parameters **A** and/or **r** inFigure 10).

Two avenues are pursued for making these models and their estimation approaches better suited to the toolbox of the community developing MP farm models. First, the considered micro-econometric multi-crop models for accounting for livestock production are extended. Second, estimating these models yield rich results but entails significant practical issues. These models are high-dimensional, feature random parameters and, in the case of the model of Koutchadé *et al* (2020), also feature endogenous switching regimes. Their estimation rely on specifically designed SAEM algorithms, which are extensions of standard EM algorithms featuring stochastic approximation and integration with simulation methods (Delyon *et al* 1999, Lavielle 2014). The approach used here consists of simplifying the specifications of these model (*e.g.*, by imposing restriction on their random parameters, by approximating some of their components) for significantly alleviating their estimation costs. The objective is to devise algorithms that are relatively easy to code and to provide suitable ranges for their tuning parameters (*e.g.*, simulation draw numbers and their evolution along the course of the algorithm).

3.4.2. Allocating chemical input uses to crops

Disaggregating variable input uses, which are reported at the farm level in FADN data, by allocating them to the crops of the considered farms can also alleviate the estimation burden of the micro-econometric multi-crop models considered above. Yet, disaggregating variable input uses for

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

obtaining estimated cost accounting data is also useful for obtaining input data for MP farm models (*e.g.*, crop production costs appear in parameter **c** inFigure 10).

Frequentist (*e.g.*, Dixon et al 1992, Carpentier and Letort 2012), Bayesian (*e.g.*, Gocht 2008, Louhichi *et al* 2012) or entropy based (*e.g.*, Léon *et al* 1999, Gocht 2008, Louhichi *et al* 2012) approaches were proposed for allocating input uses observed at the farm level to the farm activities. A major drawback of the approaches considered so far consists of their relying on crop input use models that fail to account for unobserved heterogeneity across farms. Typically, these approaches assume that crop input uses only depend on a few observed variables (*e.g.*, farm size, yield levels, regional effects), the information content of which is often limited. This information content is not sufficient for capturing the important heterogeneity in farm crop input uses. This heterogeneity is systemically displayed by cost accounting data, even in small areas (*e.g.*, Carpentier and Letort 2012, Koutchadé *et al* 2018).

This is addressed by extending previous approaches (*i*) by considering farm specific random parameter models for crop input use levels and (*ii*) by enforcing (stochastic) constraints on the estimated crop input uses at the farm level during the estimation process (Koutchadé *et al* 2021). Farm specific random parameters allow to account for unobserved heterogeneity in crop input use across farms and, as a result, permit to deliver crop farm specific crop input use estimates based on a well-defined statistical background. Enforcing (stochastic) constraints on the crop input uses estimated for each sampled farm enables to incorporate prior information in the estimation process in the form of (soft) boundary constraints. The considered input allocation equations can be estimated based on a mixed Bayesian-frequentist statistical framework (*e.g.*, Meza *et al* 2007) under stochastic constraints (*e.g.*, Wu *et al* 2019).

Promising results could be obtained with a “test” dataset that consists of an unbalanced panel dataset containing cost accounting data of a sample of 951 arable crop producers in Champagne region (NUTS2) from 1998 to 2014 (Koutchadé *et al* 2021). The estimated input allocation equations for pesticides and fertilizers account for the 11 crops that cover more that 90% of the considered farm acreages. Two types of constraints on estimated crop input uses are considered: (*i*) Standard non-negativity constraints are usually imposed on input uses. (ii) Constraints stating that the estimated crop input uses need to lie (in a probabilistic sense) between half (soft minimum bound) and twice (soft maximum bound) of the average crop input use observed in the data are imposed. The latter constraints allow to incorporate expert knowledge information or mean results of surveys on farmers’ practices in the estimation process.

Figure 11 to Figure 13 display the results for three selected crops – wheat, rapeseed and potato – under the standard non-negativity constraints. Input uses are measured per ha in 100€ at the 2005 prices. These figures plot the estimated per hectare input use levels against their “true” (observed) counterparts. Figure 11demonstrates that reasonably good results could be obtained when estimating fertilizer and pesticide input uses for winter wheat, which is produced by all sampled farmers and represents on average 35% of their arable crop acreage. Admittedly, our estimated input use levels significantly differ from their true counterparts. Yet, most estimates lie within reasonable ranges around their true counterparts. For instance the average difference between the true and estimated (in absolute value, AAD) fertilizer use equals .37 while the average fertilizer use equals about 2 (*i.e.*, about 200€/ha at the 2005 fertilizer prices). Yet, fertilizer uses are underestimated. Rapeseed is produced by 96% of the sampled farms but its average acreage share doesn’t exceed 15%. Figure 12shows that the estimated fertilizer and pesticide uses for rapeseed are less accurate than those for wheat, and that pesticide uses for rapeseed are overestimated. Figure 13shows that the estimation approach fits poorly the chemical input uses for potato production, which only concerns 11% of the sampled farms (for an average crop acreage shares of 2%).

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

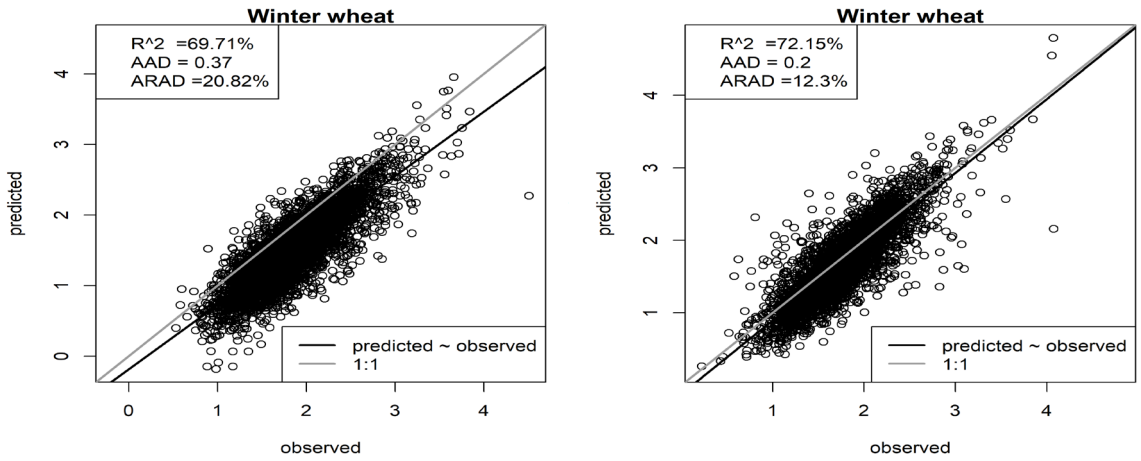


Figure 11 Estimated versus observed fertilizer (left) and pesticide (right) uses for wheat (Marne dataset, 100€/ha, at 2005 price levels)

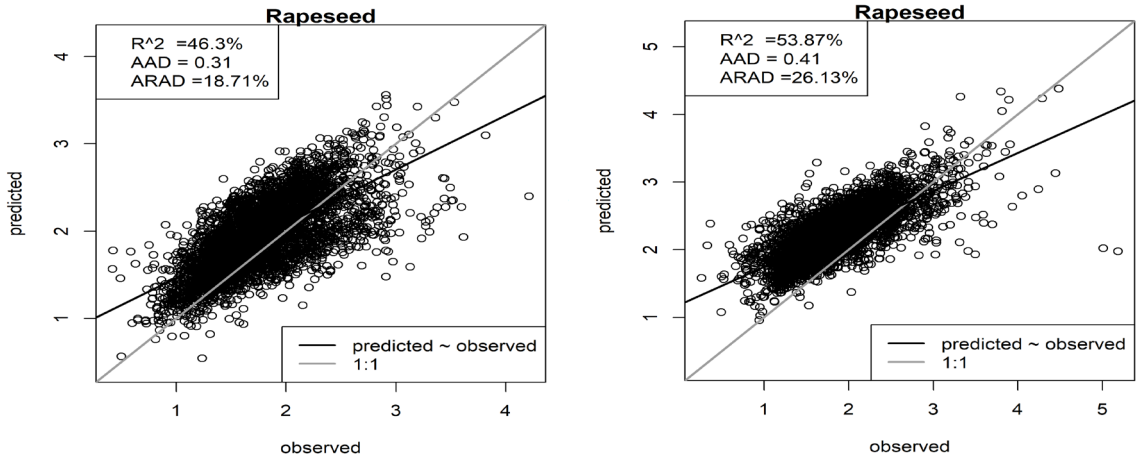


Figure 12 Estimated versus observed fertilizer (left) and pesticide (right) uses for rapeseed (Marne dataset, 100€/ha, at 2005 price levels)

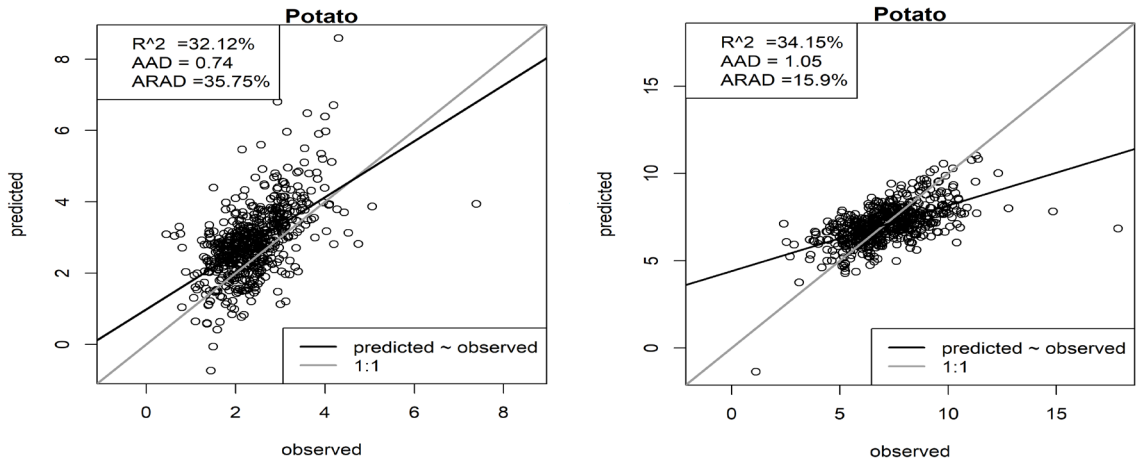


Figure 13 Estimated versus observed fertilizer (left) and pesticide (right) uses for potato (Marne dataset, 100€/ha, at 2005 price levels)

These results show that (*i*) recovering pesticide uses is generally more difficult than recovering fertilizer uses, (*ii*) estimation accuracy increases with the average acreage share of the considered crop

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

and (*iii*) average estimated input uses are close to their true counterparts, in general. As a matter of fact, these results are promising.

The effects of various constraints are currently under investigation. The final objective is (*i*) to characterize the models and constraint sets yielding the most accurate results and (*ii*) to devise estimation algorithms that are relatively easy to code (and to provide suitable ranges for their tuning parameters).

3.4.3. Uncovering adoption and characteristics of crop management practices

Accounting for adjustments in yield and variable input use levels is often difficult in MP farm models, mostly due to data constraints. The AROPAj model makes use of response functions of crop yields to nitrogen uses. These response functions are estimated based on (place specific) simulations obtained from the agronomic crop growth model STICS. Whereas AROPAj considers continuous adjustments in nitrogen uses, most other MP farm models consider discrete changes based on menus of crop production practices differing by their nitrogen use levels. Each pair composed of a crop and a practice is then considered as a specific activity in the considered MP farm models (seethe lighter shaded areas in Figure 10). GLOBIOM makes use of simulations obtained from the agronomic crop growth model EPIC for defining (place specific) “low-intensity” and “high-intensity” yield and nitrogen use levels. FarmDyn considers more comprehensive menus of production practices differing by their nitrogen use intensity levels (as well as by their tillage practices). The parameters of these menus are determined based on available agronomic data (see Figure 10).

Crop production technologies such as the ones considered in MP farm models are related to the concept of crop management practice (CMP) that is used by agronomists. Agronomists define crop management practices (CMPs) as sequences of operations and of input quantities used for producing crops. CMPs can be characterized by their relying on specific techniques (*e.g.*, reduced tillage) or by their target yield levels (*e.g.*, “high-yielding” CMPs are designed to achieve higher yield levels than “low-input” CMPs. “Low-input” refers here to chemical inputs, that is to say mineral fertilizers and chemical pesticides).

While considering varying levels of fertilizer uses (as well as of water uses) is possible thanks to available benchmark agronomic data or available agronomic crop growth models, considering varying levels of pesticide uses is much more difficult. Pesticide saving CMPs are poorly documented in the literature. This probably explains why agronomic crop growth models don’t consider crop protection in general, and pesticide use in particular. This leaves two options for adjusting pesticide uses in MP farm models. First, one can construct hypothetical CMPs, and the corresponding input use and yield levels, by combining data describing chemical input use and yield levels in conventional *versus* organic production practices. Data describing current farmers’ production choices mostly describe, by definition, the technical and economic performances of conventional practices. Organic production practices are well documented, especially because promoting organic production has been on the agenda of the EU (and of many member states) for a long time. Second, one can also try to uncover the adoption rate and the characteristics of CMPs of varying intensity levels from data describing farmers’ production choices.

