COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Gestión óptima de una batería de ion de litio mediante la formulación Zig-Zag para aproximar las funciones de pérdidas no lineales

Autor: Salvador Guerrero García

Director: Javier García González

Madrid

Julio de 2024

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Gestión óptima de una batería de ion de litio mediante la formulación Zig-Zag para aproximar las funciones de pérdidas no lineales

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2023/2024 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Salvador Guerrero García Fecha: 23/06/2024

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Javier Garcia Gonzalez

Firmado digitalmente por Javier Garcia Gonzalez
Fecha: 2024.07.03 10:22:44 +02'00'

Fdo.: Javier García González Fecha: ……/ ……/ ……

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Gestión óptima de una batería de ion de litio mediante la formulación Zig-Zag para aproximar las funciones de pérdidas no lineales

Autor: Salvador Guerrero García

Director: Javier García González

Madrid

Julio de 2024

GESTIÓN ÓPTIMA DE UNA BATERÍA DE ION DE LITIO MEDIANTE LA FORMULACIÓN ZIG-ZAG PARA APROXIMAR LAS FUNCIONES DE PERDIDAS NO LINEALES.

Autor: Guerrero García, Salvador.
Director: García González, Javier.
Entidad Colaboradora: Instituto de Investigación Tecnológica.

RESUMEN DEL PROYECTO

1 Introducción

Se espera que los sistemas de almacenamiento eléctricos se conviertan en un elemento clave para compensar la volatilidad e incertidumbre de las energías renovables. Dentro de los sistemas de almacenamiento, destacan las baterías de ion de litio por su versatilidad y coste competitivo. Además, estas baterías son clave para la electrificación del sector automovilístico, actualmente están presentes en la mayoría de vehículos eléctricos y su cuota de mercado no se espera que disminuya en el futuro.

Por lo tanto, las baterías de ion de litio son consideradas elementos fundamentales para la descarbonización de los sistemas eléctricos y el sector automovilístico. Debido a ello, el desarrollo de modelos precisos de esta tecnología es clave para su integración y operación óptima.

Este trabajo se centra en el modelado de las baterías de ion de litio en el contexto de una microrred compuesta por un panel fotovoltaico, una batería y un generador diesel. Se propone el uso de la técnica ZigZagInteger para la linealización de la función de perdidas de la batería, junto a dos métodos para modelar el funcionamiento de la misma. En el caso

de estudio se compara esta técnica frente a otras alternativas, haciendo especial hincapié en los tiempos de resolución.

2 Metodología

2.1 Modelo de baterías de Ion de Litio

El modelo usado para caracterizar las baterías de Ion de Litio está basado en [4], donde el profesor Javier García González propone una modificación al modelo descrito en [5]. Para el contexto de análisis de sistemas estáticos, las perdidas de energía durante el ciclo de descarga y carga de este tipo de baterías son caracterizadas por funciones bivariables dependientes del estado de carga de la batería y la potencia de descarga/carga.

Debido la relación no lineal que presentan ambas expresiones, su implementación en modelos de optimización clásica se hace complicada y poco práctica, ya que los solucionadores actuales todavía no permiten la optimización no lineal a gran escala. Por lo que el uso de técnicas de linearización es necesario.

2.2 Zig-Zag Integer formulation

El enfoque más popular para linealizar una función matemática genérica en el contexto de la optimización clásica es su descomposición en funciones lineales a tramos. Estas técnicas aproximan la función matemática como la combinación convexa de un conjunto de puntos de la función no lineal original. La selección de los distintos tramos de la función linealizada suele modelarse con variables binarias. Por ejemplo, la formulación clásica para una función no lineal bivariante, como la expresión de las pérdidas de Li-Ion, se presenta en [1]. La principal limitación de estos enfoques es el elevado número de variables binarias necesarias, que hacen que los problemas de programación entera mixta sean muy difíciles de resolver con los solucionadores matemáticos actuales.

[3] propone la formulación Zig-Zag Integer como modelo prometedor para la linealizacion de funciones, reduciendo drásticamente la compleji-

dad matemática en comparación con métodos alternativos.

3 Modelo de la microred

La microrred implementada consta de un generador diésel, un panel fotovoltaico, una batería de iones de litio y una demanda dependiente del tiempo. Su caracterización procede de [2].

El modelo de gestión propuesto se basa en un modelo de optimización determinista clásico que busca minimizar los costes de operación, sujeto a un conjunto de restricciones. Una de las restricciones modeladas son las pérdidas de los ciclos de carga y descarga de las baterías Li-Ion. Combinando la expresión de pérdidas de [4] y diferentes técnicas de linealización como el Zig-Zag Integer. En cuanto al modelo de la batería, se proponen dos alternativas, un enfoque directo en el que las pérdidas de carga y descarga de la batería se modelan como funciones separadas, denominado "enfoque separado" y un modelo alternativo en el que ambas funciones se combinan, denominado "enfoque combinado", que conduce a un menor número de variables discretas y reduce potencialmente la carga computacional.

4 Caso de estudio

Con el fin de evaluar la idoneidad de la técnica del Zig-Zag Integer para el modelado de las pérdidas de las baterías Li-Ion, se estudiaron diferentes escenarios, que representaban el verano y el invierno. Para comprobar el rendimiento y la escalabilidad de las técnicas de linealización, se analizaron diferentes granularidades y diferentes patrones de triangularización.

En cuanto al rendimiento de los modelos comparados, podemos concluir que el patrón de triangulación afecta considerablemente al tiempo de convergencia. El método Zig-Zag Integer con triangulación J1 (ZZI-J1) es superior al resto de métodos estudiados, en cambio, con triangulación K1 (ZZI-K1) el rendimiento es equivalente al método BM con triangu-

lación J1, cuando en teoría éste necesita más variables discretas y, por tanto, se esperaba un peor rendimiento.

En los casos con una granularidad de 16x16 el método ZZI-J1 es el único modelo que Gurobi es capaz de resolver en menos de 1 hora. Lo que demuestra la superioridad de este método para casos con granularidad alta.

Por último, al analizar los enfoques "enfoque combinado" y "enfoque separado" para la modelización, los resultados de la batería son contraintuitivos. A pesar de la reducción de variables discretas que presenta el modelo, los tiempos de convergencia son generalmente peores.

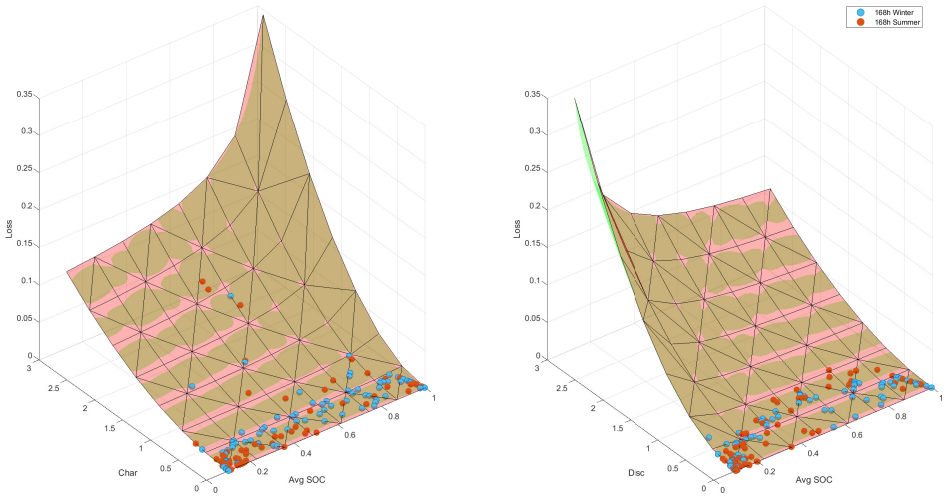
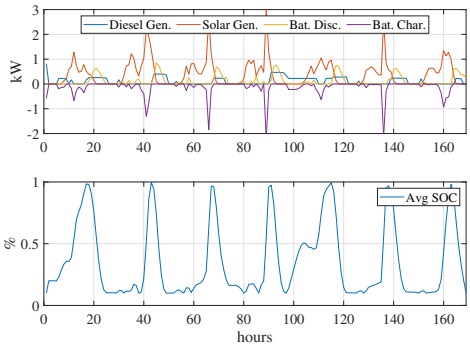


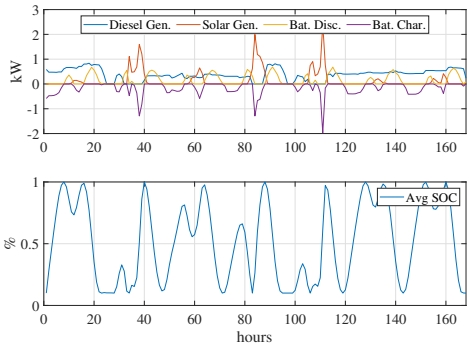
Figure 1: Puntos de operación de la batería para el caso (ZZI-J1 8×8).

Table 1: Rendimiento Computacional

Caso	Granularidad	Método	Optimizador	Obj.[€]	Tiempo[s]	Rel.Gap
168h_W	8x8	ZZI-J1-Sep	gurobi	19.1052	40.83	0.38%
168h_W		ZZI-K1-Sep		19.1188	272.77	0.48%
168h_W		C-K1-Sep		19.1277	3150.87	0.49%
168h_W		BM-J1-Sep		19.1022	94.33	0.38%
168h_W		BM-K1-Sep		19.1270	333.26	0.50%
168h_W	16x16	ZZI-J1-Sep	gurobi	19.0596	224.11	0.26%
168h_W		ZZI-K1-Sep		-	-	-
168h_W		C-K1-Sep		-	-	-
168h_W		BM-J1-Sep		-	-	-
168h_W		BM-K1-Sep		-	-	-
168h_S	8x8	ZZI-J1-Sep	gurobi	4.8732	133.61	0.29%
168h_S		ZZI-K1-Sep		-	-	-
168h_S		C-K1-Sep		4.8806	374.41	0.43%
168h_S		BM-J1-Sep		4.8777	97.32	0.40%
168h_S		BM-K1-Sep		4.8755	257.80	0.33%
168h_S	16x16	ZZI-J1-Sep	gurobi	4.8712	267.63	0.45%
168h_S		ZZI-K1-Sep		-	-	-
168h_S		C-K1-Sep		-	-	-
168h_S		BM-J1-Sep		-	-	-
168h_S		BM-K1-Sep		-	-	-
168h_W	8x15	ZZI-J1-Com	gurobi	19.070269	61.234	0.20%
168h_W		BM-J1-Com		19.103467	109.031	0.37%
168h_S	16x31	ZZI-J1-Com	gurobi	19.081705	334.625	0.37%
168h_S		BM-J1-Com		-	-	-



(a) Operación horaria (ZZI-J1 8×8, Verano).



(b) Operación horaria (ZZI-J1 8×8, Invierno).

5 Conclusiones

Este trabajo estudia la aplicación del método de linealización Zig-Zag Integer presentado en [3] para modelar las funciones de pérdidas de una batería de iones de litio en el contexto de una microrred. Se puede concluir que el método de linealización Zig-Zag Integer emparejado con la triangulación J1 presenta una considerable mejora de tiempo, mientras que mantiene con precisión un bajo error. En cuanto al método de "enfoque combinado" propuesto para modelizar las baterías, los resultados no son concluyentes y es necesario un análisis más exhaustivo.

6 Referencias

[1] Djangir A. Babayev. “Piece-wise linear approximation of functions of two variables”. In: *Journal of Heuristics* 2.4 (1997), pp. 313–320. DOI: 10.1007/bf00132502. URL: https://doi.org/10.1007/bf00132502.

[2] David Domínguez-Barbero, Javier García-González, and Miguel Á. Sanz-Bobi. “Twin-delayed deep deterministic policy gradient algorithm for the energy management of microgrids”. In: *Engineering Applications of Artificial Intelligence* 125 (2023), p. 106693. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.106693. URL: https://www.sciencedirect.com/science/article/pii/S0952197623008771.

[3] Joey Huchette and Juan Pablo Vielma. *Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools*. 2017. DOI: 10.48550/ARXIV.1708.00050. URL: https://arxiv.org/abs/1708.00050.

[4] Salvador Guerrero García Javier García-González. “Optimal Management of A Microgrid Li-Ion Battery Considering Non-Linear Losses Using The Integer Zig-Zag Formulation”. In: *PSCC* (2024). URL: https://pscc.epfl.ch/modules/request.php?module=oc_proceedings&action=view.php&id=643&file=1/643.pdf&a=Accept.

[5] Olivier Tremblay and Louis-A. Dessaint. “Experimental Validation of a Battery Dynamic Model for EV Applications”. en. In: *World Electric Vehicle Journal* 3.2 (June 2009), pp. 289–298. ISSN: 2032-6653. DOI: 10.3390/wevj3020289. URL: https://www.mdpi.com/2032-6653/3/2/289 (visited on 03/28/2022).

OPTIMAL MANAGEMENT OF A MICROGRID LI-ION BATTERY CONSIDERING NON-LINEAR LOSSES USING THE INTEGER ZIG-ZAG FORMULATION.

Author: Salvador Guerrero García.
Director: Javier García González.
Collaborating Entity: Instituto de Investigación Tecnológica.

PROJECT BRIEF

1 Introduction

Electric storage systems are expected to become a key element to offset the volatility and uncertainty of renewable energies. Within storage systems, lithium-ion batteries stand out for their versatility and competitive cost. Moreover, these batteries are key to the electrification of the automotive sector; they are currently present in most electric vehicles and their market share is not expected to decrease in the future. Therefore, lithium-ion batteries are considered fundamental elements for the decarbonization of electrical systems and the automotive sector. Because of this, the development of accurate models of this technology is key to its optimal integration and operation.

This work focuses on the modeling of lithium-ion batteries in the context of a microgrid composed of a photovoltaic panel, a battery and a diesel generator. The use of the Zig-Zag Integer technique for the linearization of the battery loss function is proposed, together with two methods for modeling the battery operation. In the case study, this technique is compared with other alternatives, with special emphasis on the resolution times.

2 Methodology

2.1 Li-Ion battery modelling

The model used to characterize Li-Ion batteries is based on [4], where Professor Javier García González proposes a modification to the model described in [6]. For the context of static system's analysis, the proposed formulation for the losses of the batteries during the discharge and discharge cycles is a bivariate non-linear function dependent on the battery state of charge and the discharge/charge power.

Due to the nonlinear relationship presented by this model, its implementation in classical optimization models becomes complicated and impractical, since current solvers do not yet allow large-scale nonlinear optimization. This makes the use of linearization techniques such as Zig-Zag Integer an imperative.

2.2 Zig-Zag Integer formulation

The most popular approach to linearize a generic mathematical function in the context of classical optimization is its decomposition in pice-wise linear functions (pwlf). These techniques approximate the mathematical function as the convex combination of a set of points from the original non-linear function. The selection of the different pieces of the pwlf is usually modeled with binary variables. For example, the classical formulation for bivariate non-linear function, such as the Li-Ion losses expression, was presented in [1]. The main limitation of these approaches is the high number of binary variables needed, which make the mixed integer programming problems very difficult to solve with current mathematical solvers.

[3] propose the Zig-Zag Integer formulation as promising model applicable to pwlf that reduces drastically the mathematical complexity compared to the alternatives.

2.3 Microgrid model

The implemented microgrid consists of a diesel generator, a photovoltaic panel, a lithium-ion battery and a time-dependent demand. Its characterization was sourced from [2].

The proposed management model is based on a classical deterministic optimization model which seeks to minimize the operating costs, subject to a set of constraints. One of the modeled constraints is the losses of the Li-Ion battery charge and discharge cycles. Combining the loss expression from [4] and different linearization techniques such as the Zig-Zag Integer. Regarding the model of the battery, two alternatives are proposed, a direct approach where the charge and discharge losses of the battery are modeled as separate functions, referred to as the "separate approach" and an alternative model were both functions are combined, referred to as the "combined approach" which leads to lower number of discrete variables and potentially reducing the computational burden.

3 Results

In order to assess the suitability of the Zig-Zag Integer technique to the modeling of the Li-Ion battery losses, different scenarios were study, representing summer and winter. To test the performance and scalability of the linearization techniques, different granularity and different triangularization patterns were analyzed (J1 y K1). The different linearization methods studied were the Zig-Zag Integer and the techniques proposed in [1] and [5] referred as C-K1 and BM.

Regarding the performance of the compared models, we can conclude that the triangulation pattern considerably affects the convergence time. The Zig-Zag Integer method with J1 triangulation (ZZI-J1) is superior to the rest of the methods studied, on the other hand, with K1 triangulation (ZZI-K1) the performance is equivalent than the BM method with J1 triangulation, when in theory this one needs more discrete variables and, therefore, a worse performance was expected.

In cases with a granularity of 16x16 the ZZI-J1 method is the only model that Gurobi is able to solve in less than 1 hour. Showing the superiority of this method for cases with high granularity.

Finally, when analyzing the "combined approach" and "separate approach" approaches for modeling, the battery results are counterintuitive. Despite the reduction of discrete variables that the model presents, the convergence times are generally worse.

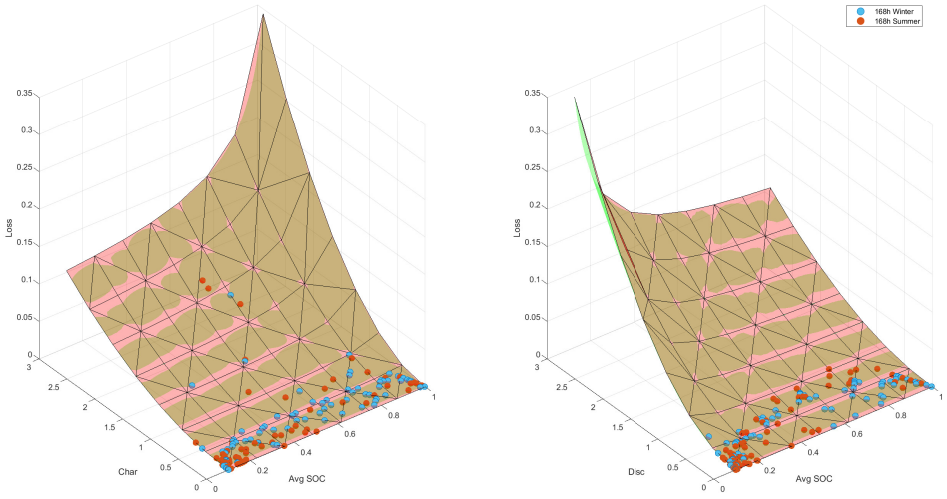
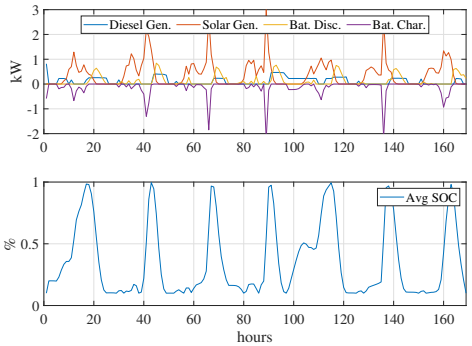
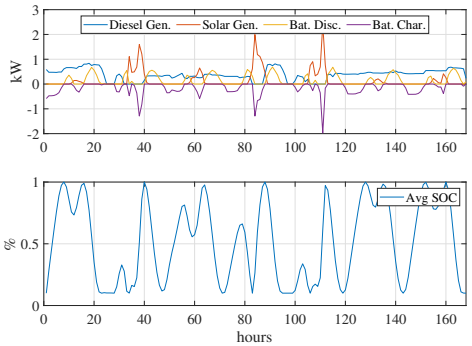


Figure 1: Battery operating points for the case: (ZZI-J1 8×8).



(a) Hourly operation (ZZI-J1 8×8, Summer).



(b) Hourly operation (ZZI-J1 8×8, Winter).

Table 1: Computational results

Case	Granularity	Linearization method	Solver	Obj.[€]	Solving Time[s]	Rel.Gap
168h_W	8x8	ZZI-J1-Sep	gurobi	19.1052	40.83	0.38%
168h_W		ZZI-K1-Sep		19.1188	272.77	0.48%
168h_W		C-K1-Sep		19.1277	3150.87	0.49%
168h_W		BM-J1-Sep		19.1022	94.33	0.38%
168h_W		BM-K1-Sep		19.1270	333.26	0.50%
168h_W	16x16	ZZI-J1-Sep	gurobi	19.0596	224.11	0.26%
168h_W		ZZI-K1-Sep		-	-	-
168h_W		C-K1-Sep		-	-	-
168h_W		BM-J1-Sep		-	-	-
168h_W		BM-K1-Sep		-	-	-
168h_S	8x8	ZZI-J1-Sep	gurobi	4.8732	133.61	0.29%
168h_S		ZZI-K1-Sep		-	-	-
168h_S		C-K1-Sep		4.8806	374.41	0.43%
168h_S		BM-J1-Sep		4.8777	97.32	0.40%
168h_S		BM-K1-Sep		4.8755	257.80	0.33%
168h_S	16x16	ZZI-J1-Sep	gurobi	4.8712	267.63	0.45%
168h_S		ZZI-K1-Sep		-	-	-
168h_S		C-K1-Sep		-	-	-
168h_S		BM-J1-Sep		-	-	-
168h_S		BM-K1-Sep		-	-	-
168h_W	8x15	ZZI-J1-Com	gurobi	19.070269	61.234	0.20%
168h_W		BM-J1-Com		19.103467	109.031	0.37%
168h_S	16x31	ZZI-J1-Com	gurobi	19.081705	334.625	0.37%
168h_S		BM-J1-Com		-	-	-

4 Conclusions

This paper studies the application of the Zig-Zag Integer linearization method presented in [3] for modeling the loss functions of a Lithium-ion battery in a context of a microgrid. It can be concluded that the Zig-Zag Integer linearization method pair with the J1 triangulation presents a considerable time improvement, while accurately maintaining a low error. Regarding the proposed "combined approach" method to model the batteries, the results are not conclusive and a more exhaustive analysis is necessary.

