Shams Elnawawi

Data Visualization for Large-Scale Model Predictive Control

May 2020

Abstract

THE AIM OF this research is to provide an interactive and user-friendly visualization tool geared towards Model Predictive Control (MPC) that provides an intuitive presentation of data, and incorporates constraint and gain data in a way that facilitates controller diagnosis. Despite extensive development in the performance aspects of MPC algorithms, there has been little development in the tools used to visualize and interact with MPC systems. Users of such systems may therefore experience difficulty in troubleshooting, especially when there are large numbers of variables. To alleviate this difficulty, we are outlining our design for an interface that incorporates principles from information visualization, computer science, and advanced process control (APC), in order to facilitate the process of controller diagnosis. Data related to the Linear Program (LP) layer of the control structure are used to develop visualization tools that aim to improve the quality of human-automation interactions. The tasks required in diagnosing a controller are analyzed and abstracted into generic computer science-based operations. These tasks are fed into a nested model for visualization design along with the relevant data, and both are combined to design the final visualization tool. Specific design decisions regarding the gain matrix and constraint data are discussed in detail, and a brief history of interactive heat maps is examined in order to contextualize the design decisions. The layout and interactivity of the final visualization tool are discussed in detail.

Contents

1 Introduction

Model Predictive Control (MPC) is an advanced feedback control algorithm that has become accepted as an industry standard in a variety of processes. While MPC implementation has advanced into a more complex and effective algorithm for controlling large multi-input multi-output (MIMO) systems, the availability of expertise in MPC commissioning, monitoring, and maintenance has become increasingly limited [1]. Since MPC is a model-based control strategy, understanding the underlying behaviour of the process model (in terms of steady-state gains, transfer functions, etc.) is essential for effective controller diagnosis.

While MPC has developed considerably since its inception, there has been an apparent lack of development in the tools used to visualize MPC controllers in action [1]. Often the packages that implement MPC, such as AspenTech's DMCPlus, use dense, tabular views to convey model information. Operators and engineers who use MPC with these types of views may experience difficulty in firstly getting acquainted with it, as well as with monitoring and troubleshooting the controllers when in use. DMCPlus, for instance, spreads out information pertaining to controller 'health' over many static tabs and sheets. For control systems with small numbers of variables, users may not experience any difficulties navigating through controller data. However, MPCs often involve more than 50 variables [2] - for instance, the MPC system used for the Fluid Catalytic Cracking (FCC) unit at the Parkland refinery in Burnaby has 44 manipulated and feedforward variables (MVs and FFs) and 64 controlled variables (CVs)[1], making navigation through model data in this way tedious. This is only one example, but the ideas presented here can be applicable to other MPCs that have similar functions and structures.

[1] MVs and FFs are the independent variables in the system, while CVs are the dependent variables. MVs and FFs are also known as 'inputs', and CVs as 'outputs'.

An improved user interface for MPC systems can therefore be of huge benefit to operators and engineers. In designing an improved visualization tool, the design must follow some methodology that incorporates evaluation in the design process. Using a rigidly defined

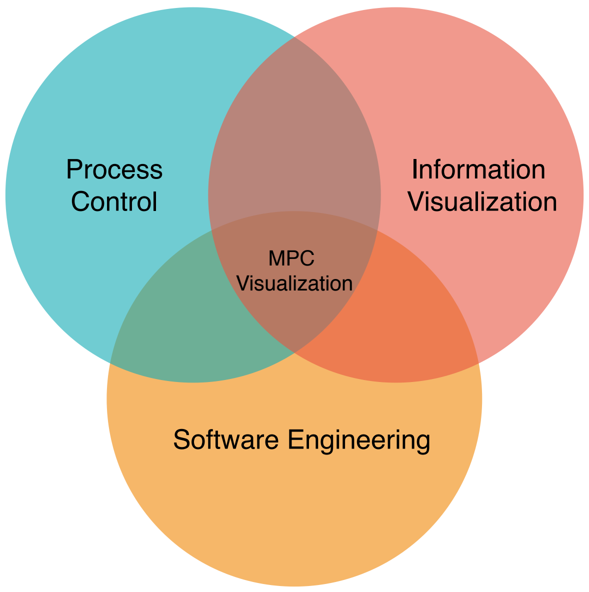


Figure 1.1: This research takes principles from information visualization and software engineering and applies them to the field of process control (specifically model-based predictive control) in order to try and create visualization tools that assist operators of such control systems in their day-to-day tasks. The tools presented here were developed with an emphasis on quality of human-automation interactions.

methodology means that design decisions are more justified, and are less prone to the pitfall defined by Munzner (2008) as 'Application Bingo': "where you pick a narrowly defined problem domain, a random technique, and then write an application paper with the claim of novelty for this particular domain-technique combination" [3]. The methodology chosen for designing the visualization is Munzner's "Nested Model for Visualization Design and Validation" [4] because it provides a clear approach to designing visualization applications, and because it provides useful suggestions for evaluating (or 'validating') any designed tools. These validation methods are incorporated into each layer of the model. The issues with current tools for MPC visualization are discussed, and alternatives are investigated.

1.1 *Problem Statement*

Small controllers can be more easily diagnosed using the aforementioned static displays, but larger controllers are often much more difficult to diagnose using the same tools. This idea forms the basis of the problem statement.

Based on initial research and discussion with industry partners, the project is driven by the following research questions:

- How can existing tools for MPC visualization be improved, with an emphasis on human-automation interactions?
- How can the process of controller diagnosis be made easier using

data visualization? Is it possible to automate this process using constraint data from the LP?

1.2 *Section Breakdown*

Section 1.3 outlines the background of MPC operation and how MPC systems solve optimization problems and control processes. This section also discusses the theory behind gain matrices in multivariable control and provides an overview of the FCC gain matrix at the Parkland refinery, then outlines the computational structure of commercial MPC systems. The principles of LP optimization follow, along with a generic process for MPC diagnosis. Section 1.4 provides a literature review that discusses previous work, then discusses the history of interactive heat maps in encoding data[2]. Chapter 2 discusses principles of data and information visualization and how humans understand visual representations of data. Section 2.1 explains the 'Nested Model' used to develop the visual tools in this project, and section 2.2 provides an overview of the 'Multi-level typology' for the development of our task analysis.

[2] Interactive heat maps have been used extensively in this project, which is why a detailed discussion of their usage in the past is important.

Chapter 3 discusses all results from this work: the task analysis for controller diagnosis is explained in section 3.1, the process for developing the final visualization tool is outlined in section 3.2, and the final visualization tool along with its features are shown in section 3.3. Chapter 4 explains the limitations of this work and considerations for future work.

1.3 *Background*

1.3.1 *MPC Concept*

MPC is an advanced form of automatic feedback control that is widely used in industry due to its robustness and compatibility with large MIMO systems [5]. MPC is a model-based algorithm, normally using a linear state space model that is often developed by applying analytical methods (such as IMC) to process data [3].

[3] Models calculated using techniques like IMC often need to be conditioned, because they can suffer from issues like ill-conditioning. Ill-conditioning can significantly hinder controller performance because it makes the degrees of freedom in the process unclear. Consequently, the controller may try to drive the process using degrees of freedom that appear to be available in the model, but are not actually available in the plant [6].

In addition to the process model, MPC systems also typically contain a *linear optimizer* (also known as Linear Program or LP). The optimizer aims to solve a dynamic optimization problem using an *objective function*, which is often an economic function of process variables that is to be minimized or maximized. At each iteration of the control system, the optimizer reads the process state relative to its reference trajectory, and uses this information to compute an optimized control plan over a specified *prediction horizon* (a defined number of time steps, during which the MPC tries to reach the process setpoint). Typical MPC operation is shown in Figure 1.2.

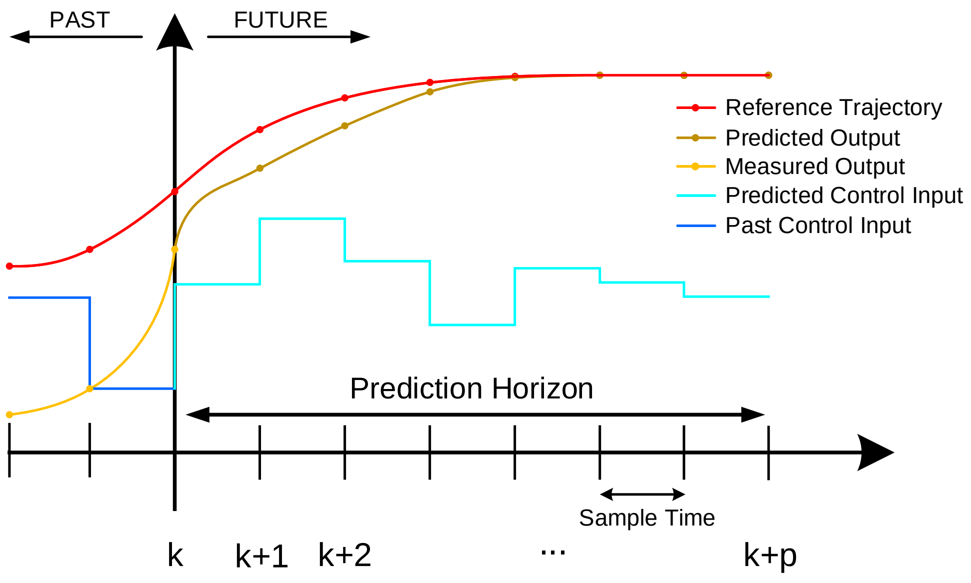


Figure 1.2: MPC concept for a single variable. Figure adapted from Bequette (2009) [7].

At any time $t = k$, the optimizer instantaneously computes the optimized control plan. At $t = k + 1$, the controller output y is set to $y(k + 1)$, and the real process response is observed. Since the real process response will likely deviate from the predicted output, the controller must take this new change into account by re-calculating its control plan at $t = k + 1$. The repetition of this process is the most basic MPC algorithm [8], [9].

1.3.2 *Gain matrix theory*

In multivariable control, CVs are affected by MVs and FFs both directly and indirectly, and these relationships can be summarized using the steady-state gain matrix (referred to as 'gain matrix' from now on). Indirect effects come in the form of MV and CV interactions, and these are harder to pinpoint but play an important role in multivariable control. The gain of a given MV/CV pair is calculated by quantifying the response of the CV to a change in the MV, while holding all other MVs constant, as defined as in equation 1.1.

$$K_{ij} = \left. \frac{\partial CV_i}{\partial MV_j} \right|_{MV_k} \tag{1.1}$$

Where K_{ij} is the gain value and MV_k represents all other MVs, which are held constant when quantifying the gain[4]. This is also

[4] Classical control texts use u to denote inputs (MVs) and y to denote outputs (CVS). In this thesis, we are following industry convention by using MV and CV instead of the control convention of using u and y.

known as 'open-loop' gain because all other MVs are held constant[5]. Equations 1.2 and 1.3 show the mathematical representation of the gain matrix.

[5] The 'closed-loop' gain, which is used to calculate Relative Gain Array (RGA), quantifies the same relationship without holding other MVs constant. Rather, the closed-loop gain requires other CVs to be held constant.

$$\begin{bmatrix} \Delta CV_1 \\ \Delta CV_2 \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} \Delta MV_1 \\ \Delta MV_2 \end{bmatrix} \tag{1.2}$$

Where K_{ij} represents the steady-state gain between CV_i and MV_j. The following equation shows a sample gain matrix for a 2x2 system:

$$\begin{bmatrix} \Delta CV_1 \\ \Delta CV_2 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} \Delta MV_1 \\ \Delta MV_2 \end{bmatrix} \tag{1.3}$$

By looking at this gain matrix, it is easy to understand the steady-state dynamics of this hypothetical plant; a unit increase in MV_1 causes a two-unit increase in CV_1 and a unit decrease in CV_2. Similarly, a unit increase in MV_2 has no direct effect on CV_1 and causes a three-unit increase in CV_2. While the increase in MV_2 has no direct effect on CV_1, it causes a threefold increase in CV_2. In order to maintain plant control, the controller may increase MV_1, which would then cause CV_1 to change. This analysis of the steady-state changes in the plant is easily done for such a small system and requires no additional tools for visualization or analysis; if CV_1 is changing for an unknown reason, users need to investigate a maximum of three variables in order to find the root cause.

1.3.3 *Parkland gain matrix*

For systems such as those in the Parkland refinery - where there are 100+ variables in total - the analysis performed in section 1.3.2 is not as easy. To illustrate this idea, table 1.1 shows a section of the FCC gain matrix in Parkland, with seven MVs and 28 CVs[6]. All tables that represent the FCC gain matrix at Parkland do not show the real gain values for confidentiality purposes. We normalized the gain matrix so that all values lie between 0 and 1, using linear scaling. This computation is shown in equation 1.4.

[6] The FCC gain matrices plotted in tables 1.1-1.2 show only seven MVs and 28 CVs because most of the nonzero gain values are quite small, so it became difficult to read them when more MVs and CVs were packed in. The 28×7 matrix is estimated to be large enough to illustrate the points discussed.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1.4}$$

Where:

- x = raw gain value
- x' = normalized gain
- x_{min} = lowest gain value

- x_{max} = highest gain value

At a glance, the gain matrix in table 1.1 is much more complicated than the sample matrix shown before[7]. Performing a similar analysis as in section 1.3.2 to determine how changes in MVs will affect CVs and other MVs is not as feasible. Engineers may want to analyze how changes in MV-1 affect the operating point of the plant and would need the gain matrix in order to do so. Table 1.2 shows the same gain matrix with the column for MV-1 highlighted.

[7] The gain matrices in tables 1.1-1.2 are plotted using CVs as rows and MVs as columns. The convention for DMCPlus plots MVs as rows, but the layout here is more intuitive, as it is a better representation of equations 1.2-1.3.

