Master‘s Thesis

**Design and Development of a MATLAB to Python GUI for Flexible Biogas Production; A Case Study on Biogas Production Management.**

Submitted by:

**Sanju Pallissery Babu**

(matriculation number: 03745203)

|  |  |
| --- | --- |
| Date of Submission: |  |
| Examiner: | Prof. Dr. Matthias Gaderer (Technical University of Munich, Straubing) |
| Supervisors: | Lingga Aksara Putra (Technical University of Munich, Straubing) |
|  | Title First name last name (name of university) |

Affidavit

I hereby certify that this bachelor's, master's, diploma or doctoral thesis was written without the help of third parties and only with the sources and aids indicated. All passages used have been identified. This thesis has not yet been submitted in the same or a similar form to any examination authority.

(Signature)

Sanju Pallissery Babu

Straubing, 01.04.2025

Abstract

Short summary (< 1/2 page) including the assignment, most important assumptions and results.

Contents

[Affidavit II](#_Toc181282314)

[Abstract III](#_Toc181282315)

[Contents IV](#_Toc181282316)

[List of Figures V](#_Toc181282317)

[List of Tables VI](#_Toc181282318)

[Nomenclature VII](#_Toc181282319)

[1 Structure Error! Bookmark not defined.](#_Toc181282320)

[2 Layout Error! Bookmark not defined.](#_Toc181282321)

[2.1 General **Error! Bookmark not defined.**](#_Toc181282322)

[2.2 Header and Footer **Error! Bookmark not defined.**](#_Toc181282323)

[2.3 Directories **Error! Bookmark not defined.**](#_Toc181282324)

[2.3.1 List of Contents **Error! Bookmark not defined.**](#_Toc181282325)

[2.3.2 List of Figures **Error! Bookmark not defined.**](#_Toc181282326)

[2.3.3 List of tables **Error! Bookmark not defined.**](#_Toc181282327)

[2.4 Nomenclature **Error! Bookmark not defined.**](#_Toc181282328)

[2.5 Bibliography **Error! Bookmark not defined.**](#_Toc181282329)

[2.6 Appendix **Error! Bookmark not defined.**](#_Toc181282330)

[3 Main Part Error! Bookmark not defined.](#_Toc181282331)

[3.1 Structure **Error! Bookmark not defined.**](#_Toc181282332)

[3.2 Text Layout **Error! Bookmark not defined.**](#_Toc181282333)

[3.2.1 Formatting **Error! Bookmark not defined.**](#_Toc181282334)

[3.2.2 Headings **Error! Bookmark not defined.**](#_Toc181282335)

[3.2.3 Hyphenation and Dashes **Error! Bookmark not defined.**](#_Toc181282336)

[3.2.4 Page Breaks **Error! Bookmark not defined.**](#_Toc181282337)

[3.2.5 Bullet Points **Error! Bookmark not defined.**](#_Toc181282338)

[3.3 Figures **Error! Bookmark not defined.**](#_Toc181282339)

[3.4 Tables **Error! Bookmark not defined.**](#_Toc181282340)

[3.5 Formulas **Error! Bookmark not defined.**](#_Toc181282341)

[Bibliography Error! Bookmark not defined.](#_Toc181282342)

[Appendix Error! Bookmark not defined.](#_Toc181282343)

List of Figures

[Figure 1: Donkey Rocket-Early stage **Error! Bookmark not defined.**](#_Toc181282344)

[Figure 2: Upper Column part of a rectification column with feed-in point for entrainer **Error! Bookmark not defined.**](#_Toc181282345)

[Figure 3: Example Excel-graphic; Fermentation process of sweet sorghum pressed juice in a 25-liter batch at 35 °C **Error! Bookmark not defined.**](#_Toc181282346)

[Figure 4: Setting the style sheets for the formula editor **Error! Bookmark not defined.**](#_Toc181282347)

List of Tables

[Table 1: Example table with particularly long labeling to demonstrate the settings for the table labeling **Error! Bookmark not defined.**](#_Toc181282348)

[Table 2: Styles, font sizes and spacing **Error! Bookmark not defined.**](#_Toc181282349)

[Table 3: Formatting without style sheets **Error! Bookmark not defined.**](#_Toc181282350)

[Table 4: Indents and tab stops **Error! Bookmark not defined.**](#_Toc181282351)

Nomenclature

**Symbol Description Unit**

**Latin symbols**** low boiler *(-)*

medium boiler *(-)*concentration *(v/v)*

 high boiler *(-)* Destillate (*kmol)*

 Destillate stream (*kmol/s)*

 Entrainer *(-)*

 Henry-Coefficient (*bar)* specific Enthalpy of vaporization (*kJ/kg)*  
Capacity *(-)*

**Greek Symbols** Initial State *(-)*

 Activity Coefficient *(-)* Density (*kg/m³)*

 Final State *(-)*

**Indices**

** Component a

** Component b

** Destillate

** Entrainer

**Abbreviations**

** Central Processing Unit

** Ethanol

** Oxygen Transfer Rate

For a standardized representation of the formula symbols, indices and abbreviations, it is recommended to edit all ‘symbols’ with the formula editor. Units should be given in round brackets

# Introduction

Climate change has emerged as one of the most pressing global challenges of the 20th century. Fossil fuels, including coal, oil, and natural gas, have long served as the primary energy sources, accounting for nearly 80% of global energy consumption. However, their extensive use has led to severe environmental consequences, including greenhouse gas emissions, pollution, and resource depletion. Additionally, the increasing energy demand from industries is accelerating the depletion of fossil fuel reserves. In response to these challenges, renewable energy sources are gaining prominence, with biogas playing a crucial role in reducing carbon dioxide emissions and promoting sustainable energy generation [[1]](Biofuels#_CTVL001b4915592c58f416aa0730e5a880c7fcc).

Biogas is produced through the anaerobic decomposition of organic materials by microorganisms. Common feedstocks for biogas production include agricultural manure, sewage, municipal waste, green waste, plant material, and other biodegradable substances. Renewable energy sources such as solar, wind, hydroelectric, biomass, and geothermal biogas hold a unique position due to their ability to provide a clean energy alternative while simultaneously offering benefits such as waste management and the production of high-quality organic fertilizer [[2]](Biogas:#_CTVL001778cdbe56915452c863286eaa9abe14d).

A biogas plant is a complex system comprising multiple components designed to efficiently convert biomass into biogas. Key elements include substrate processing, fermentation, gas storage and treatment, digestate storage, and power generation. The process begins with substrate processing, where organic waste such as animal manure, agricultural residues, and silage are prepared to maximize gas yield and prevent operational issues. Fermentation, the core stage of biogas production, occurs in an oxygen-free (anaerobic) environment, where microorganisms break down organic materials, ultimately producing methane-rich biogas. Once produced, the biogas undergoes storage and purification, followed by combustion in combined heat and power (CHP) plants to generate electricity and thermal energy [[3]](#_CTVL0012d10b859aee04bff9ab73cbcc21b7727).

Amid rising electricity and gas prices and the environmental impact of fossil fuels, the need for cost-effective and scalable biogas solutions has become increasingly important. Ensuring the economic viability of biogas plants requires the development of new, sustainable business models that enhance energy production and allow for flexible biogas generation. One key aspect of optimizing biogas production is the implementation of mathematical biogas plant models and adaptive feedback control systems [[4]](Real-world#_CTVL001e35bed4e8c2c40818bf3e931ac3ca5ae).

To achieve this, a MATLAB based graphical user interface (GUI) has been developed to facilitate the monitoring and optimization of biogas production. However, MATLAB is a proprietary software with high licensing costs, making it less accessible for small-scale plant owners and researchers [[5]](#_CTVL0011c07bec0aeb846248fb6e4098652107d). This limitation necessitates the transition to an open-source, cost-effective alternative.

This study focuses on the design and development of a Python-based GUI that replicates the functionality of the existing MATLAB interface while ensuring affordability and ease of use. By leveraging open-source tools and programming frameworks, the proposed Python system aims to provide efficient, scalable, easy-to-maintain multiple versions of shared libraries, more compact and readable than MATLAB code, and a user-friendly platform for biogas plant operators [[5]](#_CTVL0011c07bec0aeb846248fb6e4098652107d). The transition from MATLAB to python reduces software costs and increases accessibility, enabling more widespread adoption of biogas optimization technology.

