A NOVEL SAAS DEVELOPMENT PLATFORM FOR FLUID POWER STANDARD DRIVES

Heiko Baum[1]*

[1]*FLUIDON Gesellschaft für Fluidtechnik mbH, Jülicher Straße 338a, 52070 Aachen, Germany*

* Corresponding author: Tel.: +49 241 980 935 61; E-mail address: heiko.baum@fluidon.com

ABSTRACT

Today, it is state of the art to use 0D/1D simulations in the development of complex fluid technology drives, as this is the only way to evaluate the dynamic interactions of the system components. However, when modifying existing drives, developers often refrain from simulative validation because they consider the changes to be minor and/or rely on the positive experiences of the past. Consequently, design errors are only discovered in practical use, leading to production disruptions and costly troubleshooting.

A new type of SaaS development platform closes this gap and is aimed at companies with limited capacities and budgets. The platform enables companies to engage engineering service providers in creating customized design workflows. The paper illustrates the platform's application in a mobile hydraulic drive example, detailing the orchestration of pre- and post-processing tasks through a web browser interface. The familiar Excel spreadsheet used for static calculations continues to serve for parameterization source, maintaining the user's established design process while leveraging the precision of 0D/1D simulation. The simulation results are automatically converted into a format familiar to users, either as an Excel spreadsheet or a PowerPoint presentation for documentation and sales support.

Keywords: SaaS development platform, orchestrated design workflow, FMU, Spreadsheet driven simulation, automatic result reporting

1. INTRODUCTION

Today, the use of 0D/1D simulation is largely standard in the development of complex fluid power drives. This is the only way to test and evaluate the dynamic interactions of all installed components as a system. The buzzwords "digital twin" and "virtual commissioning" are not on everyone's lips for nothing. The situation is different with fluid power "standard drives". These drives, which are manufactured in small batches, can be found in mechanical and plant engineering, the raw materials industry or in mobile machinery or agricultural machinery. In terms of their total number, however, these drives form a quantitatively significant group of fluid power drives.

If a standard drive is to be modified at the customer's request or if a new product generation is planned, simulative validation of the changes is generally still not carried out. When asked about the reasons for this, the justification given is that the new design is only a minor modification or scaling of an existing drive for which there is extensive positive practical experience. The project engineer is then often satisfied with static calculations, e.g. in an Excel spreadsheet. However, dynamic interactions between the components are not considered in this way.

Practical experience shows that this is grossly careless. The simple scaling of components or even a change to the piping layout requested by the customer poses considerable risks regarding the dynamic behavior of the fluid power drive. If, for example, components need to be scaled up, their operating points will shift. It must then be considered that the operating behavior of many fluid power

components changes non-linearly when the operating point is shifted. Even if the same components are only spatially arranged differently or the piping is modified in another way, the dynamic interactions between the components change. These influences can only be considered insufficiently or not at all by quasi-static calculations.
If design errors are only recognized during practical use - i.e. too late - the consequences are no less serious than with complex systems. Production lines or machines in which the drives are installed come to a standstill and must be repaired at great expense. A simulation for troubleshooting, which is needed quickly and is therefore expensive, is often the only way to identify the problem and develop a solution.
Many of these problems could be avoided if 0D/1D simulation could be used more often in the design of supposedly simple fluid power drives. The simulation ensures that a new or reconfigured drive meets the customer's technical boundary conditions and requirements.
To achieve this goal, it is necessary to make 0D/1D simulation as accessible as possible for all those users who have yet little experience in simulation. What was missing until now was a development platform specialized for this user group.

2. A FLEXIBLY SCALABLE SAAS DEVELOPMENT PLATFORM

Cube could be just the right platform, as it has been deliberately designed to allow experienced users to prepare pre-configured workflows for the development, testing and marketing of fluid power drives. These workflows can then be used by less experienced users by giving them access to Cube. The workflows release users from routine tasks and allow them to concentrate better on their main task of interpreting the results. The following section outlines the basic procedure.

Each Cube account has a workspace in which users can create projects, either from scratch or by using templates. The most important features of this workspace are outlined in **Figure 1**.

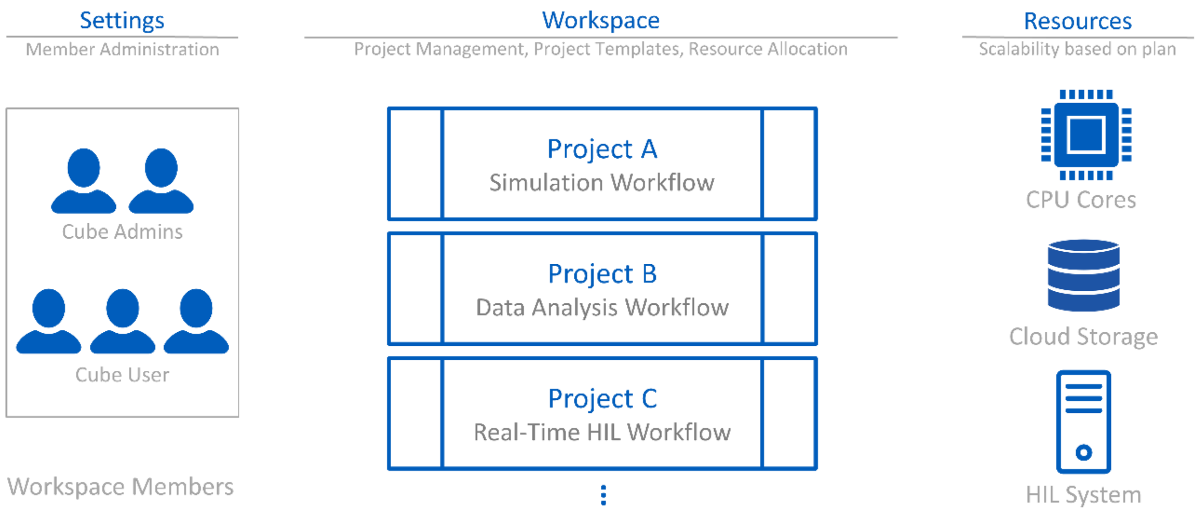


Figure 1: Workspace and Collaboration

In general, a wide range of tasks can be implemented in Cube projects. However, this paper focuses specifically on the presentation of workflows for the repetitive execution of tasks in the field of the simulative design of fluid power drives. For example, as shown in Figure 1, Project A could be a workflow for the simulation of fluid power drives, Project B could be a workflow for processing simulation results or measurement data from the test field, and Project C could be used to configure a simulation model on the Cube RT system required for the development of control systems running a real-time digital twin.
Every Cube user who has been invited to a workspace by an administrator is a member of that workspace. Members can work on their own projects or on a joint project. The workspace area gives

members an overview of the distribution of available resources. Resources are CPU cores, storage space on the cloud platform or Cube RT systems. The resources are generally located in a shared pool and can be allocated to individual projects as required.
As is usual with modern SaaS applications, a Cube account is always linked to a "plan". The plan describes the different pricing or subscription models available to users, what additional resources or additional services cost and what other services or support options are included. The basic "Cube Free" plan, in which a single workspace with a single project can be created, is free of charge.

