



Conference proceedings
Sborník odborného semináře

**Nové metody a postupy v oblasti přístrojové
techniky, automatického řízení a informatiky 2025**
**New Methods and Practices in the Instrumentation,
Automatic Control and Informatics 2025**

28. – 30. 5. 2025, Chlumeck nad Cidlinou

ISBN 978-80-01-07478-7

OBSAH

Richard Kokštein, Cyril Oswald, Michal Kuchař

EMPIRICAL COMPARISON OF PYTHON DAG FRAMEWORKS FOR LOCAL DATA PIPELINES3

Anna Kopecká

DEVELOPMENT OF WEB APPLICATION FOR MONITORING A HYDROPONIC FARM 11

Matej Masata

WEB INTERFACE FOR A AGRICULTURAL INFORMATION SYSTEM 16

Petr Valenta

DDW MODEL IDENTIFICATION 22

Valerii Lisnychuk

MATEMATICKÝ MODEL KVADROKOPTÉRY SE ZAVĚŠENÝM ZÁVAŽÍM VE 3D 28

Petr Musil

ÚLOHY PRO VIRTUÁLNÍ LABORATOŘ V JAVASCRIPTU 38

Aruzhan Yerzhanova

ZESÍŤOVÁNÍ ROHOVKY S INTAKTNÍM EPITELEM: SOUČASNÉ MOŽNOSTI A PERSPEKTIVY 43

Matouš Urbanec, Marie Martinásková

APLIKACE FLUIDNÍHO SVALU PŘI POKROČILÉM ŘÍZENÍ BALANCÉRU 46

Martin Vitoušek, Jakub Jura, Matouš Cejnek, Klára Scháňková, Luděk Laňar

SOFTWARE PRO MONITORING VÝNOSU PLODŮ 49

EMPIRICAL COMPARISON OF PYTHON DAG FRAMEWORKS FOR LOCAL DATA PIPELINES

Richard Kokštein¹, Cyril Oswald¹, Michal Kuchař¹

¹*Department of Instrumentation and Control, Czech Technical University in Prague, Technická 4, Prague 6, richard.kokstein@fs.cvut.cz*

Abstract: Building automated and reproducible data pipelines is an essential part of any data-science work. Choosing the right tool for modeling the pipelines as directed acyclic graphs is therefore crucial. This study compared the capabilities of Luigi, Snakemake, Kedro, Dagster, and Prefect for building Python-based, single-machine pipelines on Windows 10 and CPython 3.11. For each framework, a sample project was created that joined Excel files, cleaned missing values, cached intermediate results as Parquet files, and trained a linear-regression model. The frameworks were scored on their pipeline-definition features, persistent caching, node parameterization, requests for data materialization from external scripts, and pipeline visualization. Snakemake, Dagster and Prefect scored four out of five points, Luigi scored three, and Kedro scored one. Evidence shows no single tool excels in every area, but Snakemake provides the best features for building pipelines under the given constraints. These findings can guide data scientists who need to select a lightweight pipeline tool that does not rely on containers or cloud platforms.

1 Introduction

Data-science projects, be it for dataset preprocessing or data mining for reports, typically begin as unstructured interactive notebooks for Exploratory Data Analysis (EDA) and gradually evolve into pipelines composed of multiple scripts and notebooks. To remain manageable, reproducible, and automatable, these pipelines must be modeled as Directed Acyclic Graphs (DAGs). Numerous frameworks exist for this purpose, ranging from tools for small local projects to platforms designed for big data and large cloud deployments. Some frameworks focus on the scheduling and orchestration of computational resources, while others focus on the pipeline construction, caching of intermediate results, and parameter sweeps. In collaborative data-science work, a predictable and well-organized project layout is equally important so that all contributors can navigate the codebase and contribute to it. One example of such project structure is given by Cookiecutter Data Science¹. One of the choices when adopting the structure is deciding on a tool for managing data pipelines and modeling the DAGs. This empirical comparison evaluates several popular Python-based DAG frameworks for managing local data pipelines and provides practical insights for selecting the most appropriate tool.

1.1 Comparison Objective

The evaluated Python-based DAG frameworks for local data-processing pipelines in this study must **a)** support Microsoft Windows 10, **b)** support CPython 3.11 (chosen for compatibility with libraries such as PyCaret), **c)** be installable with `pip` into a project-local virtual environment managed by `venv`, and **d)** have no reliance on Windows Subsystem for Linux (WSL), Docker, or cloud-native services.

The selected frameworks are assessed on their ability to **a)** define pipelines concisely and idiomatically, **b)** cache intermediate results persistently to avoid redundant computation and automatically propagate updates through the pipeline, **c)** permit straightforward path-based parameterization of pipeline steps, **d)** enable ad-hoc task execution and data retrieval from external scripts and Jupyter Notebooks, and **e)** generate clear visualizations of the resulting DAG.

¹ <https://cookiecutter-data-science.drivendata.org>

Outside the scope of this study and therefore not evaluated are features such as **a)** task scheduling across clusters, **b)** hooks/sensors for external events, **c)** load distribution, **d)** task parallelization, **e)** direct integration with cloud storage, and **f)** web-based interfaces with features exceeding simple DAG visualization. This study focuses exclusively on single-machine, locally stored and processed data.

Each framework is assessed by implementing an illustrative yet complete pipeline that **a)** loads several Excel files, **b)** concatenates them, **c)** cleans the data by either discarding columns with missing values or imputing them with column mean, **d)** persists intermediate results as Parquet files to enable caching, and **e)** fits a linear-regression model. Ad-hoc capabilities are evaluated by executing the pipelines and retrieving the results from Jupyter Notebooks. For each framework, a short excerpt from the pipeline definition will be shown as a code listing alongside a DAG visualization made using the built-in tools. All implementations are hosted on GitHub².

1.2 Related Work

In [1], the author reviewed workflow-orchestration tools suitable for inclusion in the ANALYZA platform [2]. They emphasized functionality, scalability, security, and usability in production environments, specifically a Kubernetes platform. The evaluated frameworks were Airflow, Prefect, Snakemake, and Argo Workflows. Honorable mentions included Luigi, Flyte, Dagster, and Nextflow. The study ultimately selected Argo Workflows. In contrast, this study examines tools for locally executed Python-based data pipelines on Windows and does not address distributed, scheduled, or scalable computing.

In [3], the authors surveyed more than forty open-source and commercial pipeline solutions and organised them into four groups: **a)** Extract Transform Load (ETL) and Extract Load Transform (ELT) data pipelines, **b)** integration, ingestion, and transformation, **c)** pipeline orchestration and workflow management, and **d)** machine learning and model deployment. They selected one tool from each group and implemented four representative case studies on the IDEAL Household Energy Dataset [4, 5]. The selected tools were **a)** Apache Spark with PySpark in Jupyter Notebook, **b)** Microsoft SQL Server Integration Services (SSIS), **c)** Apache Airflow, and **d)** TensorFlow Extended (TFX). The implementations are available on GitHub³. Their preferred option was Apache Airflow. This study takes inspiration from the survey's sample implementations but focuses in greater detail on a smaller set of tools intended for locally executed Python-based data pipelines on Windows.

2 DAG Frameworks

For the comparison, the following frameworks were selected: **Luigi**⁴, **Snakemake**⁵, **Kedro**⁶, **Dagster**⁷ and **Prefect**⁸. Additional frameworks were considered but excluded for not meeting all the requirements. These are:

- **Apache Airflow**⁹ – Airflow is not directly runnable on Windows 10. This can be circumvented by using the dockerized version. Airflow is very feature rich but also resource-intensive.
- **Argo Workflows**¹⁰ – Argo Workflows is primarily focused on orchestrating parallel jobs on Kubernetes. For more information, see [1].
- **Metaflow**¹¹ – Metaflow is not natively supported on Windows. This can be circumvented by running it within a WSL 2 environment.
- **Flyte**¹² – Flyte is a Kubernetes-native workflow orchestrator that executes tasks within containers.
- **DataChain**¹³ – DataChain is an ETL framework that focuses on processing unstructured and multimodal data, such as images, text, and PDFs, rather than the typical tabular data.
- **Bonobo**¹⁴ – Bonobo is an ETL framework that did not progress beyond the alpha stage. The last supported Python version was 3.7, and the project appears to be abandoned.
- **JUDI**¹⁵ – JUDI (JUst Do It) is focused on parameter sweeps for bioinformatics. It is built on top of doit. The last tested Python version was 3.6, and the project appears to be abandoned.

² <https://github.com/Trolobezka/local-py-dag-frameworks>

⁴ <https://github.com/spotify/luigi>

⁷ <https://github.com/dagster-io/dagster>

¹⁰ <https://github.com/argoproj/argo-workflows>

¹³ <https://github.com/iterative/datachain>

⁵ <https://github.com/snakemake/snakemake>

⁸ <https://github.com/PrefectHQ/prefect>

¹¹ <https://github.com/Netflix/metaflow>

¹⁴ <https://github.com/python-bonobo/bonobo>

³ <https://github.com/DataEng-AML/surveypaper>

⁶ <https://github.com/kedro-org/kedro>

⁹ <https://github.com/apache/airflow>

¹² <https://github.com/flyteorg/flyte>

¹⁵ <https://github.com/ncbi/JUDI>

Honorable mentions of tools and frameworks related to pipeline execution and data processing are:

- **doit**¹⁶ – Doit is a general-purpose task execution tool whose task definitions are written in Python. It is similar to GNU Make or Snakemake. Although its core features are declared stable, the project has been inactive for two years, and the latest tested Python version is 3.10.
- **Joblib**¹⁷ – Joblib provides tools such as persistent caching and parallel execution for lightweight data pipelines. It is a general package that is not targeted toward a specific use case, such as managing data and workflows. It could serve as a solid foundation for developing a custom framework.
- **MLflow**¹⁸ – MLflow offers experiment tracking and model management. One notable feature was recipes, which were executable workflows for transforming data and fitting and evaluating models. This feature is now deprecated.

2.1 Luigi

“Luigi is a Python [...] package that helps you build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization, handling failures, command line integration, and much more.” [6, 7] The basic unit of work in Luigi is the `luigi.Task`. It is a class that contains the functions `requires`, `run` and `output`. An example of a derived task class from the sample project can be seen in Listing 1. The functions `requires` and `output` return the targets that the particular task requires and creates, respectively. This defines the structure of pipelines, as `requires` can return other tasks. The `run` function contains the logic for processing the input targets and creating the output targets. Tasks are parametrized with class attributes of the type `luigi.Parameter`. Targets can represent a variety of data sources. They only need to implement the `exists` function.

```
class CleanData(SingleInMixin, SingleOutMixin, luigi.Task):
    method = luigi.EnumParameter(enum=CleanDataMethod) # drop, impute

    def requires(self) -> ConcatData: # type: ignore
        return ConcatData()

    def run(self) -> None:
        concat = pandas.read_parquet(self.input_path)
        if self.method == CleanDataMethod.drop:
            clean = drop_columns_with_missing_values(concat, threshold=0.2)
        elif self.method == CleanDataMethod.impute:
            clean = impute_columns_with_mean(concat)
        else:
            raise NotImplementedError(f"Unknown method: '{self.method}'")
        clean.to_parquet(self.output_path)

    def output(self) -> luigi.LocalTarget: # type: ignore
        return luigi.LocalTarget(ROOT / "data" / "processed" /
                                   f"clean_{self.method.name}.parquet")
```

Listing 1: Subclass of `luigi.Task` defining a task in a Luigi pipeline

Pipelines in Luigi can be executed by local or central schedulers, both of which can run locally. The central scheduler offers a web-based interface that can visualize currently running and completed tasks. Figure 1 shows visualization from the web interface for the sample project. Before running a task from a pipeline, the scheduler checks whether the output targets exist. If they do, the task will not be executed. While this provides basic caching, it also prevents the automatic recalculation of outdated tasks and the propagation of updates through the pipeline.

The `requires` function cannot directly return a `luigi.LocalTarget` pointing to a file on disk. Therefore, the sample project uses a more granular definition of each pipeline step, as shown in Figure 1. Each raw input Excel file is wrapped in a parameterized task, `RawDataPoint`. Another parameterized task, `DataPoint`, then converts these into Parquet files. The rest of the pipeline is consistent with other approaches presented here. The custom mixins in Listing 1 are optional and only provide more convenient access to target paths.

¹⁶ <https://github.com/pydoit/doit> ¹⁷ <https://github.com/joblib/joblib> ¹⁸ <https://github.com/mlflow/mlflow>

2.2 Snakemake

“The Snakemake workflow management system is a tool to create reproducible and scalable data analyses.” [8, 9] Snakemake pipelines are defined in a separate file `Snakefile` using a Python-based language with YAML-like syntax for rules. An excerpt of the `Snakefile` from the sample is shown in Listing 2. Rules specify input and output files and a console command or a script file for data processing. Because of the Python support, rules can be dynamic. Rules are parameterized in two ways. In path-based parameterization, the input and output paths contain wildcards, so one rule can create multiple outputs. Rules can also use global parameters loaded from configuration files or Command Line Interface (CLI) arguments. Input dependencies are defined by referencing the outputs of other rules, which defines the data pipeline. Pipelines run by requesting specific rules or, for wildcard rules, by requesting specific data files. When a rule executes a Python script, the `snakemake` variable is automatically injected into its global namespace and contains all information about the current run.

```
rule clean_data:
    input:
        rules.concat_data.output,
    output:
        "data/processed/clean_{method}.parquet",
    params:
        threshold=0.2,
    script:
        "scripts/clean_data.py"
```

Listing 2: Rule definition in Snakemake

Before execution, a DAG is built from the rule definitions and dependencies are checked from the bottom up. Snakemake tracks modification times of relevant files, absolute paths of input files, rule parameters, and rule definition bodies. If changes are found, the rules are rerun and the update is propagated through the pipeline. In the current implementation, the rerun triggers are very sensitive, resulting in unnecessary reruns of cached rules with limited configuration options.

Snakemake is used through the CLI, similar to GNU Make, so ad-hoc requests for rules or files from scripts and notebooks are limited. A DAG visualization can be exported in Graphviz format. The rendered result for the sample project is shown in Figure 2. Snakemake also offers functionality for generating reports.

2.3 Kedro

“Kedro is an open-source Python framework to create reproducible, maintainable, and modular data science code.” [10, 11] Kedro works with projects structured as Python packages. Pipelines in Kedro are built from nodes, which are defined by pure Python functions with named inputs and outputs. The names of the input and output datasets correspond to entries in the data catalog, which uses the YAML format to define datasets. Two node definitions and an excerpt from the data catalog from the sample project can be seen in Listings 3 and 4, respectively. The data catalog specifies the location and type of the data. Kedro offers several data drivers for loading, working with, and saving different file types. These drivers operate in the background, allowing the pipeline nodes to receive the loaded datasets without interacting with the file system.

```
kedro.pipeline.node(
    func=clean,
    inputs={"concat": "concat_data", "method": "params:drop"},
    outputs="clean_drop_data",
    name=clean.__name__ + "_drop" + "_node",
),
kedro.pipeline.node(
    func=clean,
    inputs={"concat": "concat_data", "method": "params:impute"},
    outputs="clean_impute_data",
    name=clean.__name__ + "_impute" + "_node",
)
```

Listing 3: Two node definitions in Kedro

```
"clean_{method}_data":
  type: pandas.ParquetDataset
  filepath: "data/processed/clean_{method}.parquet"
```

Listing 4: Parameterized entry in data catalog in Kedro

The data catalog allows wildcards to be used in dataset entries to parameterize the catalog. Unfortunately, the matched result of the wildcard is unavailable in the node. Because of this, the node for processing the parameterized datasets must be manually repeated in the pipeline definition. This can be seen in the parameterized catalog entry in Listing 4, which defines the output of the nodes in Listing 3.

Currently, Kedro does not offer automatic file-based caching. Pipelines must execute all nodes, even when there is no change in inputs, parameters, or code. However, a third-party plugin is available that implements basic caching by tracking changes in inputs and function bodies. Kedro offers a web-based visualization tool, `kedro-viz`, that can be used as a standalone tool or within the official Visual Studio Code (VS Code) extension for Kedro. Its output for the sample project is shown in Figure 3. To invoke pipelines and retrieve datasets during EDA, Kedro provides a Jupyter Notebook extension.

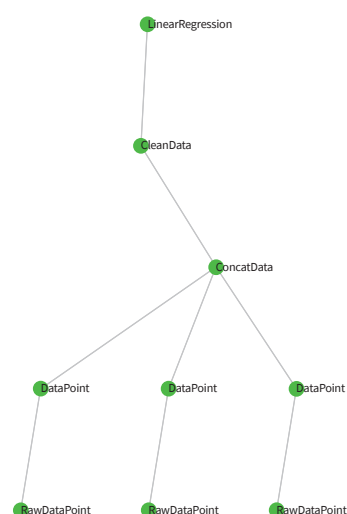


Figure 1: Visualization of the Luigi pipeline using the web-based interface

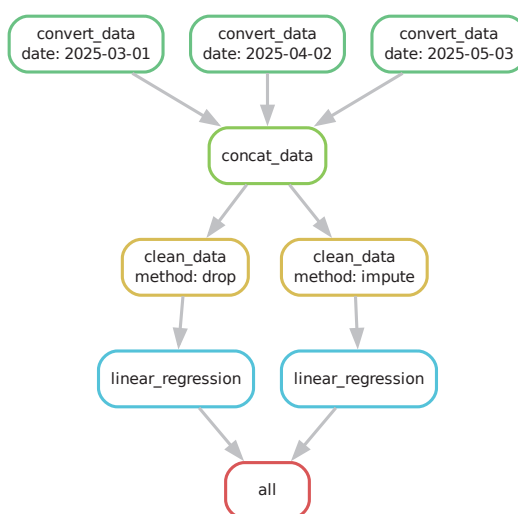
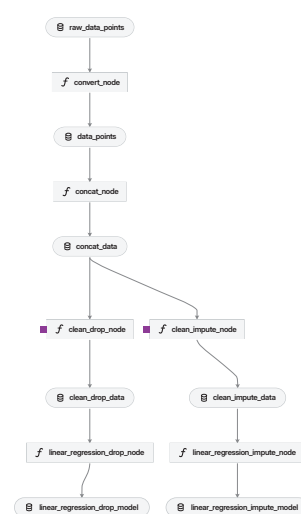


Figure 2: Visualization of the Snakemake pipeline using a CLI command and Graphviz

Figure 3: Visualization of the Kedro pipeline using the `kedro-viz` tool

2.4 Dagster

“Dagster is a next-generation open source orchestration platform for the development, production, and observation of data assets.” [12, 13] The main building block of a Dagster pipeline is an asset. An asset is defined by applying the `dagster.Asset` decorator to a Python function that manipulates data in persistent storage. Listing 5 shows a definition of an asset from the sample project. Pipelines are built by chaining assets into a DAG. The process of asset materialization collects data from upstream assets, executes the function, and saves the result to storage. When the code or dependencies change, the asset becomes unsynchronized. Without advanced automation or scheduling, assets must be materialized manually to propagate updates. When assets are requested from code, Dagster does not automatically rematerialize unsynchronized upstream assets. It either executes the whole pipeline or returns the most recent materialization, which may be outdated.

Assets can be parameterized by time, categorical, or spatial partitions, allowing multiple versions or slices of the data to exist. Downstream assets must either use the same partitions or behave as aggregators, combining all incoming partitions into one asset.

Dagster provides a local development server with a web-based interface for pipeline management and visualization. A visualization of the pipeline from the sample project is shown in Figure 4. The web interface is necessary for comfortable use. During materialization, the code can emit metadata, which are then displayed in the interface as tables, Markdown, images, or plots.

```

@dagster.asset(
    code_version="v1",
    partitions_def=clean_partition, # ["drop", "impute"]
    description="Concatenated data with columns with missing values removed or imputed.",
)
def clean_data(
    context: dagster.AssetExecutionContext,
    concat_data: pathlib.Path,
) -> pathlib.Path:
    method = context.partition_key # current partition
    concat = pandas.read_parquet(concat_data)
    if method == "drop":
        clean = drop_columns_with_missing_values(concat, threshold=0.2)
    elif method == "impute":
        clean = impute_columns_with_mean(concat)
    else:
        raise NotImplementedError(f"Unknown partition key: '{method}'")
    output_path = ROOT / "data" / "processed" / f"clean_{method}.parquet"
    context.add_output_metadata(dataframe_to_metadata(clean))
    clean.to_parquet(output_path)
    return output_path

```

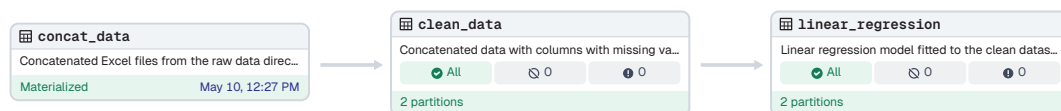
Listing 5: Asset definition in Dagster with `dagster.asset` function decorator

Figure 4: Global asset lineage in Dagster

2.5 Prefect

“Prefect is an open-source orchestration engine that turns your Python functions into production-grade data pipelines with minimal friction. You can build and schedule workflows in pure Python—no DSLs or complex config files—and run them anywhere.” [14] In Prefect, data pipelines are built from tasks and flows, which are defined using the `@prefect.task` and `@prefect.flow` function decorators, respectively. Tasks support persistent caching and represent the most granular, reusable operations. Prefect invalidates caches when task inputs or function definitions change. It is also possible to define custom invalidation triggers that check file modification times or file hashes. A task from the sample project is shown in Listing 6. Flows orchestrate tasks or other flows, effectively defining the pipeline. Both tasks and flows are parameterized using standard function arguments. Intermediate results from tasks must be manually persisted to disk within the function body.

```

@prefect.task(cache_policy=INPUTS + TASK_SOURCE + MTIME + BYTES)
def clean(input_path: Path, method: CleanDataMethod) -> Path:
    concat = pandas.read_parquet(input_path)
    if method == CleanDataMethod.drop:
        clean = drop_columns_with_missing_values(concat, threshold=0.2)
    elif method == CleanDataMethod.impute:
        clean = impute_columns_with_mean(concat)
    else:
        raise NotImplementedError(f"Unknown method: {method}")
    output_path = PROCESSED_DIR / f"clean_{method.name}.parquet"
    clean.to_parquet(output_path)
    add_table_to_artifact(clean, key=f"table-{method.name}")
    return output_path

```

Listing 6: Task definition in Prefect with `prefect.task` function decorator

Prefect provides a local server with a web-based interface. The interface lists historical and ongoing runs and visualizes the execution timeline. A visualization for the sample project can be seen in Figure 5. To execute a flow from an external script or notebook, the flow-decorated function is imported and called with the appropriate arguments. The output of a task in the executed flow is available in a returned Python dictionary containing all the intermediate results. Alternatively, a partial flow can be defined that includes only the desired tasks and returns the output from the final task. During execution, the pipelines can create artifacts, such as progress bars, Markdown text, tables, and images, which are then displayed in the web interface.