	MV-1	**MV-2**	**MV-3**	**MV-4**	**MV-5**	**MV-6**	**MV-7**
CV-1	0.004803	0	0	0	0	0	0
CV-2	0	0	0.004803	0	0	0	0
CV-3	0	0.001463	0	0	0	0	0
CV-4	-0.00438	1.22E-05	-0.00012	0.020056	0.000346	0	-0.00014
CV-5	-0.10345	0.000289	-0.00278	0.696379	0.012007	0	-0.0032
CV-6	-0.0199	0	-0.00072	0.348189	0.006003	-0.05276	0
CV-7	0	0	0	0.288496	0	0.005311	0
CV-8	0	-9.58E-05	0.002951	0.025905	0.000447	0.035774	0.001954
CV-9	-7.34E-05	0	0	0.004496	0	-8.53E-05	0
CV-10	0	0	0	0.004803	-7.23E-05	0	0
CV-11	0	0	0	0	0.009131	0.011408	0
CV-12	0	0	-0.00021	0	0	-0.00388	0
CV-13	-0.00053	0	-0.00012	0	0	0.004997	0
CV-14	0.125636	0	0.002094	-0.30362	-0.00523	0	-0.00871
CV-15	0.086447	0	0.001441	-0.13928	-0.0024	0	-0.00871
CV-16	0	0	6.92E-05	-0.00669	-0.00012	0.000367	8.02E-05
CV-17	0.011321	0	0.000487	0	0	0	0.004066
CV-18	0.035626	0	0.004023	0	0	0	0
CV-19	0.013472	0	0.001234	0	0	0	0
CV-20	0.14141	0	0.015968	0	0	0	0
CV-21	0.124814	0	0.014094	0	0	0	0
CV-22	-0.03572	0	0	0	0	0	0
CV-23	-0.08371	0	0	0	0	0	0
CV-24	-0.02888	0	0	0	0	0	0
CV-25	-0.13894	0	0.001709	0	0	0	0
CV-26	-0.03057	0	0.000376	0	0	0	0
CV-27	0.010248	0	7.85E-05	0	0	0	0
CV-28	0.048413	0	0.005467	0	0	0	0

Table 1.1: 28×7 section of the FCC gain matrix at Parkland; the full gain matrix has 44 independent variables (MVs/FFs) and 64 CVs. Only seven MVs are displayed because the normalized gain values are all quite small, so adding more MVs made it difficult to read each of the gain values.

Reading the gain matrix as it is shown in table 1.2 provides some potentially useful information; if an operator were to increase MV-1 by one unit, CV-1 would increase by 0.004803 units, CV-4 would decrease by 0.00438, CV-5 by 0.10345, CV-6 by 0.0199, and so on. The direct effects of MV-1 on the dependent variables can be determined with some analysis, but determining how interactions between MVs affect the CVs is extremely difficult in this way. For example, the same unit increase in MV-1 causes an increase of 0.125636 in CV-14. To maintain control of CV-14, the controller may reduce MV-3 by

	MV-1	MV-2	MV-3	MV-4	MV-5	MV-6	MV-7
CV-1	0.004803	0	0	0	0	0	0
CV-2	0	0	0.004803	0	0	0	0
CV-3	0	0.001463	0	0	0	0	0
CV-4	-0.00438	1.22E-05	-0.00012	0.020056	0.000346	0	-0.00014
CV-5	-0.10345	0.000289	-0.00278	0.696379	0.012007	0	-0.0032
CV-6	-0.0199	0	-0.00072	0.348189	0.006003	-0.05276	0
CV-7	0	0	0	0.288496	0	0.005311	0
CV-8	0	-9.58E-05	0.002951	0.025905	0.000447	0.035774	0.001954
CV-9	-7.34E-05	0	0	0.004496	0	-8.53E-05	0
CV-10	0	0	0	0.004803	-7.23E-05	0	0
CV-11	0	0	0	0	0.009131	0.011408	0
CV-12	0	0	-0.00021	0	0	-0.00388	0
CV-13	-0.00053	0	-0.00012	0	0	0.004997	0
CV-14	0.125636	0	0.002094	-0.30362	-0.00523	0	-0.00871
CV-15	0.086447	0	0.001441	-0.13928	-0.0024	0	-0.00871
CV-16	0	0	6.92E-05	-0.00669	-0.00012	0.000367	8.02E-05
CV-17	0.011321	0	0.000487	0	0	0	0.004066
CV-18	0.035626	0	0.004023	0	0	0	0
CV-19	0.013472	0	0.001234	0	0	0	0
CV-20	0.14141	0	0.015968	0	0	0	0
CV-21	0.124814	0	0.014094	0	0	0	0
CV-22	-0.03572	0	0	0	0	0	0
CV-23	-0.08371	0	0	0	0	0	0
CV-24	-0.02888	0	0	0	0	0	0
CV-25	-0.13894	0	0.001709	0	0	0	0
CV-26	-0.03057	0	0.000376	0	0	0	0
CV-27	0.010248	0	7.85E-05	0	0	0	0
CV-28	0.048413	0	0.005467	0	0	0	0

Table 1.2: FCC gain matrix in table 1.1 highlighting the column of MV-1 and the row of CV-7. The highlighted column shows the steady-state open-loop changes in CVs caused by a unit increase in MV_1. The highlighted row shows how CV_1 is affected by unit changes in all seven MVs.

$\frac{0.125636}{0.002094} = 59.99$, causing CV-2 to drop by $0.004803 \cdot 59.99$, and so on. Similarly, analyzing how one particular CV is affected by the MVs in the system yields an equally complex workflow.

Following a similar procedure, CV-7 increases by 0.288946 with an open-loop unit increase in MV-4 and by 0.005311 with a unit increase in MV-6. Since there are only two non-zero entries in this example, it is much easier to perform this single-input single-output (SISO) analysis on CV-7, but the interaction parameters remain difficult to quantify. This effect is exacerbated in the real FCC gain matrix, which has 37 more independent variables and 36 more CVs.

1.3.4 *Commercial MPC structure*

Typically, commercial MPC packages like DMCPlus involve more than just the MPC algorithm itself. The MPC algorithm is used to drive the plant to a specified operating point, and this is accomplished using a dynamic control plan. This operating point is not defined within the MPC algorithm, rather it is provided to it as a setpoint by the LP – a linear optimizer[8] that calculates the optimal operating point based on economic objectives instead of performance objectives. In this way, commercial MPC systems typically have a cascade-control structure, as shown in figure 1.3.

[8] The LP comprises the optimization step in MPC operation. This may also be accomplished using a Quadratic Program (QP), which uses quadratic optimization to solve the problem.

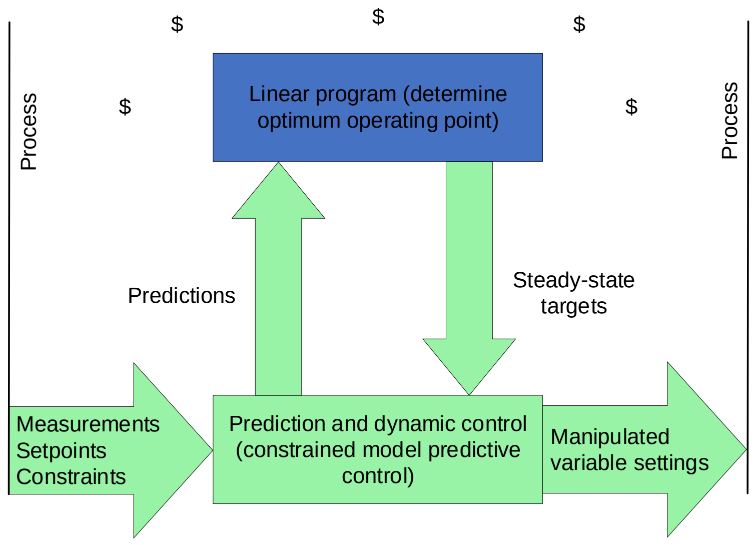

$
$
$
$
$
Process
Linear program (determine optimum operating point)
Process
Predictions
Steady-state targets
Measurements Setpoints Constraints
Prediction and dynamic control (constrained model predictive control)
Manipulated variable settings

Figure 1.3: Typical MPC package control structure (adapted from Sorensen et al). The blue section represents the steady-state economic optimization problem, while the green section represents the dynamic control problem.

Figure 1.3 shows that controlling a plant is achieved by solving two sub-problems:

1. The steady-state optimization problem, where the LP takes MPC predictions and economic targets and calculates the steady-state targets of the plant using an economic objective function;

2. The dynamic control problem, where the MPC takes the steady-state targets from the LP and drives the plant to the defined operating point.

The question of how a control system solves the 'dynamic control problem' relates to the specific control algorithm chosen in that plant. One such algorithm could involve using a series of Proportional-Integral-Derivative (PID) control loops with decoupling, where the PID controller moves MVs as necessary, while the decouplers try to eliminate interaction between loops.

Another example is Dynamic Matrix Control (DMC), which is a common control algorithm and an example of MPC technology. The dynamic control problem in DMC involves minimizing some function of the controller error[9]. Given a set of targets by the LP,

[9] Controller error is the difference between a variable's target and its measurement. A 'function' of controller error can be the square of error, for example.

DMC would compute a set of control steps (also known as a *control plan*) to drive the process to its targets in the most effective way. However, this optimization comes at a cost; the optimized control plan may involve large MV moves that are too aggressive for the system, which can cause stability and safety issues. Hokanson et al (1992) describe the issues with the optimized control plan:

> The preceding least-squares solution [DMC control plan] has two difficulties:
>
> - It tends to be ill-conditioned or very near singular.
> - It results in a very aggressive controller. Without somehow suppressing its outputs, this solution will try to move the plant as fast as mathematically, but not realistically, possible to minimize error. [10]

Consequently, DMC often uses an additional parameter known as *move suppression* to address this issue. Move suppression imposes a limit on the size of the move by which the controller changes each MV. The move suppression value set in the controller therefore determines how aggressive the controller will be, giving rise to an important trade-off in the DMC system. At any given time, the controller is trying to minimize error by driving MVs in their necessary directions. Simultaneously, operators need to maintain process stability and safety by minimizing the magnitude of the MV changes at each iteration.

At each iteration, the LP uses the predictions generated from the process model to solve the economics of the plant in terms of process variables. Sorensen et al (1998) outline the role of the LP in the controller:

> "The use of the predicted steady-state for each dependent variable permits the controller to solve the LP based on the steady-state without the process having to reach the condition. The steady-state solution from the LP is given to the dynamic part of the controller as targets for the execution... In effect, the LP solves for the optimum economic steady-state for the system within the limits specified for the independent and dependent variables (under assumptions of linearity and within the scope of the controller)." [11]

To summarize, the LP uses predictions from the process model to find the most economical operating point, and it provides this economic optimum in the form of steady-state targets for individual variables that the controller drives the process towards. The economic optimum normally lies at the point that maximizes plant revenue and minimizes costs. In order to maximize revenue, the controller often sets one or more CVs to its maximum - for instance, feed rate in an FCC unit. Setting CVs to their maximum values corresponds to process equipment operating at maximum capacity, which can cause concern for the operators, as their work involves maintaining safety and equipment functionality. Consequently, human operation of the plant often exhibits large process variability due to human error and response time in changing process parameters. An automated control system will have a smaller margin of error, meaning that it can drive the process closer to its setpoint compared to that of human operation. This effect is illustrated in figure 1.4.

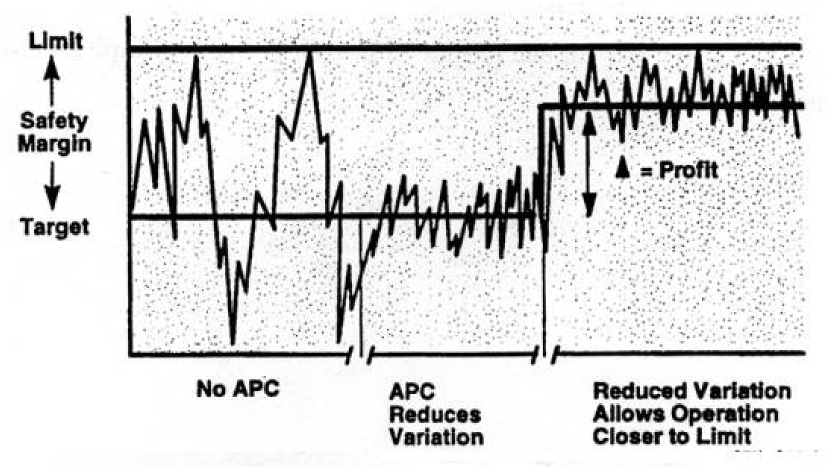


Figure 1.4: Plant performance with human operation and APC [12]. Before APC is implemented, there is significant variability, causing poor performance with respect to the target. Using APC reduces variability, which corresponds to the "MPC operating region" in figure 1.5, allowing the controller to push the process closer towards its control objective.

Reducing process variability (and, consequently, the size of the operating region) allows the process to be driven closer to its desired setpoint. With a wider operating region, human operators are unable to push the process closer towards the desired optimum, because the process variability may cause some parameters to breach their safety limits. This leads to a disconnect between the operating ranges of the true economic optimum, the MPC, and the operators. This disconnect is shown in figure 1.5.

The visualization tool developed in this project and accompanying analysis is focused on the first sub-problem outlined in figure 1.3, partly because the steady-state optimization data provide a wider lens through which we can look at the control of the plant. The decision to limit the visualization tool in this way limits the applications of the tool, which will be discussed further in chapter 4.

In order to develop tools for MPC visualization, it is necessary to first understand the work done by the target users. Forbes et al (2015) outline three key layers of MPC monitoring:

- *Management monitoring* prioritizes financial benefits and resource management, and "briefs management on MPC utilization and economic performance"

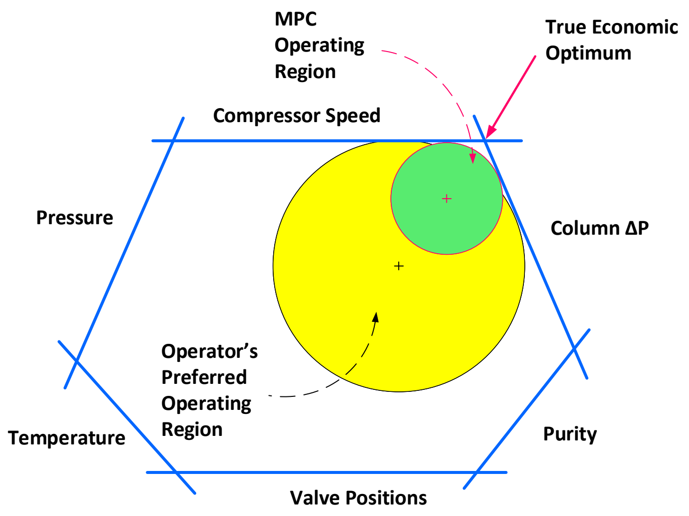


Figure 1.5: Typical preferred operating regions for human operators (yellow) and MPC systems (green) shown with respect to the true economic optimum. Human operators are not always able to run units in tighter control intervals compared to automated systems, often due to safety concerns. The size of each operating region is also indicative of the process variability during operation, so the ideal case consists of operating at the economic optimum point with no variability. This figure is adapted from Brooks (2017) [13].