Figure 2 summarizes the important benefits for a company when introducing the Cube SaaS application. A SaaS application is available immediately after the initial registration, which enables companies to introduce the new solution quickly. This also means a lower initial investment compared to traditional software development and deployment, as companies do not need to purchase expensive hardware or software licenses. Instead, they pay monthly or annual subscription fees.
SaaS applications are accessible via the Internet, which means that users can access their account at any time, from different locations and with different end devices. This is an advantage for modern company-specific working time models. In addition, thanks to the intuitive user interface typical of browser applications, users can find their way around the SaaS application quickly and without a great deal of training.

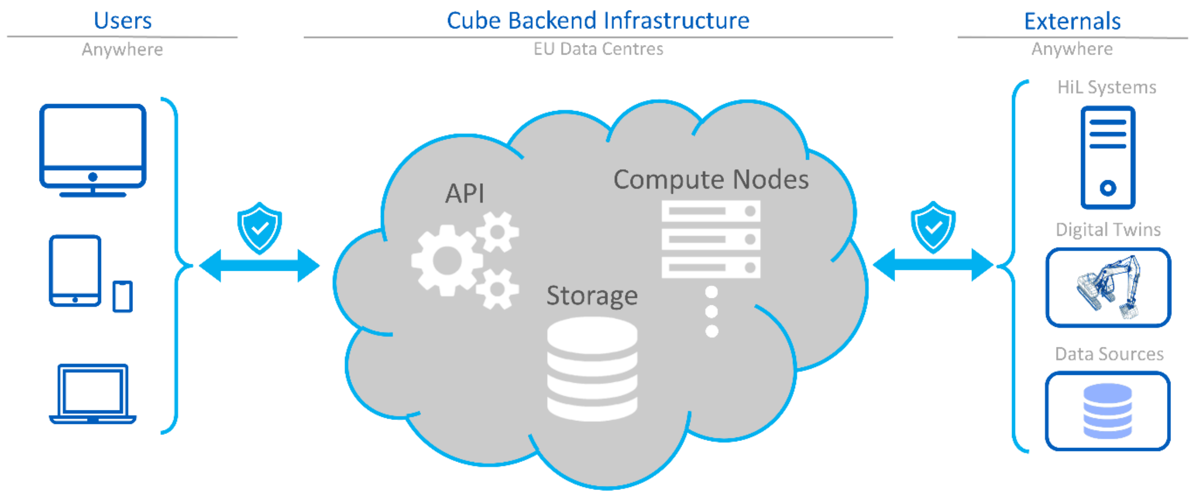


Figure 2: Flexibility, Scalability, and Security

Scalability and high availability are further advantages of SaaS applications (centre section of Figure 2). Storage space or computing power can be added at short notice by changing the plan, allowing companies to react flexibly to increased requirements without having to carry out expensive hardware upgrades.
In addition, SaaS applications guarantee high availability, as the operators of the data centers invest in redundancy and high-availability solutions. SaaS applications are also updated automatically, which relieves companies of the need to take care of maintenance and updates themselves and ensures that its users always have access to the latest functions and security patches.
Another important point in favor of SaaS applications is data security. In the case of Cube, the data is stored exclusively in European data centers. In addition to the security measures of the data centers, all areas of Cube that can be accessed via the Internet connection are secured separately in accordance with the state of the art. This measure provides companies with a level of security that generally goes beyond what would be practical for smaller companies or organizations to implement if they had to provide it themselves.

An interesting additional feature of the Cube development platform is shown on the right-hand side of Figure 2. Cube provides interfaces to access external devices or data sources from the SaaS applications. This is used, for example, to communicate with Cube's own RT system [1], access measurement data or exchange live data with other cloud applications.

The steps required to execute a workflow on Cube are described qualitatively in **Figure 3**. In preparation, the user has created a project in the workspace, set up a workflow in it, loaded any required data or simulation models and specified the CPU cores required for execution. The prepared project does not yet block any of the available computing resources. Scripts, simulation models and data only take up storage space in the cloud. Only when a compute node on which the workflow is to be executed is created by the user (Figure 3, top left), the specified number of CPU cores is allocated. Once the compute node is ready for use, the project is started.

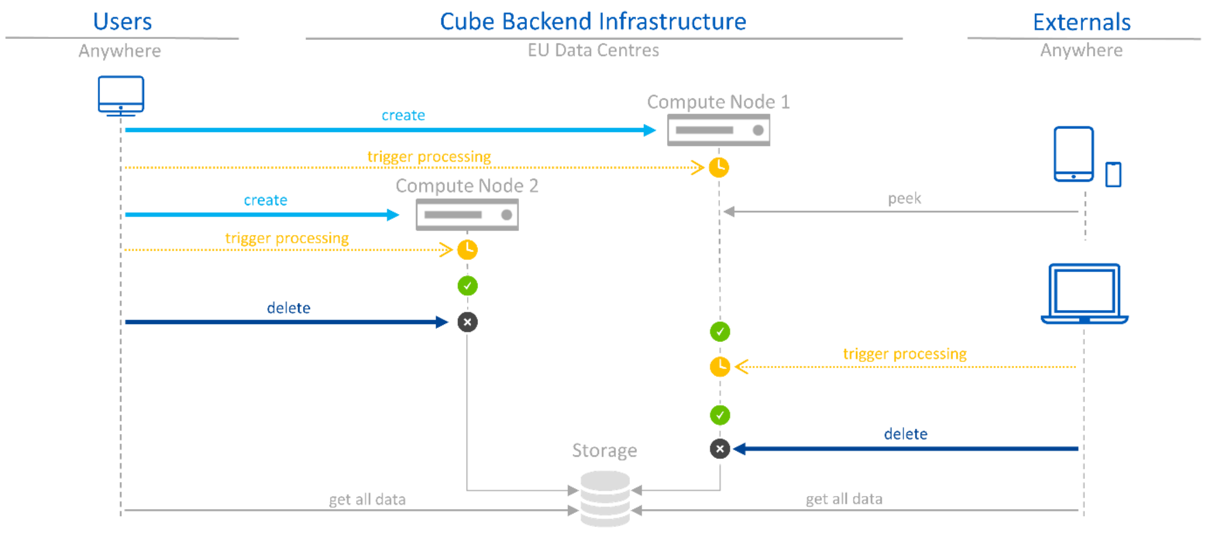


Figure 3: Computing Power on Demand

Both the start of the workflow and the interaction with the running workflow can be carried out using any end device, as illustrated once again in the right-hand section of Figure 3. For example, the tablet or mobile phone is used to check whether the execution has been completed. If necessary, the configuration of the workflow is now updated, new data is loaded into the workflow and a new execution is started.
If the results meet the requirements, the occupied computing resources can be released. To do this, the user deletes the compute node, and the CPU cores are once again available to all workspace members. The data generated during execution is automatically moved from the compute node to the cloud storage, where the user can continue to access it at any time.
If free computing resources are still available in the workspace during the execution of compute node 1, further projects can be executed, as shown in Figure 3 using the example of compute node 2.

3. DEVELOPMENT AS A TEAM ACHIEVEMENT

The possibility for several users to work together on projects in the Cube workspace offers companies the opportunity to utilize contemporary forms of development collaboration. One of these concepts, "development as a team achievement", and the resulting opportunities for companies in product development are presented below.