Figure 5: Visualization of the Prefect pipeline using the web-based interface

3 Results and Discussion

Table 1 shows the frameworks with scores for each objective. The scores were assigned based on the comparison objectives declared in Section 1.1 and insights from sample projects implemented with selected frameworks. This section explains the scores of -1 , 0 , and 1^+ with some contextual remarks at the end. Scores of 1 are not discussed.

Table 1: Comparison of selected frameworks across five feature categories. A score of 1 indicates the feature is documented, available, and readily usable. A score of 0 denotes the feature is only accessible through custom extensions or manual effort. A score of -1 means the feature is unavailable or difficult to implement. A superscript $+$ marks well-designed features with enhanced usability. The total column summarizes the scores for each tool.

Tool	Definitions	Caching	Parameterization	Requests	Visualization	Total
Luigi 3.6.0	1^+	-1	1	1	1	3
Snakemake 9.3.4	1^+	1	1^+	0	1	4
Kedro 0.19.13	0	-1	0	1^+	1	1
Dagster 1.10.17	1	0	1	1	1^+	4
Prefect 3.4.1	1	0	1	1	1^+	4

Luigi receives -1 for caching because it provides no official support beyond a simple file-existence check. Kedro also receives -1 for caching for the same reason as Luigi.

Snakemake receives 0 for requests because its CLI is not convenient to use from a script. Kedro receives 0 for definitions because datasets must be declared in the data catalog, and their names must be reused in a separate script to define nodes. This introduces manual bookkeeping in larger projects, unless the VS Code extension is used. Kedro also receives 0 for parameterization because, although wildcards can be used in the data catalog, they lead to multiple node definitions. Dagster receives 0 for caching because unsynchronized assets must be rematerialized manually through the web interface. Prefect receives 0 for caching because it does not offer cache invalidation on changes in modification times or content hashes of input files.

Luigi receives 1^+ for definitions because of its unique, simple, and object-oriented interface. Snakemake receives 1^+ for definitions because of its unique YAML-like format with Python support, which allows for dynamic rule definitions. Snakemake also receives 1^+ for parameterization because of its well-designed wildcard system, which injects matched values into the running script. Kedro receives 1^+ for requests because of its Jupyter Notebook extension, which provides straightforward access to the pipeline. Dagster receives 1^+ for visualization because of its clear, modern, and interactive pipeline visualization in the web interface. Prefect also receives 1^+ for visualization for the same reason as Dagster.

The Kedro scores are not surprising given that the framework is currently at version 0.19.13 and is in active development. It is expected that in upcoming versions, the missing or partial features will become available.

4 Conclusion

This empirical study confirms that no single framework universally outperforms the others, even under narrow objectives and strict constraints, such as a framework for local Python-based data pipelines for CPython 3.11 on Windows 10. Based on the scores in Table 1 and insights from the hands-on experience with the frameworks from the sample project implementations, Snakemake is the most suitable tool for building data pipelines under previously mentioned constraints and objectives. Snakemake builds on the strengths of tools like GNU Make while simplifying the rule definitions with its YAML-like format with Python support. The absence of functions for requesting rule executions or file materializations from external scripts and notebooks can be overcome by using the subprocess module in Python, system escape commands in Jupyter Notebooks, or by creating a custom extension utilizing Snakemake's internals.

Acknowledgment

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS24/125/OHK2/3T/12.

References

- [1] R. Kyčerka, "Orchestration tool for the modular computing subsystem of the ANALYZA platform," Master's Thesis, Masaryk University, Faculty of Informatics, Brno, 2022. [Online]. Available: <https://is.muni.cz/th/k0anq> (visited on 05/08/2025).
- [2] T. Rebok *et al.*, *ANALYZA - Výpočetní a orchestrační subsystém*, Masarykova univerzita, 2020. [Online]. Available: <https://is.muni.cz/publication/1736351> (visited on 06/12/2025).
- [3] A. Mbata, Y. Sripada, and M. Zhong, *A Survey of Pipeline Tools for Data Engineering*, 2024. DOI: [10.48550/arXiv.2406.08335](https://doi.org/10.48550/arXiv.2406.08335). arXiv: [2406.08335](https://arxiv.org/abs/2406.08335) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2406.08335> (visited on 05/12/2025).
- [4] M. Pullinger, J. Kilgour, N. Goddard, *et al.*, "The IDEAL household energy dataset, electricity, gas, contextual sensor data and survey data for 255 UK homes," *Scientific Data*, vol. 8, no. 1, p. 146, 2021. DOI: [10.1038/s41597-021-00921-y](https://doi.org/10.1038/s41597-021-00921-y).
- [5] N. Goddard, J. Kilgour, M. Pullinger, *et al.*, *IDEAL Household Energy Dataset*, University of Edinburgh, School of Informatics, 2021. DOI: [10.7488/ds/2836](https://doi.org/10.7488/ds/2836).
- [6] E. Bernhardsson, E. Freider, *et al.*, "Getting Started - Luigi 3.6.0 documentation." (2025), [Online]. Available: <https://luigi.readthedocs.io> (visited on 05/10/2025).
- [7] M. Rieger, M. Erdmann, B. Fischer, and R. Fischer, *Design and Execution of make-like, distributed Analyses based on Spotify's Pipelining Package Luigi*, 2017. DOI: [10.48550/arXiv.1706.00955](https://doi.org/10.48550/arXiv.1706.00955). arXiv: [1706.00955](https://arxiv.org/abs/1706.00955) [physics.data-an]. [Online]. Available: <https://arxiv.org/abs/1706.00955> (visited on 05/12/2025).
- [8] J. Köster *et al.*, "Snakemake | Snakemake 9.5.1 documentation." (2025), [Online]. Available: <https://snakemake.readthedocs.io> (visited on 05/10/2025).
- [9] F. Mölder *et al.*, "Sustainable data analysis with Snakemake," *FI000Research*, vol. 10, p. 33, 2021, issn: 2046-1402. DOI: [10.12688/fi000research.29032.2](https://doi.org/10.12688/fi000research.29032.2).
- [10] Kedro. "Introduction to Kedro - kedro 0.19.13." (2025), [Online]. Available: <https://docs.kedro.org/en/stable/introduction/index.html> (visited on 06/12/2025).
- [11] L. Bălan, K. Kunii, L. Hoang, *et al.*, *quantumblacklabs/kedro: 0.17.3*, version 0.17.3, 2021. DOI: [10.5281/zenodo.4707715](https://doi.org/10.5281/zenodo.4707715). [Online]. Available: <https://doi.org/10.5281/zenodo.4707715> (visited on 05/12/2025).
- [12] Dagster Labs. "Modern Data Orchestrator Platform | Dagster." (2025), [Online]. Available: <https://dagster.io> (visited on 05/10/2025).
- [13] H. Picatto, G. Heiler, and P. Klimek, *Cost-Effective Big Data Orchestration Using Dagster: A Multi-Platform Approach*, 2024. DOI: [10.48550/arXiv.2408.11635](https://doi.org/10.48550/arXiv.2408.11635). arXiv: [2408.11635](https://arxiv.org/abs/2408.11635) [cs.DC]. [Online]. Available: <https://arxiv.org/abs/2408.11635> (visited on 05/12/2025).
- [14] Prefect Technologies, Inc. "Introduction - Prefect." (2025), [Online]. Available: <https://docs.prefect.io/v3/get-started> (visited on 05/23/2025).

DEVELOPMENT OF WEB APPLICATION FOR MONITORING A HYDROPONIC FARM

Anna Kopecká¹

¹ Department of Instrumentation and Control Engineering, Czech Technical University in Prague, Technická 4, Prague, Czech Republic

anna.kopecka@fs.cvut.cz

Abstract: This article describes the development of a full stack application as a monitoring system of a hydroponic farm. Main feature of the application is retrieval of the measured data via API using Python library FastAPI and their visualization in Java Script framework Vue.js. The data are saved to PostgreSQL database, where the communication with API ensures part of back-end using Python library SQLAlchemy. The application is deployed on the Docker platform.

Keywords: full-stack application, monitoring system, hydroponic farm, FastAPI, Vue.js, Docker.

1 Introduction

Nowadays we can observe a pressure on the productivity of agriculture. That is caused by several reasons like decreasing number of human resources in agriculture, growing population and the trend of people moving to the cities. Urbanization causes an expansion of the cities which alongside other factors leads to the decreasing amount of agricultural land [1] [2].

The main challenge in agriculture is therefore an optimization of agricultural processes in the term of productivity and alongside the reduction of financial costs [1]. That is achieved by use of various technologies including IoT, sensors [2]. The core of the process optimization is measuring various quantities like temperature or pH of a substrate and the following analysis of the data. This analysis can be performed [2] in real time observation or with retrospective data analysis for which the save of the data is needed. Monitoring is therefore one of the key aspects of smart farming

The article focuses on the development of full stack application monitoring a hydroponic farm on the Faculty of Mechanical Engineering at the Czech Technical University in Prague. The main quantities controlled via PLC Siemens Simatic 1200 are temperature, pH and EC [3]. Even though there is a HMI interface and SCADA system [3], these systems often requires certain technical knowledge. However, an agricultural expert evaluating and checking the process of growth might not be proficient in the use of such systems [1]. Web application as a way of monitoring fulfills the requirement of basic knowledge of Internet and its use builds therefore another step if interconnection of human expertise from non-technical field with the technically advanced solution. The data retrieval enables the use of data for later analysis. The implementation of back-end in the Python programming language makes it easy extensible for future applications like the use of AI for data analysis.

2 Full Stack Application

2.1 Main tasks

The main task of the web application is to display the retrieved data from the sensors with the possibility of orientation in the data like a date time filtration. The data can be then analyzed in their raw form in a table or in the graphical form with the possibility of displaying data from multiple sensors in one plot. That can help in the optimization based on the analysis of dependencies of the measured quantities. The observation of the farm is ensured via the connection to the camera in the hydroponic farm. The last part is the feature enabling you to add the devices and edit the parameters like unit, name of the device and URL address of the device. These parameters of the sensors are stored in the database in the table for device management and have an unique key assigned to them. The structure for table containing the measured values contains id of the measurement, timestamp, measured value and id of the device as a secondary key through which the measured value is bound to its sensor.

2.2 Implementation and used technologies

For the database the server PostgreSQL has been chosen due to its robustness and ability to process concurrent transactions [4]. That is crucial for such an application requiring the ability to save measured data along with the fact that larger number of users can be expected to use the web application. SQLAlchemy is responsible for the database creation including tables, and building the database from metadata in case the connection has been interrupted. Table definition uses declarative mapping when the instance of table created from metadata is automatically assigned to the object used for the data manipulation [5]. That makes the code simpler and more readable compared with the imperative mapping.

Database interface is built from two parts – managers. They are classes designed to handle data by their nature. One manager handles the basic device specification, second the dataflow of the data from measurements. Especially the manager handling the device specification is crucial part of the back-end since it does not only provide access points to the data from the devices but also contains verification methods and informative methods like checking whether a device already exists in the database. Therefore it is widely used in other parts of the back-end for verification or linking the data.

Streaming is executed with the help of a runner. This runner creates thread every time a streaming device is called and has not been called before. The streaming is then performed on this thread and read from the main thread. The retrieval of the frames and the connection to the camera uses VideoCapture class from popular library for image processing OpenCV.

The connection of the database and streaming with the outside world ensures FastAPI - a lightweight and well-documented API chosen for its simplicity and extendibility [6]. The API includes routers which similarly like managers group the endpoints according to the data they handle. There are routers providing web-endpoints for the managers and router built on top of the stream runner. The request scheme as a template defines data models – prescription how the input parameters of the API endpoints should look like. These models inherit BaseModel from pydantic library which also performs validation of the input data passed to the API endpoints.

Front-end is built as a single page application in Vue.js framework. The application is composed of several single page components. The official page of Vue.js defines single page components as components where one file contains all – the component's logic written in java script, HTML template and CSS styles [7]. The web app for the hydroponic management system follows this idea with one difference - the application is designed mainly with the help of the Bootstrap toolkit and therefore styling is defined in separate *sass* files.

The components can be generally divided into two parts – views which corresponds to pages in multi-page applications and components like bricks from which the main views are built. The components use the data retrieved from API and implement its own logic to embed this data to the HTML elements defined inside the HTML template. The logic also handles the manipulation with the elements in the user interface.

Routing of single views inside the web application ensures Vue Router – an official router from Vue.js developers. The most important views include view for camera output (see *Figure 1*), views for graphical and tabular measurements representation and views for device management and registration.

The application also contains interface layer written in JavaScript for the communication with the back-end API. This communication is ensured via axios – HTTP client for browser and Node.js. Interface performs data retrieval from the API as well as basic data processing like mapping to meet the needs of the web application.

The graphical visualization (see *Figure 2*) of measured data is done via implementation of the open-source library plotly.js. The visualization offers display of data from multiple devices in one plot at the same time. The possibility of assigning measurements to one of two vertical axes on the control panel with the adjustable range helps to observe data of two groups of devices with significantly different ranges in which they measure.

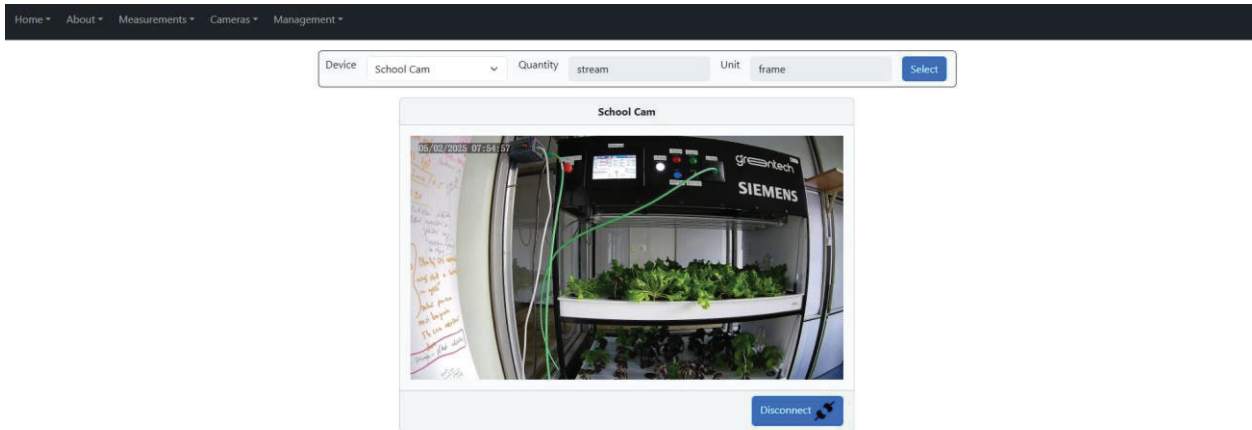


Figure 1: View for the farm observation.

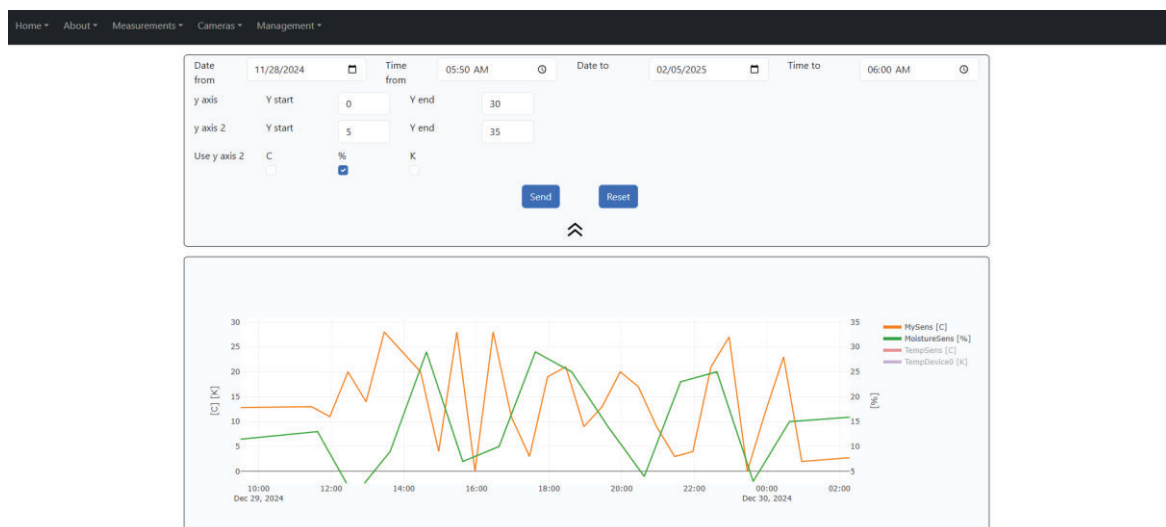


Figure 2: Data visualization using plotly.js and its control panel.

2.3 Deployment

The application is deployed on the Docker platform which enables running the application in the containers – an environment containing all dependencies and executables needed for the final application to run. That enables running the application independently on the host operating system which significantly simplifies the implementation process on the target server computer [8].

Every part - database, Python back-end and front-end have their own image which is specified in the Dockerfiles or in the Docker-compose file. The data are stored in volume, also defined by Dockerfile and stored on the host PC. Use of docker network allows to expose only the required parts like front-end port which significantly improves the security of the system. Due to the fact that front-end in Vue.js was built as client-side application, the important part of the deployment process was creating a proxy server which enables to call back-end from the context of browser. The ability of proxy server to hide its address is another step to improve the security of the server back-end. The deployment architecture is shown on the Figure 3.

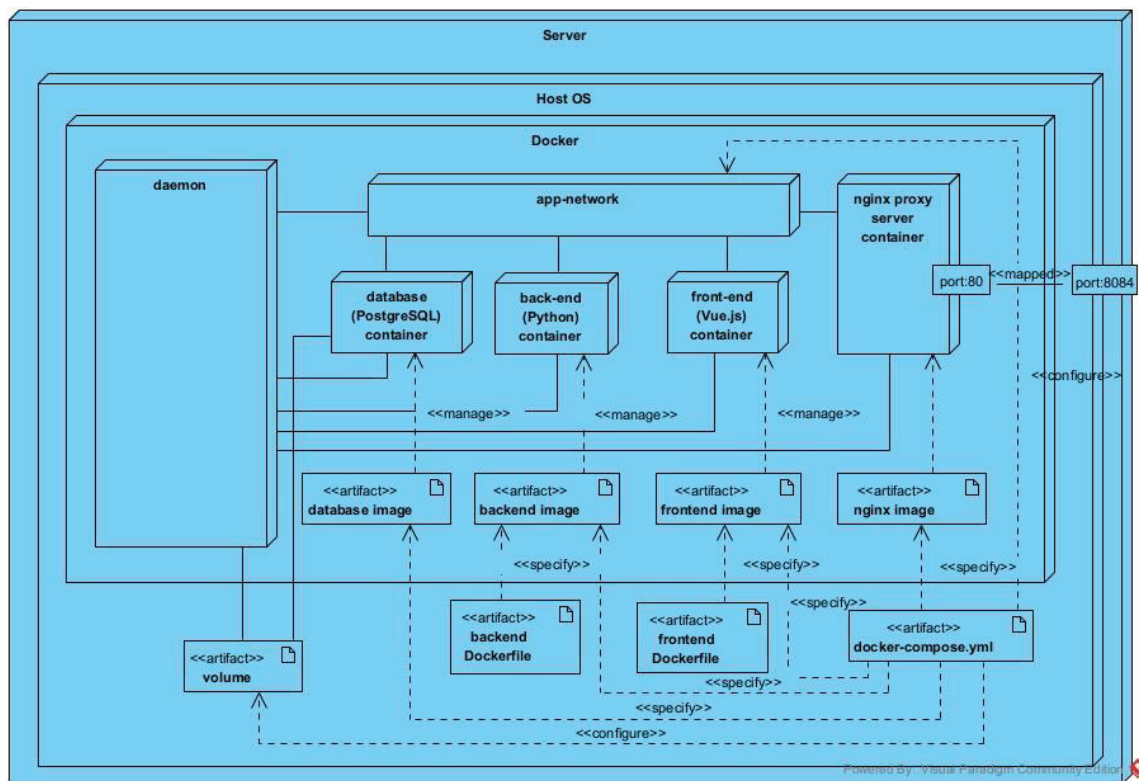


Figure 3: Deployment diagram on the server.

3 Conclusion

During the project a complex monitoring system has been developed. That included preparation of the database for the data retrieval, development of the connecting point to the database via API and building of monitoring web application enabling visualization of the data, observation and management of the connections. The use of Docker allows to implement application nearly immediately on the target device without the need to install and configure all dependencies needed for the database, back-end and front-end parts of the application.

4 References

- [1] S. Mandal, A. Yadav, F. A. Panme, K. M. Devi and S. Kumar S.M., "Adaption of smart applications in agriculture to enhance production," *Smart Agricultural Technology*, vol. 7, p. 11, March 2024.
- [2] R. Dagar, S. Som and S. K. Khatri, "Smart Farming – IoT in Agriculture," in *2018 International Conference on Inventive Research in Computing Applications*, Coimbatore, India, 2018.
- [3] Faculty of Mechanical Engineering, Czech Technical University in Prague, 18 03 2024. [Online]. Available: <https://www.fs.cvut.cz/aktuality/2555-212/optimalizujeme-algoritmus-pro-hydroponicke-pestovani/>. [Accessed 03 02 2025].
- [4] S. V. Salunke and A. Ouda, "A Performance Benchmark for the PostgreSQL and," *Future Internet*, vol. 16, p. 382, October 2024.
- [5] "ORM Mapped Class Overview," 09 01 2025. [Online]. Available: https://docs.sqlalchemy.org/en/20/orm/mapping_styles.html#orm-mapping-styles. [Accessed 04 02 2025].
- [6] B. Lubanovic, "FastAPI, Modern Python Web Development," Sebastopol, O'Reilly Media, Inc., 2023, p. 238.
- [7] "Single-File Components," [Online]. Available: <https://vuejs.org/guide/scaling-up/sfc>. [Accessed 05 02 2025].
- [8] A. Mouat, *Using Docker: Developing and Deploying Software with Containers*, O'Reilly Media, Inc., 2015, p. 354.

WEB INTERFACE FOR A AGRICULTURAL INFORMATION SYSTEM

*Matej Masata*¹

¹ *CTU in Prague, Faculty of Mechanical Engineering, Department of Instrumentation and Control Engineering, Technická 4, 160 00 Prague, Czech Republic*

Abstract: This article describes development of a web application for managing agricultural data from orchards. The goal was to create a secure web application and to extend the existing backend API. The frontend is created using the JavaScript library React. The identity provider Keycloak is used for user identity verification. The existing backend includes a PostgreSQL database and a Python micro-framework FastAPI to create the systems API.