5 Bibliography

[1] Djangir A. Babayev. “Piece-wise linear approximation of functions of two variables”. In: *Journal of Heuristics* 2.4 (1997), pp. 313–320. DOI: 10.1007/bf00132502. URL: https://doi.org/10.1007/bf00132502.

[2] David Domínguez-Barbero, Javier García-González, and Miguel Á. Sanz-Bobi. "Twin-delayed deep deterministic policy gradient algorithm for the energy management of microgrids". In: *Engineering Applications of Artificial Intelligence* 125 (2023), p. 106693. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.106693. URL: https://www.sciencedirect.com/science/article/pii/S0952197623008771.

[3] Joey Huchette and Juan Pablo Vielma. *Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools*. 2017. DOI: 10.48550/ARXIV.1708.00050. URL: https://arxiv.org/abs/1708.00050.

[4] Salvador Guerrero García Javier García-González. "Optimal Management of A Microgrid Li-Ion Battery Considering Non-Linear Losses Using The Integer Zig-Zag Formulation". In: *PSCC* (2024). URL: https://pscc.epfl.ch/modules/request.php?module=oc_proceedings&action=view.php&id=643&file=1/643.pdf&a=Accept.

[5] Shoudong Zhu Tianhong Tan. *Cornell University Computational Optimization Open Textbook: Piecewise linear approximation*. URL: https://optimization.cbe.cornell.edu/index.php?title=Piecewise_linear_approximation.

[6] Olivier Tremblay and Louis-A. Dessaint. "Experimental Validation of a Battery Dynamic Model for EV Applications". en. In: *World Electric Vehicle Journal* 3.2 (June 2009), pp. 289–298. ISSN: 2032-6653. DOI: 10.3390/wevj3020289. URL: https://www.mdpi.com/2032-6653/3/2/289 (visited on 03/28/2022).

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Electric power is distinctive among energy sources due to several defining characteristics. Among these, the most critical is the instantaneous balance of supply and demand. Power system operators must match demand with generation in real time, mainly by adjusting the production of generators. This requirement necessitates a diverse mix of generation technologies to ensure the capacity to ramp production up or down while maintaining a cost-effective and reliable dispatch.

The integration of renewable energy sources into the energy mix presents new challenges. In contrast to traditional dispatchable energy sources such as gas and coal, the majority of renewable sources are non-dispatchable, relying on meteorological stochastic factors for their production. As the proportion of renewable energy increases, the system's capacity to follow demand decreases, leading to both technical and economic issues.

Storage solutions, particularly lithium-ion (Li-ion) batteries, are anticipated to facilitate a higher proportion of renewable energy by providing the dispatchable capacity that renewables sources are not able to provide. Storage technologies would facilitate arbitrage of power, whereby energy would be stored during periods of high renewable energy availability and released during periods of low availability. Combined withe hydropower, li-ion batteries are anticipated to provide the future dispatchable capacity, particularly from 2030 onwards, in alignment with net-zero CO_2 emission targets by 2050 [1]. This is illustrated in figure 1.1.

Given the ambitious renewable energy targets set by many Western countries, power systems are expected to incorporate a significant proportion of renewable energy in the near future. For example, in 2021, 23% of the European Union's

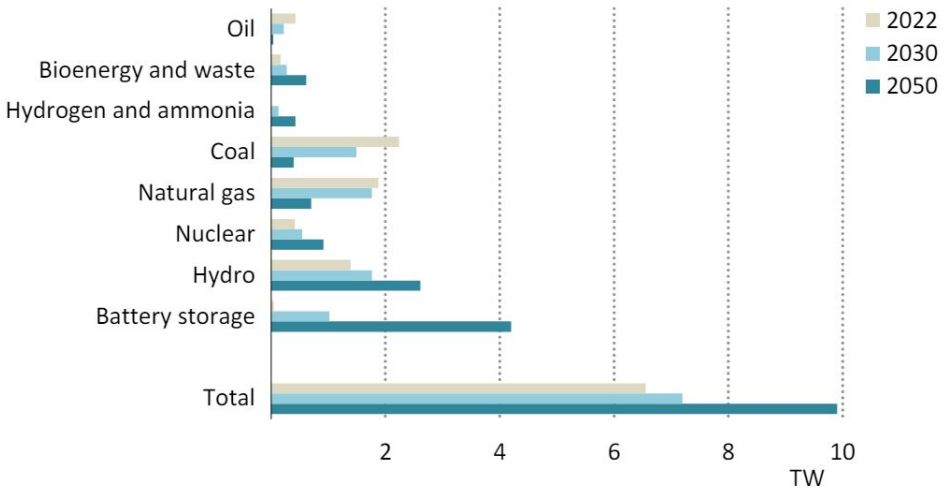


Figure 1.1: Dispatchable power capacity by technology in the Net Zero Emission Scenario, 2022, 2030 and 2050 according to the IEA [1].

electrical energy consumption was derived from renewable sources, with the target for 2030 having increased from 32% in 2018 to a minimum of 42.5% [5]. Consequently, the development of effective power storage solutions has become a crucial necessity for future power grids.

The significance of power storage extends beyond the power sector. Its importance is magnified in other industries. For instance, the transport sector is regarded as the primary contributor to CO_2 emissions. Electric vehicles (EVs) will assume two roles in the decarbonisation agenda. They will directly replace internal combustion vehicles with electric power motors and, in a secondary role, facilitate the decarbonisation of the power sector. The integration of EVs with smart chargers could be employed as a distributed battery in modern power systems, as well as a smart demand that could be dynamically adapted to the prices of electricity. This would mitigate the stochastic effect of renewable energies.

When analyzing various power storage technologies, several alternatives are available. The oldest form is pumped storage hydropower (PSH), which dates back to 1904 in Switzerland. This technology exploits the potential energy stored in water at different heights in reservoirs connected by a pump and a turbine. PSH offers rapid response capacity without the potential generation of CO_2 emissions. However, this technology has significant limitations, including high infrastructure costs and location constraints, as it requires sites with sufficient elevation difference and reservoir capacity.

A less common form of energy storage is Compressed Air Energy Storage (CAES). These systems store energy by compressing air and storing it in un-

derground caverns or large tanks. During periods of excess energy production, typically from renewable sources like wind or solar, air is compressed and stored under high pressure. When energy is needed, the compressed air is released, heated, and expanded through turbines to generate electricity. CAES systems are capable of providing large-scale energy storage, which helps to balance supply and demand on the electrical grid. They are valued for their ability to store energy over long durations and to rapidly dispatch power when required. However, as in the case of PHP, the main limitation of this technology is its high infrastructure costs and location limitations.

Redox flow batteries (RFB) are a type of rechargeable battery where energy is stored in chemical form, in liquid electrolytes contained in tanks. The electrolytes, which typically consist of vanadium or other metals dissolved in solution, are pumped through an electrochemical cell that converts chemical energy into electrical energy. The unique design allows for the separation of energy storage and power generation, offering scalability in terms of both capacity and output. This makes redox flow batteries particularly suitable for large-scale energy storage applications, such as grid stabilisation and renewable energy integration, due to their long cycle life, quick response times, and the ability to be easily recharged by replacing the electrolyte.

PSH, CAES and RFB are limited to stationary industrial scale use cases and therefore not suitable for the transportation sector or other small-scale uses. Other forms of storage, such as lead-acid or lithium-based batteries, offer the versatility to be used both for stationary and mobile applications.

Lead-acid batteries are a storage form of energy based on chemistry that is widely used. They are used for small-scale applications compared to CAES or PSH. Used for powering small electric vehicles (e.g. golf carts, forklifts) and in redundancy systems (e.g. UPS). Due to its low energy density and shorter cycle life compared to newer battery technologies, its application is limited. However, it requires lower infrastructure costs compared to other technologies.

Lithium-based batteries operate under the same principle as lead-acid batteries but present higher energy density and a longer cycle life. Their capacity to provide rapid response in conjunction with their modularity renders them suitable for power systems. Furthermore, their high energy density and suitability for mobile applications while maintaining a competitive cost have made them the dominant technology in the transport sector. Currently, over 90% of all electric vehicles are powered by Li-Ion batteries [2].

Furthermore, hydrogen is regarded as a prospective alternative for the storage of power. The conversion of electricity to hydrogen can be achieved through electrolysis, with the subsequent conversion of hydrogen back to electricity be-

ing possible through the use of hydrogen cells. Alternatively, hydrogen can be utilized directly as a fuel.

In order to assess the suitability of the different storage technologies available, both technical and economic factors must be taken into account. [16] examines the competitiveness of the different technologies across various use cases within the power grid infrastructure. The study predicts that the economic viability of Li-Ion technologies will be strengthened in the future, particularly as power systems adopt higher levels of renewable energy generation.

The report posits that Li-Ion batteries and hydrogen storage will emerge as dominant technologies, displacing CAES from the market and potentially diminishing the role of hydropower. Li-Ion batteries are expected to excel in short-term storage applications, characterized by capacities under 16 hours and frequent discharge cycles. They will play a pivotal role in exploiting the volatility of solar and wind generation. Conversely, hydrogen storage is anticipated to be particularly suited to long-term requirements, mitigating the seasonality of demand to derive value.

For these reasons, understanding the behavior of Li-Ion based batteries and developing control algorithms that accurately model this technology is key to harness its full potential. Both for optimal operation and expansion of power systems.

1.2 Objectives

This master thesis centers on the modeling of Li-Ion batteries. The characterization of these batteries is an area of research with great interest, both for the importance of this technology and the lack of consensus on how to model them. The focus will be placed on considering the non-linear behavior incurred during the charge and discharge cycles of Li-Ion batteries. Assessing if the promising Zig-Zag Integer linearization technique is applicable to the Li-Ion charge and discharge losses.

The objectives of this master thesis can be summarized as follows:

- To deepen the understanding of the Zig-Zag formulation: One of the key objectives of this research endeavor is to foster a comprehensive understanding of the Zig-Zag method to enable its integration into any algebraic modeling software, such as GAMS. This adaptability will empower re-

searchers to exercise precise control over the model, facilitating fine-tuning to better align it with their specific requirements.

- Develop the mathematical formulation adapted to the optimal management of a Li-Ion battery: The second aim of this research pertains to utilizing the Zig-Zag method for linearizing the charge and discharge battery losses in the modeling of a microgrid. This involves refining the model to ensure both efficient and accurate problem-solving. Furthermore, the research holds the potential to surpass its predefined objectives if it manages to introduce novel enhancements to the existing state-of-the-art methodologies.

- Develop a prototype in GAMS to analyze the performance of the method: The Zig-Zag method can be readily employed in the Julia programming language, thanks to a dedicated library offered by the Zig-Zag method authors. The last objective is focused on the implementation of the optimal management model in GAMS, and its verification using the off-the-self implementation of the ZigZag method that Julia Lang offers as a reference.

1.3 Organization of the document

To present the results of the research project, this document is divided into 4 chapters, each focused on different areas that, when combined, lead to the research objectives.

Chapter 2 explores the literature regarding the characterization of Li-Ion batteries. The chapter concludes with two bivariate, nonlinear expressions that model the operational losses of these batteries.

Given the mathematical complexities involved in optimizing nonlinear functions, chapter 3 explores the state-of-the-art in linearization techniques for bivariate functions, such as the characterized battery losses function. A particular focus is placed on the Zig-Zag Integer technique.

Lastly, the Li-Ion characterization explored in 2 and the Zig-Zag Integer linearization technique explored in 3 are applied in a management model of a residential microgrid, presented in 4. The accuracy and performance of the proposed model are tested through various scenarios and optimization horizons to evaluate the scalability of the methods and their suitability.

Finally, the master thesis concludes in chapter 5, describing the results of this research project.

Chapter 2

Li-Ion batteries

In this chapter, chemical batteries are introduced, specially focusing on the modelling of Li-Ion batteries. Firstly the chemical batteries operating principle is presented, next Li-Ion are characterized, and lastly the mathematical model for the Li-Ion loss functions that will be used in the following chapters is developed.

2.1 Chemical Batteries

Chemical batteries are a type of energy accumulator where electrical energy is stored in chemical form. Chemical batteries are based on the electrochemical potential of metals, which is the tendency of a metal to lose and gain electrodes. The inventor that realized this behavior was Alessandro Volta, who in 1790 invented the first battery using Zinc and Silver.

A battery consists un multiple voltaic cells. Each voltaic cell is composed of a cathode, a metal with a tendency to gain electrons, and an anode, a metal with a tendency to lose electrons. When there is an available path connecting the anode and the cathode, the anode loses electrons, which are absorbed by the cathode. The flow of electrons is the electrical current that can be used to power devices.

As the number of electrons builds in the cathode, the repelling forces of elements with the same electrical charge would slow the flow of electrons, for that reason an electrolyte submerges both the anode and the cathode. The electrolyte allows the ions, the atoms from the cathode with missing electrons, to flow to the cathode, equalizing the repelling forces and facilitating the flow of electrons. This chemical reaction ends when the anode is depleted from ions and the flow of electrons stops.

In some batteries, such as Li-Ion batteries, the reaction can be reversed. A source of energy can move electrons backward, from the cathode to the anode, leading to a negative charge buildup in the anode and the movement of the ions

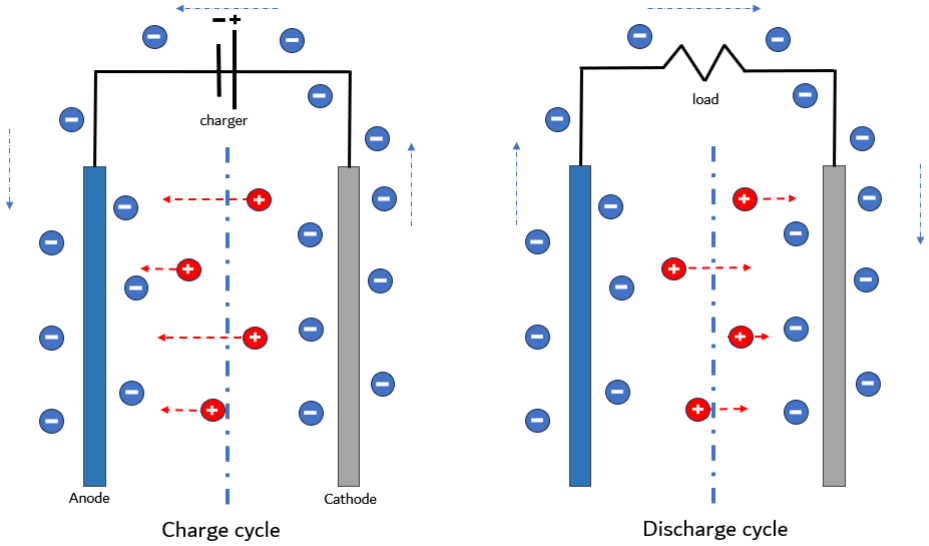


Figure 2.1: Electron flow diagram for chemical batteries.

from the cathode to the anode, which recharges the battery. In 2.1 the flows of electrons (blue bubbles) and ions (red bubbles) can be observed.

The main limitation of chemical batteries is their low energy density compared to other energy storage forms such as fuels, making them unsuitable for storing large quantities of energy. For reference, gasoline has an energy density of 47.5MJ/kg, while a Li-Ion battery has a density of 0.5MJ/kg [7]. Even considering the higher efficiency in electric machines, Li-Ion batteries energy density is still an order of magnitude lower than alternatives such as diesel. Despite this disadvantage, advancements in battery technologies and the current electrification trends are pushing the usage of Li-Ion batteries for large energy storage applications such as cars or power systems.

2.2 Li-Ion batteries

Li-Ion batteries have become the most used chemical battery technology. They started to gain popularity in small electronic devices such as smartphones or laptops due to their high energy density, but their usage was limited to small capacity batteries due to the high cost per stored Ah. As the technology evolved and the cost per Ah decreased, Li-Ion batteries began to be used for high-energy cases such as electric vehicles and battery packs for homes. Due to its increased usage in the electric power industry, modeling its dynamics has become a key

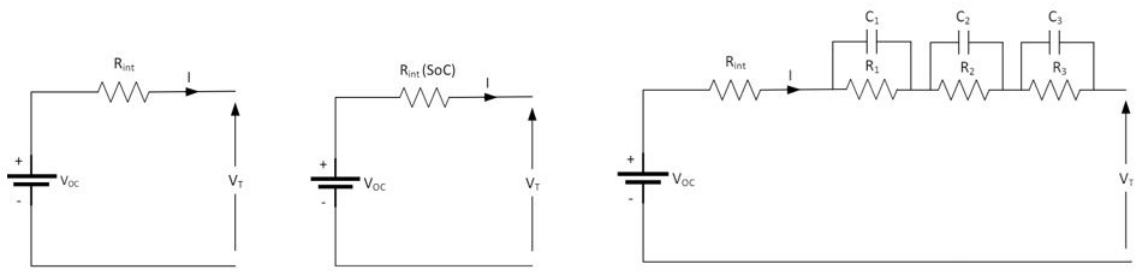


Figure 2.2: Constant Resistance model (left) SoC dependent resistance (center) & Third-order Thevenin model (right).

area of research.

2.2.1 Li-Ion Model

In order to characterize the behavior of chemical batteries such as Li-Ion, Lead-Acid, or NiMH different three main approaches can be considered, experimental, electrical and electrochemical models.

Experimental approaches are abstract models based on statistics, artificial intelligences or other analytical tools, whose accuracy depends on the amount and quality of data used to characterize the battery. Its limitation are the lost of the interpretability as well as the volume of data needed [15].

Electrical models represent batteries with an equivalent electric circuit. The most basic model would be an ideal battery in series with a constant resistance. This model yields reasonable accuracy for Lead Acid or nickel-cadmium batteries, but lacks precision with lithium based batteries. This model can be improved by making the resistance dependent on the state of charge of the battery. More advance electrical models includes a network of capacitors and resistances to model the transient effects of the batteries, referred to as Thevenin models [15].

Lastly, electrochemical models consider both the electrical and the chemical behavior of the batteries, which leads to models with higher detail. [18] proposed 2.2 to model Li-Ion batteries electrochemically. Where V_{batt} (V) express the voltage difference between the battery terminals at any given time. E_o (V) is the constant potential of the cell, that is, the theoretical potential corresponding to the anode and the cathode. R (Ω) is the internal resistance, i (A) is the current drawn from the battery, it (A·h) is the integral of the charge at time t drawn from the battery since a discharge started. K (Ω) is the polarization coefficient, and A (V) and B (dimensionless) are empirical constants. The value of the different parameters can be obtained by fitting the curved to experimental results.

$$R_{pol}(i) = \begin{cases} K\frac{Q_m}{Q_m - it} & \text{if } i > 0 \\ K\frac{Q_m}{it + 0.1 \cdot Q_m} & \text{if } i \leq 0 \end{cases} \tag{2.1}$$

$$V_{batt} = E_o - R \cdot i - R_{pol} i^* - K\frac{Q_m}{Q_m - it} it + A \cdot e^{-B \cdot it} \tag{2.2}$$

2.2.2 Li-Ion loss function

During the process of charge and discharge, small power losses occur, mainly from inefficiencies in the electrochemical reactions and the joule effect due to the internal resistance of the battery. The loss expression 2.3 can be derived from 2.2

$$\begin{aligned}(E_o - V_{batt})i = &\underbrace{R \cdot i^2}_{\text{Term1}} + \underbrace{R_{pol} \cdot i^* \cdot i}_{\text{Term2}} \\ &+ \underbrace{(\frac{Q_m}{Q_m - it} it)i}_{\text{Term3}} - \underbrace{(A \cdot e^{-B \cdot it})i}_{\text{Term4}}\end{aligned} \tag{2.3}$$

Professor Javier García Gonzalez proposed not to consider Term3 & Term4 as they lead to instances where the power loss is negative, indicating that energy is generated instead of lost [13].

In the context of high-power energy management systems, such as a microgrid operation, the transient effects are usually not considered as the problems are simplified to steady-state models. In addition, in this context current tends not to be modeled, instead power is used directly. For these two reasons, in high power management systems [13] proposed to model Li-Ion batteries as 2.4 & 2.5 for modeling discharge and charge cycles. Where soc is the State of charge, defined as $soc = \frac{Q_m - it}{Q_m}$.

$$p^{disc}_{loss,t} = 10^3 (R + \frac{K}{soc_t})(\frac{p^{disc}_t}{V_r})^2 \tag{2.4}$$

$$p^{char}_{loss,t} = 10^3 (R + \frac{K}{1.1 - soc_t})(\frac{p^{char}_t}{V_r})^2 \tag{2.5}$$

As it can be observed, 2.4 & 2.5 are non-linear expressions, which their integration challenging in classical optimization power system models. In the following chapters, these expressions are going to be linearized for its integration in an optimal management mathematical model for an isolated microgrid.

Chapter 3

Piecewise Linear approximation

In this chapter, the state of the art in the modeling of non-linear, non-convex functions as piecewise linear approximations is presented. First, the linearization problem is developed, and its modeling issues are highlighted. Next, the current literature is explored and lastly, the final formulation is presented.

3.1 Problem Statement

Developing algorithms for finding the optimal solution of a linear program (LP) is a problem that was solved in the previous century. In the case of Mixed Integer Programming (MIP), although the research is not as advanced as in LP, modern solvers can handle very complex MIP problems. The situation is different when the optimization problem contains non-linearities and non-convexities, as traceability is lost and optimality is not guaranteed. One of these examples is found in the modeling of Li-Ion batteries, due to their non-linear dynamics, as highlighted in chapter 2.

In order to exploit the advances in MIP, the traditional approach for modeling non-linear functions is its linearization. The original non-linear function can be approximated as a piecewise linear function (pwlf) where each sub-function is the convex combination of a set of points from the original function. With the help of auxiliary binary variables, each linear segment can be activated, converting a non-linear problem into a Mixed Integer Linear Program (MILP).

