TASK DESCRIPTION

The task description replaces this page in the thesis.

The thesis task description has to be taken over at the administration of the host department if not specified otherwise (by the same department). Only properly authenticated task description can replace this page in the thesis. Two options are available:

1. An original is provided by the host department, stamped and signed by the department head.
2. A downloaded and printed copy is provided based on the electronically approved version on the Thesis Portal.

The uploaded version of the thesis work must not contain again a task description as it is uploaded separately to the Thesis Portal.

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Artificial Intelligence

Abdelhamid Ahbane

DEVELOPMENT OF A DATA-SHEET ANALYSIS TOOL WITH FOCUS ON DIAGRAM AUTOMATION

SUPERVISOR

Dr. Al-Radhi Mohammed Salah,

Dr. Fischer Andreas

BUDAPEST, 2025

Contents

STUDENT DECLARATION

I, **Abdelhamid Ahbane**, the undersigned, hereby declare that the present BSc thesis work has been prepared by myself and without any unauthorized help or assistance. Only the specified sources (references, tools, etc.) were used. All parts taken from other sources word by word, or after rephrasing but with identical meaning, were unambiguously identified with explicit reference to the sources utilized.

I authorize the Faculty of Electrical Engineering and Informatics of the Budapest University of Technology and Economics to publish the principal data of the thesis work (author's name, title, abstracts in English and in a second language, year of preparation, supervisor's name, etc.) in a searchable, public, electronic and online database and to publish the full text of the thesis work on the internal network of the university (this may include access by authenticated outside users). I declare that the submitted hardcopy of the thesis work and its electronic version are identical.

Full text of thesis works classified upon the decision of the Dean will be published after a period of three years.

Budapest, 12 December 2025

Abdelhamid Ahbane

Table of Figures

Summary

A time reduction of approximately 98% in the curve-fitting phase is achieved by the automated framework developed in this thesis. This significant efficiency gain is utilized to streamline the characterization of power semiconductor devices, such as Insulated Gate Bipolar Transistors (IGBTs) and Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFETs), thereby transforming a historically labor-intensive process into a reproducible and rapid workflow. The characterization of these power semiconductor devices, constitutes a critical phase in the development of modern power electronics. To ensure reliability and efficiency, Development Lab Engineers at Infineon Technologies generate vast quantities of experimental measurement data that must be synthesized into accurate Data-Sheets. However, the transition from raw measurement files to finalized, mathematically consistent diagrams is historically a fragmented, manual, and time-consuming process, prone to variability and human error.

The objective of this thesis is to streamline this workflow by transforming an existing MATLAB-based analysis tool into a generalized, template-driven framework. The primary engineering contribution of this work is the design and implementation of a flexible XML-based configuration system. This architecture allows engineers to externalize the logic for data filtering, unit scaling, and diagram definition, thereby ensuring reproducibility across different device types without requiring modification to the source code.

Furthermore, the analytical capabilities of the software were significantly enhanced to support the autonomous modeling of non-linear semiconductor physics. A Tikhonov-regularized solver was implemented with a coordinate transformation engine, enabling the robust fitting of exponential and power-law behaviors. To eliminate the subjectivity of manual parameter tuning, a Hyperparameter Autotuning algorithm was developed. This feature utilizes a Modified Akaike Information Criterion (AICc)—incorporating specific penalties for model complexity and rewards for physical constraints—to automatically select the optimal mathematical model. The developed tool successfully reduces manual intervention and significantly accelerates the Data-Sheet generation process while guaranteeing high standards of analytical consistency.

Résumé

Une réduction du temps d'environ 98 % lors de la phase d'ajustement de courbe est obtenue grâce au cadre automatisé développé dans cette thèse. Ce gain d'efficacité significatif est mis à profit pour rationaliser la caractérisation des dispositifs semi-conducteurs de puissance, tels que les transistors bipolaires à grille isolée (IGBT) et les transistors à effet de champ à structure métal-oxyde-semi-conducteur (MOSFET), transformant ainsi un processus historiquement laborieux en un flux de travail rapide et reproductible. La caractérisation de ces dispositifs constitue une étape cruciale dans le développement de l'électronique de puissance moderne. Afin de garantir fiabilité et efficacité, les ingénieurs des laboratoires de développement chez Infineon Technologies génèrent de vastes quantités de données de mesure expérimentales qui doivent être synthétisées en fiches techniques précises. Cependant, la transition des fichiers de mesures brutes vers des diagrammes finalisés et mathématiquement cohérents est historiquement un processus fragmenté, manuel et chronophage, sujet à la variabilité et aux erreurs humaines.

L'objectif de cette thèse est de rationaliser ce flux de travail en transformant un outil d'analyse existant sous MATLAB en un cadre généralisé piloté par des modèles. La principale contribution technique de ce travail réside dans la conception et la mise en œuvre d'un système de configuration flexible basé sur XML. Cette architecture permet aux ingénieurs d'externaliser la logique de filtrage des données, de mise à l'échelle des unités et de définition des diagrammes, assurant ainsi une reproductibilité entre différents types de dispositifs sans nécessiter de modification du code source.

De plus, les capacités analytiques du logiciel ont été considérablement améliorées pour permettre la modélisation autonome de la physique non linéaire des semi-conducteurs. Un solveur régularisé de Tikhonov a été implémenté conjointement avec un moteur de transformation de coordonnées, permettant l'ajustement robuste des comportements exponentiels et des lois de puissance. Afin d'éliminer la subjectivité du réglage manuel des paramètres, un algorithme de réglage automatique des hyperparamètres a été développé. Cette fonctionnalité exploite un Critère d'Information d'Akaike corrigé (AICc) modifié intégrant des pénalités spécifiques pour la complexité du modèle et des récompenses pour le respect des contraintes physiques, pour sélectionner

automatiquement le modèle mathématique optimal. L'outil développé réduit avec succès les interventions manuelles et accélère significativement le processus de génération des fiches techniques, tout en garantissant des normes élevées de cohérence analytique.

1 Introduction

Background and Motivation

The continuous evolution of power electronics is a driving force behind the global transition towards higher energy efficiency. Modern applications, ranging from electric vehicle traction inverters to renewable energy grid integration, place increasingly stringent demands on power semiconductor devices. Consequently, the characterization and verification of components such as Insulated Gate Bipolar Transistors (IGBTs) and Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFETs) have become critical phases in the product development lifecycle [1][8]. Companies like Infineon Technologies operate at the forefront of this field, ensuring that every component meets rigorous performance and reliability standards before reaching the market.

A central output of this characterization process is the device Data-Sheet. This comprehensive document serves as the primary reference for system designers, providing essential thermal, electrical, and switching characteristics derived from extensive experimental measurements, often standardized by international electrotechnical guidelines [9]. Development laboratory engineers generate vast quantities of raw data by testing devices under a wide spectrum of operating conditions, including varying temperatures, gate voltages, and load currents, as defined in standard switching loss protocols [10]. The accuracy and clarity of the diagrams presented in these Data-Sheets are paramount, as they directly influence the design decisions made by engineers worldwide [22].

However, the transition from raw experimental measurement data to publication-ready diagrams is a complex and historically labor-intensive process. Raw measurement files, typically generated by high-speed oscilloscopes and data loggers, often contain significant noise and are recorded in base SI units that differ from the engineering units required for documentation. Furthermore, the data often represents a mix of distinct system states, such as high-side versus low-side switching events in a half-bridge configuration that must be meticulously separated before analysis. When performed manually, the tasks of filtering, scaling, and curve fitting are not only time-consuming but also prone to human error, potentially introducing inconsistencies into the final documentation.

Problem Statement

While software tools exist to assist with data visualization, the standard workflows often lack the flexibility required to keep pace with the rapidly expanding variety of device architectures. The legacy analysis framework utilized within the laboratory provided a stable architectural foundation but relied on static, hardcoded configurations optimized for specific device types. As the scope of characterization expanded to include novel MOSFET topologies and increasingly complex measurement protocols, the rigidity of this approach became a bottleneck. Engineers were frequently required to request software modifications for routine configuration changes, creating a dependency that slowed the analysis cycle.

A significant challenge within this workflow is the mathematical modeling of device characteristics. Experimental data is inherently noisy, and fitting smooth curves to this data is essential for extracting meaningful parameters. In a manual workflow, finding the optimal mathematical model involves a trial-and-error process where the engineer must iteratively adjust polynomial orders and smoothing factors. This approach relies heavily on user intuition, leading to a lack of reproducibility; different engineers might produce slightly different curves for the same dataset.

Furthermore, standard polynomial fitting methods are frequently insufficient for the robust modeling of non-linear physical behaviors, such as exponential leakage currents or power-law switching energy distributions. Consequently, significant manual intervention is required to constrain the generated curves into physically plausible trajectories. From a numerical perspective, the approximation of these complex behaviors using high-order polynomials on noisy data is recognized as an ill-posed problem [2][7]. This formulation often results in numerical instability and overfitting, thereby limiting the reliability of the analysis.

Objectives

The primary objective of this thesis is to design and implement a comprehensive extension to the existing Data-Sheet Analysis Tool, transforming it into a generalized, automated framework. The work aims to eliminate the bottlenecks associated with manual data processing by introducing a flexible, template-driven architecture. This system allows laboratory engineers to define complex analysis logic—including data

filtering, unit scaling, and diagram definition—via external configuration files, thereby decoupling the analysis parameters from the compiled source code.

Technically, the project seeks to enhance the analytical capabilities of the software by integrating advanced mathematical processing engines. A key goal is the implementation of a robust solver capable of handling logarithmic coordinate transformations, ensuring that non-linear semiconductor physics can be modeled accurately. Furthermore, the thesis aims to develop an intelligent Hyperparameter Autotuning algorithm. By utilizing a modified statistical selection criterion, this algorithm is designed to automatically identify the optimal curve-fitting configuration that balances mathematical accuracy with adherence to physical constraints, such as monotonicity and positivity. The ultimate goal is to provide a tool that significantly accelerates the Data-Sheet generation process while guaranteeing high standards of consistency and reproducibility.

