iDesignRES

Integrated Design of the Components of the Energy System to Plan the Uptake of Renewable Energy Sources: An Open-Source Toolbox

Assembling multi physics component models in a receding horizon model

D2.2 System Assembling tool

Companion report

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Document information

Deliverable Number and Name	D2.2 System Assembling tool
Work Package	WP2 System modelling tool: Modules assembling and long-term planning
Dissemination Level	Public
Author(s)	Lucas Ferreira Bernardino, David Ribes Marzá, Jon Vegard Venås, Julian Straus, Hongyu Zhang
Primary Contact and Email	Julian Straus, julian.straus@sintef.no
Date Due	31/03/2025
Date Submitted	31/03/2025
File Name	iDesignRES_Deliverable_D2_2
Status	Submitted to EU portal
Reviewed by (if applicable)	Pedro Crespo del Granado and Mostafa Barani
Suggested citation	Assembling multi physics component models in a receding horizon model – Description of methods and computational advances for speeding up optimization

This document has been prepared in the framework of the European project iDesignRES. This project has received funding from the European Union's Horizon Europe programme under grant agreement no. 101095849.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Within the iDesignRES project, multiple component models were developed and publicly released in D1.2 *Multi-physics component models* [1][1]. While the majority of the individual component models can run as stand-alone models, it is especially beneficial to assemble the individual component models into a unified framework to perform analyses with multiple energy carriers. This is achieved through the so-called *System assembling tool* consisting of several developed Julia packages for the energy system modelling framework *EnergyModelsX* (*EMX*) corresponding to deliverable D2.2.

This report accompanies and supplements the deliverable D2.2. The deliverable consists of Julia packages extending *EMX* with features for sampling of component models written in other languages and solving the resulting optimization problem within a receding horizon framework. The detailed documentation of the different packages can be found in the respective repositories. The packages include examples on their application, a description on how to use the packages, as well as an overview of the API and extension approaches for other *EMX* packages. This companion report provides a high-level overview of the implemented features and their application.

Even if it is expected that the receding horizon framework reduces the computational burden of solving complex multi-physics problem through solving sequentially smaller optimization problems (we plan to utilize within iDesignRES problems which are 26 times smaller than the full problem), additional algorithmic methods can reduce the computational time even further. This is especially relevant for complex multi-physics operational analysis with high geographical, high temporal and high technical resolution. Different approaches to reduce the computational burden of model construction and model solving as well as an overview of the potential application of quantum computing for solving complex energy system optimization problems are hence presented in the companion report. Specifically, through modifying the *system-assembling tool*, it is possible to reduce the computational burden by 60 %. Bender's decomposition is another presented method resulting in a reduction in the computational burden by up to 85 %.

[1] The individual Multiphysics component models are available on https://github.com/iDesignRES.

1. Introduction

Analysing and operational stress testing of integrated energy systems at a high geographical resolution requires both detailed models for individual technologies or sectors (for an accurate representation of the dynamic behaviour) and measures to reduce the computational burden of the resulting optimization problem (due to the complexity of the individual dynamics).

Within the iDesignRES project, detailed multiple multi-physics component models were developed within Work Package 1. The individual models are released as Deliverable D1.2 [1] with the companion report providing an overview of the individual models and their features. The models are developed as stand-alone models in different programming languages. However, there exist couplings between the individual models and the modelled energy carriers. As an example, consider the solar PV model which can provide the user with a production profile within a geographical region for a given capacity through optimally investing in the available land. As electricity is also required as input for electrolysis or natural gas reforming with CO_2 capture models, it must be included in a unified multi energy carrier model to avoid inconsistencies in the analyses. The individual component models are hence unified within an assembly tool, either through direct integration or through sampling of the multi physics component model to provide improved input data.

As outlined above, solving such a large-scale optimization problem with a high geographical and technical resolution and varying annual profiles is in general infeasible due to the high computational burden. Hence, the unified framework utilizes a receding (rolling) horizon framework in which the overall optimization problem is split into a set of sequential horizons, each corresponding to an individual optimization problem. This results in more tractable problems in which, *e.g.*, the initial conditions of horizon 2 is provided by horizon 1.

This companion report to deliverable D2.2 provides an overview of the individual developed packages, their features, and how to apply them. The main deliverable corresponds to two developed packages for the *EnergyModelsX* (*EMX*) [2] framework which are openly available on GitHub[2] under the MIT license. *EMX* is a multi-nodal, multi-carrier energy system modelling framework written in Julia, based on the *JuMP* [3] algebraic modelling language. The developed packages provide additional functionality to *EMX*. Section 2 introduces the package *EnergyModelsInterfaces* which provides functionality for interfacing C++ and Python models with Julia. Section 3 introduces the concepts behind the package *EnergyModelsRecedingHorizon* which adds support for a receding horizon framework on top of existing *EMX* models. Sections 4 and 0 introduce both features for reducing the computational time for model construction and solving the model.

[2] Available on https://github.com/EnergyModelsX.

2. Linking and sampling the component models

2.1 Concept

The concept of linking and sampling the component models revolves around creating a cohesive framework that integrates various multi-physics models developed in WP1. This framework aims to facilitate the modular assembly of these models, ensuring seamless interaction and data exchange between them. The ultimate goal is to provide a multi-carrier operational model with NUTS Level 2 resolution, capable of stress testing the energy system under extreme conditions.

The EnergyModelsX framework will be used as the core of the system-assembling tool as it provides great flexibility in including new structures to be integrated. Moreover, it is openly available and is written in the open programming language Julia that facilitates packages that can easily be used to connect to other packages written in other programming languages.

2.2 Implemented features

The system-assembling tool[3] integrates various multi-physics component models (developed in deliverable D1.2), including:

- Building demand module [4], written in Python (based on REST API).
- Industrial demand module [5], written in Python.
- Transport demand module [6], written in Julia.
- Packages in the *EMX* framework, written in Julia
  - Renewable energy technologies [7] (*EnergyModelsRenewableProducers*)
  - Hydrogen technologies [8] (*EnergyModelsHydrogen*)
  - CO_2 infrastructure [9] (*EnergyModelsCO2*)
  - Thermal energy infrastructure [10] (*EnergyModelsHeat*)
  - Transmission infrastructure [11] (*EnergyModelsGeography*)
- Nuclear module, written in C++ (part of the SMS++ [12] framework)
- Combined Heat and Power Module [13], written in C++ (part of *EnergyModelsHeat*)
- Solar PV module [4], written in Python (based on REST API).
- Wind power production module [14], written in Python

Emphasis is placed on developing generic sampling routines to facilitate the incorporation of new components. A sampling routine is a specially tailored function used to incorporate new components into the system-assembling tool by evaluating the component with its required input parameters and using the required output of the component in the system-assembling tool. The sampling routine can consider any set of arguments or keyword arguments for a Python function and any number of arguments for a C++ function.

The models that are written in C++ and should be sampled expose a function through the *CxxWrap* package [15]. The integration of components written in Python are achieved using *PyCall* [16] in Julia to evaluate Python projects installed in a conda [17] environment using, e.g., Poetry [18]. All modules in *EMX* are integrated directly and have no need for sampling.

For the routine calling Python functions, it is assumed that the current activated Python environment contains the Python module to be sampled. The `call_python_function` routine then takes the Python module name and the name of the function to be sampled in addition to its arguments as input arguments (or keyword arguments).

[3] The system-assembling tool is available on https://github.com/EnergyModelsX/EnergyModelsRecedingHorizon.jl.

For the routine calling C++ functions, slightly more manual work is needed as it is required from *CxxWrap* that the C++ code is altered to expose the function to be sampled. The manual approach needed for this package is described in the next section.

For each component, a new *EMX* element is implemented for simple integration. Constructors of these new nodes are then implemented to sample the above components with the aforementioned approach.

2.3 How to use the sampling functionality

As mentioned, all modules not implemented directly in the EMX framework must be sampled, as direct integration is not possible. First one must provide an internal function in the Python or C++ model which is externally callable, then one must call said function from an internal constructor in EMX for a node description. The node can utilize existing nodal descriptions or provide new additional information if the original function provides a non-linear output.

For models written in C++, to expose the function to be sampled, one needs to install libcxxwrap-julia [19] and wrap the sampling function into a JLXX_MODULE as described in the documentation of *CxxWrap*. Once this code is compiled into a shared library it can be used in Julia by exposing the function in a separate Julia module. The *EMX* constructor mentioned above can then evaluate the sampling function through this module.