The issue with this approach is the high number of auxiliary binary variables needed. The traditional textbook approach uses n binary variables for modeling a n segment's univariate pwlf. This dramatically affects the required computation time specially for functions with high number of segments, as the number of binary variables can be used as a proxy for the complexity of the problem.

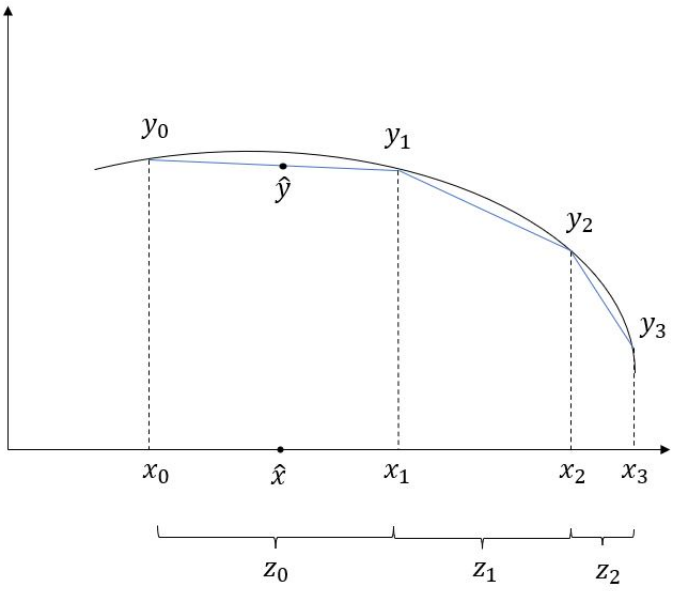


Figure 3.1: PieceWise Linear Approximation of a generic. non-linear univariate function.

3.1.1 Uni-Variate functions

In figure 3.1 it can be observed a generic non-linear function with a 3 segment piecewise linear approximation, defined in equations 3.1 & 3.2

$$f(x) \approx \hat{y} = \begin{cases} y_0\theta_0 + y_1\theta_1, & \text{if } x_0 \leq x \leq x_1 \\ y_1\theta_1 + y_2\theta_2, & \text{if } x_1 \leq x \leq x_2 \\ y_2\theta_2 + y_3\theta_3, & \text{if } x_2 \leq x \leq x_3 \end{cases} \tag{3.1}$$

$$\hat{x} = \begin{cases} x_0\theta_0 + x_1\theta_1, & \text{if } x_0 \leq x \leq x_1 \\ x_1\theta_1 + x_2\theta_2, & \text{if } x_1 \leq x \leq x_2 \\ x_2\theta_2 + x_3\theta_3, & \text{if } x_2 \leq x \leq x_3 \end{cases} \tag{3.2}$$

The text-book modeling approach [17] for unidimensional picewise linear functions ($y = f(x)$) is presented in equations 3.3 and 3.4. The first set of equations is responsible for approximating $\hat{x}$ and $\hat{y}$ as the convex combination of the parameters x_i and y_i that are real points of the function $f(x)$. The weights assigned to each of the points are θ_i, a positive continuous variable that represent the weights of the convex combination.

The second set of constraints (3.4) ensures that only two adjacent θ_i have a value different from 0, leading to the selection of a segment, that is the "if statement" from 3.1 & 3.2. Each segment has associated a binary variable z_i that

will be set to 1 if the segment is selected.

$$\hat{x} = \sum_{i=1}^{n+1} \theta_i \cdot x_i \tag{3.3a}$$

$$\hat{y} = \sum_{i=1}^{n+1} \theta_i \cdot y_i \tag{3.3b}$$

$$1 = \sum_{i=1}^{n+1} \theta_i \tag{3.3c}$$

$$\sum_{i=1}^{n+1} z_i = 1 \tag{3.4a}$$

$$\theta_i \leq z_{i-1} + z_i \quad \forall i \in 2..n \tag{3.4b}$$

$$\theta_{n+1} \leq z_n \tag{3.4c}$$

$$\theta_1 \leq z_1 \tag{3.4d}$$

$$0 \leq \theta_i \leq 1 \tag{3.4e}$$

$$\theta_i, \hat{x}, \hat{y} \in \mathbb{R} \tag{3.4f}$$

$$z_i \in \{0, 1\} \tag{3.4g}$$

For example, for the modeling of a 3 segment function such as the presented in figure 3.1 the developed equations are presented in 3.5.

$$\hat{y} = y_0\theta_0 + y_1\theta_1 + y_1\theta_1 + y_2\theta_2 + y_2\theta_2 + y_3\theta_3 \tag{3.5a}$$

$$\hat{x} = x_0\theta_0 + x_1\theta_1 + x_1\theta_1 + x_2\theta_2 + x_2\theta_2 + x_3\theta_3 \tag{3.5b}$$

$$1 = \theta_0 + \theta_1 + \theta_2 + \theta_3 \tag{3.5c}$$

$$\theta_0 \leq z_0 \quad \theta_1 \leq z_0 + z_1 \qquad \theta_2 \leq z_1 + z_2 \quad \theta_3 \leq z_2 \tag{3.5d}$$

$$z_1 + z_2 + z_3 = 1 \tag{3.5e}$$

$$\theta_0, \theta_1, \theta_2, \theta_3 \in \mathbb{R}_{\geq 0} \qquad z_1, z_2, z_3 \in \{0, 1\} \tag{3.5f}$$

As stated previously, this approach is limited because of the high number of binary variables and constraints that it introduces to the problem. For an n-segment piecewise linear approximation, n auxiliary binary variables and n+1 binary constraints are used to model the selection of a segment.

The complexity of the problem lies in the combinatorial nature of choosing the optimal segment, that is, ensuring that at most two adjacent θ have a value different from zero. A set of ordered variables, where at most 2 adjacent variables has a value different from zero, is usually referred to in the literature as a Special Ordered Set of type 2 (SOS2) and extensive research has been focused on improving the formulation of SOS2 constraints due to its application in a wide range of problems.

3.1.2 Bi-Variate functions

For the case of modeling bivariate functions $z = g(x, y)$, a similar approach as in the univariate case can be applied. The function g is approximated as a pwlf, here each sub-function is the convex combination of 3 points of the original function $g(x, y)$, therefore defining a plane. This approach can be visualized in figure 3.2. The domain of $g(x, y)$ is divided into triangles embedded in an (n+1)x(m+1) grid, where a convex combination weight (θ) is assigned to each triangle vertices.

The base for bivariate pwlf modeling are the equations that represent the convex combination of the functions points, depicted in equation 3.6.

$$\hat{x} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} \theta_{ij} \cdot x_i \tag{3.6a}$$

$$\hat{y} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} \theta_{ij} \cdot y_j \tag{3.6b}$$

$$\hat{z} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} \theta_{ij} \cdot f(x_i, y_j) \tag{3.6c}$$

$$\sum_{i=1}^{n+1} \sum_{j=1}^{m+1} \theta_{ij} = 1 \tag{3.6d}$$

$$0 \leq \theta_{ij} \leq 1 \tag{3.6e}$$

$$\theta_{ij}, \hat{x}, \hat{y} \in \mathbb{R} \tag{3.6f}$$

As in the univariate case, the convex combination should be limited, as the approximation will be valid only if three adjacent $\theta_{i,j}$ have a value different from

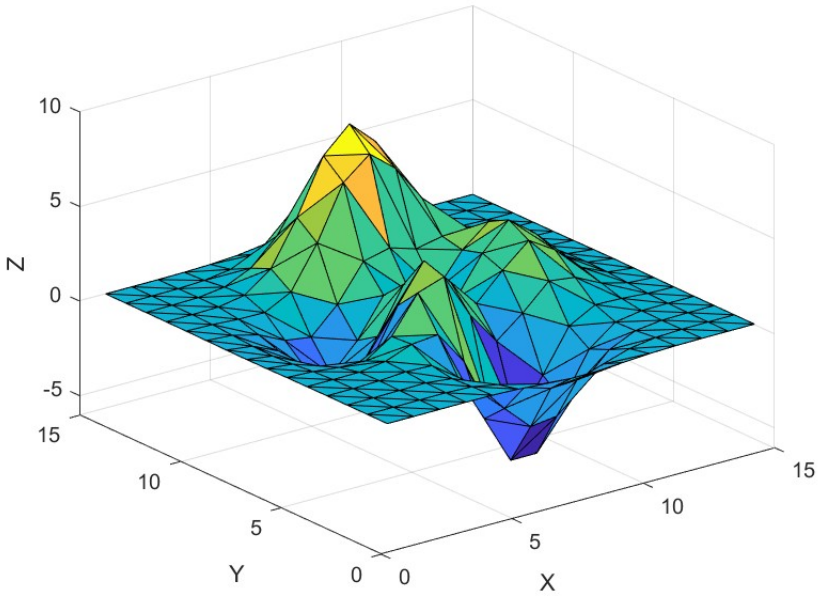


Figure 3.2: PieceWise Linear Approximation of a generic. non-linear bivariate function.

zero. If no additional constrains are used, 3.6 will model the convex hull of all the points $[x_i, y_i, z_i]$. The desired approximation will just represent the convex hull of three adjacent points, representing a triangle. Additional constraints have to force the rest of the $\theta_{i,j}$ to zero. This process will be referred to as triangle selection.

Babayev Approach

A simple approach to model the triangle selection is to assign a binary variable to each triangle and allowing only one binary variable to be 1. This approach is described in [4] and can be modeled with the equation 3.7.

$$\sum_{i \in 1..n+1} \sum_{j \in 1..m+1} \zeta^u_{i,j} + \zeta^l_{i,j} = 1 \tag{3.7a}$$

$$\begin{aligned} \theta_{ij} \leq\ & \zeta^u_{i,j} + \zeta^l_{i,j+1} + \zeta^u_{i-1,j} \\ & + \zeta^l_{i,j} + \zeta^u_{i,j-1} + \zeta^l_{i+1,j} \quad \forall i \in 1..n+1, \forall j \in 1..m+1 \end{aligned} \tag{3.7b}$$

where it is assumed that the binary variables with subscripts $i-1$, $j-1$, $(i+1,\ j+1)$ are null for the first (last) elements of I and J. In addition, $\zeta^l_{1,j} = \zeta^u_{n+1,j} = 0, \forall j \in 1..m+1$, and $\zeta^l_{i,1} = \zeta^u_{i,m+1} = 0, \forall i \in 1..n+1$.

If the variable $\zeta^l_{i,j}$ is set to 1, all θ but $\theta_{i,j}, \theta_{i-1,j}, \theta_{i,j-1}$ will be set to 0. In the other hand, if $\zeta^u_{i,j}$ is set to 1, $\theta_{i,j}, \theta_{i+1,j}, \theta_{i,j+1}$ could have a value different from 0.

Combining equations 3.6 & 3.7 a non-convex bi-variate function could be represented just by binary and continuous variables, that is, a MILP. The limitation of this approach is the high number of variables need, one for each triangle, which can make problems insolvable in practical times.

Text Book Approach

An improved approach for modeling the triangle selection is based on leveraging SOS2 constraints, as in the univariate case.

If an (n+1) x (m+1) grid is plotted and each node of the grid is linked to a θ (i.e. node (1,1) $\rightarrow \theta_{1,1}$ node (1,2) $\rightarrow \theta_{1,2}$) The selection of a triangle could be broken down into first selecting two adjacent columns of θ and two adjacent rows of θ [1]. That is, the sum of all the θ corresponding to 2 adjacent columns equal 1. And the sum of all the θ corresponding to 2 adjacent rows equal 1. The overlap of the selection would lead to just the 4 adjacent θ that are both in the selected column and row to add up to 1, that is, selecting a rectangle (figure 3.4).

The selection of the adjacent columns/rows can be understood as a SOS2 constraint, as there is an ordered set of elements (i.e. columns or rows) and only two adjacent elements can have values different from 0.

Lastly, the triangle will be selected by limiting the upper bound of a predefined set of θ to 0. This set is defined in a way that it contains at least one of the θ inside the selected rectangle. Therefore, narrowing down the θ that are not limited to zero from four to three. This selection can be modeled with auxiliary binary variables, which sets to 0 or to 1 a given set of θ.

The combination of the rectangle and triangle selection will lead to three adjacent θ to have a value different from 0. Therefore, 3.6 will represent just the convex combination of three adjacent points, not the full set.

To translate this approach to equations, 4 sets of equations are needed. Equations 3.6 are responsible for the convex combination. Equations 3.8 and 3.9 are the textbook approach for SOS2 constraints applied to the columns and rows of, θ respectively. Lastly, the final selection of a triangle is modeled with equations 3.10 or 3.11, extracted from [21]. Choosing one set of triangle selection equations over the other will lead to different triangulation patterns, as can be visualized in figure 3.3. It is interesting to mention that the method described in [4] will lead to the triangulation pattern pictured in 3.3a named, K1 triangulation.

[1] Column i: $\sum_{j \in 1..m+1} \theta_{i,j}$. Row j: $\sum_{i \in 1..n+1} \theta_{i,j}$

This method present a considerable lower number of binary variables compared to the Babayev approach, presumable leading to lower computing time.

Column selection Equations:

$$\sum_{i=1}^{n+1} \zeta_i^{col} = 1 \tag{3.8a}$$

$$\sum_{j=1}^{m+1} \theta_{ij} \leq \zeta_{i-1}^{col} + \zeta_i^{col} \quad \forall i \in 2..n \tag{3.8b}$$

$$\sum_{j=1}^{m+1} \theta_{n+1,j} \leq \zeta_n^{col} \tag{3.8c}$$

$$\sum_{j=1}^{m+1} \theta_{1,j} \leq \zeta_1^{col} \tag{3.8d}$$

$$\zeta_i^{col} \in \{0,1\} \tag{3.8e}$$

Row selection Equations:

$$\sum_{j=1}^{m+1} \zeta_j^{row} = 1 \tag{3.9a}$$

$$\sum_{i=1}^{n+1} \theta_{ij} \leq \zeta_{j-1}^{row} + \zeta_j^{row} \quad \forall j \in 2..m \tag{3.9b}$$

$$\sum_{i=1}^{n+1} \theta_{i,m+1} \leq \zeta_m^{row} \tag{3.9c}$$

$$\sum_{i=1}^{n+1} \theta_{i,1} \leq \zeta_1^{row} \tag{3.9d}$$

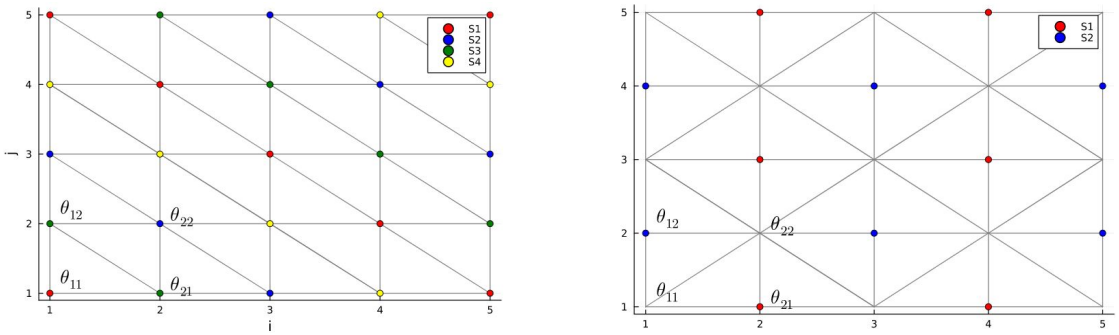
$$\zeta_j^{row} \in \{0,1\} \tag{3.9e}$$

Triangle selection Equations:

$$\begin{aligned} &\sum_{(i,j)\in S_1} \theta_{i,j} \leq z_1 \qquad \sum_{(i,j)\in S_2} \theta_{i,j} \leq 1 - z_1 \\ &\sum_{(i,j)\in S_3} \theta_{i,j} \leq z_2 \qquad \sum_{(i,j)\in S_4} \theta_{i,j} \leq 1 - z_2 \\ &z_1, z_2 \in \{0,1\} \end{aligned} \tag{3.10}$$

$$
\begin{aligned}
S1 &= \{(i,j) : i+j \equiv 2(\mod 4)^2\} \\
S2 &= \{(i,j) : i+j \equiv 0(\mod 4)\} \\
S3 &= \{(i,j) : i+j \equiv 1(\mod 4)\} \\
S4 &= \{(i,j) : i+j \equiv 3(\mod 4)\} \\
&\forall i \in [1..n+1] \\
&\forall j \in [1..m+1] \\
\sum_{(i,j)\in S_1} \theta_{i,j} \leq z_1 \qquad & \sum_{(i,j)\in S_2} \theta_{i,j} \leq 1 - z_1 \\
z_1 &\in \{0,1\}
\end{aligned}
\tag{3.11}
$$

$$
\begin{aligned}
S1 &= \{(i,j) : \text{i is even and j is odd}\} \\
S2 &= \{(i,j) : \text{j is even and i is odd}\} \\
&\forall i \in [1..n+1] \\
&\forall j \in [1..m+1]
\end{aligned}
$$

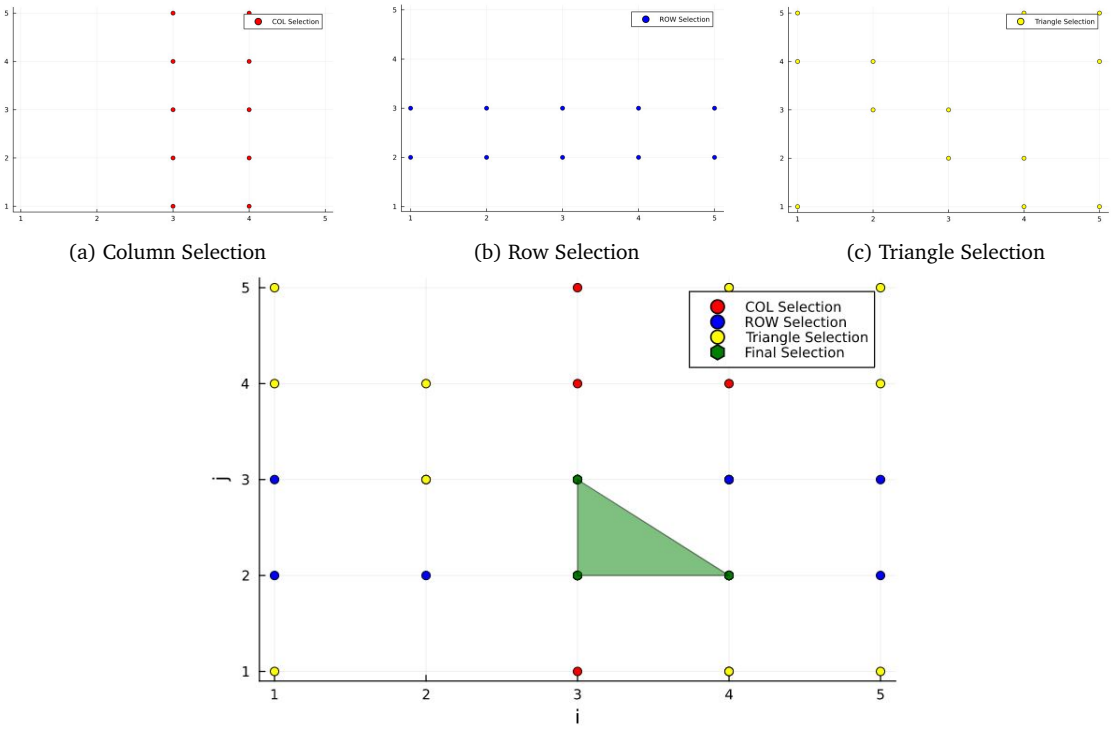


(a) K1 triangulation

(b) Union Jack or J1 triangulation

Figure 3.3: Triangulations

[2] $a \equiv b(\mod n) \longrightarrow a$ is congruent with b module c if $a - b$ is divisible by n.



(a) Column Selection (b) Row Selection (c) Triangle Selection

(d) Final Triangle Selection

Figure 3.4: Final Selection

3.2 Literature Review

As stated in the previous section, SOS2 type of constraints are essential for modeling non-linear function as a piecewise linear function, but SOS2 constraints are also used in a wide range of problems such as scheduling, or warehouse sizing. Due to its importance, it is an area of research with great contributions.

There are two main approaches for solving optimization problems with SOS2 constraints, using dedicated algorithms or leveraging integer decision-making variables with commercial MIP solvers. Inside the MIP category, the formulations can be classified into combinatorial and geometric approaches.

The goal of this research is to find a formulation that is easy to use, scalable, and platform independent. For this reason, the literature review is focused on the combinatorial and geometric approaches as dedicated algorithms are usually "black-box" algorithms restricted to certain solvers or modeling languages, making their application limited.

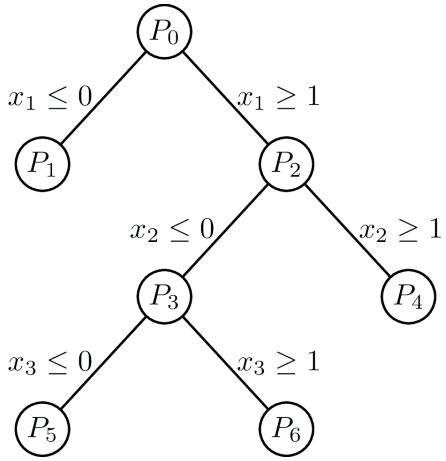


Figure 3.5: Branch & Bound tree.

To assess the different alternatives for modeling SOS2 constraints, 3 quality measurements are defined based on [12] and [20]. These measurements are empirically related to the computational performance of the MIP problems, and therefore applicable to the SOS2 constraints explored in this master thesis.

Branching behavior: Modern MILP solvers are based on a branch and bound algorithm. This algorithm can be visualized as a tree that begins by solving the LP relaxation of a MILP problem, imposing different bounds in the integer variables. Through each branch, opposite bounds are set, for example in a binary problem two opposing branches could be $x_1 \leq 0$ and $x_1 \geq 1$. After the LP relaxation is solved, the objective function is analyzed, and poor-performance branches are pruned. Problems with good branching behavior will lead to trees where in each level the size of the LP relaxation is reduced and branches are balanced, in the sense that both sub-problem's solutions change substantially and therefore the decision to prune a branch is straight forward.