Thesis Organization

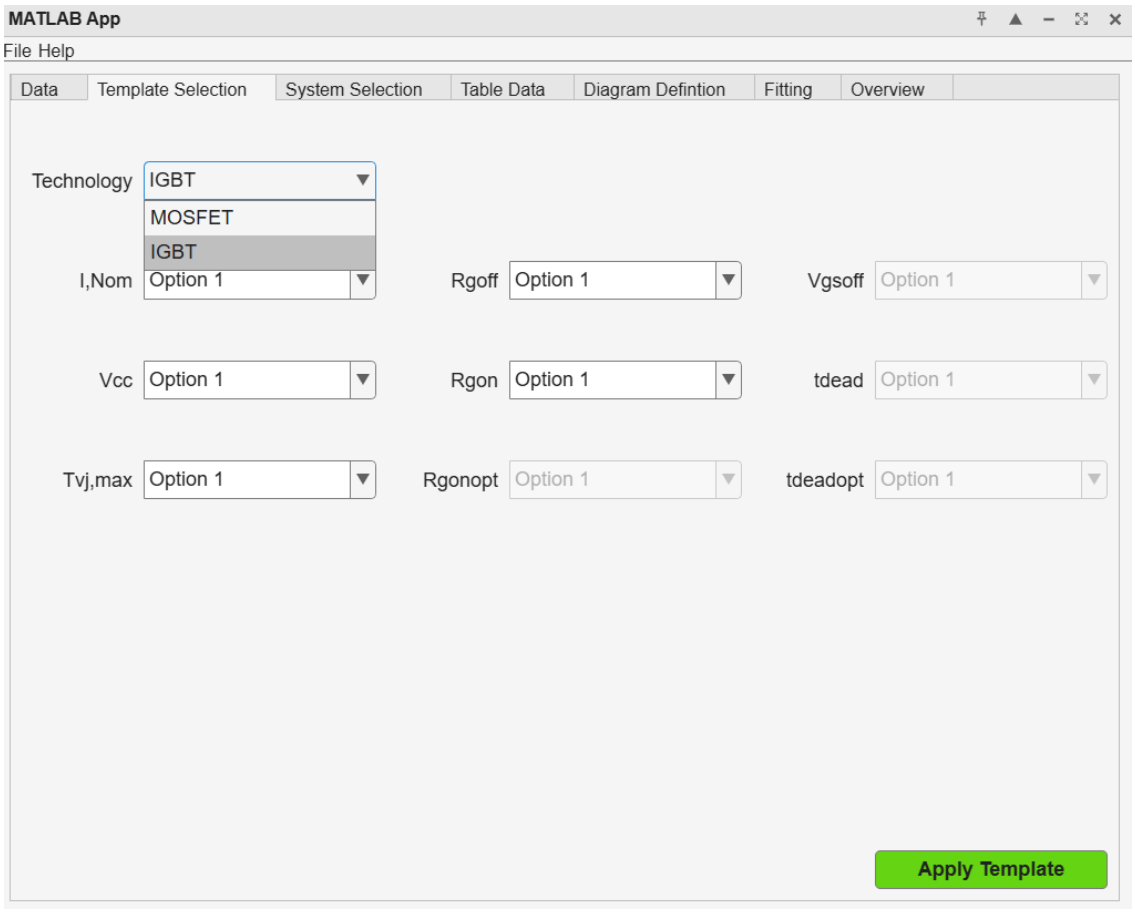
The remainder of this thesis is organized as follows. Chapter 2 provides a detailed analysis of the task, examining the state of the art of the existing software architecture and specifying the functional requirements for the extended framework. Chapter 3 outlines the theoretical preliminaries, deriving the mathematical formulations for Tikhonov regularization, the coordinate transformations used for logarithmic fitting, and the statistical criteria for model selection. Chapter 4 details the design and implementation of the engineering solution, describing the development of the XML template parser, the extended mathematical engine, and the autotuning algorithm. Chapter 5 presents the evaluation of the tool, verifying the automation framework and quantifying the improvements in analytical accuracy and workflow efficiency through test cases using real-world MOSFET data. Chapter 6 offers a critical assessment of the completed work, discussing the challenges encountered and potential areas for future development. Finally, Chapter 7 summarizes the contributions of the thesis.

2 Problem Analysis and Requirements

State of the Art

The software framework initially utilized by the development laboratory was built upon the MATLAB App Designer platform [19], providing a graphical environment for the post-processing of semiconductor measurement data. The architectural backbone of this application was a reactive node system, a design pattern that manages data dependencies through a directed acyclic graph. In this architecture, data flows from source nodes (file loaders) to calculation nodes (scalers, filters) and finally to presentation nodes (tables, diagrams). A key strength of this existing architecture was its implementation of lazy evaluation [17]; mathematical operations were executed only when the final output was explicitly requested by the user, ensuring that the application remained responsive even when varying large datasets.

In terms of user experience, the legacy interface was designed around a linear, procedural workflow, adhering to standard interaction design patterns [21]. The user was required to navigate sequentially through a series of tabs, progressing from left to right. The process began with data loading, followed by a manual configuration step where specific measurement parameters were entered into a fixed form. 1. Figure illustrates this legacy "Template Selection" interface. The design relied on a pre-defined set of input fields specifically tailored to established technologies, such as standard IGBT modules. Users would select the device technology from a dropdown menu, which would populate the interface with a static set of variables required for that specific device type.



1. Figure: Legacy Template Selection Interface.

Workflow Challenges

Despite the robustness of the underlying reactive architecture, the operational workflow exhibited significant scalability challenges as the scope of device characterization expanded. The primary limitation was the rigid coupling between the measurement definition and the application source code. As demonstrated in 1.Figure, the template logic—defining which variables were required and how they should be processed—was hardcoded within the application. This meant that the tool offered no intrinsic flexibility for the end-user to customize the analysis structure. If a new measurement protocol was introduced, or if a novel device architecture (such as a MOSFET with complex body-diode behavior) required additional parameters not present in the standard IGBT form, the software source code had to be manually modified and recompiled. This dependency created a bottleneck, preventing laboratory engineers from autonomously adapting the tool to evolving test requirements.

Furthermore, the data conditioning phase represented a substantial manual overhead. In the legacy workflow, the segmentation of raw data—distinguishing between

different system states or operating modes—required interactive user intervention. Engineers had to manually configure filtering parameters for each session, a process that was not only time-consuming but also susceptible to human error. Similarly, the mathematical analysis of the curves was a "human-in-the-loop" process. The user was required to inspect each curve visually and manually adjust fitting parameters to achieve a satisfactory representation. This trial-and-error approach introduced latency and potential inconsistency, as the criteria for a "good fit" were subjective and dependent on the individual engineer’s judgment.

Requirements Specification

To address these limitations and transform the specialized tool into a generalized analysis framework, a comprehensive set of functional requirements was defined. The overarching objective was to decouple the analysis logic from the application logic, thereby enabling a fully data-driven workflow.

The primary requirement was the design and implementation of a flexible, XML-based templating system. The application was required to parse external configuration files that define the complete analysis context. This includes the definition of user inputs, the rules for unit scaling and data normalization, and the logic for system state filtering. By externalizing these definitions, the tool would allow engineers to create and modify analysis templates without altering the codebase. The workflow was to be streamlined such that a user could simply load a raw data file and an XML template, after which the system would automatically propagate the configuration to all downstream processes, populating tables and diagrams instantaneously.

Mathematically, the system required significant enhancement to support the autonomous modeling of non-linear semiconductor physics. The linear solver needed to be extended to support logarithmic coordinate transformations, enabling the robust fitting of exponential and power-law behaviors that are characteristic of modern power devices. Furthermore, to eliminate the subjectivity of manual fitting, the software required an intelligent Hyperparameter Autotuning algorithm. This algorithm was required to automatically explore the search space of possible mathematical models—varying polynomial orders, coordinate systems, and physical constraints—and select the optimal configuration based on a statistically rigorous criterion. The combination of these features

aimed to produce a "one-click" analysis experience, where the software delivers publication-ready diagrams immediately upon template application.

3 Theoretical Preliminaries

This chapter outlines the mathematical foundations that underpin the analytical capabilities of the developed software framework. To ensure the accurate extraction of physical parameters from noisy experimental data, the system relies on advanced regularization techniques, domain-specific coordinate transformations, and statistical model selection criteria. The following sections derive the mathematical formulations used to ensure both numerical stability and physical plausibility in the curve-fitting process.

Regularized Least Squares (Tikhonov Regularization)

The core analytical task involves approximating a smooth scalar function $f(x)$ given a set of discrete, noisy measurement pairs (x_i, y_i) where $i = 1,2,\dots,N$. In the context of semiconductor characterization, the data often contains stochastic noise derived from high-speed switching events. A standard Ordinary Least Squares (OLS) approach seeks to minimize the sum of squared residuals between the model and the observations. However, when high-order polynomials are used to approximate complex behaviors, the OLS formulation becomes an ill-posed problem [2]. This frequently leads to overfitting, where the fitted curve exhibits high-frequency oscillations—known as Runge's phenomenon—at the boundaries of the domain to minimize the residual error of specific noise points.

To mitigate this instability, the mathematical engine employs Tikhonov Regularization. Instead of minimizing the residual error in isolation, the algorithm minimizes a composite cost function J that includes a regularization term penalizing the complexity or "roughness" of the solution. The continuous form of the objective function is defined as:

$$J(p) = \Sigma_{i=1}^{N} w_i \big(y_i - f(x_i)\big)^2 + \lambda \int \left(\frac{d^m f}{dx^m}\right)^2 dx$$

In this formulation, w_i represents the weight assigned to each data point, allowing for the prioritization of specific measurement ranges. The second term represents the regularization penalty, where $\lambda \geq 0$ is the Tikhonov factor (smoothing parameter). The integral measures the energy of the $m - th$ derivative of the function. For this application,

the second derivative ($m = 2$) is utilized to penalize curvature, thereby favoring smoother trajectories that are physically characteristic of thermal and electrical semiconductor responses.