Keywords: web application, agricultural data management, security, authentication, authorization, React, FastAPI, Keycloak, PostgreSQL

1 Introduction

Digitalization has an increasingly significant role in many sectors, and agriculture is no exception. Modern agricultural enterprises generate large amounts of data, the effective management of which is crucial for optimizing technological processes, planning, and for research purposes [1]. Web applications enable the collection of this data and provide secure access to it from any location.

This article describes the development and implementation of a web interface for an existing farm information system that previously lacked one. The main objective was to create a modern, secure, and user-friendly web application for managing data from fruit orchards. Emphasis was placed on three primary objectives: securing the entire system through authentication and authorization, extending the existing backend API to support these security features, and creating a responsive, dynamic frontend.

The JavaScript library React was chosen for frontend development due to its component-based architecture, vast ecosystem and popularity [2]. The existing backend is built on the FastAPI micro-framework and a PostgreSQL database. An important component for ensuring security is the identity provider Keycloak, which manages user authentication and authorization using industry standards like OpenID Connect and OAuth 2.0.

2 Security Concepts

Modern web application security relies on several core concepts and standards to protect data and manage user access. The concepts mentioned in this section are essential for development of the Agricultural web application

2.1 Authentication and Authorization

Though often used interchangeably, authentication and authorization are distinct processes [3].

- **Authentication** is the process of verifying a user's identity. It answers the question, "Who are you?". This can be achieved through credentials like a username and password or biometric information (such as fingerprints, voice or facial features) [4].
- **Authorization** is the process of granting a verified user permission to access specific resources or perform certain actions. It answers the question, "What are you allowed to do?". Authorization always follows successful authentication [5].

2.2 JSON Web Tokens (JWT)

A JSON Web Token (JWT) is an industrial standard (RFC 7519) for securely transmitting information between entities (usually client and server) as a JSON object. This information can be verified and trusted because it is digitally signed and encrypted. The token contains all the necessary information about the user (such as user ID and roles), eliminating the need for the server to maintain session state. A JWT consists of three parts: a header, a payload, and a signature [6].

2.3 OAuth 2.0 and OpenID Connect (OIDC)

- **OAuth 2.0** is an industry-standard framework (RFC 6749) for authorization. It allows a third-party application to obtain limited, delegated access to a user's resources on another service without exposing the user's credentials. It defines several flows for obtaining an **Access Token**, which represents the permission to access the specified resources [7].
- **OpenID Connect (OIDC)** is an identity layer built on top of the OAuth 2.0 framework. It allows clients to verify the identity of the end-user based on the authentication performed by an Authorization Server. OIDC introduces the concept of an **ID Token**, which is a JWT that contains claims about the user's authentication event [8].

3 Original architecture of the application

The development began with an existing backend system that provided foundational data management capabilities but lacked a user interface and a security layer. It consisted of two main parts - API and database.

3.1 API

The application layer of the app (backend) was built using the Python micro-framework FastAPI. It featured a modular structure with individual routers for handling API endpoints, services for business logic, and Pydantic schemas for data validation and serialization. Communication with the database was managed through the SQLAlchemy ORM (*Object Relational Mapper*). While functional for basic CRUD (*Create, Read, Update, Delete*) operations, the API did not have any authentication or authorization mechanisms, leaving all endpoints unsecured.

3.2 Database

The data layer consists of a PostgreSQL database. The schema is managed by Alembic, a database migration tool for SQLAlchemy. This ensures that the database structure is version-controlled and could be applied consistently across different environments. The database stores information about orchards, trees within the orchards and additional data about each tree, it also stores some additional orchard related information, for example the names of the agents used for sprayings (Fig. 1).

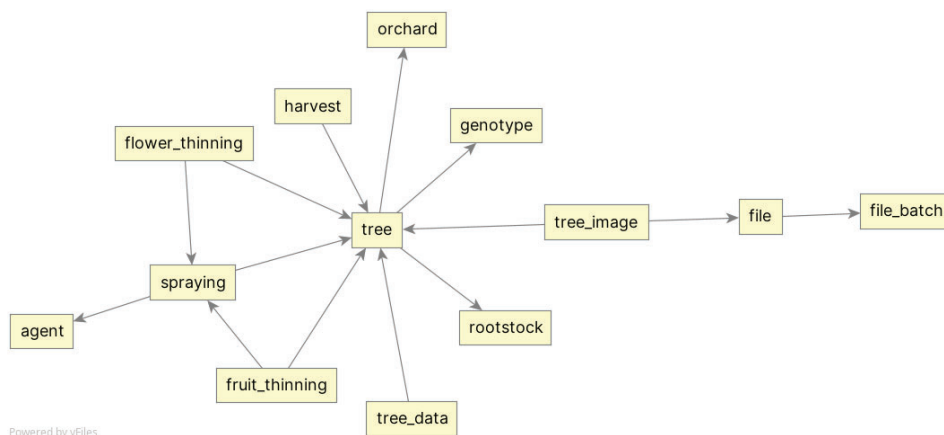


Fig. 1: High level overview of the database tables

4 Implementation

4.1 Identity provider

Security is centralized using **Keycloak**, an open-source Identity and Access Management solution. It supports a wide range of industry standards including OAuth 2.0 and OpenID Connect [9]. Keycloak handles tasks such as user registration, login, and token issuance. It is configured to support the specific needs of the application, including custom clients and roles. Its data is stored inside the same PostgreSQL server, but in separate database from the Agricultural data.

4.2 Securing the API

The security of the backend was implemented directly within the FastAPI application using its dependency injection system. This ensures that every API request to a protected endpoint undergoes a validation and authorization process before any business logic is executed.

A Role-Based Access Control (RBAC) system was designed to control data access:

- **Orchard-Global-Admin:** A superuser role with full permissions across all orchards.
- **Orchard-{ID}-Admin:** An administrative role for a specific orchard. Users with this role can modify data only within that orchard.
- **Orchard-{ID}-View:** A read-only role for a specific orchard.

To ensure the access control system remains synchronized with the application's data, the backend API was extended with logic to automatically create and delete these orchard-specific roles in Keycloak whenever an orchard is created or deleted.

For each incoming request, a security dependency is triggered to:

1. **Extract and Validate JWT:** The Access Token is extracted from the `Authorization` header. The system then validates the token's signature against Keycloak's public key, verifies its expiration date, and confirms that the token was issued for this specific client. If the token is invalid the API returns `401 Unauthorized`.
2. **Extract User Permissions:** Upon successful validation, the user's roles are extracted from the token's payload.
3. **Enforce Endpoint Authorization:** The extracted roles are then checked against the required permissions for the specific endpoint. If the user lacks the necessary role, the API immediately returns a `403 Forbidden` error, preventing any unauthorized access. For some endpoints additional filtration based on the user roles is needed to filter the data the user has access to. This is done in the services files.

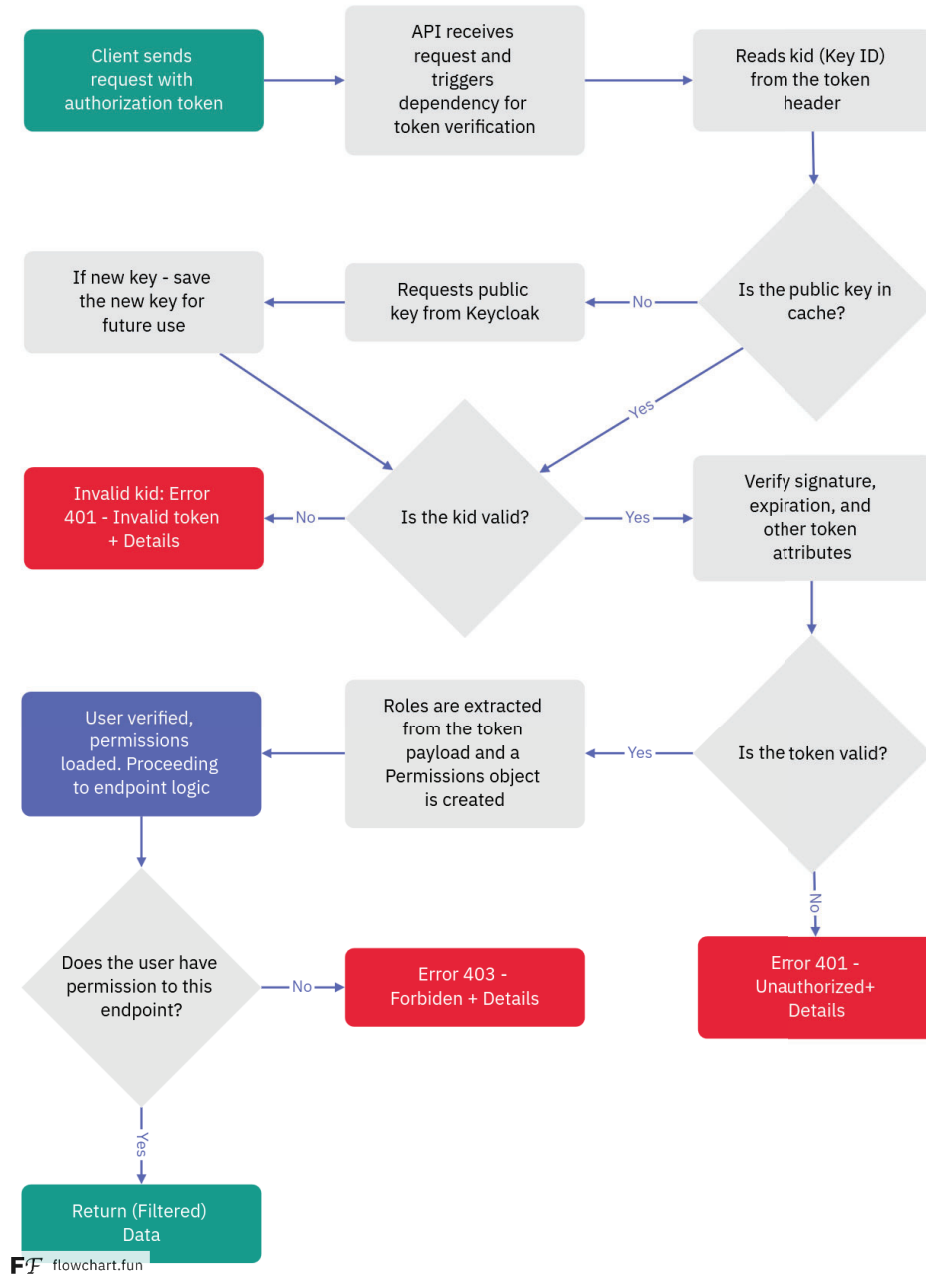


Fig. 2: Chart illustrating some parts of the API's authentication and authorization flow

4.3 Frontend

The user interface is built with **React**. It's responsible for all data presentation and user interaction. Key libraries include *React Router* for client-side navigation, *TanStack Query* for managing server state, which simplifies data fetching, caching, and synchronization with the UI and *keycloak-js*. Styling is handled with CSS Modules to ensure component style encapsulation.

4.3.1 Authentication Flow

User authentication is handled via the OpenID Connect (OIDC) protocol using the Authorization Code Flow, which is the most secure method for web applications [8].

1. The user initiates login in the React application, which redirects them to the Keycloak login page.
2. The user authenticates with their credentials directly against Keycloak.

3. Upon successful authentication, Keycloak redirects the user back to the application with a single-use authorization code.
4. The React application exchanges this code for an ID Token and an Access Token from Keycloak's token endpoint.

The Access Token (JWT) is then included in the `Authorization` header of every request to the backend API to prove the user's permissions.

4.3.2 Dynamic UI Components

Access control is visually enforced by conditionally rendering UI elements. For example, buttons for creating, editing, or deleting data are only rendered and visible to users with the correct administrative roles (e.g., *Orchard-Global-Admin* or *Orchard-{ID}-Admin*). Users with view-only permissions (*Orchard-{ID}-View*) see a read-only version of the interface with the action buttons hidden. This provides an intuitive user experience (UX) where users are presented with actions they are permitted to perform.

4.3.3 API Communication

A service layer was created to handle all communication with the backend API. A base request function ensures that the JWT Access Token is automatically attached to the `Authorization` header of every outgoing request. This approach simplifies data-fetching logic within the individual components.

4.4 Deployment

The entire application is containerized using Docker to ensure consistency and portability across different environments. The multi-container setup is managed by Docker Compose. A single configuration file, `docker-compose.yml`, defines all the services required for the application to run:

- **fms-postgres:** A PostgreSQL database instance for storing both application and Keycloak data.
- **keycloak:** The Keycloak identity provider service. It automatically imports a predefined realm configuration from a JSON file on startup.
- **api:** The FastAPI backend service.
- **frontend:** An Nginx web server that serves the built React application.
- **mailhog:** A development SMTP server for testing email functionalities.

This setup allows the entire application stack to be launched with a single command (`docker compose up -d --build`), which simplifies the deployment and guarantees that every developer works with an identical environment.

All the necessary files can be found in this Github repository https://github.com/MatejMasata/DP_FullStack

5 Conclusion

This article describes some parts of the development of a secure and modern web interface for an existing agricultural information system. The implementation of a identity provider Keycloak and a RBAC model ensures that data access is strictly controlled based on user roles. The resulting application provides a user-friendly and responsive interface that provides a solid foundation for future development.

Literature

- [1] Eli K Moore, Adam Kriesberg, Steven Schroeder, Kerrie Geil, Inga Haugen, Carol Barford, Erica M Johns, Dan Arthur, Megan Sheffield, Stephanie M Ritchie, et al. Agricultural data management and sharing: Best practices and case study. *Agronomy Journal*, 114(5):2624–2634, 2022.
- [2] npmtrends.com. Comparison of npm download trends: Angular vs. react vs. svelte vs. vue. <https://npmtrends.com/@angular/core-vs-react-vs-svelte-vs-vue>. Accessed: 2025-08-11.

- [3] Auth0. Authentication vs. authorization. <https://auth0.com/docs/get-started/identity-fundamentals/authentication-and-authorization>. Accessed: 2025-07-25.
- [4] Piyush Pant, Anand Singh Rajawat, S. B. Goyal, Pradeep Bedi, Chaman Verma, Maria Simona Raboaca, and Florentina Magda Enescu. Authentication and authorization in modern web apps for data security using nodejs and role of dark web. *Procedia Computer Science*, 215:781–790, 2022. 4th International Conference on Innovative Data Communication Technology and Application.
- [5] Okta. Authentication vs. authorization. <https://www.okta.com/identity-101/authentication-vs-authorization/>, 2024. Accessed: 2025-07-25.
- [6] Mostafa Ibrahim. What is a jwt? understanding json web tokens. <https://supertokens.com/blog/what-is-jwt>, 2024. Accessed: 2025-07-25.
- [7] D. Hardt. The OAuth 2.0 authorization framework. RFC 6749, Internet Engineering Task Force, 10 2012.
- [8] Nat Sakimura, John Bradley, Michael B. Jones, Breno de Medeiros, and Chuck Mortimore. Final OpenID Connect Core 1.0 incorporating errata set 2. https://openid.net/specs/openid-connect-core-1_0.html, 12 2023. Accessed: 2025-07-25.
- [9] Keycloak Authors. Keycloak guides. <https://www.keycloak.org/guides>. Accessed: 2025-07-25.

DDW MODEL IDENTIFICATION

*Petr Valenta*¹

*CTU in Prague Faculty of Mechanical Engineering, Department of Instrumentation and Control Engineering,
valenp11@cvut.cz*

Abstrakt: Tento článek se zaměřuje na identifikaci a modelování bubnových praček (DDW) používaných v procesech praní buničiny. Nastiňuje základy praní buničiny a zdůrazňuje potřebu účinného odstraňování ligninu a kontaminantů za účelem zlepšení kvality produktu a snížení spotřeby chemických látek. Systém DDW, který funguje na principu vytěšňovacího praní, je představen jako kompaktní a účinné řešení pro vícestupňové čištění. Práce dále modeluje dynamické chování systému DDW pomocí přístupu ARMAX, který zachycuje jak vstupně-výstupní vztahy, tak charakteristiky šumu. Výsledný model je ověřen na základě naměřených dat a prokazuje vysokou přesnost předpovědi pro většinu výstupů systému. Analýza ukazuje, že modelování metodou ARMAX je vhodné pro charakterizaci procesu DDW a může být použito k optimalizaci výkonu systému v průmyslových aplikacích.

Klíčová slova: Praní buničiny, Drum Displacer Washer, dynamický systém, model ARMAX, identifikace systému

Abstract: This paper focuses on the identification and modeling of the Drum Displacer Washer (DDW) used in pulp washing processes. It outlines the fundamentals of pulp washing, emphasizing the need for effective removal of lignin and contaminants to improve product quality and reduce chemical consumption. The DDW system, which operates on the principle of displacement washing, is introduced as a compact and efficient solution for multi-stage cleaning. The study models the dynamic behavior of the DDW system using the ARMAX approach, which captures both input-output relationships and noise characteristics. The resulting model is validated against measured data, demonstrating high predictive accuracy for most system outputs. The analysis shows that ARMAX modeling is suitable for characterizing the DDW process and can be used to optimize system performance in industrial applications.

Keywords: Pulp washing, Drum Displacer Washer, dynamic system, ARMAX model, system identification

1 Introduction

The pulp at this point is already cooked and contains no solids. However, it still contains lignin dissolved in water, which is undesirable in terms of the quality of the final product. Lignin causes the pulp to discolour because lignin is optically active, so it must be removed. For this purpose, washing technology is used to separate the lignin and other impurities. If we have washing between bleaching steps, the main purpose of washing is to save chemicals. The different bleaching steps have different pH and use different chemicals [1]. If the pulp was not washed well, a large amount of chemical would be used to neutralize the chemical from the previous bleaching step.

The separated lignin is not used as waste, but is used to generate energy, which is then used in the steam production process [1]. This step is important not only from an economic point of view, but also in terms of sustainability and reducing the environmental impact of the entire production process.

The main objective in the development of the pulp washing technology was and still is to achieve maximum savings in chemicals and water. The water and chemicals used in the process are often recycled and reused, minimising waste and reducing negative environmental impacts. The water passing through the process is not discharged immediately but is reused and reused at different stages of production, contributing to the economic efficiency of the whole process.

There are several different methods for washing pulp, which differ in the way water and impurities are removed. Each of these methods has its specific advantages and is suitable for different stages of the production process.

The dewatering washing method is based on the mechanical removal of water under increased pressure. For this purpose, presses are used which consist of perforated cylinders. These cylinders compress the pulp and at the same time extract the excess water [2]. The result of this process is a thickening of the pulp, which becomes more compact. However, after pressing, the pulp has to be ground again in order to be able to work with it further. This is why the dewatering process is followed by a tearing and thinning process [1]. Thinning is a key step as it ensures that the pulp remains thin enough to be pumped through the pipeline for further processing. Without this step, it would be difficult to ensure its efficient transport.

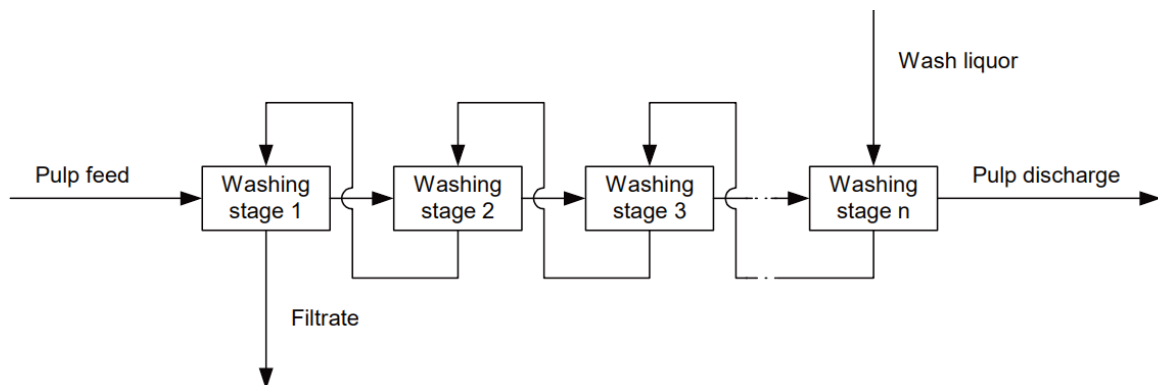


Figure 1: Counter-current principal [2]

The displacement washing method works on the principle of exchanging dirty water for clean water. The water is fed into the process in large quantities and gradually washes away the contaminants. In contrast to the dewatering method, the consistency of the pulp is not changed - its structure remains the same [3]. However, this method requires a significant amount of water, which can be a disadvantage, especially in terms of natural resource consumption and wastewater treatment costs.

In practice, these two basic washing methods are often combined. In fact, the washing of pulp does not take place in a single step but in several stages, each of which may be carried out in a different way [2]. In addition, the principle of counter-current washing is often used between the washing steps. This system is designed to minimise water consumption and maximise the efficiency of the process. Counter-current washing consists of introducing the cleanest water into the last stage of the wash, where the pulp is already almost clean. The filter water from this stage is then gradually used in the previous washing stages, where the less clean pulp is processed [2]. This principle is evident in the Figure 1.

This principle results in significant water savings because the water is not immediately discharged but is effectively recycled. This not only reduces production costs, but also contributes to a more sustainable use of natural resources and reduces the ecological footprint of the entire pulp production process.

1.1 Drum Displacer Washer

The Drum Displacer Washer (DDW) is a specialised pulp washing machine that enables efficient and economical multi-stage cleaning within a single system. Up to four stages of washing can take place in this equipment, which greatly increases its efficiency and reduces the water and chemical consumption required for the washing process [3]. The principle of the four stages of washing is shown in the Figure 2.

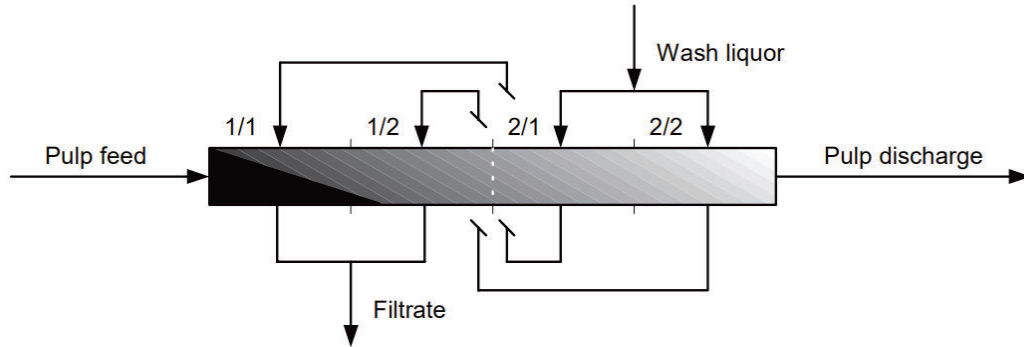


Figure 2: Multi-stage washing principal [2]

The DDW uses displacement washing, which means that dirty water is gradually replaced by cleaner water without the need for a full discharge, helping to optimize water consumption and minimize waste.