The current implementation is limited to single function evaluations in Python or C++. It is as of now not possible to provide an improved common format due to the large specialization of the individual component models. Consequently, it is not possible to develop a fully unified approach for automatically creating *EMX* nodes as the internal mathematical description is dependent on the sampled data.

3. Receding horizon framework for operational stress testing

3.1 Concept

Receding horizon is a concept coming from the control systems community, widely used in model predictive control (MPC) [20]. It refers to a method for solving dynamic optimisation problems in real time, such that current measurements and future predictions of relevant parameters are updated in time. In control problems, this is used based on the insight that information about the distant future has little impact on short-term operation. The concept of receding horizon also finds application in energy systems modelling, especially when short-term dispatch decisions are affected by uncertainties [21]. In dynamic optimisation problems, these uncertainties can be represented by operational scenarios that must be optimised simultaneously, see Figure 1. This substantially increases the complexity of the optimisation problem.

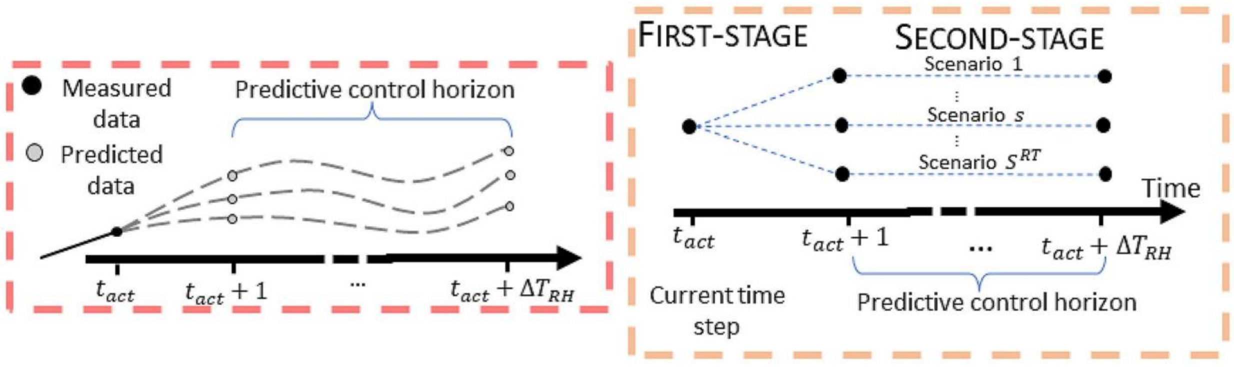


Figure 1: Example of receding horizon optimization for uncertain energy systems (adapted from [21], published under the CC BY-NC-ND 4.0 license).

In this work, the receding horizon concept is used to decompose a large-scale optimisation problem into smaller problems with limited time horizons. This means that previously intractable problems, due to large time horizons and number of system elements, can now be solved with higher computational efficiency, which will in turn facilitate stress testing of operational scenarios. This approach still allows for modelling the system operational dynamics in the fast time scales, while retaining information about the system behaviour in the longer time scales.

3.2 Package description

The receding horizon optimisation scheme is implemented as a package for the *EMX* framework. The developed package[4] introduces the required functionalities for solving optimisation problems in a receding horizon scheme, and it uses the *EMX* functionalities for description of technologies and resources. As such, systems defined within *EMX* are easily adaptable to solve in the receding horizon package. The released version of the package is compatible with the node, link, area, and transmission types introduced in *EnergyModelsBase* and *EnergyModelsGeography*, which correspond to core energy system functionalities and geographical descriptions, respectively.

The main implemented functionalities for the receding horizon optimisation package are described below.

[4] Available on https://github.com/EnergyModelsX/EnergyModelsRecedingHorizon.jl.

3.3 Implemented features

3.3.1 Horizon update functionality

The receding horizon framework is based on time decomposition of the full-scale optimisation problem. Each subproblem is solved over a finite time horizon, which we denote as *optimisation horizon*. The solution for this subproblem is stored for a usually smaller time horizon, which we denote as *implementation horizon*, and the next subproblem is posed to begin at the end of this implementation horizon. This approach is similar to MPC in which only the results of the first time step is implemented while the next iteration of the optimization problem receives feedback from the controlled process. In our case, we do not receive feedback from the process but instead have more knowledge regarding the future energy demands or generation profiles in subsequent iterations.

The implementation and optimisation horizons can be defined in different ways. To understand that, note that a dynamic problem can be posed with operational periods that have varying durations:

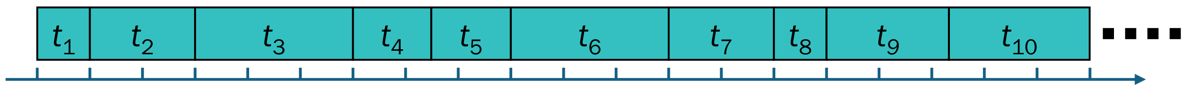


Figure 2: Example of time structure for a dynamic problem.

Using different durations for the individual operational periods allows for reducing the temporal resolution in periods in which both renewable power generation and demand profiles do not vary significantly, *e.g.*, the night, while maintaining a high temporal resolution in the other periods. Consequently, the number of operational periods can be reduced.

In the developed framework, there is support for two types of horizon structures:

- Period horizons: the length of the implementation and optimisation horizons is governed by the number of operational periods in the horizon as shown in Figure 3. As a consequence, it is possible that the total duration of the *implementation* and *optimisation horizons* can vary for each iteration.

  In Figure 3, we specify an implementation horizon of 2 periods and an optimisation horizon of 5 periods. In the first iteration, the total duration of the implementation horizon is 3 while the total duration of the optimisation horizon is 9. The second iteration changes this to 4.5 and 11, respectively. This is especially pronounced for short horizons and large variations in the duration of the individual periods.

- Duration horizons: the length of the implementation and optimisation horizons is governed by their total duration as shown in Figure 4. As a consequence, we can have a different number of operational periods in the individual iterations. Note that we specified the horizons so that the total duration of both implementation and optimization horizon must be at least the specified value. This implies that it can be larger, if the total duration would end within an operational period.

  In Figure 4, we specify an implementation horizon of 3 time units and an optimisation horizon of 8 time units. Similarly to the case of period horizons, it is possible that the effective horizon to be analysed is larger than the specified ones, here due to the calculation horizons needing to contain whole operational periods. For the first iteration, the optimisation horizon corresponds to 5 operational periods, whose duration sums up to 9 time units, and the implementation horizon corresponds to 2 operational periods, with duration of 3 time units. For the second iteration, which starts from the third operational period, the optimisation horizon corresponds to 4 operational periods, and the implementation horizon corresponds to a single operational period.

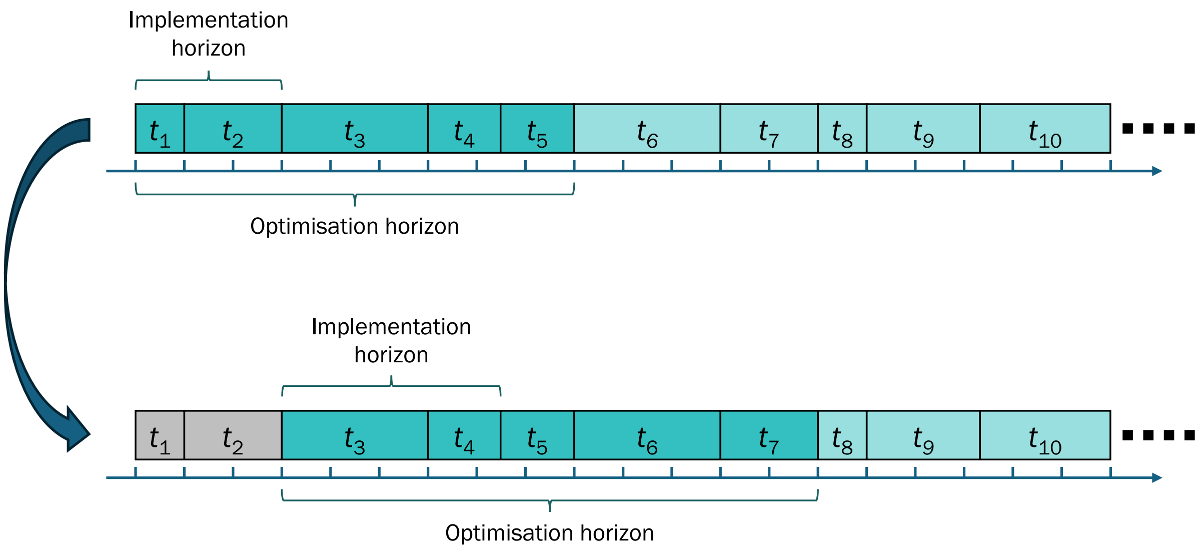

Implementation horizon
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10}
Optimisation horizon
Implementation horizon
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10}
Optimisation horizon

Figure 3: Example of problem update based on period horizons (implementation horizon of 2 periods, optimisation horizon of 5 periods).