To solve this numerically, the problem is discretized. The function $f(x)$ is approximated by a polynomial of degree k, which can be expressed as a linear combination of basis functions (powers of x). The optimization problem is then formulated in matrix notation. Let y be the vector of observed values and V be the Vandermonde matrix [12] where $V_{ij} = x_i^j$. The coefficients c of the polynomial are determined by minimizing the discrete form of the cost function, a variation of the standard linear least squares problem [13]:

$$J(c) = || W^{1/2} (y - Vc) ||_2^2 + \lambda || Lc ||_2^2$$

Here, W is the diagonal weight matrix, and L is the Tikhonov matrix, which represents the discrete difference operator corresponding to the second derivative. The minimization of this quadratic form, subject to linear inequality constraints (such as positivity or monotonicity), constitutes a Quadratic Programming (QP) problem. The solver identifies the optimal coefficient vector c that satisfies the physical constraints while minimizing the weighted sum of the residual error and the curvature penalty.

Coordinate Transformations and Logarithmic Linearization

While Tikhonov regularization ensures numerical stability for polynomial fitting, the fundamental physical characteristics of semiconductor devices often adhere to non-linear laws that are not optimally approximated by polynomials in a Cartesian coordinate system. For instance, switching energy losses (E_{rec}) frequently scale with gate resistance according to a power law, whereas leakage currents typically exhibit an exponential dependence on temperature or voltage. To extend the applicability of the linear Tikhonov solver to these non-linear domains, a Coordinate Transformation Engine was implemented. This methodology relies on the principle of linearization, wherein the original data space D is mapped to a feature space F via a bijective transformation $\Phi(x, y) \rightarrow (u, v)$. The polynomial fitting is subsequently executed within this feature space, and the result is mapped back to the original domain via the inverse transformation Φ^{-1}.

To address power-law relationships of the form $y = Ax^k$. the Log-Log transformation is utilized. Linearization is achieved by taking the natural logarithm of both sides, yielding the equation:

$$ln(y) = ln(A) + k \cdot ln(x)$$

The transformation is defined as $u = ln(x)$ and $v = ln(y)$. In this space, a first-order polynomial $P(u) = c_1 u + c_0$ corresponds directly to the physical parameters, where the slope c_1 represents the exponent k, and the intercept c_0 corresponds to $ln(A)$. This transformation allows the linear solver to determine the exponent of the power law deterministically without requiring iterative non-linear optimization methods.

Analogously, exponential relationships of the form $y = Ae^{Bx}$ are modeled via the Semi-Log Y transformation. The linearization is achieved by taking the logarithm of the dependent variable:

$$ln(y) = ln(A) + B \cdot x$$

The mapping is defined as $u = x$ and $v = ln(y)$. A linear fit $v = c_1 u + c_0$ in this space maps back to the exponential function, where the coefficient $B = c_1$ and the scaling factor $A = e^{c0}$. This mode is particularly effective for modeling leakage currents or diode forward characteristics in the sub-threshold region. Furthermore, for processes that evolve over several orders of magnitude in the independent variable, the Semi-Log X transformation is employed. The relationship $y = A + B \cdot ln(x)$ is linearized by the mapping $u = ln(x)$ and $v = y$.

It is imperative to note that the mathematical derivations provided above illustrate the fundamental first-order cases ($P(u) = c_1 u + c_0$). However, the mathematical engine is not restricted to these linear approximations within the feature space. When the polynomial order is increased, the complexity of the back-transformed function increases significantly. For example, a second-order fit performed within the Log-Log domain corresponds to a function of the form $y = A \cdot x^{(B + C \ln x)}$. This capability allows the system to capture subtle variations and second-order effects in the physical data while retaining the stability benefits of the Tikhonov solver. By pre-processing the measurement vectors x and y into the vectors u and v before constructing the Vandermonde matrices, the stable Regularized Least Squares engine described in Section 0 is effectively reused to solve for non-linear physical parameters.

Model Selection Criteria

Automating the curve-fitting process requires a quantitative metric to compare the validity of different mathematical models, such as determining whether a 2nd-order or 5th-order polynomial provides a superior representation of the data. The standard statistical metric for such selection is the Akaike Information Criterion (AIC) [11], which estimates the relative information loss of a given model [5]. For datasets with finite sample sizes N, the Corrected AIC (AICc) is preferred to prevent the selection of over-parameterized models, a correction originally proposed to address bias in small-sample statistics [3][14]. The standard formulation for AICc in the context of least squares is:

$$AIC_c = N \ln\left(\frac{RSS}{N}\right) + 2k + \frac{2k(k+1)}{N-k-1}$$

Where RSS is the Residual Sum of Squares, and k is the number of estimated parameters (polynomial order + 1). However, in the context of datasheet generation, properties such as "visual smoothness" and adherence to physical laws are often more critical than minimizing the absolute residual error of noisy data. It was observed that the standard AICc tends to be too permissible regarding model complexity, frequently selecting higher-order polynomials that capture measurement noise rather than the underlying physical trend.

To address this limitation, a Modified AIC_{mod} metric was derived for this framework, introducing two empirical hyperparameters: an Alpha factor (α) and a Gamma factor (γ). The Alpha factor serves as an enhanced complexity penalty. By scaling the standard penalty term $2k$ by α. a significantly heavier cost is imposed on increasing the polynomial order. The modified AIC_{mod} is formulated as.

$$AIC_{mod} = N \ln\left(\frac{RSS}{N}\right) + 2k\alpha + \frac{2k(k+1)}{N-k-1}$$

For this application, α is set to 10 based on empirical tuning. This high penalty ensures that the algorithm rejects higher-order models unless they provide a statistically overwhelming improvement in the RSS, thereby biasing the selection towards simpler, more robust functions.

To further prioritize physical plausibility, a reward term is introduced via the Gamma factor. If a model configuration successfully converges while satisfying a

physical constraint such as Monotonicity or Positivity, the optimization score is reduced by the Gamma factor (γ).

$$Reward_{constraint} = \gamma \cdot N_{activeConstraints}$$

For this implementation, γ is set to 10. This value was selected to sufficiently incentivize the selection of physically constrained models without overriding the fundamental data trend if the error becomes too large. Consequently, the final optimization score utilized by the Autotuning algorithm combines the modified complexity penalty and the constraint reward:

$$\text{Final Score} = AIC_{mod} - (\gamma \times N_{constraints})$$

This modified criterion mathematically formalizes the engineering preference for simple, monotonic, and physically consistent models. By balancing error reduction with strict complexity penalties and constraint rewards, the software is enabled to autonomously make decisions that align with expert human judgment.

4 Design and Implementation

This chapter details the software engineering methodologies applied to transform the legacy analysis tool into a flexible, template-driven framework. The implementation follows a strict Object-Oriented Programming (OOP) paradigm [15] within the MATLAB App Designer environment, adhering to principles of clean code architecture [16]. The design strategy focused on decoupling the measurement configuration logic from the compiled application code, achieved through the development of a custom XML parsing engine and the extension of the underlying reactive data flow architecture.

Overview of the Extended Architecture

The software is built on a Model-View-Controller (MVC) architecture pattern [6]. The View is represented by the App Designer graphical interface, while the Model is encapsulated within a Reactive Node System. It is important to note that the base CalcNode class and the fundamental concept of the reactive graph were established in a previous iteration of the software. The primary architectural contribution of this thesis is the design of the Template Controller and the specific implementations of the data processing nodes (ScaleNode, SystemFilterNode) that utilize this infrastructure to support dynamic reconfiguration.

The reactive architecture operates on a Directed Acyclic Graph (DAG) principle, utilizing a "Pull" mechanism or Lazy Evaluation. In this system, a node does not automatically push data to its dependents when a change occurs. Instead, it marks itself and its dependents as "dirty." Calculation only occurs when a terminal node (such as a Diagram or Table) explicitly requests data. This design is critical for performance, as it prevents the unnecessary execution of computationally expensive fitting algorithms during intermediate configuration steps.

The extended data flow hierarchy implemented in this work is structured as follows:

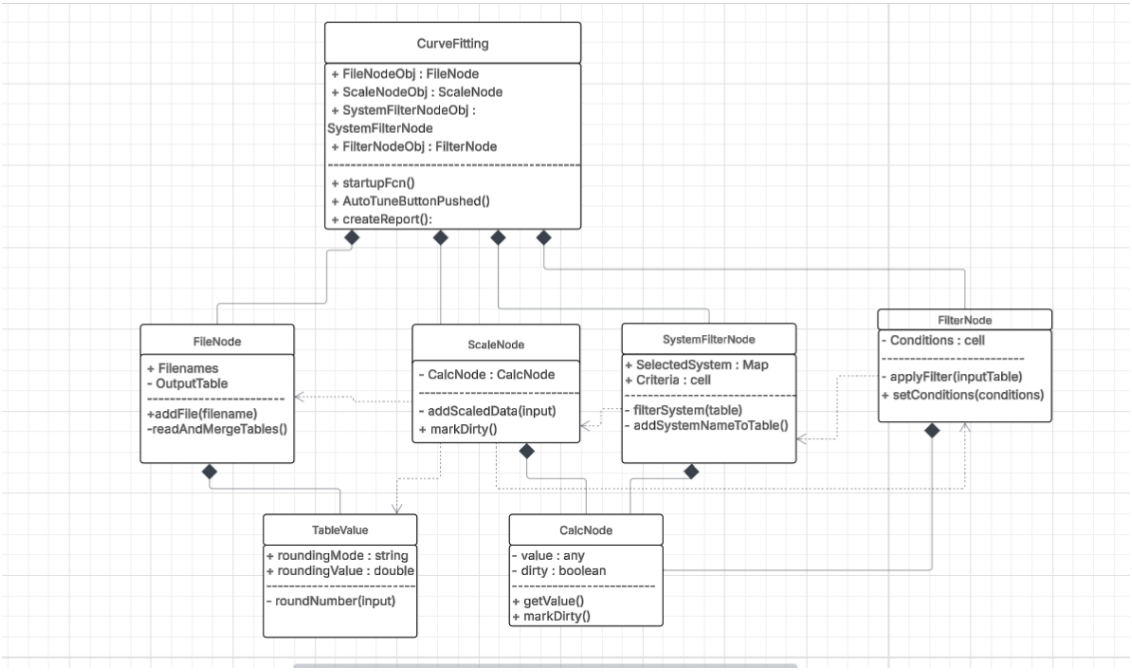
FileNode (Source): This node acts as the entry point, wrapping standard MATLAB file I/O operations. It is responsible for loading heterogeneous raw data formats (.txt, .csv, .xlsx) and merging them into a unified MATLAB table object.