Size: The number of relaxations that the branch and bound algorithm has to solve scales exponentially with the number of discrete variables in the problem. Therefore, the number of discrete variables can be used as a proxy for the problem's complexity.

Strength: As a MIP problem is solved by finding the optimal solution of its relaxed version, the characteristics of the LP relaxation is determinant for finding an optimal integer solution efficiently. A strong or ideal MIP formulation is one whose extreme points are integer.

TIMELINE 1: *ZigZag Method evolution*

- Modeling disjunctive constraints with a logarithmic number of binary variables and constraints [21].
- A geometric way to build strong mixed-integer programming formulations [11].
- Embedding Formulations and Complexity for Unions of Polyhedra [19].
- A combinatorial approach for small and strong formulations of disjunctive constraints [10].
- Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools [9].

3.2.1 Combinatorial Independent Branching approach

A method commonly used by ad-hoc approaches is to use constraint branching, replacing the binary auxiliary variables used traditionally in SOS2 constraints with constraints that are dynamically added to an LP problem. These methods are usually based on variations of the Branch & Bound algorithm where instead of branching over different values of the integer variables, the branch is done over independent sets of constraints. This method is either computed internally in some solvers or based on the branching callback options in some comercial solvers (i.e. CPLEX).

The Combinatorial Independent Branching approach exploits the idea of constraint branching but instead of using callback options in solvers, it uses a MIP formulation. The main advantage of this method is reducing the technical experience needed in implementing the Ad-hoc approaches.

An example of this method is found in the triangle selection equations 3.1.2 and 3.10. Following the Branch & Bound algorithm, a different set of constraint will be added to the problem depending on whether the binary variable z_1 is set to 0 or 1.

[21] proposes LogIB, a formulation for SOS2 constraints based on independent branching. This formulation uses $log_2(n)$ binary variables for an SOS2 constraints of n elements. A considerable improvement in the quality of the problem regarding its size when compared to the Text Book approach.

3.2.2 Geometric approach

The geometric approach leverages the geometric characteristics that the mathematical space that represent SOS2 constraints has.

Given a set of ordered variables (i.e. $\{\lambda_i \forall i \in 1..n+1\}$) Each element that composes a SOS2 constraint (i.e. $\lambda_1 + \lambda_2 = 1$, $\lambda_2 + \lambda_3 = 1$ ...) can be modeled with the classical optimization matrix form as $Ax \leq b$. Then the union of all polyhedra (A) will represent the set of SOS2 constraints (D).

For a set of n elements:

$$D = \bigcup_{i=1}^{n} P_i \qquad P_i = \{x \in \mathbb{R}^n : A^i x \leq b^i\} \tag{3.12}$$

[19] proves that D could be embedded in a higher dimensional space where each alternative P_i is assigned a unique code $h_i \in \mathbb{Z}^r$. Where r is an integer $\geq \lceil log_2(n) \rceil$ [3]. The set of all the unique codes is going to be referred to as the encoding matrix $H = (h^i)_{i=1}^n$. For some encoding matrix, the convex hull of $Em(D, H)$ will be a strong MILP formulation of the SOS2 constraint. The formulation would be composed of a set of LP inequalities that represent D and a set of constraints similar to 3.4 where auxiliary integer variables $\bar{\zeta}$ model the selection of only one P_i.

$$Em(D, H) = \bigcup_{i=1}^{d} (P_i \times h^i) \tag{3.13}$$

The main problem with this approach is the computation of the convex hull of $Em(D, H)$. [19] introduced a new paradigm for constructing the convex hull of $Em(D, H)$ for SOS2 constraints, the "Embedding Formulation" presented in 3.14.

$$\sum_{i=1}^{n+1} \lambda_i = 1 \tag{3.14a}$$

$$\sum_{i=1}^{n+1} min\{b^{k'} h^i, b^{k'} h^{i-1}\} \lambda_i \leq b^j \bar{\zeta} \leq \sum_{i=1}^{n+1} max\{b^{k'} h^i, b^{k'} h^{i-1}\} \lambda_i \quad \forall k \in 1..r \tag{3.14b}$$

$$\bar{\zeta} \in \mathbb{Z}^r \tag{3.14c}$$

where b can be defined as the spanning hyperplanes of $\{h^{i+1} - h^i\}_{i=1}^n$

3.14 allows us to construct different formulations using the same set of equations but with different encoding matrix H. However, selecting a H that yields an optimal formulation is not a trivial task. [19] proposed the embedded formulation alongside different possible encodings.

[3] $\lceil x \rceil$ Ceil: represents the smallest integer that is greater than or equal to x.

If H represents Binary Reflected Gray Codes (K) the equations 3.14 will yield a formulation for SOS2 constraints with $log_2(n)$ binary variables. This formulation is known as LogE, and if analyzed using the quality measurements introduced previously, this formulation is similar and even equivalent to LogIB formulation in some cases as described in [19] and [12].

Based on 3.14, [9] proposed two new encoding matrices, C and Z, which are based on K and the resulting SOS2 formulations are referred to as the Zig-Zag Integer (ZZI) and ZigZag Binary (ZZB) formulations respectively.

[12] proves that these two formulations are equivalent to LogE and LogIB regarding size and strength, but they have a better branching behavior, especially the ZZI formulation. Computation results provided in [9] showcase an order of magnitude speed-up for bivariate pwlf and 3x for hard stances of univariate pwlf. In this master thesis, the focus is going to be placed on the Zig-Zag Integer formulation due to its performance.

C can be defined as the number of times each row of the matrix K changes. ($C^r_{i,k} = \sum_{j=2}^{i} |K^r_{j,k} - K^r_{j-1,k}|$). On the other hand, the encoding matrix Z is a linear transformation of the rows of C. $Z^r_k = \mathcal{A}(C^r_k)$ where $\mathcal{A}(z)_k = z_k - \sum_{l=k+1}^{r} z_l \quad \forall k \in [r]$. For simplicity matrices K, C and Z can be defined recursively leading to a n x $log2(n)$ matrix, but it has to be considered that $H_0 \equiv H_1$, and $H_{n+1} \equiv H_n$ for K,Z and C.

In the case of using the encoding matrices K and C, b is the canonical base of $\mathbb{R}^r$ (i.e. if r =3 $\{(1,0,0),(0,1,0),(0,0,1)\}$). In the case of Z it can be computed by applying the inverse of the linear transformation $\mathcal{A}$ to the canonical base of $\mathbb{R}^r$.

Despite $\bar{\zeta}$ being defined as a general integer variable, due to the properties of 3.14 these variables will be naturally bounded as $\bar{\zeta} \in H$. This means that for the ZZB and LogE formulations, $\bar{\zeta}$ will behave as a binary variable and in the case of ZZI as a positive integer. This reason alongside others is what named the Zig-Zag Integer.

Lastly, it's worth to mention that despite the computation advantages of the methods described in this literature review, they are not widely used due to difficulties with their interpretation and implementation. To bridge that gap, commercial solvers such as Gurobi offers the modeling of SOS2 with different methods like the Zig-Zag Integer and Zig-Zag Binary [14]. The authors of [9] went an extra mile by providing PiecewiseLinearOpt, a library for Julia Lang that

Formulation	Constraints with Int. Variables	Integer Variables
Text Book (J1 triang.)	$n+m+8$	$n+m+1$
ZZI & ZZB (J1 triang.)	$2(log_2(n)+log_2(m)+1)+3$	$log_2(n)+log_2(n)+1$
Babayev	$(n+1)(m+1)+4$	$2(n+m)$

Table 3.1: Models size comparative for bi-variate pwlf with a (n+1) x (m+1) grid resolution.

allows the application of all the methods described for linearization of functions. In appendix 5 this library is explored.

$$K^1 = C^1 = Z^1 = (0,1)^T$$

$$K^{p+1} = \begin{pmatrix} K^p & 0^{2^p} \\ rev^4(K^r) & 1^{2^p} \end{pmatrix} \quad C^{p+1} = \begin{pmatrix} C^p & 0^{2^p} \\ C^p + 1^{2^p} \times C^p_{2^p} & 1^{2^p} \end{pmatrix} Z^{p+1} = \begin{pmatrix} Z^p & 0^{2^p} \\ Z^p & 1^{2^p} \end{pmatrix}$$

Model Size comparative

As stated previously, the number of integer variables and constraints can be used as an indicator of the problem complexity. And considering that it is really simple to assess, compared to other characteristics such as the problem strength or branching behavior, it is the most used criteria to assess MIP problems complexity. In table 3.1 the size of the presented methods is compared. It can be observed how drastically different the size of the problem can be. Although the different formulations represent the same model behavior (i.e. bivariate piece wise linearization) the number of integer variables needed can vary in orders of magnitude.

3.3 Zig-Zag Integer Method

Due to the promising results that the Zig-Zag Integer formulations has, it will be explored with great detail. Given an ordered set of variables (i.e. $\{\lambda_i \forall i \in 1..n+1\}$) the SOS2 formulation is going to be developed in this section.

Parameters

n Number of alternatives in the ordered set

C Encoding matrix for the ZZI formulation

[4] rev(A) reverses the rows of the matrix A

b_C Spanning hyperplanes of matrix C

Variables

λ_i Ordered set of variables where the SOS2 constraint is applied.

ζ_r^{zzi} Auxiliary variables for the ZZI formulation

Due to the mathematical properties of the encoding matrix C, the formulation 3.14 can be simplified to 3.15.

$$\sum_{i=1}^{n+1} C_{i-1,k}^r \lambda_i \leq \zeta_k \leq \sum_{i=1}^{n+1} C_{i,k}^r \lambda_i \quad \forall \quad k = [1, r] \tag{3.15}$$

$$0 \leq \zeta_i \leq 2^{r-i} \quad \forall \quad i = [1, r] \tag{3.16}$$

$$\zeta_k \in \mathbb{Z} \quad \forall \quad k = [1, r] \tag{3.17}$$

First 3.15 is written in scalar form, while 3.14 is defined in vector form (i.e. $\bar{\zeta} = [\zeta_1, \zeta_2, ..., \zeta_r]$). Secondly, the vector multiplication b^j by $\bar{\zeta}$ and by the encoding element is computed. Lastly, the rows of matrix C are defined as the number of times each component of the sequence of the column of K changes of value, therefore the components of the rows of C are monotonic non-decreasing. This leads to $min\{C_{i-1,k}, C_{i-1,k}\} = C_{i-1,k}$ and $max\{C_{i,k}, C_{i-1,k}\} = C_{i,k}$.

For example, if n = 4:

$$C^2 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{pmatrix} = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \\ C_{3,1} & C_{3,2} \\ C_{4,1} & C_{4,2} \end{pmatrix}$$

$$b_C^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$\begin{aligned}
&min\{[1,0]'[0,0], \quad [1,0]'[0,0]\}\lambda_1 + \\
&min\{[1,0]'[0,0], \quad [1,0]'[1,0]\}\lambda_2 + \\
&min\{[1,0]'[1,0], \quad [1,0]'[1,1]\}\lambda_3 + \\
&min\{[1,0]'[1,1], \quad [1,0]'[2,1]\}\lambda_4 \\
&\qquad \leq [1,0]'[\zeta_1, \zeta_2] \leq \quad \longrightarrow \lambda_3 + \lambda_4 \leq \zeta_1 \leq \lambda_2 + \lambda_3 + 2\lambda_4 \\
&max\{[1,0]'[0,0], \quad [1,0]'[0,0]\}\lambda_1 + \\
&max\{[1,0]'[0,0], \quad [1,0]'[1,0]\}\lambda_2 + \\
&max\{[1,0]'[1,0], \quad [1,0]'[1,1]\}\lambda_3 + \\
&max\{[1,0]'[1,1], \quad [1,0]'[2,1]\}\lambda_4
\end{aligned} \tag{3.18}$$

$$
\begin{aligned}
&min\{[0,1]'[0,0], \quad [0,1]'[0,0]\}\lambda_1+ \\
&min\{[0,1]'[0,0], \quad [0,1]'[1,0]\}\lambda_2+ \\
&min\{[0,1]'[1,0], \quad [0,1]'[1,1]\}\lambda_3+ \\
&min\{[0,1]'[1,1], \quad [0,1]'[2,1]\}\lambda_4 \\
&\qquad\qquad \leq [0,1]'[\zeta_1,\zeta_2] \leq \quad \longrightarrow \lambda_4 \leq \zeta_2 \leq \lambda_3 + \lambda_4 \\
&max\{[0,1]'[0,0], \quad [0,1]'[0,0]\}\lambda_1+ \\
&max\{[0,1]'[0,0], \quad [0,1]'[1,0]\}\lambda_2+ \\
&max\{[0,1]'[1,0], \quad [0,1]'[1,1]\}\lambda_3+ \\
&max\{[0,1]'[1,1], \quad [0,1]'[2,1]\}\lambda_4
\end{aligned}
\tag{3.19}
$$

For an ordered set of 8 elements and therefore 7 possible alternatives, 3.15 yields the following set of constraints:

$$
n = 7 \longrightarrow r = \lceil log_2(7) \rceil = \lceil 2.807 \rceil = 3
$$

$$
C^r = \begin{pmatrix}
0 & 0 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
2 & 1 & 0 \\
2 & 1 & 1 \\
3 & 1 & 1 \\
3 & 2 & 1 \\
4 & 2 & 1
\end{pmatrix}
$$

ζ_1	ζ_2	ζ_3	Selected λ
0	0	0	$\lambda_1 + \lambda_2 = 1$
1	0	0	$\lambda_2 + \lambda_3 = 1$
1	1	0	$\lambda_3 + \lambda_4 = 1$
2	1	0	$\lambda_4 + \lambda_5 = 1$
2	1	1	$\lambda_5 + \lambda_6 = 1$
3	1	1	$\lambda_6 + \lambda_7 = 1$
3	2	1	$\lambda_7 + \lambda_8 = 1$
4	2	1	$\lambda_8 + \lambda_9 = 1$

Table 3.2: Relationship between the value of ζ and λ.

$$
\begin{aligned}
0\lambda_1 + 0\lambda_2 + 1\lambda_3 + 1\lambda_4 + 2\lambda_5 + 2\lambda_6 + 3\lambda_7 + 3\lambda_8 &\leq \zeta_1 \\
0\lambda_1 + 1\lambda_2 + 1\lambda_3 + 2\lambda_4 + 2\lambda_5 + 3\lambda_6 + 3\lambda_7 + 3\lambda_8 &\geq \zeta_1 \\
\\
0\lambda_1 + 0\lambda_2 + 0\lambda_3 + 1\lambda_4 + 1\lambda_5 + 1\lambda_6 + 1\lambda_7 + 2\lambda_8 &\leq \zeta_2 \\
0\lambda_1 + 0\lambda_2 + 1\lambda_3 + 1\lambda_4 + 1\lambda_5 + 1\lambda_6 + 2\lambda_7 + 2\lambda_8 &\geq \zeta_2 \\
\\
0\lambda_1 + 0\lambda_2 + 0\lambda_3 + 0\lambda_4 + 0\lambda_5 + 1\lambda_6 + 1\lambda_7 + 1\lambda_8 &\leq \zeta_3 \\
0\lambda_1 + 0\lambda_2 + 0\lambda_3 + 0\lambda_4 + 1\lambda_5 + 1\lambda_6 + 1\lambda_7 + 1\lambda_8 &\geq \zeta_3 \\
\\
\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 + \lambda_8 &= 1 \\
\lambda_i &\in \mathbb{R} \\
\zeta_i &\in \mathbb{Z}
\end{aligned}
$$

In this example, it can be observed the advantage of this formulation. Traditionally, 7 binary variables would be used to model this SOS2 set, while with just 3 integer variables, equation 3.15 can model the same concept. The main disadvantage of this method is the lost interpretability of the auxiliary variables ζ and the constraints. In the classical method, each binary variable has a clear interpretation, while with this approach the relationship is not straightforward.

In the table 3.2 it can be observed that, as stated previously, $\bar{\zeta} \in C$ and the relationship between the value that $\bar{\zeta}$ takes and the selected λ. This relationship is the reason why H is referred to as the encoding matrix, as it defines the correspondence between $\bar{\zeta}$ and λ.

3.3.1 An illustrative example: piecewise linear functions

As demonstrated previously in section 3.1, SOS2 constraints and therefore the ZZI formulation can be used for modeling univariate and bivariate piecewise linear functions. The SOS2 constraints 3.4, 3.8 & 3.9 can substitute by 3.20 3.21 & 3.22 respectively, leading to a more efficient formulation for the univariate and bivariate piecewise linear functions.

Uni-variate:

$$\sum_{i=1}^{n+1} C_{i-1,k}^{r}\theta_i \leq \zeta_k \leq \sum_{i=1}^{n+1} C_{i,k}^{r}\theta_i \quad \forall \quad k=[1,r] \tag{3.20a}$$

$$0 \leq \zeta_k \leq 2^{r-i} \quad \forall \quad k=[1,r] \tag{3.20b}$$

$$\zeta_k \in \mathbb{Z} \quad \forall \quad k=[1,r] \tag{3.20c}$$

Bi-variate:
Row selection:

$$\sum_{i=1}^{n+1} C_{i-1,k}^{r} \sum_{j=1}^{m+1} \theta_{i,j} \leq \zeta_k^r \leq \sum_{i=1}^{n+1} C_{i,k} \sum_{j=1}^{m+1} \theta_{i,j} \quad \forall \quad k=[1,r] \tag{3.21a}$$

$$0 \leq \zeta_k \leq 2^{r-k} \quad \forall \quad k=[1,r] \tag{3.21b}$$

$$\zeta_k^r \in \mathbb{Z} \quad \forall \quad k=[1,r] \tag{3.21c}$$

where $r = \lceil log_2(n) \rceil$

Column selection:

$$\sum_{j=1}^{m+1} C_{j-1,k}^{s} \sum_{i=1}^{n+1} \theta_{i,j} \leq \zeta_k^c \leq \sum_{j=1}^{m+1} C_{j,k}^{s} \sum_{i=1}^{n+1} \theta_{i,j} \quad \forall \quad k=[1,s] \tag{3.22a}$$

$$0 \leq \zeta_k^c \leq 2^{r-k} \quad \forall \quad k=[1,s] \tag{3.22b}$$

$$\zeta_k^c \in \mathbb{Z} \quad \forall \quad k=[1,s] \tag{3.22c}$$

where $s = \lceil log_2(m) \rceil$

3.3.2 Implementation

The goal of this master thesis is the optimal management of a microgrid considering the non-linear losses presented in the Li-Ion batteries. As defined in the previous chapter, this looses can be characterized as bi-variate function, for this

reason the linearization method presented in this chapter were study to assess the most suitable approach for optimizing bivariate functions.

In this section, a generic GAMS implementation of the different linearization methods for bivariate functions is presented with as an illustration of how it has been implemented in this master thesis.

The implementation was designed with versatility in mind, as it allows the user to select between theZig-Zag Integer formulation, the Text Book or Babayev approaches as well as the different triangulations, to select the desired grid resolution and to linearize a generic mathematical function. Although in this master thesis it was applied in a cost minimization microgrid management model, it could be integrated in a wide range of cases such as a profit maximization models.