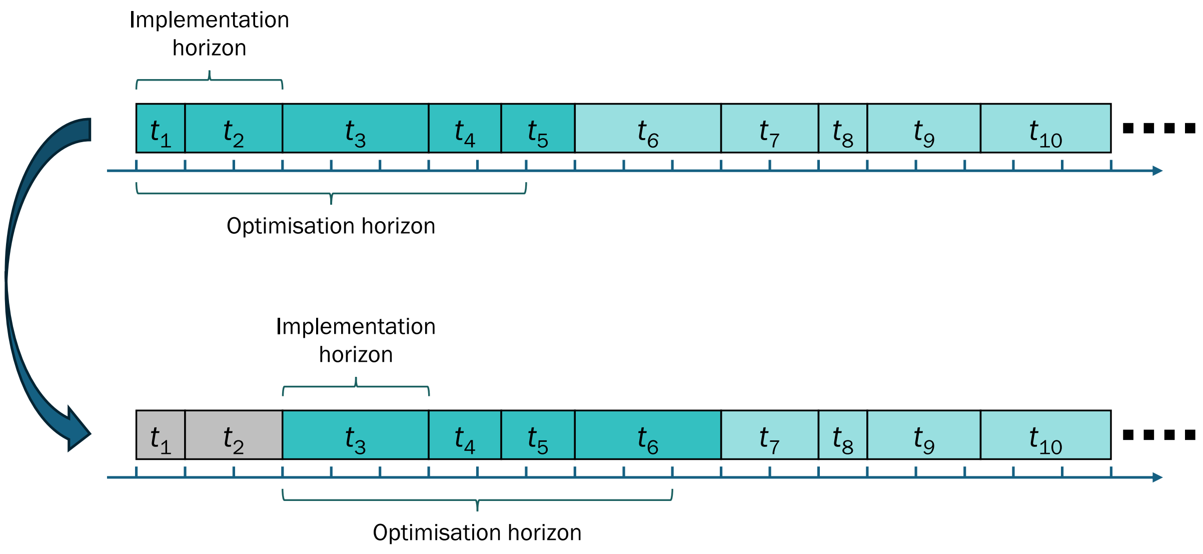

Implementation horizon
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10}
Optimisation horizon
Implementation horizon
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10}
Optimisation horizon

Figure 4: Example of problem update based on duration horizons (implementation horizon of 3 time units, optimisation horizon of 8 time units).

The two types of horizons can be exchanged by the user at will. The package will automatically adjust the chosen time periods in each iteration as well as which periods are saved in the results file.

3.3.2 Support for flexible system initialisation

The developed framework allows for explicitly defining the initial states of a system or subsystem. This is especially relevant for dynamic systems, where a state depends on the previous operation. This functionality is crucial within the receding horizon framework to continuously solve slices in time of the full optimisation problem.

In the receding horizon framework, an initialisation setting is generated for every optimisation problem solution, corresponding to the state of the system at the end of the implementation horizon. This will be the starting condition of the next optimisation problem, and it is consistent with the solution of a large-scale optimisation problem.

In the developed package, the user is provided with a toolkit that works for most dynamic systems, where the system's state is a given time can be completely determined by knowing the state of the system at the immediately previous time period. While this is the case for systems such as storages, this may not apply for systems with unit commitment, such as minimum up- or down-time. For these cases, the package can be easily extended to deal with more complex initialisation settings through the incorporation of new initial data types.

3.3.3 Future value descriptions

As explained above, the receding horizon framework optimizes multiple sequential horizons, where the outgoing state of the previous implementation horizon is used as the initial state of the next optimisation problem. The receding horizon approach can be useful to break down a complex optimisation problem into multiple smaller ones. However, the value of the outgoing state at the end of each individual optimization problem should also be accounted for in the case of dynamic states, that is states which are dependent on their value in the previous period(s). Modelling large-scale storages requires providing a value to the stored energy/mass at the end of the optimization horizon. If a value is not include, the receding horizon solution will tend to empty the storage entirely at the end of each optimisation horizon, the optimal behaviour within the optimization horizon, while this behaviour may not be optimal in the full problem.

To that end, we implement storage end values into the receding horizon calculations. These introduce an additional cost to the receding horizon optimisation problem, based on the expected future value of the storages. The reader is referred to Aaslid *et al*. [22] for a more detailed explanation of the storage end value concept. A summarized description is given below.

Introduce ℓ^{∞} as the future value of storages in a system. This value is constrained by multiple linear cutting hyperplanes according to:

$$\ell^{\infty}[v] + \sum_{(s,c_s) \in \mathcal{C}} c_s \times \ell[s, t_{end}] \le rhs[\mathcal{C}] \qquad \forall \mathcal{C} \in \mathcal{C}_v, v \in \mathcal{V}$$

Here, $\mathcal{V}$ represents the set of storage value cuts v, and $\mathcal{C}_v$ represents the set of cuts in v. Each cut $\mathcal{C}$ in $\mathcal{C}_v$ presents a set of coefficients c_s associated with a storage element s and their corresponding final value at the considered horizon $\ell[s, t_{end}]$, and a right-hand-side parameter $rhs[\mathcal{C}]$.

The penalty added to the optimisation problem is:

$$\sum_{v \in \mathcal{V}} w_v \times \mathrm{w}_v^{\mathrm{t}} \times \ell^{\infty}[v]$$

Here, w_v is a constant weight related to the set of storage value cuts v, and $\mathrm{w}_v^{\mathrm{t}}$ is a time-dependent weight, updated for each optimisation problem to be a linear combination of the active sets of cuts, if multiple sets of cuts are active.

3.3.4 Support for geographical descriptions

To describe large-scale energy systems, it is important to describe the different geographic areas with the individual energy systems, as well as the interconnections between these areas. This was done through developing compatibility with the existing sister package *EnergyModelsGeography*, which has the required functionalities for describing these types of problems. This means that it is also possible to solve problems with such complexity within the receding horizon framework.

3.4 How to use the package

The main points to address when adapting an existing *EMX* model definition to the current package are related to defining the settings of the new functionalities. This includes the

- definition of the horizon structure for the problem,
- initialisation settings for each system node and how these are used in the model, and
- definition of the future value formulation for the applicable nodes.

We illustrate how to address these points through the examples provided with the package. Note that you must provide the profiles of the complete time horizon for the individual technologies, that is the profiles spanning the complete horizon, typically a year of analysis.

Contrary to other *EMX* packages, the current package does not solve a single optimisation problem, but a sequence of optimisation problems. As such, even though all subproblems are implemented in *JuMP* [3], running a model in this package does not return a list of the solved models. It instead returns the overall results in a *DataFrame* [23] format, containing results related to the respective implementation horizons.

The package is implemented with the focus on calculating supply-demand balances for systems in an operational time scale to identify potential problems with the invested capacities from capacity expansion models. It does not support investment analysis. In addition, the calculation of operational costs for full periods is not implemented, although these can be calculated with the existing information through the store values of the variables. This will be implemented in future releases.

4. Algorithmic approaches for improving computational speed

As noted before, the iDesignRES project aims to have advanced computational technologies to enhance energy system modeling capabilities. The project will implement a cloud-hub platform (WP4 and WP5) to enable direct online execution of some energy system models with the aim to significantly improve computational performance through cloud services and supercomputing resources. This infrastructure facilitates more efficient software testing, version control, and accessibility for average users. The platform supports high spatial-temporal resolution modelling (sub-hourly at NUTS level 2 regions) while maintaining detailed energy system physics and modular model assembly capabilities. Within the work carried out in Task 2.2, we developed or improved methods and algorithms to reduce the computational time in executing energy system models. This chapter details these results while chapter 5 outlines exploring new frontiers in computational paradigms by applying energy system models in quantum computers.

4.1 Improved construction of models

As outlined previously, the *EnergyModelsX* framework, based on the Julia algebraic modelling language *JuMP*, is utilized for the implementation of the receding horizon framework. The *EMX* framework utilizes immutable types for providing the parameters to the optimization problem to avoid issues with unintentional overriding of parameters within the programme. Similarly, the individual parameters in the optimization problem are directly included in the *JuMP* problem.