ScaleNode (Intermediate): A dependent node that consumes the raw table from FileNode. It applies linear transformations to convert base units into engineering units based on injected scaling rules.

SystemFilterNode (Intermediate): A dependent node that consumes the scaled table. It implements a subtractive filtering logic to categorize data rows (e.g., "High-Side" vs. "Low-Side") based on boolean masks.

Diagram and Curve (Terminal): These objects represent the final analytical output. They request processed data from the SystemFilterNode and perform the mathematical curve fitting.

The Template Controller, implemented within the main application class (CurveFitting.mlapp), acts as the orchestrator. Upon loading an XML template, this controller parses the definition and programmatically instantiates and links these nodes, effectively rewriting the analysis logic at runtime without requiring a software recompile.



2. Figure: Simplified UML Class Structure diagram

The XML Template Framework

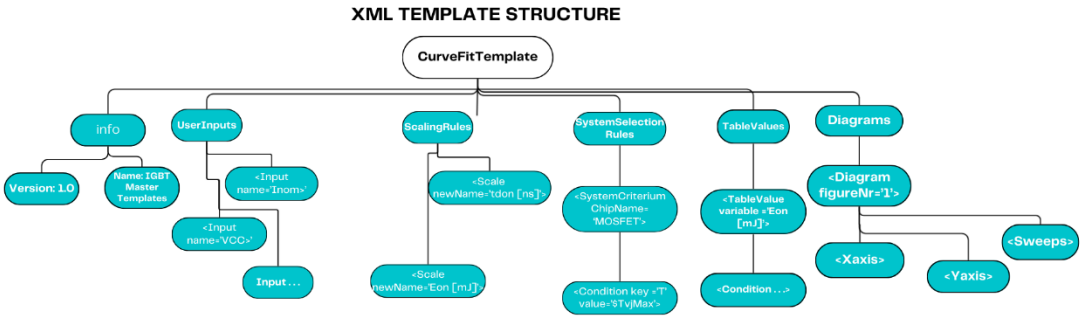
To satisfy the primary requirement of generalization, a robust configuration system was necessary to decouple the analysis logic from the compiled application code. The Extensible Markup Language (XML) [18] was selected as the unified configuration

standard for this framework. This design choice was driven by several key technical and operational advantages suited to the MATLAB environment and the end-user profile.

Firstly, the strict hierarchical nature of XML naturally mirrors the nested object-oriented architecture of the application (e.g., a Diagram contains Sweeps, which contain Conditions). This allows the configuration file to be deserialized directly into the runtime object graph without complex intermediate mapping. Secondly, the verbose, tag-based syntax of XML offers superior human-readability compared to formats such as JSON or binary configurations. This is critical for laboratory engineers who must create and modify templates manually without a specialized editor. Finally, MATLAB provides native support for the W3C Document Object Model (DOM) via the xmlread function, enabling efficient, standard-compliant parsing without the need for third-party libraries.

The application utilizes a custom DOM parser to traverse this structure. The parsing logic is divided into functional blocks that process the definition sequentially, ensuring that dependencies—such as variable definitions—are resolved before they are utilized by downstream components.

The logical hierarchy of these elements, illustrating the relationship between inputs, rules, and visualization objects, is depicted in 3. Figure: XML hierarchy tree.



3. Figure: XML hierarchy tree

To demonstrate the practical application of this schema, a minimal configuration template designed to generate a single "Switching Energy vs. Current" diagram is presented below:

```
<?xml version="1.0" encoding="UTF-8"?>
<CurveFitTemplate>
    <!-- 1. General Meta-Data -->
    <Info>
        <Name>Minimal MOSFET Template</Name>
        <Version>1.0</Version>
        <Description>Example    Minimal    configuration    for    Thesis
Documentation</Description>
    </Info>

    <!-- 2. User Inputs: Variables defining the operating point -->
    <UserInputs>
        <Input name="Inom"   column="Ic [A]" />
        <Input name="Imax"   column="Ic [A]" />
        <Input name="Vcc"    column="Vcc [V]"/>
        <Input name="TvjMax" column="T [°C]"/>
        <Input name="Rgon"   column="G1_Rgon [Ohm]"/>
    </UserInputs>

    <!-- 3. Scaling Rules: Normalization logic (e.g., Joules to mJ) -->
    <ScalingRules>
        <Scale  newName="Eon  [mJ]"  oldName="Eon  links  [J]"  scale="1000"
offset="0" />
    </ScalingRules>

    <!-- 4. System Selection: Logic to filter specific measurement rows -->
    <SystemSelectionRules>
        <SystemCriterium ChipName="MOSFET" Criterium="Eon [mJ]">
            <Condition key="T [°C]"    value="$TvjMax" />
            <Condition key="Ic [A]"    value="$Inom" />
            <Condition key="Vcc [V]"   value="$Vcc" />
            <Condition key="valid"     value="1" />
        </SystemCriterium>
    </SystemSelectionRules>

    <!-- 5. Diagram Definition: Visualization logic -->
    <Diagrams>
        <Diagram figureNr="1">
            <XAxis>Ic [A]</XAxis>
            <YAxis>Eon [mJ]</YAxis>
            <XLimits min="0" max="$Imax" />

            <!-- Global filters for this diagram -->
            <Conditions>
                <Condition key="Vcc [V]"    value="$Vcc" />
                <Condition key="valid"      value="1" />
            </Conditions>

            <!-- Curve families to generate -->
            <Sweeps>
                <SweepParameter>T [°C]</SweepParameter>
                <SweepParameter>Rgon [Ohm]</SweepParameter>
```

```
                <!-- Curve 1: Nominal Gate Resistor at Max Temp -->
                <SweepCombination>
                    <Value>$TvjMax</Value>
                    <Value>$Rgon</Value>
                </SweepCombination>
            </Sweeps>
        </Diagram>
    </Diagrams>
</CurveFitTemplate>
```

4.1.1 User Inputs and Variable Substitution

The root of the configuration is the <UserInputs> section. This block defines global variables such as nominal current (I_{nom}), bus voltage (V_{cc}), or junction temperature (T_{vj}) which establish the specific operating conditions of the dataset. Upon loading a template, the parser identifies these tags and dynamically generates an interactive UI table, prompting the user for numerical entry.

Internally, the Template Controller utilizes a containers.Map data structure to store these key-value pairs. This map serves as a lookup table for variable substitution. During the parsing of subsequent nodes, the algorithm inspects every attribute string for the $ delimiter. If a string such as value="$$I_{nom}$" is encountered, the system queries the map and injects the corresponding numerical value into the object property. This mechanism ensures that a generic template can be reused across different device ratings simply by updating the input table.

4.1.2 Scaling Rules

The <ScalingRules> block defines the data normalization logic. Each rule is encapsulated in a <Scale> tag containing four mandatory attributes: the target column name (oldName), the desired variable name (newName), a multiplicative factor (scale), and an additive offset (offset). This structure allows for the mapping of raw data columns (e.g., "t_fall [s]") to standardized internal names (e.g., "tf [ns]") required by the plotting engine.

4.1.3 System Selection Logic

The <SystemSelectionRules> section enables the automated segmentation of the dataset. The schema employs a nested structure where a parent <SystemCriterium> tag contains multiple child <Condition> tags. The logic implies an AND operation between conditions within a criterium. The parser reads these definitions and passes them to the

SystemFilterNode, where they are converted into logical vector masks that filter the data table.

4.1.4 Diagram Definitions

The visualization logic is defined in the <Diagrams> section. This part of the schema mirrors the hierarchical structure of the Diagram class. A <Diagram> element defines the axes and limits, while nested <Sweeps> and <SweepCombination> elements define the parameter sets required to generate specific curve families. The parser iterates through these nested elements to instantiate the necessary objects, automatically assigning the correct X and Y data sources and filtering conditions.

Implementation of Unit Scaling

The practical implementation of data normalization is encapsulated within the ScaleNode class but is controlled via the main application workflow. The core logic applies a linear transformation equation ($y = m \cdot x + c$) to the data vectors.

To ensure the stability of the tool in a production laboratory environment, the scaling algorithm was implemented with a strong focus on fault tolerance. Experimental datasets frequently vary; a specific column defined in a standard template might be missing from a specific measurement file due to channel configuration changes on the oscilloscope. To handle this, the scaling iteration loop is wrapped in a try-catch block.

When the addScaledData method executes, it attempts to locate the oldName column in the source table. If the column is missing, the system catches the exception, logs a specific warning to the console, and gracefully skips that individual rule without interrupting the execution of the remaining scaling operations. This design decision ensures that the analysis pipeline remains robust, allowing engineers to visualize the available data even if the dataset is partially incomplete.