The basic design element of this device is a hollow perforated cylinder that rotates slowly. During this movement, the cylinder moves the pulp between the individual shower nozzles, which flush it intensively. A slight vacuum is maintained in the cylinder to help effectively remove water from the pulp, thereby effectively drying it and reducing its dirt content [2]. The water from the nozzles that is used for rinsing is collected and subsequently recycled in the next stages of the process [3]. This recirculation system contributes significantly to the economy and environmental sustainability of the operation.

Compared to other multi-stage washers, the Drum Displacer Washer has one significant advantage - its compact design allows for significant space savings, which is especially crucial in operations with limited space [2]. Thanks to its innovative technology, the DDW is an efficient solution for industrial pulp washing with optimised operating costs.

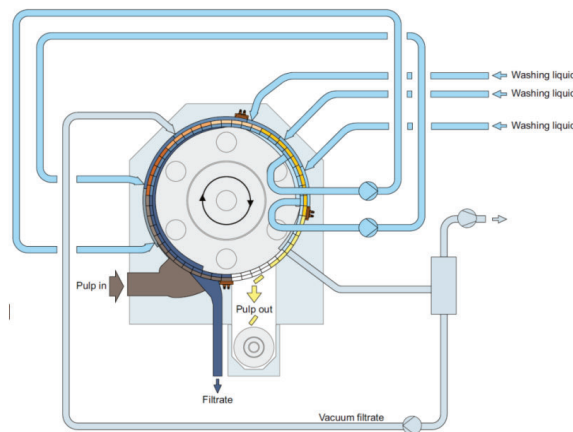


Figure 3: DDW scheme [4]

The Figure 3 shows that the main input is pulp in, and the other inputs are washing targets. It can also be seen how water is collected and reinjected into the DDW. The outputs of the DDW are clean pulp and filtrate. The inputs that have been mentioned will be used in the following sections to describe the static and dynamic behaviour of DDW.

2 Dynamic model of DDW2

Dynamic systems are essential in many fields, as they describe how systems change over time. These systems appear in both natural environments and human-made structures, helping us interpret, predict, and influence real-world behavior. They are typically classified by characteristics such as linearity, randomness, and whether they operate in continuous or discrete time [5]. Mathematical tools like differential and difference equations are used to model how these systems evolve.

This paper will focus on a brief description of the ARMAX method and its subsequent use to identify a dynamic model of the DDW device.

3 ARMAX model

The ARMAX (AutoRegressive Moving Average with eXogenous input) model is an extension of the basic ARX model to include a part that models noise using a moving average. This model is suitable for cases where the noise in the system is autocorrelated, i.e. does not exhibit white noise properties [6]. By adding the MA (moving average) part, the structured effect of random disturbances on the system output can be better captured.

The general form of the discrete linear ARMAX model can be written in back-sliding (regression) form as follows [7]:

$$y[k] = - \sum_{i=1}^n a_i \cdot y[k-i] + \sum_{j=1}^m b_j \cdot u[k-j] + \sum_{i=1}^l c_i \cdot \epsilon[k-i] \quad (1)$$

$$A(q^{-1}) \cdot y[k] = B(q^{-1}) \cdot u[k] + C(q^{-1}) \cdot \epsilon[k] \quad (2)$$

Matrices A and B behave in the same way in the ARMAX model as in the ARX model - they describe the dynamics of the system in the relationship between output and input. Specifically, matrix A represents the autoregressive part of the model, i.e. the dependence of the current output on its previous values, while matrix B describes the influence of the input variable on the output of the system [7]. A new matrix also appears in the ARMAX model C , which represents the so-called moving average part and is used to model the noise dynamics. This component allows to capture the correlated noise at the output [7], leading to a more accurate model, especially when the disturbance is of a structured (time-dependent) nature. These matrices are shown in the diagram in Figure 4, and their use is also illustrated.

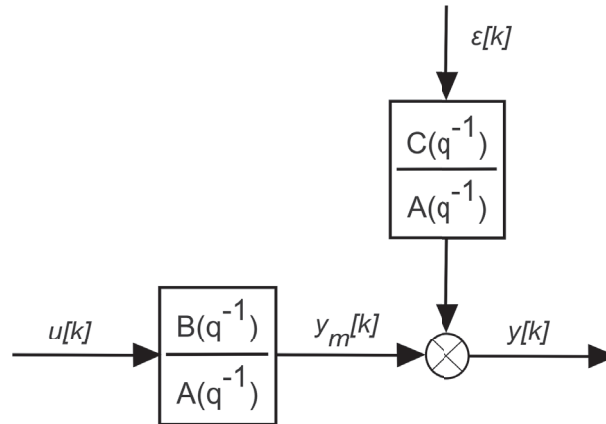


Figure 4: ARMAX model diagram

The fourth order of the system was chosen for the ARMAX method. For matrix B , the second order was tested first, but this did not lead to a successful result. Therefore, the matrix B is finally a combination of second and fourth order.

The resulting matrices look as follows:

$$A = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 2 & 1 \\ 4 & 2 & 0 \\ 4 & 4 & 2 \end{bmatrix}, \quad C = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$$

The identification of the ARMAX model was performed in MATLAB using the `armax()` function from the System Identification Toolbox, which accepts matrices A , B and C . The conversion of the model to the transfer functions was performed using the `tf()` function.

The ARMAX model provided results in the form of a discrete mathematical model from which the individual transfer functions of the system were subsequently derived. These transfer functions are expressed in detail in Equations 3 - 11. A notable exception is equation G_{23} , which was set to zero. The reason for this decision is that the analysis did not identify any measurable dependence between the input variable MV20 and the output variable CV22. The other transfer functions were determined in a standard manner and reflect the expected dynamic behavior of the system.

$$G_{11} = \frac{CV21}{MV18} = \frac{-0.0014z^{-1} + 0.0014z^{-2}}{1 - 2.2z^{-1} + 0.66z^{-2} + 1.1z^{-3} - 0.65z^{-4}} \quad (3)$$

$$G_{12} = \frac{CV21}{MV19} = \frac{-0.00065z^{-1} + 0.00065z^{-2}}{1 - 2.2z^{-1} + 0.66z^{-2} + 1.1z^{-3} - 0.65z^{-4}} \quad (4)$$

$$G_{13} = \frac{CV21}{MV20} = \frac{-1.3 \times 10^{-7}z^{-1}}{1 - 2.2z^{-1} + 0.66z^{-2} + 1.1z^{-3} - 0.65z^{-4}} \quad (5)$$

$$G_{21} = \frac{CV22}{MV18} = \frac{0.0051z^{-1} + 0.00039z^{-2} - 0.0055z^{-3} - 2.7 \times 10^{-5}z^{-4}}{1 + -z^{-1} - 0.5z^{-2} + 0.0013z^{-3} + 0.5z^{-4}} \quad (6)$$

$$G_{22} = \frac{CV22}{MV19} = \frac{0.0043z^{-1} - 0.0043z^{-2}}{1 + -z^{-1} - 0.5z^{-2} + 0.0013z^{-3} + 0.5z^{-4}} \quad (7)$$

$$G_{23} = \frac{CV22}{MV20} = 0 \quad (8)$$

$$G_{31} = \frac{CV23}{MV18} = \frac{-0.0022 - 0.00043z^{-1} + 0.0074z^{-2} - 0.0048z^{-3}}{1 - 2.2z^{-1} + 0.75z^{-2} + 1.2z^{-3} - 0.74z^{-4}} \quad (9)$$

$$G_{32} = \frac{CV23}{MV19} = \frac{-0.0026 + 0.004z^{-1} - 0.00025z^{-2} - 0.0012z^{-3}}{1 - 2.2z^{-1} + 0.75z^{-2} + 1.2z^{-3} - 0.74z^{-4}} \quad (10)$$

$$G_{33} = \frac{CV23}{MV20} = \frac{-6.3 \times 10^{-7} + 6.3 \times 10^{-7}z^{-1}}{1 - 2.2z^{-1} + 0.75z^{-2} + 1.2z^{-3} - 0.74z^{-4}} \quad (11)$$

Figure 5 compares the model values with the measured data. It can be seen that the model predicts the CV value very well. The model performs best for CV23, where the MSE value is equal to 1.8677. The model for CV22 came out the worst. This may be due to some unmonitored fault that is present in the system, here it would be good to look more into the process. But despite the minor discrepancies, the model follows the trends very well.

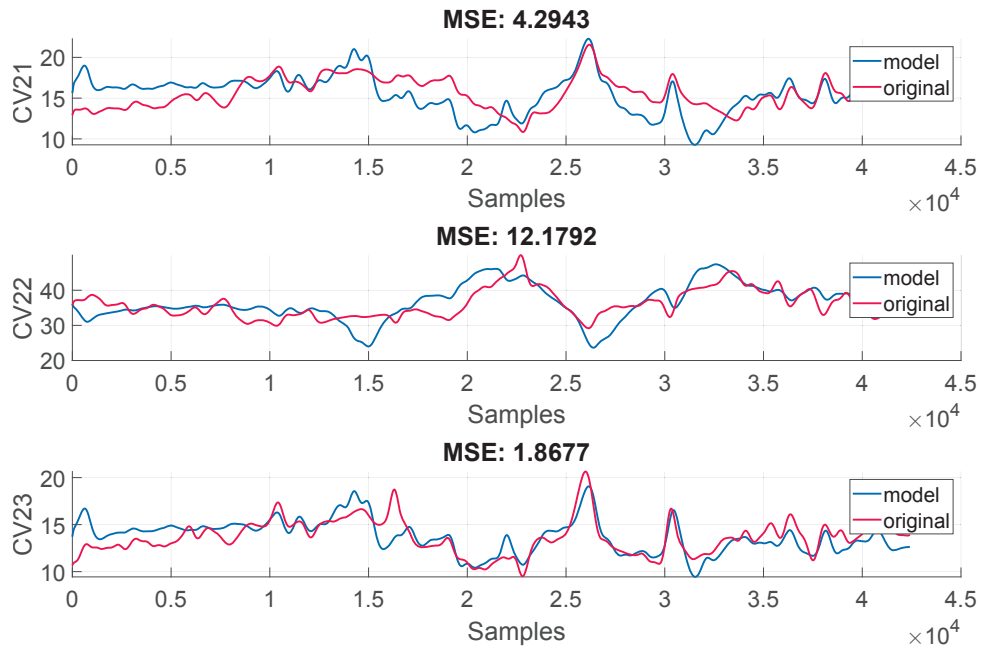


Figure 5: Validation of model ARMAX

4 Conclusion

This study presents a successful identification of a dynamic model for the Drum Displacer Washer (DDW) system using the ARMAX method. By capturing both the deterministic dynamics and the stochastic noise components, the model offers a realistic and predictive framework for describing system behavior. Validation against real process data shows high accuracy, especially for the CV23 output. Although discrepancies were noted for CV22, these are likely due to unobserved system disturbances. Overall, the ARMAX-based model proves effective for optimizing DDW operations, contributing to more efficient, cost-effective, and environmentally sustainable pulp washing processes.

The results and conclusions presented in this article are based on the author's master's thesis [8], which addresses the topic in greater depth. More detailed theoretical background, methodology, and extended data analysis can be found in the full version of the thesis.

References

- [1] ANDRITZ Pulp & Paper. Internal training: Washing line. Internal training on the washing line, Mondi Štětí, January 2018.
- [2] Herbert Sixta. *Handbook of Pulp*. Wiley-VCH, 2nd edition, 2020.
- [3] Monica Ek, editor. *Pulp and Paper Chemistry and Technology, Vol. 2: Pulping Chemistry and Technology*. De Gruyter, Berlin, Germany, 2009.
- [4] Pasi Immonen. Evolution of the andritz dd washer. *SPECTRUM*, (36):46–47, 2017.
- [5] Milan Hofreiter. *Identifikace systémů I*. České vysoké učení technické v Praze, Praha, Czech Republic, 2009.
- [6] James D. Hamilton. *Time Series Analysis*. Princeton University Press, Princeton, NJ, 1994.
- [7] Roberto Diversi, Roberto Guidorzi, and Umberto Soverini. Identification of armax models with noisy input and output. *IFAC Proceedings Volumes*, 44(1):13121–13126, 2011. 18th IFAC World Congress.
- [8] Petr Valenta. Drum displacement washing machine model identification. Master thesis, CTU Prague, Prague, 2025. Supervisor: Jaroslav Bušek.

MATEMATICKÝ MODEL KVADROKOPTÉRY SE ZAVĚŠENÝM ZÁVAŽÍM VE 3D

Valerii Lisnychuk¹

¹ České vysoké učení technické v Praze, Fakulta strojní, Ústav přístrojové a řídicí techniky,
Valerii.Lisnychuk@fs.cvut.cz

Abstrakt: Tento článek se zabývá odvozením pohybových rovnic soustavy kvadrokoptéry se zavěšeným závažím ve 3D prostoru, které byly následně použity k analýze a modelování systému a k návrhu různých algoritmů řízení kvadrokoptéry v rámci mé diplomové práce. Odvození proběhlo na základě Eulerovy-Lagrangeovy rovnice druhého druhu a bylo symbolicky zpracováno v MATLAB pomocí Symbolic Math Toolbox. Dále byla provedena linearizace systému a byl odvozen lineární State-Space model. Na konci uvádím odezvy nelineárního matematického modelu na různé hodnoty otáček motorů s cílem ověřit, zda chování kvadrokoptéry se závažím odpovídá stanoveným požadavkům.

Klíčová slova: Kvadrokoptéra se zavěšeným závažím, rovnice pohybu kvadrokoptéry, Eulerova-Lagrangeova rovnice druhého druhu, MATLAB, linearizace, State-Space.

1. Úvod

V posledních letech zaznamenaly bezpilotní letecké prostředky (UAV – Unmanned Aerial Vehicles), zejména kvadrokoptéry, značný rozvoj nejen v oblasti zábavy, ale i v průmyslových a vědeckých aplikacích. Stále častěji se uplatňují také i v armádních a bezpečnostních operacích, například v průzkumu, sledování nebo při zásazích speciálních jednotek. Významný impuls k jejich vývoji a využívání poskytl ozbrojený konflikt na Ukrajině, kde se UAV staly klíčovým prvkem moderního bojiště. Schopnost kvadrokoptéry manévrovat v omezeném prostoru, být ve stavu visení, přistávat na malých plochách a nést různé druhy nákladu otevírá nové možnosti využití, například v logistice, záchranných operacích při přírodních katastrofách, fotografování atd. Proto vytvoření matematického modelu je klíčovým krokem k efektivnímu řízení takových systémů. Tento článek se zaměřuje právě na odvození modelu kvadrokoptéry se zavěšeným závažím ve 3D prostoru. Model je součástí mé diplomové práce, kde slouží jako základ pro návrh a testování různých řídicích algoritmů.

2. Odvození

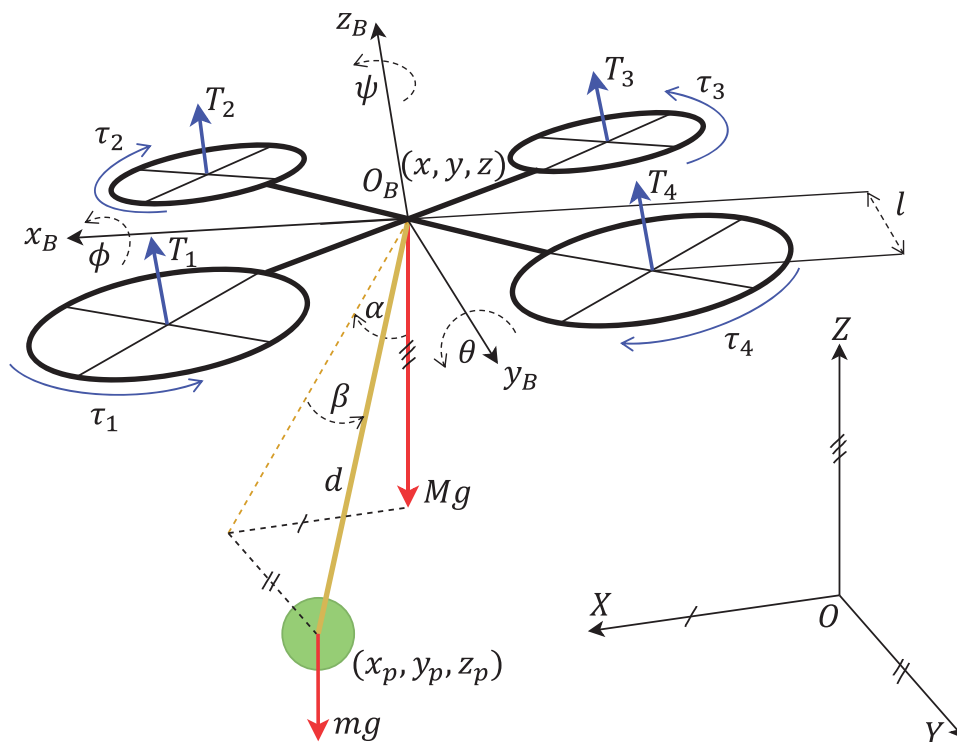
2.1 Popis systému

Systém kvadrokoptéry (též dron) o hmotnosti M se zavěšeným závažím o hmotnosti m a délkou lana d je zobrazen na Obr. 1. Rám kvadrokoptéry je považován za tuhý a symetrický. Pro popis pohybu jsou zavedeny dva souřadnicové systémy. Prvním je inerciální systém $O = \{X, Y, Z\}$, také globální souřadnicový systém, jehož osa Z je svislá a kolmá k zemi a osa X směřuje na sever. Druhým je souřadnicový systém tělesa $O_B = \{x_B, y_B, z_B\}$, který je umístěn v hmotném středu dronu a je s ním pevně spojen. Na rozdíl od jiných prací [[1], [2], [3]], ve kterých jsou osy soustavy tělesa shodné s osami rámu

kvadrokoptéry, tj. v konfiguraci “plus”, používám konfiguraci “X”, tzn. jak je vidět z obrázku, osy x_B a y_B jsou umístěny mezi vrtulemi.

Kvadrokoptéra má čtyři vrtule, každá z nich generuje tahovou sílu T_i a odporový moment τ_i vlivem odporu prostředí. Aby se zabránilo rotaci kvadrokoptéry kolem osy z_B a vyrovnal se moment, protiběžné motory se otáčejí v opačných směrech. Konkrétně motory 1 a 3 se otáčejí proti směru hodinových ručiček, zatímco motory 2 a 4 se otáčejí ve směru hodinových ručiček.

Pro pohyb kvadrokoptéry požadovaným směrem je nutné měnit otáčky příslušných motorů. Pro zvednutí/snížení je nutné, aby se rychlost každého ze čtyř motorů zvyšovala/snižovala o stejné hodnoty. Pokud se rychlost motorů 1 a 2 sníží a rychlost motorů 3 a 4 se zvýší, dojde k natočení kolem osy y_B o úhel *pitch* (θ) a zároveň k pohybu ve směru X . Obdobně, pokud se zvýší rychlost motorů 1 a 4 a sníží se rychlost motorů 2 a 3, dojde k natočení kolem osy x_B o úhel *roll* (ϕ) a zároveň k pohybu ve směru Y . A konečně, pokud se rychlost motorů 1 a 3 zvýší a rychlost motorů 2 a 4 se sníží o stejné hodnoty, dojde k natočení kolem osy z_B o úhel *yaw* (ψ). V počátečním stavu se předpokládá, že orientace souřadnicového systému tělesa O_B a globálního souřadnicového systému O je stejná (tj. odpovídající osy jsou rovnoběžné a úhly jsou nulové), přičemž translační poloha systému O_B vůči systému O může být libovolná. Tato orientace dronu je definovaná jako počáteční a bude označena jako $O_{B_0} = \{x_{B_0}, y_{B_0}, z_{B_0}\}$.



Obr. 1 Schéma kvadrokoptéry se zavěšeným závažím [autor]

Ke středu hmotnosti kvadrokoptéry je pomocí nepružného a tuhého lana připevněno závaží. Jeho poloha v prostoru je dána úhly α a β , které vyjadřují vychýlení závazí vzhledem k počáteční orientaci O_{B_0} . Jinými slovy, při jakékoli orientaci dronu jsou úhly závazí vyjádřeny vzhledem k jeho počáteční orientaci. Úhel α je úhel mezi vektorem tíhy kvadrokoptéry a projekcí lana v rovině XZ . Úhel β je úhel mezi samotným lanem a projekcí lana v rovině XZ . Zvolené úhly lepe popisují kmitání závazí a nezpůsobují problémy při linearizaci systému než klasické sférické souřadnice v [4].

Poloha těžiště kvadrokoptéry v inerciální soustavě je dána vektorem $\xi = [x, y, z]^T$, orientace se popisuje přes Eulerovy úhly a je reprezentovaná vektorem $\eta = [\phi, \theta, \psi]^T$ (tj. *roll*, *pitch*, *yaw*), nakonec vychýlení závazí je dáno vektorem $\mu = [\alpha, \beta]^T$. Tím pádem je vektor zobecněných souřadnic označen jako $q = [x, y, z, \phi, \theta, \psi, \alpha, \beta]^T$.

2.2 Popis transformačních matic

Popis pohybu a rotace tělesa ve 3D prostoru je mnohem složitější než popis ve 2D. S přidáním další dimenze a os se zvyšuje složitost, zejména u rotačního pohybu. Je nutné zohledňovat pořadí rotací, typ rotace (vnitřní, vnější) a další faktory, které ovlivňují dynamiku pohybu. Proto uvádím popis matic, které jsou nezbytné pro popis pohybu kvadrokoptéry v prostoru.