Devilliers *et al* (2021) obtained preliminary results by pursuing the second option. These results tend to show that trying to recover the adoption rates and the characteristics of CMPs of varying chemical input use intensity from observational data is difficult and unlikely to be very useful for the purposes of the MIND STEP project. First, identifying characteristics and adoption patterns of latent CMPs is challenging. This requires cost accounting data that are rarely available. This also requires significant modelling and estimation efforts. Second, the obtained results tend to show that most farmers stick to conventional CMPs, that is to say to CMPs that are high yielding and intensive in chemical input

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

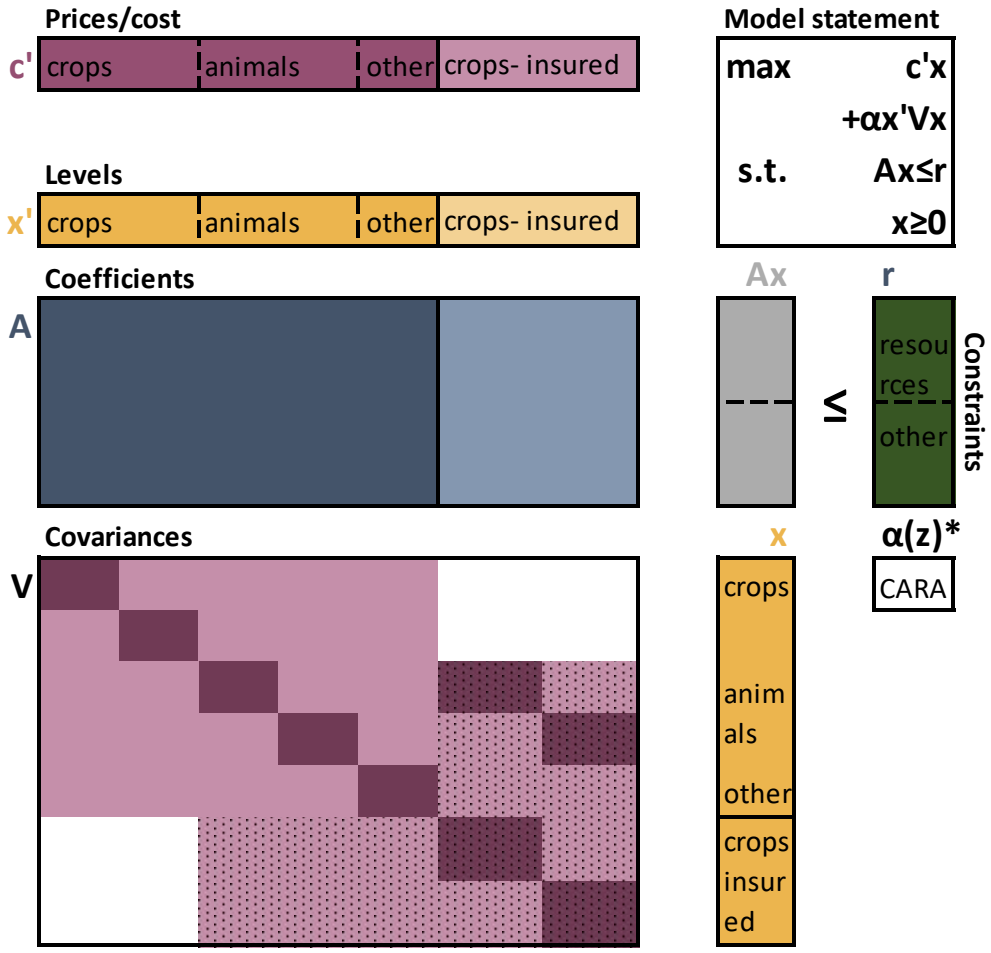
uses. These results are striking since the considered empirical application concerns an area where LI-CMPs were tested by scientists with the support of local farm organizations and of local downstream firms. While this hypothesis needs much more empirical support to be confirmed, these results strongly suggest that the best option to determine the characteristics of chemical input saving CMPs is to rely on data provided by agronomists and to process them for obtaining relevant input data for MP farm models.

3.5. Risk management models

The aim of Task 3.5 is to improve the capacity to model CAP risk management policies by developing an IDM model which captures the acceptance and risk-reducing impacts of different risk management instruments (RMI). The empirical application will focus on weather risk in crop farms. The analysis will be based on a large data set combining new survey data with existing data from detailed regional FADN and biophysical data (weather, soils, yields) according to the procedures developed in WP2. The survey will explore risk preferences and attitudes to use RMI of farmers in Germany and Italy. The output of this Task is a module to analyse the acceptance and risk-reducing impacts of different RMI. The module allows to analyse the propensity to adopt RMI for a range of behavioural theories and farm and farmers characteristics (e.g. household; off-farm income; wealth; personal traits), and to analyse the impact of RMI that reduce income volatility .

Incorporating RMIs into the framework of a MP model hinges on two types of conceptual decisions: First, how should the riskiness of a farm plan, and the farmer's attitude towards it, be represented in the objective function and second, what are the mechanisms through which the respective RMI contributes to the overall reduction of risk in the model. Concerning the former, mean-variance approaches are a typical way to incorporate the variability of farm outcomes and the farmer's preferences for lower or higher levels of this variance. A classical approach (Hazell and Norton, 1986) is to represent the mean farm outcome by a linear term in the objective function and the outcome variance by a quadratic term, weighted by the farmer's constant absolute risk aversion. Such a quadratic objective function results in a non-linear programming model, which may limit the choice of reliable numerical optimization algorithms. However, IFM-CAP and FarmDyn, the two currently operational MP models in the MIND STEP project can accommodate quadratic objective functions. The estimation of risk preferences within quadratic objective functions has been addressed by Arata et al. (2017). Figure 14 gives an example of such an MP model. The relation between levels of decision variables and the overall variance of farm outcomes is represented here by the covariance matrix V, weighted by the absolute risk aversion parameter α. Usually, not all variances and covariances between farming decisions can be measured, which is the reason for V to be partially empty. This requires ensuring that the core model can accommodate empty slots in the risk module, either by omitting the affected equation or by requiring the module to provide default values. Also, the source of risk is not specified in this diagram, it can be risk resulting from environmental conditions affection physical crop yields or risk from fluctuations of input and output prices on markets. The risk aversion parameter can be parameterized based on the survey results in 3.5 to depend on farm and/or farmers' characteristics.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.



*Risk aversion coefficient depending on vector z of farm/farmer's characteristics

Figure 14 Mean-Variance model with crop insurance as risk management instrument

While a quadratic representation of risk preferences in the objective function of the targeted IDM is generally fine regarding the numerical optimization, it should be emphasized that other equations in such an MP model should remain linear.

The adoption as well as the impact of RMI can be modelled endogenously in a mean-variance-framework by including the impact of the RMI on the mean return and variability of the respective activities. This requires a specification of the variance-covariance-matrix with RMI, e.g. based on past observations and the calculated impact of the RMI on single-activity returns (e.g., Hazell et al., 1986). However, "a growing body of evidence suggests that deviations from expected utility are quite common in agricultural technology adoption decisions, and agricultural adoption behaviour might be better explained by models incorporating this insight" (Streletskaya et al., 2020). While some alternative risk formulations or measures (e.g. mean absolute deviation) could be accounted for by respective alternative specifications of the objective function (and/or additional constraints), alternative behavioural theories may require a two-step modelling (e.g., the optimization of the objective function of cumulative prospect theory generally poses severe computational challenges when allowing for multiple risky activities). The first stage will cover the modelling the adoption of RMI, e.g. based on heuristics and/or cumulative prospect theory in a narrowly framed decision problem (see e.g. Babcock, 2015, for an application to insurance coverage), and the second step

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

modelling the impact on farm activity levels and income volatility given the chosen RMI using the extended model system. Similar to the risk aversion coefficient, parameters of the alternative adoption decision functions (e.g. probability weighting, loss aversion) can be specified to depend on farm/farmer's characteristics, and default values will be provided from the case studies in Italy and Germany. FADN input to the module includes farm (group) specific information on the variance of activity-specific yields and prices to specify the variance-covariance matrices, and farm/farmer's characteristics (e.g. farm size, farmer's age) which may influence risk behaviour.

4. THE IMPACT OF MODULARITY ON INTERACTIONS WITH WP4 AND WP5

The modular structure developed in WP3 enables two possible pathways in the interaction between models/modules developed in WP3 and established models working at landscape scale and beyond in WP4 and WP5. First, simulation results can be provided by the overarching model under different configurations as described in section 3.1. Second, each econometric model developed in the tasks 3.3-3.5 is able to provide by itself parameters to improve not only models in WP3, but also in WP4 and WP5. Regardless of the pathway, these data exchanges require clearly defined interface definitions, including protocols for potential aggregation and upscaling. The resulting interfaces process the parameters and simulation results from WP3 to allow their integration into the upstream models. Data processing by an interface can firstly encompass alignment of units and definitions, aggregation, for instance across IDMs, production activities or products, in order to match the resolution and extent of different scales such as space, time, products, activities, agents etc. of the large-scale model. Secondly, it can use more sophisticated micro-econometric or machine learning approaches to derive behavioural functions to integrate in the WP4 or WP5 models. The development of the specific interfaces and their implementation is foreseen in task 4.1 and 5.1, whereas this section as part of task 3.1 lays down the general requirements for model connections and develops guidelines for interface development. Figure 15 5 illustrates a complete list of connections between the models in the different WPs, with their corresponding interface levels. Each interface refers either to the use of simulation results from the IDM models or to the parameterization of large-scale models based on econometrics models.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

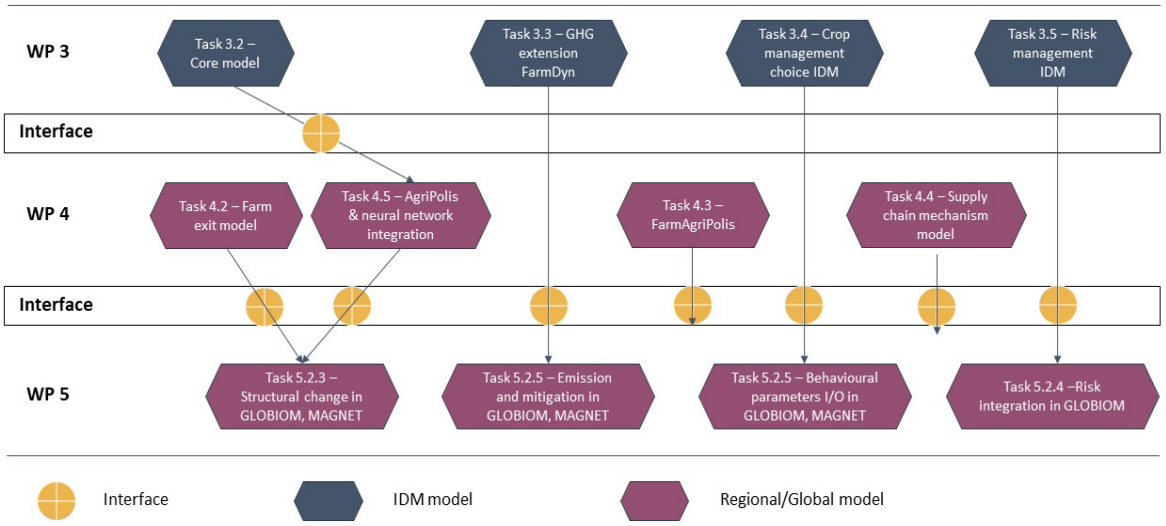


Figure 15: Connection of MIND STEP IDM models WP4 and WP5 models as defined in the Grant Agreement

4.1. General guidelines for both overarching and econometric model connections to WP4 and WP5

The interface of IDM models in WP3 and WP4/WP5 all differ in their capability to reflect different details with respect to space, agents such as farmers differentiated by farm type and size, time and dynamics, and policy specifications. In order to harmonise the IDM in WP3 and WP4/WP5 models, this section develops general requirements and guidelines for all interfaces which are going to be developed in the course of the MIND STEP project.

Regional and farm type specification: The IDM models in WP3 need to be parameterized and set-up in order to reflect farm types, farm sizes, management options, technology, available crops and policy and market conditions for a specific region; factors which all vary substantially across the EU member states and regions within, reflecting the highly heterogenous EU farming structure. The upstream WP4/5 model receiving simulation results or parameters therefore has to define the regional focus for the application of the WP3 IDM models. This focus can stretch from a small region within one Member State to multiple Member States. Regardless of the spatial scope, the upstream model has either to indicate that a representative full population of farms has to be simulated or provide information on relevant farm types, farm sizes etc. to cover in simulations. To harmonize the regional nomenclature of both models, the regional specifications of the interface should be given in NUTS regions. Equally, farm typologies (specialization, size etc.) should follow the typology for agricultural holdings as used in the FADN database.

Time and dynamic specification: The interface have to define for which years (ex-post or ex-ante) the simulation results or parameters are to be provided for. Simulation runs in IDM models are often done with yearly steps, either recursive or fully dynamic. In contrast, the WP5 market scale models rather run in comparative-static mode and depict a new equilibrium over a typically medium time horizon, such as ten years. Resulting from this, the interface has to specify the time horizon and the yearly intervals relevant for the IDM model.

Policy and market environment specification: Policy measures within the CAP and other EU legislation such as the Nitrate and Water framework directive increasingly relate to specific single farm attributes. The overarching model needs to be able to capture Member state and potentially region-specific implementations of key policy measures in detail. A potential list of measures and related scenarios is

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

provided by task 1.1 WP4 and WP5 models, operating at landscape scale and beyond, abstract with varying degree from the farm specific implementation of policy measures. This can relate to using average farm attributes while still depicting the policy measure in some detail. An example provides the implementation of Ecological Focus Area conditions within CAPRI-FT. Market scale models might instead summarize the impact of policy instead by some ad-valorem subsidy or tax rate as typically found in Computable General Equilibrium models, potentially combined with some parameter adjustments to reflect, for instance, reduce price responsiveness due to restrictions. The interface definitions need hence to specify which policy measures to cover and which parameters at which resolution are used/adjusted in the upstream model to depict them.

A challenge provides price endogeneity in market models. This precludes simply superimposing future growth rates of prices in IDM models which are endogenous in the WP4/WP5 target model. This might require some looped structure to update prices in IDM and parameters in market scale model recursively, depending on the application. Independent of this, changes in all prices which are exogenous to both the WP3 and WP4/WP5 model should be harmonized in in ex-ante analysis.

4.2. Requirements related to the interface of the overarching model and models in WP4/ WP5

As discussed in section 3.1 the core IDM model is a BEFM model programmed as a mixed-integer mathematical programming model, potentially with a strictly convex quadratic objective instead of a linear one. The use of integer variables is necessary to consider for instance indivisibilities in farm assets and if conditions in the implementation of policies. Performant algorithms for large-scale modelling combining integers and non-linear constraints are still not available such that linear constraints only are envisaged. Non-linear production functions etc. need therefore to be approximated by a convex set of Leontief production activities. The interface definitions specify which of the many simulations results from optimizing the IDM model are provided to the WP4/WP5 models. This is likely a restricted set of key farm indicators.