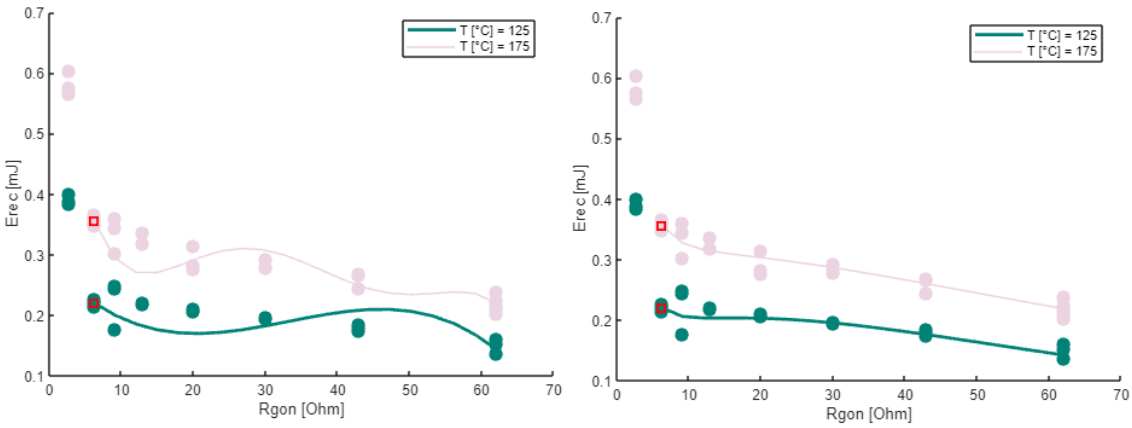
Extended Curve Fitting Engine

The mathematical core of the application resides within the Curve class. To enable the analysis of non-linear semiconductor characteristics without abandoning the numerical stability of the Tikhonov-regularized linear solver, the fitting engine was fundamentally refactored. The extended implementation introduces a "Transform-Solve-

Inverse" architectural pattern, encapsulated primarily within the fitHelper and regularizedPolyFitWeighted methods.

The process begins in the fitHelper method, which acts as a pre-processing pipeline. Before any optimization occurs, the engine evaluates the state of the LogX and LogY boolean properties. A critical step in this phase is data validation. Since the natural logarithm function is undefined for non-positive values, the method applies a logical filter to the raw data vectors. Data points where $x \leq 0 \; or \; y \leq 0$ (depending on the active axes) are automatically excluded from the fitting set. This defensive programming ensures that the solver never encounters domain errors, which is particularly important when handling raw measurement data that may contain zero-crossings due to sensor noise or offset calibration issues.

Once validated, the data is projected into the target feature space. If a logarithmic mode is active, the log transformation is applied to the respective vectors. The efficacy of this transformation is illustrated in 4. Figure. The raw measurement data, exhibiting a non-linear power-law decay (Left), is mapped to a linear trajectory within the logarithmic feature space (Right).



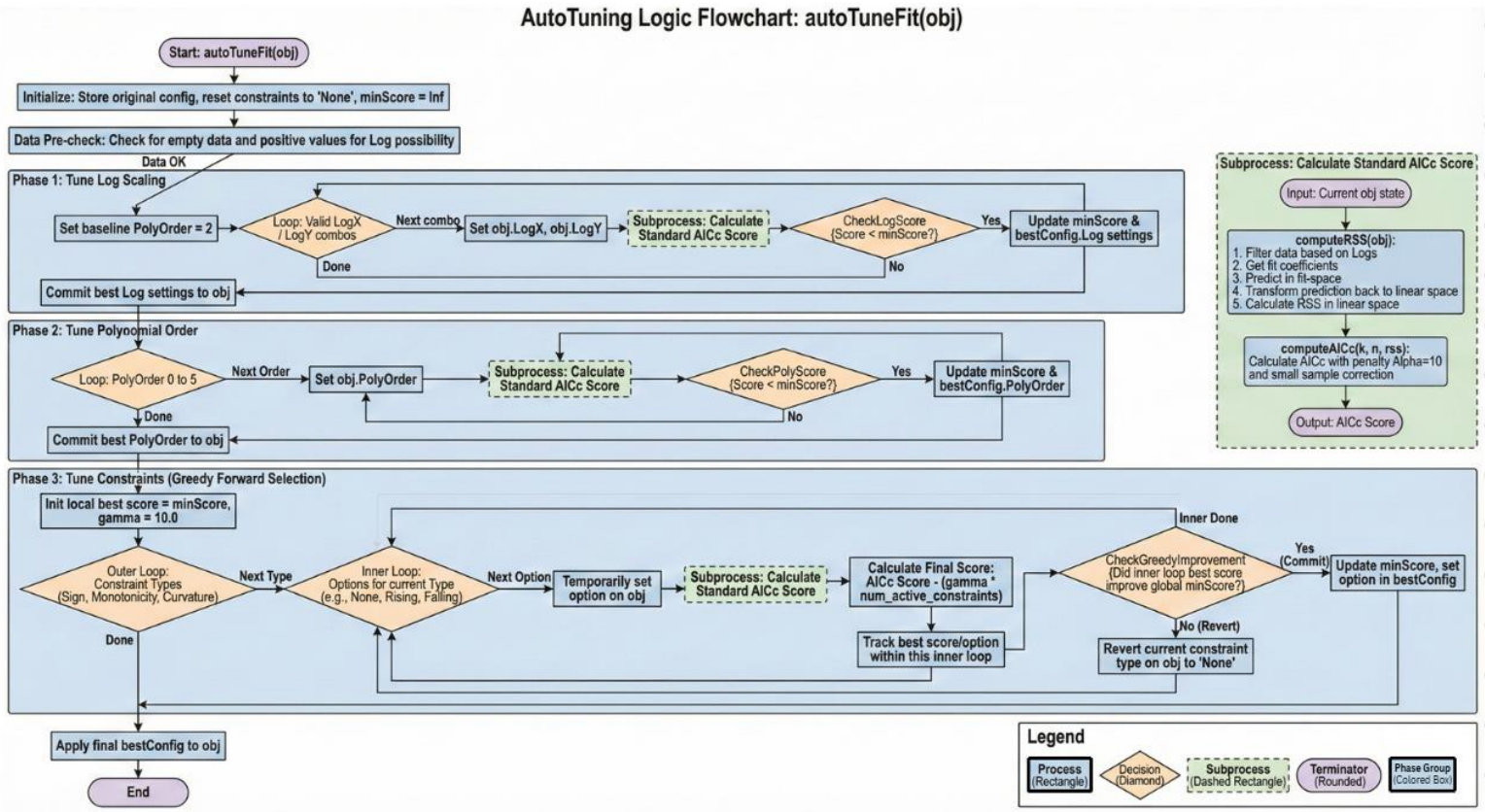
4. Figure: Left: Raw Data (Left) vs Transformed Data (Right)

The solver constructs the Tikhonov regularization matrix (H) and the linear constraint vectors within this transformed, normalized coordinate system. The optimization problem is solved using the quadprog function from the MATLAB Optimization Toolbox [20] (Interior-Point-Convex algorithm [4]) to determine the optimal polynomial coefficients. Finally, the resulting curve is mapped back to the original physical domain via the inverse transformation. For logarithmic axes, the exp() function is applied to the fitted vector. This architecture ensures that the resulting curve

strictly adheres to the physical trend such as a power law while maintaining the smoothness properties enforced by the regularization term.

Hyperparameter Autotuning Algorithm

To automate the selection of the optimal mathematical model, the autoTuneFit method was implemented within the Curve class. This method treats the fitting configuration as a discrete optimization problem where the objective is to minimize the Modified Akaike Information Criterion (AICc) derived in Chapter 3. Given the high dimensionality of the search space comprising coordinate systems, polynomial orders, and multiple physical constraints an exhaustive search would be computationally prohibitive for real-time interaction. Consequently, the implementation utilizes a deterministic Greedy Forward Selection strategy, executed in three sequential stages.



5. Figure: Flowchart of the Autotuning Logic

4.1.5 Step 1: Coordinate System Selection

The algorithm first seeks to identify the coordinate space that best linearizes the data. To isolate the effect of the coordinate transformation from model complexity, the polynomial order is temporarily fixed to a low baseline (Order = 2). The algorithm iterates

through the four permutation states of the LogX and LogY properties (Linear, Semi-LogX, Semi-LogY, and Log-Log). For each permutation, the Residual Sum of Squares (RSS) is computed in the linear back-transformed space to ensure comparable error metrics. The configuration yielding the lowest initial AICc score is committed as the foundation for the subsequent steps.

4.1.6 Step 2: Complexity Optimization

With the optimal coordinate system locked, the algorithm proceeds to optimize the model complexity. The logic iterates through polynomial orders ranging from 0 (constant) to 5. During this loop, the AICc score is calculated using the specific Alpha penalty factor (α=10) implemented in the computeAICc helper function. This high penalty factor imposes a strict barrier against overfitting; a higher-order polynomial is selected only if it provides a reduction in residual error substantial enough to outweigh the complexity cost. This step effectively locates the "knee point" of the error curve, identifying the simplest polynomial that adequately captures the data trend.

4.1.7 Step 3: Constraint Selection (Greedy Approach)

The final stage refines the model by applying physical constraints. The implementation utilizes a greedy search pattern to test constraints in a fixed hierarchy: first Sign, then Monotonicity, and finally Curvature.

For each constraint type, the algorithm iteratively tests all available options (e.g., for Monotonicity: 'None', 'Rising', 'Falling'). A specialized scoring logic is applied here:

$$\text{Final Score} = AIC_{mod} - (\gamma \times N_{constraints})$$

The autoTuneFit method explicitly defines the Gamma reward factor ($\gamma = 10$). If applying a constraint (e.g., forcing the curve to be monotonic) results in a score lower than the current best score, the constraint is accepted and committed to the configuration. This mathematical bias ensures that the algorithm prioritizes physically plausible models. For example, even if a noisy dataset suggests a slight local oscillation, the reward factor will drive the selection toward a monotonic curve if the increase in residual error is minimal. This logic effectively replicates the decision-making process of an expert engineer who prioritizes physical consistency over absolute fitting precision.