- *Engineering monitoring* is concerned with the construction and commissioning of new controllers and ensuring their quality. This requires in-depth knowledge of MPC operation.

- *Operational monitoring* is concerned with maintaining and troubleshooting existing controllers, and typically involves the "front-line engineering support staff in order to probe effective utilization operationally" [1]

For the purposes of this project, management monitoring is not the focus. Management monitoring is an important component to consider, but there are other tools used in this stage, and a visualization tool would not be particularly useful for it.

Engineering and operational monitoring are both important layers with regards to controller diagnosis. Both layers involve different responsibilities but are instrumental in determining how well a given MPC will perform, and how a given MPC is diagnosed when it performs poorly. Engineering monitoring often involves subject matter experts who need to commission the MPC so that it is 'fitted' for the plant. [10]

[10] Commissioning often involves mathematical analysis of the plant through system identification and model conditioning, in order to ensure that the model in the MPC is suitable for automatic control (i.e. the model does not contain ill-conditioned subsystems) and reflects the plant accurately.

Operational monitoring involves process engineers, control engineers, and operators, who are concerned with the day-to-day performance of the control system. Tasks related to this layer include monitoring process variables and their statuses (i.e. if they are constrained, out of service, etc.), detecting process faults, and diagnosing the controllers. Since this layer is the one concerned with controller diagnosis, those involved in operational monitoring are chosen as the targeted end-users of the visualization tool developed.

1.3.5 *LP operation*

The optimum operating point given to the MPC, as discussed before, is at some intersection of process constraints. The decision of which variables to constrain is made by the LP at each iteration and often changes throughout MPC operation. In order to understand the controller diagnosis process and constraint analysis in more detail, it is important to first understand how the LP operates and how it chooses which set of constraints should be active at any given point.

The LP solves an economic objective function to obtain the optimum operating point, and this objective function is determined using cost data for all variables in the system. The objective function can be a profit function using such cost data, as shown in equation 1.5.

$$\text{Profit} = \sum_i \text{Product}_i \cdot PValue_i - \sum_j \text{Feed}_j \cdot FCost_j - \sum_k \text{Utility}_k \cdot UCost_k \tag{1.5}$$

Where:

- Profit = Plant profit function ($/day)
- Product_i = product flowrate i (quantity/day)
- $PValue_i$ = product value i ($/quantity)
- Feed_j = feed flowrate j (quantity/day)
- $FCost_j$ = feed cost j ($/quantity)
- Utility_k = utility usage k (quantity/day)
- $UCost_k$ = utility cost k ($/quantity)

The LP maximizes this objective function subject to given constraints, which can change throughout process operation [11]. The constraints in the LP relate to the engineering limits of process variables in the controller. Depending on the number of constraints and the 'shape' of the controller, the solution to this constrained optimization may take multiple forms.

The 'shape' of the controller refers to the number of dependent and independent variables. In a perfectly square controller, there are equal numbers of dependent and independent variables (MVs = CVs), meaning that the control problem has a unique solution. This is rare in reality; the more likely case is that of the *fat plant*, where there are more MVs than CVs, allowing for greater degrees of freedom in the system[11]. More degrees of freedom give the LP the flexibility to pursue multiple operating points, depending on what the objective function dictates. In contrast, the *thin plant* case is one where there are more CVs than MVs, which can occur when control action is lost through valve saturation or manual control. This creates an infeasible control problem. In this case, the controller cannot meet the specified objectives; constraints must be relaxed so that more degrees of freedom are available [14].

[11] The 'degrees of freedom' in a control system refer to the number of unconstrained MVs (which equals the number of constrained CVs). A square controller has zero degrees of freedom, a fat plant has a positive amount, and a thin plant has a negative amount. Degrees of freedom can be thought of as the degree of specification of a 2-dimensional system; the square controller is perfectly specified with a unique solution. The fat plant is underspecified and has a multidimensional solution (a line or plane), while the thin plant is overspecified.

In the 'fat plant' case, the controller has degrees of freedom available so that it can drive the process to the operating point designated by the LP. The constraints chosen in this case are determined solely by the cost function, because the controller has the freedom to move the operating point as necessary. On the other hand, the 'thin plant' case imposes additional restrictions on the controller. If more setpoints are specified than there are degrees of freedom available, the controller must relax some constraints in order to make the control problem feasible. The LP must decide on a set of CVs that create the least concern for operators and relax their constraints, and this decision is influenced by both economics and process safety. Sorensen and Cutler (1998) summarize this decision process as follows:

> "Operators, engineers and managers associated with the process operation are asked to evaluate each dependent variable when it is exceeding its most important limit and assign a delta value beyond the limit that represents a crisis situation. The delta value beyond the limit for the most critical [CV] is used as a reference. Effectively, the personnel with the greatest knowledge of the operations create a list of delta values, in engineering units, that represents *equal concern for limit violations*." [11]

Calculating delta values and equal concern values allows the LP to minimize concerns when facing an infeasible control problem. These calculations are also used as a basis for the 'cost' of each CV; costs are assigned to negative slack variables[12] in the LP, giving the LP a quantitative basis on which to decide the constraints to be relaxed[13] [11].

[12] Slack variables are unknown variables added to inequalities in order to transform them into equalities. Constraint statements are often inequalities, so adding slack variables turns the set of constraint statements into equations.

[13] Costs of each CV are inversely proportional to their 'equal concern' value, and directly proportional to their importance in the system.

Given a feasible control problem, the LP then determines *how* to approach the calculated operating point. In doing so, the LP uses the current state and predictions to calculate a set of MV moves over the specified time horizon. The LP maintains economic objectives over this transition period by ensuring that the MV moves are economically sound. The LP must therefore rank MVs based on their impact on the optimal solution, and this impact is quantified as the 'LP cost'. The

LP cost of a given MV is defined as in equation 1.6.

$$LPCost_i = -\frac{\partial \text{Profit}}{\partial MV_i} \approx -\frac{\Delta \text{Profit}}{\Delta MV_i} \tag{1.6}$$

Using the cost for each MV, the LP is able to determine the most profitable (or least costly) path between operating points. These costs are combined with the 'move suppression' parameters (discussed in section 1.3.4) in order to produce an optimized and cost-effective control plan in order to move the plant between operating points.

Additionally, the objective function can be expressed in terms of minimizing cost rather than maximizing profit. Often some variables cannot fall neatly into categories of 'product', 'feed', or 'utility', so expressing the objective function in terms of the total cost in the plant makes for a clearer objective function. Based on the LP cost and move size of each MV, the objective function can be simplified as follows:

$$\text{Cost} = \sum_i LPCost_i \cdot \Delta MV_i \tag{1.7}$$

Where:

- Cost = the total cost
- $LPCost_i$ = LP cost value of MV_i
- ΔMV_i = change in MV_i

This form of the objective function is often the standard form used in APC systems, because it can account for all variables in the system without ambiguity regarding the category that each variable falls into. In this way, if the LP cost of a given MV is negative then the LP will try to maximize it, and vice versa.

Having quantities that can prioritize both MVs and CVs, the LP is able to determine:

- *Where* the operating point needs to be in terms of which constraints must be satisfied, and
- *How* to drive the plant in that direction, in the form of an MV control plan to relay to the controller.

An additional parameter often considered in linear programming and constrained optimization is the 'shadow price', which is: "the change in the optimum value of the objective function per unit change in a constraint limit (the right-hand side of an inequality constraint)" [15]. With reference to the LP objective function in equation 1.5, the shadow price of a given CV is the change in profit per unit change in the constraint limit (similar to the LP cost calculation for MVs). Shadow prices allow for detailed sensitivity analyses to be performed, and may relate to problematic variables when performing controller diagnosis. The shadow price is also only relevant to variables that are actively constrained; unconstrained variables, by definition, have a shadow price of zero[14] [16]. If a certain CV is falling/rising for unknown reasons and reducing profit, engineers need to find the MV responsible for these changes. Because the shadow price is nonzero only for constrained variables, it may give additional insight that can aid engineers in the controller diagnosis process.

[14] Increasing the constraint limit of a given unconstrained variable will have no effect on profit, because the current profit is dependent on other constraints.

1.3.6 MPC diagnosis

In operating and maintaining MPC systems, there are many issues commonly faced in different plants that engineers need to troubleshoot on a regular basis. Issues that are commonly discussed in control literature include model-plant mismatches (MPMs) and underperforming control loops, where many papers offer various data-driven strategies for MPM detection/elimination and MPC tuning. Often the solution to these issues is some method of re-calculating the MPC model and the optimal MPC tuning parameters, but these processes are time-consuming and can cause major disruption to production. For example, in recalculating the MPC model by performing step tests, engineers need to go through numerous administrative and logistical barriers before being able to actually run the tests. These can include ensuring that instruments are functional, setting up data collection, planning the specific variables to be tested, scheduling the step tests, and coordinating with other departments to confirm that the scheduled window does not disrupt any other work. Moreover, in the testing window, engineers need to run their system identification computations and ensure that the model is realistic by analyzing it using partial pivoting, RGA, or SVD. These factors make model recalculation quite tedious, thus the model is not recalculated frequently.

Rather, typical issues that users face on a more frequent basis can include incorrect limit clamping and instrumentation issues. Through discusssion with our industry partner at the Parkland refinery, it is found that incorrect clamping and instrumentation issues are common issues that MPC users often need to troubleshoot on a day-to-day basis. Troubleshooting MPCs to solve such problems is also time-consuming, but these human factor issues occur much more frequently than MPMs. Consequently, the question of how to automate this troubleshooting process is an important research question for this research.

As described by Guerlain et al (2002), some MPC variables may have their limits tightened (or 'clamped') due to instrumentation issues or as a response to local operating conditions. For large-scale MPC systems, operating conditions may take days or weeks to return to normal, meaning that operators or engineers may forget to unclamp those variables. If the typical limits are not reinstated when operating conditions return to normal, the clamped variables can limit the ability of the controller to control the process [2]. The clamped variables each take up one degree of freedom from the controller, causing it to drive the process towards its desired operating point through undesirable moves, such as changing feed rate or other economic handles. While this may seem like a trivial issue that is easy to solve (by simply identifying the clamped variables), it is often difficult to understand undesirable MV moves without detailed understanding of the process and investigation of process data, especially for large-dimensional controllers.

As for the instrumentation issues, poorly functioning instrumentation can result in far-reaching consequences for the control system. Field instruments can fail for numerous reasons, such as fouling, leaks, or poor maintenance. Instrumentation issues are often detected by performing quality checks on the data in the DCS, to ensure that bad data are not sent to the MPC system from failed instruments. However, some instrument failures may go undetected through these quality checks, because certain instruments are more complex and their failures are not easily detectable based only on the quality of data. For example, the Parkland refinery uses a pressure-compensated temperature (PCT) inferential to estimate C_2 content in a de-ethanizer column bottoms stream[15], and this PCT inferential is biased by gas chromatography (GC) readings. This bias is only present when the GC device is online; when the GC device is offline, the bias must be manually removed by the operator. When the bias is not removed while the GC is offline, the PCT reading is incorrectly biased, but the MPC will not detect this incorrect bias and continue to use the PCT for control. Over a longer period of time, the C_2 prediction may hit an upper or lower limit while still being incorrectly biased, causing the controller to use alternative degrees of freedom to drive the process towards its setpoint. In a large control system, it will not be immediately obvious that the PCT reading is incorrectly biased, and finding this result will be difficult without detailed knowledge of how the instrumentation is tied to the inferential control and DCS systems.

[15] The de-ethanizer column is fed the outlet stream from the top of the main fractionation column and separates lighter hydrocarbons ($C2$ content) from heavier hydrocarbons. The C_2 content of the bottoms stream in the de-ethanizer column is therefore a measure of separation and of the purity of the bottoms stream.

Both of these issues are common in the day-to-day operation of large-scale MPC applications, but are not strongly emphasized in control literature. Controller diagnosis is an essential component of MPC operation and is made more difficult by the prevalence of these issues on a regular basis. We provide a generic process for controller diagnosis that requires no process knowledge; rather, this method of diagnosis comprises an iterative process of elimination involving a visual representation of the gain matrix.

Given a problem CV, we can use the gain matrix and knowledge of active constraints to find the variable causing the problem. Isolating that row[16] in the gain matrix lets us identify which MVs are controlling the problem CV. For each of those MVs, the constraint status indicates whether the given MV is the root cause; if the given MV is constrained, we need to check if unclamping that constraint would solve the problem. If the problem is not found, then we must repeat this process for other MVs in that row that have nonzero gains and that have not been checked before. If no other MVs exist in that row that satisfy these conditions, then the user must identify the column of the last MV being investigated. The same process follows for checking whether the CVs in that column are causing the problem; checking their constraint statuses and repeating for other CVs in the column. This process is iterated, checking every variable until a final root cause is established. By iterating through every set of variables influenced by a given MV or CV, this data structure can be represented by a stack: a type of list where items are stacked on top of each other, and only one element is accessible at a time, which is the element that last entered the stack. The concept of a stack is shown in figure 1.6, and the algorithm is summarized in figure 1.7.

[16] This form of the gain matrix shows CVs as rows and MVs as columns, which does not follow DMCPlus convention, where MVs are rows and CVs are columns. This is a more intuitive form, as outlined in equations 1.1-1.2

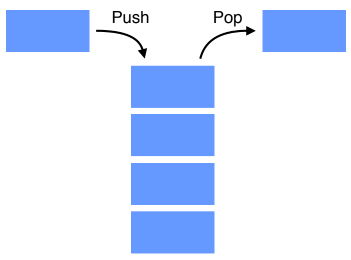


Figure 1.6: Representation of a stack data structure. The stack is filled by 'pushing' elements, and 'popping' removes the top element from the stack.

Consider a 4×3 system (4 CVs, 3 CVs) with three active constraints and, consequently, two degrees of freedom. Figure 1.8 summarizes the features of this system, including a heat map representation of the gain matrix.