Key indicators: The core model provides a list of key farm indicators regardless of the data receiving large scale model. Such key indicators relate to the economic domain (revenues, variable and fixed costs, subsidies received, and indicators derived thereof such as profits etc.), social domain (such as labour use and distribution over time), environmental ones (GHG emissions, nitrogen and phosphate balances etc.) and to material balances (product output and input use). Existing indicators are extended by new indicators developed within the MIND STEP project task 1.2 and additional indicators relevant for policies developed in task 1.1. Especially for economic indicators and material balances information on the exact definition of production activities, inputs and outputs needs to be provided, typically in form of concordance lists also stating the relevant units and definitions. Table 1 and 2 provide examples for indicators and netputs, respectively, and should be used as templates for the concrete implementation of interface definition in task 4.1

Table 1: Examples of defining concordance list to develop interface definitions for economic indicators

Economic indicators	WP3 model	WP5 model	Required action for interface
Revenues relating to output sales	Measured at farm gate prices included VAT, local currency	Market prices, EU	Estimation of marketing mark ups, exchange rates where needed
Subsidies	Linked to production activities	Linked to output/input use	Conversion with yields for outputs

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Economic indicators	WP3 model	WP5 model	Required action for interface
….	…..	….	….

Table 2: Example of output matching tables

Output	WP3 model	WP5 model	Required action for interface
Oats	Metric tons, output including all losses up to sale from farm gate	Other cereals, post-harvest losses not considered (separate element in market balance), in 1000 metric tons	Aggregation, multiply with (1+loss rate), unit conversion
Rye	"	"	"
….	…..	….	….

In order to illustrate such a connection, figure 14 shows as an example of a more complex model connection between the farm level model FarmDyn (task 3.2/3.3/4.5) and AgriPolis (task 4.3/4.5) realized in the task 4.5.

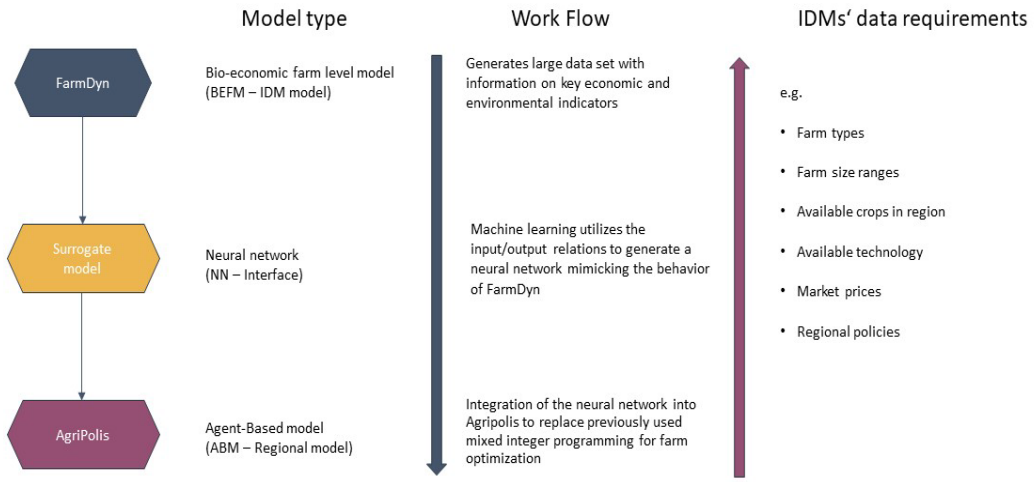


Figure 16: Simplified illustration of the integration of FarmDyn as a neural network into AgriPolis and the related requirements defined by the large-scale model.

The linkage between AgriPolis (WP4 Agent Based Model at landscape scale) and FarmDyn (WP3 bio-economic farm-scale model) is realized through a neural network trained by data which is generated with FarmDyn. The neural network is subsequently implemented into AgriPolis to replicate as best as possible the simulation behaviour of FarmDyn. Specifically, the neural network shall replicate the input-output relationships of different farm types with varying size classes, under different market and policy environments and the initial available technology. The required parameters to consider in FarmDyn are determined by the specific region to which AgriPolis is applied. This limits the list of available crops, farm branches and technology etc. This type of model connection presents a more

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

complex linkage between the models where the interface itself is a sophisticated model (here a neural network). Defining such an interface requires close interactions between experienced users of the down- and upstream model. Other interfaces, such as to the stand-alone models in WP3, might be easier to achieve as presented in the following.

4.3. Requirements related to the interface of the econometric model and models in WP4/ WP5

The IDM models developed in task 3.4 and 3.5 are a micro-econometric multi-crop choice model and a micro-econometric risk model, respectively. These models can be used in stand-alone mode to provide specific insights on crop choices and risk or can be used to parameterize WP3 to WP5 models. Generally, these econometric models have to follow the general guidelines defined in section 4.1, however, the level of detail might differ substantially to the overarching model with respect to policies, for example. The direct upscaling from the econometric model to a WP5 model bypasses the application of the single-farm model such that information relating to simulation results such as key farm indicators is not necessary. However, the connection between the econometric models and the models in WP4/WP5 are contingent on a well elaborated scenario design.

Scenario design specifications: The parameterization of large-scale models from econometric IDM models require assumptions on the potential market/policy environments depicted by both models. Depending on the envisaged application of the large-scale model, the parameters or functions provided of the IDM model might have to capture a wide range of price variations. A simple example would be the use of own-price elasticities which calls for defining the relevant range of considered own- price changes in the IDM model. A systematic approach considering different magnitudes of price changes will also deliver information on the stability of the elasticities. Deriving a full set of cross-price elasticities is a more challenging exercise as relevant combinations of price changes need to be defined. Here, design of experiments and similar methods can help to reduce the number of necessary experiments and account for co-variances. The discussion on price elasticities underlines further challenges in interface development which go beyond harmonization of definitions of multiple scales with the related data processing.

5. QUALITY CRITERIA GUIDELINES FOR MODELS IN THE MIND STEP TOOLBOX

The MIND STEP model toolbox contains both long-standing models developed and extended over time within previous projects and new models to be developed in the course of the project. To ensure that collaboration between partners within the toolbox is easy to realize, we define in this section quality criteria guidelines which focus on the documentation, testing strategies, and proposed coding conventions. The quality criteria requirements are loosely based on the “Quality criteria for models and datasets” from the Wageningen Modelling Group (Appendix 2). Testing strategies (Appendix 3) and coding conventions (Appendix 1) from existing models are presented to exemplify potential implementations of the defined quality criteria. It is important to stress that these quality criteria, however, are to be seen as guidelines and not mandated to be implemented. Especially, the code of long-standing existing models is not to be rewritten to adhere to coding conventions, nor does this document impose a fixed testing strategy to be implemented. Rather, it gives examples and guidelines how to implement and improve code and testing strategies. With the variety of economic models ranging from mathematical programming and econometric IDM models to regional and global models encompassing agent-based models, computable general equilibrium, and partial equilibrium models, we deviate here from the thought of “one size fits all”, but rather set the framework for a consistent

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

model quality and transparency of models within the MIND STEP toolbox. In the following, we will present the quality criteria for the MIND STEP model toolbox:

5.1. Documentation

A short description with purpose/objective of the model and its area of application (spatial and conceptual) with the underlying theoretical framework has to be provided. The conceptual model is described and presented as a flow-diagram to improve understanding of the underlying assumptions and simplifications of the model code. Information on technical implementation and environment including the programming language (PL), integrated developer environment, graphical user interface and the version of the used PL and packages should be provided. This basic information is to be made available on the project website together with contact information for each of the models within the project.

5.2. Quality management

The quality management for the MIND STEP model toolbox comprises coding conventions and testing strategies. Coding conventions include syntactic and code commenting guidelines to, first, improve a common understanding of the model code for all involved project partners, and, second, to ensure that code can be maintained even if original coders are no longer available. Commenting of code includes a description of all parameters and variables in the code, as well as their respective units. Furthermore, with the modular structure of several MIND STEP toolbox models, connections and dependencies are highlighted in the code. In addition, the coding conventions encompass that input data and their sources are described, as well as a description of output data to link them to scientifically relevant indicators. An exemplary coding implementation for the MIND STEP model FarmDyn written in GAMS is given in Appendix 3.

The implementation of testing strategies is crucial in multi-partner projects which work both with one model or even with linkages between models. Testing strategies are essential to mitigate the risk of the error prone developments of models when multiple persons are involved. Such errors can range from simple syntactic errors given at compile time, to execution errors where mathematical infeasibilities are returned by the model. Where those two types of errors are most often picked up by compile time and run time tests, the more hazardous errors lie in wrong outcomes which can have grave impacts in policy debates and hence require a proper interpretation of specialists in the respective fields. Test results should not only be validated by the responsible modeller, but also validated by either literature or external experts and evaluated against a benchmark. The implemented testing strategy requires that tests are documented, following a certain protocol and are periodically re-evaluated to ensure that new components of the model are included. Furthermore, the testing strategy has to be adapted to the specifics of the programming language, type and size of the model and should also account for modularity in the technical implementation. An exemplary testing strategy is presented for FarmDyn, and can possibly relatively easily adapted for those models using the GGIG Graphical User Interface Generator developed by Wolfgang Britz (CAPRI, IFM-CAP, FarmDyn). Each model should use a version control system (SVN, GIT, or similar) to track and monitor changes in the model code. This does not only facilitate the collaboration between programmers, but also helps to follow different development path of the model and to simplify the merging of different model threads. Project partners with models using multiple branches are encouraged to have a common development plan to prevent duplicate work or to alleviate potential conflicts in the development or data structure.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

6. CONCLUSION

This deliverable 3.1 outlined concepts for establishing a modular model toolbox in the MIND STEP project. A crucial building block for this deliverable was the literature review and in-depth review of four applied models by Britz at al., 2021. Starting from their approach to structure a modular system of interlinked models around a core model, chapter 3 provided an overview on the included task and their respective links and contributions to a common simulation model at farm level. Subsequently, the links and work-flows related to models at higher organisational scales are outlined in chapter 4. Due to the importance of model maintenance and quality management, chapter 5 highlighted some important concepts and pointed to the appendices in the remainder of this deliverable. In there, guidelines for good coding practices, quality management and model transparency are provided. As this deliverable aims to serve as a reference handbook for model development and integration for the MND STEP partners, these appendices may be useful to the involved model developers and users.

7. ACKNOWLEDGEMENTS

This deliverable 3.1 is developed as part of the H2020 MIND STEP project which received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

8. REFERENCES

Arata, L., Donati, M., Sckokai, P., Arfini, F., 2017. Incorporating risk in a positive mathematical programming framework: a dual approach. Aust. J. Agric. Resour. Econ. 61, 265–284. https://doi.org/10.1111/1467-8489.12199

Appel F., Balmann, A., Dong, C. and J. Rommel, 2018. FarmAgriPoliS-An Agricultural Business Management Game for Behavioral Experiments, Teaching, and Gaming. IAMO Discussion Paper 173, DOI: 10.13140/RG.2.2.27125.68320.

Bamberg, Sebastian. "Changing environmentally harmful behaviors: A stage model of self-regulated behavioral change." *Journal of Environmental Psychology* 34 (2013): 151-159.

Babcock, Bruce A. (2015): Using Cumulative Prospect Theory to Explain Anomalous Crop Insurance Coverage Choice. In: American Journal of Agricultural Economics 97 (5), S. 1371–1384. DOI: 10.1093/ajae/aav032.

Britz, W., 2020. Automated calibration of farm-scale mixed linear programming models using bi-level programming. Discuss. Pap. Ser. "Food Resour. Econ. 2020.

Britz, W. and L. Arata, 2019. Econometric mathematical programming: an application to the estimation of costs and risk preferences at farm level. Agricultural Economics, 50(2):191–206.

Britz, W., Ciaian, P., Gocht, A., Kanellopoulos, A., Kremmydas, D., Müller, M., Petsakos, A., Reidsma, P., 2021. A design for a generic and modular bio-economic farm model. Agric. Syst. 191, 103133. https://doi.org/10.1016/j.agsy.2021.103133

Britz W, Witzke P. CAPRI model documentation 2014, http://www.capri-model.org/docs/capri_documentation.pdf; 2014

Britz W., Lengers, B,. Kuhn, T. and D. Schäfer, 2016. A highly detailed template model for dynamic optimization of farms - FARMDYN, University of Bonn, Institute for Food and Resource Economics, http://www.ilr.uni-bonn.de/em/rsrch/farmdyn/farmdyn_docu.pdf.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Bussieck, M.R., Meeraus, A., 2004. General Algebraic Modeling System (GAMS), in: Kallrath, J. (Ed.), Modeling Languages in Mathematical Optimization. Springer, pp. 137–157. https://doi.org/10.1007/978-1-4613-0215-5_8

Carpentier, A., F. Féménia and Ph. Koutchadé, 2014. Accounting for unobserved heterogeneity in agricultural production choice models: a random parameter approach. Annual Meeting of the Agricultural and Applied Economics Association. Minneapolis, Minnesota, Juillet 2014.

Carpentier, A. and E. Letort, 2014. Modelling acreage decisions within the Multinomial Logit framework: profit functions and discrete choice models. Environmental and Resource Economics, 59(4), 537–559.

Carpentier, A. and E. Letort, 2012. Accounting for heterogeneity in multicrop micro-econometric models. Implications for variable input demand modeling. American Journal of Agricultural Economics, 94(1): 209–224.

Chavas, J.-P., and M.T. Holt, 1990. "Acreage decisions under risk: the case of corn and soybeans." American Journal of Agricultural Economics 72(3):529–538.

Cherchye L., C. Lovell, W. Moesen and T. Van Puyenbroeck, 2007. One market, one number? A composite indicator assessment of EU internal market dynamics, European Economic Review 51 (3), 749-779.

Delattre, M., and M. Lavielle 2012. Maximum likelihood estimation in discrete mixed hidden Markov models using the SAEM algorithm. Computational Statistics & Data Analysis, 56(6):2073–2085.

Delyon, B., Lavielle, M., and E. Moulines, 1999. "Convergence of a stochastic approximation version of the EM algorithm." Annals of Statistics 27(1):94–128.

Devilliers, E., A. Carpentier and O.P. Koutchadé, 2021. Uncovering adoption and characteristics of "low-input" versus "high-yielding" crop management practices for wheat production in France: a heterogeneous hidden Markov approach. Working paper in progress, UMR SMART-LERCO, INRAE-Agrocampus Ouest, Rennes.

Dijkstra, E.W., 1982. Selected writings on computing-a personal perspective. Texts and monographs in computer science. Springer, doi 10, 971–978.