5 Evaluation

This chapter presents the comprehensive evaluation of the developed Data-Sheet Analysis Tool. The primary objective of this phase was to validate the functional correctness of the template-driven architecture and to quantify the improvements in analytical accuracy and workflow efficiency compared to the legacy manual approach. The evaluation process was structured to isolate and verify each stage of the extended pipeline, from initial data parsing to the final generation of mathematically optimized diagrams.

Test Environment and Methodology

The software validation was conducted using MATLAB R2022b on standard laboratory workstations. To ensure the results reflected real-world engineering challenges, the evaluation utilized experimental datasets provided by Infineon Technologies. These datasets consisted of characterization measurements for discrete IGBT and MOSFET power devices, generated during standard double-pulse testing procedures.

These specific datasets were selected because they represent the "worst-case" scenarios for automated analysis. They are characterized by high dynamic ranges, with currents spanning from milliampere-level leakage measurements to transient short-circuit currents exceeding hundreds of Amperes. Furthermore, the data exhibits significant stochastic noise, particularly in derivative-based parameters such as current slope ($\frac{di}{dt}$) and voltage slope ($\frac{dv}{dt}$), which are derived from high-speed oscilloscope acquisitions. The validation methodology focused on three key performance indicators: the correct propagation of XML configuration variables, the robustness of the logarithmic fitting engine when applied to non-linear physical behaviors, and the ability of the Autotuning algorithm to converge on physically plausible models without human intervention.

Verification of Automation Framework

The initial phase of testing focused on the verification of the XML Templating System. A comprehensive test template, MOSFET_Template.xml, was developed to define a complete characterization session. This template included definitions for user-

defined global variables, scaling rules for twenty distinct measurement channels, and the structural definitions for twelve unique diagrams.

5.1.1 Template Loading and Variable Substitution

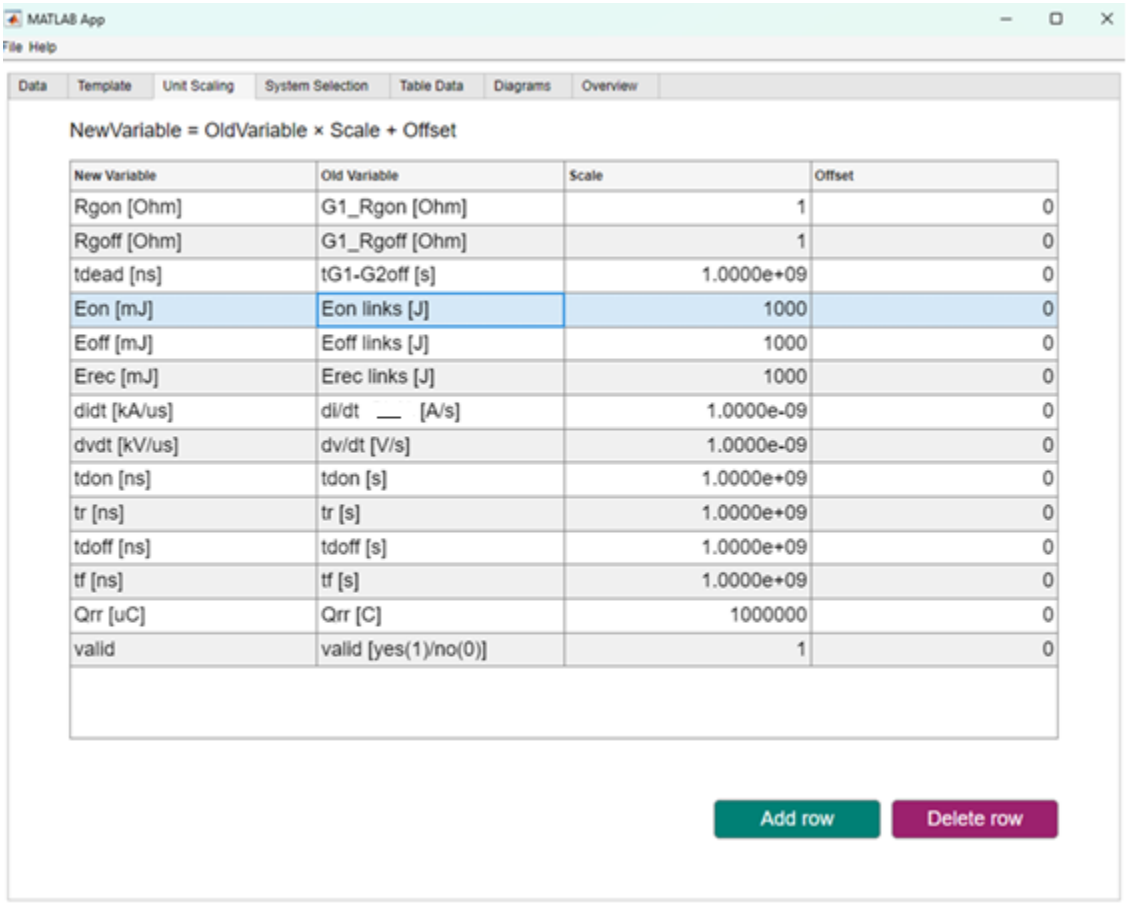
The application's ability to parse and instantiate the analysis context was tested by loading the XML template. Upon file selection, the Template Controller successfully identified the <UserInputs> section and dynamically generated the interactive parameter table within the user interface. Values entered by the operator such as the nominal current (I_{nom}), bus voltage (V_{cc}), and maximum junction temperature (T_{vjMax}) were correctly captured by the internal map container.

The propagation of these variables was verified by examining the downstream filtering logic. The SystemFilterNode correctly substituted the user-defined voltage thresholds (e.g., $Vgsoff) into the logical mask definitions. This allowed the software to automatically segregate the raw dataset, distinguishing between high-side and low-side switching events based on the dynamic criteria defined in the template, a task that previously required manual row selection in external spreadsheet software.

5.1.2 Automated Unit Scaling

The robustness of the data normalization engine was verified through the ScaleNode output. The raw measurement files contained data in base SI units (Joules for energy, Seconds for time). The template defined transformation rules to convert these into standard engineering units (milliJoules and nanoseconds). Inspection of the processed data table confirmed that the linear transformations were applied correctly to all target columns.

Furthermore, the fault-tolerance of the scaling logic was validated by intentionally loading a raw dataset that was missing specific columns defined in the template. As designed, the try-catch mechanism within the scaling loop successfully trapped the missing column exception, logged a warning to the console, and proceeded to process the remaining valid columns without terminating the application. This confirmed that the architectural requirement for stability in the presence of inconsistent experimental data was met.



6. Figure: Unit Scaling Tab showing automatic conversion of units

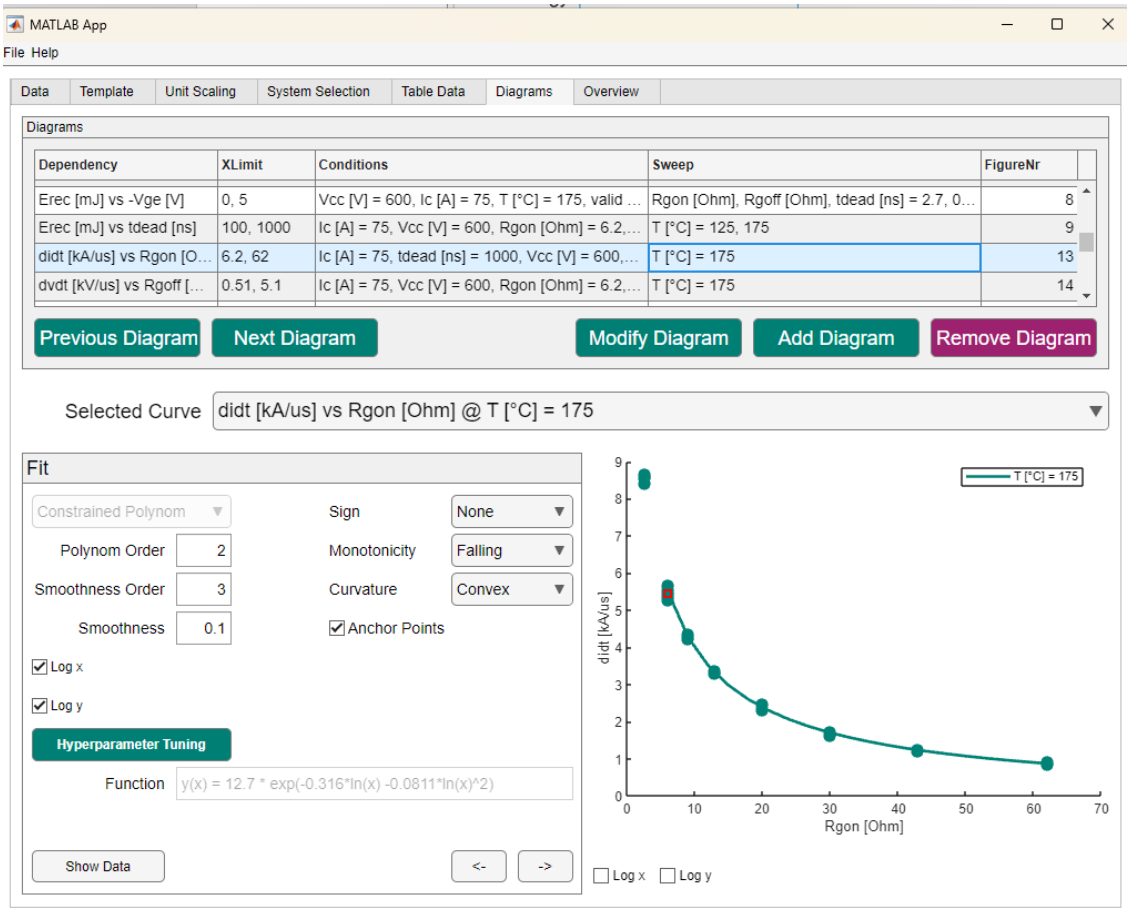
Evaluation of Analytical Capabilities

A critical limitation of the previous linear solver was its inability to robustly model the non-linear dependencies characteristic of power semiconductor switching behaviors. To evaluate the effectiveness of the extended mathematical engine, a comparative analysis was performed using the rate of current rise ($\frac{di}{dt}$) as a function of the turn-on gate resistance (R_{gon}). This relationship represents a fundamental trade-off in power electronics: increasing the gate resistance slows down the switching transient, reducing electromagnetic interference but increasing switching losses. Physically, this manifests as an inverse or power-law decay curve which exhibits sharp curvature at low resistance values and an asymptotic approach to zero at high resistance values.