The concept outlined in **Figure 4** emphasizes the idea that the development of products, services or projects is most effectively achieved through the collaboration and coordinated work of a team rather than individual efforts. The concept thus reflects the reality of many engineering projects and the importance of teamwork in engineering. This is because engineering projects often require experts from different disciplines to work together to solve complex problems and drive technical innovation.

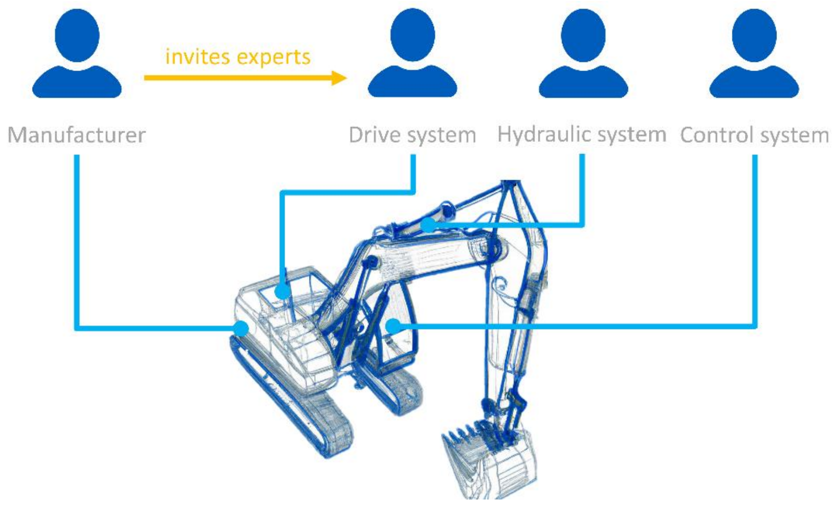


Figure 4: Engineering as a Team Achievement

The Cube development platform promotes this approach by automatically making the company's internal experts, as members of the workspace, members of the team. If necessary, the company can also integrate external service providers into the team by inviting them to join the workspace.

External service providers can leverage the Cube development platform to offer their solutions not only to a single customer but to multiple companies. (**Figure 5**).

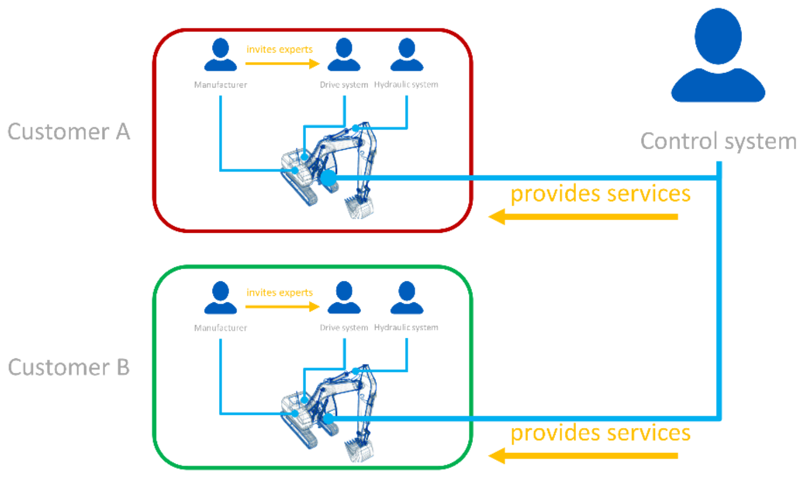


Figure 5: Engineering as a Service

A service provider for control systems can, for example, make his algorithms or simulation model available to companies. Cube provides interfaces to protect the specific expertise of the provider. This enables a business model in which service providers set up workflows, prepare Python modules, provide measurement data or create visualizations for Cube's animation task and make them available to Cube users. This can be done directly within an existing project or as a well-documented project template for the Cube development platform.

4. A SIMULATION WORKFLOW EXAMPLE

As a typical example of a ready-to-use and easily parameterizable Cube workflow, the simulation of a standard mobile hydraulic drive is presented in the following section. Such a hydrostatic drive (**Figure 6**) usually comprises at least one pump and several motors.

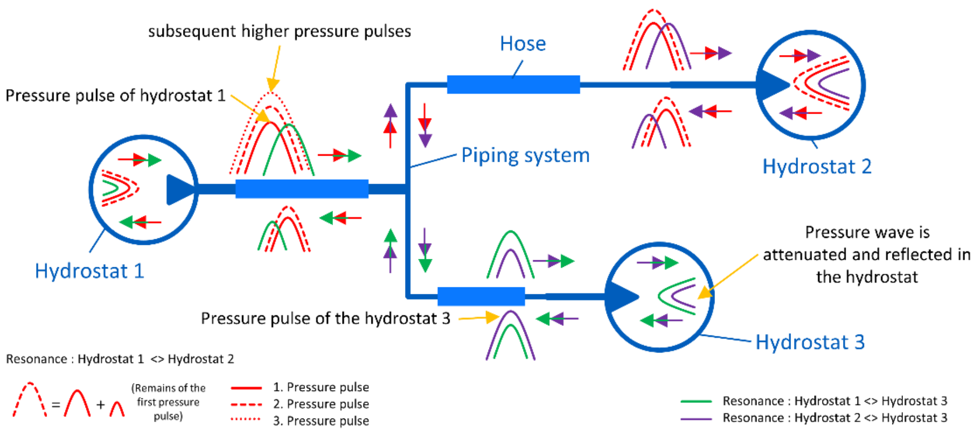
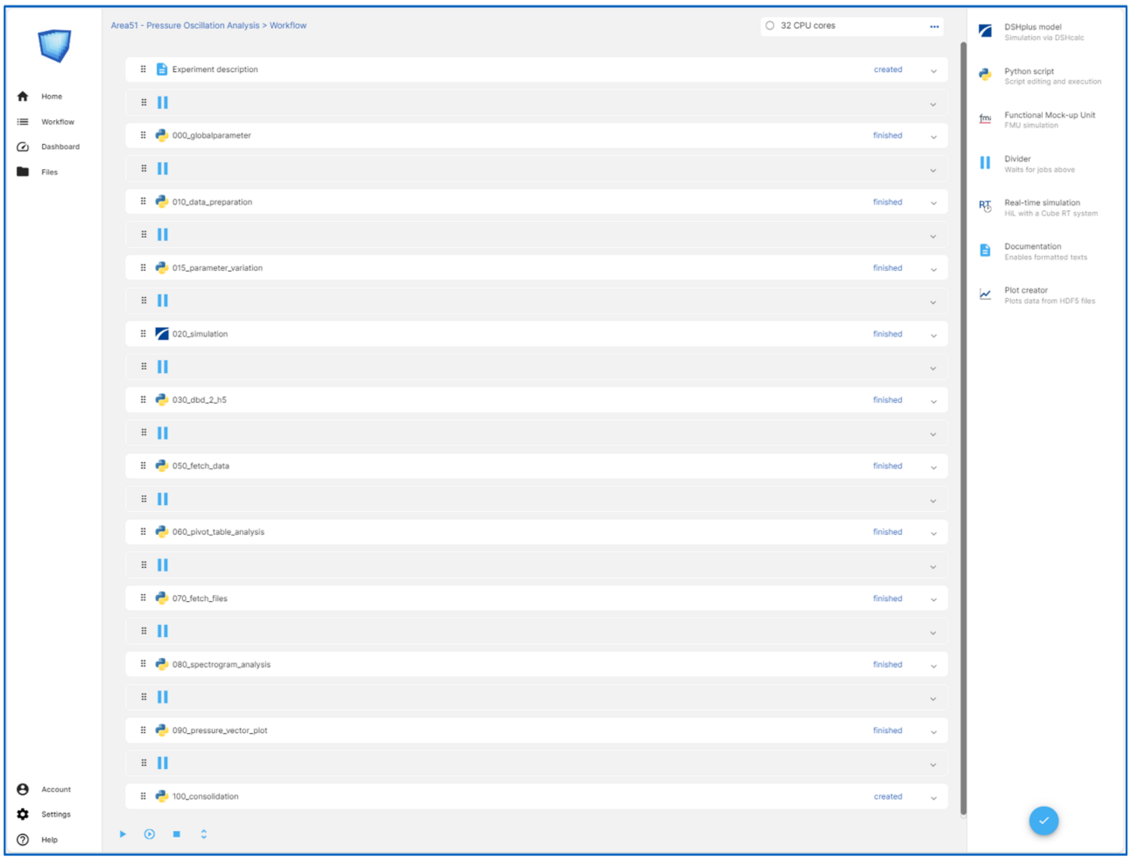