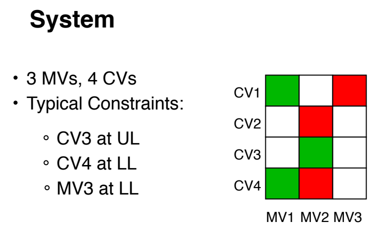


Figure 1.8: Sample system to illustrate the proposed generic controller diagnosis process. The steady-state gain matrix shows the directions of each MV/CV pairing, with negative gains in red and positive gains in green.

For the purposes of this system, we consider CV3 to be the main controlled variable and is constrained at its upper limit, similar to the feed rate in an FCC unit. Consequently, a typical fault in this system could be where CV3 becomes unconstrained and drops. To diagnose this fault, users can use the gain matrix and knowledge of the typical constraints in order to eliminate variables that are not the root cause. For each variable identified in the gain matrix, whether or not that variable is

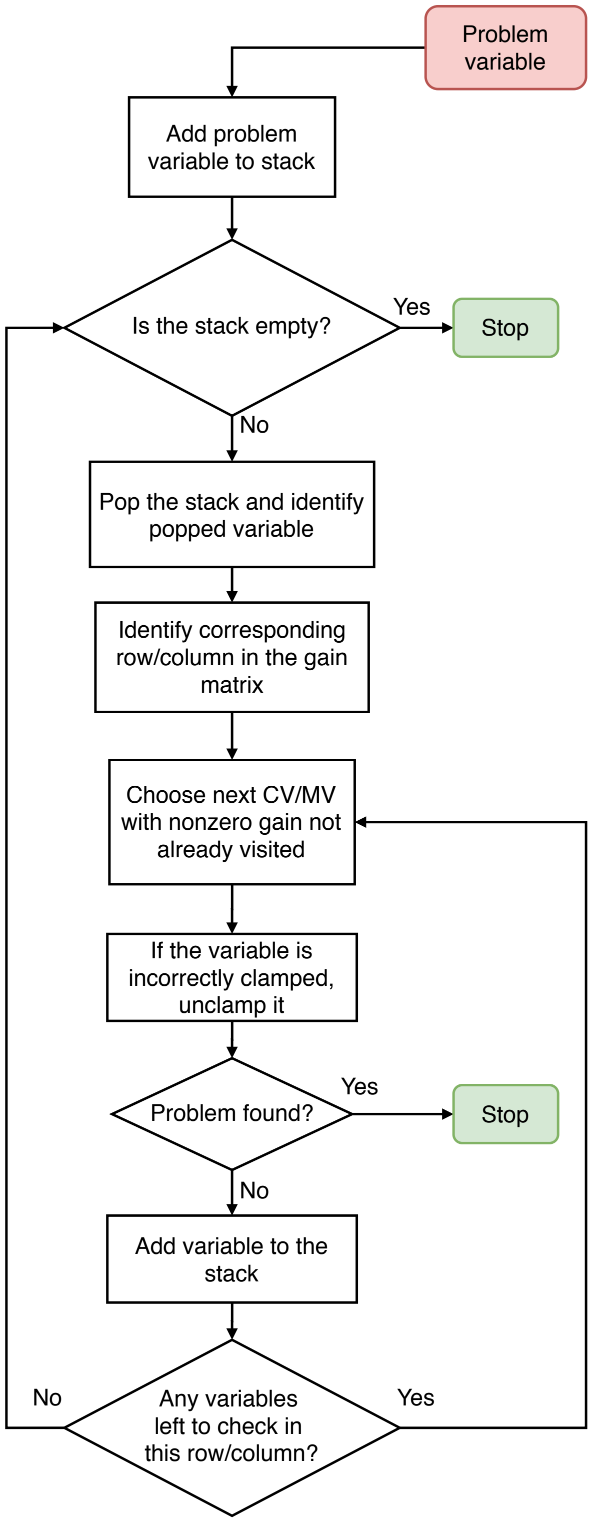

Problem variable
Add problem variable to stack
Is the stack empty?
Yes
Stop
No
Pop the stack and identify popped variable
Identify corresponding row/column in the gain matrix
Choose next CV/MV with nonzero gain not already visited
If the variable is incorrectly clamped, unclamp it
Problem found?
Yes
Stop
No
Add variable to the stack
Any variables left to check in this row/column?
No
Yes

Figure 1.7: Abstract diagnosis process for identifying variables that are incorrectly clamped. The stack represents the list of variables that users need to check; users first need to identify the variables directly related to the problem variable, then investigate the variables related to each variable added to the stack. In the realm of software engineering, the algorithm represents a 'Breadth-First Search' (BFS) algorithm. In contrast, a 'Depth-First Search' (DFS) algorithm would begin with identifying a single variable related to the problem variable, then going to that variable's row/column in the gain matrix, and so on. These can both be classified as 'naive' search algorithms, because no process knowledge is employed to eliminate irrelevant variables from the stack.

the root cause is determined by the presence of nonzero gains in that variable's row or column. This process is summarized in figure 1.9, and the descriptions indicated by letters in this figure are outlined in table 1.3. Each variable has a constraint status adjacent, and figure 1.10 illustrates the meaning of the constraint symbols.

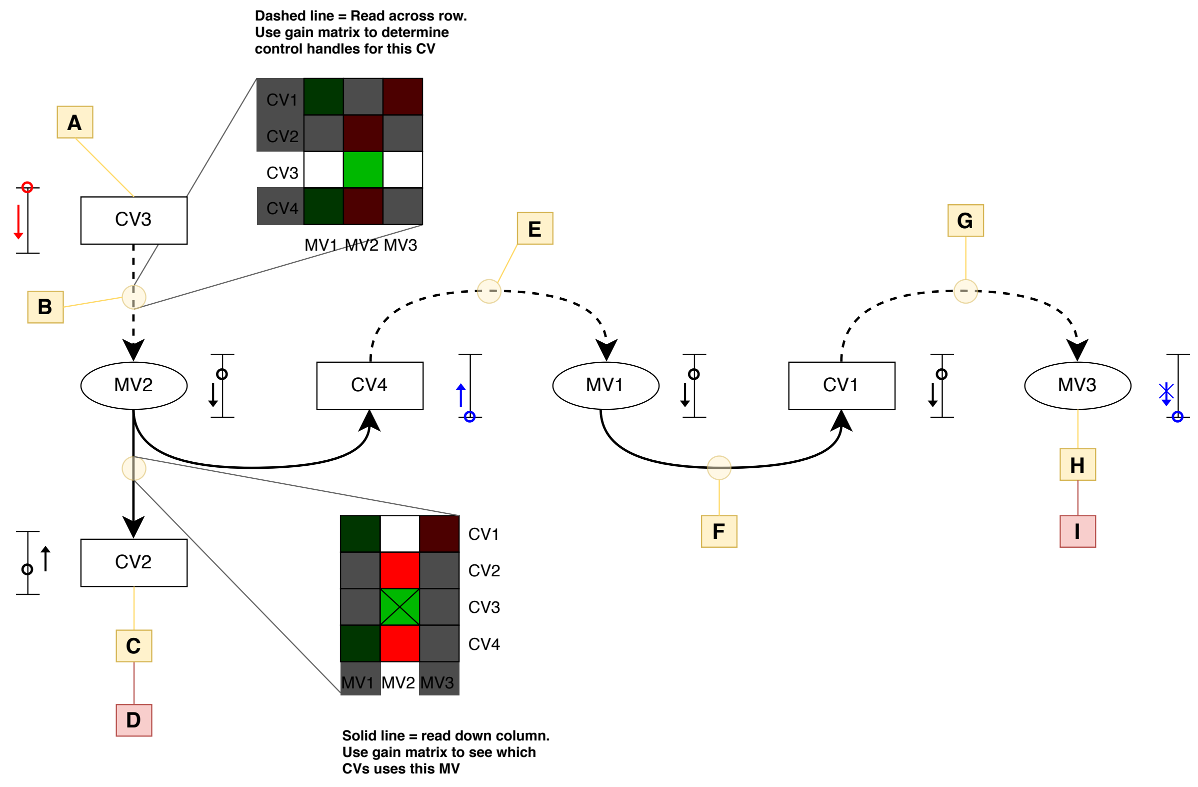

Dashed line = Read across row.
Use gain matrix to determine
control handles for this CV
A
CV3
B
CV1
CV2
CV3
CV4
MV1 MV2 MV3
E
G
MV2
CV4
MV1
CV1
MV3
H
I
F
CV2
C
D
CV1
CV2
CV3
CV4
MV1 MV2 MV3
Solid line = read down column.
Use gain matrix to see which
CVs uses this MV

Figure 1.9: Generic controller diagnosis process without background process knowledge. The gain matrix is used to eliminate variables in each of the transitions, and the symbols adjacent to each variable represent that variable's constraint status, which are explained in figure 1.10. Each letter represents an analytical thought process, and these are summarized in table 1.3.

A legend explaining the constraint symbols in figure 1.9 is shown in figure 1.10.

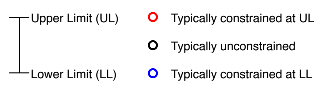


Figure 1.10: Explanation of the constraint symbols in figure 1.9.

The diagnosis process begins with CV3 becoming unconstrained and dropping from its upper limit. Looking at the gain matrix, CV3 is controlled only by MV2 with a positive gain, so the focus of our analysis shifts to MV2. MV2 is not part of the constraint set, so is not the root cause of the problem; the controller is dropping

MV2 because of another variable. MV2 controls CVs 2, 3, and 4, and we have already looked at CV3, so we can eliminate it. Both CV2 and CV4 have negative gains with MV2, meaning that they increase when MV2 is reduced. CV2 is not typically constrained and has no other control handles, so we can eliminate it. CV4 is typically constrained at its lower limit and MV2 is driving it upwards; however, CV4 is also controlled by MV1, with which it has a positive gain. Consequently, while MV2 drives CV2 upwards, the controller will drive MV1 down in order to maintain CV4 at its constraint. We then shift to MV1, which controls CV1 and CV4 (both with positive gains), thereby eliminating both CV4 and MV1 as the root cause. Since MV1 is moving down, CV1 moves down as well. The only other MV that controls CV1 is MV3, with a negative gain. MV3 is typically constrained at its lower limit; in order to maintain CV1 at its level, MV3 needs to move down. Because the limit is clamped, MV3 cannot move down any more, meaning that MV3 is the root cause of the problem. These findings are summarized in table 1.3.

Letter	**Explanation**
A	CV3 is typically constrained at its UL. During a fault, CV3 becomes unconstrained, so it will move down. Why did CV3 move down?
B	MV2 is the only MV that has a control handle over CV3, and this pairing has a positive gain. Since CV3 moves down, MV3 moves down. MV2 is a control handle over CV2 and CV4. Why is the controller moving MV2 down?
C	CV2 is typically unconstrained, so the drop in MV2 causes it to move up.
D	CV2 is unconstrained and has no other control handles, so is **NOT** the root cause.
E	CV4 is typically constrained at its LL. MV2 is driving it upwards. To keep CV4 at its LL, the only handle available is MV1, with a positive gain. Why is the controller moving MV1?
F	MV1 is unconstrained and controls CV1 and CV4. MV1 moves down to maintain CV4, but what happens to CV1?
G	CV1 is unconstrained and has a positive gain with MV1 so it moves down. To maintain CV1, MV3 is the only available control handle.
H	MV3 is constrained at its LL. It needs to move down to maintain CV1, but the limit is clamped, so it can't move.
I	MV3 is the root cause.

Table 1.3: Explanation of the thought process for understanding the cause of the drop in CV3 for the generic controller diagnosis process.

This diagnosis process utilizes the gain matrix to eliminate MVs and CVs based on their constraint statuses and the number of variables that they control/are controlled by. The problem being solved here is that of an incorrectly clamped MV/CV, as explained earlier in this section.

This algorithm for controller diagnosis can also be represented as a graph, with nodes representing MVs/CVs and their required computations, and edges representing nonzero gains between specified MV/CV pairings. Using a graph to represent the system clarifies the distinction between the BFS and DFS algorithms used to traverse the gain matrix. A graph representing the diagnosis process starting from CV3 is shown in figure 1.11.

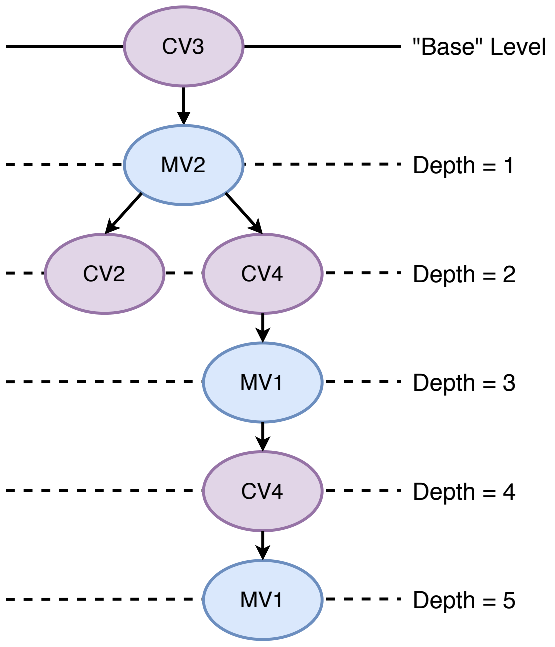


Figure 1.11: Directed Acyclic Graph (DAG) representation of the controller diagnosis process. CVs are highlighted in purple, and MVs are highlighted in blue. Each node represents a given variable, and the computation required at each node is the investigation of that variable's constraint data (i.e. whether that variable is incorrectly clamped). Each edge in the graph represents a nonzero value in the gain matrix for that particular MV/CV pairing. The 'root' node is CV3 (where the diagnosis process begins), and the 'depth' of each node is the number of edges between that node and the root node. Because the gain matrix in this example is fairly sparse, the DFS and BFS algorithms are quite similar. In a larger matrix, each node has more sub-nodes, and so the order of visiting variables is highly impacted by the search algorithm chosen.

Presenting the naive diagnosis process in this way provides some additional insight. The 'computation' required at each node is the investigation of that particular variable's constraint status. The process begins with investigating the constraints of CV3, and the nonzero gain between CV3 and MV2 is represented by that edge, so MV2 becomes the next node. We investigate whether MV2 is improperly clamped, then move on to CV2 and CV4, and so on. If the graph begins with a different problem CV, the size of the graph (total number of nodes; total number of variables being investigated) may change. In the best case scenario, the diagnosis process yields a root cause of the problem after one 'level' of investigating (i.e. one unit of depth). In the worst case scenario, we must investigate all variables in the system to find the root cause of the problem. Hence, the size of the graph (i.e. the number of nodes that must be visited) can be mathematically represented as a range between $[2, N(\text{nonzero gains})]$.