Through this research, a case study on biogas production management will be conducted to validate the effectiveness of the newly developed Python-based GUI. The study will assess its ability to streamline operations, enhance control mechanisms, and maximize biogas yield while maintaining reliability and usability for plant operators.

Research Question:

* What are the computational challenges and algorithmic modifications required when effectively replacing a MATLAB-based feedback control system for flexible biogas production with a Python-based platform to enhance accessibility, cost-effectiveness, and user experience while maintaining system accuracy and control efficiency?

# State Of The Art

Biogas is produced through the anaerobic digestion of biodegradable organic materials such as animal manure, agricultural waste, sewage sludge, and municipal organic waste [[6]](Recent#_CTVL001535b8c6472a34f0793588f3a9ae7e194). This process involves microbial decomposition in an oxygen-free environment, resulting in the production of methane (CH₄) and carbon dioxide (CO₂). Anaerobic digestion is a complex biological process that requires stable conditions, including temperature control and substrate balance. Due to fluctuations in substrate composition, microbial activity, and environmental conditions, the manual operation of biogas plants often leads to inefficiencies in gas production [[7]](Anaerobic#_CTVL0013cec95b721e64e73b2719a47ba753264).

To improve efficiency, system modeling and simulation techniques are used to predict biogas yields and optimize plant operation. Different levels of model approaches to biogas production are categorized as White-box, Grey-box, and Black box or bottom-up and top-down approaches [[8]](modeling-approaches-of-biogas-production-3,#_CTVL001d1dddd2f472c4642be3a78105134725c). These biogas models can provide dynamic information on the anaerobic digestion process, which can be used to forecast biogas yields or optimize the process.

White-box models are physics-based models that describe systems using mathematical equations based on fundamental laws, such as mass balance and reaction kinetics. These models are categorized as linear or nonlinear differential equations that predict system behaviour deterministically [[9]](Model#_CTVL001ae439aacb330454f9cd8dc9d1ffe1daf).

White-box models are difficult to develop and implement because of their complexity, which necessitates the design and implementation of all influencing parameters [[10]](Biogas#_CTVL00169895cb17ee542ceb2ad76dececf58de). The widely used White-box model is Anaerobic Digestion Model No.1(ADM1). ADM1 includes biochemical reactions and physicochemical reactions to simulate biogas production[[10]](Biogas#_CTVL00169895cb17ee542ceb2ad76dececf58de). ADM1 is effective in identifying process inhibitors such as ammonia toxicity and pH imbalances. However, it was originally developed for wastewater treatment plants, and its direct application to agricultural biogas plants is limited due to variations in substrate composition. Agricultural substrates contain a mix of proteins, fats, fibres, and carbohydrates, which require model modifications for accurate biogas yield predictions[[11]](Application#_CTVL0015010092eb0784ea8ad98d5ae1e120e09).

Moreover, an essential part of the model validation is the analysis of the residual errors, which refer to the difference between the actual and predicted value in a statistical or machine learning model. The white-box model often struggles to produce completely random errors (uncorrelated); instead, its error shows a pattern. This happens because, while White-Box models are designed for accuracy, they tend to be either too complex or too restrictive in their assumptions. Even when they work well on paper, they often miss important real-world factors like how well a digester is mixed, which plays a big role in biogas production [[4]](Real-world#_CTVL001e35bed4e8c2c40818bf3e931ac3ca5ae). Hence, white-box models require more advanced differential equations, such as stochastic differential equations [[9]](Model#_CTVL001ae439aacb330454f9cd8dc9d1ffe1daf). to account for the natural uncertainties in the system.

Black-box models, like machine learning and deep learning, can work with complex and nonlinear data patterns. Black-box models use statistical methods, experiments, or real-world measurements to find relationships between input and output variables. While Black-box models offer high adaptability and predictive power, they also have certain limitations[[12]](Comparison#_CTVL0018df6a57258d84ec2878d224db31b6d28), Like they requires large amount of training data, lack interpretability, making its difficult to understand why certain predictions are made and they fails in unseen scenarios [[12]](Comparison#_CTVL0018df6a57258d84ec2878d224db31b6d28).

Grey-Box model is a hybrid model, it is the combined form of both White-box and Black-box models. Grey-box have both physical and empirical significance [[8]](modeling-approaches-of-biogas-production-3,#_CTVL001d1dddd2f472c4642be3a78105134725c). Grey-box has the capability to handle complex systems more efficiently than the white-box and black-box models. It requires less data and have improved generalization and adaptability. This model is attempted, the algorithms were only evaluated on a simulation. Furthermore, the idea has only been investigated for one substrate, but the effect of various substrate interactions on the outcomes is still unknown [[4]](Real-world#_CTVL001e35bed4e8c2c40818bf3e931ac3ca5ae).

Once a system model is in place, the next crucial step is designing a feedback control mechanism to ensure smooth and efficient operation. Traditional controllers, like Proportional-Integral (PI) and Proportional-Integral-Derivative (PID) controllers, are widely used because they are both simple and effective at maintaining stable conditions. These controllers work particularly well in systems with minimal disturbances and steady setpoints, so they are popular in industrial settings such as biogas plants where reliability and consistency are key [[13]](Flexible#_CTVL00118d4364fb5294b90ab338208a99d4701).

Traditional controllers might not work well in more complicated, multivariable systems where constraints require careful management and setpoints are subject to frequent changes. Model Predictive Control (MPC) and other sophisticated control techniques are useful in this situation. Utilizing a dynamic model, MPC forecasts future system behavior and modifies control actions accordingly. MPC is particularly useful in settings requiring frequent setpoint changes and stringent constraint management because it can manage several variables and constraints simultaneously. In demanding applications, MPC clearly outperforms conventional PI controllers thanks to its predictive capability.

Because of variations in substrate inputs, gas production rates, and environmental factors, operating conditions in biogas plants are rarely constant. PI controllers may find it difficult to adjust to these fluctuations, even though they can efficiently maintain steady-state operations. Model Predictive Control (MPC) is helpful in this situation. Through constant real-time data analysis and predictive adjustments, MPC enhances system performance more dynamically. Nevertheless, there are difficulties in putting MPC into practice despite its benefits. It is more difficult to set up than traditional controllers because it needs a well-calibrated system model and more processing power [[14]](Adaptive#_CTVL00101b7f09945ee41738833d576f3404fb7).

Therefore the choice between MPC and traditional PI control ultimately depends on the complexity of the system, its performance needs, and how practical it is to implement. While PI controllers work well for maintaining stability, MPC offers greater flexibility in dynamic environments. To further improve prediction accuracy and adapt more effectively to changing substrate compositions, future research should explore integrating grey-box modeling with MPC. Additionally, real-world testing is essential going beyond simulations will help verify the reliability and practical benefits of these advanced control strategies in biogas production systems [[15]](Black-,#_CTVL0015da80d3e2222414e8321a3702f359ea8).

A feedback control system is under development especially for the upward model in the Python GUI implementation. The controller design seeks to orient all closed loop poles at the origin. But given a second-order difference equation, the upward model shows more complexity than the descending one. This more complexity makes the root locus approach less applicable.

While creating a feedback control system with the root locus approach is still possible, even with a PID controller relocating all closed loop poles to the origin proves impossible. As such, the upward model will be implemented using a state space controller. The major benefit of the state space approach is that it lets direct manipulation of the internal state variables of the system rather than only considers input-output interactions. This capacity is especially important in resolving the complicated dynamics inherent in biogas generation as production rises [[16]](Dochain#_CTVL001c76bdfacdf8742038f823d468a2475e0).