Declaration of Parameters, Sets & Variables: This code section includes the declaration of parameters, sets & variables. Of those parameters, the user will only assign the desire value to those that define the mathematical domain and grid resolution of the desired function to be linearized. The rest of parameters and sets will be populated automatically.

```
* Definition of scalars
* =====================
scalars
*1:K1 triangulation 2: J1 triangulation
   type_triangulation /1/
*1:ZZI   2:TextBook    3:Babayev
   type_method        /1/

   num_columns  x resolution  /4/
   num_rows     y resolution  /4/
   x_u          x upper limit /10/
   x_l          x lower limit /0/
   y_u          y upper limit /10/
   y_l          y lower limit /0/

   r_max
   s_max
;

* Definition of sets
* ==================
*ZigZag Sets:
\$offOrder
sets
    enteros   /1*100/
    i(enteros)
    j(enteros)
    r(enteros)
    s(enteros)

*K1 triang
    s1(enteros,enteros)
```

```
    s2(enteros,enteros)
    s3(enteros,enteros)
    s4(enteros,enteros)

*J1 triang
    s1_jk(enteros,enteros)
    s2_jk(enteros,enteros);

* Definition of parameters
* =========================
parameters
* Convex combination points
   x_grid(enteros)
   y_grid(enteros)
   z_grid(enteros,enteros)
;

*ZigZag:
Table cr(enteros,enteros);
Table cs(enteros,enteros);

*ZZI or benchmark
if (type_method=1,
    r_max = ceil(log2(num_rows    -1));
    s_max = ceil(log2(num_columns -1));
);
if (type_method=2,
    r_max = num_rows-1;
    s_max = num_columns-1;
);

i(enteros)=yes\$(enteros.ord <= num_columns and enteros.ord >= 1);
j(enteros)=yes\$(enteros.ord <= num_rows    and enteros.ord >= 1);

r(enteros)=yes\$(enteros.ord <= r_max       and enteros.ord >= 1);
s(enteros)=yes\$(enteros.ord <= s_max       and enteros.ord >= 1);

* Definition of variables
* =======================
positive variables
    x
    y
    z
;
Variables
    theta(enteros,enteros)
;
integer Variables
    chi_r(enteros)
    chi_s(enteros)
;
binary variables
    z1
    z2
*Babayev variables:
    chi_u(enteros,enteros)
    chi_w(enteros,enteros)
;
```

Sets and parameters population: In order to populate the different sets and

parameters, a script of Python is embedded in the GAMS source code thanks to the instructions *&onEmbeddedCode Python:* and *&offEmbeddedCode*. It is in this section where the desired bivariate function is declared. For example, in the presented code, the function $f(x, y) = x^2 + y^2$ is linearized. Additionally, the auxiliary sets and parameters such as the C matrix for the ZZI formulation is populated.

```
$onEmbeddedCode Python:
import numpy as np

def f(x,y):
    return x**2 + y**2

for i in gams.get("num_columns"):
    num_columns = int(i)
for i in gams.get("num_rows"):
    num_rows = int(i)
for i in gams.get("x_u"):
    x_u = i
for i in gams.get("x_l"):
    x_l = i
for i in gams.get("y_u"):
    y_u = i
for i in gams.get("y_l"):
    y_l = i

#################### AUX Functions  ####################
def range_plus(start, stop, num_steps):
    range_size = stop-start
    step_size = float(range_size)/(num_steps-1)
    for step in range(num_steps):
        yield round(start + step*step_size,3)

def congruent_modulo(a,b,n):
    return a%n == b%n

def iseven(num):
    if num % 2 == 0:
        return True # Even
    else:
        return False # Odd

def S_1(I,J):
    S1 = []
    for i in I:
        for j in J:
            if congruent_modulo(i,j,2) & congruent_modulo(i+j,2,4):
                S1.append((i+1,j+1))
    return S1
def S_2(I,J):
    S2 = []
    for i in I:
        for j in J:
            if congruent_modulo(i,j,2) & congruent_modulo(i+j,0,4):
                S2.append((i+1,j+1))
    return S2
def S_3(I,J):
```

```
    S3 = []
    for i in I:
        for j in J:
            if (not congruent_modulo(i,j,2)) & congruent_modulo(i+j,3,4):
                S3.append((i+1,j+1))
    return S3
def S_4(I,J):
    S4 = []
    for i in I:
        for j in J:
            if (not congruent_modulo(i,j,2)) & congruent_modulo(i+j,1,4):
                S4.append((i+1,j+1))
    return S4

def S_1_jk(I,J):
    S1 = []
    for i in I:
        for j in J:
            if ((iseven(i) == True) & (iseven(j) == False)):
                S1.append((i+1,j+1))
    return S1

def S_2_jk(I,J):
    S2 = []
    for i in I:
        for j in J:
            if ((iseven(i) == False) & (iseven(j) == True)):
                S2.append((i+1,j+1))
    return S2

def C_matrix(R):
    C = {}
    C[1] = np.array([[0], [1]])
    for r in range(1, int(R)+1):
        C[r+1] =
            np.vstack((
                np.hstack((C[r], np.zeros((2**r, 1)))),
                np.hstack((C[r] +
                    np.ones((2**r, 1)) * C[r][2**r-1],
                    np.ones((2**r, 1))
                    ))
            ))
    return C
##################### End AUX Functions  ####################


I = list(range(1,num_columns+1))
J = list(range(1,num_rows+1))

s1 = []
s1 = S_1(I,J)
gams.set("s1",s1)

s2 = []
s2 = S_2(I,J)
gams.set("s2",s2)

s3 = []
s3 = S_3(I,J)
gams.set("s3",s3)

s4 = []
```

```
s4 = S_4(I,J)
gams.set("s4",s4)

s1_jk = []
s1_jk = S_1_jk(I,J)
gams.set("s1_jk",s1_jk)

s2_jk = []
s2_jk = S_2_jk(I,J)
gams.set("s2_jk",s2_jk)

x_grid = list(range_plus(x_l,y_u,num_columns))
aux = []
for i,val in enumerate(x_grid):
    aux.append((i+2,val))
gams.set("x_grid",aux)

y_grid = list(range_plus(x_l,x_u,num_rows))
aux = []
for j,val in enumerate(y_grid):
    aux.append((j+2,val))
gams.set("y_grid",aux)


auxZ = []
for i,x_val in enumerate(x_grid):
    for j,y_val in enumerate(y_grid):
        z_val = f(x_val,y_val)
        auxZ.append( (i+2,j+2,z_val) )

gams.set("Z_grid",auxZ)

r = int(np.ceil(np.log2(num_rows    - 1)))
s = int(np.ceil(np.log2(num_columns - 1)))
CRm1 = list(range(1, 2**r+1))
CSm1 = list(range(1, 2**s+1))
C = {}
C = C_matrix(max(r,s))

aux = []
for i in CRm1:
    for j,val in enumerate(C[r][ i-1 ]):
        aux.append((i+1,j+2,val))
gams.set("cr",aux)

aux = []
for i in CSm1:
    for j,val in enumerate(C[s][ i-1 ]):
        aux.append((i+1,j+2,val))
gams.set("cs",aux)
$offEmbeddedCode s1 s2 s3 s4 s1_jk s2_jk x_grid y_grid z_grid cr cs
```

Convex Combination Equations: Implementation of the equations 3.6, in charge of approximating the non-linear function. The GAMS variables x,y and z should be used in other constraints to model the desired behavior, (i.e. Li-Ion losses).

```
eq_SOS2_1    ..
    x        =E= sum((i,j), x_grid(i)*theta(i,j) )
;
eq_SOS2_2    ..
    y        =E= sum((i,j), y_grid(j)*theta(i,j) )
;
eq_SOS2_3    ..
    z        =E= sum((i,j), z_grid(i,j)*theta(i,j) )
;
eq_SOS2_4    ..
    1        =E= sum((i,j),theta(i,j) )
;
```

Column & Row selection: This section declares the Zig-Zag Integer equations 3.21 and 3.22 as well as the Text Book equations 3.8 and 3.9. This code section could be replaced for the ZZB method, or other SOS2 alternatives.

```
*Row Selection

*ZigZag Integer formulation:
eq_zigzag_a1(r)..
sum((i)  , cr["1" ,r]                              *theta(i,"1"))  +
sum((i,j), cr[j-1 ,r]$(not j.first                )*theta(i, j )) =l= chi_r(r)
;
eq_zigzag_a2(r) ..
sum((i,j), cr[j  ,r]$(not j.last                  )*theta(i,j))    +
sum((i,j), cr[j-1,r]$(    j.last                  )*theta(i,j))   =g= chi_r(r)
;

*TextBook formulation:
eq_zigzag_a1_bm..
sum(j, chi_r(j)   )        =e=   1
;
eq_zigzag_a2_bm(j)..
sum(i, theta(i,j) )        =l= chi_r(j)$(not j.last) + chi_r(j-1)$(not j.first)
;

************************************************
*Column Selection


eq_zigzag_b1(s) ..
sum((j)  , cs["1",s]                               *theta("1",j))        +
sum((i,j), cs[i-1,s]$(not i.first                 )*theta(i  ,j))       =l= chi_s(s)
;
eq_zigzag_b2(s) ..
sum((i,j), cs[i  ,s]$(not i.last                        )*theta(i,j))   +
sum((i,j), cs[i-1,s]$(    i.last                        )*theta(i,j))  =g= chi_s(s)
;

*TextBook formulation:
eq_b1_bm..
sum(i, chi_s(i)   )        =e=   1
;
eq_b2_bm(i)..
sum(j, theta(i,j) )        =l= chi_s(i)$(not i.last) + chi_s(i-1)$(not i.first)
;
```

Triangle Selection Equations: The triangle constraint equations are implemented. In this case, both the K1 & J1 triangulation methods are defined, but when solved only the desired equations should be included.

```
*K1 triangulation
eq_trian1..
z1   =g= sum(s1(i,j),theta(i,j) )
;
eq_trian2..
1-z1 =g= sum(s2(i,j),theta(i,j) )
;
eq_trian3..
z2   =g= sum(s3(i,j),theta(i,j) )
;
eq_trian4..
1-z2 =g= sum(s4(i,j),theta(i,j) )
;
***********************************************
*J1 triangulation
eq_trian1_jk..
z1   =g= sum(s1_jk(i,j),theta(i,j) )
;
eq_trian2_jk..
1-z1 =g= sum(s2_jk(i,j),theta(i,j) )
;
```

Babayev constraints: Lastly, the set of constraints responsible for modeling the Babayev method are presented.

```
eq_babayev_a1(i,j)..
 theta(i,j)   =l= chi_w(i  ,j  )              + chi_u(i  ,j) +
                  chi_w(i  ,j+1)$(not j.last) + chi_u(i  ,j-1)$(not j.first) +
                  chi_w(i+1,j  )$(not i.last) + chi_u(i-1,j  )$(not i.first)
;
eq_babayev_a2..
sum((i,j), chi_u(i,j) + chi_w(i,j)  ) =E= 1
;
```

3.3.3 Verification of the Implementation

Due to the large amount of parameter, set and constraints used for the model implementation, testing is crucial. To verify the implementation, 3 tests were conducted. The first one was comparing the GAMS equations with the equations generated by the Julia Lang library PiecewiseLinearOpt. Ensuring that the constraints that modeled the column and row selection as well as the triangulation selection are correctly implemented. In listing 3.1 the Julia Lang script equivalent to the GAMS implementation for the ZZI formulation is presented. And thanks to the "print" instruction, the constraint generated by the "piecewiselin-

ear" function are display in the terminal for its comparation with the with those generated by GAMS. In Appendix 5 Julia and PiecewiseLinearOpt is presented with more detail.

The second test conducted was by forcing the value of the variables x and y and analyzing the values that the variables ζ,z_1,z_2 took. Lastly a visual analysis was conducted, by plotting the bi-vatiate non-linear function, the linearized version as well as the GAMS variables results of x, y and z.

Listing 3.1: Bivariate pwlf implementation with Julia & PiecewiseLinearOpt.

```
using JuMP, Gurobi, PiecewiseLinearOpt

#Function and domain declaration
num_columns = 4
num_rows    = 4
x_u = 10
x_l = 0
y_u = 10
y_l = 0
x_range = range(x_l,x_u,num_columns)
y_range = range(y_l,y_u,num_rows)
f(x) = x^2 + y^2

#Model inicialization
model = Model(Gurobi.Optimizer)

#Variable declaration
@variable(model, x)
@variable(model, y)

#K1 Triangulation
z = piecewiselinear(m, x, y, BivariatePWLFunction(x_range, y_range, f, pattern=:
K1),method=:ZigZagInteger)

#J1 Triangulation
z = piecewiselinear(m, x, y, BivariatePWLFunction(x_range, y_range, f, pattern=:
UnionJack),method=:ZigZagInteger)

print(model)
```

Chapter 4

Optimal microgrid management

In this chapter, the concepts described in chapters 2 & 3 are going to be applied to model the optimal management of an electric power microgrid leveraging classical optimization. The goal of this chapter is to devise the performance of the linearization methods described in chapter 3 and assess the viability of its application in the accurate modeling of the losses of a Li-Ion battery as described in chapter 2.

4.1 Problem Statement

Renewable sources of energy based on solar or wind energy have become crucial for the decarbonization of the electric power industry, but the stochastic nature of its power production is limiting its integration into the energy mix. During peak renewable generation periods, excess energy leads to curtailment, while during periods of low renewable energy leads to procure the energy from other sources such as gas or coal power plants. Energy storage solutions are expected to solve this issues by storing excess solar and wind energy and offloading it to the grid during high energy demand periods, acting as a buffer and therefore reducing the stochastic impact of solar and wind power generation. Therefore, the correct modeling of energy storage is crucial for their integration into the energy grid and the consequent decarbonization of the electrical sector.

The current technologies available for energy storage are widely diverse, but the most relevant at power system level are Pumped Storage Hydropower, and Li-Ion batteries. Pumped Storage Hydropower has been available since the 20th century, being the firstly used in Switzerland in 1907. Literature for modeling this technology is extensive, as it presents non-linearities that make finding an optimal solution complex. For example, [23] explore the application of the Zig-Zag Integer formulation presented in 3 to linearize the relationship between the

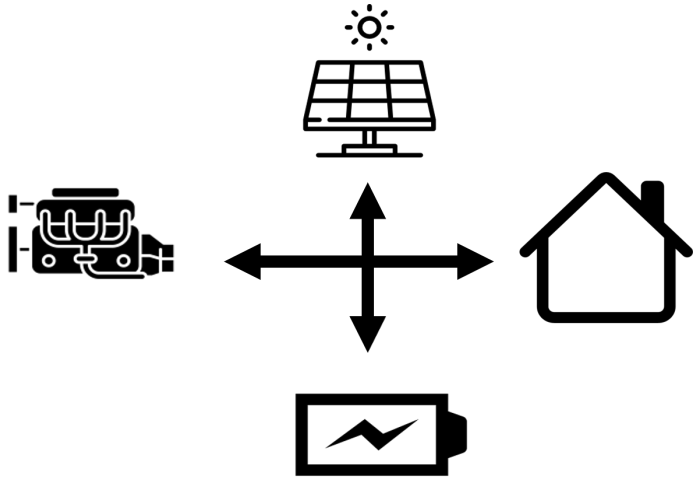


Figure 4.1: Microgrid diagram.

power generated/consumed and the water flow and level of the reservoir.

The goal of this model is the optimal schedule for a residential isolated microgrid comprising a solar panel, a diesel generator, and a Li-Ion battery. The approach taken to tackle this problem is a deterministic MIQP classical optimization problem, were given a set of operation constraints (i.e. maximum power outputs) the operation cost of the system is minimized.

4.2 Mathematical Model

4.2.1 Variables

e_t	Energy stored at the end t [kWh]
p_t^{pv}	Power produced by the PV panel in t [kW]
p_t^{d}	Power produced by the Diesel generator in t [kW]
p_t^{pns}	Non-served power in period t [kW]
u_t^{d}	Commitment of the diesel generator in t {0,1}
u_t^{disc}	Battery discharge decision in t {0,1}
u_t^{cha}	Battery charge decision in t {0,1}
soc_t	Battery state of charge in t {0,1}

Separate Charge & Discharge model variables

p_t^{char} Power consumed by the battery in t [kW]

p_t^{disc} Power generated by the battery in t [kW]

$p_{loss,t}^{cha}$ Power losses when charging in t [kW]

$p_{loss,t}^{disc}$ Power losses when discharging in t [kW]

Combined Charge & Discharge model variables

p_t^{batt} Power generated & consumed by the battery in t [kW]

$p_{loss,t}^{batt}$ Power losses during the battery charge and discharge cycles in t [kW]

4.2.2 Sets

$t \in \mathcal{T}$ time periods {1 to T}

4.2.3 Parameters

D_t Demand at each time period t, [kW]

Δ_t Duration of each time period t, [h]

a, b, c Quadratic, linear, and constant terms of the diesel cost function, [€/h/(kW)^2], [€/kWh], [€/h]

E_o Initial energy stored at the battery, [kWh]

$\overline{P}^d$ Maximum capacity of the diesel generator, [kWh]

$\overline{P}^{disc}$ Maximum discharge power of the battery, [kW]

$\overline{P}^{cha}$ Maximum charge power of the battery, [kW]

$\overline{E}$, $\underline{E}$ Maximum and minimum stored energy [kWh]

4.2.4 Equations

Objective Function

The only costs taken into consideration in this problem are the fuel and the start-up costs associated with the diesel generation, as well as the non-served energy cost. The fuel cost as it can be observed in equation 4.1 is a quadratic function dependent on the unit commitment of the generator and the produced power. Although the linearization techniques described in chapter 3 could be applied to model this quadratic function, it is going to be solved directly with a non-linear solver.

$$\min \sum_{t} [a \cdot (p_t^d)^2 + b \cdot p_t^d + c \cdot u_t^d + c_{pns} \cdot p_t^{pns}] \cdot \Delta_t \tag{4.1}$$

Generation

The model for the diesel generation is a continuous positive variable whose only limitation is its upper bound, which will be controlled by a binary variable (u_t^d) that represents whether the diesel generation is on or off. Regarding the solar generation, the modeling approach will be similar to the diesel generator. A continuous positive variable represents the power injected in the system (p_t^{pv}). This variable has a parametric upper bound whose value varies over time and represents the maximum available solar energy (P_t^{pv}).

$$p_t^d \leq u_t^d \cdot \overline{P}^d, \forall t \in T \tag{4.2}$$

$$0 \leq p_t^{pv} \leq P_t^{pv}, \forall t \in T \tag{4.3}$$

Li-Ion Operation

For modeling the Li-Ion operation, two alternative models are going to be considered, Separate Charge & Discharge Model and Combined Charge & Discharge Model. Both models are based in the Li-Ion loss functions presented in chapter 2 and the linearization methods described in chapter 3.

Separate Charge & Discharge Model

This model considers the charge and discharge of the battery as two separate variables (p^{disc},p^{char}). For this reason, there will be the corresponding charge and discharge loss variables (p_{loss}^{disc},p_{loss}^{char}) and the corresponding charge discharge loss functions.
As an illustrative example, the linearization technique presented in the equations

is the Zig-Zag Integer J1 method, although equations 4.7, 4.8, 4.9 & 4.11, 4.12, 4.13 could be replaced for the other linearization methods.

$$P_{loss}^{disc} \approx 10^3(R + \frac{K}{SOC})(\frac{P^{disc}}{V_r})^2 \tag{4.4}$$

$$P_{loss}^{char} \approx 10^3(R + \frac{K}{1.1 - SOC})(\frac{P^{char}}{V_r})^2 \tag{4.5}$$

Battery Discharge Loss:

$$soc_t = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt}^{disc} \cdot SOC_i \quad \forall \quad t \in T \tag{4.6a}$$

$$p_t^{disc} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt}^{disc} \cdot P_j^{disc} \quad \forall \quad t \in T \tag{4.6b}$$

$$p_{loss,t}^{disc} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt}^{disc} \cdot P_{loss}^{disc}(SOC_i, P_j^{disc}) \quad \forall \quad t \in T \tag{4.6c}$$

$$\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt}^{disc} = 1 \quad \forall \quad t \in T \tag{4.6d}$$

$$0 \leq \theta_{ijt}^{disc} \leq 1 \quad \forall \quad t \in T \tag{4.6e}$$

$$\theta_{ijt}^{disc}, soc_t, p_t^{disc} \in \mathbb{R} \tag{4.6f}$$

$$\sum_{i=1}^{n+1} C_{i-1,k}^{r} \sum_{j=1}^{m+1} \theta_{ijt}^{disc} \leq \zeta_{kt}^{r,disc} \leq \sum_{i=1}^{n+1} C_{i,k} \sum_{j=1}^{m+1} \theta_{ijt}^{disc} \quad \forall \quad k = [1, r], t \in T \tag{4.7a}$$

$$0 \leq \zeta_{kt}^{r,disc} \leq 2^{r-k} \quad \forall \quad k = [1, r], t \in T \tag{4.7b}$$

$$\zeta_{kt}^{r,disc} \quad \forall \quad k = [1, r] \in \mathbb{Z} \tag{4.7c}$$

$$\sum_{j=1}^{m+1} C_{j-1,k}^{s} \sum_{i=1}^{n+1} \theta_{ijt}^{disc} \leq \zeta_{kt}^{c,disc} \leq \sum_{j=1}^{m+1} C_{j,k}^{s} \sum_{i=1}^{n+1} \theta_{ijt}^{disc} \quad \forall \quad k = [1, s], t \in T \tag{4.8a}$$

$$0 \leq \zeta_{kt}^{c,disc} \leq 2^{r-k} \quad \forall \quad k = [1, s], t \in T \tag{4.8b}$$

$$\zeta_{kt}^{c,disc} \quad \forall \quad k = [1, s] \in \mathbb{Z} \tag{4.8c}$$

$$\sum_{(i,j)\in S_1} \theta_{ijt}^{disc} \leq z_{1t}^{disc} \qquad \sum_{(i,j)\in S_2} \theta_{ijt}^{disc} \leq 1 - z_{1t}^{disc} \quad \forall \quad t \in T$$
$$z_{1,t}^{disc}, z_{2,t}^{disc} \in \{0,1\} \tag{4.9}$$

$$S1 = \{(i,j) : \text{i is even and j is odd}\}$$
$$S2 = \{(i,j) : \text{j is even and i is odd}\}$$
$$\forall i \in [1..n+1]$$
$$\forall j \in [1..m+1]$$

Battery Charge Loss:

$$soc_t = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt} \cdot SOC_i \quad \forall \quad t \in T \tag{4.10a}$$

$$p_t^{char} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt} \cdot P_j^{char} \quad \forall \quad t \in T \tag{4.10b}$$

$$p_{loss,t}^{char} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt} \cdot P_{loss}^{char}(SOC_i, P_j^{char}) \quad \forall \quad t \in T \tag{4.10c}$$

$$\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt}^{char} = 1 \quad \forall \quad t \in T \tag{4.10d}$$

$$0 \leq \theta_{ijt}^{char} \leq 1 \quad \forall \quad t \in T \tag{4.10e}$$

$$\theta_{ijt}^{char}, soc_t, p_t^{char} \in \mathbb{R} \tag{4.10f}$$

$$\sum_{i=1}^{n+1} C_{i-1,k}^{r} \sum_{j=1}^{m+1} \theta_{ijt} \leq \zeta_{kt}^{char,r} \leq \sum_{i=1}^{n+1} C_{i,k} \sum_{j=1}^{m+1} \theta_{ijt} \quad \forall \quad k = [1,r], t \in T \tag{4.11a}$$

$$0 \leq \zeta_{kt}^{char,r} \leq 2^{r-k} \quad \forall \quad k = [1,r], t \in T \tag{4.11b}$$

$$\zeta_{kt}^{char,r} \quad \forall \quad k = [1,r] \in \mathbb{Z} \tag{4.11c}$$

$$\sum_{j=1}^{m+1} C_{j-1,k}^{s} \sum_{i=1}^{n+1} \theta_{ijt} \leq \zeta_{kt}^{char,c} \leq \sum_{j=1}^{m+1} C_{j,k}^{s} \sum_{i=1}^{n+1} \theta_{ijt} \quad \forall \quad k = [1,s], t \in T \tag{4.12a}$$

$$0 \leq \zeta_{kt}^{char,c} \leq 2^{r-k} \quad \forall \quad k = [1,s], t \in T \tag{4.12b}$$

$$\zeta_{kt}^{char,c} \quad \forall \quad k = [1, s] \in \mathbb{Z} \tag{4.12c}$$

$$\sum_{(i,j)\in S_1} \theta_{ijt}^{char} \leq z_{1t} \qquad \sum_{(i,j)\in S_2} \theta_{ijt}^{char} \leq 1 - z_{1t} \quad \forall \quad t \in T$$
$$z_{1,t}, z_{2,t} \in \{0, 1\} \tag{4.13}$$
$$S1 = \{(i, j) : \text{i is even and j is odd}\}$$
$$S2 = \{(i, j) : \text{j is even and i is odd}\}$$
$$\forall i \in [1..n + 1]$$
$$\forall j \in [1..m + 1]$$

Equation 4.14 is responsible for modeling the storage of energy from one period to the next while considering the charge and discharge of the battery and the loss in the process.

$$e_t = e_{t-1} + [(p_t^{char} - p_{loss,t}^{char}) - (p_t^{disc} + p_{loss,t}^{disc})]\Delta_t, \forall t \in T \tag{4.14}$$

$$soc_t = \frac{\frac{1}{2}(e_{t-1} + e_t)}{\overline{E}} \tag{4.15}$$

Due to periods where there is excess PV energy available, and therefore the system marginal cost is 0, the optimizer might decide to charge and discharge the battery simultaneously. To prevent this behavior, equation 4.16 uses two auxiliary binary variables (u_t^{disc} & u_t^{char}) that only allows the charge or the discharge of the battery.

$$p_t^{char} \leq u_t^{cha} \cdot \overline{P}^{cha}, \forall t \in T \tag{4.16a}$$
$$p_t^{disc} \leq u_t^{disc} \cdot \overline{P}^{disc}, \forall t \in T \tag{4.16b}$$
$$u_t^{disc} + u_t^{char} \leq 1, \forall t \in T \tag{4.16c}$$

Combined Charge & Discharge Model

The proposed improvement of the model will use the continuous variable p^{batt} to model the charge and the discharge of the battery simultaneously. Considering that the battery is injecting energy to the system if $p^{batt} < 0$ and consuming energy if $p^{batt} > 0$. The charge/discharge loss function is going to be modeled as a piecewise function, where if $p^{batt} \geq 0$ the function will be the charge loss function, and if $p^{batt} < 0$ the function will be the discharge loss function. This new combined loss function is going to be linearized using the same methods as described in chapter 3.