První důležitou transformační maticí pro daný systém je rotační matice $\mathbf{R}_{zyx}$, která provádí transformaci ze soustavy tělesa do inerciální soustavy (z lokálního do globálního). Předpokládá se pořadí elementárních vnitřních rotací z - y' - x'' , kde každá následující rotace se provádí vzhledem k ose nové otočené souřadnicové soustavy tělesa. To znamená, že první rotace o úhel ψ (*yaw*) se provádí kolem osy z_B původní soustavy tělesa, druhá rotace o úhel θ (*pitch*) se provádí kolem osy y'_B , která byla otočena po první rotaci, a třetí rotace o úhel ϕ (*roll*) kolem osy x''_B , která byla otočena po druhé rotaci. Výsledná rotační matice je definovaná jako součin elementárních rotačních matic kolem zmíněných os. Pro zjednodušení zápisu se používají zkratky $s(x) = \sin(x)$, $c(x) = \cos(x)$, $t(x) = \tan(x)$.

$$\begin{aligned} \mathbf{R}_{zyx} &= \mathbf{R}_z(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_x(\phi) = \\ &= \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \end{aligned} \quad (1)$$

Druhou transformační maticí je speciální matice $\mathbf{W}_{zyx}$, která udává vztah mezi vektorem úhlové rychlosti v soustavě tělesa $[p, q, r]^T$ a rychlostí změny Eulerových úhlů $[\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ popisujících orientaci tělesa v inerciální soustavě [[2], [3], [5]]. Matice se určí postupným promítáním derivací Eulerových úhlů z jejich odpovídajících souřadnicových systémů do souřadnicového systému kvadrokoptéry s ohledem na sekvenční charakter rotace popsany dříve.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{W}_{zyx} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & c(\theta)s(\phi) \\ 0 & -s(\phi) & c(\theta)c(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (2)$$

2.3 Rovnice sil kvadrokoptéry

Každý motor kvadrokoptéry generuje tahovou sílu T_i podél osy z_B a odporový moment τ_i kolem osy z_B v odpovídajícím směru (otáčení proti směru hodinových ručiček je považováno za kladný směr). Z aerodynamických zákonů a vlastností vrtule síla a moment jsou úměrné druhé mocnině úhlové rychlosti motoru ω_i přes tahový koeficient k a odporový koeficient b . Pokud předpokládáme konstantní hodnoty koeficientů a že motory a vrtule mají stejné vlastnosti, pak se vektor tahové síly $\mathbf{T}_B$ a vektor točivého momentu $\boldsymbol{\tau}_B$ působící v soustavě tělesa rovnají:

$$\mathbf{T}_B = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}, \quad T = \sum_{i=1}^4 T_i = k \cdot \sum_{i=1}^4 \omega_i^2 \quad (3)$$

$$\boldsymbol{\tau}_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l \cdot (T_1 + T_4 - T_2 - T_3) \\ l \cdot (T_3 + T_4 - T_1 - T_2) \\ \tau_1 + \tau_3 - \tau_2 - \tau_4 \end{bmatrix} = \begin{bmatrix} k \cdot l \cdot (\omega_1^2 + \omega_4^2 - \omega_2^2 - \omega_3^2) \\ k \cdot l \cdot (\omega_3^2 + \omega_4^2 - \omega_1^2 - \omega_2^2) \\ b \cdot (\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \end{bmatrix}, \quad (4)$$

kde l je délka kolmice od motoru do osy (je stejná do os x_B a y_B , protože dron je symetrický).

Tim pádem, v systému kvadrokoptéry se zavěšeným závažím máme čtyři řídicí vstupy ve tvaru tahové síly T a točivých momentů $[\tau_\phi, \tau_\theta, \tau_\psi]^T$ v soustavě tělesa. Správnější by bylo říci, že skutečnými řídicími vstupy jsou druhé mocniny úhlových rychlostí motorů ω_i^2 , avšak pro zjednodušení zápisu rovnic a analýzu systému budeme uvažovat vstupy ve tvaru síly a momentů.

2.4 Rovnice pohybu

Uvažujeme následující předpoklady při odvození matematického modelu kvadrokoptéry se zavěšeným závažím:

- kvadrokoptéra je tuhé symetrické těleso s rovnoměrným rozložením hmoty v rovině x_B - y_B
- lano je tuhé, nepružné a nehmotné
- lano a závaží považujeme za matematické kyvadlo, které je připojeno ke hmotnému středu dronu
- neuvažujeme odpor prostředí a vnější síly jako vítr jsou nulové
- motory a vrtule jsou stejné
- parametry systému se nemění s časem

Pohybové rovnice jsou symbolicky odvozovány v MATLAB za použití Symbolic Math Toolbox a vychází z Eulerovy-Lagrangeovy rovnice druhého druhu, která má následující tvar:

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{q}_j} \right) - \frac{\partial \mathcal{L}}{\partial q_j} = Q_j^{nc}, \quad j = 1 \dots 8, \quad (5)$$

kde j je číslo zobecněné souřadnice, q_j je j -tá zobecněná souřadnice a Q_j^{nc} je j -tá nekonzervativní síla působící na soustavu. Z popisu systému je vektor zobecněných souřadnic definován jako $\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8]^T = [x, y, z, \phi, \theta, \psi, \alpha, \beta]^T$. Odpovídající nekonzervativní síly jsou definovány jako $[Q_1^{nc}, Q_2^{nc}, Q_3^{nc}]^T = \mathbf{R}_{zyx} \cdot [0, 0, T]^T$, $[Q_4^{nc}, Q_5^{nc}, Q_6^{nc}]^T = \boldsymbol{\tau}_B$ a nakonec $Q_7^{nc} = Q_8^{nc} = 0$, neboť na závaží nepůsobí žádné nekonzervativní síly.

Lagrangian $\mathcal{L}$ je definován jako rozdíl mezi kinetickou a potenciální energií soustavy:

$$\mathcal{L} = E_k - E_p. \quad (6)$$

Kinetická energie soustavy se rovná součtu kinetických energií translačního a rotačního pohybu každého z těles. Kinetická energie kvadrokoptéry se rovná:

$$E_{k1} = \frac{1}{2} M \dot{x} + \frac{1}{2} M \dot{y} + \frac{1}{2} M \dot{z} + \frac{1}{2} \dot{\boldsymbol{\eta}}^T \mathbf{J} \dot{\boldsymbol{\eta}}, \quad (7)$$

kde $\mathbf{J} = \mathbf{W}_{zyx}^T \cdot \mathbf{I}_B \cdot \mathbf{W}_{zyx}$ je Jakobiho matice vyjadřující matici setrvačnosti $\mathbf{I}_B$ kvadrokoptéry v inerciální soustavě [2], neboli energii rotačního pohybu tělesa v inerciální soustavě. Matice mají následující tvar:

$$\mathbf{J} = \begin{bmatrix} I_{xx} & 0 & -I_{xx}s(\theta) \\ 0 & I_{yy}c(\phi)^2 + I_{zz}s(\phi)^2 & I_{yy}c(\phi)c(\theta)s(\phi) - I_{zz}c(\phi)c(\theta)s(\phi) \\ -I_{xx}s(\theta) & I_{yy}c(\phi)c(\theta)s(\phi) - I_{zz}c(\phi)c(\theta)s(\phi) & I_{zz}c(\phi)^2c(\theta)^2 + I_{yy}c(\theta)^2s(\phi)^2 + I_{xx}s(\theta)^2 \end{bmatrix}, \quad (8)$$

$$\mathbf{I}_B = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}, \quad I_{xx} = I_{yy}, \quad (9)$$

kde I_{xx}, I_{yy}, I_{zz} jsou hlavní momenty setrvačnosti kolem odpovídajících os tělesa.

Kinetická energie závaží se rovná:

$$E_{k2} = \frac{1}{2} m \dot{x}_p + \frac{1}{2} m \dot{y}_p + \frac{1}{2} m \dot{z}_p, \quad (10)$$

kde $[x_p, y_p, z_p]^T$ jsou souřadnice středu hmotnosti závaží v inerciální soustavě, které ale jsou závislé proměnné a vyjadřují se přes zobecněné souřadnice takto:

$$\begin{aligned} x_p &= x + d \cdot \cos(\beta) \sin(\alpha) \\ y_p &= y + d \cdot \sin(\beta) \\ z_p &= z - d \cdot \cos(\beta) \cos(\alpha) \end{aligned} \quad (11)$$

Potenciální energie systému se rovná součtu potenciální energie kvadrokoptéry a potenciální energie zavěšeného závaží a je popsána následujícím vztahem:

$$E_p = Mgz + mgz_p = Mgz + mgz - d \cdot \cos(\beta) \cos(\alpha), \quad (12)$$

kde g je tíhové zrychlení působící proti osy Z světa.

Dosažením rovnic (7)-(12) do rovnice (6) dostaneme Lagrangián, který vypadá následovně:

$$\begin{aligned} \mathcal{L} = & -Mgz - mgz + mgdc(\alpha)c(\beta) + \frac{1}{2}(M+m)(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) + \frac{1}{2}I_{xx}\dot{\phi}^2 + \frac{1}{2}(I_{yy}c(\phi)^2 + \\ & I_{zz}s(\phi)^2)\dot{\theta}^2 + \frac{1}{2}(I_{xx}s(\theta)^2 + I_{yy}c(\theta)^2s(\phi)^2 + I_{zz}c(\phi)^2c(\theta)^2)\dot{\psi}^2 + \frac{1}{2}md^2\dot{\beta}^2 + \frac{1}{2}md^2c(\beta)^2\dot{\alpha}^2 - \\ & I_{xx}s(\theta)\dot{\phi}\dot{\psi} + (I_{yy} - I_{zz})c(\phi)c(\theta)s(\phi)\dot{\theta}\dot{\psi} + md(c(\beta)\dot{y}\dot{\beta} + c(\alpha)c(\beta)\dot{x}\dot{\alpha} + c(\beta)s(\alpha)\dot{z}\dot{\alpha} + \\ & c(\alpha)s(\beta)\dot{z}\dot{\beta} - s(\alpha)s(\beta)\dot{x}\dot{\beta}). \end{aligned} \quad (13)$$

Dále dosažením (13) do rovnice (5) dostáváme osm nelineárních pohybových rovnic popisující soustavu kvadrokoptéry se zavěšeným závažím. Z důvodu rozsáhlosti a složitosti rovnic nebudou tyto rovnice uvedeny v odvozeném tvaru.

2.5 Rovnice pohybu v maticovém tvaru

Pro lepší vzhled lze rovnice odvozené v předchozí části zapsat v následujícím maticovém tvaru:

$$\mathbf{M}(\mathbf{q}) \cdot \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \cdot \dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{L}(\mathbf{q}) \cdot \mathbf{u}, \quad (14)$$

kde $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{8 \times 8}$ je matice hmotnosti a setrvačnosti, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{8 \times 8}$ je Coriolisova matice obsahující gyroskopické a dostředivé efekty, $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^{8 \times 1}$ je gravitační matice a $\mathbf{L}(\mathbf{q}) \in \mathbb{R}^{8 \times 4}$ je matice koeficientů vstupů. Vektor $\mathbf{q} \in \mathbb{R}^{8 \times 1}$ je už známý vektor zobecněných souřadnic $\mathbf{q} = [x, y, z, \phi, \theta, \psi, \alpha, \beta]^T$ a $\mathbf{u} = [T, \tau_\phi, \tau_\theta, \tau_\psi]^T \in \mathbb{R}^{4 \times 1}$ je vektor vstupů systému tvořený tahovou silou a točivými momenty. Matice byly rovněž symbolicky vyjádřeny v MATLAB a mají následující tvar:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} m_{11} & 0 & 0 & 0 & 0 & 0 & m_{17} & m_{18} \\ 0 & m_{22} & 0 & 0 & 0 & 0 & 0 & m_{28} \\ 0 & 0 & m_{33} & 0 & 0 & 0 & m_{37} & m_{38} \\ 0 & 0 & 0 & I_{xx} & 0 & -I_{xx}s(\theta) & 0 & 0 \\ 0 & 0 & 0 & 0 & m_{55} & m_{56} & 0 & 0 \\ 0 & 0 & 0 & -I_{xx}s(\theta) & m_{65} & m_{66} & 0 & 0 \\ m_{71} & 0 & m_{73} & 0 & 0 & 0 & md^2c^2(\beta) & 0 \\ m_{81} & m_{82} & m_{83} & 0 & 0 & 0 & 0 & md^2 \end{bmatrix}, \quad (15)$$

kde $m_{11} = m_{22} = m_{33} = M + m$, $m_{17} = m_{71} = mdc(\alpha)c(\beta)$, $m_{18} = m_{81} = -mds(\alpha)s(\beta)$,
 $m_{28} = m_{82} = mdc(\beta)$, $m_{37} = m_{73} = mdc(\beta)s(\alpha)$, $m_{55} = I_{yy}c^2(\phi) + I_{zz}s^2(\phi)$,
 $m_{56} = m_{65} = c(\phi)c(\theta)s(\phi)(I_{yy} - I_{zz})$, $m_{66} = I_{zz}c^2(\phi)c^2(\theta) + I_{yy}c^2(\theta)s^2(\phi) + I_{xx}s^2(\theta)$,
 $m_{38} = m_{83} = mdc(\alpha)s(\beta)$,

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & c_{17} & -md \cos(\beta) \sin(\alpha) \dot{\beta} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -md \sin(\beta) \dot{\beta} \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{37} & md \sin(\alpha) \cos(\beta) \dot{\beta} \\ 0 & 0 & 0 & 0 & c_{45} & c_{46} & 0 & 0 \\ 0 & 0 & 0 & c_{54} & 0 & c_{56} & 0 & 0 \\ 0 & 0 & 0 & c_{64} & c_{65} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{77} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{87} & 0 \end{bmatrix}, \quad (16)$$

kde $c_{54} = \dot{\psi} (I_{xx}c(\theta) + I_{yy}c^2(\phi)c(\theta) - I_{zz}c^2(\phi)c(\theta) - I_{yy}c(\theta)s^2(\phi) + I_{zz}c(\theta)s^2(\phi)) -$

$$\begin{aligned}
& -\dot{\theta} (2I_{yy}c(\phi)s(\phi) - 2I_{zz}c(\phi)s(\phi)), c_{64} = \dot{\psi} (2I_{yy}c(\phi)c^2(\theta)s(\phi) - 2I_{zz}c(\phi)c^2(\theta)s(\phi)) - \\
& -\dot{\theta} (I_{xx}c(\theta) - I_{yy}c^2(\phi)c(\theta) + I_{zz}c^2(\phi)c(\theta) + I_{yy}c(\theta)s^2(\phi) - I_{zz}c(\theta)s^2(\phi)), c_{45} = \\
& \dot{\theta} (I_{yy}c(\phi)s(\phi) - I_{zz}c(\phi)s(\phi)) - \dot{\psi} (I_{xx}c(\theta) + I_{yy}c^2(\phi)c(\theta) - I_{zz}c^2(\phi)c(\theta) - I_{yy}c(\theta)s^2(\phi) + \\
& I_{zz}c(\theta)s^2(\phi)), c_{65} = -\dot{\theta} (I_{yy}c(\phi)s(\phi)s(\theta) - I_{zz}c(\phi)s(\phi)s(\theta)) - \dot{\psi} (2I_{zz}c(\theta)s(\theta)c^2(\phi) + \\
& + 2I_{yy}c(\theta)s(\theta)s^2(\phi) - 2I_{xx}c(\theta)s(\theta)), c_{46} = -\dot{\psi}c(\phi)c^2(\theta)s(\phi)(I_{yy} - I_{zz}), \\
& c_{56} = \dot{\psi}c(\theta)s(\theta)(I_{zz}c^2(\phi) + I_{yy}s^2(\phi) - I_{xx}), c_{17} = -md(\dot{\alpha}c(\beta)s(\alpha) - 2\dot{\beta}c(\alpha)s(\beta)), \\
& c_{37} = md(\dot{\alpha}c(\alpha)c(\beta) - 2\dot{\beta}s(\alpha)s(\beta)), c_{77} = -md^2s(2\beta)\dot{\beta}, c_{87} = md^2c(\beta)s(\beta)\dot{\alpha},
\end{aligned}$$

$$\mathbf{G}(\mathbf{q}) = \begin{pmatrix} 0 \\ 0 \\ (M+m)g \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad (17)$$

$$\mathbf{L}(\mathbf{q}) = \begin{pmatrix} mgd \sin(\alpha) \cos(\beta) \\ mgd \cos(\alpha) \sin(\beta) \\ s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) \\ c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) \\ c(\phi)c(\theta) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \quad (18)$$

2.6 Linearizace a lineární stavový model

V případě analýzy systému a pro návrh řídicích algoritmů je nutný lineární model systému. Zprvce přepíšeme maticový tvar (14) na soustavu diferenciálních rovnic prvního řádu (nelineární stavový model), která má tvar:

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\
\mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))'
\end{aligned} \quad (19)$$

kde funkce $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ je vektor stavových rovnic, často také vektor pravých stran, a funkce $\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))$ je vektor výstupních rovnic, $\mathbf{x}(t)$ je stavový vektor, $\mathbf{u}(t)$ je vektor vstupů systému a $\mathbf{y}(t)$ je vektor výstupů systému, který obvykle reprezentuje měřitelné veličiny systému. Nelineární stavový popis systému pak vypadá následovně:

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \begin{bmatrix} \dot{\mathbf{q}}(t) \\ \ddot{\mathbf{q}}(t) \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}}(t) \\ \mathbf{M}^{-1}(\mathbf{L}(\mathbf{q}) \cdot \mathbf{u}(t) - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \cdot \dot{\mathbf{q}}(t) - \mathbf{G}(\mathbf{q})) \end{bmatrix}, \\
\mathbf{y}(t) &= \mathbf{C} \cdot \mathbf{x}(t)
\end{aligned} \quad (20)$$

kde $\mathbf{x}(t) = [\mathbf{q}(t), \dot{\mathbf{q}}(t)]^T$ tvoří stavový vektor. Pro daný systém funkce $\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))$ je lineární a je definovaná diagonální maticí $\mathbf{C}$, která má jedničky na pozicích odpovídajících těm stavovým veličinám, které jsou požadovány na výstupu (které jsou měřitelné).

Linearizace probíhá kolem rovnovážného bodu, v kterém kvadrokoptéra se závažím je v rovnováze a ve stavu visení, tj. motory generují celkovou tahovou sílu rovnou tíze systému. Hodnoty stavů a vstupů v pracovním bodě se rovnají $\mathbf{x}_s = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$ a $\mathbf{u}_s = [(M+m)g, 0, 0, 0]^T$.

Pak lze lineární rovnice reprezentovat ve stavovém prostoru takto:

$$\begin{aligned}\Delta \dot{\mathbf{x}}(t) &= \mathbf{A}_c \cdot \Delta \mathbf{x}(t) + \mathbf{B}_c \cdot \Delta \mathbf{u}(t) \\ \Delta \mathbf{y}(t) &= \mathbf{C}_c \cdot \Delta \mathbf{x}(t) + \mathbf{D}_c \cdot \Delta \mathbf{u}(t)\end{aligned}\quad (21)$$

kde $\Delta \mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_s$ a $\Delta \mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}_s$ jsou odchylky od bodu linearizace. Matice $\mathbf{A}_c$, $\mathbf{B}_c$, $\mathbf{C}_c$ a $\mathbf{D}_c$ jsou Jakobiho matice, které jsou definovány následovně:

$$\mathbf{A}_c = \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{(\mathbf{x}_s, \mathbf{u}_s)} \quad (22)$$

$$\mathbf{B}_c = \left. \frac{\partial f}{\partial \mathbf{u}} \right|_{(\mathbf{x}_s, \mathbf{u}_s)} \quad (23)$$

$$\mathbf{C}_c = \left. \frac{\partial g}{\partial \mathbf{x}} \right|_{(\mathbf{x}_s, \mathbf{u}_s)} \quad (24)$$

$$\mathbf{D}_c = \left. \frac{\partial g}{\partial \mathbf{u}} \right|_{(\mathbf{x}_s, \mathbf{u}_s)} \quad (25)$$

Derivací pravé strany z rovnice (20) a dosazením hodnot pracovního bodu byly určeny matice lineárního stavového modelu v parametrickém tvaru, které jsou uvedeny dále. Výstupní matice $\mathbf{C}_c$ je definována stejně jako matice $\mathbf{C}$ a matice $\mathbf{D}_c$ je nulová.

$$\mathbf{A}_c = \begin{matrix} & & & & \mathbb{Q}^{8 \times 8} & & & & \mathbb{I}^{8 \times 8} \\ & & & & \frac{(M+m)g}{M} & 0 & \frac{mg}{M} & 0 & \\ & & & & 0 & 0 & 0 & \frac{mg}{M} & \\ & & & & 0 & 0 & 0 & 0 & \\ & & & & 0 & 0 & 0 & 0 & \\ & & & & 0 & 0 & 0 & 0 & \\ & & & & 0 & 0 & 0 & 0 & \\ & & & & -\frac{(M+m)g}{Md} & 0 & -\frac{(M+m)g}{Md} & 0 & \\ \left[\begin{array}{cccc} 0 & 0 & 0 & \frac{(M+m)g}{Md} \\ & & & 0 \\ & & & 0 \\ & & & 0 \end{array} \right. & & & & 0 & 0 & 0 & -\frac{(M+m)g}{Md} & \left. \right] \end{matrix} \quad \mathbb{Q}^{8 \times 8} \quad (26)$$

$$\mathbf{B}_c = \begin{matrix} & & & & \mathbb{Q}^{10 \times 4} & & & \\ & & & & \frac{1}{M+m} & 0 & 0 & 0 & \\ & & & & 0 & \frac{1}{I_{xx}} & 0 & 0 & \\ & & & & 0 & 0 & \frac{1}{I_{yy}} & 0 & \\ & & & & 0 & 0 & 0 & \frac{1}{I_{zz}} & \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right. & & & & & & & & \end{matrix} \quad (27)$$

Jak nelineární, tak i lineární model byly realizovány v Simulink pomocí bloků *Integrator* a *MATLAB Functions*, do kterých byly vepsány odpovídající části pohybových rovnic.

3. Ověření dynamiky

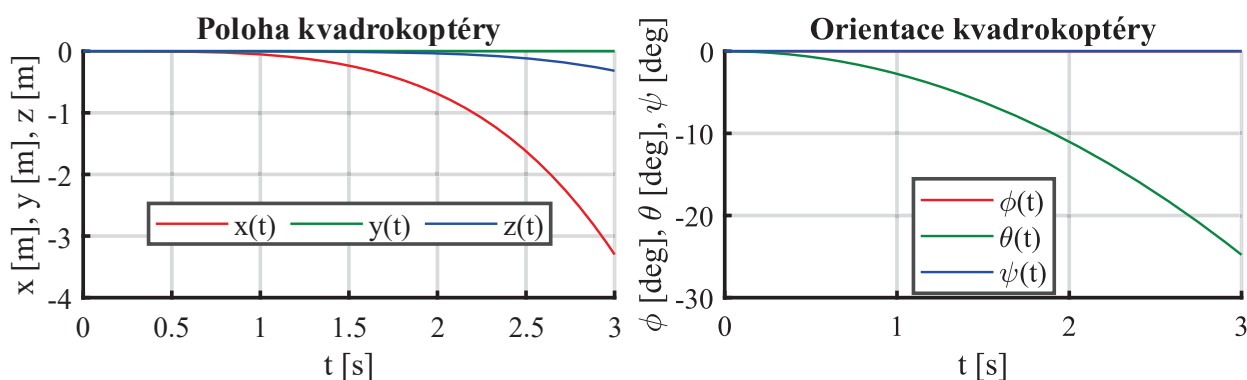
V dané části uvádím odezvy odvozeného nelineárního matematického modelu na různé hodnoty otáček motorů. Cílem je ověřit dynamiku kvadrokoptéry se zavěšeným závažím a potvrdit, že její chování odpovídá předpokladům a popisům uvedeným v části 2.1. Simulace probíhaly v MATLAB/Simulink s parametry systému uvedenými v následující tabulce.