State space controllers provide significant benefits for the management of intricate biogas systems, especially for upward models that require enhanced output to satisfy fluctuating energy demands. Unlike traditional PI controllers, state space controllers utilise a mathematical representation of the system through state, input, and output variables in matrix form, facilitating more accurate modelling of multivariable dynamics [[17]](Modern#_CTVL001c11784a67fb94496ac7878754bde6a2f). In biogas applications, state space controllers exhibit higher efficacy in managing the nonlinear traits of the anaerobic digestion process, yielding enhanced response times and superior setpoint tracking compared to conventional control methods [[18]](State#_CTVL0014a2419ab53884e5c81846967e4e7c440). State space controllers enable the positioning of all eigenvalues of the closed-loop system at specified locations, hence facilitating optimal dynamic performance using pole placement methods. This technique has been empirically proven in biogas applications, demonstrating considerable reductions in settling time compared to PID controllers when adjusting to changes in substrate composition [[19]](Dynamical#_CTVL0012fd6627c02fd4417bb0f5c086051b376).

To analyze the optimal model by their validity and predictive capacity, several statistical metrics has been used. The Akaike Information Criterion (AIC) functions as a primary metric for model selection in this work, providing a fair evaluation of model fit while imposing penalties for excessive complexity. Akaike defines AIC as , where k denotes the number of parameters and L signifies the model's greatest probability. This formulation offers a quantitative assessment that incentivises goodness of fit while deterring overfitting. AIC has proven to be superior in selecting parsimonious models for biogas production modelling, effectively capturing complicated biological processes without excessive parameterisation [[20]](Multimodel#_CTVL0019a2d83223ab94cd6b800e0d629908b3b).

The coefficient of determination (R²) is commonly employed in model evaluation, although its use in nonlinear systems necessitates cautious interpretation. R² numbers may be deceptive in nonlinear scenarios, such as biogas production modelling [[21]](An#_CTVL001099ce34d0ece44b2903c3a908e054856). Their research underscored cases where models with elevated R² values yielded inferior predictive performance compared to those with diminished values, advising caution about excessive dependence on this statistic alone. In biogas process modelling, R² should be regarded as a supplementary metric rather than a conclusive selection criterion.

The Sum of Squares Error (SSE) provides an additional helpful perspective by quantifying the overall divergence between observed and model-predicted values. SSE alone fails to consider model complexity, potentially promoting overparameterized models that excel on training data yet generalise inadequately [[22]](Black-box#_CTVL0011975434d2eba4f478cbdc5048fe4a17e). In biogas production, where process variability are prevalent, the Mean Absolute Error (MAE) serves as a more intuitive and resilient metric of model accuracy, exhibiting reduced sensitivity to outliers compared to the Sum of Squared Errors (SSE) [[23]](Root#_CTVL00189d7fb21df7e4a089a8204d73992627b).

The ultimate model selection procedure must incorporate various factors, acknowledging that no singular indicator encompasses all dimensions of model performance. Integrating information-theoretic methodologies (e.g., AIC) with empirical error measurements (such as MAE) and validation strategies establishes a robust framework for model selection [[24]](Information#_CTVL001a27b4fed9fce49d7b16978af32545a70). This balanced methodology is especially appropriate for biogas production modelling, where both theoretical accuracy and practical usability are critical factors.

A Matlab-based GUI (Graphical user interface) was built to achieve the optimization of biogas production through a feedback control system. To improve operational efficiency, this interactive tool integrates control algorithms and model estimation techniques to enable plant operators to visualize and manage biogas production in real time. The system enables users to input different substrate feeding plans and analyze their effect on the amount of biogas produced, and by observing the results, operators can make data-driven decisions to maximize biogas production efficiency. The GUI runs simulations; through simulation, it fine-tunes the control system (state space controller) before real-world application and ensures stable and efficient performance. However, due to the high licensing costs of Matlab, this study focuses on converting the Matlab GUI to Python, which is an open-source alternative that offers accessibility and affordability for large-scale and small-scale biogas production businesses.

## Matlab

MATLAB has become a very popular tool in the research community because of its versatility, powerful analytical capabilities, algorithm development, data visualization, and interoperability [[25]](Unlocking#_CTVL001b8d239d476bd48458dbd9c71467ba9f7). MATLAB is one of the most popular Computer Algebra System programs used in problem-solving, computation, and visualization of mathematics through mathematical exploration, etc. MATLAB consists of a wide range of specialized toolboxes. Numerical, symbolic, and state-of-the-art graphic visualization capabilities are integrated with an extremely user-friendly computer program environment [[26]](MATLAB#_CTVL001d6a24f304a5940afb082cf2201c27bce). The toolbox aims to treat optimization, control, and simulation algorithms for biogas plants separately. The toolbox makes it simple to apply and modify different optimization algorithms by using a generic optimization scheme. Although a wide variety of optimization algorithms are available, this list can be readily expanded with the help of a clear interface [[27]](MATLAB#_CTVL0018f52302af1ca44d38c14b5b33bbfd560).

These are some of the optimization techniques.

* 1. Genetic Algorithm and Direct Search Toolbox (The Genetic Algorithm and Direct Search Toolbox is a collection of functions that are used to extend the capabilities of the optimization toolbox and the numeric computing environment [[28]](Genetic#_CTVL0013b27a191ed1741929a08f4506aee083e).)
  2. PSO (Particle Swarm Optimization) Algorithm. (It is a population-based algorithm similar to the genetic algorithm and the most popular natural inspired metaheuristic optimization [[29]](Particle#_CTVL001d6812a06b4624a25be06ad3206aa91c2).)
  3. Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (CMA-ES is the most successful and influential evolutionary algorithm designed for continuous optimization, optimizing real-valued vectors while adapting covariance matrices to improve efficiency [[30]](Covariance#_CTVL001d70e38978fbb4b3d9d57d518ef48ee76).)
  4. Differential Evolution(It is a simple yet efficient heuristic optimization algorithm designed for global optimization in continuous spaces, particularly useful for handling nonlinear and non-differentiable functions [[31]](Differential#_CTVL001d510ad9fd24442dc964bb7760b27a507).)
  5. Improved Evolution Strategy using Stochastic Ranking (The Improved Evolution Strategy using Stochastic Ranking introduces an approximation method to reduce function evaluations in nonlinear optimization by employing surrogate models for both the objective function and constraint violations, updating them sequentially to enhance efficiency [[32]](Approximate#_CTVL001012f3c7890d64517b7de85dd775f402e)​.)

Moreover, MATLAB has a huge number of functions, and while Simulink remains unmatched with no true alternative, it is much easier for beginners.

Despite its powerful capabilities and a huge range of Toolboxes, MATLAB does have many limitations.

1. One major drawback of MATLAB is that its algorithms are proprietary, meaning users cannot access the underlying code of most built-in functions. This requires a level of trust that the implementations are correct, without the ability to verify or modify them if needed.
2. Another challenge is the cost. MATLAB is super expensive, which makes it inaccessible to many individuals and small organizations.
3. MATLAB imposes restrictions on code portability. Running your code on another computer isn’t always straightforward. While you can use the MATLAB Component Runtime (MCR) to distribute compiled applications, the app must match the exact version of the installed MCR. Given that MATLAB releases a new version every six months, this can quickly become inconvenient.
4. Additionally, third-party developers are unable to expand MATLAB's functionality due to its proprietary nature.

## Python

MATLAB's restrictions lead Python, with its open-source character, platform independence, and straightforward syntax, to become a strong substitute. It is easily readable and modifiable, which qualifies both for researchers and developers. Python is a great choice for computational chores because of its dynamic typing and automatic memory management, as well as MATLAB's resemblance. Python is a flexible tool for flexible biogas production applications since it lets efficient data processing and GUI development possible with a rich ecosystem of libraries including NumPy, Pandas, and SciPy [[33]](#_CTVL0015190933aff7f46b791d409947b817ef6).

Python is designed for general-purpose, high-level programming; Python supports procedural, object-oriented, and functional programming, among other paradigms. Guido van Rossum invented Python, which initially came out in 1991. It finds extensive application in many fields, including data science, software development, scientific and numerical computing, and web development [[34]](A#_CTVL001e94745b58e644bc0824431af6772a241).