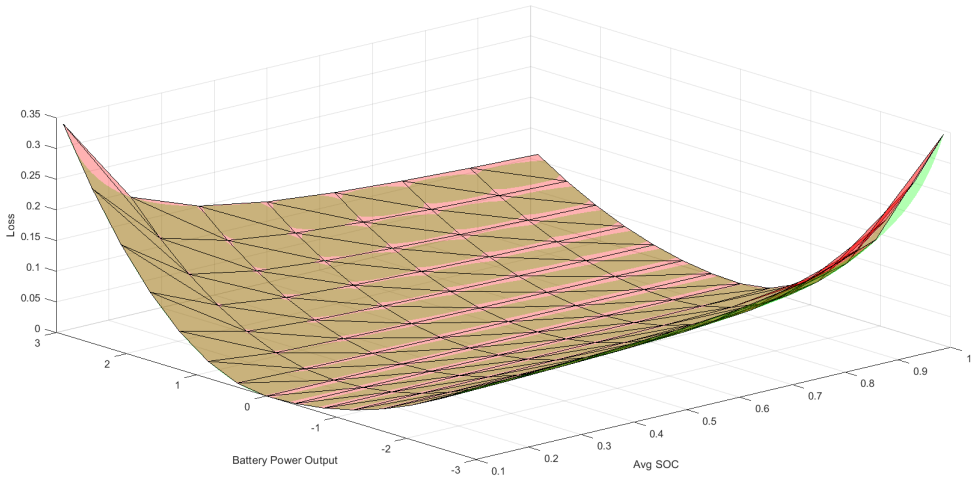


Figure 4.2: Combined Battery Loss linearization using the K1 triangulation

$$p^{batt} \geq 0 \longrightarrow : \text{Battery charge}$$
$$p^{batt} < 0 \longrightarrow : \text{Battery discharge}$$

$$P_{loss}^{batt} \approx \begin{cases} 10^3(R + \frac{K}{1.1-SOC})(\frac{P^{batt}}{V_r})^2 & \text{if } p^{batt} \geq 0 \\ 10^3(R + \frac{K}{SOC})(\frac{P^{batt}}{V_r})^2 & \text{if } p^{batt} < 0 \end{cases} \tag{4.17}$$

As described in chapter 3 the number of integer variables is correlated with the computation burden of the problem. As both the charge and discharge mode are functions dependent on soc_t, there is no need to model them as a separate, independent function. For this reason, the hypothesis is that for the presented linearization techniques for bivariate functions, if the two pwlf for modeling the charge and discharge loss are combined, the number of integer variables will be reduced and, in theory, the computation time will improve.

In the case of the Zig-Zag Integer method, the reduction will be greater due to the logarithmic relationship between the resolution of the triangulation grid and the number of auxiliary integer variables. If the loss function is modeled as two separate functions with the same resolution, the resulting triangulation of the domain would be two grids of $(n+1) \times (m+1)$, and the model would have $TimeFrames*2(log_2(m)+log_2(n)+1)$ auxiliary integer variables. If the proposed loss function is modeled with the same resolution, the resulting domain could be triangulated with a $(n+1) \times (2*m)$ grid which needs $TimeFrames*(log_2(2*m-1)+log_2(n)+1)$ auxiliary integer variables, a considerable reduction. Additionally,

the two binary variables that prevent the simultaneous charge and discharge of the battery (u_t^{disc} & u_t^{char}) and its constraints (4.16) are no longer needed, as the model do not allow charging and discharging simultaneously, reducing even more the problem size.

$$soc_t = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt} \cdot SOC_i \quad \forall \quad t \in T \tag{4.18a}$$

$$p_t^{batt} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt} \cdot P_j^{batt} \quad \forall \quad t \in T \tag{4.18b}$$

$$p_{loss,t}^{batt} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt} \cdot P_{loss}^{batt}(SOC_i, P_j^{batt}) \quad \forall \quad t \in T \tag{4.18c}$$

$$\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \theta_{ijt} = 1 \quad \forall \quad t \in T \tag{4.18d}$$

$$0 \leq \theta_{ijt}^{batt} \leq 1 \quad \forall \quad t \in T \tag{4.18e}$$

$$\theta_{ijt}, soc_t, p_t^{batt} \in \mathbb{R} \tag{4.18f}$$

$$\sum_{i=1}^{n+1} C_{i-1,k}^{r} \sum_{j=1}^{m+1} \theta_{ijt} \leq \zeta_{kt}^{r} \leq \sum_{i=1}^{n+1} C_{i,k} \sum_{j=1}^{m+1} \theta_{ijt} \quad \forall \quad k = [1, r], t \in T \tag{4.19a}$$

$$0 \leq \zeta_{kt}^{r} \leq 2^{r-k} \quad \forall \quad k = [1, r], t \in T \tag{4.19b}$$

$$\zeta_{kt}^{r} \quad \forall \quad k = [1, r] \in \mathbb{Z} \tag{4.19c}$$

$$\sum_{j=1}^{m+1} C_{j-1,k}^{s} \sum_{i=1}^{n+1} \theta_{ijt} \leq \zeta_{kt}^{c} \leq \sum_{j=1}^{m+1} C_{j,k}^{s} \sum_{i=1}^{n+1} \theta_{ijt} \quad \forall \quad k = [1, s], t \in T \tag{4.20a}$$

$$0 \leq \zeta_{kt}^{c} \leq 2^{r-k} \quad \forall \quad k = [1, s], t \in T \tag{4.20b}$$

$$\zeta_{kt}^{c} \quad \forall \quad k = [1, s] \in \mathbb{Z} \tag{4.20c}$$

$$\begin{gathered} \sum_{(i,j)\in S_1} \theta_{ijt} \leq z_{1t} \qquad \sum_{(i,j)\in S_2} \theta_{ijt} \leq 1 - z_{1t} \quad \forall \quad t \in T \\ z_{1,t}, z_{2,t} \in \{0, 1\} \end{gathered} \tag{4.21}$$

$$S1 = \{(i,j) : \text{i is even and j is odd}\}$$
$$S2 = \{(i,j) : \text{j is even and i is odd}\}$$
$$\forall i \in [1..n+1]$$
$$\forall j \in [1..m+1]$$

The battery energy balance constraint will be reduced to:

$$e_t = e_{t-1} + [p_t^{batt} - p_{loss,t}^{batt}]\Delta_t, \forall t \in T \tag{4.22}$$

$$soc_t = \frac{\frac{1}{2}(e_{t-1} + e_t)}{\overline{E}} \tag{4.23}$$

Load Balance

The load balance constraint ensures that the energy demand meets the energy generation for all periods. In addition to the generation, a slack variable is introduced (p^{pns}) to model the non-served energy. Constraint 4.24 will model the Li-Ion battery in the "Separate Charge & Discharge" while constraint 4.25 will be only used in the "Combined Charge & Discharge" model.

$$p_t^{pv} + p_t^d + p_t^{disc} - p_t^{char} + p_t^{pns} = D_t, \forall t \in T \tag{4.24}$$

$$p_t^{pv} + p_t^d - p_t^{batt} + p_t^{pns} = D_t, \forall t \in T \tag{4.25}$$

Variable Bounds

$$\begin{aligned}
\underline{E} \leq e_t &\leq \overline{E} \quad \forall t \in T \\
\underline{soc} \leq soc_t &\leq \overline{soc} \quad \forall t \in T \\
0 \leq p_t^{disc} &\leq \overline{P}^{disc} \forall t \in T \\
-P^{disc} \leq p_t^{batt} &\leq \overline{P}^{cha} \forall t \in T \\
0 \leq p_t^{char} &\leq \overline{P}^{cha} \forall t \in T \\
0 \leq p_t^{pns}, & \quad \forall t \in T
\end{aligned} \tag{4.26}$$

4.3 Data

The characteristics of the microgrid showed in table 4.1 were sourced from [8]. Regarding the battery parameters, they were estimated from [3] where the author measured the losses of a microgrid operation where an electric vehicle was used as the battery storage of the power system. A brute force approach was

Table 4.1: Microgrid parameters

Diesel	$P^d_{\max}$	1.0	[kW]
	c	0.0157	[€]
	b	0.1080	[€/kW]
	a	0.3100	[€/kW^2]
Li-Ion battery	$\overline{E}$	2.9	[kWh]
	$\underline{E}$	0	[kWh]
	$\overline{soc}$	100	[%]
	$\underline{soc}$	10	[%]
	e_0	0	[kWh]
	$\overline{P}^{cha}$	2.9	[kW]
	$\overline{P}^{disc}$	2.9	[kW]
	K	8.0625	[mΩ]
	R	26.46	[mΩ]
	V_r	51.2	[V]
	c_{ens}	1	[€/kWh]

used to fit the data points provided in the paper to the battery loss equations (2.5 & 2.4) presented in chapter 3.

Regarding the two time variant parameters, the load and the photovoltaic available power, a winter and a summer data set will be used, referred to with the notation _w and _s respectively.

Lastly, the simulation horizon will vary according to the case. 3 timeframes are going to be study 48 hours, 96 hours and 168 hours all with hourly resolution.

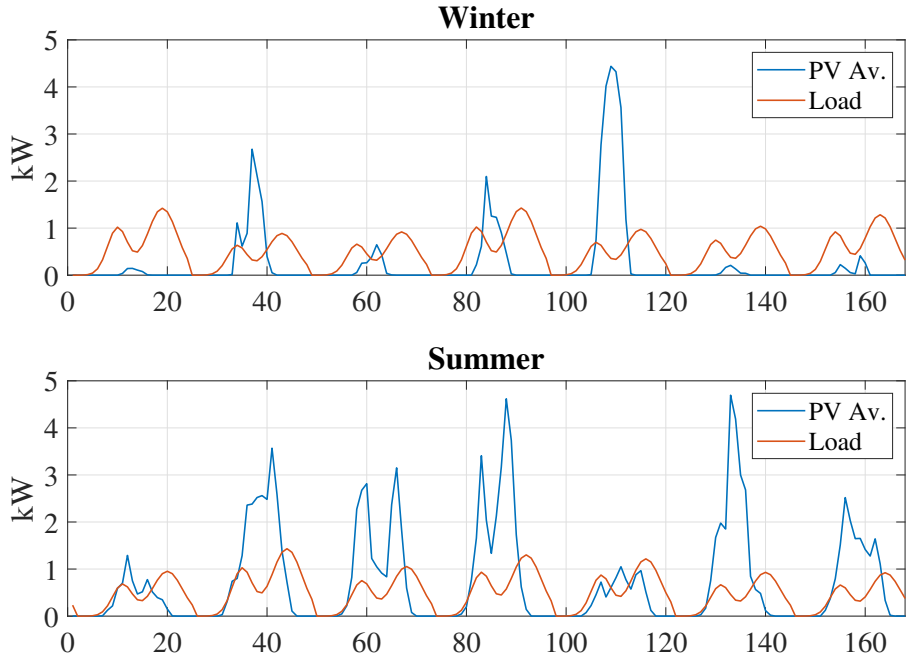


Figure 4.3: Hourly load and available PV.

4.4 Results

To evaluate the performance of the various models and linearization techniques presented in this master thesis, a series of simulations were conducted. From the perspective of the power system, models with the same grid granularity and triangulation are equivalent; the only difference lies in the linearization approach.

The study will primarily focus on comparing the solving time for the different linearization methods presented in chapter 3, the two approaches for modeling batteries presented in chapter 4, and different commercial solvers (Gurobi 10.0.3 & Cplex 22.1.1.0).

Table 4.2: Linearization Methods analyzed.

Method	Description
ZZI-J1	ZZI method with J1 triangulation
ZZI-K1	ZZI method with K1 triangulation
C-K1	Babayev [4]
BM-J1	Text Book method with J1 triangulation
BM-K1	Text Book method with K1 triangulation

4.4.1 Operation Analysis

Since all the compared models share very similar physical characteristics, their operations are also almost identical. Thus, this part of the study focuses solely on simulated cases with different grid granularity.

Battery Operations Points

Figures 4.6,4.7 & 4.8 depict the operational points of the battery for various grid granularity (4x4, 8x8, 16x16) using the J1 patterns. As anticipated, most operation points are situated in areas with minimal losses. During charging, there are instances where the optimizer opts to operate the battery in regions with higher losses. This decision can be rationalized by the fact that unutilized solar power would otherwise be lost without any cost, making it worthwhile to operate in zones with higher losses during charging cycles. Conversely, during discharge, energy incurs a cost at the marginal system price, therefore energy losses has a cost for the system. The optimizer tends to minimize losses, as 1kWh lost today in discharged losses might reduce the operational cost tomorrow. This behavior is evident in figure 4.5, where the dispersion in losses during charging is greater than during discharge.

When comparing the operation point with the different grid granularity, as expected, the operation points are situated in the piecewise linear function, not in the real loss function. When analyzing the error of the linearization, as the grid granularity increases, the absolute error is reduced. This can be observed in 4.4, as the granularity increases the histogram of the absolute loss error becomes more right skewed and the average frequency is reduced.

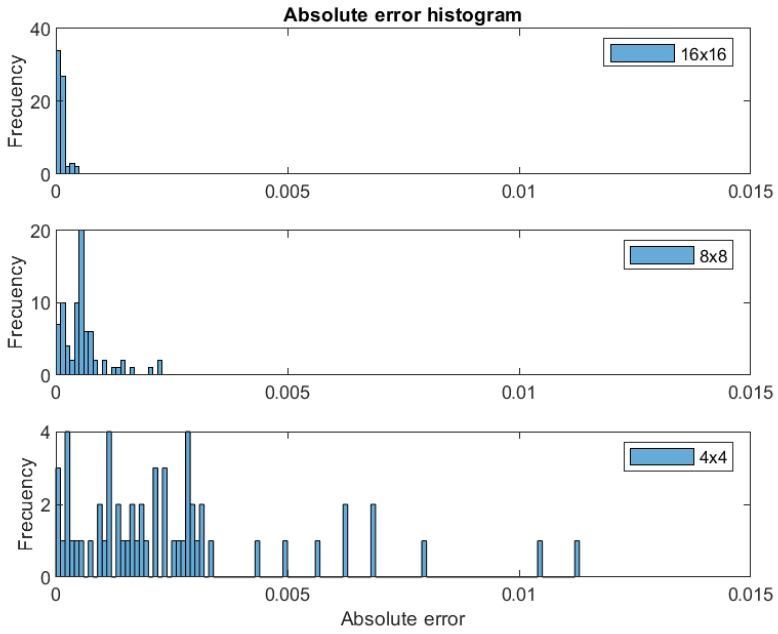


Figure 4.4: Histogram of the absolute charge loss for the studied grid granularities

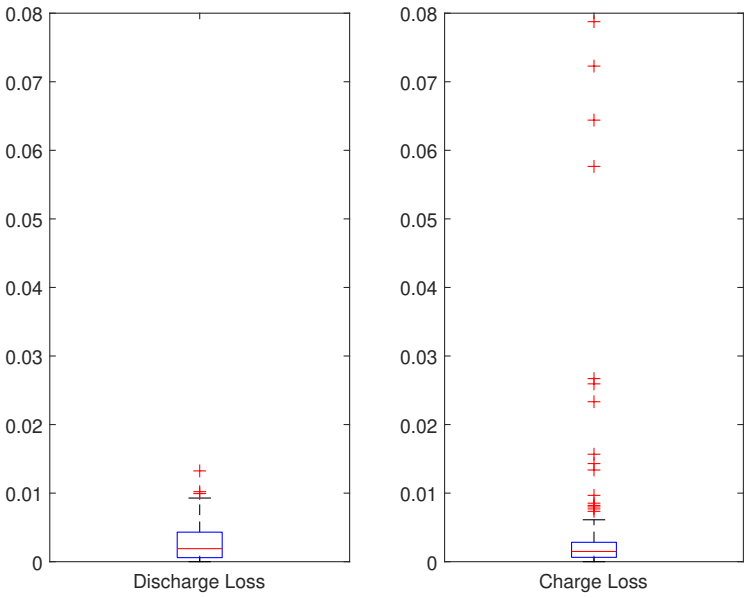


Figure 4.5: Box plot of the charge and discharge losses.

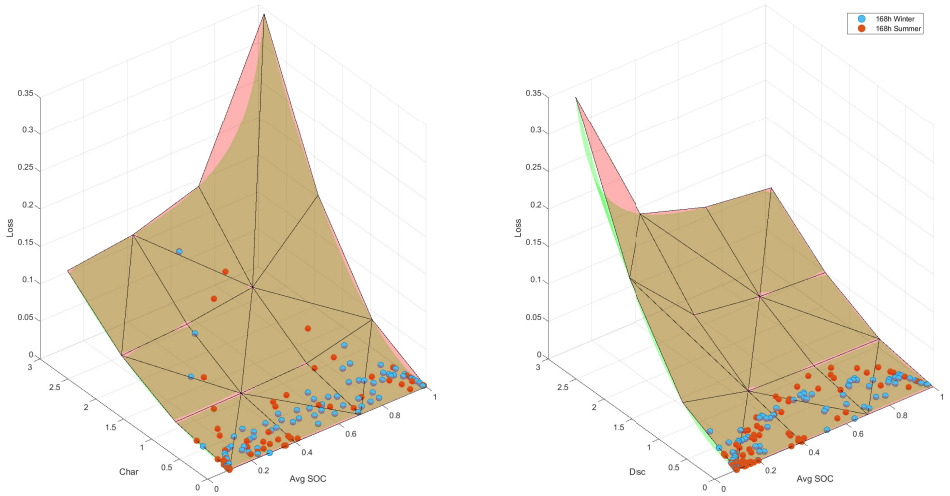


Figure 4.6: Charge and discharge operation points for the 4x4 grid with method ZZI-J1.

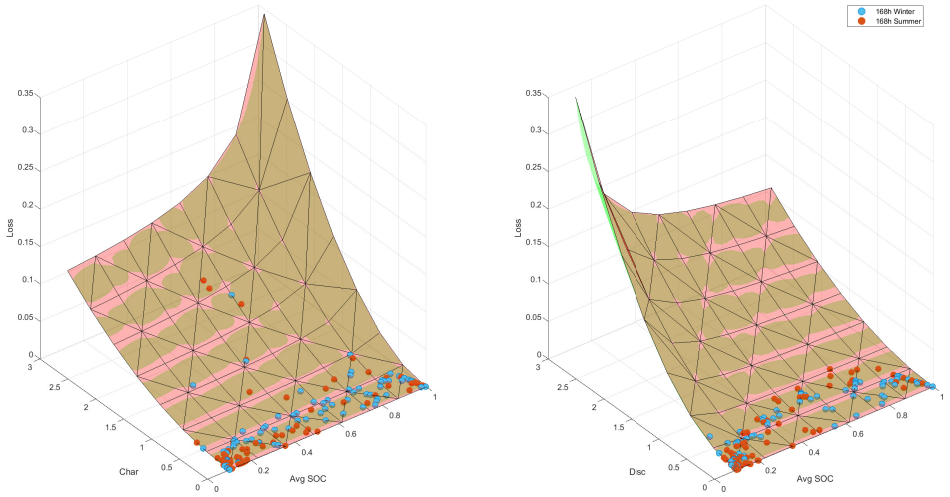


Figure 4.7: Charge and discharge operation points for the 8x8 grid with method ZZI-J1.