When fitted using the legacy standard polynomial approach in a linear coordinate system, the model frequently exhibited unphysical artifacts. To minimize the residual sum of squares across the high-dynamic-range axis, high-order polynomials would oscillate significantly, often dipping below zero in the asymptotic region to accommodate the steep

gradient at the origin. These oscillations resulted in physically impossible negative values for the current slope.

In contrast, the extended engine was tested using the Log-Log transformation mode ($x' = \ln x,\ y' = \ln y$). As illustrated in the resulting analysis, the solver successfully linearized the data before optimization. The fit function generated by the tool, expressed as $y(x) = 12.7 \times \exp(-0.316 \ln(x) - 0.0811 \ln(x)^2)$, confirms that the "Transform-Solve-Inverse" architecture functioned correctly. The resulting curve adhered strictly to the physical trend, passing smoothly through the measurement points without oscillation and maintaining a strictly positive trajectory throughout the domain. This test confirmed that the integration of coordinate transformations allows the Tikhonov-regularized solver to model complex power-law behaviors with high fidelity.



7. Figure: "didt vs Rgon" curve after auto-tuning

Autotuning Performance and Workflow Analysis

The Hyperparameter Autotuning algorithm was evaluated to determine its ability to replicate the decision-making process of an expert engineer. The evaluation utilized the same $\frac{di}{dt}$ dataset to verify the Greedy Forward Selection logic.

The execution log of the algorithm revealed the sequential optimization process. In the first stage, the algorithm compared the Modified AICc scores across linear and logarithmic coordinate systems, correctly identifying that the Log-Log space provided the most significant reduction in linearization error. In the second stage, the algorithm evaluated polynomial complexity. Despite the potential for a higher-order polynomial to reduce the residual error marginally further, the Alpha penalty factor (α=10) imposed by the Modified AICc metric heavily penalized complexity. Consequently, the algorithm converged on a stable 2nd-order polynomial, effectively identifying the mathematical "knee point" where additional complexity no longer yielded statistically significant accuracy gains.

In the final stage, the constraint selection logic was verified. The greedy search iteratively tested physical constraints. The algorithm correctly identified that applying a "Monotonicity: Falling" constraint and a "Curvature: Convex" constraint improved the overall optimization score. This improvement was driven by the Gamma reward factor (γ=10), which incentivized the selection of a physically descriptive model over a purely unconstrained mathematical fit. The final output was a smooth, monotonic, and convex curve that aligned perfectly with theoretical expectations for a gate resistance dependency.

To quantify the efficiency gains, a workflow timing analysis was conducted. In the manual workflow, generating such a diagram required the user to visually inspect the data, hypothesize the correct axis transformation, and iteratively adjust the polynomial order and smoothing factors to remove oscillations. This process typically required several minutes of interaction per diagram. The automated workflow shifted this burden to computational complexity. Although the autotuning algorithm performed approximately 70 to 80 quadratic programming solves to explore the search space, the entire optimization process completed in less than two seconds on standard laboratory hardware. This represents a time reduction of approximately 98% for the curve-fitting

phase, while simultaneously ensuring that the generated models are mathematically consistent and reproducible across different analysis sessions.

Below is a snippet of the MATLAB Console Output showing the Autotuning progress step by step:

```
--- Starting Hyperparameter AutoTuning ---
Data Pre-check:
  - X-axis data is all positive. LogX will be tested.
  - Y-axis data is all positive. LogY will be tested.

--- Step 1: Tuning Log Scaling (using baseline PolyOrder=2) ---
  Testing LogX=0, LogY=0: RSS=34, n=39, k=3 -> AICc Score = 55.3601
    *** NEW BEST FOUND -> Score: 55.3601 ***
  Testing LogX=0, LogY=1: RSS=7.78, n=39, k=3 -> AICc Score = -2.2050
    *** NEW BEST FOUND -> Score: -2.2050 ***
  Testing LogX=1, LogY=0: RSS=0.189, n=39, k=3 -> AICc Score = -147.1032
    *** NEW BEST FOUND -> Score: -147.1032 ***
  Testing LogX=1, LogY=1: RSS=0.178, n=39, k=3 -> AICc Score = -149.5366
    *** NEW BEST FOUND -> Score: -149.5366 ***
  ==> Best Log Config Found: LogX=1, LogY=1

--- Step 2: Tuning Polynomial Order ---
  Testing PolyOrder=0: RSS=326, n=39, k=1 -> AICc Score = 102.9456
  Testing PolyOrder=1: RSS=4.23, n=39, k=2 -> AICc Score = -46.3134
  Testing PolyOrder=2: RSS=0.178, n=39, k=3 -> AICc Score = -149.5366
  Testing PolyOrder=3: RSS=0.178, n=39, k=4 -> AICc Score = -128.9455
  Testing PolyOrder=4: RSS=0.218, n=39, k=5 -> AICc Score = -100.5093
  Testing PolyOrder=5: RSS=0.184, n=39, k=6 -> AICc Score = -86.1928
  ==> Best Polynomial Order Found: 2

--- Step 3: Tuning Constraints (Greedy Forward Selection) ---
  -> Evaluating constraint type: Sign
    Testing Sign='None'    : AICc=-149.5366, Reward=0.0 -> Final Score = -
149.5366
    Testing Sign='Positive': AICc=-115.4613, Reward=10.0 -> Final Score = -
125.4613
    Testing Sign='Negative': AICc=152.6612, Reward=10.0 -> Final Score =
142.6612
    -> No improvement found. Keeping Sign = 'None'.
  -> Evaluating constraint type: Monotonicity
    Testing Monotonicity='None'    : AICc=-149.5366, Reward=0.0 -> Final
Score = -149.5366
    Testing Monotonicity='Rising'  : AICc=190.3432, Reward=10.0 -> Final
Score = 180.3432
    Testing Monotonicity='Falling': AICc=-149.5366, Reward=10.0 -> Final
Score = -159.5366
    *** NEW BEST FOUND -> Committing Monotonicity = 'Falling', New Best
Score: -159.5366 ***
  -> Evaluating constraint type: Curvature
    Testing Curvature='None'   : AICc=-149.5366, Reward=10.0 -> Final Score
= -159.5366
    Testing Curvature='Convex' : AICc=-149.5366, Reward=20.0 -> Final Score
= -169.5366
    Testing Curvature='Concave': AICc=143.5232, Reward=20.0 -> Final Score
= 123.5232
```

```
    *** NEW BEST FOUND -> Committing Curvature = 'Convex', New Best Score:
-169.5366 ***

--- AutoTuning Complete. Applying Final Configuration ---
Final Best Score: -169.5366
            LogX: 1
            LogY: 1
       PolyOrder: 2
            Sign: 'None'
    Monotonicity: 'Falling'
       Curvature: 'Convex'
```

To provide a comprehensive overview of the validation results, the performance metrics observed during the evaluation phase are synthesized in 1. Table: Comparative Evaluation of Performance Metrics. This comparative analysis underscores the substantial advancements in workflow efficiency and analytical fidelity achieved by the automated framework relative to the legacy manual approach.

Metric	**Legacy Manual Workflow**	**Automated Framework**	**Outcome**
Processing Latency	Required approximately 3–5 minutes per diagram due to iterative manual adjustment.	Completed in less than two seconds via autonomous optimization algorithms.	~98% reduction in the time required for curve fitting and diagram generation.
Reproducibility	Characterized by subjectivity; results were dependent on operator intuition and trial-and-error.	Deterministic; governed by the Modified Akaike Information Criterion (AIC_{mod}).	Elimination of human variability, ensuring consistent results across different analysis sessions.
Analytical Fidelity	Prone to numerical instability (Runge's phenomenon) and unphysical negative artifacts.	Strictly adhered to physical constraints (e.g., Monotonicity, Positivity, Convexity).	Robust modeling of non-linear semiconductor behaviors, even with high-noise datasets.

Scalability	Relied on hardcoded logic; required source code modification and recompilation for new devices.	Driven by XML templates; allows for dynamic runtime reconfiguration.	Decoupled architecture enabling immediate adaptation to novel device topologies without software development.

1. Table: Comparative Evaluation of Performance Metrics

6 Critical Assessment

The development of the Data-Sheet Analysis Tool represented a multifaceted engineering challenge that required the synthesis of software architecture, mathematical modeling, and user interface design. This chapter assesses the outcomes of the project against the initial requirements defined in Chapter 2 and discusses the specific technical challenges encountered during the implementation and verification phases.

Assessment of Completed Work

The primary objectives of this thesis to generalize the analysis workflow and to automate the mathematical modeling process have been successfully met. The transformation of the software from a rigid, code-dependent utility into a flexible, template-driven framework was achieved through the design and implementation of the XML Template Controller. By decoupling the analysis logic from the compiled source code, the system now satisfies the requirement for scalability; laboratory engineers can adapt the tool to novel device architectures and measurement protocols solely by modifying external configuration files, without requiring intervention from software developers.