However, in the case of a receding horizon framework, we must update temporal profiles, initialization data, and other parameters of the optimization problem within each iteration of the problem as demand and generation profiles as well as initial data change. While the updates can be achieved through recreating the case and optimization problem, it can be computationally costly to recreate both in each iteration. Consequently, new approaches must be developed for updating both the case description and the optimization problem. These new approaches must be flexible enough to be extended to new parameters that must be changed in the different iterations.

The actual improvement in computation speed is depending on the complexity of the optimization problem. If the optimization problem is in general hard to solve due to the inclusion of binary variables and unit commitment constraints (*e.g.*, requiring a minimum operational point or alternatively a minimum time for startup or shutdown), the improvement may be small as solving the optimization problem can take significantly longer than building the optimization problem.

4.1.1 Lenses for resetting values of immutable types

The developed package utilizes the *Julia* package *Accessors* for changing the values of the individual parameters of the technology elements. This package provides so-called lenses which point towards the position of a given parameter within the immutable type. The lenses can be used to both access and reset the values of immutable types by creating a new version of the original type at an increased speed. Tests showed that changing the values of parameters through lenses increases the updating of model parameters by 10-20 %, depending on the number of parameters to be updated.

Lenses are incorporated into the model through a new case type which provides a comprehensive overview of the different parameters that must be updated in each iteration of the receding horizon framework as well as the original and updates elements. Most of the required functionality for solving a receding horizon model is directly included in the package while unforeseen functionality can be implemented in a simple way as outlined in the package's documentation.

4.1.2 Declaring parameters as variables

While the inclusion of lenses significantly improves updating the *EMX* case, it is still necessary to create a new *JuMP* optimization problem in each iteration of the receding horizon framework, even if only a limited number of parameters is updated. This can be circumvented through the utilization of the Julia package *ParametricOptInterface* (*POI*) which is an extension to *JuMP*. With *POI*, parameters can be included in the optimization problem as variables with a fixed value. The value of the variables can be updated in each iteration. While this approach is feasible in all algebraic modelling languages, it would lead in most cases to quadratically constrained optimization problems due to the multiplication of variables, and hence, bilinear terms. *POI* treats the parameter variables differently, and hence, the implementation results in a linear problem when parameters are multiplied with variables. The application of *POI* can improve the computational speed significantly. In the case of a simple model in which 50 iterations are run (corresponding to a full year with an implementation horizon of 1 week), the receding horizon problem is solved in 40 % utilizing *POI* compared to the required time when not using *POI*.

POI is included as an extension to the receding horizon framework. This implies that the user can decide to use it in each individual model run. It utilizes the same case type as outlined in Section 4.1.1 for lenses and what new values should be used for the parameters. It can only be used with horizons based on the number of periods as the optimization problem would change when the number of periods changes (see also Section 3.3.1 for an explanation on horizon types). In addition, it is necessary that the duration of the individual periods is the same in each optimization horizon. It is still possible to have varying durations for the individual periods, but the durations must be repeating over the optimization subproblems.

4.2 Solution methods for improving computational speed

4.2.1 Introduction

Managing multi-timescale uncertainty is crucial for infrastructure planning, especially in long-term energy system planning, which plays a key role in achieving net-zero energy transition. Uncertainties in energy planning span multiple timescales, typically including long-term (years/decades) and short-term (hourly or sub-hourly) uncertainties [24], [25]. Tactical timescales, involving seasonal storage, have also been investigated for systems with significant seasonal storage [26].

A widely used approach for managing multi-timescale uncertainty is Multi-Horizon Stochastic Programming (MHSP)[24], a type of multi-stage stochastic programming with block separable recourse[27]. MHSP reduces problem size by partially disconnecting short-term and long-term decision nodes. Despite this, MHSP problems remain computationally challenging, requiring efficient decomposition algorithms.

To address computational difficulties, various decomposition algorithms have been proposed [28], [29], [30], [31]. Benders decomposition and Lagrangean decomposition have been used to decompose MHSP, and progressive hedging has been explored as a solution method. Stabilized Benders decomposition [29], [30], with adaptive oracles, was introduced to improve efficiency in large-scale problems. This method avoids solving all subproblems in every iteration, reducing computational effort while maintaining accuracy.

This report summarizes the findings of a separate publication on different improved formulations of Benders decomposition. The study extends the literature by utilizing parallel computing for Benders decomposition in MHSP. Stabilization techniques, such as centered point stabilization, address oscillation issues in highly degenerate models, which are common in multi-region energy system planning. Recent research has shown that stabilization significantly improves computational performance.

While previous work has focused on computational methods, the impact of multi-timescale uncertainty on planning decisions remains underexplored. The study provides a comprehensive analysis of how short-term and long-term uncertainties influence energy system planning. Using the Renewable Resource Investment for the Energy Transition (REORIENT) model and the proposed solution algorithm to an integrated European energy system planning problem under uncertainty [32], the study shows that the parallel stabilized Benders decomposition method is up to 7.5 times faster than the serial version.

4.2.2 Parallel stabilized Benders decomposition

Parallel Stabilized Benders Decomposition is a method to improve the computational efficiency of solving large-scale stochastic programming problems with multi-timescale uncertainty. Traditional Benders decomposition, widely used for such problems, breaks them down into a master problem (MP), which determines high-level strategic decisions, and multiple subproblems (SPs), which handle operational decisions under uncertainty. However, when applied to energy system planning, this method becomes computationally expensive due to the large number of decision nodes and the high-dimensional solution space.

To address this challenge, the study presents a parallelized version of Benders decomposition. Instead of solving subproblems sequentially, they are distributed across multiple processors, allowing them to be solved simultaneously. Once all subproblems are solved, their solutions are aggregated and used to update the master problem. This synchronous execution significantly reduces computation time. A centered point stabilization technique [30] is employed to mitigate oscillation issues that often arise in Benders decomposition when applied to highly degenerate models, such as multi-region energy systems.

Parallelization is implemented in two ways. First, multi-threading distributes workloads across multiple cores within a single computer. Second, multiprocessing allows subproblems to be solved on separate machines connected via a network. The hybrid parallelization approach ensures efficient use of computing resources. A master processor coordinates data transfer, distributes computational tasks, and synchronizes results. While multi-threading enables faster computations within a single machine, multiprocessing improves scalability by leveraging multiple computers, though it introduces some communication overhead.

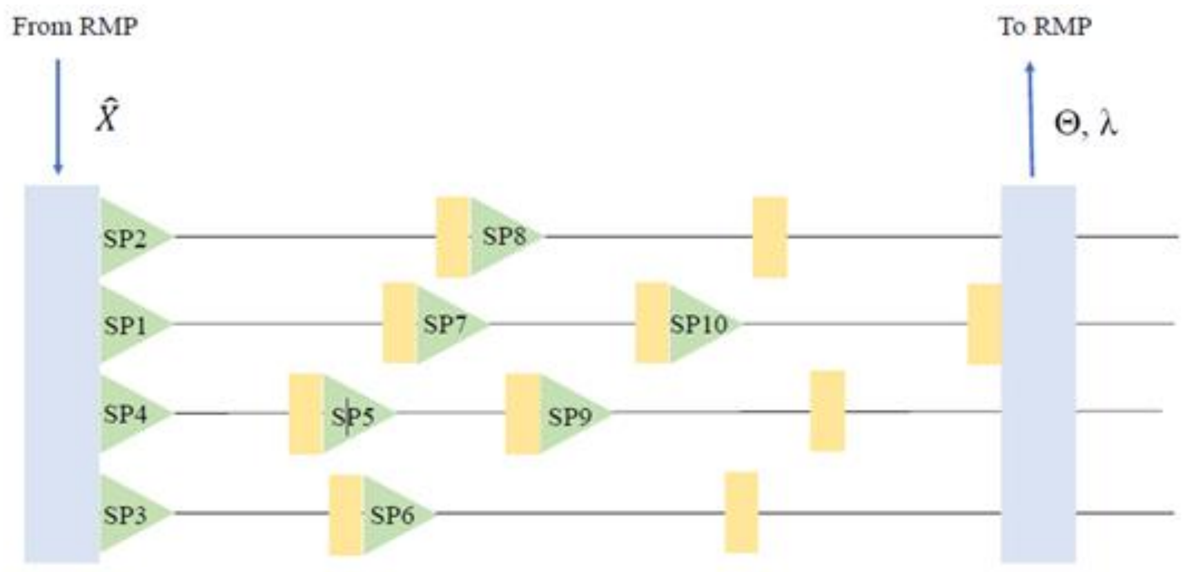


Figure 5: Whenever the computer resource is available, we start to solve a new subproblem. When all subproblems are solved, the information is fed back to the Relaxed Master Problem (RMP).