Tab. 1: Parametry systému.

Parametr	Hodnota
M	2 kg
m	1 kg
$I_{xx} = I_{yy}$	$0,0222 \text{ kgm}^2$
I_{zz}	$0,0429 \text{ kgm}^2$
l	$0,2051 \text{ m}$
d	1 m
k	$2,3 \cdot 10^{-3} \text{ N}(\text{ot/s})^{-2}$
b	$5,4 \cdot 10^{-6} \text{ Nm}(\text{ot/s})^{-2}$
g	$9,81 \text{ m/s}^2$

3.1 Scénář 1: Natočení *pitch*

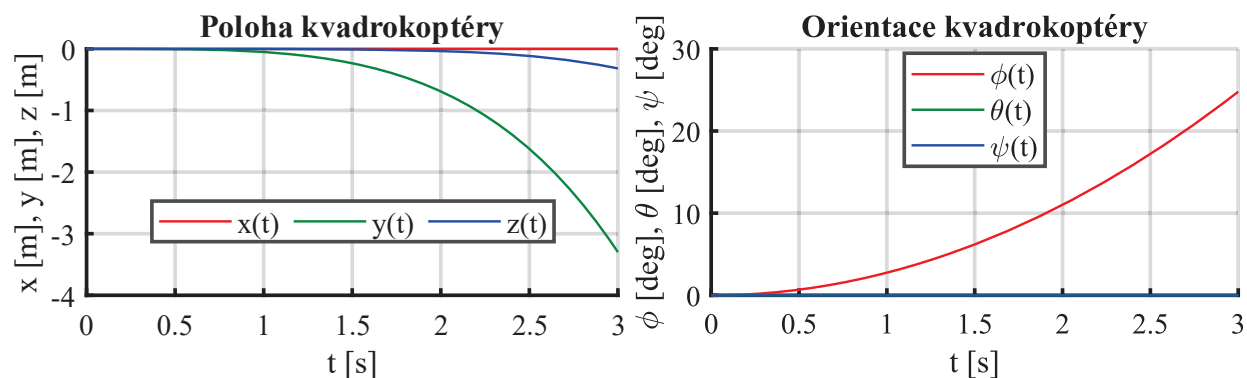
V tomto scénáři posílám na vstupy modelů takové otáčky motorů, že motory 1 a 2 generují větší tahovou sílu než motory 3 a 4. Vlivem nerovnováhy vzniká točivý moment, který vyvolává zápornou rotaci *pitch* (θ). Současně vlivem rozkladu celkové tahové síly na svislou i vodorovnou složku dochází k pohybu dronu v záporném směru osy x .



Obr. 2 Poloha a orientace kvadrokoptéry, scénář 1 [autor]

3.2 Scénář 2: Natočení *roll*

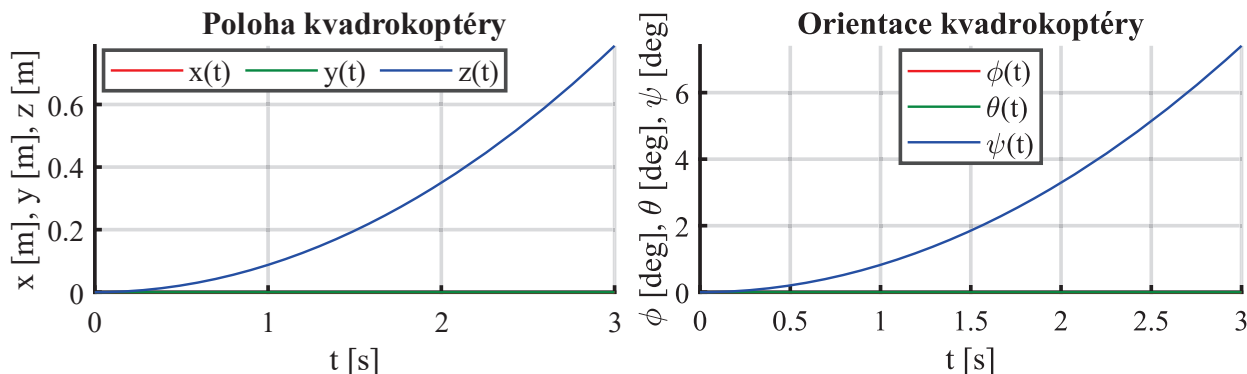
Obdobně jako výše, v tomto scénáři posílám na vstupy takové otáčky motorů, že motory 1 a 4 generují větší tahovou sílu než motory 2 a 3. Vlivem nerovnováhy vzniká točivý moment, který vyvolává kladnou rotaci *roll* (ϕ). Současně vlivem rozkladu celkové tahové síly na svislou i vodorovnou složku dochází k pohybu dronu v záporném směru osy y . Protože dron je symetrický, odezva je stejná (bez uvažování znaménka), jenom pro jiné uhly a souřadnice.



Obr. 3 Poloha a orientace kvadrokoptéry, scénář 2 [autor]

3.3 Scénář 3: Natočení *yaw* a změna výšky

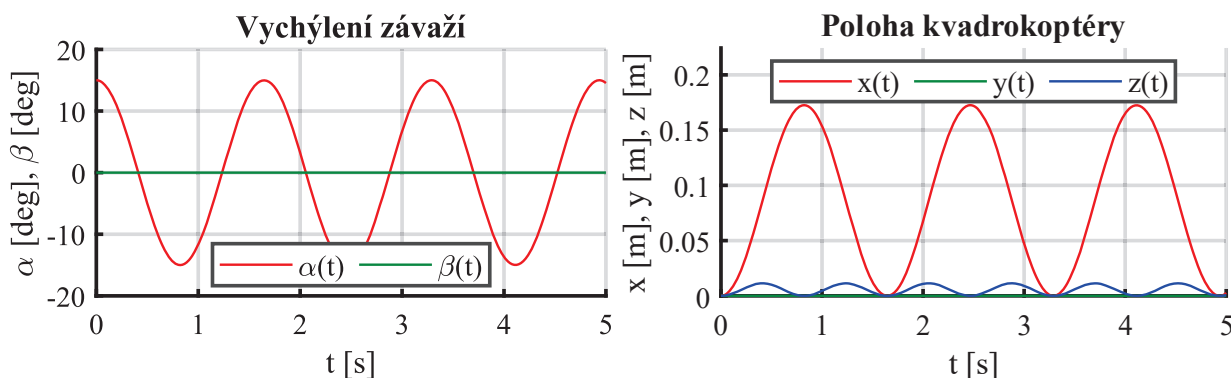
V tomto scénáři kombinuji dva nezávislé pohyby dronu a to tak, že zvyšuji otáčky motorů 1 a 3 a neměním otáčky motorů 2 a 4. Celková tahová síla se zvýší a zároveň odporové momenty na vrtulích nejsou v rovnováze. To vyvolává pohyb kvadrokoptéry nahoru a kladnou rotaci *yaw* (ψ) kolem svislé osy.



Obr. 4 Poloha a orientace kvadrokoptéry, scénář 3 [autor]

3.4 Scénář 4: Vliv kmitání závaží

V posledním scénáři pro zajímavost uvádím vliv kmitání závaží na pohyb kvadrokoptéry, která je v rovnovážném stavu visení. Počáteční poloha závaží je nastavena na $\alpha = 15^\circ$ a $\beta = 0^\circ$, tzn. kmitání bude probíhat v rovině $x_B z_B$ systému O_{B_0} . Jak je vidět z obrázku, závaží a dron mají ustálené kmity z důvodu absence aerodynamického odporu. Dále je vidět, že kmitání podél úhlu α vyvolává kmity kvadrokoptéry v ose x a ve výšce z . Analogické chování lze pozorovat, pokud závaží bude kmitat podél úhlu β , což způsobuje kmitání dronu v ose y .



Obr. 4 Poloha kvadrokoptéry a orientace závaží, scénář 4 [autor]

4. Závěr

V tomto článku byl představen podrobný popis odvození pohybových rovnic soustavy kvadrokoptéry se zavěšeným závažím ve 3D. Výsledkem je sada matic v parametrickém tvaru, na jejichž základě byl vytvořen model systému v MATLAB/Simulink. Dále bylo provedeno ověření dynamiky pro různé hodnoty vstupních otáček motorů. Výsledky simulací ukazují, že dynamika dronu odpovídá skutečnosti a schématu na Obr. 1. Konkrétně vzniká pohyb ve směru osy x resp. y při natočení o úhel *pitch* (θ) resp. *roll* (ϕ) změnou otáček odpovídajících motorů. Dále dochází k svislému pohybu (stoupání/klesání) a natočení *yaw* (ψ) kolem svislé osy. Na Obr. 2 lze vidět snížení výšky z kvadrokoptéry. K tomu dochází v důsledku natočení kvadrokoptéry a rozkladu celkové tahové síly na svislou a vodorovnou složku. Protože není implementována zpětnovazební regulace, otáčky a tahové síly na motorech zůstávají konstantní a při dalším natočení dronu se hodnota svislé složky celkové tahové síly zmenšuje, což vede k neschopnosti udržet kvadrokoptéru ve stavu visení.

5. Poděkování

Tato práce byla podpořena grantem Studentské grantové soutěže ČVUT č. SGS24/125/OHK2/3T/12.

Literatura

- [1] HASHEMI, Danial a HAMIDREZA, Heidari. Trajectory Planning of Quadrotor UAV with Maximum Payload and Minimum Oscillation of Suspended Load Using Optimal Control. Online. *Journal of Intelligent & Robotic Systems*. 2020, vol. 100, s. 1369-1381. ISSN 1573-0409. Dostupné z: <https://doi.org/10.1007/s10846-020-01166-4>. [cit. 2025-02-11].
- [2] MARTINI, Simone; SÖNMEZ, Serhat; RIZZO, Alessandro; STEFANOVIC, Margareta; RUTHERFORD, Matt J. et al. Euler-Lagrange Modeling and Control of Quadrotor UAV with Aerodynamic Compensation. Online. In: *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. Dubrovnik, Croatia, 2022, s. 369-377. ISBN 978-1-6654-0593-5. ISSN 2575-7296. Dostupné z: <https://doi.org/10.1109/ICUAS54217.2022.9836215>. [cit. 2025-02-12].
- [3] CRAIG, Kevin C. Quadrotor Equations of Motion and Control KCC Final 4 2023 Video. Online. YouTube. 2023-4-16. Dostupné z: <https://www.youtube.com/watch?v=xCoFaTyn5dg>. [cit. 2024-04-14].
- [4] *Spherical pendulum*. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, stránka naposledy edit. 04.01.2024 v 19:17. Dostupné z: https://en.wikipedia.org/wiki/Spherical_pendulum. [cit. 2025-04-14].
- [5] *6DOF (Euler Angles)*. Online. THE MATHWORKS, INC. Mathworks.com. © 1994-2025. Dostupné z: <https://www.mathworks.com/help/aeroblks/6dofeulerangles.html>. [cit. 2025-04-14].

Úlohy pro virtuální laboratoř v JavaScriptu

Petr Musil¹

¹ *Technická 4, 160 00, Praha 6, petr.musil@fs.cvut.cz*

Abstrakt: Tento dokument popisuje návrh, jak bylo postupováno během vytváření nové virtuální laboratoře pro předmět automatické řízení. Důvodem bylo, že původní virtuální laboratoř přestala být vyhovující z hlediska provozu v moderních webových prohlížečích. Dalším důvodem bylo zatraktivnit úlohy pro nové studenty a tím také zvýšit zájem o předmět. Úlohy byly napsány v jazyce JavaScript s HTML a CSS a to bez frameworků, díky tomu bylo docílené, že je lze spouštět v libovolném webovém prohlížeči.

Klíčová slova: JavaScript, virtuální úlohy, logické řízení, HTML Canvas

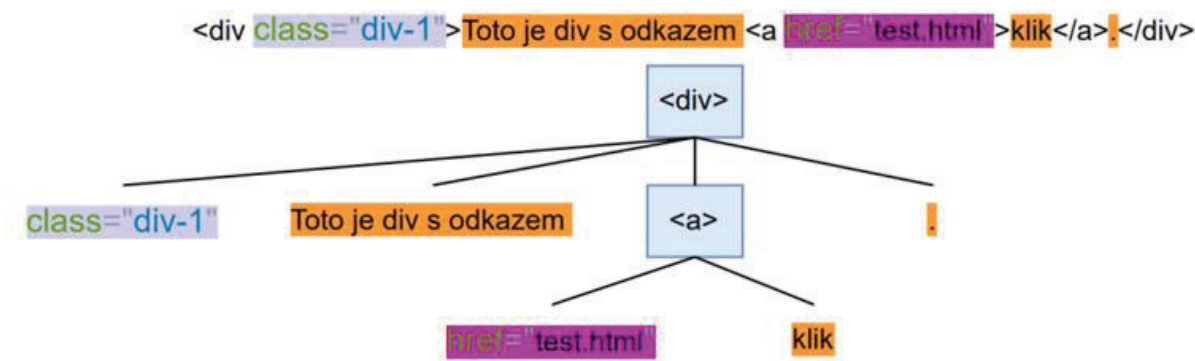
Abstract: This document describes the design and process of creating a new virtual laboratory for the Automatic Control course. The main reason was that the original virtual laboratory was no longer compatible with modern web browsers. Another goal was to make the tasks more attractive for new students, thus increasing interest in the subject. The tasks were written in JavaScript with HTML and CSS without using frameworks, allowing them to be run in any web browser.

Keywords: JavaScript, virtual tasks, logical control, HTML Canvas

1. Úvod

Cílem této práce bylo vytvořit novou virtuální laboratoř pro předmět Automatické řízení vyučovaný na ČVUT v Praze. Původní laboratoř byla vytvořena pomocí Java appletů, které byly vykreslovány ve webovém prohlížeči. Tyto appletové aplikace jsou však dnes již zastaralé a moderní webové prohlížeče je nepodporují, zejména z důvodu bezpečnostních rizik. Pro spuštění původní laboratoře bylo nutné instalovat doplňky třetích stran a zapnout v prohlížeči režim zpětné kompatibility, což značně omezovalo přístupnost a použitelnost laboratoře. Nová virtuální laboratoř je vytvořena v jazyce JavaScript s použitím HTML a CSS, což zajišťuje její dostupnost ve všech moderních webových prohlížečích bez nutnosti dalších úprav. Projekt se zaměřil na zajištění budoucí udržitelnosti kódu a na atraktivitu úloh, aby se zvýšil zájem studentů o předmět.

2. Použité technologie



Obrázek 1: DOM struktura

Nová laboratoř byla vyvinuta s využitím tří klíčových webových technologií: HTML, CSS a JavaScript.

HTML (HyperText Markup Language): HTML slouží k vytvoření základní struktury a obsahu webových stránek. Poskytuje univerzální formát pro publikování textů, obrázků a dalších multimediálních prvků. [1]

CSS (Cascading Style Sheets): CSS umožňuje stylování a rozvržení webových stránek. Odděluje obsah HTML od jeho vizuální prezentace, což zlepšuje čitelnost a údržbu kódu. Použití responsivního designu pomocí media queries zajišťuje, že aplikace je vhodná pro různé velikosti obrazovek. [1]

JavaScript: JavaScript přidává interaktivitu a dynamické chování. Byl použit pro logické řízení, manipulaci s DOM a vytváření animací v HTML Canvas, což umožňuje zobrazování výsledků řízení v reálném čase.[2]

3. Návrh a implementace

Nová virtuální laboratoř byla navržena tak, aby zachovala klíčové funkce původní verze, ale zároveň přinesla modernější a uživatelsky přívětivější prostředí. Celý vývoj probíhal od základu, bez použití kódu z původní laboratoře, což bylo nutné kvůli zastaralosti původní technologie a potřeby modernizace. To umožnilo kompletně přepracovat laboratoř, a zároveň zajistit nezávislost na starých a již nepodporovaných prvcích, jako byly Java applety. Nová verze laboratoře byla navržena s ohledem na budoucí rozšiřitelnost a snadnou údržbu, což bylo jedním z hlavních požadavků.

Vývojové prostředí a nástroje:

Pro vývoj laboratoře byl použit editor Visual Studio Code, který poskytuje bohaté možnosti pro práci s kódem v jazycích HTML, CSS a JavaScript. Tento editor umožňuje využívat rozšíření, jako je Live Server, který simuluje běh aplikace na serveru a automaticky aktualizuje zobrazení v prohlížeči po každé úpravě kódu. To významně urychlilo testování a ladění aplikace, protože jakákoliv změna byla ihned viditelná.

Pro grafickou stránku laboratoře byl využit software GIMP (GNU Image Manipulation Program), který sloužil k úpravám a tvorbě rastrové grafiky. Některé prvky byly navíc generovány pomocí umělé inteligence a následně upraveny tak, aby vyhovovaly požadavkům

na animace a vizuální ztvárnění úloh. Tvorba grafiky byla časově náročná, jelikož musela být přizpůsobena potřebám jednotlivých úloh a zároveň zajišťovat estetickou a funkční stránku aplikace.

Struktura a organizace kódu

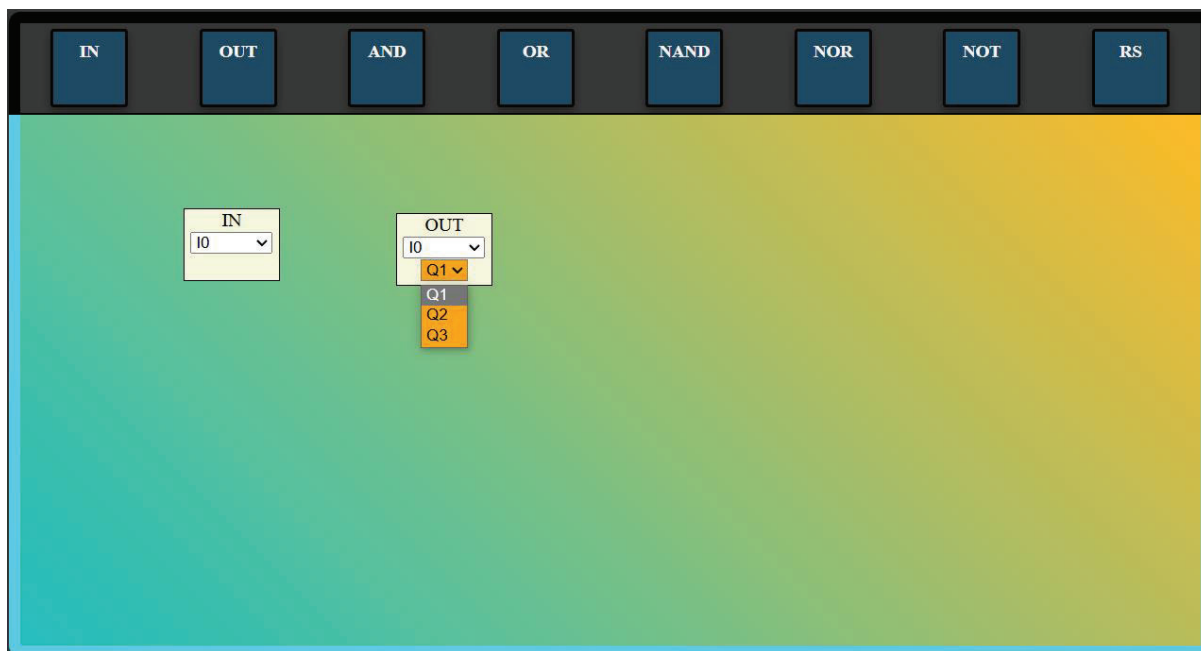
HTML soubory: Tyto soubory definují základní strukturu každé úlohy v laboratoři. Každá úloha má svůj vlastní HTML soubor, který obsahuje veškeré prvky potřebné pro vykreslení úlohy v prohlížeči.

JavaScript soubory: JavaScript je srdcem celé laboratoře a zajišťuje veškerou logiku a interakci. Kód byl rozdělen na dvě části – jeden soubor obsahuje skripty společné pro všechny úlohy (např. manipulace s logickými bloky a jejich propojení), zatímco druhá část je věnována specifickým skriptům pro jednotlivé úlohy.

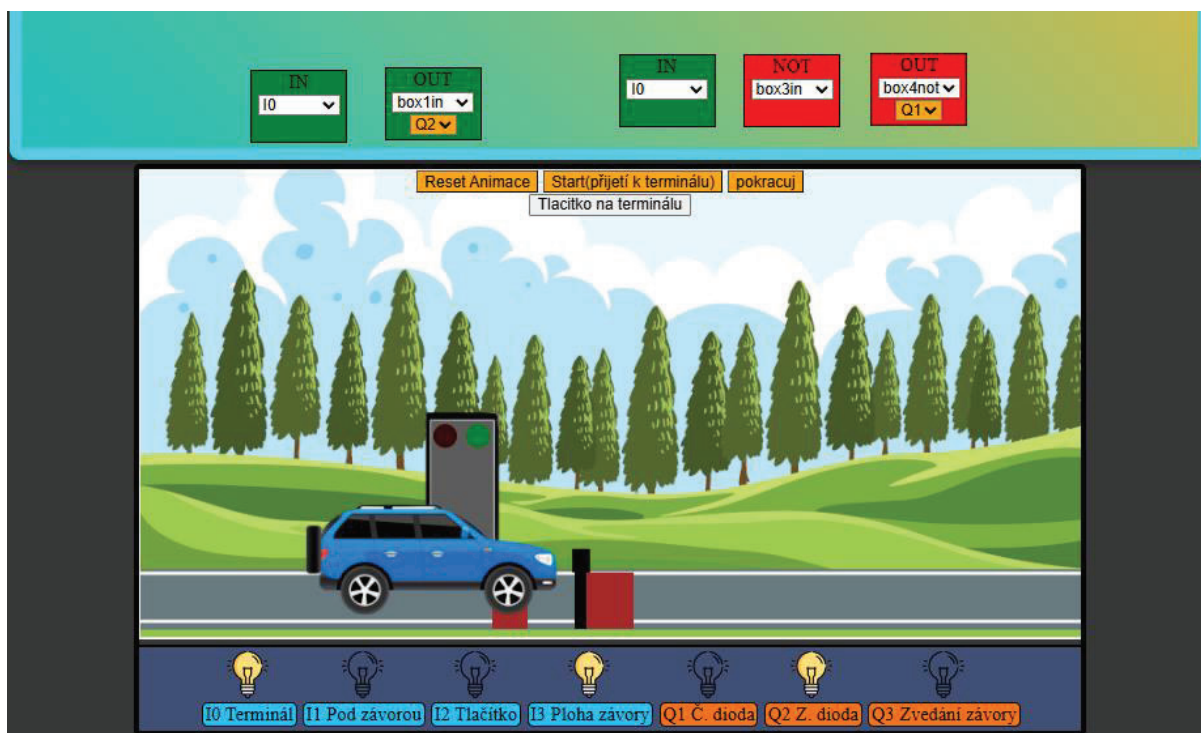
CSS soubory: Pro stylování byly vytvořeny CSS soubory, které zajišťují vizuální stránku laboratoře. Každá úloha má svůj vlastní CSS soubor pro specifické styly, zatímco jeden hlavní CSS soubor spravuje obecné styly a rozložení celé aplikace. Díky tomu lze snadno upravovat vzhled celé laboratoře nebo jednotlivých úloh bez nutnosti složitých zásahů do kódu.