Using a naive search algorithm like BFS (as in figure 1.7) provides a very basic form of controller diagnosis. Variables are investigated systematically based on their position in the gain matrix, rather than how they physically affect the problem variable. In reality, this mental process is more accurately represented by a less naive search algorithm, as engineers will not investigate variables they know to be irrelevant. The actual process of diagnosis would be more accurately represented by a greedy search algorithm; greedy algorithms find the solution in the system by making the most locally optimal decision in each step. [17] Greedy algorithms often use heuristics to aid the solution process, so in this case, engineers can incorporate process knowledge as a heuristic in order to evaluate the various branches to be explored. In this way, greedy algorithms can allow for more informed decisions to be made when traversing through the graph. Therefore, real diagnosis processes can comprise a combination of naive searching and informed elimination of nodes, until a reasonable solution is reached.

[17] An example of such a system is the A* algorithm, which uses weighted edges and predictions to optimize the solution.

In the previous example, it was necessary to investigate all seven variables, corresponding to seven nonzero values in the gain matrix. No process knowledge was used to eliminate variables, and each variable was eliminated by looking at its constraint data. Process knowledge allows engineers to eliminate variables in each layer, because it tells engineers which variables are physically capable of causing the problem. For example, engineers can eliminate CV2 from the diagnosis process if they knew that it physically cannot be causing a problem in CV3. This may seem like a trivial difference, but in real systems such as the FCC gain matrix at Parkland, MVs and CVs may have 30 or so nonzero gains in their corresponding columns and rows, corresponding to 30 or so edges coming out of a given node. Without the use of process knowledge to eliminate irrelevant variables at each step, diagnosing the controller is an extremely tedious process.

1.4 *Literature Review*

1.4.1 *Previous work*

There are five principal sources of inspiration that have investigated similar concepts in the past. The first is the MPC Elucidator, a Honeywell visualization tool, on which this project is based. The Elucidator aimed to solve, for the most part, the same problems that we are aiming to solve in this research. Its design was based on the implementation of ‘Representation Aiding Strategies’, which help to “represent relevant domain, task, and system constraints through visual properties of the display, and thus encourage people to perceive these relationships with little cognitive effort.” [2]. Cognitive task and work analyses were performed in order to provide guidelines for the design. Even though the design involved extensive consultation with end users, discussions with industry partners revealed that the product was not as widely implemented as expected. The authors also outline challenges and limitations to the research, which mostly comprise issues with the representation aiding approach as well as the sheer volume of relationships present in the system. The representation aiding approach may cause the designers to encode data that is not easily decodable, potentially causing some representations to be more easily understood than others. The second limitation relates to the volume of complex relationships that designers are attempting to represent; using one representation to do so may be impractical, and it may be helpful to use analytical tools to simplify the data being encoded.

Although this article was published in 2002, many of the arguments made are still relevant today, as the human-automation interactions between operators and controllers have been neglected relative to the development of the algorithms themselves. A response to a blog post by Jim Cahill [17], Chief Blogger for Emerson Automation Solutions, provides a succinct description of MPC engineering:

"MPC math is simple and elegant; MPC engineering is not." [18].

Secondly, Lindscheid et al (2016) discuss the importance of trust in automation - specifically regarding MPC and nonlinear MPC (NMPC) - and outline some requirements for a visualization tool to facilitate human-automation interactions [19]. It is important to note that this same paper does not mention existing visualization tools for such systems.

The next major application of this research is an ExxonMobil patent outlining a process for MPC analysis in 2011 [20]. This patent explains the process of matrix 'pivoting' (also known as 'partial inversion') in the context of multivariable control[18]. Partial pivoting has mostly been used in applications such as statistics, linear algebra, and programming, but has seen limited usage in APC applications. It is surprisingly difficult to find partial pivoting explained in academic process control literature. In addition to this patent, a literature review revealed two other instances of partial pivoting being used in APC applications.

[18] Partial pivoting has been independently studied in different academic disciplines and has a different name in each one, such as "Principal Pivot Transform", "Exchange", or "Sweeping" [21].

The next major source of inspiration is a presentation on "Durable MPC" by Hoffman et al (2010) [22] in the American Institute of Chemical Engineers (AIChE). Hoffman describes a mixing problem and uses partial pivoting to rearrange the gain matrix such that a desired set of variables is on the 'independent' axis, and another is on the 'dependent' axis[19]. The next example of partial pivoting in APC

[19] In reality, often one of these chosen sets would represent the set of variables that are typically constrained, so that operators can understand how the constraints in the system are affected by MV changes.

is in an article by Sorensen et al (1998) in *Hydrocarbon Processing* [11]. Sorensen uses the partial pivoting method ('partial inversion' in the text) in the context of an FCC unit, where it modifies the structure of the model to make it more suitable for integrating the LP into the control system.

In an APC context, partial pivoting would be applied to steady-state gain matrices in order to highlight the relationships between constrained and unconstrained variables. Gain matrices are 2-dimensional matrices with independent variables (MVs and FFs) on one axis, and dependent variables (CVs) on the other [20]. Partial pivoting transforms the matrix such that some MVs/CVs/FFs are moved from one axis to the other based on their constraint statuses. Moving all constrained variables to one axis and all unconstrained variables to the other axis allows operators to understand how a change in an unconstrained variable will affect the other constraints and, consequently, the economic optimum calculated by the LP. While partial pivoting is not directly applied to MPC visualization in the ExxonMobil patent, it can be applied to provide significant benefits in this context. These benefits are discussed in chapter 4.

[20] The steady-state gain matrix in a multivariable control problem relates the changes in control inputs (MVs) to the steady-state changes in process outputs (CVs).

The fifth and final major previous application of similar research appears in the work of Kozub et al (2002) [23]. This paper discusses issues with monitoring MIMO systems like MPCs, specifically in terms of the relationships between the dynamic and steady-state components of MPC control. The steady-state component comprises the LP, which is concerned with the economic performance of the plant. One major issue discussed is the importance of the LP layer in process monitoring. The LP determines the targets towards which the controller drives the process, and so the relationship between the LP layer and the dynamic control layer cannot be ignored. This paper also provides some visual tools that represent important factors to consider in visualizing MPC systems, which are discussed in section 3.2.

1.4.2 *History of interactive heat maps*

One feature of the visualization tool developed in this research is the representation of the gain matrix as a colour-coded heat map. The decision to incorporate heat maps into this visualization is contextualized by this discussion of the history of heat map representations. This is because heat maps are not a common representation of data in process control.

Heat maps have been used most extensively in the field of bioinformatics, especially when involving gene expression data [24]. The idea of permuting matrices to reveal underlying structure started with

Brinton (1914), representing educational data for US states in the form of a shaded matrix [25]. The next major contribution in this particular application of heat maps was from Jacques Bertin (1967), with his physical reorderable matrix; a replica designed by Perin et al is shown in figure 1.12.

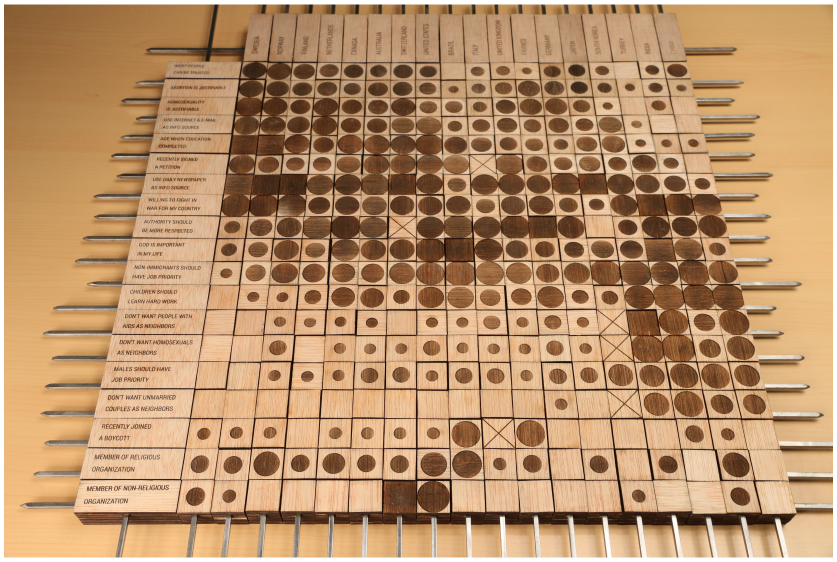

SWEDEN
NORWAY
FINLAND
NETHERLANDS
CANADA
AUSTRALIA
SWITZERLAND
UNITED STATES
BRAZIL
ITALY
UNITED KINGDOM
FRANCE
GERMANY
JAPAN
SOUTH KOREA
TURKEY
INDIA
CHINA
MOST PEOPLE CAN BE TRUSTED
ABORTION IS JUSTIFIABLE
HOMOSEXUALITY IS JUSTIFIABLE
USE INTERNET & E-MAIL AS INFO SOURCE
AGE WHEN EDUCATION COMPLETED
RECENTLY SIGNED A PETITION
USE DAILY NEWSPAPER AS INFO SOURCE
WILLING TO FIGHT IN WAR FOR MY COUNTRY
AUTHORITY SHOULD BE MORE RESPECTED
GOD IS IMPORTANT IN MY LIFE
NON-IMMIGRANTS SHOULD HAVE JOB PRIORITY
CHILDREN SHOULD LEARN HARD WORK
DON'T WANT PEOPLE WITH AIDS AS NEIGHBORS
DON'T WANT HOMOSEXUALS AS NEIGHBORS
MALES SHOULD HAVE JOB PRIORITY
DON'T WANT UNMARRIED COUPLES AS NEIGHBORS
RECENTLY JOINED A BOYCOTT
MEMBER OF RELIGIOUS ORGANIZATION
MEMBER OF NON-RELIGIOUS ORGANIZATION

Figure 1.12: Replica of Bertin's reorderable matrix, created by Perin et al [26]. This matrix was displayed on blocks of wood, each with its data 'value' represented as a circle with a specified area etched onto the surface. Each row and column were held in place using metal rods, and the matrix was reordered by removing and rearranging the metal rods.

The idea was that, by reorganizing the structure of a matrix, Bertin was able to highlight underlying structures in the data. This has been applied extensively in bioinformatics, but has not been applied in the area of process control[21]. Consequently, much of the value of this research lies in the novelty of it; presenting underlying structures in multivariable control applications can yield significant benefits that go beyond simply sorting the gain values for a specific variable.

[21] The method used to actually reorder the matrix is also known as seriation. Seriation began with Petrie (1899) attempting to reorder a matrix of anthropological data to recover temporal ordering of the dataset[24]

In encoding the FCC gain matrix as a heat map with functionality to reorder the matrix, the user exposes numerous opportunities for further analysis. The main analysis performed in this project involves reordering the gain matrix based on gains of MV/CV pairings, and one important benefit of doing so is revealing underlying structure that is unavailable in the raw gain matrix. There are numerous other parameters available by which the user can reorder the matrix to reveal different structures, depending on what information the user is looking for. For example, the gain matrix can be sorted by LP cost to analyze variable gains with respect to their importance in the LP problem[22]. Users can also sort the gain matrix by subcontroller or by name. Alternatively, one could also use the gain matrix to analyze a specific MV; sorting the matrix by gains of that MV tells the user how changing that MV affects the system more effectively than just the

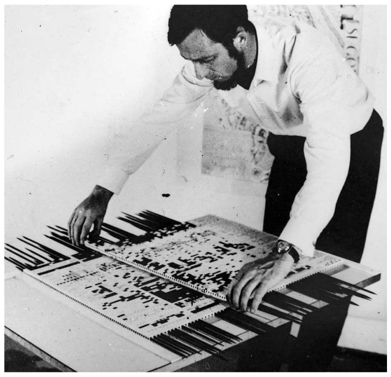


Figure 1.13: Manipulation of the Bertin reorderable matrix [26]

[22] The user's ability to interpret the information shown in the gain matrix is dependent on its arrangement, and this effect becomes more prevalent as the gain matrix grows.

raw gain matrix. Similarly, selecting one CV and sorting the matrix based on that row tells the user how strong each MV's control handle is on that particular CV.

Much of this research is focused on the gain matrix. As shown in section 1.3.2, the information obtained from analyzing one row or column of the gain matrix is fairly straightforward; i.e. how much does MV_1 affect CV_1 and in what direction? Similarly, if CV_1 needs to increase, which MVs can be used to do so? These questions can be answered using the raw gain matrix. It is also known that the raw gain matrix can be analyzed in a number of ways to make its presentation more effective, through techniques like partial pivoting. This technique combines the gain matrix with real-time optimization data, making it more useful for operators. The novelty of this research lies in the following questions: can we apply principles from information visualization and matrix seriation in order to extract additional information from the gain matrix? Since partial pivoting incorporates optimization data into the gain matrix to make it more useful, how can we use the available constraint data with the gain matrix to provide additional benefits? Finally, with the information provided by such analyses, how can we effectively display such information? The benefits of performing these analyses are known to control engineers; Hoffman et al (2010) use partial pivoting to ensure that the gain matrix represents a model that is properly conditioned, so that the controller always uses degrees of freedom that are available in the real system [22]. Peterson et al (2011) use partial pivoting to quantify how certain MV moves will affect the operating point in terms of the active constraint set [20]. Hence, further investigation in this area and incorporating information visualization principles can potentially provide significant benefits to users of MPC systems.

2 Data and Information Visualization

Brehmer et al. (2013) provide a concise definition of information visualization:

> "The foundation of information visualization is the characterization of how known facts about human perception should guide visual encoding of abstract datasets" [27]

The terms 'information visualization' (or infovis) and 'data visualization' are often used interchangeably, but they mean different things in most contexts. In a general sense, data is simply a collection of facts and statistics, without the inclusion of context or meaning. Information is produced from data by adding context and meaning to it. As such, to avoid ambiguity, the definitions used in this discussion are:

- *Data visualization* is the representation of data without context;
- *Information visualization* is the representation of data in a specified context for a specified purpose.