The efficiency gains increase as the problem size grows, although performance does not scale linearly with the number of computing resources. Simply adding more processors does not guarantee a proportional reduction in solving time due to the increased communication demands. Nonetheless, this approach significantly enhances the feasibility of large-scale energy system planning, enabling the analysis of more complex and realistic scenarios.

Figure 5 provides a visual representation of the parallelization process, illustrating how subproblems are solved in parallel and how information flows back to the master problem. This figure clarifies the computational structure of the method and highlights how synchronized execution reduces delays.

4.2.3 Problem and Model Description

The energy system planning problem is formulated within the REORIENT model, which integrates investment and operational decisions for a European energy system under uncertainty. The objective is to determine an optimal strategy for investment, abandonment, and retrofit planning while also considering operational scheduling to meet energy demand at minimum cost.

The problem incorporates two layers of uncertainty: short-term uncertainty, which includes variability in renewable energy availability, hydropower production, and electricity demand, and long-term uncertainty, which covers factors such as oil and gas prices, CO_2 emission regulations, and technological cost developments. These uncertainties are modelled using multi-horizon stochastic programming (MHSP), ensuring that both short-term and long-term variations are explicitly considered in decision-making.

Investment decisions involve determining the capacity of various energy generation technologies. These include traditional thermal generators (coal, gas, nuclear, and biomass), generators with carbon capture and storage (CCS), renewable sources (wind, solar, wave, hydro, geothermal), energy storage technologies (hydropump storage, lithium-ion batteries), and hydrogen infrastructure (electrolyzers, hydrogen storage, and pipelines). Additionally, investments in electric transmission networks and clean energy hubs are considered. The capital and fixed operational costs for these technologies are assumed to be known.

The retrofit planning aspect includes upgrading existing natural gas pipelines for hydrogen transport and repurposing offshore platforms for clean energy hubs. Additionally, abandonment decisions for mature fossil fuel fields are considered. The model determines both the capacity of new and retrofitted technologies and the optimal operation of the system, including the scheduling of generation, storage, and power flows between regions. The objective is to minimize the combined investment, operational, and environmental costs while ensuring compliance with emission targets.

The REORIENT model maintains the standard assumptions and modelling strategies used in previous research while it incorporates multi-timescale uncertainty more comprehensively. The geographical scope covers 27 regions across Europe, each of which can deploy 36 different technologies. The network includes 87 transmission lines, as well as existing and candidate hydrogen pipelines that may be retrofitted from natural gas infrastructure.

The energy system planning problem formulated in this model captures the complexity of decision-making under uncertainty. By integrating short-term fluctuations with long-term policy and market uncertainties, it provides a structured framework for assessing investment strategies and ensuring a cost-effective and resilient transition to a low-carbon energy system.

4.2.4 Case study and Computational Results

The extended REORIENT model is applied to the integrated strategic planning of the European energy system [32]. The network topology is illustrated in Figure 6. The investment planning horizon extends to 2050, with five-year planning steps to account for long-term infrastructure decisions and operational adjustments.

The model and solution algorithm are implemented in Julia 1.8.2 [33], using JuMP [3] for mathematical programming and Gurobi 10.0 [34] as the solver. The computational experiments are performed on a high-performance computing cluster, where each node is equipped with a 2x 3.5 GHz 8-core Intel Xeon

Gold 6144 CPU and 384 GB of RAM, running CentOS Linux 7.9.2009. The largest problem instances contain up to 55 million continuous variables, 1,876 binary variables, and 116 million constraints, demonstrating the scalability of the proposed approach.

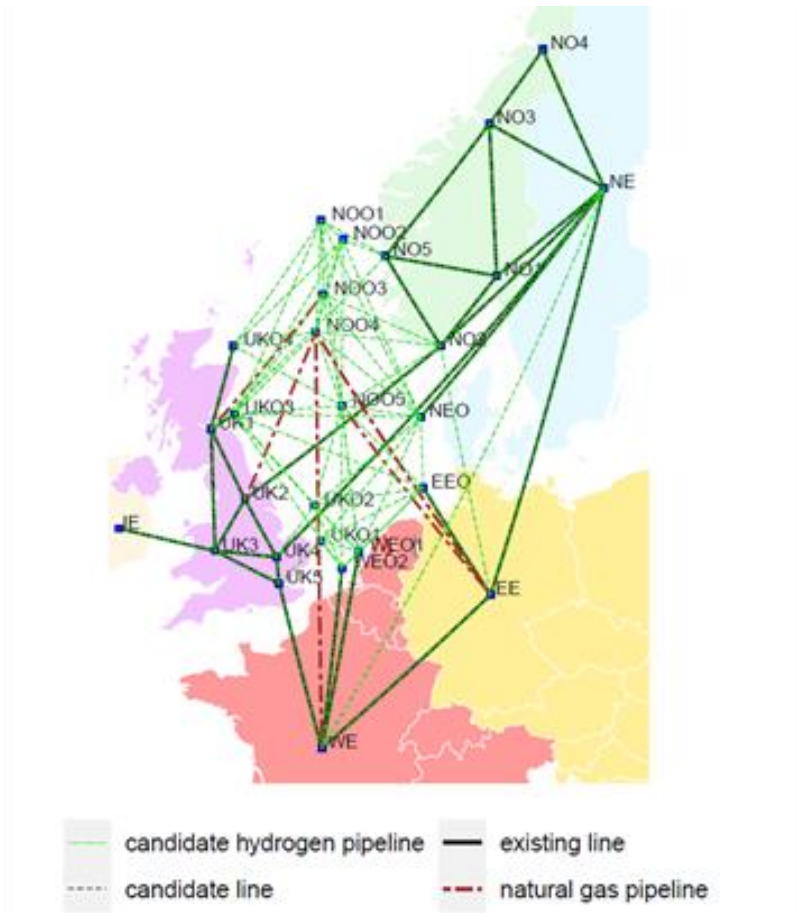


Figure 6: Illustration of the considered European energy system. The considered system includes 27 regions (each region can deploy 36 technologies), 87 transmission lines, 7 existing natural gas pipelines that can be retrofitted for hydrogen transport (some are overlapped), and 87 candidate new hydrogen pipelines.

The computational performance of the parallel stabilized Benders decomposition is evaluated by comparing three different implementations:

1. Serial implementation (standard Benders decomposition)
2. Multi-threaded implementation (parallel execution on a single computer)
3. Distributed implementation (parallel execution across multiple computers)

The problem instances used for performance evaluation are summarized in **Error! Reference source not found.**, which lists the number of operational periods, short-term and long-term scenarios, and decision nodes for each case. The four cases differ in problem size, with the largest instance involving 985 operational periods, 4 short-term scenarios, and 8 long-term scenarios, resulting in 55 million continuous variables and 120 million constraints.

	Distributed			Multi-threaded			Serial		
	Solving time (s)	Overhead time (s)	Total (s)	Solving time (s)	Overhead time (s)	Total (s)	solving time (s)	Overhead time (s)	Total (s)
Case 1	1826	324	2150	2373	195	2569	6722	179	6901
Case 2	4805	588	5393	6254	288	6543	18 781	213	18 994
Case 3	2630	357	2987	5377	360	5737	22 052	209	22 261
Case 4	8241	720	8961	14 801	733	14 801	55 028	240	55 269

Table 1: Overview of the cases used in the computational study.

The results of the study indicate that both multi-threaded and distributed implementations significantly outperform the serial implementation. Compared to the serial approach, the distributed implementation achieves a speedup of up to 7.5 times, while the multi-threaded implementation is up to 3.9 times faster. As the problem size increases, the advantages of parallel computation become more pronounced, highlighting the importance of scalability in large-scale energy system planning.

To further investigate the efficiency of the distributed implementation, additional experiments are conducted with an increased number of subproblems (105 subproblems). The results show that increasing the number of processors improves solving time, but the performance gain is not perfectly linear due to communication overhead. A comparison between three and six processors demonstrates that while additional computational resources enhance performance, the efficiency gain diminishes as more processors are added.

Overall, these results confirm that parallel stabilized Benders decomposition is a highly efficient solution method for large-scale stochastic optimization problems. The approach enables the simultaneous consideration of multi-timescale uncertainty while maintaining computational tractability, making it well-suited for strategic energy system planning at the European scale.

5. Exploring new frontiers to improve computational time in energy systems – Quantum computing

As part of task 2.2, iDesignRES explored new frontiers and applications of quantum computing to energy system modelling.

Quantum computing leverages the principles of superposition and entanglement to solve complex problems more efficiently than classical methods. Among quantum approaches, quantum annealing is particularly suited for optimization problems, as it seeks the minimum energy state of a Hamiltonian, which corresponds to the optimal solution of an objective function.