Dixon, B.L., and R.H. Hornbaker, 1992. Estimating the technology coefficient in linear programming models. American journal of agricultural economics, 74: 1029–1039.

Femenia F. and E. Letort, 2016. How to significantly reduce pesticide use: An empirical evaluation of the impacts of pesticide taxation associated with a change in cropping practice, Ecological Economics, 125, 27–37.

Gocht A and N. Röder, 2014. Using a Bayesian estimator to combine information from a cluster analysis and remote sensing data to estimate high-resolution data for agricultural production in Germany. International Journal of Geographical Information Science, 28 (9): 1744-1764, doi: 10.1080/13658816.2014.897348.

Gocht, A., 2008. Estimating input allocation for farm supply models. Paper presented at the 107th EAAE Seminar "Modeling of Agricultural and Rural Development Policies". Sevilla (Spain), January 29th -February 1st, 2008.

Grashof-Bokdam C., Cormont A., Polman N., Westerhof E., Franke J. and P. Opdam, 2017. Modelling shifts between mono-and multifunctional farming systems: the importance of social and economic drivers. Landscape Ecology 32, no. 3 (2017): 595-607.

Happe K., Kellermann, K., and A. Balmann, 2006. Agent-based Analysis of Agricultrual Policies: An

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Illustration of the Agricultural Policy Simulator AgriPoliS, its Adaptation and Behaviour. Ecology and Society, 11(1).

Harding, M.C., and J. Hausman, 2007. Using a Laplace approximation to estimate the random coefficients logit model by nonlinear least squares. International Economic Review, 48(4):1311–1328.

Havlík P, Valin H, Herrero M, Obersteiner M, Schmid E, Rufino MC, et al., 2014. Climate change mitigation through livestock system transitions. Proceedings of the National Academy of Sciences.

Hazell P, Bassoco LM, Arcia G (1986) A model for evaluating farmers' demand for insurance: applications in Mexico and Panama." In: Hazell P, Pomareda C, Valdes A (eds) Crop insurance for agricultural development, issues and experience. Johns Hopkins Univ Press, Baltimore, Maryland, 35–66

Hazell, P.B.R., Norton, R.D., 1986. Mathematical programming for economic analysis in agriculture. Macmillan, New York.

Heckelei, T., Britz, W., and Y. Zhang, 2012. "Positive mathematical programming approaches–recent developments in literature and applied modelling." Bio-Based and Applied Economics 1(1):109–124.

Heckeleï, T., and H. Wolff, 2003. "Estimation of constrained optimisation models for agricultural supply analysis based on generalised maximum entropy." European Review of Agricultural Economics 30(1):27-50.

Hoog, van der, S. 2017. Deep Learning in (and of) Agent-Based Models: A Prospectus. arXiv:1706.06302.

Janssen, S., van Ittersum, M.K., 2007. Assessing farm innovations and responses to policies: A review of bio-economic farm models. Agric. Syst. 94, 622–636. https://doi.org/10.1016/j.agsy.2007.03.001

Kahneman D. and A. Tversky 1979. Prospect Theory: An Analysis of Decision Under Risk. Econometrica 47(2):263–92.

Koutchadé O.P., Carpentier A. and F. Féménia, 2018. Modeling Heterogeneous Farm Responses to European Union Biofuel Support with a Random Parameter Multicrop Model. American Journal of Agricultural Economics, 100(2), 434-455.

Koutchadé, O.P., A. Carpentier and F. Féménia. 2020. Modelling corners, kinks and jumps in crop acreage choices: impacts of the UE support to protein crops. American Journal of Agricultural Economics, forthcoming.

Koutchadé, O.P., F. Féménia and A. Carpentier. 2021. Variable input allocation among crops: a random parameter approach with stochastic constraints using a SAEM algorithm. Selected presentation, Triennial Congress of European Association of Agricultural Economists, Prague.

Lamperti F., Roventini A and S. Amir, 2017. Agent-Based Model Calibration Using Machine Learning Surrogates. LEM Working Paper. Available at SSRN: https://ssrn.com/abstract=2943297 or http://dx.doi.org/10.2139/ssrn.2943297.

Lavielle, M. 2014. Mixed effects models for the population approach: models, tasks, methods and tools. New York: Chapman and Hall/CRC.

Leip, A., Koeble, R., Rottlan Puig, X., Reuter H. and M. Lamboni, In prep. Homogeneous Spatial Units – a Pan-European geographical basis for environmental and socio-economic modelling.

Lengers B., Britz, W. and K. Holm-Müller, 2014. What drives marginal abatement costs of greenhouse

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

gases on dairy farms? A meta-modelling approach, Journal of Agricultural Economics 65(3): 579–599

Léon, Y., L. Peeters, M. Quinqu and and Y. Surry. 1999. The Use of Maximum Entropy to Estimate Input-Output Coefficients From Regional Farm Accounting Data. Journal of Agricultural Economics, 50:425–439.

Lieberherr, K. J., & , I. M. Holland 1989. Assuring good style for object-oriented programs. IEEE software, 6(5), 38-48.

Louhichi, K., F. Jacquet and J.-P. Butault. 2012. Estimating input allocation from heterogeneous data sources: A comparison of alternative estimation approaches. Agricultural Economic Review, 13(2):83–102.

Louhichi K., P. Ciaian, M. Espinosa, A. Perni and S. Gomez y Paloma, 2017. Economic impacts of CAP greening: application of an EU-wide individual farm model for CAP analysis (IFM-CAP). European Review of Agricultural Economics, https://doi.org/10.1093/erae/jbx029.

Loyce, C. and J.-M. Meynard, 1997. Low input wheat management techniques are more efficient in ethanol production. Industrial Crops and Products, 6:271–283.

Loyce, C., J.-M. Meynard, C. Bouchard, B. Rolland, P. Lonnet, P. Bataillon, M.-H. Bernicot, M. Bonnefoy, X. Charrier, B. Debote, T. Demarquet, B. Duperrier, I. Félix, D. Heddadj, O. Leblanc, M. Leleu, P. Mangin, M. Méausoone and G. Doussinault, 2012. Growing winter wheat cultivars under different management intensities in France: A multicriteria assessment based on economic, energetic and environmental indicators. Field Crops Research, 125:167–178.

Loyce, C., J.-M. Meynard, C. Bouchard, B. Rolland, P. Lonnet, P. Bataillon, M.-H. Bernicot, M. Bonnefoy, X. Charrier, B. Debote, T. Demarquet, B. Duperrier, I. Félix, D. Heddadj, O. Leblanc, M. Leleu, P. Mangin, M. Méausoone and G. Doussinault, 2008. Interaction between cultivar and crop management effects on winter wheat diseases, lodging, and yield. Crop Protection, 27:1131–1142.

McLachlan, G., and T. Krishnan. 2007. The EM algorithm and extensions, 2nd. ed. New York: John Wiley & Sons.

Mérel, P., and R. Howitt. 2014. “Theory and application of positive mathematical programming in agriculture and the environment.” Annual Review of Resources Economics 6(1):451–470.

Mittenzwei, K. and W. Britz, 2018. Analysing farm-specific payments for Norway using the Agrispace model. Journal of Agricultural Economics (https://doi.org/10.1111/1477-9552.12268)

Neuenfeldt S and A. Gocht, 2014. A handbook on the use of FADN database in programming models. Braunschweig: Johann Heinrich von THUENEN-Institut, 75 p, THUENEN Working Paper 35, DOI:10.3220/WP_35_2014

Neuenfeldt S, Gocht A, Ciaian P and T. Heckelei, 2018. Explaining farm structural change in the European agriculture: A novel analytical framework. European Review of Agricultural Economics. Forthcoming.

Nowicki, P., et al., 2009. Study on the Impact of Modulation. Contract No. 30 - CE-0200286/00-21. Directorate-General Agriculture and Rural Development, European Commission, Brussels.

Parnas, D.L., 1972. On the criteria to be used in decomposing systems into modules, in: Pioneers and Their Contributions to Software Engineering. Springer, pp. 479–498.

Pinheiro, J. C., and D.M. Bates. 1996. Unconstrained parametrizations for variance-covariance matrices. Statistics and computing, 6(3):289–296.

Reidsma, P., Janssen, S., Jansen, J., van Ittersum, M.K., 2018. On the development and use of farm

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

models for policy impact assessment in the European Union – A review. Agric. Syst. 159, 111–125. https://doi.org/10.1016/j.agsy.2017.10.012

Russell, A.L., 2012. Modularity: An interdisciplinary history of an ordering concept. Inf. Cult. 47, 257–287.

Schouten M., Opdam P., Polman N. and E. Westerhof, 2013. Resilience-based governance in rural landscapes: experiments with agri-environment schemes using a spatially explicit agent-based model. Land Use Policy 30, no. 1 (2013): 934-943.

Soregaroli, C., Sckokai P., and D. Moro, 2011. Agricultural policy modelling under imperfect competition. Journal of Policy Modelling, 33(2), 195-212. doi: 10.1016/j.jpolmod.2010.12.001

Sahrbacher C., A. Sahrbacher and A. Balmann, 2013. Parameterisation of AgriPoliS: A Model of Agricultural Structural Change. In: Empirical Agent-Based Modelling - Challenges and Solutions, Editors: Alexander Smajgl, Olivier Barreteau. Springer New York. DOI 10.1007/978-1-4614-6134-0_6

Spiegel, A., Britz, W., Djanibekov, U. and R. Finger, 2018. Policy analysis of perennial energy crop cultivation at the farm level: Short rotation coppice (SRC) in Germany, Biomass and Bioenergy 110: 41-56

Stevens, W.P., Myers, G.J., Constantine, L.L., 1974. Structured design. IBM Syst. J. 13, 115–139.

Storm, H., Mittenzwei, K. and T. Heckelei, 2015. Direct payments, spatial competition and farm survival in Norway. American Journal of Agricultural Economcis 97(4): 1192-1205

Streletskaya, N., S. Bell, M. Kecinski, T. Li, S. Banerjee, L. Palm‐Forster, D. Pannell (2020): Agricultural adoption and behavioral economics: bridging the gap. Appl. Econ. Perspect. Pol., 42 (1) (2020), pp. 54-66. https://doi.org/10.1002/aepp.13006

van der Hoek, A., Lopez, N., 2011. A design perspective on modularity, in: Proceedings of the Tenth International Conference on Aspect-Oriented Software Development. pp. 265–280.

Woltjer, G. and M. Kuiper (eds.) 2014. The MAGNET model - Module description. LEI Report 14-057. The Hague: LEI - part of Wageningen UR (University & Research Centre). Wageningen, the Netherlands.

Wu, J., J.X. Wang and S.C. Shadden. 2019. Adding constraints to Bayesian inverse problems. Proceedings of the AAAI Conference on Artificial Intelligence, 33(1):1666–1673.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

APPENDIX 1: CODING GUIDELINES FOR CORE MODEL AND CONTRIBUTED MODULES IN MIND STEP

Objective

The core model in MIND STEP developed in Task 3.2. will be programmed in the programming language GAMS. The objective of this guideline is to develop a coding convention for all developed models and contributed modules in MIND STEP written in GAMS to ensure that the code:

- can be easily *understood* by another programmer
- can be successfully *maintained* and updated
- and can source an *automated code documentation* system.

The Java code conventions (http://java.sun.com/docs/codeconv/html/CodeConventions.doc) give the following reasons to establish coding conventions: "*Code conventions are important to programmers for a number of reasons:*

- *80% of the lifetime cost of a piece of software goes to maintenance.*
- *Hardly any software is maintained for its whole life by the original author.*
- *Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.*
- *If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product you create."*

As the models in MIND STEP are developed and maintained by different teams, the arguments above are also valid for the involved partners. Using code conventions is not "l'art pour l'art". Whoever has ever tried to work on program code which was coded by somebody else knows from own experience that unfortunate naming of symbols, missing or mis-guiding comments, bad structured code, highly personal coding style etc. can cost a lot of time and provoke terrible errors. It is highly egoistic to spare a few minutes by writing sluggish, un-documented code, and let others later deal with the problem to maintain it. The set of rather simple rules compiled in our guide supports us all to save costs and time, and to ensure that we can maintain in future the code of the models.

Coding conventions in GAMS

Compared to other programming languages such as FORTRAN, PASCAL, C(++), Java or C#, GAMS does not break its code into functions and/or subroutines which clearly defined inputs and outputs. Equally, GAMS does not provide scoping for symbols: all GAMS symbols are known and accessible past the point where they had been declared; they have all global scope. Whereas coding conventions for most programming languages typically have a strong focus on modularisation of the code and clear scoping, we need to solve that issue for GAMS differently. Accordingly, naming conventions and clearly structured code are even more important in GAMS where every symbol has global scope!

Naming conventions

1) Use clear and easy to understand names for symbols and files.

A good name is self-explanatory, but short. Please keep in mind that the code basis of the developed models can be very large, a name such as "*p_emissionFactor*" is still rather general (but clearly better than *"p_factor"* and much better than "*p_f*"). In doubt, ask a colleague not familiar with the problem you are working on if she or he is able to understand the chosen symbol names.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

If a symbol name consists logically of several words, each new word except for the first one should start with upper case (we save space compared to using underscores). That so-called "**camelCase**" is a standard e.g. proposed in Java coding conventions:

```
PARAMETER p_data(rall,cols,rows,years) "Generic data cube";
PARAMETER p_popGrowthRate(rall) "Population growth rate";
```

An exception can be made if the tokens already comprise acronyms in upper case so that reading becomes cumbersome:

```
PARAMETER p_CAPMTRPolicy "Policy parameters for the MTR of the CAP"
```

In that case, it is better to use:

```
PARAMETER p_CAP_MTR_policy "Policy parameters for the MTR of the CAP"
```

Discouraged is the use of short symbols where the meaning is not clear in the context, such as:

```
PARAMETER i,p,q;
```

Please keep in mind that the very same name could be used by somebody else for a different symbol! If you introduce a new symbol, first use "search in files" from the GAMSIDE to make sure that the symbol name is not already in use.

Always add an explanatory long text to the declaration of symbols, if possible, stating physical units or other elements helping to provide a clear definition:

```
PARAMETER p_minFeedSharePerc(regions,animals,feed) "Minimum feed shares per
region, animal and feed stuff in % of dry matter intake"
```

Bad is:

```
PARAMETER p_minFeed;
```

As, (1) no domains are given, (2) the name is ambiguous (could be per animal, in a region …) and (3) an explanatory text is missing.