From an analytical perspective, the integration of the coordinate transformation engine has resolved the tool's previous inability to robustly model non-linear physical behaviors. The Tikhonov-regularized solver, originally limited to linear polynomial fitting, now operates effectively across logarithmic and semi-logarithmic domains. This extension ensures that critical semiconductor parameters, such as power-law switching energy dependencies and exponential leakage currents, are modeled with high fidelity. Furthermore, the development of the Hyperparameter Autotuning algorithm has successfully removed the subjectivity from the curve-fitting process. By utilizing the Modified Akaike Information Criterion, the software consistently selects models that balance statistical accuracy with physical plausibility, thereby ensuring reproducibility across different analysis sessions and reducing the manual effort required to generate publication-ready diagrams.

Challenges Encountered

A significant challenge during the development phase was the calibration of the Autotuning Algorithm. While the Akaike Information Criterion is a standard statistical measure for model selection, the concept of "visual smoothness" required for datasheets is a subjective engineering preference that is difficult to quantify mathematically. Finding the optimal balance for the empirical Alpha (complexity penalty) and Gamma (constraint reward) factors required extensive testing with real-world datasets. Initial experiments revealed that standard AICc penalties were insufficient for high-noise oscilloscope data, often leading the algorithm to select high-order polynomials that modeled the noise rather than the signal. Conversely, an excessively high Gamma factor occasionally forced the selection of overly simplistic models that ignored genuine trend inflections. A robust set of default factors was eventually determined empirically, ensuring the algorithm behaves consistently across the typical dynamic ranges found in power electronics measurements.

In addition to algorithmic tuning, the extension of the reactive architecture introduced complex state-management issues. The underlying reactive graph relies on persistent listeners to propagate data changes. Dynamically reconfiguring this graph destroying old diagram objects and instantiating new ones when a template is loaded created synchronization challenges. Early iterations of the software suffered from memory leaks and execution errors caused by "zombie" listeners attempting to update deleted figures. This was resolved by implementing a rigorous cleanup protocol within the base node classes, ensuring that all event listeners are explicitly detached and object handles are cleared before the state is reset.

Future Development Options

While the developed tool successfully streamlines the interactive analysis workflow, the evaluation identified a clear pathway for further automation through batch processing. Currently, the Autotuning function must be triggered manually for each curve or diagram to allow the user to verify the result. A logical future enhancement would be to integrate autotuning instructions directly into the XML schema. By introducing a specific attribute—for example, a flag indicating that a specific curve should be auto-tuned upon loading—the software could iterate through the entire session immediately after the template is applied. This would enable the generation of a complete,

mathematically optimized report containing dozens of diagrams with zero additional user interaction, further accelerating the characterization cycle for standardized products.

7 Conclusion

The analysis of experimental measurement data constitutes a cornerstone of power semiconductor development. As device complexity increases to meet the demands of modern energy efficiency standards, the reliance on manual data processing workflows has emerged as a significant bottleneck, introducing latency and potential inconsistencies into the datasheet generation process. The necessity to characterize devices under a wide array of operating conditions generates vast datasets that require meticulous filtering, scaling, and mathematical modeling to extract meaningful physical parameters.

This thesis presented the design and implementation of an advanced Data-Sheet Analysis Tool aimed at resolving these challenges through the principles of automation and generalization. By extending an existing MATLAB App Designer framework, the project introduced a robust XML-based templating system. This architectural innovation allows laboratory engineers to externalize the logic for data filtering, unit scaling, and diagram definition, effectively decoupling the analysis configuration from the application source code. Consequently, the software has been transformed from a rigid, device-specific utility into a flexible platform adaptable to any semiconductor technology without requiring code modification.

Furthermore, the analytical capabilities of the tool were significantly expanded to address the specific physical behaviors of power devices. The implementation of a Tikhonov-regularized solver capable of coordinate transformations ensures that non-linear characteristics—such as exponential leakage currents and power-law switching energy dependencies—are modeled with high fidelity and numerical stability. The introduction of the Hyperparameter Autotuning algorithm, driven by a modified Akaike Information Criterion with specific penalties for complexity and rewards for physical constraints, guarantees that the generated curves are not only mathematically optimal but also adhere to the physical laws of the system.

In conclusion, the developed tool successfully transforms a fragmented, manual workflow into a streamlined, reproducible, and highly efficient process. It empowers laboratory engineers at Infineon Technologies to focus on the interpretation of device physics rather than the mechanics of data manipulation, thereby significantly accelerating

the characterization and development cycle of next-generation power electronic components.

References

[1] J. Lutz, H. Schlangenotto, U. Scheuermann, and R. De Doncker, Semiconductor Power Devices: Physics, Characteristics, Reliability, 2nd ed. Berlin, Germany: Springer-Verlag, 2011.

[2] P. C. Hansen, Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion. Philadelphia, PA: SIAM, 1998.

[3] C. M. Hurvich and C. L. Tsai, "Regression and time series model selection in small samples," Biometrika, vol. 76, no. 2, pp. 297–307, 1989.

[4] J. Nocedal and S. J. Wright, Numerical Optimization, 2nd ed. New York, NY: Springer, 2006.

[5] K. P. Burnham and D. R. Anderson, Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach, 2nd ed. New York, NY: Springer-Verlag, 2002.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley, 1994.

[7] A. N. Tikhonov and V. Y. Arsenin, Solutions of Ill-Posed Problems. Washington, DC: V. H. Winston & Sons, 1977.

[8] B. J. Baliga, Fundamentals of Power Semiconductor Devices. New York, NY: Springer Science & Business Media, 2008.

[9] Semiconductor devices – Part 9: Discrete devices – Insulated-gate bipolar transistors (IGBTs), IEC 60747-9:2007, International Electrotechnical Commission, 2007.

[10] Infineon Technologies AG, "Definition of Switching Losses for IGBTs and MOSFETs," Application Note AN-2015-06, V1.0, Munich, Germany, 2015.

[11] H. Akaike, "A new look at the statistical model identification," IEEE Transactions on Automatic Control, vol. 19, no. 6, pp. 716–723, 1974.

[12] G. H. Golub and C. F. Van Loan, Matrix Computations, 4th ed. Baltimore, MD: Johns Hopkins University Press, 2013.

[13] C. L. Lawson and R. J. Hanson, Solving Least Squares Problems. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[14] N. Sugiura, "Further analysis of the data by Akaike's information criterion and the finite corrections," Communications in Statistics - Theory and Methods, vol. 7, no. 1, pp. 13–26, 1978.

[15] I. Sommerville, Software Engineering, 10th ed. Boston, MA: Pearson, 2015.

[16] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River, NJ: Prentice Hall, 2008.

[17] M. Fowler, Patterns of Enterprise Application Architecture. Boston, MA: Addison-Wesley, 2002.

[18] E. R. Harold and W. S. Means, XML in a Nutshell, 3rd ed. Sebastopol, CA: O'Reilly Media, 2004.

[19] MathWorks, MATLAB App Designer User's Guide, R2022b. Natick, MA: The MathWorks, Inc., 2022.

[20] MathWorks, Optimization Toolbox User's Guide, R2022b. Natick, MA: The MathWorks, Inc., 2022.

[21] J. Tidwell, Designing Interfaces: Patterns for Effective Interaction Design, 2nd ed. Sebastopol, CA: O'Reilly Media, 2010.

[22] W. S. Cleveland, The Elements of Graphing Data, 2nd ed. Summit, NJ: Hobart Press, 1994.

Annex

I. Declaration on the Use of Generative Artificial Intelligence

☐ **I have not used** any generative AI tools.

☒ **I have used generative AI tools.** I have verified the content generated by AI, ensured the accuracy of the outputs, and properly indicated each instance of use in the table below.

Usage type	**Name of Generative AI Tool(s)**	**Affected Sections** (chapter, page number, reference)	**Estimated Proportion of Use** (per usage type)
Literature Review	Chatgpt 4	Chapters 1 to 7	40% of the text was checked/improved
Brief Summary of the Prompt	Language proofreading and style improvement.		
Program Code Generation			
Brief Summary of the Prompt			
Generating New Ideas or Solution Proposals			
Brief Summary of the Prompt			
Creating an Outline (text structure, bullet points)			
Brief Summary of the Prompt			
Creating Text Blocks			
Brief Summary of the Prompt			
Generating Images for Illustrative Purposes			
Brief Summary of the Prompt			

Data Visualization, Generating Charts Based on Data Points			
Brief Summary of the Prompt			
Preparing a Presentation			
Brief Summary of the Prompt			
Other (please specify)			
Brief Summary of the Prompt	Proofreading and finding errors		
Aggregated Percentage Value (for the core part of the task)			**0%**
Brief Textual Justification of the Aggregated Value: I performed the professional content, research, measurements, and software development independently. I used generative AI exclusively to check the grammar of the English text written by myself and to improve the scientific style and vocabulary. The AI did not add any new professional ideas or results to the thesis.			

II . Attachements

The following digital artifacts are attached to this thesis to demonstrate the functionality of the developed tool. all the attachments can be found in the Digital repository linked at 4:

1. DataSheetAnalysisTool.exe: The standalone compiled executable of the developed software. This application runs the full analysis pipeline demonstrated in the evaluation chapter.
2. MOSFET_Template.xml: The XML configuration file used for the primary evaluation cases (Switching energy, Di/Dt).
3. IGBT_Template.xml: An additional configuration template demonstrating the adaptability of the system to different device technologies.

4. Digital Repository[i]: Snippets of code, Functions and methods developed for this thesis. https://github.com/abdel-ahbane/DataSheet-Analysis-Automation-Tool

[i] The complete source code for the application is proprietary to Infineon Technologies and is not included in the public repository. The core algorithms developed specifically for this thesis (Autotuning and Logarithmic Logic) are documented in the annexed Digital repository.