### Features of Python

* **Easy to use**: Python's simple, developer-friendly syntax makes learning and quick code writing quick and easy.
* Free and open source: Python is free to use and has a sizable community helping to shape since it is open source.
* GUI support: Python provides TkInter, PyQt, and PySimpleGUI, among several tools for creating graphical user interfaces (GUIs).
* Portability:Python code can run on various platforms, including Linux, macOS, and Windows, without any changes or modifications.
* Dynamically Typed: Python autonomously identifies variable data types, minimizing the necessity for explicit declarations.
* Interpreted: Python code is executed sequentially, facilitating debugging and enhancing accessibility for novices.
* Language Interoperability: Python can interface with code authored in other languages such as C, C++, and Java, facilitating versatile application development.
* Object Oriented: Python facilitates object-oriented programming, allowing for the development of classes and objects to represent real-world entities and their interrelations.

The conversion of a MATLAB GUI to a Python-based interface for adaptable biogas production identifies Tkinter as the ideal tool for constructing the graphical user interface due to its simplicity and cross-platform compatibility. Tkinter, sometimes known as "Tk interface," is a module in Python developed in TCL (Tool Command Language) that offers an interface to the TK GUI toolkit, multiplatform supporting Linux, macOS, and Windows. Including three main ideas: widgets, geometry management, and event handling, it is used to produce graphical user interfaces (GUIs). Widgets in a GUI are its outward elements frames, labels, buttons, and text entries. Usually applying the "grid" approach, geometry management arranges these widgets on the screen. Tkinter's event handling controls the event loop that manages user actions including button presses and keystrokes, so enabling widgets to react to these events via callbacks or event bindings. Tkinter's simplicity and availability of many tutorials and code examples help to explain it rather widely [[35]](11th#_CTVL00175be5d9c9e944e3cb754ec65afb52850).

# Materials

The Grub biogas facility functions as the principal experimental location for this study, supplying critical operational data for model validation and system evaluation. This agricultural biogas facility in Bavaria, Germany, is among the region's most technologically sophisticated installations, comprising a 75 kW gas engine combined heat and power (CHP) unit, a 1206 m³ concrete-covered digester, and a fermentation residue storage capacity of 2714 m³. The facility functions with semi-automated control via Siemens Programmable Logic Controllers (PLC), which oversee and document essential characteristics such as temperature, pressure, and gas flow rates. The plant employs various organic resources for substrate feeding, predominantly including slurry, corn, cattle and sheep dung, grass silage, and fodder leftovers in different ratios. The operational data gathered from this plant encompasses several periods, documenting both steady-state conditions and dynamic reactions to feeding modifications, thereby offering a thorough dataset for the development and validation of mathematical models of the anaerobic digestion process. This empirical data is especially significant as it illustrates the practical difficulties encountered in commercial biogas production, encompassing fluctuations in substrate quality, environmental factors, and operational limitations.

# Methodology

The initial Matlab GUI for adaptable biogas production was developed as a specialized scientific instrument aimed at system model estimate and feedback control for demand-driven biogas generation. The interface was systematically arranged into a sequential workflow that directed users through data preprocessing, model estimates, and control system design. The GUI predominantly focused on two estimating techniques - PT1 approximation and the Time percentage method - hence constraining users' analytical ability in evaluating various modeling strategies.

The interface presented visualisations of biogas production rates and substrate feeding rates, incorporating filtering functionalities to diminish measurement noise using discrete low-pass and smoothing filters. Principal visualisations encompassed time-series graphs for step responses, filtered measurement data, and model estimations, facilitating users in discerning the correlation between substrate feeding and resultant biogas generation. The system enabled users to enter substrate combination data (comprising maize, manure, grass silage and fodder wastes) and process flow rates recorded by the biogas plant's PLC system.

Notwithstanding its functionality, the Matlab implementation exhibited several significant limitations: its constrained modelling approach provided only two estimation methods instead of the comprehensive six methods (PT1-approximation, time percentage method, turning tangent method, sum of time constants method, PT1-estimator, and PT2-estimator) delineated in the research. This limitation diminished users' capacity to ascertain the most suitable model for certain biogas plant settings. The system was hindered by an inflexible workflow structure that complicated comparison analysis, while reliance on Matlab licenses imposed financial constraints on implementation at smaller biogas facilities. The interface was deficient in contemporary user experience elements, providing minimal customisation possibilities and necessitating extensive understanding of control theory for efficient functionality.

## Approach to Conversion

Using a mixed approach direct translating fundamental algorithms with strategic redesign of the interface and implementation structure the Matlab GUI for flexible biogas generation was converted to Python. Although the mathematical underpinnings and data analysis techniques from the original Matlab implementation were maintained, the Python version was developed with improved user interface and expanded capability.

Starting with a careful study of the original Matlab code to pinpoint important components data loading, preprocessing filters, model estimation techniques, and control system computations the conversion process started While the implementation strategy used Python's strengths in modern GUI development and data analysis, these fundamental mathematics and engineering ideas were directly translated to preserve scientific integrity.

The main GUI framework used in the Python implementation is Tkinter, augmented with specific widgets such tkcalendar's DateEntry for enhanced date selection and tkt.notebook for tabbed navigation. The code for scientific computing depends on accepted Python libraries that parallel Matlab's capability, notably NumPy for numerical operations, pandas for data manipulation, SciPy for signal processing (especially the Butterworth filter implementation), and Matplotlib for visualisation. While leveraging Python's more easily available programming paradigm, this mix offers similar analytical power.

Expanding the model estimation capabilities from the two methods (PT1-approximation and Time percentage method) from the original Matlab GUI to include all six estimation approaches described in the research: PT1-estimator, PT2-estimator, Turning tangent method, Sum of time constants method, PT1-estimator, and PT2-estimator was a major redesign aspect. This improvement lets users concurrently evaluate several modelling techniques, therefore offering a more complete analysis tool for the optimisation of biogas plants.

Designed with enhanced user experience in mind, the interface was rebuilt using a tabbed interface guiding users progressively from data loading to preprocessing, model estimation, control system design, and generating of feeding schedules. Beyond what was feasible in the initial Matlab implementation, interactive elements such as sliders for data selection and responsive visualisations offer instantaneous feedback, hence improving the exploratory analytical capacities.

Careful attention was made to guarantee computational correctness while enhancing usability throughout the conversion process, hence producing a more accessible but similarly strong tool for flexible biogas production modeling and control.

## Load Data Tab

### Overview

Effective loading and visualisation of biogas production and substrate-feeding data from CSV files is made possible by the "Load Data" tab found inside the Python GUI. This procedure guarantees that the data is organised and available for next preparation and analysis, therefore complementing approaches developed in pertinent research.

**4.2.2 Data Acquisition Process**

The system focuses on acquiring two primary datasets:

* Biogas Production Rate Data: Extracted from "Gas flow rate Fermenter - current.csv".
* Substrate Feeding Rate Data: Extracted from "Substrate feeding Fermenter - today.csv".

Python's pandas package automatically searches and reads from the specified directory these files from. To allow correct time-series analysis, the timestamps are transformed to date time format. Concurrent concatenation of the data frames guarantees data continuity.

**4.2.3 Visualization of Step Graphs**

The GUI supports visualizing step-down and step-up behaviors in the biogas production process. The visualization approach includes:

* The plot\_step\_graph() function creates step graphs depicting biogas production and substrate feeding over time.
* Two graphs were plotted

1. Step Downwards Figure 2 (Examines decreasing production and feeding trends.)
2. Step Upwards Figure 3 (Examines increasing production and feeding trends.)

* Matplotlibpython library is utilized for plotting, with the primary Y-axis (a blue line) representing the biogas production rate and the secondary Y-axis (a red line) depicting the substrate feeding rate. The x-axis is set up to show hourly timestamps; the last tick mark shows the month and year.

A graph with blue and red lines

AI-generated content may be incorrect.