Note that vowels often can be dropped to shorten names, e.g. "*p_cnsQunt*" is almost as easy too read as *"p_consQuant"*. The use of "scientific" names such as *"p_alpha", "v_gamma*" etc. is discouraged for two obvious reasons. Firstly, their meaning is far from clearly defined and highly context depending. Secondly, there is a huge danger that the very same symbol name is introduced somewhere else in the code, leading to possible conflicts.

Tipp: "Find in files" from the GAMS IDE can be used to find all occurrences of a string over directories and files – easing dramatically the task to rename a symbol in a project.

2) Let equation names start with "e_"

There is tradition in other GAMS models to let equations end with an underscore which at least for old code can be kept.

3) Let parameter names start with "p_" and variables names with "v_".

That eases it dramatically to read equations in model equations as the GAMS notations is ambiguous in the sense that one cannot see what a parameter is and what a variable.

Parameters which are endogenous during calibration in equations should start with PV_, variables which are fixed during calibration should be start with VP_. Sets do not a have a prefix. The conventions should it make easier to understand what type of GAMS symbol is used.

4) Use clear and easy to understand codes for set elements, and always add an explanatory text to set elements.

5) Usage of sets

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Sets are a central element of the GAMS language. They structure logically the code by spanning the "problem dimensions", such as time, space, products or processes. Set names should be clear, but generally short as otherwise, statements become very long.

6) Use domain checking wherever possible.

Domain checking means that a symbol declaration in GAMS includes the information which sets are allowed on a specific dimension of a symbol, e.g.

```
p_maxFeedShare(RALL,PACT,A,FEED)    "Maximum shares for each feedingstuff, expressed in dry matter"
```

Domain checking might be cumbersome to implement and might require the use of SAMEAS, but it can avoid terrible errors which are otherwise very hard to detect.

7) Use sub-sets wherever possible.

Sub-sets are derived from other sets. They hence structure a domain clearly.

8) Do not declare the same collection of set members a second time.

GAMS offers the so-called alias for that, the so far mostly used notation in other GAMS models in alias statements is to add a 1, 2 …, e.g.

```
ALIAS (regions, regions1, regions2)
```

If you need the same collection in another set to allow for domain checking, use the possibility to introduce a complete set in a GAMS set declaration. It is proposed to use for sets which only used for that purpose the "SET_" notation is seen below, e.g.

```
SET  SET_FUELS     /gasoline,diesel/;
SET  fuelRows(Rows)      /set.SET_FUELS/;
SET  fuelCols(Cols)      /set.SET_FUELS/;
```

That notation can also to be used to avoid repeating collections of set elements in sub-sets, e.g.

```
SET  SET_FINFUELS  /gasoline,diesel/;
SET  SET_RAWFUELS  /natGas,crudeOil/;
SET  fuels         /set.SET_FINFUELS,set.SET_RAWFUELS/;
SET  finFuels(fuels)     /set.SET_FINFUELS/;
```

Coding style and structuring

9) Declare symbols used in one file only at the top of that file.

If the file is used in a loop or if statement, so that declaration in that file is not allowed, put the declarations into a separated file with "_decl" appended to the file name, and store it in the same sub-directory.

10) Separate processing code from data

Put the numerical data entering the code if possible in the relevant directory under "dat", and beyond a certain size, generate a GDX file from tables so that the GAMS code does not comprise an unnecessary high amount of code lines.

11) Generate files with a clearly defined purpose.

Each file should have clearly defined inputs and outputs, and especially the latter should form a logical unit. To give an example: a file which defines animal requirements should not as a kind of by-product correct herd sizes.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

12) Avoid unnecessary deep include structures (> 3).

Deep include structures require to open many files at the same time in the editor.

13) Statements

One declaration per line is recommended since it encourages commenting. In other words,

```
PARAMETER   p_level(domain1,domain2);
p_size(domain3);
```

is preferred over

```
PARAMETER   p_level(domain2,domain2), p_size(domain3);
```

Each line should contain at most one statement. Example:

```
iTry = iTry + 1;
iTry = iTry +1; RUNR(MS) = NO;
```

Avoid lines longer than 80 characters, since they're not handled well by many terminals and tools.

14) Indentation and program flow structures

When an expression will not fit on a single line, break it according to these general principles (from the Java coding conventions):

- Break after a comma.
- Break before an operator.
- Prefer higher-level breaks to lower-level breaks.
- Align the new line with the beginning of the expression at the same level on the previous line.
- If the above rules lead to confusing code or to code that's squished up against the right margin, just indent 6 spaces instead.
- Loop and other program structures should be clearly visible by 3 spaces indentation:

```
LOOP(RU,
   Statements in here must be indented to show the structure of the program
);
```

- $ operators are generally preferred over IF statements:

```
p_myParam(RU) $ (p_otherParam) = 10;
```

is preferred over:

```
IF ( p_otherParam,
     p_myParam(RU) = 10;
);
```

And certainly one should not use a loop as shown below – it is not only harder to read, but also slows down program execution:

```
LOOP(RU $ otherParam(RU),
     p_myParam(RU) = 10;
);
```

However, that is bad style to code as follows, as is not immediately visible that several assignments all depend on the same condition:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
p_myParam(RU) $ (p_otherParam) = 10;
p_myParam1(RU) $ (p_otherParam) = 20;
p_myParam(RU) $ (p_otherParam) = 30;
```

- Avoid unnecessary complex if and loop structures, or $-controls in statements.
- Remove duplicate code by moving it to an include file

15) $BATINCLUDE

"Batinclude" statements allow passing arguments to an included file. Inside the included file, the passed arguments are referred to with "%1, %2 etc." according to the order they are handed over. It is extremely cumbersome to read such a program as "%6" is simply meaningless. That problem can be circumvented with the following coding trick which works as a rename:

```
$setlocal regions %1
p_myParam(%regions%) = p_someOtherParem(%regions%);
```

16) $ONMULTI may be used only locally for well-motivated cases, followed by $OFFMULTI.

$ONMULTI allows for several declaration of the same symbol. That is really dangerous, as conflicting use of the same symbol might not be detected.

17) Use of $IF

$IF is a compile time command, i.e. it is defining what pieces of the code are executed.

- $IF should always be replaced by $IFI – the not case sensitive version.
- $IFI should only be used for single line statements:

```
$IFI %MODE%==Dummy $INCLUDE "dummy\someFile.gms".
```

- If several lines refer to the same $IFI statements, $IFHTENI … $ENDIF should be used. Accordingly, avoid constructions such as:

```
$IF %MODE%==Dummy   p_x(RS) = p_y(RS)
$IF %MODE%==Dummy   * p_o(RS)
$IF %MODE%==Dummy   * p_z(RS);
```

GAMS might treat the second line as a comment (it starts with a "*")! There, according to the rule above, use:

```
$IFTHENI %MODE%==Dummy
      p_x(RS) = p_y(RS)
      * p_o(RS)
      * p_z(RS);
$ENDIF
```

18) Find a compromise between the number of files included and their length.

Files should whenever possible not be longer than 1000 lines, but also should consists of more than 10 statements or so. A top level module should reveal its structure in the GAMS code.

19) Error trapping

Error trapping means that the code itself comprises tests which throw an error, instead of doing bad calculation due to missing or erroneous data or provoking run time errors. Imagine e.g. a program which works on market balances. Besides stock changes, all elements of the market balance are defined to be non-negative. Continuing with the code while trapping with $ and "if" statements negative market balance elements is probably the wrong tactic, as the results will anyway not make sense. It is hence good to test first if such logically nonsense data are present and then to stop execution and warn the program user about such errors.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

20) If an include file has a well-defined data manipulation task, try hard to include a test at the top of the file which raises an exception if necessary data is missing or does not satisfy some lowest standard.

Use %system%.fn and %system.incline% so that errors trapped provide information where the problem happens. Example:

```
ABORT $ exceptionFilenameRegions "Error in %system.fn%, line %system.incline%: Population data missing for the following regions:", problemRegions;
```

21) Comments

GAMS code is computer code – it is not preliminary designed to provide easy to read text to humans. Indeed, it is often necessary to write of e.g. equations differently as they are documented in a paper to allow for an efficient use of GAMS. The meaning of the GAMS code is therefore often not immediately evident. Mis-interpretation of the code however can provoke bad errors – somebody might change a statement as she or he has not clearly understood what the purpose is.

Comments, on the other hand, are directed towards our colleagues who want to understand the code – often, because there is the need to change or debug it. Comments should especially explain those things which are not easy to deduct from the code itself – they should not repeat the obvious, but motivate why a certain task is coded in a specific way. Comments also help us to quickly locate a statement or block of statements related to a specific task. Generally, comments are at least as important as the GAMS code itself.

22) Introduce yourself!

Those who contribute a bit of code should label it with their name. We use pre-defined file headers (see next) where the name of the author(s) is one of the fields.

23) Generate a file header explaining the purpose of the file.

Use the predefined template for doing so, so that the HTML based documentation can collect that information automatically. The following standard pieces of information should be included:

- Name of the author
- Name of the file
- Purpose of the file
- In case of a file used with “$batinclude”: descriptions of the arguments

24) Add clear and easy to understand comments to any not self-explaining GAMS code.

Try hard to write self-explaining code, but assume that it is not possible – hence add comments! Motivate and explain statements and code structure, instead of repeating what the code does again in plain English. Good code is like a good paper: it is structured such that the reader can easily follow the flow; comments support that. A typical example of a completely useless comment which does not add information is shown below:

```
* Set P_myParam to P_otherParam
p_myParam(Domain) = p_otherParam(Domain);
```

Save others the time to deal which such useless comments.

Include *references* wherever possible to comments, e.g. to the methodological documentation or project deliverables. If the GAMS code is developed from a reference (e.g. the IPCC guidelines to structure GHG emissions), note the full reference and the page (see also the section on meta data), so that the code can be verified quickly.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

Comments are introduced in a separate line *above* the code to comment. The preferred standard style of a comment referring to a statement is shown in the following. The same indentation as the code commented upon should be used (i.e. if the code start in column 10, the "---" starts also in column 10):

```
* --- Here comes the comment
```

Block comments should be used to structure a file logically into different sections:

```
*-----------------------------------------------------------------
*     Here comes the description of the block
*-----------------------------------------------------------------
```

It is good style to insert a comment above an include statements which briefly explains the purpose of the included file.

SVN and testing

The software versioning system SVN allows us to work efficiently as a distributed team of developers, especially to synchronize easily to the common established code base and to document changes to the code from version to version. Information on TortoiseSVN, the plug-in for Windows, can be found at http://tortoisesvn.tigris.org/.

25) Only commit fully functioning and tested code to SVN.

Any exemptions must be made public beforehand and are subject to agreement of all others involved. That holds especially for the trunk. Any major changes, especially those leading to different results, should also be announced to all other programmers involved.

Accompany your commit with a clear description what was changed and why. If a whole block of files is subject to your change, commit them if possible together. Avoid committing whole bundles of unrelated changes with one commit.

If you introduce complex new features or refactor substantially existing code, provide a separate short technical note. Such a short note should comprise (1) a short *motivation* including references to project deliverables etc., (2) which *files* had been added (or changed), (3) a clear description of inputs and outputs, and (4) any unusual technical solution.

26) Update before committing!

Make sure that you have updated the files you plan to commit, and do so before any tests, to make sure that you are testing the latest available version.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

APPENDIX 2: QUALITY CRITERIA FOR MODELS AND DATASETS ACCORDING TO THE WAGENINGEN MODELLING GROUP.

This section gives an overview on the quality criteria applied by the Wageningen Modelling Group, Categorized at 2 quality levels in 22 Requirements within 7 themes across 3 perspectives on quality.

For more information see https://intranet.wur.nl/Project/WRModellingToolbox, or contact Geerten.Hengeveld@wur.nl

Perspective: Science & Technology (ST)

ST.1 The model/dataset is described

1 THERE IS A GENERAL DESCRIPTION OF THE MODEL/DATASET

A *purpose * area of application * theoretical framework * paradigms*
The general description includes statements on the purpose/goal/aim in developing the model/dataset. It provides a delineation of the area of application both spatial (generic (e.g., field, country) or specific (e.g., Netherlands, Wageningen) and conceptual (e.g., under constant climate). With the theoretical framework the world as addressed within the model/dataset is framed according to some basic paradigms and core assumptions.

2 THE CONCEPTUAL AND FORMAL MODEL ARE DOCUMENTED

A *explicitly documented * assumptions * simplifications * embedded in literature*
The theoretical framework is worked out into a conceptual model (most relevant components and their relationships/dependencies) and subsequently into a formal model (mathematical model, decision tree, database structure etc.). The assumptions and simplifications made at each step are presented along with their justification (e.g., a table with assumptions). The formal model is explicitly documented (e.g., a full set of formulae, decisions, measurements). This is embedded in scientific literature as illustrated by targeted references.

AA *motivated complexity * peer-reviewed scientific publication*
From the idea that the model should be as simple as possible, but not more simple than that, a reflection is included on the relationship between the complexity of the conceptual and formal model (e.g., number of variables included, statistical design, number of equations, precision reached) and the foreseen use (e.g., application, data availability, evaluation options, past experience). The model/dataset has been published in a peer reviewer scientific journal.

ST.2 The technical implementation of the model/dataset is documented

1 THE IMPLEMENTATION IS DOCUMENTED

A *Basic structure * flow diagram*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

The documentation of the implementation should support the tech-savvy reader in the interpretation of the computer code, scripts etc.. A flow diagram highlighting the main modules of the program, or a database scheme is provided. It is considered good practice to explicitly provide a link between the elements in the diagram and the text.

AA *Code commenting * motivated (modular) design * code review*
The structure of the program is motivated, consistent, and modular where relevant. Code is commented. Discussing choices for algorithms and model structure facilitates interpretation of the logic of the program or database. A code review (external to the development team) is performed to get feedback on best-practices, consistencies and potential (minor) programming errors. This review is to be organised by the development team. The feedback from the code-review is used in refining the development plan.

2 THE TECHNICAL ENVIRONMENT IS DOCUMENTED

A *Language * IDE * settings * limitations*
The development environment is documented, both in general and specific terms. This includes the computer language, IDE, versions used and critical settings. Technical limitations due the implementation (e.g., numeric precision, memory or multi-processor use) are mentioned.

3 THE MODEL/DATASET IS TESTED

A *Tests documented * protocol * untested components named*
There is a clear list/table (protocol) of tests that can / are relevant / need to be performed on the model/dataset to ensure correct technical implementation of the model/dataset (e.g. unit tests, order of magnitude tests, checksums, algebraic or numerical recalculation of simplified or extreme cases). In the documentation the performance on these tests is noted. Deviations from expected test results are discussed. If components of the model/dataset are not tested (yet), these components are identified and a motivation is provided.

AA *Tests on schedule * periodic evaluation on completeness*
During model/dataset implementation tests are performed on a regular basis to ensure correct implementation. Periodically the protocol is evaluated for missing components of the model/dataset.

ST.3 The parameters, variables, inputs to and output of the model/dataset are described

1 THE PARAMETERS AND VARIABLES OF THE MODEL/DATASET ARE DOCUMENTED

A *Quantities * units * default values * default source * description*
A full list of parameters and variables is provided. For each of these the quantities and units of measurement are given and a description of the interpretation is provided. When available, default values are given with their source.

AA *Range * Precision*
The documentation of the parameters and variables is extended with the range of possible, or likely, values, uncertainty range for default values and the associated precision is provided (i.e., number of decimals). N.B. this does not refer to the floating point limit of the implementation.

2 CALIBRATION OF PARAMETERS IS DESCRIBED

A *Procedure * results discussed*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Calibration is defined as the deviation of parameter values based on (partial) model output in order to reach a pre-defined or desired output value. Depending on the model, calibration can constitute formal (mathematical) calibration, expert judgement or 'tuning' of parameters based on literature review. Calibration is generally applied to estimate values for parameters for which no default values can be derived based on first principles or experiments. The (preferred or applied) procedure for calibration is described. Impact of calibration is discussed.

3 THE INPUT AND OUTPUT IS DESCRIBED

A *Structure * format * quantities * units * precision * description * link variables & parameters * version echo*

The structure and format of the input and output are described, including quantities, units used and the precision of values. From the description the link between input/output and the variables and parameters in the model is clear (linked to formal model description and implementation). The output should include a reference (echo) to the version number of the model/dataset that generated this specific output.

AA *(Inter)national standards * input echo & timestamp*

The choice for input/output in a specific format, or using specific units or projections has implications for re-use and interoperability. Discuss which international standards are used and why (not). The output should include a reference (echo) to the input (query) and parameters settings used and time of execution, as to enable tracking the source of the specific output. Echoing is also important for extractions of data from (dynamic) databases.

4 THE ORIGIN OF INPUT DATA IS DESCRIBED

A *Data preparation pipeline * source * scripts tested*

Preparation of data into the format required for operation in the model, or inclusion into the dataset is described. This can include a clearly delineated data preparation pipeline from source to input. Scripts used should adhere to the same principles of technical documentation and testing described.

AA *Protocol for acquisition * periodically updated*

A protocol for acquisition of (raw) data is mostly appropriate for (dynamic) datasets or periodically performed standard calculations. Such a protocol includes source of data, measurement protocol, and contact persons for institutional sources. This protocol is periodically updated.

ST.4 The functioning of the model/dataset is evaluated

1 A SENSITIVITY ANALYSIS IS PERFORMED

A *Tailored to model/dataset type * documented * discussed*

The sensitivity of the model to variation in parameter values and initial conditions is analysed. For different types of models, different methods are appropriate (e.g., one at a time, monte carlo simulation, analytical sensitivity analysis). Sensitivity of different components of the output to the same input can be different. The sensitivity analyses performed are documented, motivated and the results are discussed, e.g., with respect to model performance, precision or accuracy of measurements and input data.

AA *Protocol * evaluated*

The sensitivity analysis is performed according to a protocol that prompts repeated sensitivity analysis for subsequent (sub)versions of the model. N.B. changes in one part of the model can impact the sensitivity to other parts of the model. Periodically this protocol is evaluated for completeness.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

2 AN UNCERTAINTY ANALYSIS IS PERFORMED

A *Qualitative discussion*
The uncertainties underlying the assumptions, structure and data sources are analysed in a qualitative way, naming both known and unknown uncertainties. The documentation includes a brief description of the methods used. The results of the analysis are interpreted and discussed, e.g. with respect to model performance and reliability of the output/dataset. For datasets, this includes an analysis of the measurement error and sample design.

AA *Quantitative analysis * evaluated*
With the quantitative uncertainty analysis, the impact of known and quantifiable uncertainties in the input on the output is analysed. Various methods exist for different types of models, the choice for the method used should be motivated. The extent of the uncertainty analysis, both with respect to the quantification of uncertainties in the input, as with respect to the model components and outputs considered is periodically evaluated.

3 THE MODEL/DATASET IS VALIDATED

A *Discussed * non-validated components named*
Through validation one judges the validity of model output or dataset content based on external information. This information can come from various sources (e.g., measurements, literature or expert judgement). The method of validation used is documented. Components that are not validated (yet) are named. Validation status of the model/dataset is discussed, e.g., with respect to the interpretation and reliability of the model output or dataset content.

AA *Protocol * evaluated*
As with the sensitivity and uncertainty analyses, validation should be performed according to a protocol for each new version of the model/dataset. The extent of the validation, with respect to model components and outputs is periodically evaluated, e.g., considering new data, new model components etc..

4 THE USE OF THE MODEL/DATASET IS MONITORED

A *Example studies listed*
Monitoring of the use of the model implies that one is aware of the use of the model inside and outside of the development team, this can range from tracking citations to evaluation of the application of the model/dataset. From this monitoring, example studies are drawn and referred to.

AA *Use & use(r) experience tracked * evaluated * documented*
Options for the evaluation of the use and use(r) experience are model/dataset specific. Core is that the feedback from users (active or passive) is reflected upon, documented and feeds into the development plan.

5 THERE IS A GENERAL ASSESSMENT OF MODEL/DATASET QUALITY

A *Relate goal to: test * sensitivity * uncertainty * validation * use*
The performance of the model/dataset (as reflected in the five evaluations: tests, sensitivity, uncertainty, validation and use) is related to the specifications (goal, intended area of application) to reflect 'fitness for purpose'. The fitness for purpose is documented.

AA *Include reliability * precision * data used * external review*
The general assessment of the quality is extended to include the quality (reliability, accuracy, precision) of the data used. During an external scientific review a number of scientific peers is asked to provide a fresh view on the fitness for purpose and relate assumptions to the relevant scientific

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

state of the art. The aim of this review is to provide input for future developments (constructive) rather than to provide a binary judgement.

Development & Organisation (DO)

DO.5 The development of the model/dataset is planned

1 THERE IS A DEVELOPMENT PLAN

A *List of plans * progress reported * based on evaluation*
There is a point-wise list of planned developments. The progress of these planned developments is periodically reported. The evaluations of model performance (partly) motivate these plans.

AA *Further evaluation * periodically updated*
A timeline for the planned developments is provided. To assure continuous development, further -extended- model evaluations are planned. The development plan, that includes a motivation for the planned developments, is periodically updated. The time horizon will be model/dataset specific, but should be mentioned.

2 A VERSION CONTROL SYSTEM IS IN PLACE

A *Documented * acceptance criteria * (WUR) central archiving*
The method for keeping track of versions is documented, including what is, and what is not under version control. Version control should encompass both the development versioning (subversions during implementation) and production versioning (versions released for use). Version acceptance criteria are documented explicitly. Differences between versions are reported and discussed. The version control makes use of (WUR) central archiving.

AA *Protocol for documenting & code-commenting*
The protocol for version control includes a protocol for documentation and code-commenting (e.g., who produces text for what type of documentation, during development, and how and at which level of detail code is commented).

DO.6 The organisation around the model/dataset is planned

1 THE METADATA OF THE MODEL/DATASET IS AVAILABLE

A *Domain appropriate format*
The metadata is provided in a domain appropriate format. The metadata reported should at least include items included in the WR modellibrary metadata format, that is available at https://intranet.wur.nl/Project/WRModellingToolbox.

AA *FAIR*
Up to date metadata is publicly provided and according to FAIR principles. FAIR principles state that (meta)data should be Finable, Accessible, Interoperable and Reusable.

2 THERE IS A MANAGEMENT PLAN

A *Responsiblities: content * technical * next-in-line * ownership * financial cover*
The management plan minimally lists the distribution of core responsibilities; content, technical development & maintenance, next-in-line responsibility and ownership. (Un)availability of funds to cover planned developments should be mentioned explicitly.

AA *Vision on future * periodically updated*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

The management plan is further extended with a vision on the future use and development of the model/dataset, encompasses current and potential use, anticipated internal and external developments (e.g., personnel, technical) and how these could provide opportunities or pose threats to future operation. The management plan is periodically updated.

3 DEPENDENCIES ARE DISCUSSED

A *Datasources * (third-party) use*
A list is provided with the input and output dependencies of the model/dataset (e.g., updates in source data, monitoring networks, (third-party) use of model output). With dependencies we aim at (dynamic) data sources on which future model/database use is dependent, and at (third party) users that rely on (future) model output or dataset versions.

AA *Tracked * Obligations * liabilities*
Because these dependencies could pose threats for continuation or might provide opportunities for shared future development, the continued availability or planned demands from these dependencies should be tracked. Risks of losing input sources (by lack of alternatives) and explicit obligations for future operation should be highlighted.

4 EXTERNAL USE IS FORMALISED

A *Conditions for use * User support*
Conditions for use outside the development group are defined. The responsibility for user support is named.

AA *User agreement * legally checked * financial paragraph*
A user agreement is available that is legally checked and in line with the ownership of the model/dataset. This user agreement contains a financial paragraph, even if no fees are charged.

Interpretation & Use (IU)

IU.7 User documentation is provided