subsequent higher pressure pulses
Pressure pulse of hydrostat 1
Hose
Piping system
Hydrostat 2
Hydrostat 1
Pressure wave is attenuated and reflected in the hydrostat
Pressure pulse of the hydrostat 3
Hydrostat 3
Resonance : Hydrostat 1 <> Hydrostat 2
(Remains of the first pressure pulse)
1. Pressure pulse
2. Pressure pulse
3. Pressure pulse
Resonance : Hydrostat 1 <> Hydrostat 3
Resonance : Hydrostat 2 <> Hydrostat 3

Figure 6: Overview of the pressure oscillation situation of a hydrostatic drive train

A setup with only two motors was selected for the example. The hydrostats are connected by a piping system that, depending on the topology of the vehicle, branches out into several segments of different lengths, which in turn often consist of a combination of steel and hose lines. For the sake of simplicity, only the pressurised part of the piping system is considered.

To operate the drive in the optimum efficiency range or to enable special operating conditions, all hydrostats are adjusted individually. This results in a broadband pulsation excitation of the piping system, which can cause pressure oscillation problems not only between the source and the consumers, but also between the consumers themselves [2].

In the "classic" design of the piping system, it is almost impossible for reasons of time and resources to experimentally find the configuration that leads to the lowest pressure pulsation load on the piping system from the large number of possible variants. This is where the Cube workflow (**Figure 7**) comes into play to determine the best configuration for the piping system.



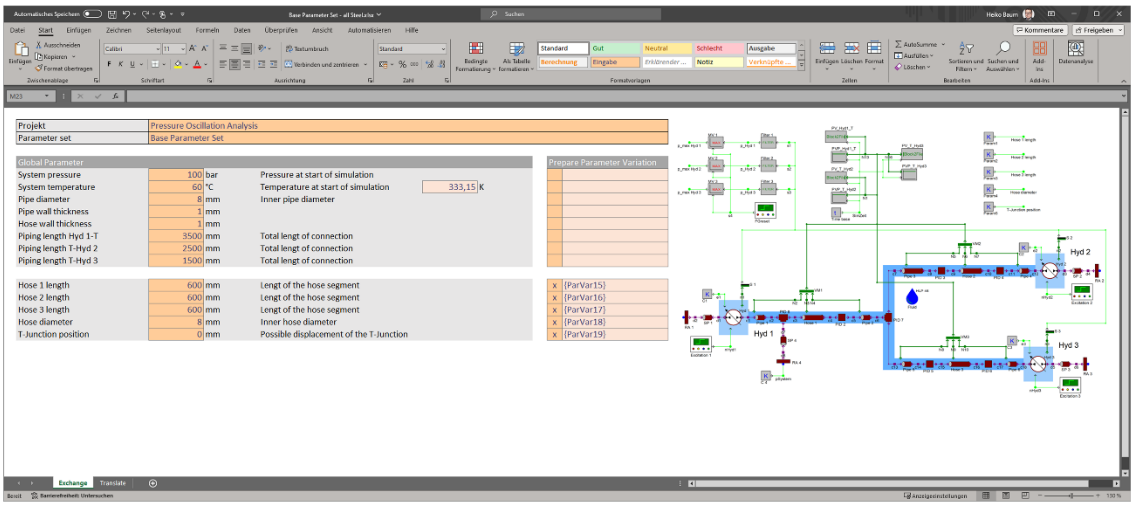
Area51 - Pressure Oscillation Analysis > Workflow

Experiment description	created
000_globalparameter	finished
010_data_preparation	finished
015_parameter_variation	finished
020_simulation	finished
030_dbd_2_h5	finished
050_fetch_data	finished
060_pivot_table_analysis	finished
070_fetch_files	finished
080_spectrogram_analysis	finished
090_pressure_vector_plot	finished
100_consolidation	created

Figure 7: Pressure oscillation workflow of the hydrostatic drive train

The workflow consists of eleven tasks, which are arranged in their logical working order. Except for task *020_simulation*, which includes the simulation model of the hydrostatic drive, all other tasks are described by Python scripts.
Such Python scripts can be freely created by the user, who can also incorporate (install) their own personal Python module library into the project. However, the Cube environment also offers a collection of predefined Python modules that users can integrate into their own scripts.
Task *020_simulation* of the example workflow is a dedicated DSHplus module that can import DSHplus simulation models directly [3]. If this task were replaced by Cube's FMI module, the workflow could run any Linux 64bit Functional Mock-up Unit (FMU) model generated by an FMI-compliant development tool [4]. Of course, the model parameters of such a FMU must be accessible to carry out the following parameter study!

At the beginning of a parameter study, it is necessary to define the limits within which the component parameters are varied and to determine how the results are to be evaluated. **Figure 8** shows an Excel workbook that is set up for this purpose. Excel was chosen as the data input medium to fulfil the requirement that the workflow should also be usable by non-simulation experts.