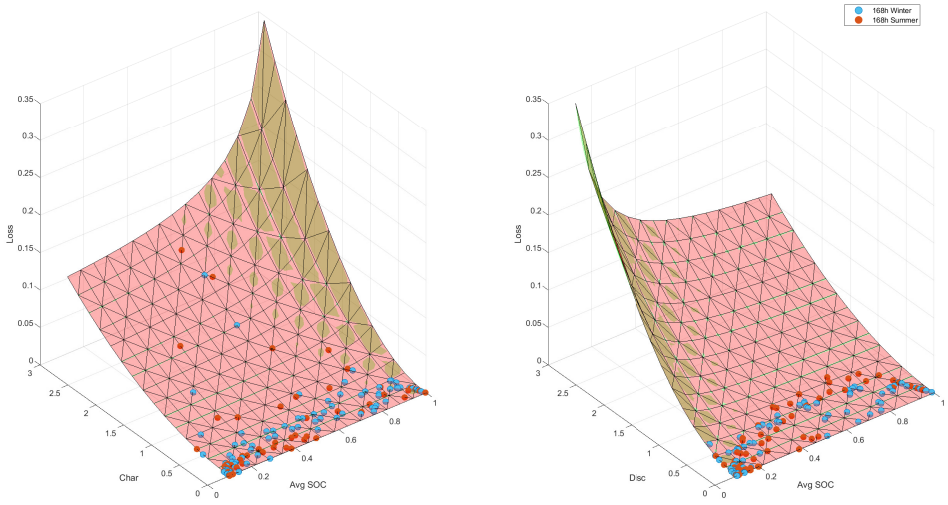


Figure 4.8: Charge and discharge operation points for the 16x16 grid with method ZZI-J1.

Hourly Operation

Regarding hourly operations, the analyzed models effectively manage the microgrid, charging during peak photovoltaic generation and discharging during periods of lower generation to minimize reliance on diesel generation.

A notable contrast emerges when comparing summer and winter simulations. In summer, with greater solar power availability, the battery predominantly charges from this source, with diesel generation activated only during critical hours. Conversely, in winter simulations, the diesel generator is operating almost at a base load. The battery is charged by a combination of diesel and solar energy during this season. The reason for this behavior is the quadratic diesel cost function, which makes each additional kWh from the diesel generator more costly.

This can behavior is easily observed in the plots of the average soc. During the winter simulation, the soc follows the demand profile as the battery is used for peak shaving, while in summer it follows the solar profile.

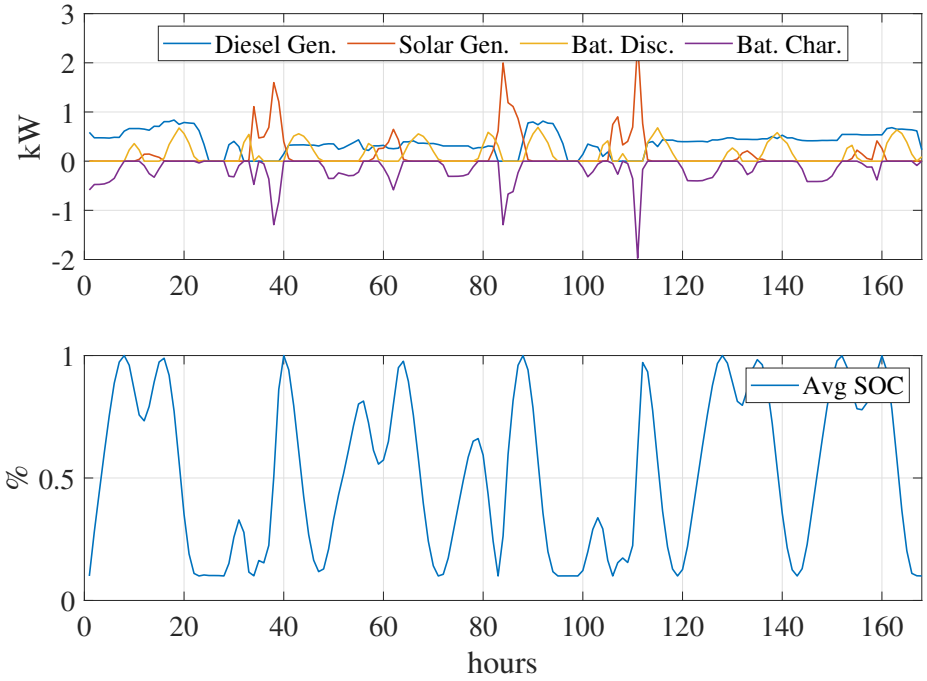


Figure 4.9: Obtained results (ZZI-J1 8×8, GUROBI, Winter).

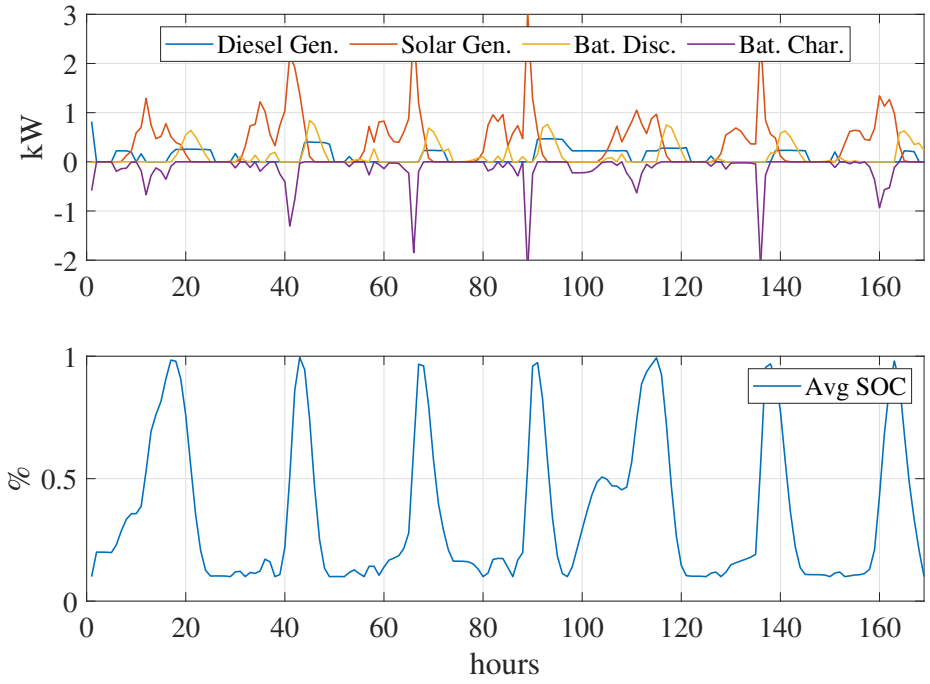


Figure 4.10: Obtained results (ZZI-J1 8×8, GUROBI, Summer).

4.4.2 Computational performance

In this section, emphasis is placed on the solving time of the various models introduced in this master thesis. As elucidated in chapter 3, the linearization methods outlined are rooted in MIQP. Despite the proficiency of commercial solvers in handling large MIQP problems, solving times can be prolonged. Given the substantial number of integer variables employed in the presented models, scrutinizing performance becomes paramount and constitutes the primary focus of the master thesis.

This section of the study will be center on four key areas: problem size comparative, the solver employed, the chosen linearization method, and lastly the battery model.

Problem Size Comparative

As stated throughout this master thesis, the number of integer variables is an accurate proxy for the complexity of a problem. In table 4.3 the number of integer variables in the microgrid model is presented for a case with just 1 time slide, that is 1 hour. The majority of the integer variables of the model are dedicated to model the linearization of the Li-Ion losses, except for the 2 variables that ensure that the battery can only charge or discharge and the variable that model

the unit commitment of the diesel generator.

As expected, the method C-K1 has the biggest number of binary variables, specially as the grid granularity increases, reaching 1000 integer variables for a 16x16 granulation when the battery losses are modeled separated.

When comparing the "Combined" and the "Separated" battery models, the grid of the combined model has been selected, so both models represent exactly the same loss function. It can be observed how the number of integer variables needed for the "Combined" model is considerable smaller than the "Separated" battery models.

Table 4.3: Number of Integer variables comparison.

Method	Grid	Integer Variables
ZZI-J1-Sep	4x4	13
ZZI-K1-Sep	4x4	15
BM-J1-Sep	4x4	17
BM-K1-Sep	4x4	19
C-K1-Sep	4x4	37
ZZI-J1-Sep	16x16	21
ZZI-K1-Sep	16x16	23
BM-J1-Sep	16x16	65
BM-K1-Sep	16x16	67
C-K1-Sep	16x16	901
ZZI-J1-Com	4x7	7
ZZI-K1-Com	4x7	8
BM-J1-Com	4x7	11
BM-K1-Com	4x7	12
ZZI-J1-Com	16x31	11
ZZI-K1-Com	16x31	12
BM-J1-Com	16x31	47
BM-K1-Com	16x31	48

Solver comparison

The initial conclusion drawn from the results underscores the performance gap between the two commercial solvers employed. In general, Gurobi exhibited solving times an order of magnitude faster than Cplex. In certain instances, the disparity widened to two orders of magnitude, and notably, in one case, Cplex failed to yield a valid solution within a one-hour time limit. This disparity in performance remained consistent across the various linearization techniques evaluated.

Table 4.4: Computational performance: Solver comparation

Case	Grid	Method.	Solver	Obj.[€]	Time[s]	Rel.Gap
48h	4x4	ZZI-J1	cplex	6.1153	44.25	0.50%
48h	4x4	ZZI-J1	gurobi	6.1283	1.37	0.40%
48h	8x8	ZZI-J1	cplex	6.0687	572.55	0.50%
48h	8x8	ZZI-J1	gurobi	6.0893	19.56	0.37%
48h	4x4	ZZI-K1	cplex	6.1134	280.08	0.50%
48h	4x4	ZZI-K1	gurobi	6.1260	3.81	0.35%
48h	8x8	ZZI-K1	cplex	6.0688	378.48	0.50%
48h	8x8	ZZI-K1	gurobi	6.0928	10.87	0.44%
48h	4x4	C-K1	cplex	6.1134	333.48	0.50%
48h	4x4	C-K1	gurobi	6.1151	2.89	0.19%
48h	8x8	C-K1	cplex	8.2120	3600.00	28.38%
48h	8x8	C-K1	gurobi	6.0952	229.33	0.48%

Linearization method comparison

The comparison between the different linearization method presented in chapter 3 is not as conclusive as in the solver comparison. A crucial observation lies in the significance of the triangulation pattern; in the majority of cases, J1 triangulation outperforms K1 triangulation. This trend persists even with the implementation of ZZI-K1, which, on paper, should have outperformed the BM-J1 method.

Computation-wise, the most effective method is ZZI with J1 triangulation, especially as grid granularity increases. In larger cases, such as those spanning a week with a 16x16 grid granularity, Gurobi struggles to find solutions with other methods. As anticipated, C-K1 consistently performs the poorest across all scenarios.

In summary, the analysis indicates that ZZI-J1 and BM-J1 are the top performers in terms of solving time. Considering model complexity, BM-J1 might be as a preferable alternative to ZZI-J1, as the latter's approach is more complex to understand and implement.

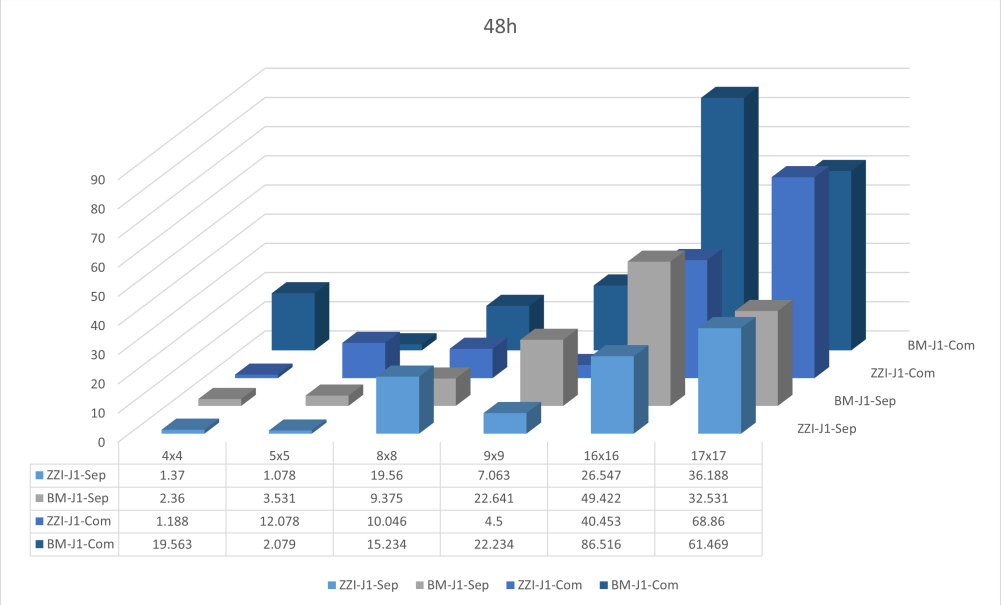
Table 4.5: Computational performance: method comparation

Case	Grid	Method.	Solver	Obj.[k€]	Time[s]	Rel.Gap
48h		ZZI-J1		6.1283	1.37	0.40%
48h		ZZI-K1		6.126	3.81	0.35%
48h	4x4	C-K1	gurobi	6.1151	2.89	0.19%
48h		BM-J1		6.1226	2.36	0.38%
48h		BM-K1		6.1138	1.40	0.25%
48h		ZZI-J1		6.0893	19.56	0.37%
48h		ZZI-K1		6.0928	10.87	0.44%
48h	8x8	C-K1	gurobi	6.0952	229.33	0.48%
48h		BM-J1		6.0856	9.37	0.32%
48h		BM-K1		6.0708	21.92	0.08%
48h		ZZI-J1		6.0817	30.56	0.36%
48h		ZZI-K1		6.0832	71.00	0.39%
48h	16x16	C-K1	gurobi	-	-	-
48h		BM-J1		6.0754	49.42	0.26%
48h		BM-K1		6.0706	21.92	0.37%
168h_W		ZZI-J1		19.1052	40.83	0.38%
168h_W		ZZI-K1		19.1188	272.77	0.48%
168h_W	8x8	C-K1	gurobi	19.1277	3150.87	0.49%
168h_W		BM-J1		19.1022	94.33	0.38%
168h_W		BM-K1		19.1270	333.26	0.50%
168h_W		ZZI-J1		19.0596	224.11	0.26%
168h_W		ZZI-K1		-	-	-
168h_W	16x16	C-K1	gurobi	-	-	-
168h_W		BM-J1		-	-	-
168h_W		BM-K1		-	-	-
168h_S		ZZI-J1		4.8732	133.61	0.29%
168h_S		ZZI-K1		-	-	-
168h_S	8x8	C-K1	gurobi	4.8806	374.41	0.43%
168h_S		BM-J1		4.8777	97.32	0.40%
168h_S		BM-K1		4.8755	257.80	0.33%
168h_S		ZZI-J1		4.8712	267.63	0.45%
168h_S		ZZI-K1		-	-	-
168h_S	16x16	C-K1	gurobi	-	-	-
168h_S		BM-J1		-	-	-
168h_S		BM-K1		-	-	-

Battery model & scalability comparison Due to the large number of simulation analyzed, all the results are presented in appendix 5. In this section, different simulations results are going to be extracted to draw conclusions.

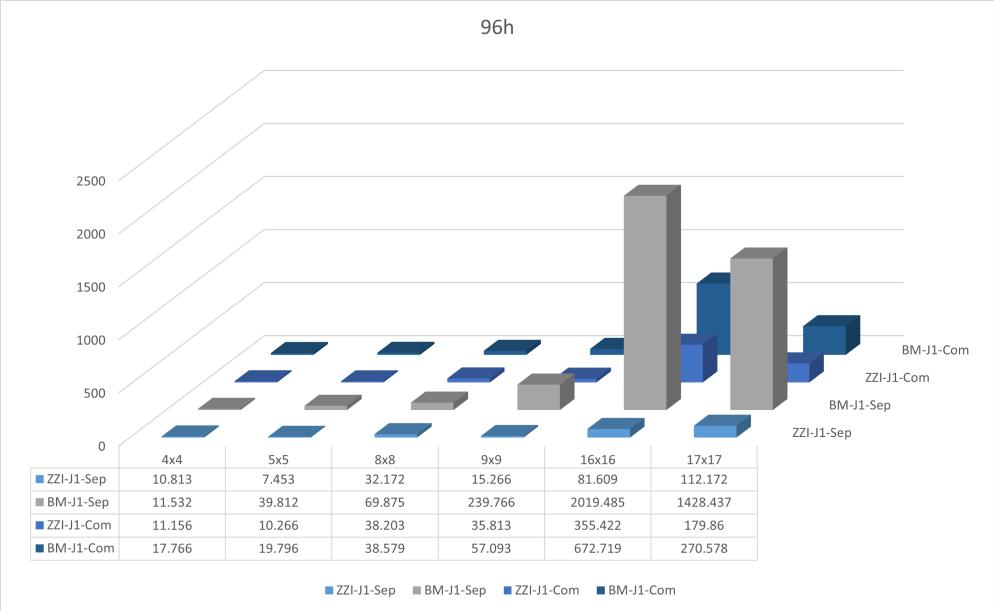
The next series of simulations were focus in understanding the scalability of the most promising linearization methods, "ZZI J1" and "BM J1" as well as the two proposed battery models. Regarding the scalability of the models, it can be observed, the grid granularity affects the computing performance drastically. Highlighting the importance of selecting the lowest grid resolution that precision allows, as well as optimize the domain triangulation to the non-linear function. Overall, the ZZI-J1 outperformed the BM-J1 specially for problems with large granularity.

When the two battery models are compared, conclusions are counterintuitive. On paper, the "Combined" model should have outperformed the "Separate" model, as the number of integer variables of the latter is considerable larger. In reality, that behavior is generally not absolved. Only in the 96-hour simulation with the BM-J1 case the "Combined" method yields lower solving times compared to the "Separated".



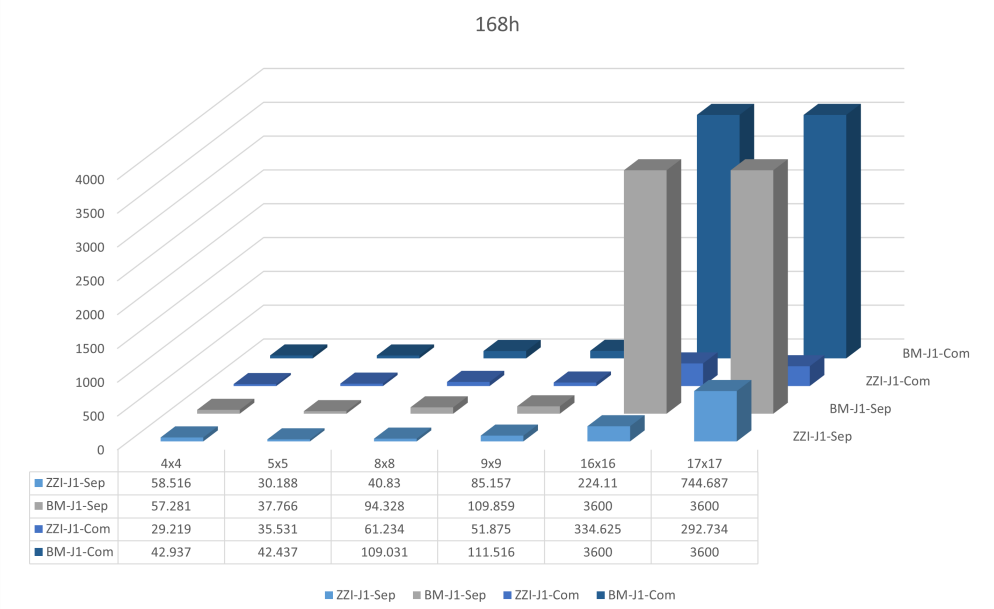
	4x4	5x5	8x8	9x9	16x16	17x17
ZZI-J1-Sep	1.37	1.078	19.56	7.063	26.547	36.188
BM-J1-Sep	2.36	3.531	9.375	22.641	49.422	32.531
ZZI-J1-Com	1.188	12.078	10.046	4.5	40.453	68.86
BM-J1-Com	19.563	2.079	15.234	22.234	86.516	61.469

Figure 4.11: Solving Time[s] of the 48 Hours case study.



	4x4	5x5	8x8	9x9	16x16	17x17
ZZI-J1-Sep	10.813	7.453	32.172	15.266	81.609	112.172
BM-J1-Sep	11.532	39.812	69.875	239.766	2019.485	1428.437
ZZI-J1-Com	11.156	10.266	38.203	35.813	355.422	179.86
BM-J1-Com	17.766	19.796	38.579	57.093	672.719	270.578

Figure 4.12: Solving Time[s] of the 96 Hours case study.



	4x4	5x5	8x8	9x9	16x16	17x17
ZZI-J1-Sep	58.516	30.188	40.83	85.157	224.11	744.687
BM-J1-Sep	57.281	37.766	94.328	109.859	3600	3600
ZZI-J1-Com	29.219	35.531	61.234	51.875	334.625	292.734
BM-J1-Com	42.937	42.437	109.031	111.516	3600	3600

Figure 4.13: Solving Time[s] of the 168 Hours case study.

4.4.3 Model Precision

Lastly, the analysis is focused on comparing different approaches for modeling the nonlinearities. Three alternative method are analyzed. Firstly, a simple but

widely used model where the battery efficiency is assumed to be constant (CTE η), as in equations 4.27 & 4.28. The selected efficiency was the average efficiency from the operations of the battery considering the non-linear loss expression.

Secondly, the linearization of the Li-Ion loss functions with piecewise function, which includes the presented models in this master thesis (ZZI-J1), and lastly the complete modeling of the non-linear function directly with a non-linear solver (NL). In this case, the solver used is Knitro (14.0.0) which was executed with Julia.

As the piecewise linearization and constant efficiency models are approximations, the objective function value is an expectation. To evaluate the real objective function value, the results of the battery operation was evaluated in a model considering the Li-Ion losses, referred as CTE η-SIM and ZZI-J1-SIM. These models were the same as NL, but the variables controlling the charge and discharged of the battery were fixed to the control signals provided by the CTE and ZZI-J1 models.