Složka s obrázky: Obrázky byly rozděleny do podadresářů podle jednotlivých úloh, což umožňuje snadnou správu a úpravu grafiky. Každý obrázek je propojen přímo s konkrétní úlohou, což usnadňuje jejich výměnu nebo aktualizaci. V této složce jsou také umístěny zdrojové soubory pro GIMP, které umožňují úpravy a přizpůsobení grafiky dle potřeby.

4. Funkčnost nové virtuální laboratoře



Obrázek 2: kompletní část pro tvorbu logiky



Obrázek 3: Ukázka použití logiky a animace

Nová laboratoř nabízí uživatelsky přívětivé rozhraní, které umožňuje studentům lépe pochopit principy automatického řízení díky interaktivním úlohám. Logické bloky jsou vykreslovány přímo v prohlížeči pomocí HTML Canvas, což umožňuje vizualizaci výsledků logického řízení v reálném čase. Animace a interakce mezi prvky zajišťují, že studenti mohou přímo sledovat změny a vyhodnocovat chování systému.

Laboratoř je navržena tak, aby byla snadno rozšiřitelná a upravitelná. Využití čistého JavaScriptu bez frameworků zajišťuje, že kód bude kompatibilní s budoucími verzemi webových standardů, což usnadňuje dlouhodobou údržbu a přizpůsobení.

5. Závěr

Nová virtuální laboratoř splňuje požadavky moderních webových aplikací a odstraňuje nedostatky původní verze. Díky využití JavaScriptu, HTML a CSS je laboratoř plně kompatibilní s dnešními prohlížeči, což výrazně zlepšuje přístupnost a použitelnost. Zároveň se podařilo zvýšit atraktivitu úloh pro studenty, což přispívá k většímu zájmu o předmět Automatické řízení. Nová laboratoř představuje robustní a flexibilní nástroj, který lze snadno přizpůsobit budoucím potřebám výuky.

Reference

- [1] DUCKETT, Jon. HTML & CSS: design and build websites. Indianapolis, IN: Wiley, [2011]. ISBN 978-1-118-00818-8.
- [2] FLANAGAN, David. JavaScript: The Definitive Guide. O'Reilly Media. 6th Edition. Sebastopol, CA: O'Reilly Media, 2011. 1098 s. ISBN 978-0-596-80552-4
- [3] javascript.info [online]. [cit. 2024-07-10]. Dostupné z: [Elektronická učebnice - ELUC \(ikap.cz\)](https://www.javascript.info)
- [4] CROFT, Jeff; LLOYD, R. Ian a RUBIN, Dan. Mistrovství v CSS: pokročilé techniky pro webové designéry a vývojáře. Mistrovství. Brno: Computer Press, 2007. ISBN 978-80-251-1705-7.

ZESÍŤOVÁNÍ ROHOVKY S INTAKTNÍM EPITELEM: SOUČASNÉ MOŽNOSTI A PERSPEKTIVY

Aruzhan Yerzhanova¹,

I České vysoké učení technické v Praze, yerzharu@student.cvut.cz

Abstrakt: Zesíťování rohovky s intaktním epitelem (transepiteliální CXL) je alternativní metoda ke standardnímu postupu, který vyžaduje odstranění epitelu. Transepiteliální přístup eliminuje nutnost mechanické de-epitelizace, čímž se snižuje výskyt některých nežádoucích účinků, jako je bolest nebo riziko infekce. Na druhé straně je jeho účinnost stále předmětem výzkumu, především kvůli omezenému průniku riboflavinu přes epitel.

Klíčová slova: Zesíťování, rohovka, iontoforéza, epitel,

1. Úvod

Keratokonius představuje postupně se zhoršující ektatické onemocnění rohovky, které může způsobit vážné narušení zrakových funkcí. Nemoc se zpravidla začíná projevovat v pozdní fázi puberty nebo na začátku dospělosti a její průběh může být různě rychlý. Bez odpovídající léčby může stav pokročit natolik, že bude vyžadovat transplantaci rohovky. Zastavení progresu keratokoniu se primárně dosahuje prostřednictvím zesíťování kolagenových vláken rohovky. Původní drážďanský postup pro zesíťování zahrnuje odstranění rohovkového epitelu [1]

2. Úvodní přehled o technice zesíťování rohovky

2.1 Zaklání princip

Rohovkové stroma představuje komplexní strukturu tvořenou především kolagenními vlákny typu I a V, která jsou propojena prostřednictvím základních proteinů hlavních proteoglykanů. Mezi tyto klíčové proteoglykany patří keratokan, lumikan, mimekan a dekorin, jejichž vzájemné interakce hrají zásadní roli v udržování průhlednosti stromatu.[2]

Procedura zesíťování posiluje zmíněné vazby vytvořením dodatečných kovalentních spojení mezi aminotermíny kolagenních postranních řetězců a základními proteiny proteoglykanů, které se nacházejí v mezibuněčné hmotě. Tento proces významně přispívá ke zpevnění stromální struktury. [2]

Mechanismus, kterým CXL zvyšuje tuhost stromatu, souvisí s modifikací průměru kolagenních vláken a snížením vzdáleností mezi jednotlivými fibrilami. Výzkumné studie prokázaly přibližně 5% nárůst tloušťky kolagenních fibril po aplikaci CXL. Současně byla

dokumentována pozitivní korelace mezi intenzitou ozáření a redukcí interfibrilárních vzdáleností. [2]

3. Techniky bez porušení epitelové vrstvy

3.1 Současný stav transepiteliálních technik

Transepiteliální neboli "epithelium-on" (epi-on) techniky CXL byly vyvinuty s několika hlavními cíli. Primárně se zaměřují na snížení pooperační bolesti pacientů, minimalizaci komplikací souvisejících s ránou a redukcí rizika infekce. Dalším významným přínosem je možnost provádění léčby mimo operační sál, což přináší praktické výhody pro zdravotnická zařízení i pacienty.[3]

Přesto se tyto techniky stále považují za méně účinné ve srovnání s konvenčním přístupem odstraňujícím epitel. [3]

3.2 Některé metody

3.2.1 Zrychlené zesíťování

Bunsen-Roscoeův zákon stanovuje, že biologická odpověď tkáně na ozáření je funkcí celkové absorbované dávky energie, přičemž tato dávka představuje produkt intenzity světelného zdroje a času působení. Na základě tohoto pravidla reciprocity je možné zachovat stejný léčebný účinek při použití vyšší intenzity světla a odpovídajícím zkrácením délky ozařování. Tento fyzikální princip poskytuje odůvodnění pro implementaci rychlejších protokolů cross-linkingové terapie.[4]

3.2.2 Iontoforéza

Iontoforéza je neinvazivní metoda pro dodávání léčivých latek do těla pomocí mírného elektrického proudu, který usnadňuje pronikání nabitých molekul biologickými tkáněmi. Vzhledem ke svému negativnímu elektrickému náboji a příznivým charakteristikám byl riboflavin použit tento typ podání v rámci procedury zesíťování. [4]

4. Porovnání konvenční techniky s transepiteliální

Studie zaměřené na biomechanické vlastnosti využívající Brillouinovu mikroskopickou analýzu zkoumaly mechanickou pevnost rohovky jako měřítko efektivity cross-linkingu. Tato měření umožnila přímé srovnání mezi přístupem zachovávajícím epitel a standardní metodou s odstraněním epitelu podle protokolu z Drážďan. Výsledky prokázaly, že mechanická rigidita rohovky dosažená při zachování epitelu je přibližně o 70 % nižší než u konvenčního postupu s odstraněným epitelem. Následná konfokální mikroskopická analýza odhalila, že při metodě se zachovaným epitelem se apoptotické změny vyskytují nerovnoměrně a penetrují pouze do hloubky 80-100 mikrometrů, zatímco standardní technika s odstraněným epitelem dosahuje do hloubky 250-300 mikrometrů. [4]

Tyto nálezy naznačují, že ačkoliv přístup se zachovaným epitelem může přechodně modifikovat optické charakteristiky rohovky prostřednictvím povrchového působení, jeho penetrační kapacita do hlubších vrstev stromatu zůstává limitovaná.[4]

5. Závěr

Dosavadní výsledky metody epi-on CXL ukazují na její omezenou účinnost, a proto je nutné další zkoumání, které by mohlo vést k jejímu širšímu uplatnění. Alternativní řešení bylo zkoumáno v rámci diplomové práce, která se zabývala návrhem optické komponenty.

Literatura

- [1] BALIDIS, Miltos, Spyridon KORONIS, Penelope Burle de POLITIS, Georgios SIDIROPOULOS a Achilleas RASOGLOU. Two-year Outcomes of Transepithelial Customized Cross-linking for Mild to Moderate Keratoconus. *International Journal of Keratoconus and Ectatic Corneal Diseases* [online]. 2023, vol. 10, no. 1–2, s. 26–31 [cit. 2025-05-31]. ISSN 2589-8678. Dostupné z: <https://doi.org/10.5005/jp-journals-10025-1196>
- [2] CHUNYU, Tian, Xiujun PENG, Zhengjun FAN, Xia ZHANG a Feihu ZHOU. *Corneal collagen cross-linking in keratoconus: A systematic review and meta-analysis*. *Scientific Reports* [online]. 2014, roč. 4, článek č. 5652 [cit. 2025-05-31]. DOI: 10.1038/srep05652.
- [3] MAZZOTTA, Cosimo a Roy S. RUBINFELD. *Epi-on Versus Epi-off: Where Do We Stand?* *Cataract & Refractive Surgery Today* [online]. 2017, červenec [cit. 2025-05-31]. Dostupné z: <https://crstoday.com/articles/2017-jul/epi-on-versus-epi-off-where-do-we-stand>
- [4] BOURDON, Hélène, et al. *Efficacy and safety of accelerated (45 mW/cm²) transepithelial corneal collagen crosslinking for progressive keratoconus: 12-month results*. *Journal Français d'Ophtalmologie* [online]. 2021, roč. 44, č. 10, s. 1420–1427 [cit. 2025-05-31]. DOI: 10.1016/j.jfo.2021.06.002. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S1773224721002045>

APLIKACE FLUIDNÍHO SVALU PŘI POKROČILÉM ŘÍZENÍ BALANCÉRU

Matouš Urbanec¹, Marie Martinásková²

¹ FS ČVUT, matous.urbanec@seznam.cz

² FS ČVUT, marie.martinaskova@fs.cvut.cz

Abstrakt: Tato práce se zabývá návrhem a implementací pokročilého balancéru využívajícího fluidní sval jako nosný i senzorický prvek. Popsány jsou dvě varianty řízení – základní systém s tlakovým senzorem a solenoidovým ventilem a pokročilejší verze s Festo Motion Terminálem, který umožňuje plynulé řízení pohybu podle velikosti působící síly. Výsledné zařízení poskytuje intuitivní a ergonomické ovládání manipulace s břemenem bez potřeby tlačítek či mechanických ovladačů. Cílem je zvýšit bezpečnost, přesnost a komfort obsluhy při zachování jednoduchosti konstrukce.

Klíčová slova: fluidní sval, balancér, manipulátor, Festo motion terminal, digitální pneumatika

1 Úvod

V moderním průmyslovém prostředí, kde efektivita, bezpečnost a ergonomie hrají klíčovou roli, se čím dál častěji setkáváme s technologiemi, které usnadňují manipulaci s těžkými či objemnými břemeny. Mezi tyto technologie patří především manipulátory a balancéry – zařízení, která se sice mohou na první pohled jevit podobně, ale ve skutečnosti plní odlišné funkce a jsou navržena pro různé typy úloh.

Manipulátor je aktivní systém, který umožňuje přesné řízení pohybu objektu v několika osách. Bývá vybaven pneumatickými nebo elektrickými pohony, senzory a ovládacími prvky, díky nimž může obsluha snadno a bezpečně manipulovat s břemeny – a to nejen ve vertikálním směru, ale i horizontálně, včetně natáčení, naklánění nebo přesného usazení dílu. Manipulátory často připomínají robotickou paži a v některých případech se skutečně jedná o poloautomatizovaná zařízení schopná komplexních pohybů.

Balancér naproti tomu funguje spíše jako "asistent hmotnosti". Je to zařízení, které pomocí pneumatického, elektrického nebo mechanického systému vyvažuje hmotnost břemene, čímž umožňuje jeho snadné zvedání a spouštění prakticky bez vynaložení síly ze strany operátora. Na rozdíl od manipulátoru neumožňuje složité řízení pohybu, ale zato vyniká jednoduchostí, rychlostí a velmi intuitivním ovládáním – operátor může břemeno doslova „vést rukou“. Hlavní rozdíl tedy spočívá v úrovni kontroly a složitosti pohybu. Zatímco manipulátor slouží pro přesné a vícerozměrné manipulace, balancér je ideální pro rychlé a jednoduché vertikální přemísťování s minimalizací fyzické námahy. (1)

Jedním z netradičních, ale stále častěji využívaných prvků v oblasti průmyslové manipulace je fluidní sval. Tento prvek se konstrukčně i funkčně liší od běžných pneumatických válců – jeho chování je měkké, přirozenější a do jisté míry připomíná biologický sval. Fluidní sval se skládá z pružného vnitřního elastomerového tubusu, který je po celé délce opleten vláknitou síťovinou (např. kevlarovým nebo nylonovým pletivem). Když se do svalu přivede stlačený vzduch, tak se sval zkracuje a generuje tahovou sílu. Pokud se tlak uvolní, sval se vrací zpět do své původní délky. Tato vlastnost – zkrácení při natlakování – dělá z fluidních svalů ideální prvek pro aplikace, kde je potřeba plynulý, poddajný a bezpečný pohyb. Na rozdíl od klasických válců nemá pevnou konstrukci, a proto je lehký, flexibilní a schopný dobře reagovat na vnější síly. Navíc jeho deformace při zatížení přirozeně generuje změnu vnitřního tlaku, což umožňuje využít jej nejen jako pohon, ale zároveň i jako senzor. V této práci se fluidní sval využívá jako nosný a zároveň měřicí prvek, který v kombinaci s tlakoměrem nebo pokročilým řídicím systémem umožňuje vytvořit jednoduchý a intuitivní systém pro manipulaci se zavěšeným břemenem. (2)

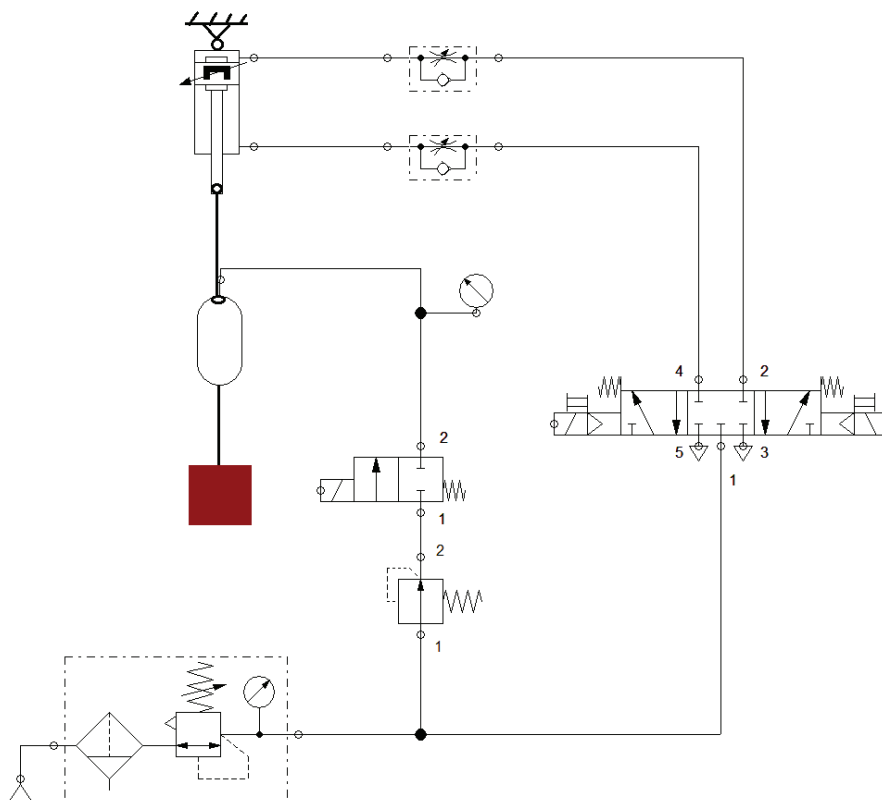
2 Využití fluidního svalu pro inteligentní balancér

Základní verze navrženého zařízení využívá jednoduchý, ale efektivní princip založený na měření tlaku v pneumatickém svalu. Pneumatický sval zde slouží jako nosný prvek, na kterém je zavěšeno břemeno přes běžný pneumatický válec. Při statickém zavěšení závaží se tlak v pneumatickém svalu ustálí na určité hodnotě, která odpovídá tzv. klidovému stavu.

Na výstup svalu je připojen tlakoměr (tlakový senzor), který nepřetržitě snímá aktuální tlak. Tento senzor je následně propojen s řídicí elektronikou, která porovnává naměřený tlak s předem nastavenou referenční hodnotou (setpointem). V okamžiku, kdy dojde ke změně působení síly – například uživatel zatáhne dolů za břemeno nebo ho začne nadlehčovat – tlak ve svalu se nepatrně změní.

Pokud tato změna přesáhne stanovený práh, elektronika vyšle signál do solenoidového ventilu, který se otevře. Tím se umožní pohyb pneumatického válce – buď směrem dolů (pokud se zatáhne), nebo nahoru (v případě odlehčení). Pohyb trvá po dobu, kdy tlak zůstává mimo toleranční pásmo. Jakmile se síla přestane měnit a tlak se vrátí blízko k setpointu, ventil se opět uzavře.

Celý systém tak umožňuje jednoduché a intuitivní ovládání pohybu břemene – bez tlačítek, nebo jiných prvků, kdy uživatel má obě ruce volné pro práci. Uživatel pouze působí silou a zařízení na to reaguje změnou tlaku.



Obrázek 1, návrh zapojení prototypu balancéru s fluidním svalu

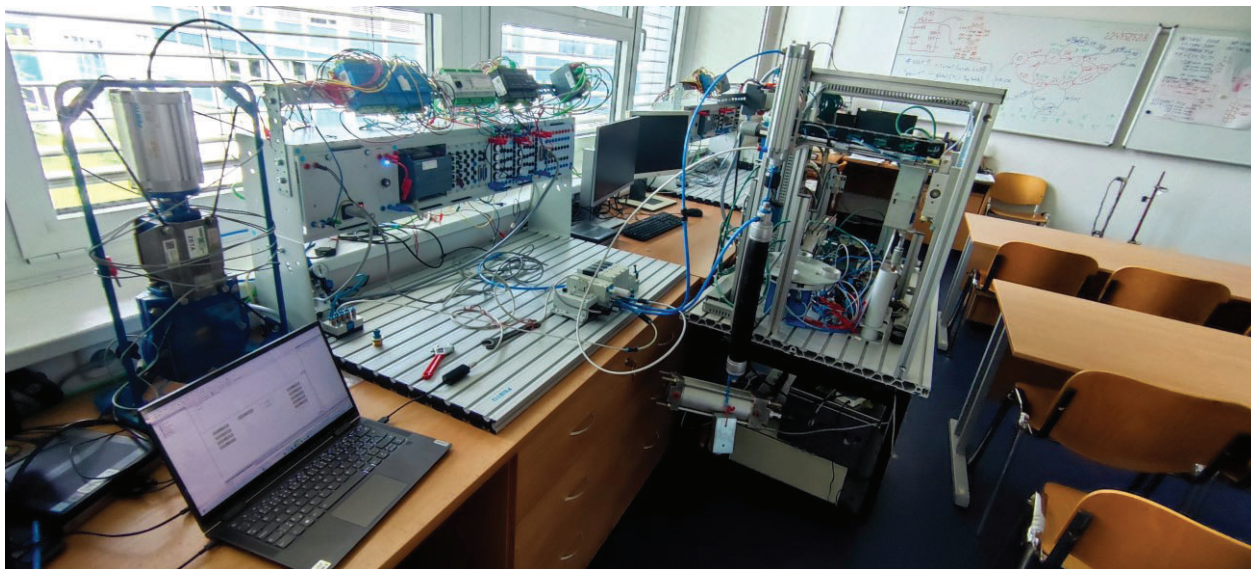
3 Využití Festo Motion Terminálu pro inteligentní balancér

Druhá, vylepšená verze zařízení využívá Festo Motion Terminál, který umožňuje pokročilé řízení na základě tlaku ve fluidním svalu a poskytuje mnohem jemnější odezvu než pouhé zapínání ventilu.

Na rozdíl od předchozího návrhu, kde bylo nutné manuálně plnit fluidní sval a nastavovat jeho referenční tlak, je nyní celý proces automatizován pomocí funkcí Festo Motion Terminálu. Zařízení samo zajišťuje počáteční plnění svalu a také uzavření pneumatického obvodu fluidního svalu před spuštěním polohování. Poté je dosaženo požadovaného klidového tlaku při zavěšení břemene. Tím se eliminuje potřeba zásahu obsluhy při kalibraci a zajišťuje se opakovatelnost a stabilita systému.

Velkým krokem vpřed je také to, že zařízení už nereaguje jen binárně („pohyb dolů – pohyb nahoru“), ale využívá lineární závislost rychlosti pohybu na velikosti síly působící na sval. Díky vysoké přesnosti snímání tlaku ve svalu a schopnosti Festo Motion Terminálu plynule ovládat škrcení ventilů (nejen je otvírat a zavírat), lze dosáhnout velmi jemného a přirozeného pohybu. Malá změna síly znamená pomalý pohyb, větší změna zase rychlejší reakci – podobně jako když člověk drží zavěšený předmět a lehce s ním manipuluje.

Celý systém tak působí intuitivně a ergonomicky. Operátor břemeno jednoduše nadlehčí nebo zatáhne – a zařízení na to plynule a přirozeně reaguje bez nutnosti stisků tlačítek nebo externího ovládání. Výsledkem je chytrý pneumatický balancér, který spojuje výhody jednoduchého balancéru a aktivně řízeného manipulátoru.