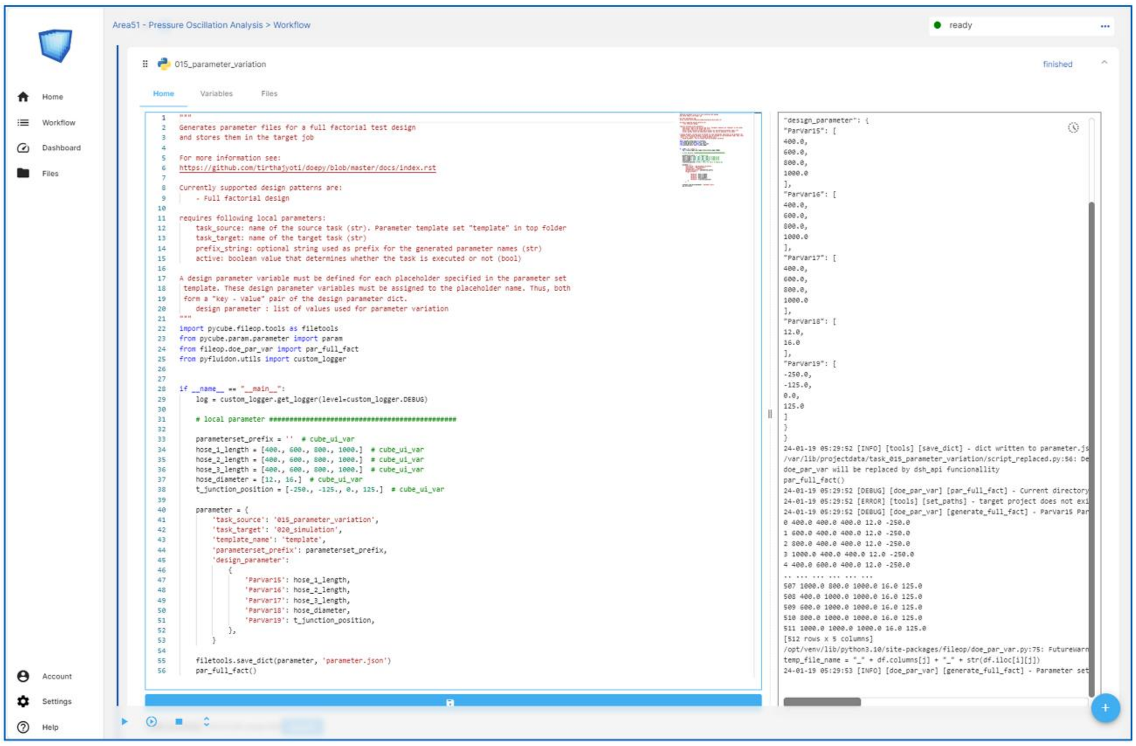


Projekt	Pressure Oscillation Analysis					
Parameter set	Base Parameter Set					
Global Parameter					**Prepare Parameter Variation**	
System pressure	100	bar	Pressure at start of simulation			
System temperature	60	°C	Temperature at start of simulation	333,15 K		
Pipe diameter	8	mm	Inner pipe diameter			
Pipe wall thickness	1	mm				
Hose wall thickness	1	mm				
Piping length Hyd 1-T	3500	mm	Total lengt of connection			
Piping length T-Hyd 2	2500	mm	Total lengt of connection			
Piping length T-Hyd 3	1500	mm	Total lengt of connection			
Hose 1 length	600	mm	Lengt of the hose segment		x	{ParVar15}
Hose 2 length	600	mm	Lengt of the hose segment		x	{ParVar16}
Hose 3 length	600	mm	Lengt of the hose segment		x	{ParVar17}
Hose diameter	8	mm	Inner hose diameter		x	{ParVar18}
T-Junction position	0	mm	Possible displacement of the T-Junction		x	{ParVar19}

Figure 8: Parameterization of the hydrostatic drive train

For installation space reasons, the overall lengths of the piping system must remain constant. However, the lengths of hoses 1 to 3 can be designed variably between 400 mm and 1000 mm. Formula relationships in the components automatically adjust the lengths of the remaining steel pipes accordingly. Other possible variations include internal hose diameters of 12 mm and 16 mm, as well as the positioning of the T-branch, where the pipe coming from the source (hydrostat 1) to the two consumers (hydrostat 2 and 3) can be moved from -250 mm to +125 mm from its starting position.
The medium pressure of the system is set to 100 bar. All three hydrostats are modelled as nine-piston variable displacement units. This means that not only the basic excitation frequency of the pump is considered, but also higher pump orders as excitation of the pipe system.

Figure 9 shows a screenshot of the Python script of task *015_parameter_variation* responsible for the variation of the design parameter. The source code view of the task is intentionally displayed to illustrate how straightforward it is to setup parameter variations using publicly available Python libraries.
The script lines 33 to 38 are used to enter the parameter range to be varied. The script generates a full-factorial experimental design [5] with 512 parameter sets and stores them directly in the parameter directory of the task *020_simulation*. The simulation module now starts a parallel calculation making use of all available 32 CPU-Cores and thus utilizing the full performance of the computer hardware.



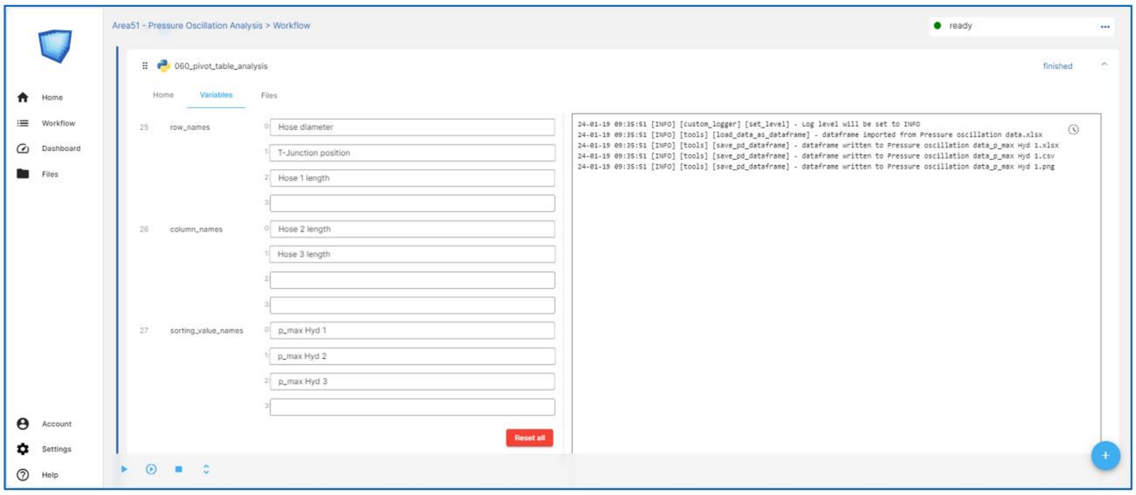
```python
"""
Generates parameter files for a full factorial test design
and stores them in the target job

For more information see:
https://github.com/tirthajyoti/doepy/blob/master/docs/index.rst

Currently supported design patterns are:
    - Full factorial design

requires following local parameters:
    task_source: name of the source task (str). Parameter template set "template" in top folder
    task_target: name of the target task (str)
    prefix_string: optional string used as prefix for the generated parameter names (str)
    active: boolean value that determines whether the task is executed or not (bool)

A design parameter variable must be defined for each placeholder specified in the parameter set
 template. These design parameter variables must be assigned to the placeholder name. Thus, both
 form a "key - value" pair of the design parameter dict.
    design parameter : list of values used for parameter variation
"""
import pycube.fileop.tools as filetools
from pycube.param.parameter import param
from fileop.doe_par_var import par_full_fact
from pyfluidon.utils import custom_logger


if __name__ == "__main__":
    log = custom_logger.get_logger(level=custom_logger.DEBUG)

    # local parameter ##################################################

    parameterset_prefix = ''  # cube_ui_var
    hose_1_length = [400., 600., 800., 1000.]  # cube_ui_var
    hose_2_length = [400., 600., 800., 1000.]  # cube_ui_var
    hose_3_length = [400., 600., 800., 1000.]  # cube_ui_var
    hose_diameter = [12., 16.]  # cube_ui_var
    t_junction_position = [-250., -125., 0., 125.]  # cube_ui_var

    parameter = {
        'task_source': '015_parameter_variation',
        'task_target': '020_simulation',
        'template_name': 'template',
        'parameterset_prefix': parameterset_prefix,
        'design_parameter':
            {
                'ParVar15': hose_1_length,
                'ParVar16': hose_2_length,
                'ParVar17': hose_3_length,
                'ParVar18': hose_diameter,
                'ParVar19': t_junction_position,
            },
    }

    filetools.save_dict(parameter, 'parameter.json')
    par_full_fact()
```

```
"design_parameter": {
"ParVar15": [
400.0,
600.0,
800.0,
1000.0
],
"ParVar16": [
400.0,
600.0,
800.0,
1000.0
],
"ParVar17": [
400.0,
600.0,
800.0,
1000.0
],
"ParVar18": [
12.0,
16.0
],
"ParVar19": [
-250.0,
-125.0,
0.0,
125.0
]
}
}
24-01-19 05:29:52 [INFO] [tools] [save_dict] - dict written to parameter.js
/var/lib/projectdata/task_015_parameter_variation/script_replaced.py:56: De
doe_par_var will be replaced by dsh_api funcionallity
par_full_fact()
24-01-19 05:29:52 [DEBUG] [doe_par_var] [par_full_fact] - Current directory
24-01-19 05:29:52 [ERROR] [tools] [set_paths] - target project does not exi
24-01-19 05:29:52 [DEBUG] [doe_par_var] [generate_full_fact] - ParVar15 Par
0 400.0 400.0 400.0 12.0 -250.0
1 600.0 400.0 400.0 12.0 -250.0
2 800.0 400.0 400.0 12.0 -250.0
3 1000.0 400.0 400.0 12.0 -250.0
4 400.0 600.0 400.0 12.0 -250.0
.. ... ... ... ... ...
507 1000.0 800.0 1000.0 16.0 125.0
508 400.0 1000.0 1000.0 16.0 125.0
509 600.0 1000.0 1000.0 16.0 125.0
510 800.0 1000.0 1000.0 16.0 125.0
511 1000.0 1000.0 1000.0 16.0 125.0
[512 rows x 5 columns]
/opt/venv/lib/python3.10/site-packages/fileop/doe_par_var.py:75: FutureWarn
temp_file_name = "_" + df.columns[j] + "_" + str(df.iloc[i][j])
24-01-19 05:29:53 [INFO] [doe_par_var] [generate_full_fact] - Parameter set
```