1 INTERPRETATION GUIDANCE IS PROVIDED

A *Goal * area of application * theoretic framework * summary of evaluations * general public*
Interpretation of model output or dataset contents is in general not trivial. The interpretation implies understanding of the theoretic framework, conceptualisations and formalisations of the model - i.e., the assumptions and simplifications - and of the outcome of model evaluations. Guidance should be supplied on what these mean for the value of the outcome and on when (not) to use the model/dataset. This interpretation guidance should be readable for more general public than the scientific community.

AA *Reflection on goal, area of application, structure, complexity*
The interpretation guidance is extended with a reflection on the tension between goal, area of application and complexity of the model, limitations in the implementation, data quality and availability and the realised model performance.

2 THERE IS A USER MANUAL

A *Operation instructions * installation guide * summary of technical documentation * minimal system requirement * format of input & output * contact information*
The user manual includes clear operation instructions, an installation guide, a summary of the technical documentation, listing the minimum system requirements, and clear documentation of the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

format, structure and content of the user-relevant input and output files. Contact information for user support is provided.

APPENDIX 3: TESTING STRATEGY AS A KEY ASPECT IN QUALITY MANAGEMENT OF AGRI-ECONOMIC MODELS

This appendix presents a basic testing strategy of agri-economic models which aims to improve their quality management and to facilitate distribution of work over partners. We start by introducing a layered testing strategy and their basic technical and organizational requirements. This is followed by a testing strategy example for the IDM model FarmDyn (Britz et al. 2016) technically realized in connection with GAMS Graphical Interface Generator (GGIG) (Britz 2014). The testing strategy is layered on three tiers requiring increasing labour input, from compile time tests, to execution run tests, and finally to outcome tests. The technical implementation is available to all models within the MIND STEP model toolbox using GGIG such as IFM-CAP and CAPRI. In the case where the GGIG is not used, the conceptual framework of the three-tier layered testing strategy can be adapted and applied.

1.1 Testing strategy in agri-economic models – Layered testing and technical basics

Modularity is key for agri-economic models with increasing complexity to safe on the one hand computational resources and to provide on the other hand flexibility in the assessed entities depending on the research question at hand. Furthermore, a modular model setup also gives the option to independently work on one model with multiple developers without too much interference, allowing developers to focus on specific modules, as modules can be switched on and off. Despite its advantages, modularity and simultaneous working on the code comes also at a cost, especially when no agreed testing strategy is implemented, and multiple modules are closely connected.

One can imagine the situation where a developer introduces new code to a module while multiple other modules are switched off. Assuming the developer does not encounter any problems in his model setup, he is confident to upload the new model code. There are now four possible outcomes for other developers downloading the new code and accepting the changes to their versions. First, everything runs smoothly in their model setup and the outcome is accurate. Second, they are faced with compilation errors, i.e. syntactical errors. Third, the model returns execution errors which occur due to e.g. mathematical infeasibilities. Fourth, there are no apparent programming related errors; however, there are erroneous model outcomes not spotted by the developers. Where compilation and execution errors are a nuisance for every updating developer, the last type of error is more critical as it can propagate in subsequent model versions and potentially affect future projects.

Hence, a clearly defined testing strategy is a key aspect of quality management in agri-economic models. In this section we present a three-layered testing approach which targets the three types of aforementioned errors: compile time errors, run time errors, and outcome errors. All three testing blocks require increasing efforts in: definition of test instances, with regard to computation time, and in controlling their outcomes. A short summary of the most important aspects is provided by table X.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

1.1.1 TIER 1 – COMPILE TIME TESTS

Compile time tests check the syntactical correctness of the software code, but do not produce any numerical output which needs to be assessed. They require thus very limited computation time and little control efforts, and therefore can be used to cover a large number of different input settings in the test design. Test instances should reflect the different possible combination of modules, and option to configure these modules. Based on their short run time, compile time tests should be executed before each commit of new code by the developer, and additionally automated/by a designated person in regular time steps. Testing before committing ensures that the developer does not immediately impact the work of other developers. The additional regular compile time tests provide a back-up security system for the case that a developer skipped the compilation time tests.

1.1.2 TIER 2 – RUN TIME TESTS

Run time tests comprise tests of key combinations of modules which are most frequently used in the model applications. These tests aim to prevent that changes in model structure and default parameterization provoke infeasibilities or execution errors. As run time tests can take longer, there only a limited number of test instances, however, before each commit developers should do the run time tests to ensure that there are no infeasibilities in other module combinations. Similarly to the compile time tests, a regular (automated) check of run time tests should be executed.

1.1.3 TIER 3 –OUTCOME TESTS

Outcome tests focus on checking the plausibility of model outcomes on a selected number of test instances. Due to the required running time and limited number of potential model experts to check outcomes, the test instances should be selected carefully and only for the most important combinations of modules. The aim of these numerical tests should be to prevent that code changes in the same or in parallel running projects lead to unforeseen outcomes in key results for on-going applications, and hence would propagate in future research projects as well. Outcome tests might require considerable time of domain experts to determine if code changes led to unwanted changes in key indicators. The frequency of numerical tests should be oriented towards the commit activity of all developers given by established statistics of developing activities in the model. Given the low frequency of outcome tests, a data file which automatically collects the model results of each committed version should be implemented to facilitate the identification of versions which led to outcome changes. The responsibility of numerical tests should be shared among the domain experts to limit the time burden for each researcher.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Table 3: Frequency, responsibility, and time investment for testing in agri-economic models

	# of instances	Responsible person	Frequency	Time investment
Compile time tests	Very High	Developer	Each commit	Low
		Model expert	Weekly	Low
Run time tests	Medium	Developer	Each commit	Low
		Model expert	Weekly	Low
Outcome tests	Low	Domain expert	Dependent on developing activities	High

1.1.4 TECHNICAL REQUIREMENTS AND ORGANIZATION OF THE TESTING STRATEGY

Implementing a testing strategy for Mind Step models as described above requires tools such as a software version system. Technical solutions for software versioning systems are tortoise (SVN) or GitHub (Git), which can be either hosted locally or through available cloud solutions. Project partners must be trained in the use of these tools, and the project management must ensure that they are used. Similar to knowledge of statistical packages or AMLs, training courses on such tools can also be centralized at department level or above. Especially for young researchers, certified participation in training courses can foster their career. Considerably resources are necessary to introduce a testing strategy for a first time, including cost factors such as license fees, dedicated hardware for a model repository, but also human resources for setting up the technical implementation and the training of staff.

1.2 The IDM model FarmDyn

FarmDyn (Britz et al. 2016) is a farm-scale bio-economic model realized in the AML GAMS (GAMS Development Corporation 2019) and hosted on an SVN based versioning system. It depicts in detail farm management options, such as organic and mineral fertilizer application on a monthly basis. Typical model configurations in comparative-static deterministic mode comprise between 1.000 to 10.000 variables and constraints. Several dozen of binary or integer variables depict indivisibilities in investments and labour use, and if-conditions related to command-and-control and opt-in policies. The constraints comprise equalities and inequalities. The objective function in a deterministic comparative-static set-up will maximize different elements of the farm-household income, depending on the model's configuration encompassing besides profit withdrawals from the agricultural enterprise also off-farm wages or income of renting out or selling land.

In line with the idea of modularity in the MIND STEP model toolbox, FarmDyn is realized to some degree in a modular fashion. Blocks of equations relating, for instance, to certain farm branches, can be integrated in the current model set-up or not. The layout also opens up the possibility to define case study specific block of equations, to depict, e.g., specific regional command-and-control or opt-in policy measures. Technically, this type of modularity is mostly realized based on conditional includes.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

FarmDyn belongs to the long-standing and accomplished models in the MIND STEP project and produced several publications within multiple projects (e.g. Lengers et al. 2014, Schäfer et al. 2017, Kuhn et al. 2019, Seidel and Britz 2020, Heinrichs et al. 2021, Britz 2020).

FarmDyn features a Graphical User Interface (GUI) realized in GGIG, a package realized in Java which combines an interface generator for GAMS or R based projects with a report generator (Britz et al. 2014). Besides FarmDyn, GGIG is used for a range of other economic models within the Mind Step project including the IDM model IFMCAP at the EU's Joint Research Center (Louichi et al. 2018) and the partial equilibrium model CAPRI (Britz and Witzke 2014).

1.3 Technical implementation of the testing strategy with GGIG

Like other GUI generators, a control definition file interpreted by GGIG generates user-operable controls on the interface, typically along with admissible inputs. This eases mainly model steering, as users select directly from the available input choices in the Look & Feel they are used from other software packages. GGIG supports two ways to perform model runs. In interactive mode, a user selects the relevant choice via the GUI (see figure 15). This relates for FarmDyn, for instance, to the chosen farm branches, if the model is run in comparative-static or dynamic mode, in a deterministic or stochastic setting, to farm endowments such as land and labour, available crop and technologies etc., but also technical choices such as which solver to use. After all choices are made, the model can be started from the interface. Afterwards, results can be explored either via the GAMS listing or based on the reports provided by the GGIG report generator, potentially comparing different model runs. This interactive mode of running the model is not well suited for systematic testing. One would need to give a tester a document with the predefined input sets for each test. They would need to be entered manually control-by-control, afterwards a tester would then wait until a run is ready and next check the results. This is an error prone and tiring process.

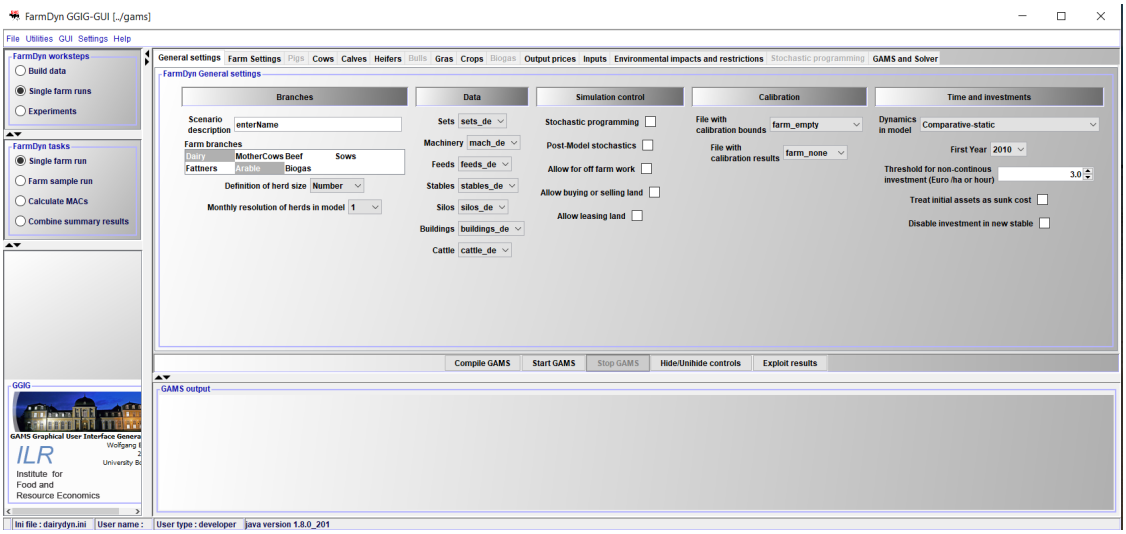


Figure 17: FarmDyn's graphical user interface

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

How to: The batch mode can be selected from the GUI menu, item “Batch execution”. In the new window, the batch file to run can be selected. The batch file can be set-up with information from include files generated by GGIG.

Therefore, the so-called batch mode of GGIG as the second way to perform model runs is the preferred option for testing. In this mode, multiple input sets defined in a text file are started automatically after each other. GGIG stores for each run the listing file, and reports the used settings and the return code from GAMS on a HTML page (see Figure 16). The batch mode can either be started via a dialog from the GUI or be deployed from a command prompt or other software, which open the doors to start tests in some automated fashion. Input files for GGIG can be easily set-up by copy-and-paste from include files, generated when a GAMS run is started by the GUI in interactive mode.

GGIG batch execution report

Runs are generated from	**C:\cgebox\gui\testCompile.txt**
at	**Sa, 16 Jan 21 11:26:56**

Workstep	*Task*	*User*	*Directories: work result dat restart*	*Settings*	*Gams options*	*Started Ended Time used*	*RC*	*Listing GdxDiff*
Simulation	Simulation		C:\cgebox\gams ../results ../data -	**All settings in use**(click to show/hide) **Non default settings in use**(click to show/hide)	compile --scen=com_inc --ggig=on -threads=8 -procDirPath=c:\scrdir optdir=opt	Sa, 16 Jan 21 11:26:57 Sa, 16 Jan 21 11:26:58 0 min and 0 secs	0	c:\scrdir\16.01.2021_11.26.56 \1.lst
Simulation	Simulation		C:\cgebox\gams ../results ../data -	Use pre-defined configurations=true **All settings in use**(click to show/hide) **Non default settings in use**(click to show/hide)	compile --scen=com_inc --ggig=on -threads=8 -procDirPath=c:\scrdir optdir=opt	Sa, 16 Jan 21 11:26:58 Sa, 16 Jan 21 11:26:59 0 min and 0 secs	0	c:\scrdir\16.01.2021_11.26.56 \2.lst
Simulation	Simulation		C:\cgebox\gams ../results ../data -	Dynamics=Recursive dynamic **All settings in use**(click to show/hide) **Non default settings in use**(click to show/hide)	compile --scen=com_inc --ggig=on -threads=8 -procDirPath=c:\scrdir optdir=opt	Sa, 16 Jan 21 11:26:59 Sa, 16 Jan 21 11:26:59 0 min and 0 secs	0	c:\scrdir\16.01.2021_11.26.56 \3.lst

Figure 18: HTML page with results from test run generator by GGIG

As an additional option, the batch mode of GGIG allows to compare previous against current results. This requires that both results sets are stored as parameters in so-called GDX containers. GDX is a proprietary, binary format of GAMS for which application programming interfaces in different programming languages are available. The two GDX containers with the old and new results are compared by the GDXDiff utility from GAMS, called from within GGIG. It produces as outcome a parameter in a third GDX container which reports the differences between the two result sets, subject to a user chosen threshold. GGIG then read this GDX container and formats its content as HTML code added to the report, as shown in Figure 18.

How to: add a line such as the following before the execute=gamsrun in the batch file:

Gdxdiff =

This automated calculation of differences is normally applied to a smaller vector of key model outcomes. The HTML page reports the number of records where differences larger than a predefined threshold are found, in the (artificial) example shown in Figure 3 below these are 23 cases at the chosen threshold of 1%. The user can then with a mouse click inspect the individual records for changes, as shown in the lower panel. If changes are considered acceptable, the test is considered successful. Otherwise, the information which indicators changed by how much might already provide useful information to find and correct errors.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

Single farm runs	Single farm run		C:\dairydyn\gams ../results ../dat -	**Updated settings**(click to show/hide) **All settings in use**(click to show/hide) **Non default settings in use**(click to show/hide)	run --scen=incgen/runInc --ggig=on procDirPath=c:\scrdir -threads=8 optdir=opt	So, 17 Jan 21 10:18:16 So, 17 Jan 21 10:18:23 0 min and 6 secs	0	c:\scrdir\17.01.2021_10.18.06\2.lst c:\scrdir\17.01.2021_10.18.06\diff_2.gdx p_sumRes # of diffs > 1.0% : 23(click to show/hide) WinterWheat,dif1,40,60 WinterWheat,dif2,36,46 WinterRape,new,16,40 profit,dif1,29278,44 profit,dif2,208230,15 NSurplus,dif1,-57,59 NSurplus,dif2,10,17 PSurplus,dif1,-13,68 PSurplus,dif2,0,32 totalLabour,dif1,1081,50 totalLabour,dif2,7200,00 labourUse,dif1,0,15 labourUse,dif2,1,00 margArab,dif1,1123,84 margArab,dif2,962,46 MaizSil,old,20,54 gras2_12_graz100,old,4,13 gras4_14_4cuts_sil100,old,35,87 cows,old,138,51 gras,old,40,00 NorgAppl,old,162,49 labourBoundMonthly,old,3,00 margGras,old,740,29

Figure 19: Automated reporting of differences in GGIG