Figure 2: Step downward plot of biogas production rate and substrate feeding before preprocessing

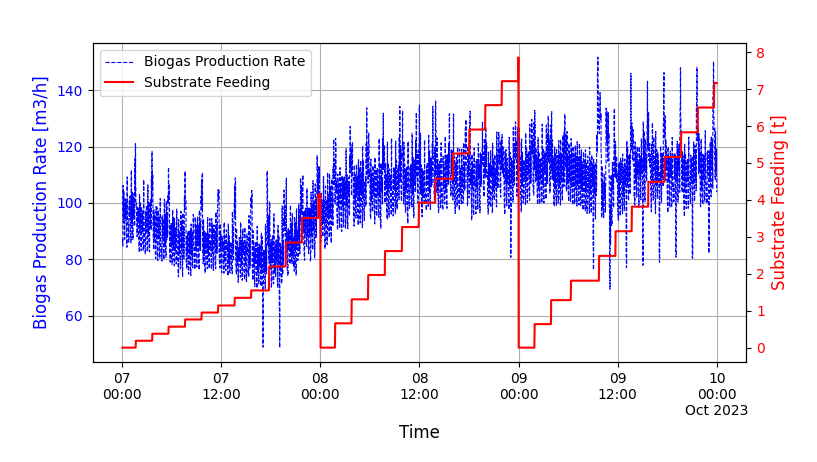


Figure 3: step upward plot of biogas production rate and substrate feeding before preprocessing

The "Load Data" tab offers an easy-to-use visualising tool for important step changes in biogas production and substrate feeding and streamlines data collecting. This method guarantees that the data is ready for preprocessing and next control system analysis, therefore supporting the general aim of improving system modelling and correctness.

## Preprocessing Tab

### Preprocessing of Biogas Production Data

Preprocessing is a key step in ensuring the accuracy and reliability of biogas production data. The fundamental goal is to reduce noise and improve data clarity so that subsequent analysis and management strategies may be built on clean, consistent datasets. Two main phases define the preprocessing framework:

* Low-pass filtering to eliminate high-frequency noise and periodic peaks.
* Smoothing using a moving average technique to stabilize the data trend.

These approaches fit accepted research techniques and guarantee that the control system runs on accurate and improved data inputs.

Both biogas production rate and substrate feeding rate data received the same preprocessing methods, which included Butterworth low-pass filtering and moving average smoothing. These methods' settings were constantly modified in response to the individual data properties.

#### Low Pass Filtering and Moving Average Smoothing

A fourth-order Butterworth low-pass filter was engineered to address the periodic peaks in biogas production data, which typically arise every 2–3 hours.

The cutoff frequency was determined according to the established periodicity:

This frequency effectively focusses and attenuates periodic peaks, allowing only meaningful and stable trends to pass through the filter. This stage is critical for maintaining data integrity and the control system's reliability.

Following low-pass filtering, the data was flattened, and remaining noise was further reduced by a moving average filter. This approach eliminates tiny swings and stabilises the data trend, resulting in increased clarity. Smoothing is applied with a window size of 5, ensuring that local variances are reduced but not distorting the general trend. This step improves the clarity of the biogas production rate by removing transient variations.

#### Extraction of Feeding Rate

The extraction feeding rate uses two steps to refine data. Using a predefined smoothing window, first a uniform moving average smoothing technique is applied to the low-pass filtered data ensuring the retention of only positive derivatives since daily reset at midnight would otherwise produce negative values that do not reflect actual feeding rate changes. A post-processing stage then follows to preserve data uniformity since differences in real-world data can magnify noise levels. This entails a threshold-based correcting mechanism (with a threshold value of 0.1) that detects and smooths out considered unnatural rapid decreases in feeding rate. The method iteratively goes over the substrate segment data, comparing adjacent values to ascertain whether they satisfy the threshold; when such cases are found, it substitutes the previous value for the current one, so preserving only real changes in feeding rate in the final dataset.

### Visualization

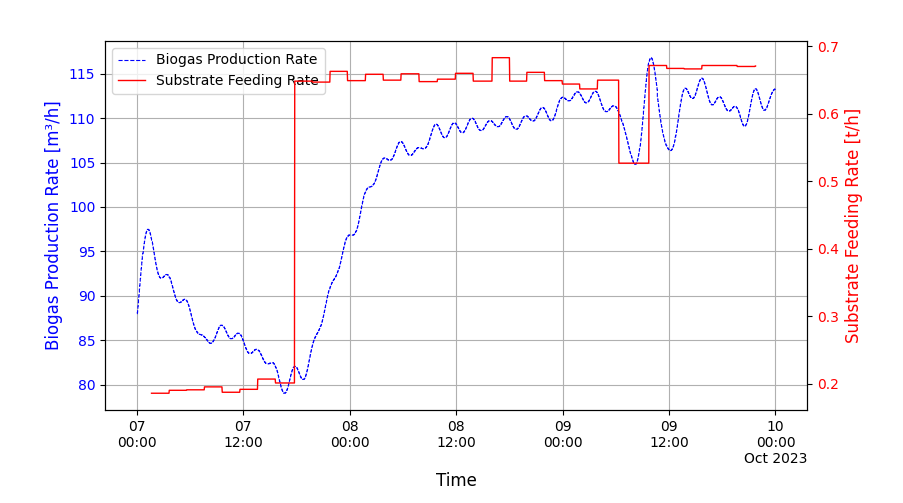


Figure 4: Biogas production rate and substrate feeding rate after preprocessing

The plot Figure 4 shows the biogas production rate and substrate feeding rate after preprocessing. The rate of substrate feeding is represented by a solid red line, while the blue dashed line represents the response of biogas production to a sudden change in feeding rate. Although the data still contains a small amount of noise, it is insignificant and has no discernible impact on the measurement.

### GUI Interaction and Control

The Python GUI contains interactive components intended to ease and control the preprocessing steps:

• The Preprocessing Button allows users to apply filtering and smoothing to specified data segments.

• Use sliders to dynamically choose data segments for preprocessing, assuring focussed results.

## Model Estimation Tab

The Model estimation tab uses preprocessed data to try to streamline the analysis and choice of gas production flow rate models. There are two plots for the visualization of the upward and downward steps models. Every part of the models includes graphical representations, correlation measures, and similarity tables in order to evaluate their performances. Six estimating models were implemented: Time Percentage, Pt1\_Model, Time Constant Sum, Turning Tangent, Pt1 Estimator, and Pt2 Estimator. The system lets the plant operator select which model to run using a radio button; once a model is chosen, it generates the required changes to the tabular and visual representations. For comparative study, Matplotlib shows similarity metrics in tabular form and facilitates data visualization.

### PT1 - Approximation

The PT1-approximation involves employing a first-order differential equation to the prediction of the trajectory of the biogas production rate equ. (4.1)

|  |  |
| --- | --- |
|  | 4.1 |

In this equation, signifies the time constant of the system, and indicates the system gain. The system gain is determined by the ratio of the change in output to the change in input, expressed as

|  |  |
| --- | --- |
|  | 4.2 |

The variation in biogas production rate before and following the step process is denoted as , and the substrate feeding rate is later referred to as . For a step response, where, we have . Given that and assuming an initial value it follows that . The time constant

Where is the starting time of the step process. In this case, the is determined by identifying the point at which the output reaches 63% of its final value. This equation encapsulates the essential dynamic behaviour of the system, highlighting its response time and steady-state characteristics.

An important feature of PT1 systems is their typical response to step inputs, where the output attains about 63% of its final value after one time constant. This percentage is not randomly selected; it arises from the mathematical resolution of the PT1 differential equation.

Assuming that the input variable u(t) is a unit step function, the solution will be:

Due to the unit step function where , we have . Since , consequently . The previous equation may then be rewritten as:

For

we obtain:

which approximates to about 63% of the final value . This property offers an effective approach for determining the system's time constant using experimental data.

Once the fundamental parameters have been established, the subsequent step is to develop the system's transfer function. At the outset, the calculated K and T values are employed to construct a continuous transfer function. The function is subsequently converted to a discrete form to facilitate digital implementation, with a sampling time of 120 seconds. The dynamic nature of the system is preserved while the model is effectively utilized with discrete measurement data through the implementation of the conversion process through signal.cont2discrete.

The continuous transfer function was implemented in Python using the control systems representation format, where num = [pt1['K']] indicates the numerator coefficient K and den = [pt1['T'], 1] represents the denominator polynomial . This method directly converts the mathematical representation of the first-order system to a programmable format.

The method uses the SciPy Signal package to convert discrete transfer functions. Specifically, signal.cont2discrete((num, den), pt1['Ta'], method='zoh') converts the continuous transfer function to its discrete equivalent using the zero-order hold (ZOH) method and a sampling period. This transformation is consistent with the method outlined in the algorithm documentation for converting to .