Figure 9: Python script for parameter variation with full factorial experimental design

However, the detailed results of the 512 simulations are initially of secondary interest to the user. First and foremost, the user wants to use the parameter variation to ensure that no disruptive resonance situation occurs in the drive train with the base piping design he has selected and to find out which of the possible line configurations has the lowest pressure pulsation amplitudes. The pressures at the hydrostat flanges serve as criteria for the evaluation.

The extraction of all data relevant for the subsequent analysis from the simulation results is carried out in task *050_fetch_data*. The values are consolidated so that, for example, only the maximum values of the pressures are taken during the simulation run. All values are saved in an Excel file and can therefore be further analysed offline or processed directly in the workflow.
Within the workflow this is done by the Python script of task *060_pivot_table_analysis* (**Figure 10**).



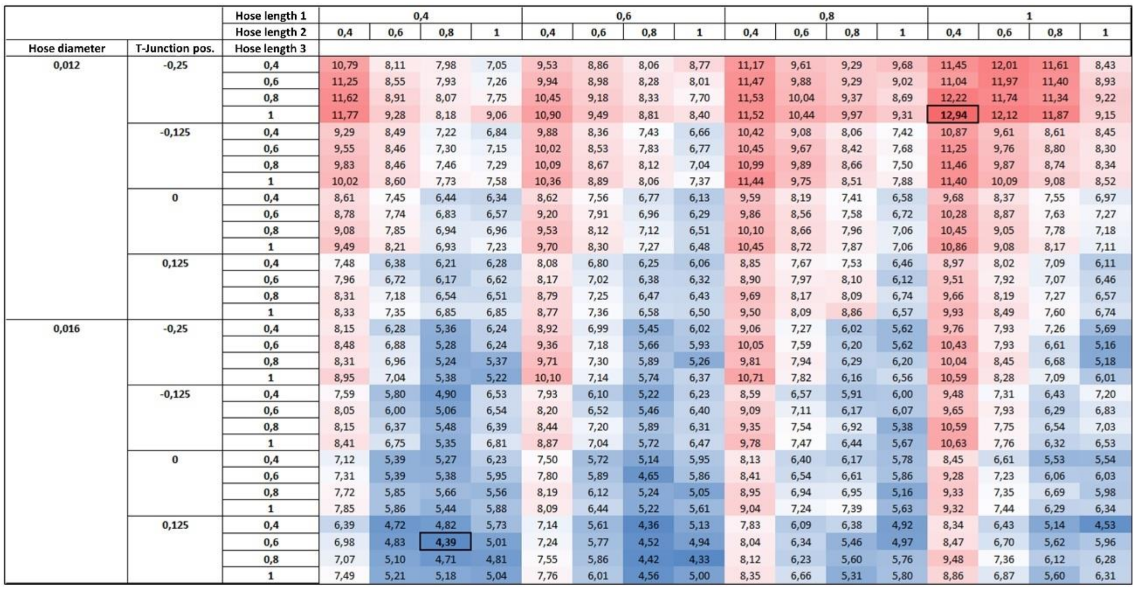
25	row_names	0	Hose diameter
		1	T-Junction position
		2	Hose 1 length
		3	
26	column_names	0	Hose 2 length
		1	Hose 3 length
		2	
		3	
27	sorting_value_names	0	p_max Hyd 1
		1	p_max Hyd 2
		2	p_max Hyd 3
		3	

```
24-01-19 09:35:51 [INFO] [custom_logger] [set_level] - Log level will be set to INFO
24-01-19 09:35:51 [INFO] [tools] [load_data_as_dataframe] - dataframe imported from Pressure oscillation data.xlsx
24-01-19 09:35:51 [INFO] [tools] [save_pd_dataframe] - dataframe written to Pressure oscillation data_p_max Hyd 1.xlsx
24-01-19 09:35:51 [INFO] [tools] [save_pd_dataframe] - dataframe written to Pressure oscillation data_p_max Hyd 1.csv
24-01-19 09:35:51 [INFO] [tools] [save_pd_dataframe] - dataframe written to Pressure oscillation data_p_max Hyd 1.png
```

Figure 10: Python script for pivot analysis

Instead of presenting the Python script editor, as done in Figure 9, the left part of the task window now shows the UI (user interface) section, in which the user defines the formatting of the pivot table, which is saved again as an Excel-file for further processing.

Figure 11 presents the resulting pivot table, which is an overview of the pressure amplitudes at hydrostat 3. Colour coding makes it easy to see which pipe system configuration causes higher and which lower pressure values.