The overall purpose of this field is to communicate information through visual media, allowing users to gain insight into data without analytical tools. [1]

[1] This is accomplished by replacing cognitive tasks that the user would perform with visual tasks facilitated by the user interface. These cognitive tasks are 'translated' into visual tasks through the use of visual encoding, where meaningful information is represented by visual media.

Each medium used to represent information has associated strengths and weaknesses, meaning that each medium is more suited for certain types of data. The effectiveness of a visual medium in representing a dataset is dependent on multiple factors, such as the type of the dataset and its dimensionality. These different types of datasets and the effectiveness of various encoding techniques was first explored by French cartographer Jacques Bertin. Bertin laid the foundation of information visualization in *Semiology of Graphics*, in which he identified three data types (quantitative, ordinal, or nominal) and numerous 'retinal' variables [28]. Quantitative data are any data that have exact numbers; ordinal data are not numerical, but have an ordering to them (i.e. low, medium, high); nominal data are everything else (normally categorical data). The 'retinal' variables are methods for visual encoding that utilize different visual channels to convey information.

Common variables defined in visualization literature are: position, length, area, shape, texture, orientation, and colour. Position is a fairly self-explanatory variable, as this is how information is represented in any scatter or line plot. Distinguishing points based on position is shown in figure 2.1.

Length, area, and shape are all similarly obvious, though they would represent different types of data. Lengths and areas are suitable for quantitative data, as they are typically used to represent data in the form of bar and pie charts, respectively. Examples are shown in figure 2.2.

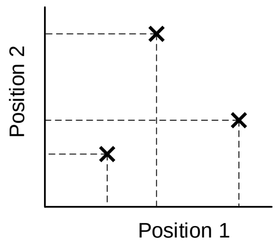
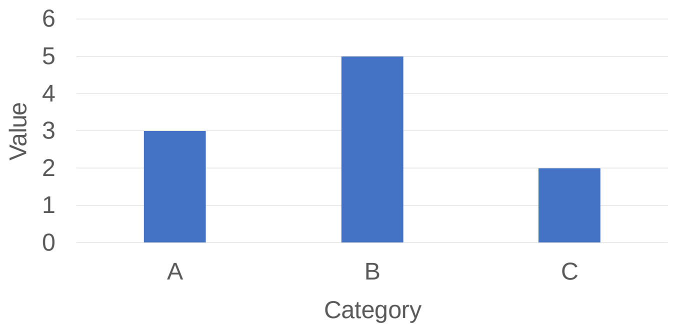
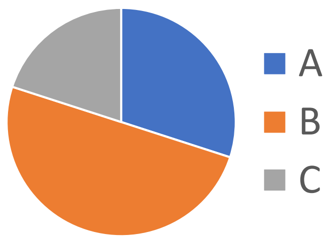


Figure 2.1: Encoding numerical data in the form of each point's visual position. This is the premise behind typical graphs with x- and y-axes (positions 1 and 2, respectively).



(a) A bar chart represents quantitative data through each bar's length.



(b) A pie chart represents quantitative data through relative areas of each section.

Figure 2.2: Comparison of data representation through length (bar chart) and area (pie chart). Both graphs are representing the same dataset.

Different shapes are not as suitable for representing quantitative data, because the correlations between the shapes are not inherently quantitative. The effectiveness of shape as a visual medium is entirely dependent on the specific selection of shapes; it is difficult to select shapes that are inherently quantitative or ordinal, so shape is often used to differentiate categories. An example of different shapes that can be used in visualization is shown in figure 2.3.

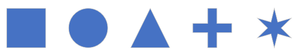


Figure 2.3: Different shapes can be used to encode data, often for nominal datasets.

Texture in this context refers to the patterns appearing on a data representation, such as diagonal lines or a cross-line pattern, but it is not commonly used in visualizations. Texture can also represent the feel of a physical representation, such as the Bertin reorderable matrix. An example of different textures is shown in figure 2.4.

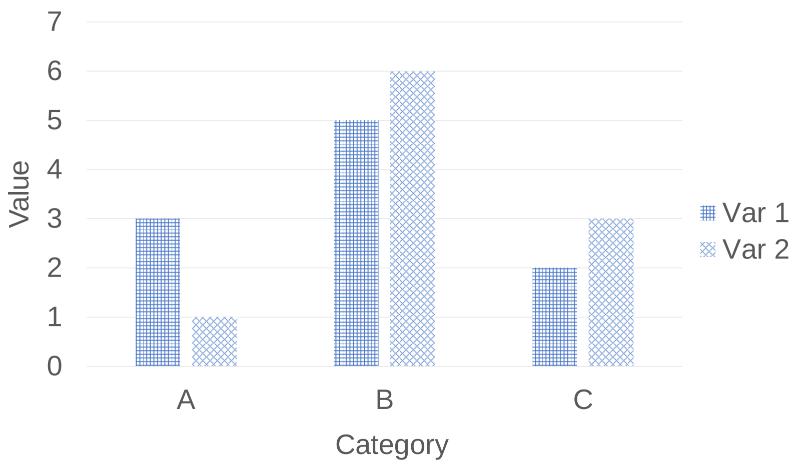


Figure 2.4: Bar chart differentiating variables (Var 1 and Var 2) through the pattern on the bar. This can be a common method of representing nominal data when colour is not available (e.g. on black/white documents).

Orientation is self-explanatory, but its implementation is not as straightforward. For instance, it is visually easy to distinguish a vertical line from a horizontal line, but it is not as easy to differentiate a line at 30°from a line at 40°. An example is shown in figure 2.5.

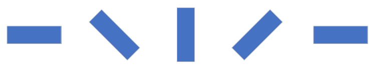
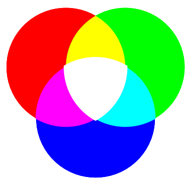


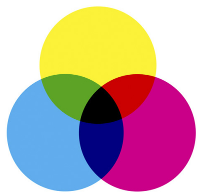
Figure 2.5: Various orientations of the same object. Theoretically, orientation can be used to represent any dataset.

Colour is an interesting variable, mostly because there are multiple ways to quantify colour. Visually encoding a dataset as a colour scale can be accomplished in multiple ways, each being effective in particular contexts. Examples common to digital visualization include RGB (Red, Green, Blue), CMYK (Cyan, Magenta, Yellow, Black), and HSL (Hue, Saturation, Luminosity). RGB and CMYK combine the numerical values for each colour to create the final output[2], while HSL uses a combination of other factors. The structure of RGB and CMYK colouring is shown in figure 2.6.

[2] An easy way to look at these methods of colour mixing is that they are linear combinations of the number assigned to each colour; when red, green, and blue values are identical in an RGB setting, the output is a shade of white. Similarly, when CMYK values are identical, the output is a shade of black.



(a) RGB colour structure. The combination of maximum values for each colour results in the white centre.



(b) CMYK colour structure. The combination of maximum values for each colour results in the black centre.

Figure 2.6: Common colour models used in computer graphic operations. RGB is often a part of digital pixels, making it a common colour model in computer graphics. CMYK is often used in colour printing systems.

Figure 2.6 shows how the linear combination of individual colour values results in a final output colour[3]. These details are important to understand because they dictate how colour is used to encode data.

Mackinlay (1986) summarized the effectiveness of various perceptual tasks in encoding the three types of data. In this summary, position is shown to be the most effective in all types of data, while other visual encodings vary considerably between the data types. Figure 2.7 shows these results.

[3] HSL operates differently; hue represents the identity of the colour, saturation is a percentage representing the amount of 'colour' (0% is a gray colour, 100% is the colour itself), and luminosity is a percentage representing brightness (0% luminosity is black, 100% luminosity is white)

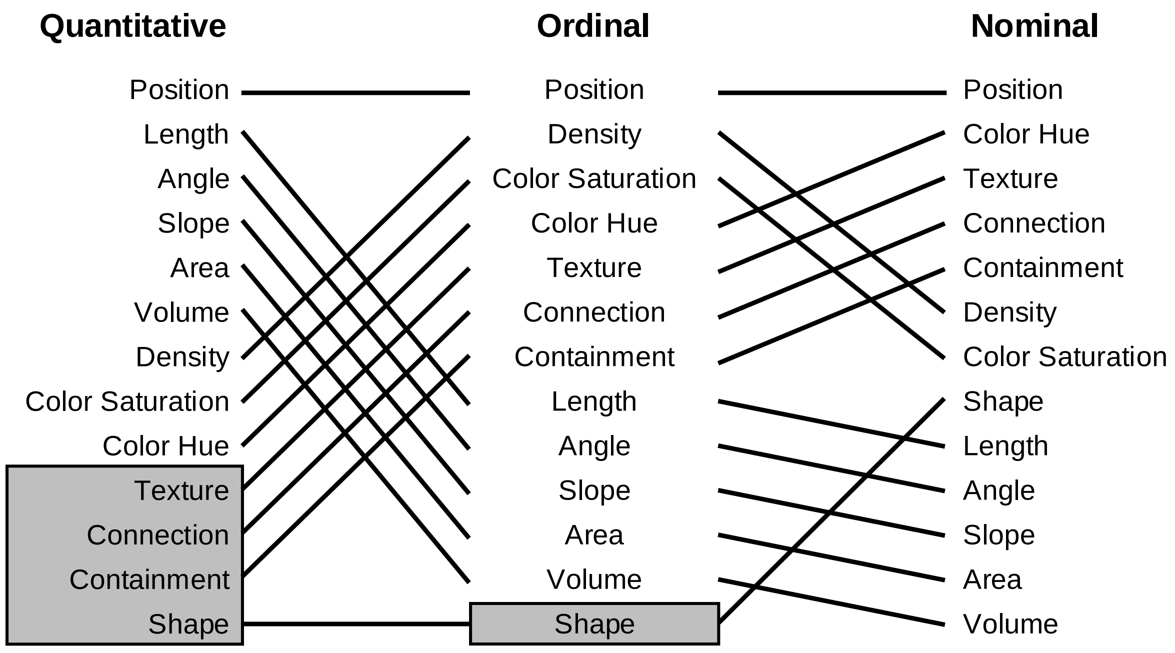


Figure 2.7: Mackinlay's ranking of perceptual tasks. The tasks in the gray boxes are not relevant to the types of data. Figure adapted from Mackinlay (1986) [29]

Figure 2.7 illustrates an important pattern in visual encodings; encodings that have inherently quantitative properties are much more suitable for quantitative data. Similarly, those that are not inherently quantitative are more suitable for ordinal or nominal data. This may seem obvious, but it is important to clarify, because it provides basic guidelines as to how various data should be encoded. The pattern presents itself in figure 2.7 in the changes of each variable's effectiveness between the datasets[4].

[4] More quantitative encodings like length, area, and angle (orientation) become much less effective in non-quantitative datasets. Similarly, less quantitative encodings (colour hue, texture, shape) become much more effective in non-quantitative datasets.

As explained here, there are numerous methods available to encode data, each with their own strengths. However, most existing tools for MPC visualization do not take this into account. The problems outlined by Guerlain et al (2002) are still prevalent in many APC systems; for instance, many systems show the gain matrix in static displays of raw data, spread out over numerous tabs. This is not an effective way to display data in terms of the user's ability to understand underlying patterns. This is partially because of the nature of human perception when interacting with data; a numerical display of data is not a visual medium with which to convey information, rather it is a cognitive one[5]. As such, the perception of underlying patterns in a table of numerical values would be difficult, in comparison to a table of more visual representations of said numbers, such as colours or shapes. The goal of this section is therefore to move away from the static, tabular views that are common to MPC, and move towards a visual tool that is more interactive, effective, and user-friendly.

[5] Note that, in figure 2.7, numerical presentation of data is not shown at all.

Aside from the actual encodings used to present relevant MPC data, interactivity is especially important in creating new visual tools, because interactivity allows the user to reveal underlying structures that were previously imperceptible. The use of this concept in the context of MPC (specifically regarding the gain matrix) was discussed in section 1.4.2, where reordering the gain matrix allows the user to visually sort the control system using multiple criteria, each revealing various hierarchies by which the MPC system operates. For example, sorting the matrix by LP cost allows users to understand which variables are prioritized in the LP problem, whereas sorting by gain value informs the user on how different variables are related to each other. This is an important take-away point in this research that was also emphasized by Guerlain et al (2002) in their "MPC Elucidator" product [2]; by creating visually effective and interactive designs, users are more able to obtain important information from the controller visually rather than cognitively. Jansen (2014) discusses this idea with regards to using a matrix display that provides reordering functionality:

> "...the analyst can interact with [the display] and actively explore the data by manipulating rows and columns to expose patterns in the data. Its design thereby directly supports visual thinking through action [Kirsh and Maglio, 1994]."[30]

2.1 *Nested model for visualization design and validation*

The overall objective of the visualization tool is to extract useful information from numerical plant data and present it in such a way that facilitates controller diagnosis. In order to develop this visualization, we must adhere to some methodology that is suitable to our goals. In this way, we can effectively evaluate the design in terms of how it helps to solve specific problems that MPC operators experience. As mentioned previously, the methodology chosen is Munzner's "Nested Model for Visualization Design and Validation", which breaks the problem of visualization development into four components, as shown in Figure 2.8.

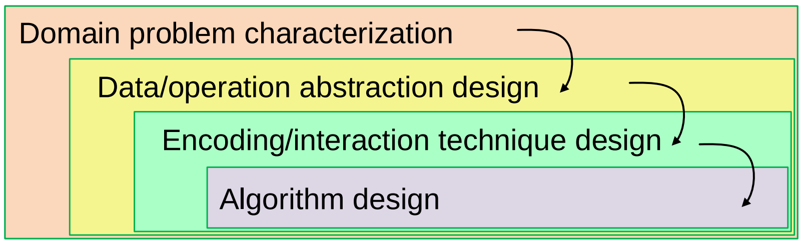

Domain problem characterization
Data/operation abstraction design
Encoding/interaction technique design
Algorithm design

Figure 2.8: Munzner's nested model for visualization design and validation [4].

2.1.1 *Domain problem characterization*

The first level of visualization development is characterizing the domain problem. This involves specifying who the users are and what their intentions are for the visualization, as well as identifying what realm the data to be visualized is relevant to. An example could be data scientists in the commerce sector. It is important to note that domain problem characterization is expressed in terms of domain-specific vocabulary.

2.1.2 *Data and task abstraction*

The second level is concerned with the abstraction of two components: data and tasks.