A quantum annealer encodes an optimization problem into an Ising model or a Quadratic Unconstrained Binary Optimization (QUBO) formulation. The system is initialized in a well-defined quantum state and then evolves according to the adiabatic theorem, gradually transforming into a final state where the lowest-energy configuration represents the optimal or near-optimal solution. This makes quantum annealing particularly effective for combinatorial optimization problems, where classical approaches struggle with exponential scaling.

As an example, D-Wave’s quantum annealers are designed specifically for optimization tasks, employing superconducting qubits to implement these annealing processes. Hybrid solvers – such as D-Wave's LeapHybridCQMSolver - integrate classical preprocessing and postprocessing steps with quantum computations, allowing them to tackle larger and more complex problem instances (*i.e.*, mixed problems) than purely quantum approaches. However, their advantage over classical solvers remains problem-dependent, with the best performance observed in cases closely related to the native Ising model structure.

5.1 Unit Commitment Problem – An Energy System Application

Unit commitment problems are critical in energy system optimization, involving the scheduling of generating units to minimize total operational costs over a given period. They are formulated as Mixed Integer Linear Problems (MILPs), consisting of minimizing a cost function while ensuring various constraints are met, such as demand satisfaction, logical conditions, capacity limits, and ramping limits. Generally, they entail both binary and continuous variables. Thus, D-Wave's LeapHybridSolver is suited for this problem, which has been tested on multiple problem scales.

The largest analysed problem scale has 44544 variables, 42899 constraints, 48h time periods and 4 linearization segments. Initial tests using the default minimum runtime of 20 s fail to produce feasible solutions. As seen in Figure 7, increasing the runtime to 150 s and 400 s leads to improvements, though the solutions remain suboptimal. The best solutions obtained are significantly worse than those produced by Gurobi, which finds near-optimal solutions within 180

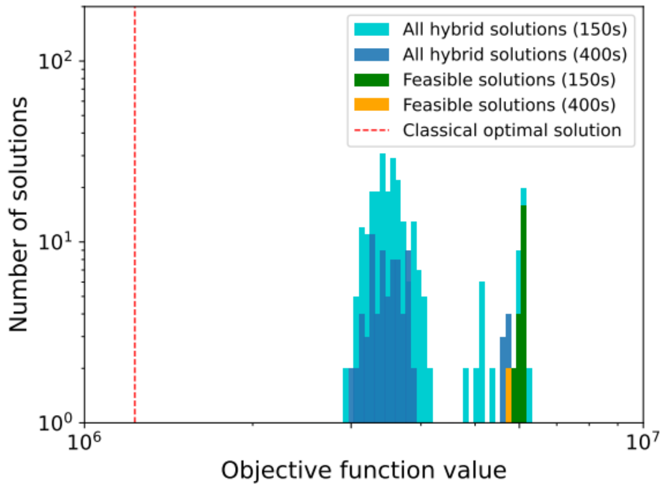
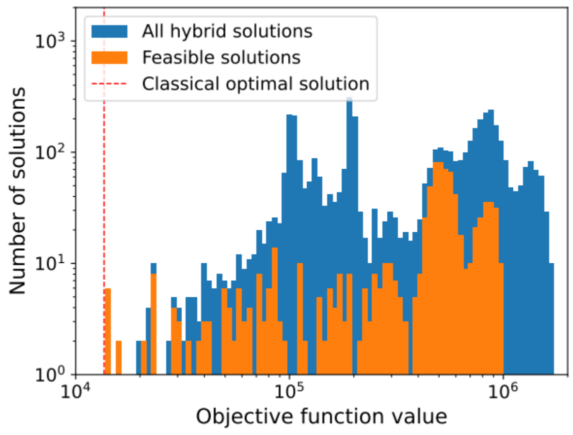


Figure 7: Solution distribution from D-Wave's LeapHybridCQM Solver from the full-scale unit commitment problem. The different cases are 6 runs for an increase run time of 150s and 2 runs for one of 400s. Each run supplies a sampleset with an average of 100 solutions.

To analyse how D-Wave performs with a downsized problem, it is solved in two new different instances, *Reduced-1* and *Reduced-2*, which have a decreased number of time periods, from 48 h to 12 h and 2 h respectively (see Figure 8). Although D-Wave finds feasible solutions for Reduced-1, they differ by a factor of 10 from the optimal solution found by Gurobi. For the Reduced-2 instance, some feasible solutions are found with a gap of (approximately) 2.4%. However, while D-Wave takes on average 5s to find the optimal solution, Gurobi finds it in less than a second.



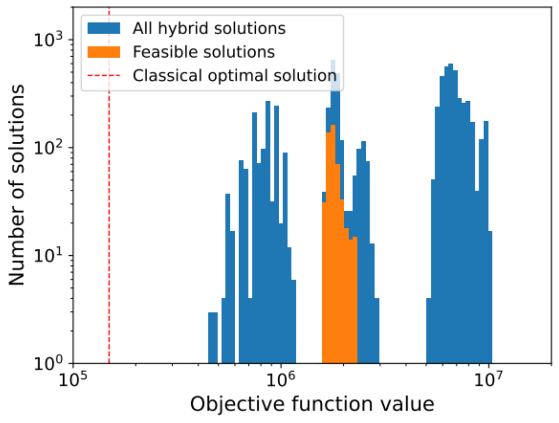


Figure 8: Solution distribution from D-Wave's LeapHybridCQM Solver from *Reduced-1* problem on the left, and the *Reduced-2* problem on the right. Both distributions show the objective function values of 70 runs, with an average of 100 solutions per iteration.

These results suggest that for mixed-integer linear programming (MILP), including the unit commitment problem, D-Wave's solver fails to surpass classical solvers like Gurobi. Solution quality remains significantly lower, and no computational advantage is observed.

5.2 Conclusions

Quantum annealing is a promising approach for solving certain NP-hard and optimization problems, particularly those that align well with its mathematical structure. Continuous improvements in quantum hardware, such as the implementation of topologies like Zephyr or the development of fault-tolerant qubits, are expected to enhance the capabilities of quantum annealers. Advances in hybrid algorithms that better integrate classical and quantum computing resources will likely improve the performance and applicability of quantum annealing.

While quantum annealing shows significant potential for QUBO problems, expanding its applicability to a wider range of optimization problems, including those with non-binary and non-quadratic constraints, remains a key area of research. Regular benchmarking and performance evaluations will be crucial to track progress and identify areas where quantum annealing can offer a computational advantage.

Due to its current limitations in scalability, parameter optimization, and handling complex constraints, quantum annealing is not yet a universal solution. However, it has a promising future with ongoing advancements in hardware and algorithms expected to expand its capabilities and applicability. The field has grown enormously in recent years, and this trend is expected to continue, further enhancing the potential of quantum annealing to solve complex optimization problems.

6. Outreach and Dissemination Activities

Here we summarize some selected highlights regarding dissemination activities as part of Task2.2 work output presented in international conferences, workshops, and scientific publications.

6.1 Dissemination activities on Bender's decomposition – Conferences and publications

The development of the new method in benders decomposition was presented at the 25th International Symposium on Mathematical Programming (ISMP 2024[5]) in Montreal. The ISMP features parallel sessions on optimization topics including nonlinear programming, global optimization, and machine learning applications. There, the results described in Chapter 4 were accepted for presentation under the title:

- Integrated energy system planning under short-term and long-term uncertainty: Modelling and algorithms, presented by *Hongyu Zhang (NTNU)*

This work was extended and developed in a scientific publication submitted to a top international journal under the title:

- Modelling and analysis of multi-timescale uncertainty in energy system planning; co-authored by *Hongyu Zhang, Erlend Heir, Asbjørn Nisi, Asgeir Tomasgard (NTNU)*

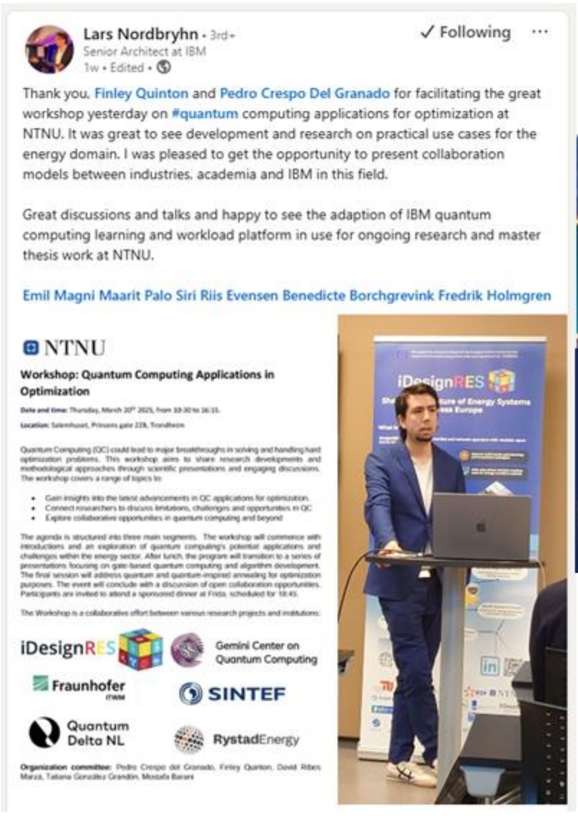
6.2 Quantum Computing Applications in Optimization – Workshop and publications

A workshop about Quantum Computing Applications in Optimization was organized to bring together researchers and industry professionals to discuss the latest advancements and challenges in applying quantum computing to optimization problems with a focus on energy systems. The event attracted participants from academia, research institutions, and companies, including Rystad Energy and IBM, as well as universities from Norway, Finland, Germany and Spain.