The final stage is to simulate the system response using signals.dlsim((num\_d, den\_d, pt1['Ta']), t\_input, t), which processes the input signal using the discrete transfer function to get the output response. This applies the difference equation (4.3), completing the transfer from the s-Laplace domain to the discrete-time domain.

|  |  |
| --- | --- |
|  | 4.3 |

The implementation identifies and manages both step-up and step-down responses within the system. This adaptability is essential as actual biogas production systems may encounter fluctuations in substrate feeding rates. The code modifies its calculations according to the direction of the step change, guaranteeing precise modelling irrespective of whether the system is increasing or decreasing.

Through the update\_model\_down\_plot\_pt1 (Figure5) and update\_model\_up\_plot\_pt1 (Figure 6) functions, the implementation includes thorough visualization capabilities. These features produce thorough plots displaying the actual biogas production rate, model predictions, and substrate feeding rate. Easy visual assessment of model performance is made possible by the various line styles and colors used in the plots to distinctly separate measured data from model predictions.

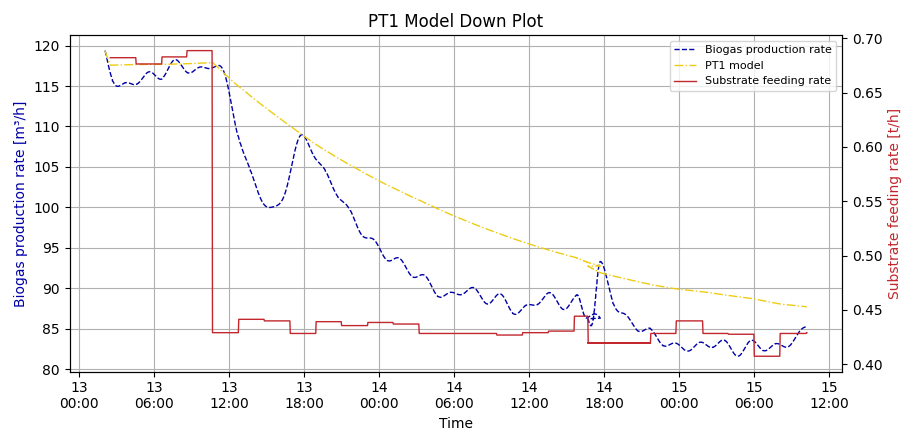


Figure 5: Step downwards visualization of the PT1-approximation method

A graph with different colored lines

AI-generated content may be incorrect.

Figure 6: Step upwards visualization of the PT1-approximation method

The successful implementation of the PT1 approximation is dependent on the fulfilment of certain system criteria. The system should mostly exhibit first-order behaviour, with step changes in input being precisely described. The system dynamics should be collected at an appropriate sample rate (120 seconds in our instance). Furthermore, the signal-to-noise ratio should be sufficient to make exact parameter estimates.

The PT1 approximation is a method that is both accessible and powerful for modelling biogas production systems. The robust modelling tool is the result of the successful integration of theoretical understanding with practical considerations. Although the method's mathematics may be intricate, the implementation is made comprehensible by the clear code structure and comprehensive error handling. The model's performance is easily validated and comprehended through the visual and numerical outputs, rendering it a valuable resource for system optimization and analysis.

### Time Percentage Method

The time percentage method was employed to model and examine the dynamic relationship between substrate feeding rate and biogas production in anaerobic digestion systems. This method utilizes a second-order differential equation approach, providing superior modeling flexibility relative to basic first-order approximations. The fundamental principle of the method is predicated on the subsequent differential equation:

|  |  |
| --- | --- |
|  | 4.4 |

Ta and Tb represent the time constants, K is the static gain, y(t) is the system output, and is the system input. The process starts with careful data preparation of the substrate feeding rate and biogas-generating rate signals. The step change index is found, and the first and final values are ascertained depending on the step direction; hence, minimum, and maximum values are set for inputs and outputs. Then, by computing the goal value, which represents 72% of the step change, characteristic points are found at which the output meets this value, that is, . The second characteristic time point . The is computed along with the ratio ​ where the is the output at the time . This ratio is used to look up the coefficients ​ and ​ from the reference table, enabling the calculation of time constants ​.

Creating the continuous transfer function components using Equ.(4.5), transforming to discrete form using a zero-order hold approach (Zoh), and extracting coefficients for the discrete transfer function in the form Equ. (4.6) is the essence of transfer function construction. Model validation is lastly calculated by simulating the system response with the found model and computing performance measures to evaluate model accuracy.

|  |  |
| --- | --- |
|  | 4.5 |
|  |  |

|  |  |
| --- | --- |
|  | 4.6 |

The implementation uses Numpy and Scipy libraries to handle numerical operations and transfer function manipulations. The gain calculation is performed using Equ. (4.2) Characteristic points are determined by calculating the goal value according to the step direction. At the same time, the time to achieve 72% of the change () is ascertained by index matching and timestamp conversion.

To calculate time constants, the algorithm uses the ratio of characteristic points from the step response. The normalised output value at the first characteristic point is the ratio of the difference from final value to the entire step magnitude. The normalised time constant row in Table 1 is indexed by this ratio. The program finds equivalent coefficients by searching the reference table for the closest match. After finding the matching row, multiplying the tabulated ratio values by the observed time yields the actual time constants and , scaling the normalised constants to the system's dynamics.

Table 1: Time constants for the time percentage method

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0.300 | 0.7950 | 0.000875 |
| …. | …. | …. |
| 0.1610 | 0.4056 | 0.3902 |

Transfer functions are formulated by integrating the distinct components of the second-order system. The components of the continuous transfer function for the PT2 element are mathematically integrated using convolution operations to achieve the comprehensive system representation. This continuous transfer function is then converted to its discrete-time equivalent using the zero-order hold (Zoh) method with a predefined sampling period. The discretised transfer function produces the required coefficients that define the system in the z-domain. The coefficients are extracted to create the difference equation, which establishes a recursive computational model appropriate for digital implementation and simulation of the system's dynamic behaviour.

Figure 7 and Figure 8 show the visualization of the time percentage method, illustrating the model's performance for upward and downward step responses respectively.

A graph with different colored lines

AI-generated content may be incorrect.

Figure 7 : Step downwards visualization of the Time Percentage method

A graph showing a number of time

AI-generated content may be incorrect.

Figure 8 : Step upwards visualization of the Time Percentage method

### Sum of Time Constants Method

The Sum of Time Constants Method is a graphical and computational technique employed to simulate higher-order dynamic systems by modeling them as a sequence of first-order systems. The method simplifies complex dynamic responses by summing individual time constants, making estimating and modelling system behavior easier.

The process starts by validating the data through the Validate inputs and then calculating the gain as the ratio between the output and input variables. The process calculates the system gain as the ratio between the output range () and input range () Equ. (**4.2**) It represents the steady state behavior of the system. The inflection point methodology is a key component of this identifying strategy. The method finds an inflection point (ind\_wp) in the step response curve and computes its slope (steigung\_wp).

The core of this identification technique is the inflection point methodology [[36]](Identification#_CTVL001b4e580aaed1b4b309c1ac253c050c437). The central point detection is implemented in the find\_inflection\_point, which functions using numerical gradient calculations. Using the first and second derivatives of the response curve allows the method to identify the inflection point whereby the second derivate approaches zero. By now, the tangent slope is extracted from the first derivative, allowing the construction of the linear approximation to find important temporal markers.

Based on the upward and downward step direction, the method finds the delay time () and the rise time () indices by identifying the tangent line intersecting the initial and final steady-state values. The precise time values are computed by converting time differences converted to seconds with the convert\_timedelta\_to\_seconds function to guarantee compatibility with many time formats.