		Hose length 1	0,4				0,6				0,8				1			
		Hose length 2	0,4	0,6	0,8	1	0,4	0,6	0,8	1	0,4	0,6	0,8	1	0,4	0,6	0,8	1
Hose diameter	T-Junction pos.	Hose length 3																
0,012	-0,25	0,4	10,79	8,11	7,98	7,05	9,53	8,86	8,06	8,77	11,17	9,61	9,29	9,68	11,45	12,01	11,61	8,43
		0,6	11,25	8,55	7,93	7,26	9,94	8,98	8,28	8,01	11,47	9,88	9,29	9,02	11,04	11,97	11,40	8,93
		0,8	11,62	8,91	8,07	7,75	10,45	9,18	8,33	7,70	11,53	10,04	9,37	8,69	12,22	11,74	11,34	9,22
		1	11,77	9,28	8,18	9,06	10,90	9,49	8,81	8,40	11,52	10,44	9,97	9,31	12,94	12,12	11,87	9,15
	-0,125	0,4	9,29	8,49	7,22	6,84	9,88	8,36	7,43	6,66	10,42	9,08	8,06	7,42	10,87	9,61	8,61	8,45
		0,6	9,55	8,46	7,30	7,15	10,02	8,53	7,83	6,77	10,45	9,67	8,42	7,68	11,25	9,76	8,80	8,30
		0,8	9,83	8,46	7,46	7,29	10,09	8,67	8,12	7,04	10,99	9,89	8,66	7,50	11,46	9,87	8,74	8,34
		1	10,02	8,60	7,73	7,58	10,36	8,89	8,06	7,37	11,44	9,75	8,51	7,88	11,40	10,09	9,08	8,52
	0	0,4	8,61	7,45	6,44	6,34	8,62	7,56	6,77	6,13	9,59	8,19	7,41	6,58	9,68	8,37	7,55	6,97
		0,6	8,78	7,74	6,83	6,57	9,20	7,91	6,96	6,29	9,86	8,56	7,58	6,72	10,28	8,87	7,63	7,27
		0,8	9,08	7,85	6,94	6,96	9,53	8,12	7,12	6,51	10,10	8,66	7,96	7,06	10,45	9,05	7,78	7,18
		1	9,49	8,21	6,93	7,23	9,70	8,30	7,27	6,48	10,45	8,72	7,87	7,06	10,86	9,08	8,17	7,11
	0,125	0,4	7,48	6,38	6,21	6,28	8,08	6,80	6,25	6,06	8,85	7,67	7,53	6,46	8,97	8,02	7,09	6,11
		0,6	7,96	6,72	6,17	6,62	8,17	7,02	6,38	6,32	8,90	7,97	8,10	6,12	9,51	7,92	7,07	6,46
		0,8	8,31	7,18	6,54	6,51	8,79	7,25	6,47	6,43	9,69	8,17	8,09	6,74	9,66	8,19	7,27	6,57
		1	8,33	7,35	6,85	6,85	8,77	7,36	6,58	6,50	9,50	8,09	8,86	6,57	9,93	8,49	7,60	6,74
0,016	-0,25	0,4	8,15	6,28	5,36	6,24	8,92	6,99	5,45	6,02	9,06	7,27	6,02	5,62	9,76	7,93	7,26	5,69
		0,6	8,48	6,88	5,28	6,24	9,36	7,18	5,66	5,93	10,05	7,59	6,20	5,62	10,43	7,93	6,61	5,16
		0,8	8,31	6,96	5,24	5,37	9,71	7,30	5,89	5,26	9,81	7,94	6,29	6,20	10,04	8,45	6,68	5,18
		1	8,95	7,04	5,38	5,22	10,10	7,14	5,74	6,37	10,71	7,82	6,16	6,56	10,59	8,28	7,09	6,01
	-0,125	0,4	7,59	5,80	4,90	6,53	7,93	6,10	5,22	6,23	8,59	6,57	5,91	6,00	9,48	7,31	6,43	7,20
		0,6	8,05	6,00	5,06	6,54	8,20	6,52	5,46	6,40	9,09	7,11	6,17	6,07	9,65	7,93	6,29	6,83
		0,8	8,15	6,37	5,48	6,39	8,44	7,20	5,89	6,31	9,35	7,54	6,92	5,38	10,59	7,75	6,54	7,03
		1	8,41	6,75	5,35	6,81	8,87	7,04	5,72	6,47	9,78	7,47	6,44	5,67	10,63	7,76	6,32	6,53
	0	0,4	7,12	5,39	5,27	6,23	7,50	5,72	5,14	5,95	8,13	6,40	6,17	5,78	8,45	6,61	5,53	5,54
		0,6	7,31	5,39	5,38	5,95	7,80	5,89	4,65	5,86	8,41	6,54	6,61	5,86	9,28	7,23	6,06	6,03
		0,8	7,72	5,85	5,66	5,56	8,19	6,12	5,24	5,05	8,95	6,94	6,95	5,16	9,33	7,35	6,69	5,98
		1	7,85	5,86	5,44	5,88	8,09	6,44	5,22	5,61	9,04	7,24	7,39	5,63	9,32	7,44	6,29	6,34
	0,125	0,4	6,39	4,72	4,82	5,73	7,14	5,61	4,36	5,13	7,83	6,09	6,38	4,92	8,34	6,43	5,14	4,53
		0,6	6,98	4,83	4,39	5,01	7,24	5,77	4,52	4,94	8,04	6,34	5,46	4,97	8,47	6,70	5,62	5,96
		0,8	7,07	5,10	4,71	4,81	7,55	5,86	4,42	4,33	8,12	6,23	5,60	5,76	9,48	7,36	6,12	6,28
		1	7,49	5,21	5,18	5,04	7,76	6,01	4,56	5,00	8,35	6,66	5,31	5,80	8,86	6,87	5,60	6,31

Figure 11: Pivot table of pressure at hydrostat 3

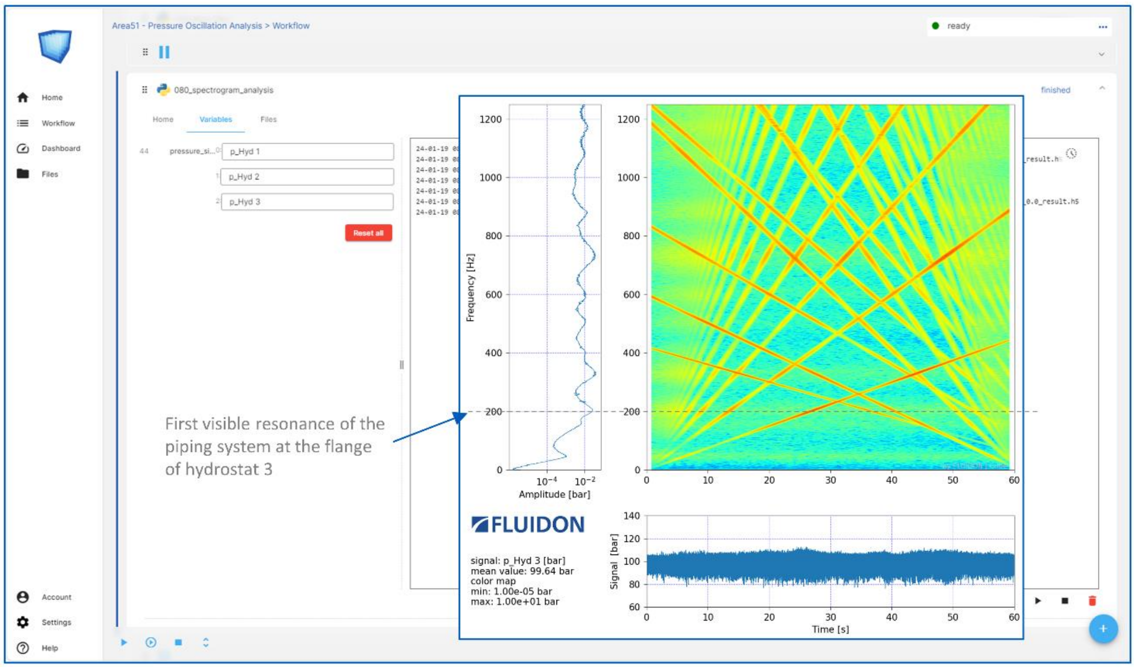
The assignment of the pressure values to a certain piping system configuration is done by the row and column sorting of the table. The pressure values are sorted line by line first by the two hose diameters, followed by sorting by the T-branch position and finally by hose length 1. The columns are sorted first by hose length 2 and then by hose length 3.