Data abstraction expresses domain-specific data as generic data types that can be used to build visualizations. For instance, a matrix of steady-state gains can be transformed into a table of floating-point values that are not specific to process control.

Similarly, task abstraction applies this concept to operations that are performed on the data[6]. For instance, an operator's task to monitor controller health using Statistical Process Control (SPC) can

[6] Tasks, in this context, are generic and not domain-specific.

be translated into a set of mathematical rules that apply to time-series data.

2.1.3 Encoding and interaction design

This level is where a design process can be employed in order to create a well-functioning visualization. The output of this layer in the methodology is an interface for the users that displays what has been abstracted in the previous layer, in order to achieve goals defined in the first layer.

2.1.4 Algorithm design

Algorithm design involves writing effective algorithms that create the encoding and interaction techniques developed in the previous layer. Concerns in this level would include the speed and performance of the algorithms, as well as any “bugs" that arise from user interaction.

2.1.5 Threats to validity

Each layer in the nested model has associated *threats to validity* and, consequently, associated methods for evaluating visualizations. At the domain characterization level, the principal threat to validity is mischaracterization of the problem, in that the target users do not actually experience the problems being described. For instance, users of a small MPC system may not experience the issues discussed in section 1.1, meaning that they do not need a new visualization tool to improve MPC workflow. This threat can be addressed using qualitative measures like discussion with the target audience, ethnographic field studies, or investigating the tool’s adoption rate after implementation.

At the level of abstracting data and operations, the main threat to validity is that the target users’ problems are not being solved by the designated data types and operations. A relevant example of this threat is that the task analysis abstracted from a controller diagnosis workflow is not accurate, and is not reflective of how users actually diagnose controllers. Designers can address this threat by testing how target users do their work using the created tool, often using formal field studies to report the users’ experiences.

The level of encoding and interaction design is threatened by the inability of the chosen designs in “communicating the desired abstraction to the person using the system” [4]. In terms of a visualization for MPC systems, an example of this threat is that a colour-coded heat map does not help users understand the MPC structure. Approaches taken to deal with this threat include heuristic evaluations, expert reviews, or formal user studies at a later stage.

Finally, the level of algorithm design is primarily threatened by the poor performance of the chosen algorithm in terms of time and memory. Designers can measure and analyze the performance of the algorithm in order to tackle this threat. Another threat that is more relevant in this project is that the algorithm chosen is incorrect, and does not meet the needs required for encoding the data and operations chosen in the previous layers. In other words, the software packages chosen to create and display the visualization tools may not contain the functionality required for certain tasks; in this case, one example is that the Dash and Plotly packages in Python provide only high-level modules for visualization, making it difficult to customize the visualizations in the application. The limitations resulting from this particular software package are discussed in chapter 4. To manage this, designers can present the created tool to the target audience, who can directly evaluate algorithm 'correctness' [4].

A summary of the threats and validation methods in all layers and how they relate to components of this project is displayed in figure 2.9.

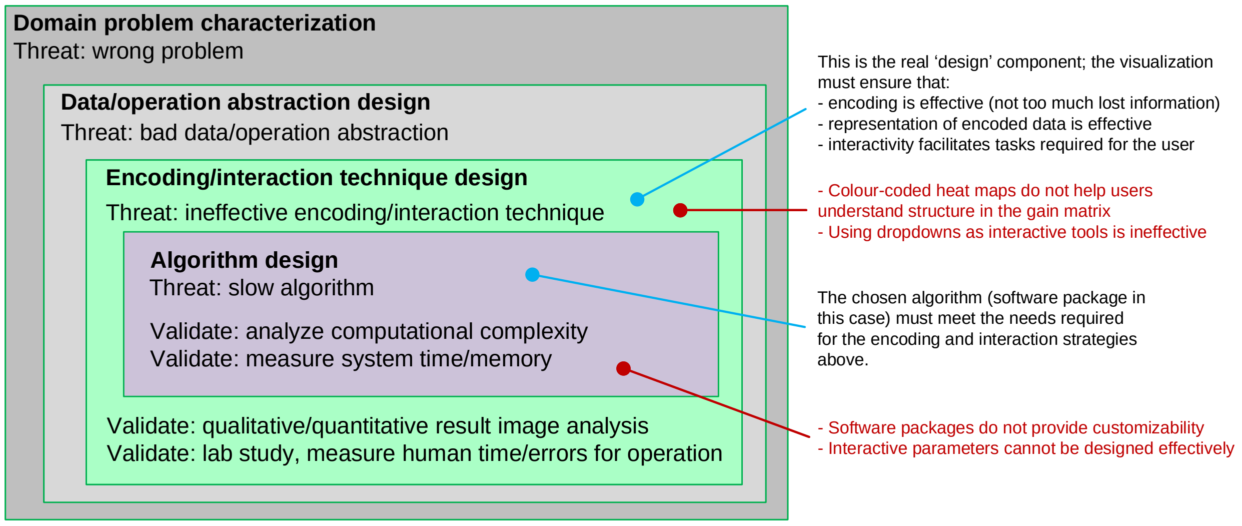

Domain problem characterization
Threat: wrong problem
Data/operation abstraction design
Threat: bad data/operation abstraction
Encoding/interaction technique design
Threat: ineffective encoding/interaction technique
Algorithm design
Threat: slow algorithm
Validate: analyze computational complexity
Validate: measure system time/memory
Validate: qualitative/quantitative result image analysis
Validate: lab study, measure human time/errors for operation
This is the real 'design' component; the visualization must ensure that:
- encoding is effective (not too much lost information)
- representation of encoded data is effective
- interactivity facilitates tasks required for the user
- Colour-coded heat maps do not help users understand structure in the gain matrix
- Using dropdowns as interactive tools is ineffective
The chosen algorithm (software package in this case) must meet the needs required for the encoding and interaction strategies above.
- Software packages do not provide customizability
- Interactive parameters cannot be designed effectively

Figure 2.9: Nested model showing each layer's threats to validity as well as validation methods. The 'Domain problem characterization' and 'Encoding/interaction technique design' layers are greyed out because they are not as relevant as the other layers; creating all the tools in this project involved extensive consultation with potential users.

Figure 2.9 summarizes the threats to validity and validation methods associated with each layer in the nested model. In this research, threats related to encoding/interaction and algorithm design are more relevant than those regarding domain characterization and data abstraction. Consequently, they are the only threats highlighted in figure 2.9.

2.2 *Multi-level typology for task analysis*

In developing visualizations, it is important for designers to take into account the overall goals of the tools being developed. Decisions made for lower-level tasks such as algorithm design and interaction parameters should be driven by high-level decisions pertaining to the 'big-picture' goals of the tool. This seems like an obvious point to consider, but is often a pitfall of such designs. As stated by Munzner (2013) [27]:

> "The considerable previous work characterizing visualization usage has focused on low-level tasks or interactions and high-level tasks, leaving a gap between them that is not addressed. This gap leads to a lack of distinction between the ends and means of a task, limiting the potential for rigorous analysis"

To avoid this pitfall, a task analysis following Munzner's "Multi-Level typology of abstract visualization tasks" is performed and discussed[7]. Performing this task analysis on the typical workflow for MPC diagnosis allows the designer to create visual encoding and interaction strategies that are better than existing strategies and, more importantly, that are justifiable. Otherwise, designers may end up choosing random visualization designs and having to justify them later. [8].

[7] This task analysis pertains specifically to the *Data and Task Abstraction* component of the nested model. The goal here is to translate domain-specific tasks into sets of generic tasks that can be used to more effectively design visualization tools.

[8] Randomly choosing visualizations and justifying them after the fact is what Munzner (2008) refers to as 'Application Bingo' [3]

The main idea presented in this model, as opposed to other task analyses, is its inherent linking of low-level tasks (*how* a given task is performed) to high-level classification of said task (*why* a given task is necessary). The exact methodology revolves around defining the *what*, *why*, and *how* of a given task. Figure 2.10 provides an overview of the methodology:

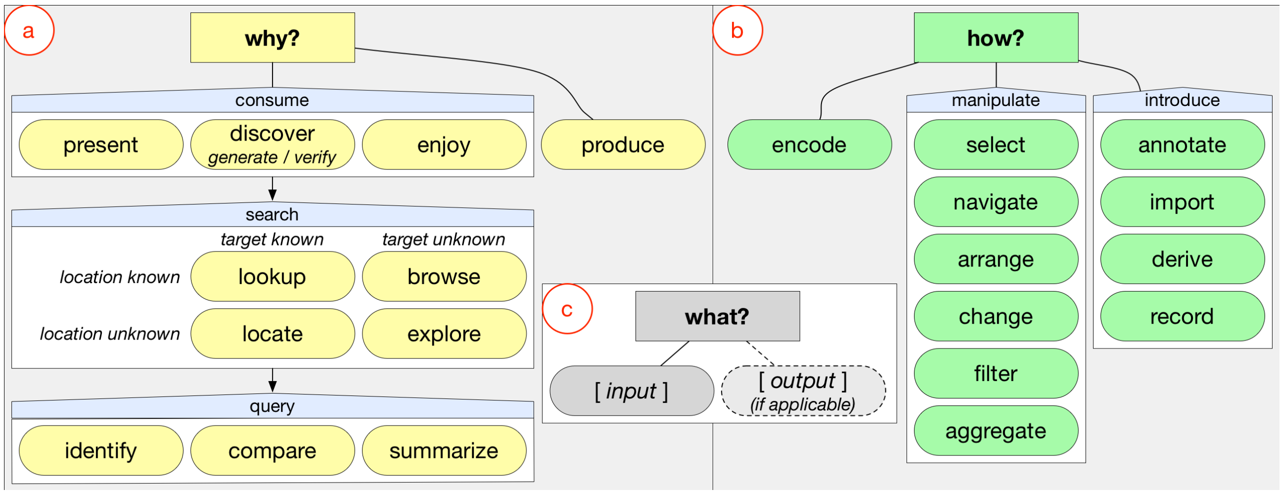


Figure 2.10: Munzner's multi-level typology of abstract visualization tasks.

In this context, the terms presented in figure 2.10 have specified definitions that may conflict with definitions presented in similar work. These definitions are provided in table 2.1.

Term	**Definition**
Present	Display of data within the context of planning, decision making, forecasting, etc.
Discover	Generating and verifying hypotheses through scientific investigation
Enjoy	Casual visualization driven by curiosity
Produce	Generating new information based on input data
Lookup	Searching for a known element in a known location
Browse	Searching for an unknown element in a known location
Locate	Searching for a known element in an unknown location
Explore	Searching for unknown elements in unknown locations
Identify	Finding characteristics of known targets or references relating to unknown targets
Compare	Comparing identified elements to each other
Summarize	Compiling findings from comparisons
Select	Pinpointing a specified element and distinguishing it from others
Navigate	Changing the viewpoint (panning, zooming, etc.)
Arrange	Spatially organizing visual elements
Change	Altering the encoding of a specified element (change colour, transparency, etc.)
Filter	Using specified criteria to highlight elements satisfying those criteria
Aggregate	Alter granularity of visual elements
Annotate	Marking visual elements with graphical or textual additions
Import	introduce new elements to the visualization
Derive	Using existing elements to calculate new ones
Record	Capture current state of visual elements

Table 2.1: Terms presented in figure 2.10 and their associated definitions in the context of the multi-level typology. Higher-level terms are omitted because they are fairly self-explanatory [27].

3
Results

3.1 Task analysis

3.1.1 Case study: feed rate

Before designing the visualization, we were given a 'case study' that emulated the typical workflow for MPC diagnosis. The goal was to pinpoint the root cause of a drop in the product rate (or feed rate) of the plant. This diagnosis procedure resembled the IDS procedure discussed in section 1.3.6. The product rate data is shown in figure 3.1.

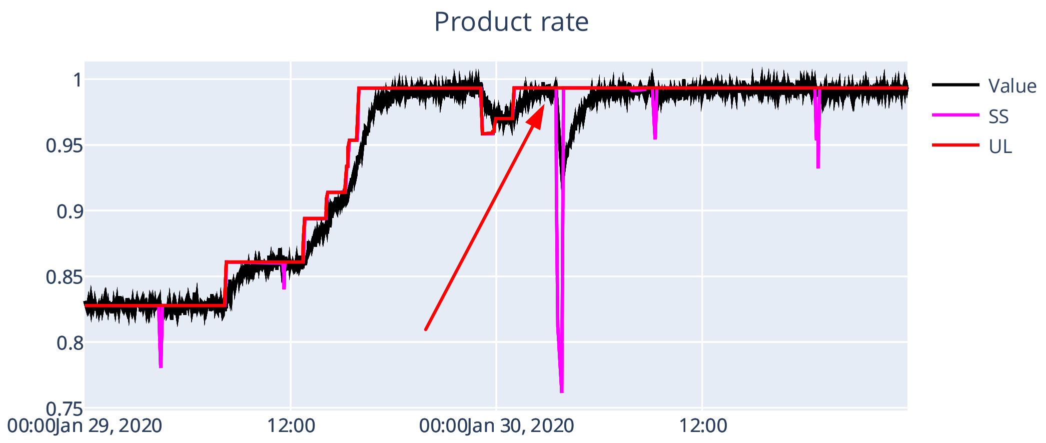
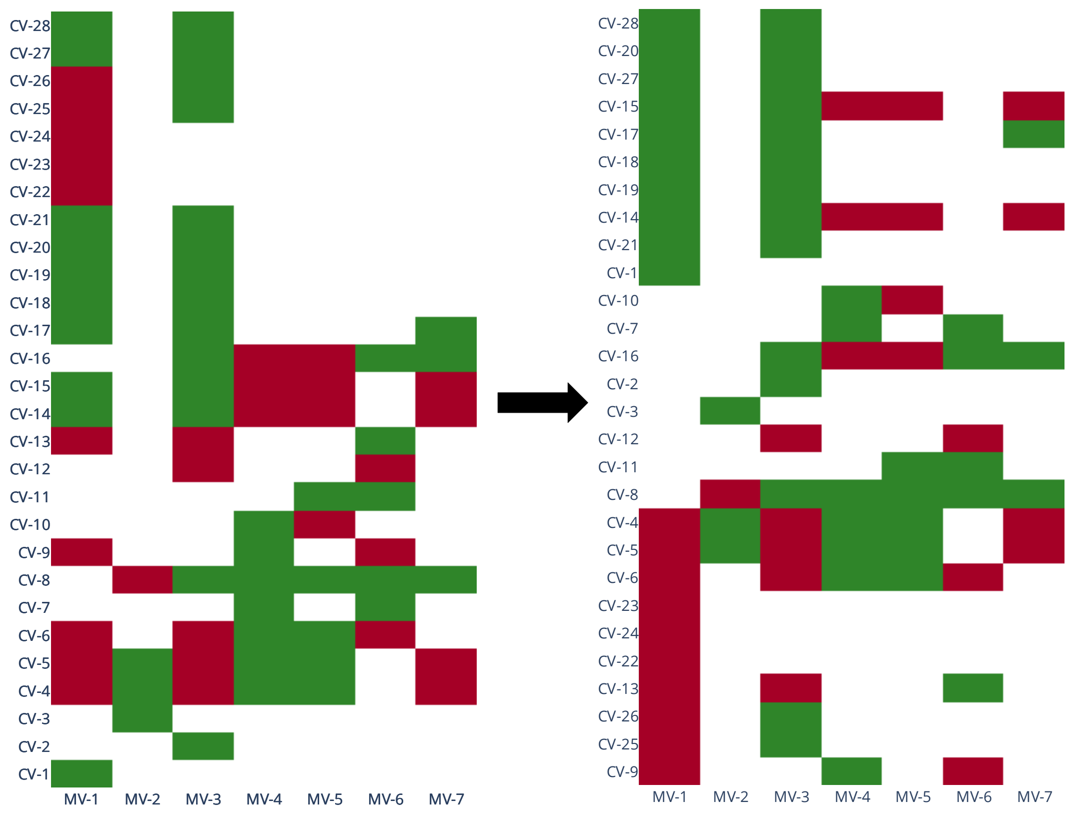


Figure 3.1: Product rate over time from January 29th, 2020 to January 31st. The aforementioned drop is apparent in the 'Value' and 'SS' tags around 3:00AM on January 30th; the red arrow is pointing to the drop. The lower constraint is omitted to illustrate the magnitude of the drop.