Obrázek 2, reálné zapojení systému s fluidním svalu a Festo Motion Terminálem

4 Závěr

Byly vyzkoušeny dvě varianty realizace s využitím fluidního svalu, princip založený na měření tlaku v pneumatickém svalu a rozšířený princip, který navíc využívá schopností Festo Motion Terminálu, díky kterému celý systém působí intuitivně a ergonomicky. Výsledkem je chytrý pneumatický balancér, který spojuje výhody jednoduchého balancéru a aktivně řízeného manipulátoru.

5 Poděkování

Děkuji firmě Festo za zapůjčení techniky a podporu při projektu. Děkuji paní Martináskové za vedení projektu.

6 Literatura

- [1] Zhang, W., Song, S., & Yoon, J. *A review of industrial assistive devices for material handling*. in: Robotics and Computer-Integrated Manufacturing, 2018.
- [2] Tondur, B., & Lopez, P. *Modeling and control of McKibben artificial muscle robot actuators*. in: Control Systems Magazine, 2000.

SOFTWARE PRO MONITORING VÝNOSU PLODŮ

Martin Vitoušek¹, Jakub Jura¹, Matouš Cejnek¹, Klára Scháňková², Luděk Laňar²

¹ Ústav přístrojové a řídicí techniky, České vysoké učení technické v Praze, Technická 4, Praha 6, martin.vitousek@fs.cvut.cz

² Výzkumný a šlechtitelský ústav ovocnářský Holovousy s.r.o., Holovousy 129, 508 01 Holovousy

Abstrakt: Článek prezentuje softwarový systém pro monitoring výnosu plodů, který je součástí širšího konceptu digitalizace ovocnářství a precizního zemědělství. Vytvářený Farming Management System (FMS) má za cíl zajištění efektivního sběru, ukládání a analýzy dat z různých zdrojů, včetně snímkování sadů a vinic mobilní platformou. Systém využívá moderní architekturu založenou na technologiích Python, FastAPI a PostgreSQL, Docker, přičemž je navržen jako modulární a snadno rozšiřitelný. Software dokáže zpracovávat velké množství heterogenních dat, vytvářet přehledné vizualizace a generovat reporty, které podporují rozhodování při plánování sklizně, probírky či dalších zásahů v sadech či vinicích. V dlouhodobém horizontu umožňuje sledovat trendy meziročně a zlepšovat predikce výnosu pomocí modelů strojového učení. Systém může fungovat nejen jako nástroj pro evidenci a analýzu, ale také jako základ pro systémy včasného varování a digitálního dvojčete sadu či vinice. Výsledkem je efektivnější, datově řízený a udržitelnější způsob pěstování ovoce.

Klíčová slova: Monitoring výnosu, automatizace ovocnářství, predikce sklizně, precizní zemědělství

Abstract: The article presents a software system for fruit yield monitoring, developed as part of a broader concept of digitalizing modern orchard management. The Farming Management System (FMS) is designed to enable efficient collection, storage, and analysis of data from various sources, including orchard and vineyard imaging. The system is built on a modern architecture utilizing Python, FastAPI, and PostgreSQL technologies and is designed to be modular and easily extensible. The software processes large volumes of heterogeneous data, generates clear visualizations, and produces reports that support decision-making in harvest planning, fruit thinning, and other management interventions. In the long term, it allows year-to-year trend tracking and improves yield prediction through machine learning models. The system serves not only as a tool for data recording and analysis but also as a foundation for early warning systems and digital twins of orchards and vineyards. The result is a more efficient, data-driven, and sustainable approach to fruit production.

Keywords: Yield monitoring, orchard automation, yield prediction, precision agriculture

1 Úvod

Současné ovocnářství prochází obdobím zásadní proměny. Roste tlak na efektivitu výroby, zajištění kvality produkce a udržitelnost hospodaření [1]. Aby mohli sadaři a vinaři činit správná rozhodnutí, potřebují mít k dispozici přesná a včasná data o stavu porostů i o vývoji úrody. Tradiční metody, založené na terénním monitoringu stromů či keřů, se ukazují jako nedostatečné – jsou časově náročné, nákladné a ve velkých výsadbách prakticky nezvládnutelné. Navíc neposkytují kontinuitu ani možnost dlouhodobého porovnávání, což komplikuje plánování a zpětnou analýzu.

Rozvoj moderních technologií, zejména v oblasti strojového vidění, datové analýzy a autonomních systémů, který započal před více než dvěma dekádami [2], otevírá cestu k digitalizaci zemědělství. Automatizovaný sběr dat z kamer, senzorů či lidarů umožňuje získat komplexní a přesný přehled o stavu sadu či vinice v reálném čase. Klíčovou roli však hraje i následná syntéza těchto dat, která je realizována pomocí metod umělé inteligence. Tyto metody umožňují zpracovávat rozsáhlé a heterogenní datové zdroje, odhalovat skryté vzorce a vztahy a přetvářet surová data na prakticky využitelné znalosti. Takto získané informace pak mohou sloužit nejen k predikci výnosů a efektivnímu plánování sklizně, ale také k včasnému odhalování problémů, jako je výskyt chorob či škůdců, a k hodnocení účinnosti jednotlivých zásahů během vegetační sezóny.

V rámci spolupráce ČVUT, VŠÚO a ČZU vzniká modulární software pro správu dat a podporu rozhodování v ovocnářství – Farming Management System (FMS). Tento systém je navržen tak, aby dokázal zpracovávat velké objemy heterogenních dat, poskytoval uživatelům intuitivní přístup k informacím a umožňoval generování přehledných reportů. Současně je připravován s ohledem na budoucí rozšíření o autonomní vozidla a drony, které zajistí pravidelný monitoring bez nutnosti lidské obsluhy.

Cílem článku je představit architekturu vyvíjeného softwaru, popsat jeho klíčové funkce a ukázat, jak může přispět k transformaci moderního zemědělství směrem k datově řízenému a udržitelnému hospodaření.

2 Implementace

2.1 Technologická architektura systému řízení zemědělství

Navržený FMS je založen na moderním a modulárním softwarovém stacku, který byl navržen pro dosažení vysokého výkonu, škálovatelnosti a udržitelnosti. Tato architektura reflektuje standardní postupy v oblasti vývoje enterprise aplikací.

2.2 Backend

Aplikační logika je implementována v programovacím jazyce *Python*, který byl zvolen pro svou efektivitu, čitelnost kódu, rozsáhlou nabídku různých knihoven, blízkost tématům strojového učení a datové analýzy. Pro vývoj webového rozhraní a *API* je využíván framework *FastAPI*. Jeho asynchronní architektura umožňuje efektivní zpracování paralelních požadavků, což přispívá k celkové robustnosti a rychlosti systému. Automatická generace dokumentace prostřednictvím specifikace *OpenAPI* usnadňuje integraci s ostatními komponentami a externími službami. Struktura backendu je navržena tak, aby ji bylo možné v budoucnu dále rozšiřovat implementací nových služeb a *API* routerů.

Pro abstrakci práce s databází a minimalizaci přímých *SQL* dotazů je použita *Python* knihovna *SQLAlchemy* [3], jež umožňuje využít při dotazování databáze přístup objektově relačního mapování (ORM, Object-Relational Mapping). Tento přístup umožňuje pracovat s databází objektově orientovaným způsobem a zjednodušit tak práci s databází pomocí *SQL* na práci s objekty. Zjednodušení práce s databází je důležité pro dlouhodobou udržitelnost softwaru, neboť vede programátora k dodržování již předepsané struktury a omezuje tak prostor pro vznik velkého množství jednoúčelových složitých dotazů.

Změny databázového schématu jsou řízeny prostřednictvím nástroje *Alembic* [3], který zajišťuje verifikované a kontrolované migrace. Migrační soubory je možné generovat automaticky na základě změn databázových modelů.

Pro analytické zpracování dat a generování výstupních reportů na pozadí jsou využívány známé *Python* knihovny jako je *NumPy* [4], *SciPy* [5], *Pandas* [6] a *Matplotlib* [7].

2.3 Perzistence dat

V rámci systému FMS jsou pro ukládání dat v současné době využívány dvě technologie: *PostgreSQL* a *MinIO*.

PostgreSQL je robustní relační open-source databázový systém, který slouží jako primární úložiště pro strukturovaná data v systému FMS. Jeho pokročilé funkce, jako je transakční integrita, podpora složitých datových typů a spolehlivost, z něj dělají ideální volbu pro správu kritických informací. V kontextu našeho systému se *PostgreSQL* využívá pro ukládání všech dat, která mají jasně definovanou strukturu, například záznamy o stromech (Tree), sklizních (Harvest), postřicích (Spraying) a dalších metadatech (Obrázek 1). Díky tomu je možné data snadno propojovat a analyzovat.

Na rozdíl od *PostgreSQL*, které je optimalizováno pro strukturovaná data, slouží *MinIO* k ukládání nestrukturovaných binárních dat. Jedná se o open-source server pro ukládání objektů, který je plně kompatibilní s *Amazon S3 API*. V systému FMS se používá jako specializované úložiště pro velké soubory, které by jinak neefektivně zatěžovaly primární databázi. Konkrétně se jedná o fotografie a videa spojená se stromy, které jsou spravovány databázovými modely. *MinIO* umožňuje efektivní práci s těmito velkými soubory, aniž by to mělo vliv na výkon hlavního databázového systému. Ukládáním binárních dat na *MinIO* je zajištěna vysoká dostupnost a škálovatelnost. Primární databáze ukládá pouze metadata k souborům a unikátní identifikátor, který slouží k napárování souboru v *MinIO* se záznamem v *SQL* databázi.

2.4 Komunikační rozhraní

Veškerá funkcionalita systému je dostupná prostřednictvím standardizovaného *REST API*. Toto rozhraní poskytuje sadu operací *CRUD* (Create, Read, Update, Delete - Vytvořit, Číst, Aktualizovat, Smazat) pro manipulaci s

datovými entitami, což zajišťuje konzistentní a předvídatelnou komunikaci mezi backendem a klientskými aplikacemi.

2.5 Uživatelské rozhraní a generování reportů

Uživatelské rozhraní systému je navrženo s důrazem na jednoduchost a přístup k důležitým informacím. Využívá se server-side rendering s *Jinja2*, což umožňuje backendu dynamicky generovat kompletní *HTML* stránky a poskytovat je uživatelům, což přispívá k rychlosti a minimálním nárokům na klienta.

Jednou z klíčových funkcí je generování komplexních reportů, které poskytují přehled o stavu sadu za daný rok. Tyto reporty jsou přístupné prostřednictvím specifického *API* endpointu `/report`, jenž umožňuje generování různých reportů na základě připojených parametrů dotazu. Například `/report?year=2022`, kde je využit volitelný parametr *year*, na základě kterého se generuje report za konkrétní rok, v tomto případě rok 2022. V případě produkčního nasazení pak bude cestě endpointu předcházet konkrétní hostname. V případě provozu na lokálním zařízení je možné využít adresu `http://localhost:8000/` za předpokladu, že byl ponechán port 8000 z výchozí konfigurace. Celá adresa by pak v tomto případě byla `http://localhost:8000/report?year=2022`.

Výstupní reporty jsou strukturovány do několika sekcí, zahrnujících vizualizace dat a tabulkové přehledy. Reporty obsahují grafy generované v reálném čase. Příkladem je matice korelací (Obrázek 3) zobrazující vztahy mezi různými metrikami. Dalším typickým prvkem je graf porovnávající skutečné hodnoty s predikcemi modelu (Obrázek 2). Nejdůležitější data jsou prezentována v tabulkách, které jsou organizovány podle řad a polí. Tabulky využívají barevné mapování, které vizuálně znázorňuje hodnoty v relativní škále, což usnadňuje rychlou interpretaci dat a odhalení vzorů (Obrázky 4 a 5).

2.6 Kontejnerizace a nasazení

Aplikace a veškeré její závislosti jsou kontejnerizovány pomocí technologie *Docker*. Pro orchestraci a správu více kontejnerů, které tvoří celý systém (jako je backend, databáze a úložiště objektů), se používá nástroj *Docker Compose*. Ten umožňuje definovat a spouštět celou aplikaci pomocí jediného konfiguračního souboru, což zajišťuje konzistentní chování systému v různých provozních prostředích (vývoj, testování, produkce) a výrazně zjednodušuje proces nasazování. Konfigurace systému je řízena externě pomocí souboru `.env`, což umožňuje snadné přizpůsobení aplikace bez nutnosti měnit kód. Všechny softwarové závislosti jsou spravovány v souboru `requirements.txt`. Provozní prostředí je díky těmto technologiím oddělené od hostitelského systému, což minimalizuje konflikty a zjednodušuje údržbu.

3 Závěr

Představený systém FMS přináší moderní a modulární přístup k monitoringu výnosu plodů a správě dat v ovocnářství. Díky využití otevřených technologií a kontejnerizace nabízí flexibilní a snadno rozšiřitelné řešení, které lze přizpůsobit různým typům sadů a vinic. Schopnost zpracovávat rozsáhlé a heterogenní datové soubory pomocí vyvinuté AI metodologie, generovat přehledné vizualizace a podporovat rozhodovací procesy představuje zásadní posun oproti tradičním metodám evidence a plánování.

FMS tak může výrazně přispět k efektivnějšímu a udržitelnějšímu hospodaření. V krátkodobém horizontu umožňuje lepší plánování sklizně a cílené zásahy do porostů, zatímco z dlouhodobého hlediska otevírá cestu k přesnější predikci výnosů, implementaci systémů včasného varování a tvorbě digitálních dvojčat sadů a vinic.

Další rozvoj systému bude směřovat k užší integraci pokročilejších metod strojového učení, rozšíření o data z autonomních platforem a k zapojení širší škály uživatelů. Díky tomu se FMS může stát nejen nástrojem pro monitoring, ale také klíčovou součástí digitální transformace ovocnářství a precizního zemědělství.

Poděkování

Software a článek byl podpořen projektem QK21010170 Nová koncepce sadů s nástupem technologií 4.0. Práce Martina Vitouška byla podpořena z grantu SGS24/125/OHK2/3T/12 Studentské grantové soutěže ČVUT.

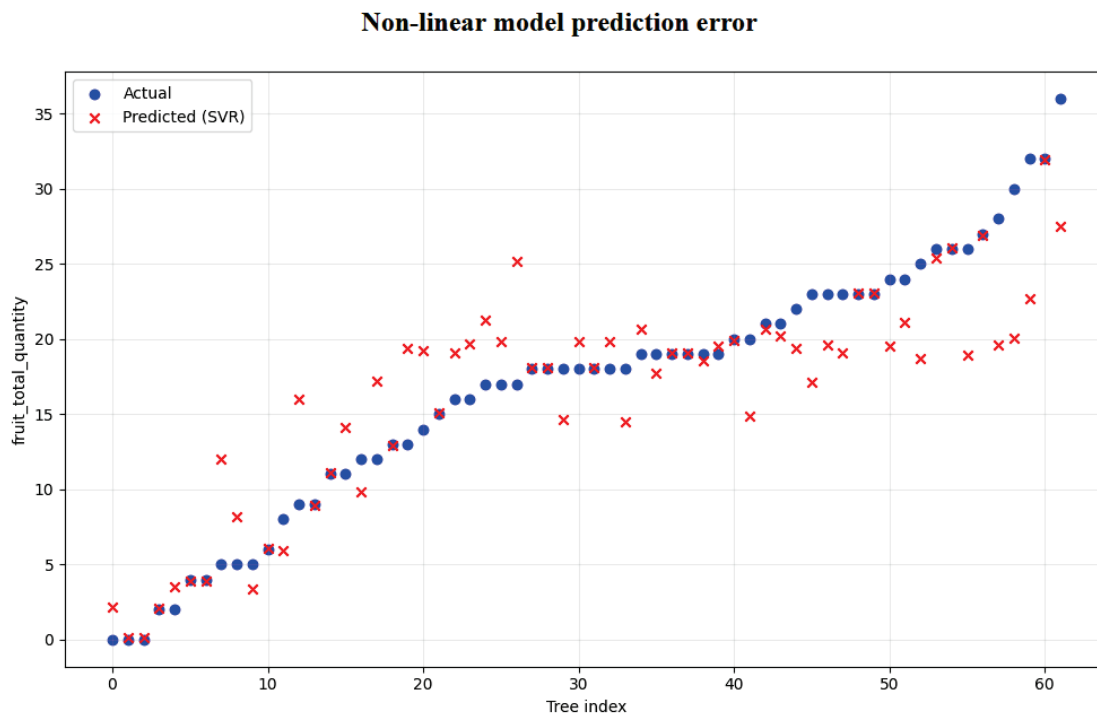
Literatura

- [1] LR Khot. Precision orchard management in digital agriculture era. In *XII International Symposium on Integrating Canopy, Rootstock and Environmental Physiology in Orchard Systems 1346*, pages 519–526, 2021.

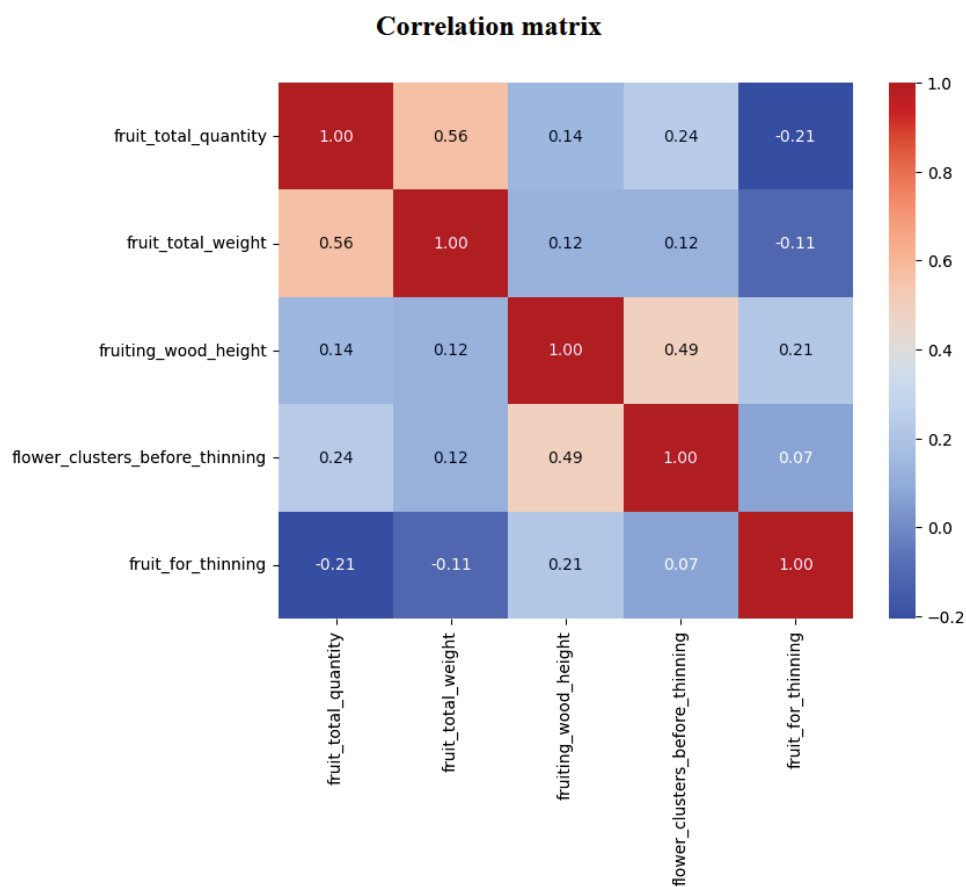
- [2] Yud-Ren Chen, Kuanglin Chao, and Moon S Kim. Machine vision technology for agricultural applications. *Computers and electronics in Agriculture*, 36(2-3):173–191, 2002.
- [3] Michael Bayer. Ssqlalchemy. In Amy Brown and Greg Wilson, editors, *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*. aosabook.org, 2012.
- [4] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.
- [5] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
- [6] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.
- [7] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.



Obr. 1: Diagram zobrazující strukturu databáze



Obr. 2: Ukázka nelineární prediktivního modelu nabitovaném na reálných datech z roku 2023.



Obr. 3: Ukázka korelační matice pro reálná data ze sadu. Z matice je možné vidět, jak silnou přímou souvislost měly v daném roce pozorované proměnné se velikostí sklizně.

Flowers before thinning (relative scale)

Tree row id: 1 (1 of 1)

Field	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1					0.27	0.22	0.39			0.73	0.16	0.33		0.42	0.48	0.00		0.47	0.64	0.48
2	0.47	0.38			0.00	0.55	0.17			0.16	0.55	0.50		0.25	0.05		0.09	0.19	0.30	0.19
3	0.48	0.50			0.28	0.31	0.05		0.58	0.42	0.06			0.28	0.61	0.36	0.39	0.20	0.12	
4		0.75	0.92	0.45		0.25	0.64	0.73			0.41		0.36		0.20			0.36	0.56	0.66
5		0.11	0.41	0.34		0.45		0.23	0.44			0.62	0.44	0.66				0.44	0.59	
6	0.69	0.47		0.58		0.36	0.08	0.47			0.48	0.38	0.30		0.44		0.02	0.05		0.22
7	0.50	0.94		0.16	0.83	0.41		0.58	0.34		0.22		0.62	0.47	0.16		0.67	0.16	0.88	
8	0.48	0.16	0.59		0.48	1.00	0.72													

Obr. 4: Tabulka zobrazující stromy u kterých byly zaznamenány počty květů před probírkou. Počty květů jsou normalizovány na škálu pro vyhodnocení nutné síly probírky.

Harvested fruit (absolute numbers)

Tree row id: 1 (1 of 1)

Field	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1					15	24	31			18	27	17		8	17	9		18	13	13
2	13	24			11	13	12			19	26	24		26	22		24	17	19	16
3	24	21			31	19	17		21	18	6			14	18	20	21	20	14	
4		21	39	21		11	12	21			14		13		10			18	20	21
5		13	12	15		21		22	23			19	23	26				20	29	
6	11	25		31		31	28	34			26	21	21		22		11	18		18
7	15	16		18	18	17		15	14		11		13	16	10		22	8	15	
8	18	20	20		17	19	23													

Obr. 5: Tabulka zobrazující absolutní počty sklizených plodů u stromů, pro které byla tato metrika kolektována během sklizně.



Editor: Vladimír Hlaváč
Název díla: Nové metody a postupy v oblasti přístrojové techniky,
automatického řízení a informatiky 2025
Vydalo: České vysoké učení technické v Praze
Zpracoval: Ústav přístrojové a řídicí techniky Fakulty strojní ČVUT v Praze
Kontaktní adresa: Ústav přístrojové a řídicí techniky, FS ČVUT v Praze, Technická 4, Praha 6
Tel.: +420 2 2435 9750
Počet stran: 57 Pořadí vydání: 1.
Vydáno dne 26. 9. 2025 jako pdf soubor na CD.
Dostupné na webových stránkách Ústavu přístrojové a řídicí techniky: <https://iat.fs.cvut.cz/nmp>
ISBN 978-80-01-07478-7