The algorithm then finds a specific reference point at 57.5% of the overall step change (zt['wert\_fuer\_n\_2'] = 0.575) and calculates when the system reaction exceeds this threshold. The overall time constant sum () is determined by the time between step initiation and this crossing point.  
The system order () is estimated using the following formula:

The individual time constant () is then calculated by dividing the total time constant sum by the system order:

Using the Python control library, the method creates a transfer function model. It starts with a first-order transfer function with the calculated gain and time constant:

The method then adds (n-1) identical first-order systems in series to represent the complete nth-order system behavior:

Finally, the approach discretizes the transfer function with a sampling time of 120 seconds and simulates the system's reaction to the actual input signal to confirm the model fit. The simulation modifies the baseline based on the step direction (up or down) to achieve proper alignment with the original data.

This approach allows for accurate modelling of complex dynamic systems while also providing meaningful parameters (gain, time constants, and system order) that characterise the system's response properties, all of which are implemented with numpy for numerical operations and the control library for transfer function manipulation and simulation.

### Turning Tangent Method

Using inflection point analysis, the turning tangent method applied in the wendetangente function offers a comprehensive method of dynamic system identification. This approach uses the geometrical features of step response curves to create estimates of second-order transfer function for process dynamics, therefore offering a useful framework for control system design.

The process starts with the determination of the static gain parameter (), defined as the ratio between the output range ( and input range (). This proves, in steady-state settings, the basic amplitude relationship between system input and output. Create a temporal reference for the next computations; the program then finds the step start point, ind\_sprung.

Determining the inflection point (ind\_WP) and matching slope (steigung\_WP), which define the fastest rate of change in the step response, is fundamental to the turning tangent approach. The second derivative of the response curve equals zero at the inflection point, therefore signifying a concavity transition [[36]](Identification#_CTVL001b4e580aaed1b4b309c1ac253c050c437). By now, the method creates a tangent line with the equation:

where the offset is computed to guarantee that the tangent line intersects the inflection point. The tangent line illustrates the asymptotic behaviour of the system's rate of change and functions as an essential geometric component for deriving temporal parameters.

By finding where the tangent line crosses the first and final steady-state values, the technique finds two main time parameters. While the rising time () is found as the first point where the tangent goes below the final steady-state value (), the delay time (Tu) corresponds to the last point where the tangent surpasses the initial steady-state value (). For rising steps, the reverse reasoning holds. Time differential computations translate these temporal markers into seconds thereby guaranteeing conformity with technical units.

Based on theoretical connections detailed in the turning tangent approach generates three different time constant estimates [[37]](#_CTVL0017519eba8b47847498102588e2108b376):

* T\_Twp: Time constant derived from the inflection point time () with a 1:1 relationship
* T\_Tu: Time constant derived from the delay time () with a 1:0.282 relationship
* T\_Tg: Time constant derived from the rise time () with a 1:2.718 relationship

The analytical solution of second-order systems with equal time constants forms the basis of these connections, where the rising characteristics, inflection point, and delay follow particular mathematical correlations. The technique also averages these three estimations to increase robustness and determine a mean time constant (T\_mean).

The method generates a second-order transfer function with the same time constants in the form: for any time constant estimate:

where T stands for the appropriate time constant T\_Twp, T\_Tu, T\_Tg, or T\_mean). Common in process control applications, this form correlates with the process having two identical first-order elements in series. Using the zero-order hold (ZOH) approach with a defined sample time (), the technique converts continuous-time transfer functions from scipy.signal library to discrete-time counterparts.

The dlsim function then computes the response to the real input signal modified by the suitable steady-state value, hence simulating the discrete-time models. Unlike straightforward step response simulation, this method guarantees correct reproduction of system behaviour under different input conditions. Offset to match the original process levels, the final simulation results offer a direct comparison between model predictions and measured data.

Reflecting the empirical finding that the rise time usually provides the most consistent basis for model approximation in practical applications, the technique specifies the -based model () as the main output. This method conforms with Isermann's suggestions for process identification, which state that robust performance throughout a broad spectrum of industrial processes is provided by tangent-based parameters generated from observable response characteristics.

By means of this methodical development from geometric analysis to transfer function modeling and simulation, the turning tangent technique offers a useful framework for characterizing dynamic systems without involving sophisticated frequency-domain testing or specific identification tools. For industrial uses where simplicity, dependability, and physical interpretability are fundamental factors in control system design, this makes it more beneficial.

### PT1-Estimator

The PT1-Estimator Method determines a first-order transfer function model through the application of least squares estimation to input-output time series data.

The PT1 model is represented by the first-order differential equation:

This can be rewritten in the discrete domain as:

Where:

* y(k) is the system output (biogas production rate)
* u(k) is the system input (feeding rate)
* a and b are model parameters related to gain and time constant.

Starting with matrix setup where we build the least squares problem matrices the PT1-Estimator process While vector b comprises the ensuing output values y(2) through y(n), matrix A is generated by aggregating columns of past output values y(1) through y(n-1) and input values u(1) through u(n-1). This layout shows the time-series connection of the system.

Solving the least squares problem then allows one to estimate parameters a and b that minimise the prediction error between the model and real system behaviour. With scipy.sparse.linalg.lsqr, the implementation effectively computes these parameters even in big datasets.

Following parameter definition, a discrete transfer function is built in the manner G\_d(z) = b/(1-az^(-1). This is carried done using control.tf([b],[1, -a],sampling\_time), therefore producing a mathematical representation of the input-output link of the system.

By means of the identical input sequence applied in the original data and comparing the model's output against measured values, system response simulation validates the found model at last. This validation phase guarantees the accuracy and fit of the model for control or prediction uses.

The PT1-Estimator method has a computational simplicity relative to the Time Percentage Method as one advantage. It does not depend on characteristic points or lookup tables and calls for fewer computations.

Systems with dominating first-order behaviour, situations when computing economy is sought, and applications with limited data or noisy observations will find the PT1-Estimator appropriate. Higher-order dynamics could thus not be as well captured as the Time Percentage Method, hence model errors could result when the system shows notable second-order characteristics. When deciding the modelling technique to use for biogas generation systems, this restriction becomes a significant factor since the suitable technique should be selected depending on the expected complexity of the underlying process dynamics.

The continuous-time transfer function matching the determined model is:

T is the temporal constant; K, derived from the discrete parameters a and b, is the static gain.

By means of data-driven approaches, the PT1-Estimator method offers a direct identification of the model parameters, therefore removing the requirement for either graphical or manual parameter selection.

### PT2-Estimator

Using a second-order discrete-time model directly built from input-output data, the PT2-Estimator approach approximates system dynamics. Unlike graphical techniques, this method finds the coefficients of a second-order differential equation via least squares parameter estimation.

A second-order difference equation in the form discussed models the system:

The process starts with matrix formulation, in which input and output data build a matrix form system of equations. Whereas vector b comprises the matching output values to be anticipated, matrix A has rows of historical output and input values. Solving the least squares problem then helps to estimate the parameters by obtaining the parameter vector that reduces the squared error between the projected and real outputs. The discrete transfer function is then built from these found parameters, producing a rational function with a numerator comprising b-parameters and a denominator comprising a-parameters, all with suitable delay terms . At last, model validation is carried out by computing the response of the model to the input sequence and then verifying the accuracy of the identification process by means of a measured output comparison.

Applications of biogas plant modelling benefit much from the PT2-Estimator approach. By so faithfully capturing oscillatory dynamics and second-order effects common in complex biological systems, it provides better model fidelity than PT1 models. One of its main advantages is direct parameter identification, therefore removing the necessity for certain transient response tests that graphical techniques usually demand. By use of its least squares formulation, the method shows resilience in managing both input and output noise, hence producing stronger parameter estimate than conventional graphical methods. When producing biogas, when dynamics cannot be sufficiently explained by simpler first-order models, the PT2-Estimator is especially appropriate. Its second-order structure effectively models the complex behaviours often observed in biogas production plants by faithfully representing both fast and slow dynamic components that define biological processes.

The discrete model has an equivalent continuous transfer function of the form:

Thus, the PT2-Estimator offers a data-driven method to derive a second-order model of biogas generation without depending on particular step response tests or distinctive point measurements as necessary in the Time Percentage Method.

# Result and Discussion

References

[1] L. Cherwoo *et al.,* “Biofuels an alternative to traditional fossil fuels: A comprehensive review,” *Sustainable Energy Technologies and Assessments*, vol. 60, p. 103503, 2023, doi: 10.1016/j.seta.2023.103503.