The pivot table shows, that the configuration with hose lengths 1 = 1000 mm, 2 = 1000 mm, 3 = 400 mm, diameter 12 mm and T-branch position -250 mm has the highest dynamic pressure pulsation amplitude at almost 13 bar and is therefore rather unsuitable, whereas, for example, the configuration with hose lengths 1 = 600 mm, 2 = 400 mm, 3 = 800 mm, diameter 16 mm and T-branch position +125 mm has a low dynamic pressure pulsation amplitude at approx. 4.4 bar and appears suitable.

However, this finding must still be cross-checked with the pressure pulsation amplitudes at the other two measuring points to finally evaluate whether the configuration provides the best overall result.

5. DETAILED ANALYSIS OF THE RESULTS

Of course, the developer can also take a closer look at the simulation results at any time. For this purpose, additional Python script tasks are added to the workflow as needed. **Figure 12** shows the results of the task *080_spectrogram_analysis*.



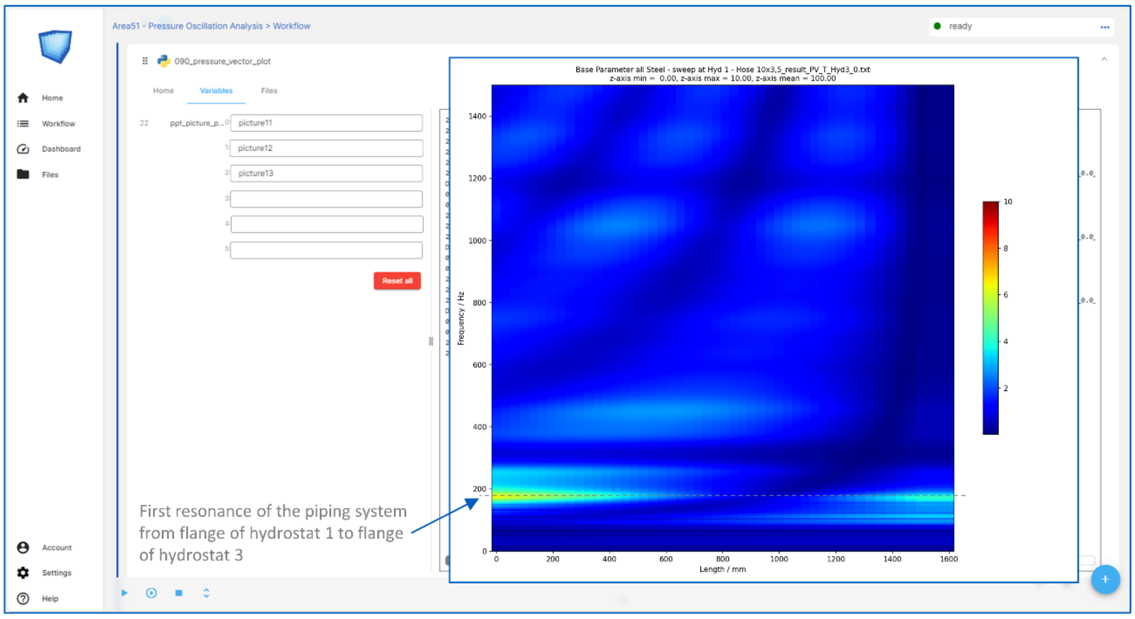
pressure_si... p_Hyd 1, p_Hyd 2, p_Hyd 3

First visible resonance of the piping system at the flange of hydrostat 3

signal: p_Hyd 3 [bar]
mean value: 99.64 bar
color map
min: 1.00e-05 bar
max: 1.00e+01 bar

Figure 12: Python script for spectrogram analysis

Data source for the diagram is again the pressure signal at the flange of hydrostat 3. At the yellow lines of the individual excitation orders of the hydrostats it can be clearly seen that all three hydrostats change their speed. Whenever one of the excitation frequencies "hits" a resonance frequency of the piping system, the colour changes from yellowish to reddish. The first piping system resonance is at approximately at 190 Hz. And there are more piping system resonances with even higher pulsation amplitudes that become visible at higher orders.
For now, it is of interest which pressure oscillation mode of the piping system is associated with this frequency. The pressure oscillation mode is identified by task *090_pressure_vector_plot* (**Figure 13**).



ppt_picture_p... picture11, picture12, picture13

Base Parameter all Steel - sweep at Hyd 1 - Hose 10x3,5_result_PV_T_Hyd3_0.txt
z-axis min = 0.00, z-axis max = 10.00, z-axis mean = 100.00

First resonance of the piping system from flange of hydrostat 1 to flange of hydrostat 3

Figure 13: Python script to create pressure vector diagrams

The resulting diagram visualizes the pressure oscillation situation along the piping's middle axis vs. excitation frequency. The visible oscillation mode at 190 Hz, indicated in Figure 13, is the fundamental λ/2 resonance frequency of a piping system with both ends closed. However, an investigation or detailed explanation would be another story. For a detailed description and interpretation of spectrograms and pressure vector plots, please refer to [2].

The compact exploration of the detailed time series results, along with the corresponding diagrams, demonstrates that the example workflow not only identifies whether the piping system configuration is favourable or unfavourable but also offers valuable insights into why a particular configuration is considered as such. This provides users with information regarding which parameters have the most significant impact on the system's dynamic behaviour.

6. SUMMARY

Using the example of a hydrostatic drive, it was shown how an easy-to-use, yet powerful simulation workflow can be set up within the Cube SaaS development platform. However, Cube is not limited to fluid power issues, but was designed as a platform for solving simulation or design issues from all technical areas. With the FMU plug-in, digital twins from various simulation tools can be integrated into the platform. Thanks to the flexible composition of the workflows from predefined plug-ins and Python scripts, Cube can be used to solve a wide range of problems.

REFERENCES

[1] VEL - a modular virtual commissioning environment for mobile machinery, 11th Colloquium Mobile Hydraulics, September 10, 2020, Karlsruhe, Germany, pp. 133 - 146

[2] Baum H.: Pressure Oscillation Analysis of Hydrostatic Drive Train, O+P, 06-2019, page 36 - 41

[3] DSHplus - Simulation of hydraulic and pneumatic systems in a mechatronic environment, https://www.fluidon.com/, last visited 01.2024

[4] FMI-standard - Functional Mock-up Interface for Model Exchange and Co-Simulation, https://fmi-standard.org, last visited 01.2024

[5] Statistical Design of Experiments, https://de.wikipedia.org/wiki/ Statistical_design_of_experiments, last visited 01.2024