The workshop covered such as gate-based quantum computing, quantum annealing, quantum inspired algorithms and challenges in energy modelling, exploring their potential to tackle complex optimization challenges. Through expert talks, research presentations, and interactive discussions, participants shared insights into recent technological developments, fostered collaboration between academia and industry, and identified key challenges and opportunities for future research. A particular focus was placed on quantum applications in the energy sector, highlighting computational challenges and emerging solutions.

The final session was dedicated to open collaboration opportunities, encouraging dialogue between participants and paving the way for future joint research initiatives. The event fostered valuable discussions, strengthened professional networks, and highlighted the growing interest in quantum computing for solving real-world optimization problems.

[5] https://ismp2024.gerad.ca/schedule/TA


Lars Nordbryhn • 3rd+
Senior Architect at IBM
1w • Edited •

✓ Following

Thank you, Finley Quinton and Pedro Crespo Del Granado for facilitating the great workshop yesterday on #quantum computing applications for optimization at NTNU. It was great to see development and research on practical use cases for the energy domain. I was pleased to get the opportunity to present collaboration models between industries, academia and IBM in this field.

Great discussions and talks and happy to see the adaption of IBM quantum computing learning and workload platform in use for ongoing research and master thesis work at NTNU.

Emil Magni Maarit Palo Siri Riis Evensen Benedicte Borchgrevink Fredrik Holmgren

NTNU

Workshop: Quantum Computing Applications in Optimization

Date and time: Thursday, March 20th 2025, from 10:30 to 16:15.

Location: Salenhuset, Prinsens gate 22B, Trondheim

Quantum Computing (QC) could lead to major breakthroughs in solving and handling hard optimization problems. This workshop aims to share research developments and methodological approaches through scientific presentations and engaging discussions. The workshop covers a range of topics to:

- Gain insights into the latest advancements in QC applications for optimization.
- Connect researchers to discuss limitations, challenges and opportunities in QC
- Explore collaborative opportunities in quantum computing and beyond

The agenda is structured into three main segments. The workshop will commence with introductions and an exploration of quantum computing's potential applications and challenges within the energy sector. After lunch, the program will transition to a series of presentations focusing on gate-based quantum computing and algorithm development. The final session will address quantum and quantum-inspired annealing for optimization purposes. The event will conclude with a discussion of open collaboration opportunities. Participants are invited to attend a sponsored dinner at Frida, scheduled for 18:45.

The Workshop is a collaborative effort between various research projects and institutions:

iDesignRES Gemini Center on Quantum Computing Fraunhofer ITWM SINTEF Quantum Delta NL RystadEnergy

Organization committee: Pedro Crespo del Granado, Finley Quinton, David Ribes Marza, Tatiana Gonzalez Grandón, Mostafa Barani

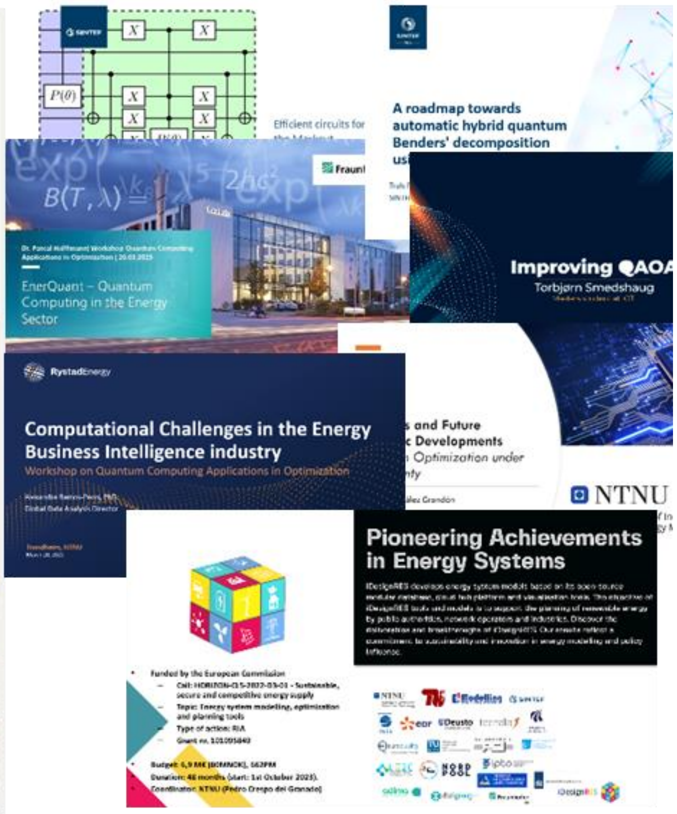

Efficient circuits for
A roadmap towards automatic hybrid quantum Benders' decomposition
EnerQuant – Quantum Computing in the Energy Sector
Improving QAOA
Torbjørn Smedshaug
RystadEnergy
Computational Challenges in the Energy Business Intelligence industry
Workshop on Quantum Computing Applications in Optimization
and Future Developments
Optimization under
NTNU
Pioneering Achievements in Energy Systems
Funded by the European Commission

Figure 9: Workshop organized by NTNU together with iDesignRES. Overview of presentations, social media impact and showcasing the iDesignRES project.

Moreover, in this workshop, the main results described in chapter 5 were presented. This work was extended and developed in a scientific publication published in a top international journal (Nature Scientific's Report) under the title:

- *Quantum annealing versus classical solvers: Applications, challenges and limitations for optimisation problems*, co-authored by Finley Alexander Quinton, Per Arne Sevle Myhr, Mostafa Barani, Pedro Crespo del Granado, Hongyu Zhang

7. Summary

This deliverable summarizes the development within *Task 2.2: Assembling multi-physics modules and components*. It is considered as companion report to the software deliverable developed through two individual packages for the *EnergyModelsX* framework. The two packages are available on GitHub including their documentation and individual examples.

New generic sampling routines for the individual component models are introduced that can be utilized to incorporate data/process descriptions from detailed models. These sampling routines are described within Section 2. As the individual functions are generic, it is necessary to provide a specific technology description for the *EnergyModelsX* framework as outlined in the code for the power generation from wind farms.

The *EnergyModelsX* framework is furthermore extended with a receding (rolling) horizon framework for operational stress testing of an energy system as described in Section 4. The receding horizon framework allows for differing horizon and future value descriptions. It is in general compatible to *EnergyModelsX* models although it requires potentially minor modifications to the input data set. The reduction in computational burden is dependent on the complexity of the problem, the chosen length of the optimization horizon as well as the complete horizon. While it is possible for simple models that the additional overhead for creating the individual models outweighs the benefits offered by smaller optimization problems, this is reversed for complex models.

Although a receding horizon approach can reduce the computational burden through splitting the complete horizon in several smaller horizons and solving these sequentially, it is still necessary to consider approaches for improving the speed of the optimization. To this end, implemented features for model constructions (Section 4.1), methods for improving the computation speed in solving the optimization problem (Section 4.2) as well as an outlook for potential applications of quantum computing (Section 0) are presented to allow the reader to obtain an overview of the current and future potential for reducing the computational burden of solving energy system models.

The developed *EMX* packages will be subsequently utilized within the individual iDesignRES case studies in WP3, specifically in the upcoming case study focusing on the North Sea region in Task 3.2. It is planned to present the framework, including a smaller case study to highlight its benefits, at the Smart Energy Systems Conference in September 2025.