The input choices offered to the user, defined in the interface definition file from which GGIG generates the GUI, also span up the potential test range of inputs for the economic model. This opens up the possibility for automated tests. The batch mode of GGIG can generate automatically test instances from the available input choice for certain types of controls. These tests build on the default setting for all controls on the interface. In a loop, for each single selection control such as a checkbox, all potential settings are subject to a test, documented on the HTML page. Afterwards, the control is reset to its default and the settings for the next control are tested. Controls which define numerical input (such as sliders or spinners) are normally not subject to such tests, as introducing a different number in the code is unlikely to provoke errors in compile time tests. It is however possible to define, besides the default registered with the control, a second numerical value for a test. This is useful if the code treats, for instance, a zero different from a non-zero value. Equally, controls can be excluded from testing, for two reasons. First, the GAMS code itself comprises some tests; they throw an error at compile time for certain combinations of input settings which are considered illegal. Including these cases would show failed tests, which is actually not true as the user cannot execute the model with these settings. Second, the GUI control definitions comprise so-called dependencies. Choosing between risk behavioural models, to give an example, is only possible for the user if the stochastic version of the model is used. With the deterministic version being the default, tests of the risk behavioural models are not possible and therefore excluded from testing. Such cases require defining tests manually.

1.3.1 COMPILE TIME TESTS – FARMDYN

Solely compiling the model code requires little time, typically less than a second, which opens up the possibility to perform many compile time tests. The HTML pages generated by GGIG flag runs with errors in red, which allows to quickly find failed test instances. The compile time tests are mainly automatically performed by GGIG itself as detailed above. A major advantage of the automatically set-up tests by GGIG is that any change in the GUI definitions – which is equivalent to changing the model's potential input data set – is reflected without the need of manually updating test instances. For Farmdyn, these fully automated tests comprise currently about 150 different input sets. They refer partly to different model configurations, such as comparative-static versus different types of dynamic runs, but also comprise technical options which should not affect outcomes, such as switching listings on and off.

How to: To run compile test select the "batch_test_compilation" in the batch mode within the GGIG. After running the batch file, results can be seen in the HTML output file as seen in figure 2. Runs with compile time errors are marked red and the responsibility for corrections lies in the person running the tests before a commit.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

To these automated tests a few manually defined test instances are added. They complement cases which are not captured by testing each control individually with all others at their default values.

1.3.2 RUNE TIME TESTS - FARMDYN

How to: To make run time test select the "batch_test_execution" in the batch mode within the GGIG. The execution file contains a list of predefined farms to check. After running the batch file, results can be seen in the HTML output file as seen in figure 2. The HTML output file shows if run time errors occured, such as infeasibilities, which than requires action by the testing person.

The intermediate layer comprises run time tests where key farm management choices such as herd sizes and crop shares are fixed to a known optimal solution for the input data generated, based on previous model code. These tests aim at excluding that changes in model structure and default parameterization provoke infeasibilities. Fixing key management choices will also reduce solving time, as the set of fitting integer solutions to given herd sizes and crop acreages is limited. The GAMS code will throw an error if the model is integer or otherwise infeasible, such that these failed test instances can be easily detected in the HTML report page. Another advantage of these tests is that key indicators such as profits or Green House Gas emissions are unlikely to change much when the core farm program is fixed. Larger changes in such indicators are hence a rather sure indication of some flaw in recent changes in the code, as long as the former results were deemed correct.

1.3.3 OUTCOME TESTS - FARMDYN

The final tests focus on checking the plausibility of model outcome on a selected number of test instances. This is an expensive test strategy. Optimizing a test instance can take up to multiple minutes due to the integer variables. More important, deciding if a test has failed, should no run-time error occur, requires a plausibility assessment of simulation results by a FarmDyn researcher. This clearly restricts the number of test instances which are regularly run, and requires their careful selection. For the current test strategy, these test cases mostly comprise typical case studies from on-going projects. The main aim is hence to exclude that code changes in the same or in parallel running projects lead to unforeseen outcomes in key results for on-going applications.

How to: The outcome result tests use the same batch file as the execution test. After running the tests, the HTML output shows potential differences for all chosen indicators against the last revision. Here, a chose, rotating representative checks on a regular basis if significant changes occur and, if this happens, contacts the person who committed changes.

The plausibility assessment focusses on key indicators such as profits, herd sizes, crop acreages and some selected environmental indicators. They can be directly retrieved from the HTML pages generated by GGIG. Additionally, the results are collected automatically in an EXCEL workbook, with one sheet for each test instance. The different indicators are in the rows, while the columns refer to a revision number tested. This depicts indicators changes along the history of code change. To ease the assessment, a colouring scheme is applied to the relative changes which visualize the size of the change.

Outcome tests implicitly assume determinism, i.e. that the same input executed with the same script produces each time identical results, for instance by different teams which use the same economic model. This is however not necessarily the case.

First, the software code executing the script might differ. There are, for instance, frequent updates to AMLs. GAMS.com has released more than fifty versions over the last two decades, which typically also

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

include updates of solvers available with GAMS[1]. Improvement in the solver algorithms, such as related to scaling, might provoke (slightly) different results on the same problem across releases. Depending on their currently version in use, users might therefore face different results for same input data and script.

Second, coders might themselves provoke differences in outcome even when using the very same release of a software product; e.g. by defining maximal solving times for a model instance. With such limits active, depending on the hardware used and other processes running in parallel on it, a model might be solved to full optimality or end up as intermediate infeasible or optimal. Such cases might not be easily detected if they don't relate to the core model solve. It is not uncommon, to give an example, to use some constrained optimization problem to find minimal deviations to raw data in data balancing problems, or when calibrating behavioural or technical parameters during benchmarking. Such problems might even run in parallel, for instance, for different sectors or regions. Coders might in such cases have decided to accept e.g. intermediate optimal solutions to keep overall running time at an acceptable level and might turn off solution reports of such pre-steps to the listing. A user might therefore even not notice that such problems have not been solved to full optimality which might lead to unintended changes in input data entering its final simulation model. The impact of this can be clearly reduced by running such intermediate solves only once for a range of scenarios, to ensure that differences reflect changes in the shock, only, and not from using (slightly) different data or parameters.

Third, and most important for the case discussed in here, MIP problems are usually not solved to full optimality and often run deliberately in non-deterministic mode, such that the solver does not guarantee identical results on the same problem in repeated runs. This choice is made as non-deterministic solves are typically faster. Note here that guaranteed determinism from a solver perspective is defined strictly technical, not from a conceptual viewpoint. For instance, simply changing the order in which the equations enter the model might change results even in deterministic mode. The same clearly holds if purely informational equations are added which cannot alter from a conceptual viewpoint the optimal allocation. There might be subtle changes even in a MIP solution solved to full optimality in non-deterministic mode, which reflect scaling, feasibility and optimality tolerances.

What are the consequences of these observations for testing? First, tests should be run in a defined environment, such as the same software release and hardware, avoiding parallel load. Second, solvers should be used in deterministic mode. Third, if maximal solution times are deemed necessary, exceeding them should trigger a run time error instead of continuing execution based on such intermediate optimal outcomes. These conditions aim at ensuring that differences found must be based on changes in the code (or input data). Testing becomes much more demanding if (almost) identical results should also be guaranteed over a range of software releases and hardware set-ups. This is not discussed in here.

1.4 Statistics on code changes

In order to develop a test strategy, the frequency of code changes must be reflected after which testing is necessary. A clean checkout of FarmDyn encompasses about 500 files, all added at some point in the past the first time to the repository, and probably changed later multiple times. Such changes are called "commits", and as seen below, commit activity is not equally distributed over time, but shows peaks. In average, around 170 commits are taking place each year, the maximum number

[1] See the overview on past releases on the GAMS website (https://www.gams.com/34/docs/RN_MAIN.html, visited 2.2.2021)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

so far encountered were around 400 in 2018. This peak year reflects the generation of a so-called "stable release" where all team members re-integrated changes from their projects in a common master version. The red area indicates the contribution of the original developer.

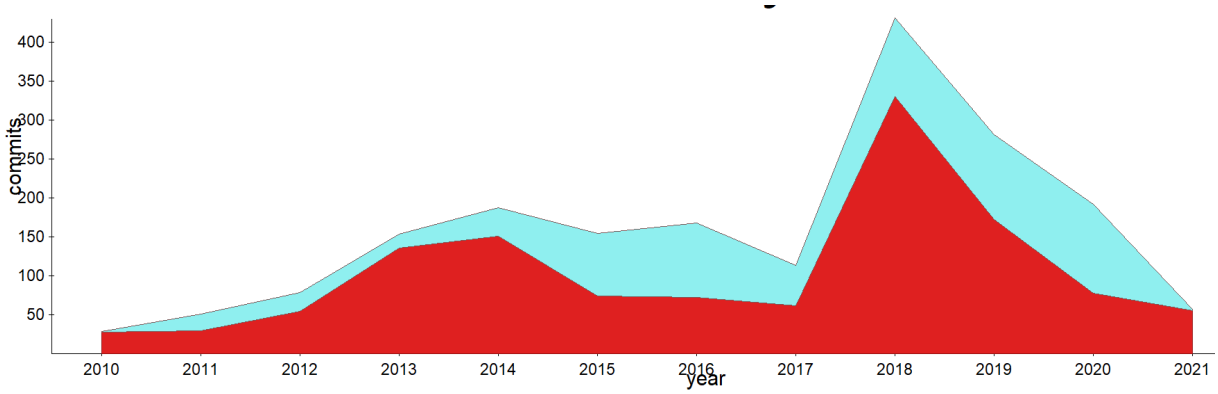


Figure 20: Commit activity over the last decade

Source: Generated with Tortoise SVN stats tool

How to: Statistics graph from Tortoise can be generated by producing a log ("Show log") and next pressing the "Statistics button" in the log window. There are multiple graphs, such as the selected "Commits by date".

A look at the last year 2020, see Figure 5, shows a span between a single commit in a month and close to eighty, the average is around 20. Most changes for a larger part of this year were done by a team member in a project which expanded the model in several directions (shown in red, not identical to the original developer). Around the middle of the year, there was a period were several developers in parallel changed larger parts of the code. These are periods were testing is probably most warranted. Each coder might concentrate his own testing on his current project input data and model configuration, and code changes might not be harmonized with each other. In average, there are changes to around 40 files per months, affecting around 10% of the total code base. Limited commit activity of a team member might reflect periods of concentration on other research activities, such as data work, literature research or publication of results. The charts suggest that more or less continuous testing is required, however, with varying load. The next section discusses a matching layered test approach.

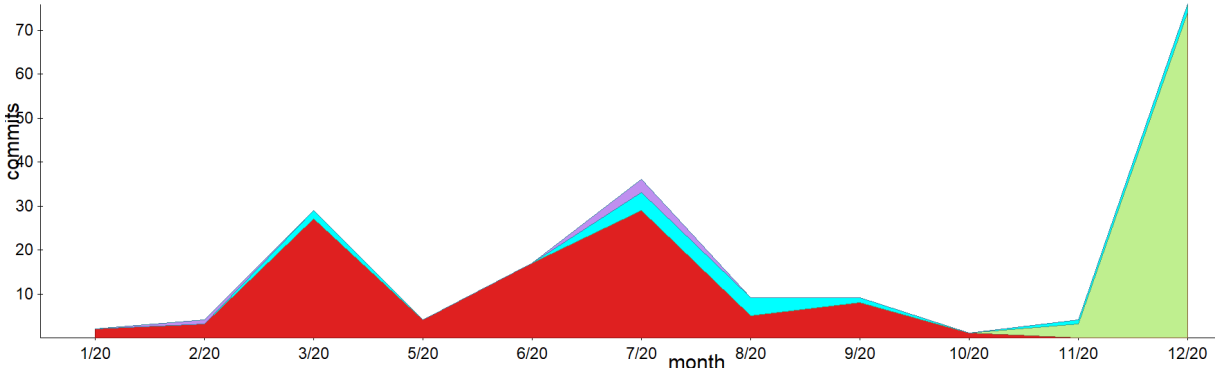


Figure 21: Commit activity in the year 2020

Source: Tortoise SVN stats tool

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

1.5 Test frequency and other details

A regular problem with the tests performed by researchers during their on-going code development is that not all of their code is fully synchronized with the master version. Their local tests can therefore fail on the master version once partial code updates are integrated. Maintaining code comprising local modifications might be necessary for a team member, and be it only to exclude that code changes committed by others to the master change results. Delaying partial commits, and instead sending off huge sets of changes files in one commit reduces problems related to inconsistent, more fine-grained changes. But it might also lead to very long log entries which are hard to digest when looking at the history of a single file. Commits changing many files might also mean that all other coders have to invest considerable time to deal with merged files, or even worse, with conflicts in their local working copies.

Table 4: Overview on the testing strategy in FarmDyn

	Frequency	# of test instances	Time investment
Compile time tests	Every hour if a commit occurred	~150	Very low
Run time tests	Every hour if a commit occurred	~20	Low
Outcome tests	The outcome of run time instances is checked once a week	~20	High

To avoid that a model version comprising local modifications is (involuntary) tested instead of the current master, it is necessary to use a separate so-called clean working copy for the tests, which has to be kept synchronized with the current head revision. Before tests are run, this local copy must hence be updated to reflect recent commits. The first two elements of the test strategy for FarmDyn, which comprise compile and run time tests which do not require manual checks, are run once per hour if there are new changes to the code and otherwise does not start the testing process. They are automatically triggered and executed by a Windows Task Scheduler which calls the batch file containing the compile and execution time test instances. The QM itself is written in R. It is the task of the current QM manager to check on a weekly basis if a new HTML page with test results is available. If the page reports failed tests, the QM manager will check the commit log to find out who performed the commit(s). These team members are informed and asked to correct the errors. They decide then if they debug the problem themselves, or if they involve additional team members to develop a solution strategy to fix the error.

Tests belonging to the outcome layer which require manual checks are also run on a weekly basis, should commits be observed during the week. The automated GDXDiff output will allow checking quickly if any changes in key indicators have occurred. The QM manager has to check these instances and to assess if the changes are critical. If this is the case, he will consult with the coders who committed during the week if the changes are intended or indeed implausible. In the latter case, the coders will have to decide who will have a look first.

1.6 What to do with failed tests?

Setting up the test strategy is a one-time activity, and updating the test instances is only necessary when new projects start which require their own test instances, or after larger structural changes to

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

the model. That leaves the regular testing as the major work load. Checking the tests themselves is a repeated activity with more or less known time requirements. As discussed above, this not the case for failed tests. In many cases, errors can be detected and corrected quickly. In other ones, errors reflect that some combinations of inputs possible from the GUI cannot be handled by the code. In these cases, options can be removed, restricted or errors thrown if such input combinations are detected. It is likely that bug fixing provokes new errors somewhere else, or that further run-time or outcome errors are detected afterwards when more code is executed by GAMS before aborting after initial errors are removed. It is therefore recommended to repeat all tests after errors are fixed

1.7 Appendix references

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

Britz, W., & Witze, P. (2014). CAPRI model documentation, version 2014. University Bonn, https://www.capri-model.org/docs/CAPRI_documentation.pdf

Britz W., Lengers, B., Kuhn, T. and Schäfer, D. (2016): A highly detailed template model for dynamic optimization of farms - FARMDYN, University of Bonn, Institute for Food and Resource Economics, Version September 2016, 147 pages.

Britz, W., 2020. Automated calibration of farm-scale mixed linear programming models using bi-level programming. Discuss. Pap. Ser. “Food Resour. Econ. 2020.

GAMS Development Corporation (2019): General Algebraic Modeling System (GAMS). Version 27.1.0. Fairfax, VA, USA.

Heinrichs, J., Kuhn, T., Pahmeyer, C., Britz, W. (2021): Economic effects of plot sizes and farm-plot distances in organic and conventional farming systems: A farm-level analysis for Germany. *Agricultural systems* Volume 187

Kuhn, T., Schäfer, D., Holm-Müller, K., Britz, W. (2019): On-farm compliance costs with the EU-Nitrates Directive: A modelling approach for specialized livestock production in northwest Germany, Agricultural Systems 173: 233-243

Lengers, B., Britz, W., Holm-Müller, K. (2014): What drives marginal abatement costs of greenhouse gases on dairy farms? A meta-modelling approach, Journal of Agricultural Economics 65(3): 579–599

Louhichi, K., Ciaian, P., Espinosa, M., Perni, A., & Gomez y Paloma, S. (2018). Economic impacts of CAP greening: application of an EU-wide individual farm model for CAP analysis (IFM-CAP). *European Review of Agricultural Economics*, *45*(2)

Schäfer, D., Britz, W., Kuhn, T. (2017): Flexible Load of Existing Biogas Plants: A Viable Option to Reduce Environmental Externalities and to Provide Demand-driven Electricity?, German Journal of Agricultural Economics 66(2): 109-123

Seidel, C., Britz, W. (2020): Estimating a Dual Value Function as a Meta-Model of a Detailed Dynamic Mathematical Programming Model, Bio-based and Applied Economic 8(1): 75-99

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.