[2] M. K. Jameel *et al.,* “Biogas: Production, properties, applications, economic and challenges: A review,” *Results in Chemistry*, vol. 7, p. 101549, 2024, doi: 10.1016/j.rechem.2024.101549.

[3] PowerUp – Parts for Gas-engines, *What are the Main Components of Biogas Plants? I PowerUP.* [Online]. Available: https://​www.powerup.at​/​knowledge/​biogas-​plant/​components/​ (accessed: Feb. 5 2025).

[4] L. A. Putra, B. Huber, and M. Gaderer, “Real-world application of a discrete feedback control system for flexible biogas production,” *Adv. Model. and Simul. in Eng. Sci.*, vol. 10, no. 1, 2023, doi: 10.1186/s40323-023-00251-1.

[5] Dr. Phillip M. Feldman, *Eight Advantages of Python Over Matlab.* [Online]. Available: https://​phillipmfeldman.org​/​Python/​Advantages\_​of\_​Python\_​Over\_​Matlab.html (accessed: Feb. 5 2025).

[6] I. Sárvári Horváth, M. Tabatabaei, K. Karimi, and R. Kumar, “Recent updates on biogas production - a review,” *BRJ*, vol. 3, no. 2, pp. 394–402, 2016, doi: 10.18331/BRJ2016.3.2.4.

[7] K. Obileke, N. Nwokolo, G. Makaka, P. Mukumba, and H. Onyeaka, “Anaerobic digestion: Technology for biogas production as a source of renewable energy—A review,” *Energy & Environment*, vol. 32, no. 2, pp. 191–225, 2021, doi: 10.1177/0958305X20923117.

[8] “modeling-approaches-of-biogas-production-3,”

[9] A. K. Duun-Henriksen *et al.,* “Model identification using stochastic differential equation grey-box models in diabetes,” *Journal of diabetes science and technology*, vol. 7, no. 2, pp. 431–440, 2013, doi: 10.1177/193229681300700220.

[10] M. Heiker, M. Kraume, A. Mertins, T. Wawer, and S. Rosenberger, “Biogas Plants in Renewable Energy Systems—A Systematic Review of Modeling Approaches of Biogas Production,” *Applied Sciences*, vol. 11, no. 8, p. 3361, 2021, doi: 10.3390/app11083361.

[11] K. Waszkielis, I. Białobrzewski, and K. Bułkowska, “Application of anaerobic digestion model No. 1 for simulating fermentation of maize silage, pig manure, cattle manure and digestate in the full-scale biogas plant,” *Fuel*, vol. 317, p. 123491, 2022, doi: 10.1016/j.fuel.2022.123491.

[12] F. Amara, K. Agbossou, A. Cardenas, Y. Dubé, and S. Kelouwani, “Comparison and Simulation of Building Thermal Models for Effective Energy Management,” *SGRE*, vol. 06, no. 04, pp. 95–112, 2015, doi: 10.4236/sgre.2015.64009.

[13] N. Raeyatdoost, R. Eccleston, and C. Wolf, “Flexible Methane Production Using a Proportional Integral Controller with Simulation‐Based Soft Sensor,” *Chem Eng & Technol*, vol. 43, no. 1, pp. 75–83, 2020, doi: 10.1002/ceat.201900401.

[14] S. Fawzy, M. Saeed, A. Eladl, and M. El-Saadawi, “Adaptive Control System for Biogas Power Plant Using Model Predictive Control,” *Journal of Modern Power Systems and Clean Energy*, vol. 9, no. 5, pp. 1193–1204, 2021, doi: 10.35833/MPCE.2019.000170.

[15] K. Yetilmezsoy, K. Karakaya, M. Bahramian, S. A. Abdul-Wahab, and B. İ. Goncaloğlu, “Black-, gray-, and white-box modeling of biogas production rate from a real-scale anaerobic sludge digestion system in a biological and advanced biological treatment plant,” *Neural Comput & Applic*, vol. 33, no. 17, pp. 11043–11066, 2021, doi: 10.1007/s00521-020-05562-7.

[16] P. V. Denis Dochain, “Dochain Vanrolleghem 2015 dynamical modelling\_compressed-1-120,” 2001.

[17] K. Ogata, “Modern control engineering,” 2010.

[18] F. Haugen, R. Bakke, and B. Lie, “State Estimation and Model-Based Control of a Pilot Anaerobic Digestion Reactor,” *Journal of Control Science and Engineering*, vol. 2014, pp. 1–19, 2014, doi: 10.1155/2014/572621.

[19] Bernard, O., Hadj-Sadok, Z., Dochain, D., Genovesi, A., & Steyer, J. P, “Dynamical model development and parameter identification for an anaerobic wastewater treatment process,” 2001.

[20] K. P. Burnham and D. R. Anderson, “Multimodel Inference,” *Sociological Methods & Research*, vol. 33, no. 2, pp. 261–304, 2004, doi: 10.1177/0049124104268644.

[21] A.-N. Spiess and N. Neumeyer, “An evaluation of R2 as an inadequate measure for nonlinear models in pharmacological and biochemical research: a Monte Carlo approach,” *BMC pharmacology*, vol. 10, p. 6, 2010, doi: 10.1186/1471-2210-10-6.

[22] Lennart Ljung, “Black-box Models from Input-output Measurements,” Oct. 2001.

[23] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature,” *Geosci. Model Dev.*, vol. 7, no. 3, pp. 1247–1250, 2014, doi: 10.5194/gmd-7-1247-2014.

[24] G. K. Sadanori Konishi, “Information Criteria and Statistical Modeling,” 2007.

[25] T. A. S. Srinvias, A. D. Donald, M. Sameena, K. Rekha, and I. D. Srihith, “Unlocking the Power of Matlab: A Comprehensive Survey,” *IJARSCT*, pp. 20–31, 2023, doi: 10.48175/IJARSCT-9005.

[26] M. Abdul Majid, Z. A. Huneiti W. Balachandran Y. Balarabe, “MATLAB as a teaching and learning tool for mathematics,”

[27] Daniel Gaida, Michael Bongards, Christian Wolf, Thomas Bäck, “MATLAB Toolbox for Biogas Plant Modelling and Optimization,” 2011. [Online]. Available: https://​www.researchgate.net​/​publication/​224969105

[28] “Genetic Algorithm and Direct Search Toolbox User's Guide,” 2004.

[29] Mahamad Nabab Alam, “Particle Swarm Optimization: Algorithm and its Codes in MATLAB,” doi: 10.13140/RG.2.1.4985.3206.

[30] O. S. Ajani, A. Kumar, and R. Mallipeddi, “Covariance matrix adaptation evolution strategy based on correlated evolution paths with application to reinforcement learning,” *Expert Systems with Applications*, vol. 246, p. 123289, 2024, doi: 10.1016/j.eswa.2024.123289.

[31] K. P. RAINER STORN, “Differential Evolution; A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,” 1997.

[32] T. P. Runarsson, “Approximate Evolution Strategy using Stochastic Ranking,” pp. 745–752, 2006, doi: 10.1109/CEC.2006.1688386.

[33] StatAnalytica, *Python vs Matlab: Which One Is the Best Language - StatAnalytica.* [Online]. Available: https://​statanalytica.com​/​blog/​python-​vs-​matlab/​ (accessed: Feb. 13 2025).

[34] N. S. Vineesh Cutting, “A Review on using Python as a Preferred Programming Language for Beginners,” Aug. 2021.

[35] D.B. Beniz, A.M. Espindola [LNLS, Campinas, and Brazil], “11th International Workshop on Personal Computers and Particle Accelerator Controls: October 25th-28th, 2016, Campinas, Brazil,” Sep. 2017.

[36] R. Isermann and M. Münchhof, “Identification of Dynamic Systems,” 2011, doi: 10.1007/978-3-540-78879-9.

[37] W. Schneider, *Praktische Regelungstechnik: Ein Lehr- und Übungsbuch für Nicht-Elektrotechniker ; mit 72 Tabellen und 54 Übungsaufgaben,* 3rd ed. Wiesbaden: Vieweg + Teubner, 2008. Accessed: Feb. 27 2025.