8. References

[1] Gunhild A. Reigstad, “Integrated Design of the Components of the Energy System to Plan the Uptake of Renewable Energy Sources: An Open Source Toolbox,” iDesignRES, Companion report D1.2, 2024.

[2] L. Hellemo, E. F. Bødal, S. E. Holm, D. Pinel, and J. Straus, “EnergyModelsX: Flexible Energy Systems Modelling withMultiple Dispatch,” *J. Open Source Softw.*, vol. 9, no. 97, p. 6619, May 2024, doi: 10.21105/joss.06619.

[3] I. Dunning, J. Huchette, and M. Lubin, “JuMP: A Modeling Language for Mathematical Optimization,” *SIAM Rev.*, vol. 59, no. 2, pp. 295–320, Jan. 2017, doi: 10.1137/15M1020575.

[4] Tecnalia, “iDesignRES/Tecnalia,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/iDesignRES/Tecnalia

[5] Deusto, “DeustoTech/IDR-IIsim: Repository with industry models tailored to the integration on large scale energy system models and circular economy assessment,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/DeustoTech/IDR-IIsim

[6] A. Golab, “antoniamgolab/iDesignRES_transcompmodel,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/antoniamgolab/iDesignRES_transcompmodel

[7] S. E. Holm, “EnergyModelsX/EnergyModelsRenewableProducers.jl: Package for extending EnergyModelsX with descriptions of renewable electricity technologies.,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/EnergyModelsX/EnergyModelsRenewableProducers.jl

[8] J. Straus, “EnergyModelsX/EnergyModelsHydrogen.jl: Package for extending EnergyModelsX with descriptions of hydrogen technologies.,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/EnergyModelsX/EnergyModelsHydrogen.jl

[9] S. E. Holm, “EnergyModelsX/EnergyModelsCO2.jl: Package for extending EnergyModelsX with descriptions of CO2 capture and storage technologies.,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/EnergyModelsX/EnergyModelsCO2.jl

[10] L. Hellemo, “EnergyModelsX/EnergyModelsHeat.jl: Package for extending EnergyModelsX with technology descriptions within the heat sector.,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/EnergyModelsX/EnergyModelsHeat.jl

[11] E. F. Bødal, “EnergyModelsX/EnergyModelsGeography.jl: Package for extending EnergyModelsX with geographical features.,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/EnergyModelsX/EnergyModelsGeography.jl

[12] EDF, “SMS++ / The SMS＋＋ Project · GitLab,” GitLab. Accessed: Mar. 18, 2025. [Online]. Available: https://gitlab.com/smspp/smspp-project

[13] G. del A. Serrano, “EnergyModelsHeat.jl/submodels/bioCHP_plant at main · EnergyModelsX/EnergyModelsHeat.jl,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/EnergyModelsX/EnergyModelsHeat.jl/tree/main/submodels/bioCHP_plant

[14] d G. Svendsen, “Harald G Svendsen / wind_power_timeseries · GitLab,” GitLab. Accessed: Mar. 18, 2025. [Online]. Available: https://gitlab.sintef.no/harald.svendsen/timeseries

[15] B. Janssens, “JuliaInterop/CxxWrap.jl: Package to make C++ libraries available in Julia,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/JuliaInterop/CxxWrap.jl

[16] S. G. Johnson, “JuliaPy/PyCall.jl: Package to call Python functions from the Julia language,” GitHub. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/JuliaPy/PyCall.jl

[17] Anaconda Software Distribution, “Conda Documentation — conda 25.1.1 documentation.” Accessed: Mar. 18, 2025. [Online]. Available: https://docs.conda.io/projects/conda/en/stable/

[18] S. Eustace, “Poetry - Python dependency management and packaging made easy.” Accessed: Mar. 18, 2025. [Online]. Available: https://python-poetry.org/

[19] B. Janssens, *JuliaInterop/libcxxwrap-julia*. (Feb. 22, 2025). C++. JuliaInterop. Accessed: Mar. 18, 2025. [Online]. Available: https://github.com/JuliaInterop/libcxxwrap-julia

[20] J. Mattingley, Y. Wang, and S. Boyd, “Receding Horizon Control,” *IEEE Control Syst.*, vol. 31, no. 3, pp. 52–65, Jun. 2011, doi: 10.1109/MCS.2011.940571.

[21] F. Gulotta, P. Crespo Del Granado, P. Pisciella, D. Siface, and D. Falabretti, “Short-term uncertainty in the dispatch of energy resources for VPP: A novel rolling horizon model based on stochastic programming,” *Int. J. Electr. Power Energy Syst.*, vol. 153, p. 109355, Nov. 2023, doi: 10.1016/j.ijepes.2023.109355.

[22] P. Aaslid, M. Korpås, M. M. Belsnes, and O. B. Fosso, “Pricing electricity in constrained networks dominated by stochastic renewable generation and electric energy storage,” *Electr. Power Syst. Res.*, vol. 197, p. 107169, Aug. 2021, doi: 10.1016/j.epsr.2021.107169.

[23] M. Bouchet-Valat and B. Kamiński, “**DataFrames.jl** : Flexible and Fast Tabular Data in *Julia*,” *J. Stat. Softw.*, vol. 107, no. 4, 2023, doi: 10.18637/jss.v107.i04.

[24] M. Kaut, K. T. Midthun, A. S. Werner, A. Tomasgard, L. Hellemo, and M. Fodstad, “Multi-horizon stochastic programming,” *Comput. Manag. Sci.*, vol. 11, no. 1–2, pp. 179–193, Jan. 2014, doi: 10.1007/s10287-013-0182-6.

[25] C. L. Lara, J. D. Siirola, and I. E. Grossmann, “Electric power infrastructure planning under uncertainty: stochastic dual dynamic integer programming (SDDiP) and parallelization scheme,” *Optim. Eng.*, vol. 21, no. 4, pp. 1243–1281, Dec. 2020, doi: 10.1007/s11081-019-09471-0.

[26] S. Hummelen, E. Hordvei, M. Petersen, S. Backe, H. Zhang, and P. Crespo Del Granado, “Exploring European Hydrogen Demand Variations Under Tactical Uncertainty with Seasonal Hydrogen Storage,” 2024, *SSRN*. doi: 10.2139/ssrn.4879862.

[27] F. V. Louveaux, “Multistage stochastic programs with block-separable recourse,” in *Stochastic Programming 84 Part II*, vol. 28, A. Prékopa and R. J.-B. Wets, Eds., in Mathematical Programming Studies, vol. 28. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 48–62. doi: 10.1007/BFb0121125.

[28] H. Zhang, I. E. Grossmann, and A. Tomasgard, “Decomposition methods for multi-horizon stochastic programming,” *Comput. Manag. Sci.*, vol. 21, no. 1, p. 32, Jun. 2024, doi: 10.1007/s10287-024-00509-y.

[29] H. Zhang, N. Mazzi, K. McKinnon, R. G. Nava, and A. Tomasgard, “A stabilised Benders decomposition with adaptive oracles for large-scale stochastic programming with short-term and long-term uncertainty,” *Comput. Oper. Res.*, vol. 167, p. 106665, Jul. 2024, doi: 10.1016/j.cor.2024.106665.

[30] H. Zhang, I. E. Grossmann, K. McKinnon, B. R. Knudsen, R. G. Nava, and A. Tomasgard, “Integrated investment, retrofit and abandonment energy system planning with multi-timescale uncertainty using stabilised adaptive Benders decomposition,” 2023, *arXiv*. doi: 10.48550/ARXIV.2303.09927.

[31] A. Reiten and M. E. Mikkelsen, “A Stochastic Capacity Expansion Model for the European Power System - Using a Distributed Progressive Hedging Approach to Handle Long-Term Uncertainty,” Master’s thesis, Norwegian University of Science and Technology, Trondheim, 2018. [Online]. Available: http://hdl.handle.net/11250/2577471

[32] E. Heir and A. Nisi, “Navigating Uncertainty - Stochastic Energy Planning in a European Context,” Master’s thesis, Norwegian University of Science and Technology, Trondheim, 2024. [Online]. Available: https://hdl.handle.net/11250/3156158

[33] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, “Julia: A Fresh Approach to Numerical Computing,” *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, Jan. 2017, doi: 10.1137/141000671.

[34] *Gurobi*. (2022). Gurobi Optimization, LLC. [Online]. Available: https://www.gurobi.com

iDesignRES

Integrated Design of the Components of the Energy System to Plan the Up-take of Renewable Energy Sources: An Open-Source Toolbox

More information on iDesignRES project:

https://www.linkedin.com/company/idesignres

https://idesignres.eu/

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.