In table 4.6 the results of the simulations are presented, which aligns with the expectation. The model CTE η yields the lowest objective function, but when its operation is evaluated with the real loss function the resulted operation is slightly infeseable as in some hours the soc is lower than 10% limit, due to higher than expected losses in the battery. Additionally, the real objective function value is increase a 0.71%.

Regarding the ZZI-J1 methods, the expected objective function value is the same as the real objective function (with a 4 digits approximation) showcasing its accuracy. Regarding its operation, it did not reach any infeasibility. When compare the different grid granularity, the objective function value decreases as the granularity increase. It can be justified because the linearize loss function overestimates the losses, as it can be observed in figure 4.6.

In the 8x8 and 16x16 cases its objective function value is lower than the CTE η showing an improvement in accuracy and optimality compared to the simpler model.

Lastly, it's worth mentioning that the NL model lead to the operation with the minimum cost, but the computation burden is almost 20 times the slowest alternative method. Additionally, the bigger case with 168 hours was simulated, but no solution was found in a 2 hour solving time limit.

Results showcase the inverse relationship between accuracy and solving time. Additionally, the percentage improvement in the objective function do not seem relevant, but it has to be considered that the optimization horizon was 48 hours, along the useful life of the battery the difference can become relevant.

$$P_{loss}^{disc} \approx P^{disc}(1-\eta)/\eta \tag{4.27}$$

$$P_{loss}^{char} \approx P^{char}(1-\eta) \tag{4.28}$$

Table 4.6: Computational performance: Model precision

Case	**Grid**	**Method.**	**Solver**	**Obj.[k€]**	**Time[s]**	**Diff.**[1]
48h_W	-	NL	Knitro	6.0562	429.056	-
48h_W	-	CTE η	Gurobi	6.0512	0.25	0.08%
48h_W	-	CTE η-SIM	Knitro	6.0990	-	0.71%
48h_W	4x4	ZZI-J1	Gurobi	6.1283	1.370	1.19%
48h_W	-	ZZI-J1-SIM	Knitro	6.1283	-	1.19%
48h_W	8x8	ZZI-J1	Gurobi	6.0893	19.56	0.55%
48h_W	-	ZZI-J1-SIM	Knitro	6.0893	-	0.55%
48h_W	16x16	ZZI-J1	Gurobi	6.0817	26.547	0.42%
48h_W	-	ZZI-J1-SIM	Knitro	6.0817	-	0.42%

[1]Percentage difference compared to the NL method.

Chapter 5

Conclusions

This master's thesis delves into the modeling of Li-Ion batteries within the framework of power systems. With the increasing penetration of intermittent energy generation sources into modern power grids, energy storage has emerged as a crucial tool to mitigate the stochastic nature of wind and solar power generation and as an enabler for higher percentage of renewable energy sources in the generation stacks.

Traditionally, the primary method for large-scale electricity storage in power systems has been pump hydropower. However, recent attention has shifted towards chemical batteries due to their lower infrastructure costs and versatility. Among chemical battery technologies, Li-Ion batteries stand out due to their high energy density and promising cost-effectiveness. This makes the accurate modeling of this technology an imperative.

The focus of this master's thesis is placed on considering the non-linear behavior incurred during the charge and discharge cycles of Li-Ion batteries. Assessing if the promising Zig-Zag Integer linearization technique is applicable to the Li-Ion charge and discharge losses.

The project has three primary objectives. Firstly, to explore the Zig-Zag Integer linearization technique and develop a comprehensive understanding of its formulation. This objective was achieved by creating a generic platform in GAMS that allows researchers to use the Zig-Zag Integer technique as well as alternative methods discussed in this master's thesis.

This platform has proven to be very useful. The research results highlight the importance of testing different linearization techniques and fine-tuning their parameters, as performance is significantly impacted by the underlying non-linear function that is being linearized. Integrating various configuration parameters

and linearization techniques into one platform considerably reduces development time and facilitates correct usage of the models.

The second objective was to assess the applicability of the Zig-Zag Integer technique for managing Li-Ion batteries in a microgrid. This effort resulted in two optimal management models. The first model directly linearizes the charge and discharge losses of Li-Ion batteries, while the second approach combines both the charge and discharge loss functions, leading to a mathematical model with fewer discrete variables which potentially reduced mathematical complexity.

Lastly, the final goal was to implement the proposed microgrid management models in GAMS and verify their accuracy. The authors of the Zig-Zag method provided a Julia Lang library to easily integrate linearization methods into Julia models. This library was used to validate the correct implementation of the developed models.

By reaching these three goals, the following conclusions were drawn:

Importance of accurate energy storage modeling: In light of the results presented in this master thesis, it is evident that to fully harness the potential of batteries, optimal management models, such as the one proposed, operates the batteries across its entire efficiency region. This underscores the critical importance of accurate modeling across the entire operational domain, rather than focusing solely on the high-efficiency region.

Having an accurate model of the loss profile is key to maximizing overall optimality from the system perspective, not just from the battery operation point of view. That being said, at the current solver development state, optimizing directly the non-linear efficiency of batteries is not feasible, and linearization techniques such as those presented in this master thesis show a balance between accuracy and performance.

Solver selection significance: The choice of the mathematical solver for integer programming significantly impacts results, specially during the linearization of non-linear functions. Gurobi (version 10.0.3) consistently outperforms Cplex (version 22.1.1.0) in all studied simulations, with Cplex sometimes failing to find a solution within the solving time limits.

Linearization approach importance: Results underscore the significance of the triangulation pattern used for the triangulation of the Li-Ion losses functions. For instance, opting for K1 over J1 could lead to solver inability in meeting the

solution criteria. Across various cases, the Zig-Zag Integer formulation pair with J1 triangulation consistently outperformed other methods. Conversely, methods like C-K1, BM-K1 and ZZI-K1 exhibited worst results, while BM-J1, despite its simplicity compared to ZZI-J1, showed potential.

Overall, the effectiveness of linearization methods heavily hinges on the specific problem at hand. Thus, practitioners are advised to experiment with diverse formulations to determine the most suitable approach. Specially focusing on the BM-J1 and ZZI-J1 models.

Finally, the proposed combined charge and discharge model did not yield the anticipated promising results. The conclusions are nuanced, with performance improvements noted in some simulations and reductions in others. This renders a definitive assessment challenging. Due to the theoretical improvements of this method, it is going to be studied with greater detail in future projects.

In conclusion, accurate modeling of Li-Ion batteries is crucial for effective energy storage management in power systems. Solver selection and modeling approaches play significant roles in achieving optimal results and warrant careful consideration in practical applications. But overall, applying the Zig-Zag Integer linearization technique presented in this master thesis have shown to be an appropriate approach for modeling Li-Ion batteries both accurate and in reasonable times.

Bibliography

[1] International Energy Agency. Batteries and secure energy transitions.

[2] International Energy Agency. Global ev outlook 2021.

[3] Elpiniki Apostolaki-Iosifidou, Paul Codani, and Willett Kempton. Measurement of power loss during electric vehicle charging and discharging. *Energy*, 127:730–742, 2017.

[4] Djangir A. Babayev. Piece-wise linear approximation of functions of two variables. *Journal of Heuristics*, 2(4):313–320, 1997.

[5] European Comission. Renewable energy targets.

[6] George B. Dantzig. On the significance of solving linear programming problems with some integer variables. *Econometrica*, 28(1):30–44, 1960.

[7] Beloit College Department of Chemistry. Energy density of different materials.

[8] David Domínguez-Barbero, Javier García-González, and Miguel Á. Sanz-Bobi. Twin-delayed deep deterministic policy gradient algorithm for the energy management of microgrids. *Engineering Applications of Artificial Intelligence*, 125:106693, 2023.

[9] Joey Huchette and Juan Pablo Vielma. Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools, 2017.

[10] Joey Huchette and Juan Pablo Vielma. A combinatorial approach for small and strong formulations of disjunctive constraints, 2018.

[11] Joey Huchette and Juan Pablo Vielma. A geometric way to build strong mixed-integer programming formulations. *Operations Research Letters*, 47(6):601–606, 2019.

[12] Joseph Andrew Huchette. Advanced mixed-integer programming formulations: Methodology, computation, and application. 2018.

[13] Salvador Guerrero García Javier García-González. Optimal management of a microgrid li-ion battery considering non-linear losses using the integer zig-zag formulation. *PSCC*, 2024.

[14] Gurobi Optimization. Gurobi 9.5 launch event presentation.

[15] Gaizka Saldaña, José Ignacio San Martín, Inmaculada Zamora, Francisco Javier Asensio, and Oier Oñederra. Analysis of the current electric battery models for electric vehicle simulation. *Energies*, 12(14), 2019.

[16] Dr Oliver Schmidt and Dr Iain Staffell. *Monetizing Energy Storages*. Oxford University Press, 2023.

[17] Shoudong Zhu Tianhong Tan. Cornell university computational optimization open textbook: Piecewise linear approximation.

[18] Olivier Tremblay and Louis-A. Dessaint. Experimental Validation of a Battery Dynamic Model for EV Applications. *World Electric Vehicle Journal*, 3(2):289–298, June 2009.

[19] Juan Pablo Vielma. Embedding formulations and complexity for unions of polyhedra, 2015.

[20] Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57:3–57, 2015.

[21] Juan Pablo Vielma and George L. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128:49–72, 6 2011.

[22] B. V. Wagle. Mathematical Programming. *Journal of the Royal Statistical Society Series D: The Statistician*, 14(2):176–177, 12 2018.

[23] Siyuan Wang, Jian Liu, Haotian Chen, Rui Bo, and Yonghong Chen. Modeling state transition and head-dependent efficiency curve for pumped storage hydro in look-ahead dispatch. *IEEE Transactions on Power Systems*, 36(6):5396–5407, 2021.

Extension of Computational Results

In this Appendix, the different simulations results from section 4.4.2 are presented in tables 1 & 2.

Table 1: Computational performance results: Separate Charge & Discharge model

Case	Grid	Method.	Solver	Obj.[€]	Time [s]	Rel.Gap
48h	4x4	ZZI-J1-Sep	gurobi	6.1283	1.37	0.40%
48h	5x5	ZZI-J1-Sep	gurobi	6.107321	1.078	0.48%
48h	8x8	ZZI-J1-Sep	gurobi	6.0893	19.56	0.37%
48h	9x9	ZZI-J1-Sep	gurobi	6.085555	7.063	0.36%
48h	16x16	ZZI-J1-Sep	gurobi	6.081719	26.547	0.36%
48h	17x17	ZZI-J1-Sep	gurobi	6.072211	36.188	0.21%
48h	4x4	BM-J1-Sep	gurobi	6.122561	2.36	0.38%
48h	5x5	BM-J1-Sep	gurobi	6.095146	3.531	0.24%
48h	8x8	BM-J1-Sep	gurobi	6.085568	9.375	0.32%
48h	9x9	BM-J1-Sep	gurobi	6.092477	22.641	0.48%
48h	16x16	BM-J1-Sep	gurobi	6.075384	49.422	0.26%
48h	17x17	BM-J1-Sep	gurobi	6.072211	32.531	0.21%
96h	4x4	ZZI-J1-Sep	gurobi	11.085198	10.813	0.30%
96h	5x5	ZZI-J1-Sep	gurobi	11.06311	7.453	0.46%
96h	8x8	ZZI-J1-Sep	gurobi	11.017093	32.172	0.32%
96h	9x9	ZZI-J1-Sep	gurobi	11.017136	15.266	0.38%
96h	16x16	ZZI-J1-Sep	gurobi	10.995312	81.609	0.24%
96h	17x17	ZZI-J1-Sep	gurobi	10.983104	112.172	0.13%
96h	4x4	BM-J1-Sep	gurobi	11.111537	11.532	0.50%
96h	5x5	BM-J1-Sep	gurobi	11.029741	39.812	0.16%
96h	8x8	BM-J1-Sep	gurobi	11.035023	69.875	0.48%
96h	9x9	BM-J1-Sep	gurobi	10.99445	239.766	0.16%
96h	16x16	BM-J1-Sep	gurobi	10.973862	2019.485	0.04%
96h	17x17	BM-J1-Sep	gurobi	11.01592	1428.437	0.43%
168h w	4x4	ZZI-J1-Sep	gurobi	19.224436	58.516	0.33%
168h w	5x5	ZZI-J1-Sep	gurobi	19.15099	30.188	0.34%
168h w	8x8	ZZI-J1-Sep	gurobi	19.1052	40.83	0.38%
168h w	9x9	ZZI-J1-Sep	gurobi	19.084593	85.157	0.31%
168h w	16x16	ZZI-J1-Sep	gurobi	19.0596	224.11	0.26%
168h w	17x17	ZZI-J1-Sep	gurobi	19.046203	744.687	0.19%
168h w	4x4	BM-J1-Sep	gurobi	19.250236	57.281	0.46%
168h w	5x5	BM-J1-Sep	gurobi	19.160412	37.766	0.39%
168h w	8x8	BM-J1-Sep	gurobi	19.102198	94.328	0.38%
168h w	9x9	BM-J1-Sep	gurobi	19.083944	109.859	0.39%
168h w	16x16	BM-J1-Sep	gurobi	-	3600	-
168h w	17x17	BM-J1-Sep	gurobi	-	3600	-

Table 2: Computational performance results: Combined Charge & Discharge model

Case	Grid	Method.	Solver	Obj.[€]	Time [s]	Rel.Gap
48h	4x7	ZZI-J1-Com	gurobi	6.123703	1.188	0.39%
48h	5x9	ZZI-J1-Com	gurobi	6.106321	12.078	0.40%
48h	8x15	ZZI-J1-Com	gurobi	6.081934	10.046	0.27%
48h	9x17	ZZI-J1-Com	gurobi	6.074056	4.5	0.18%
48h	16x31	ZZI-J1-Com	gurobi	6.086722	40.453	0.45%
48h	17x33	ZZI-J1-Com	gurobi	6.08793	68.86	0.47%
48h	4x7	BM-J1-Com	gurobi	6.13757	19.563	0.43%
48h	5x9	BM-J1-Com	gurobi	6.084684	2.079	0.08%
48h	8x15	BM-J1-Com	gurobi	6.077588	15.234	0.20%
48h	9x17	BM-J1-Com	gurobi	6.090247	22.234	0.45%
48h	16x31	BM-J1-Com	gurobi	6.073538	86.516	0.23%
48h	17x33	BM-J1-Com	gurobi	6.08441	61.469	0.43%
96h	4x7	ZZI-J1-Com	gurobi	11.115581	11.156	0.46%
96h	5x9	ZZI-J1-Com	gurobi	11.043198	10.266	0.29%
96h	8x15	ZZI-J1-Com	gurobi	11.004264	38.203	0.21%
96h	9x17	ZZI-J1-Com	gurobi	10.993983	35.813	0.15%
96h	16x31	ZZI-J1-Com	gurobi	10.990093	355.422	0.19%
96h	17x33	ZZI-J1-Com	gurobi	10.979162	179.86	0.10%
96h	4x7	BM-J1-Com	gurobi	11.08584	17.766	0.21%
96h	5x9	BM-J1-Com	gurobi	11.023765	19.796	0.10%
96h	8x15	BM-J1-Com	gurobi	11.014901	38.579	0.30%
96h	9x17	BM-J1-Com	gurobi	11.020678	57.093	0.39%
96h	16x31	BM-J1-Com	gurobi	10.997086	672.719	0.25%
96h	17x33	BM-J1-Com	gurobi	11.011443	270.578	0.39%
168h w	4x7	ZZI-J1-Com	gurobi	19.256026	29.219	0.44%
168h w	5x9	ZZI-J1-Com	gurobi	19.131402	35.531	0.24%
168h w	8x15	ZZI-J1-Com	gurobi	19.070269	61.234	0.20%
168h w	9x17	ZZI-J1-Com	gurobi	19.116959	51.875	0.48%
168h w	16x31	ZZI-J1-Com	gurobi	19.081705	334.625	0.37%
168h w	17x33	ZZI-J1-Com	gurobi	19.05362	292.734	0.22%
168h w	4x7	BM-J1-Com	gurobi	19.229333	42.937	0.32%
168h w	5x9	BM-J1-Com	gurobi	19.10463	42.437	0.09%
168h w	8x15	BM-J1-Com	gurobi	19.103467	109.031	0.37%
168h w	9x17	BM-J1-Com	gurobi	19.092153	111.516	0.40%
168h w	16x31	BM-J1-Com	gurobi	-	3600	-
168h w	17x33	BM-J1-Com	gurobi	-	3600	-

Julia & PiecewiseLinearOpt

Julia is a high level programming language developed in the MIT. It shares similarities with Python in terms of syntax but has higher focus on scientific computing, performance and deployability. These characteristics have make Julia a popular language for both the research community and industry.

[9] introduces the Julia library: PiecewiseLinearOpt, a complement for Julia's JuMP, its modeling language for mathematical optimization. JuMP is the equivalent to Pyomo in Python, GAMS or AMPL. It is a modeling language that allows the formulation of optimization problems and their resolution with external solvers such as Gurobi or Cplex.

Utilizing JuMP to solve mathematical optimization problems is straightforward. At its core is the JuMP Model object, to which users can assign decision variables, constraints, the objective function, and a solver packages.

To streamline the implementation of advanced linearization techniques, PiecewiseLinearOpt provides the "piecewiselinear" function. This function augments the desired JuMP model object with the auxiliary decision variables and constraints responsible for modeling piecewise linear functions.

The function "piecewiselinear", support the linearization of uni-variate and bi-variate functions. It requires as inputs the jump mathematical model, the decision variables to optimize, the mathematical region to optimize the function over, and the linearization method to use. Regarding the domain of the optimization, for uni-variate functions $y = f(x)$, the user has to input the vector of the breakpoints $\{x_1, x_2, x_3...x_n\}$ and its associate y values or the function $f(x)$. For bi-variate expressions $z = g(x, y)$, the PiecewiseLinearOpt function "BivariatePWLFunction" establishes the domain of the optimization, its inputs are the vector of breakpoints of x and y, the function to linearize $g(x, y)$, and the triangulation pattern to use.

PiecewiseLinearOpt significantly simplifies the usage of piecewise linear func-

tions. It closes the gap between state-of-the-art techniques and practical application by abstracting complex constraints to a lower level. Moreover, it greatly improves development time.

As highlighted in the conclusions of this master thesis, testing different linearization techniques is crucial, as performance varies significantly depending on the problem. This library empowers users to experiment easy with different formulations, supporting a wide array of linearization techniques.

Supported linearization methods: "Incremental" ([22],[6]), "LogarithmicIB" (LogIB: [10]), "Logarithmic" (LogE: [19]), "ZigZag" (ZZB: [9]), "ZigZagInteger" (ZZI: [9]) etc.

Supported triangulation patterns: "Upper","Lower","BestFit","UnionJack (J1)", "K1", "Random".

In listing 1 & listing 2 two simple optimization problems (1 & 2) are implemented and solved using Julia and the library PiecewiseLinearOpt. Showcasing the simplicity of the implementation.

$$\begin{aligned} min \quad & y \\ s.t. \quad & y = f(x) = e^x \\ 0 \leq x \leq 3 & \\ x, y \in \mathcal{R} & \end{aligned} \tag{1}$$

Listing 1: Implementation of equations 1

```
using JuMP, Gurobi, PiecewiseLinearOpt

#Model inicialization
m = Model(Gurobi.Optimizer)

#Variable declaration
@variable(m, x)

#Constraint declaration
x_range = range(0,3,3)
f(x) = exp(x)
y = piecewiselinear(m, x, x_range, f,method=:ZigZagInteger)

#Objective function
@objective(m, Min, y)

#print model
print(m)
```

```
#Solve model
optimize!(m)
```

$$\begin{aligned} min \quad & z \\ s.t. \quad & z = f(x,y) = e^x + e^y \\ 0 \leq & x \leq 3 \\ 0 \leq & y \leq 3 \\ x, y, z \in & \mathcal{R} \end{aligned} \tag{2}$$

Listing 2: Implementation of equations 2

```
using JuMP, Gurobi, PiecewiseLinearOpt

#Model inicialization
m = Model(Gurobi.Optimizer)

#Variable declaration
@variable(m, x)
@variable(m, y)

#Constraint declaration
x_range = range(0,3,3)
y_range = range(0,3,3)
f(x,y) = exp(x) + exp(y)
z = piecewiselinear(m, x, y, BivariatePWLFunction(x_range, y_range, f, pattern=:
    K1),method=:ZigZagInteger)

#Objective function
@objective(m, Min, z)

#print model
print(m)

#Solve model
optimize!(m)
```

Sustainable Development Goals

The Sustainable Development Goals (SDGs) comprise 17 objectives adopted by United Nations member states in 2015, aiming to address the world's most pressing challenges.

These goals encompass a broad spectrum of issues, ranging from eradicating poverty and ensuring access to clean water to preserving nature and combating climate change.

This master's thesis directly aligns with three of these goals:

SDG 7: Affordable and clean energy. The primary motivation of this work is to enhance the accurate integration of Li-Ion batteries into decision-making models, thereby advancing the optimal management of batteries. Batteries play a pivotal role in energy systems with high levels of renewable energy penetration. Given the inherent uncertainty associated with common renewable sources such as wind and solar, batteries serve as crucial energy storage solutions. Thus, improving battery models and management strategies directly contributes to the expansion of affordable and clean renewable energy sources.

SDG 9: Industry, Innovation, and Infrastructure. This master's thesis encompasses significant research components, exploring the current state-of-the-art in linearization techniques to assess their applicability to battery models. By bridging the gap between mathematical research and industry applications, this work fosters innovation in battery control and infrastructure development.

SDG 11: Sustainable Cities and Communities. Li-Ion batteries find application in urban areas at both individual household and community levels. In such settings, decision-making processes are often automated due to the impracticality of skilled manual intervention. Consequently, contributing to decision-making models in these contexts is vital for maximizing the usage of renewable energy sources on urban environments and therefore becoming more self sustainable.

By addressing these SDGs, this master's thesis endeavors to contribute to global efforts toward sustainable development and a more resilient future.