The first step in diagnosing this controller is to identify variables that are directly related to product rate. The easiest way to do this with the provided data is through the gain matrix; by isolating the row/column of the product rate and sorting the resulting variables by value, the user is able to quickly filter out variables that are not directly impacting product rate. The heat map representing the gain matrix, and the resulting heat map sorted by product rate (MV-1), are shown in Figure 3.2.



.

Figure 3.2: Heat maps representing the same section of the gain matrix as shown in table 1.1, with negative values in red and positive values in green. This section is shown instead of the full gain matrix to highlight the reordering functionality. The justifications for using heat maps to represent the gain matrix are discussed in section 3.2.1

By sorting the gain matrix visually, it is easy to filter out the variables that have nonzero gains with respect to product rate. Identifying this set of variables is made considerably easier with the functionality to reorder the matrix; this feature is discussed further in section 3.2.1. The heat maps shown in figure 3.2 show the final iteration of the gain matrix representation; before representing the colours as solid, the colours were on a continuous scale. This issue is discussed in detail in section 3.2.1.

After identifying the variables that directly impact product rate, the next step is to examine their behaviour in order to find the variable[1] that is exhibiting abnormal behaviour around the time the problem started. This involved presenting the time-series process data for each variable and visually analyzing its behaviour around the time the product rate fell. There are 36 variables that directly impact product rate, so this step of the process is fairly time-consuming. Six of the potential 'culprits' are displayed in figure 3.3.

[1] This step involved choosing a *single variable* and not a set of abnormally-behaving variables because it was necessary to iterate through multiple times. Choosing a single variable allows iteration without needing to investigate an excessive amount of data.

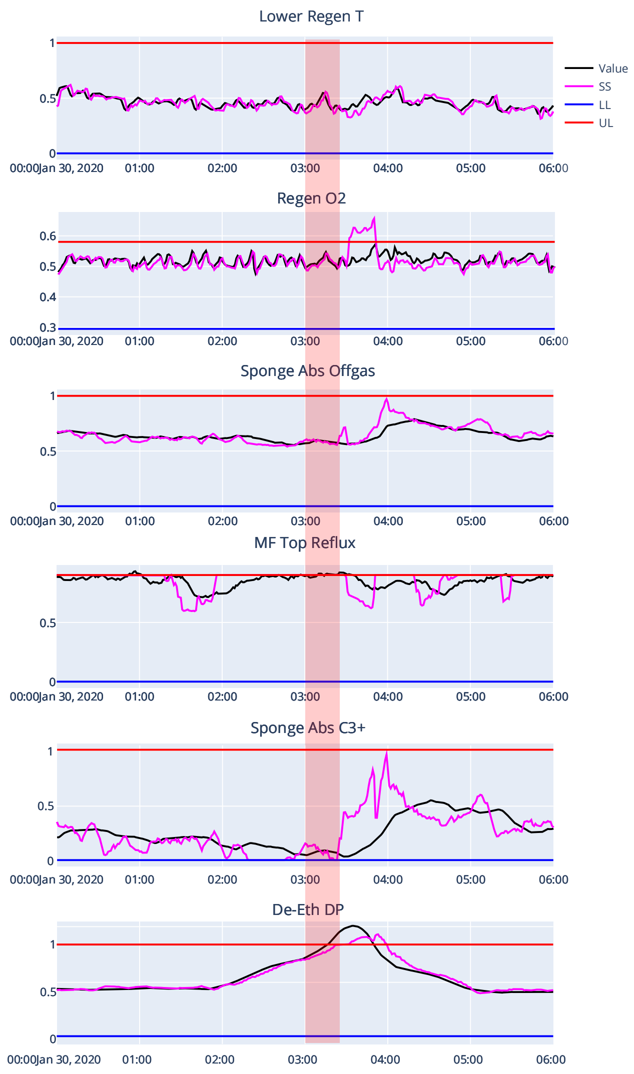


Figure 3.3: Analyzing the CVs with nonzero gains with respect to product rate. The x-axis for each figure represents a narrow time period within which the drop in product rate occurred, between 00:00 and 06:00. The red region is the time period around which feed rate drops, and around which each variable is investigated.

Out of the plots shown in figure 3.3, De-Eth DP, Sponge Abs C3+, and MF Top Reflux (the bottom three plots) were initially thought to be the potential culprits. In order to determine the exact variable causing the drop in product rate, it was necessary to understand what each variable represented in the plant. From there, a process of elimination began; for each considered variable, it was necessary to evaluate the validity of the hypothesis that the behaviour of that variable can cause the change in product rate. It was found that De-Eth DP (the sixth plot in figure 3.3) was a likely cause of the drop in product rate, through consultation with industry partners. De-Eth DP represents differential pressure in the de-ethanizer column in the refinery. In this step, it was necessary to incorporate process knowledge, as this allowed us to eliminate hypotheses that are unrealistic from a physical standpoint.

De-Eth DP is a CV in this MPC system, and so the unusual spike shown in figure 3.3 (in the sixth plot) was caused by some abnormality in MV behaviour, which was driven by the controller. Consequently, it is necessary to dig further and perform the same analysis to understand what was causing the controller to drive De-Eth DP upwards so intensely. This task is iterated until a reasonable root cause of the problem can be identified. This is the last distinct task in the diagnosis workflow; subsequent tasks involve repeating these steps over different sets of variables. We repeated this procedure - sorting the gain matrix and filtering out unnecessary variables - for De-Eth DP, which led us to believe the issue was caused by De-Eth Reb Stm (reboiler steam flow rate for the same column). The process plots for product rate, De-Eth DP, and De-Eth Reb Stm are shown in figure 3.4.

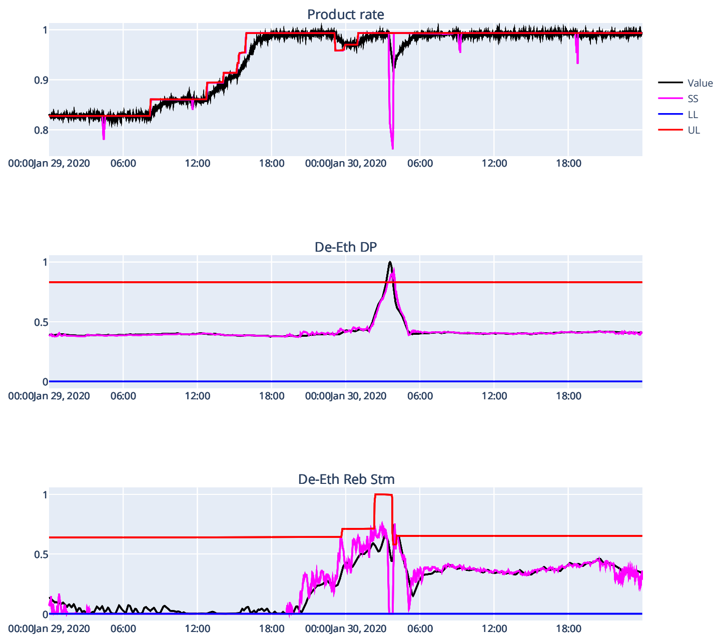


Figure 3.4: Two 'levels' of tracking the cause of the drop in product rate. This corresponds to a depth of 2 in a graphical representation of the diagnosis process, as discussed in section 1.3.6.

The reboiler steam flow rate in the third plot of figure 3.4 rises well before the issues in both De-Eth DP and product rate occur, meaning that this hypothesis is reasonable. It is also important to note that, for both De-Eth DP and reboiler steam flow rate, the process behaviour is very stable in the regions before the drop in product rate; this concept is key to identifying 'abnormal process behaviour'. Digging further using this same technique led us to the final culprit, the 'De-Eth Btms C2', which represents the relative amount of lighter hydrocarbons in the system. The bottoms C2 content also represents the purity of the bottoms stream in the de-ethanizer column. The compiled findings of the diagnosis are shown in figure 3.5.

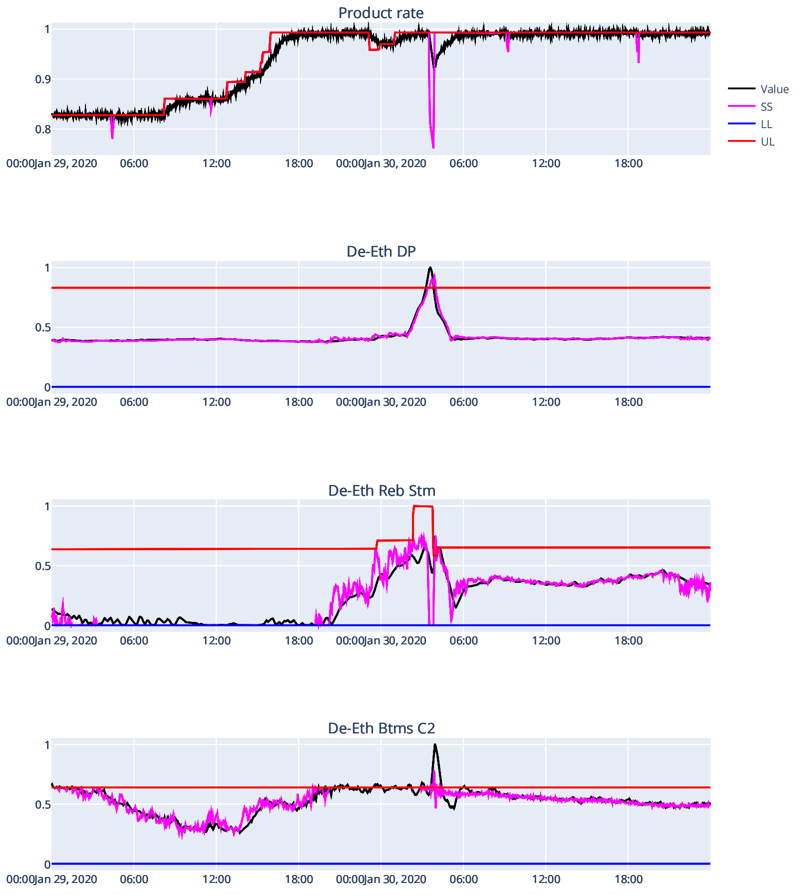


Figure 3.5: Compiled findings of the controller diagnosis, investigating the drop in product rate at around 3:00AM on January 30. As C_2 content hit its upper limit, the controller increased steam flow in the reboiler to drive the lighter hydrocarbons to the top, sharply increasing pressure drop throughout the system. As the De-Eth DP hit its upper limit, the controller relieved the system by cutting the feed rate.

Conceptually, higher bottoms C2 content (De-Eth Btms C2) means that the bottoms stream of the column contains larger proportions of lighter hydrocarbons, meaning that the bottoms stream is less pure than it should be. Consequently, the reboiler must provide more heat to drive lighter components to the top of the column. However, increasing the steam flow through the reboiler causes an increased pressure differential due to the greater amount of vapour re-entering the column from the reboiler. Consequently, to bring the system back to an equilibrium point, the controller can reduce the product rate target. Doing this reduces the downstream demand, relieving the reboiler of the high load required. This diagnosis came about as a result of analyzing process data in this way; using visualization to generate hypotheses, then using process knowledge to verify them.

Evidently, it is necessary to incorporate process knowledge in multiple stages of diagnosis; it is required when deducing which variables are causing which problems, and is also necessary to know when to stop 'digging' through variables[2]. Through consultation with industry partners, we conclude that the root cause of the problem is in the bottoms C2 content in the De-Eth column, because this is a variable concerned with the conditions of one feed stream entering the plant.

[2] Otherwise, we are left with 'naive' methods of diagnosis where variables are checked based on their position in the gain matrix, instead of how likely they are to cause the problem.

3.1.2 *Rigorous task definition*

The task analysis provided in section 3.1.1 is a discussion of the tasks required to diagnose the MPC using the gain matrix in domain-specific language; it is not a generic description of the workflow. For this task analysis to be useful in developing a visualization tool, the task analysis is abstracted into the definitions stipulated by Munzner's multi-level typology[3].

[3] Performing a detailed and methodical task analysis provides an additional benefit, which is that the user can isolate specific tasks without performing an entire controller diagnosis. The visualization can therefore be designed to be used for these individual tasks, making it more user-friendly.

The controller diagnosis steps discussed in section 3.1.1 can be broken down as follows:

1. Identify the problem variable (referred to as PV_1 in this discussion)
2. Pinpoint the time at which PV_1 is causing a problem
3. Find the row/column in the gain matrix corresponding to PV_1
    - *Input*: a variable identifier (tag name/description)
    - *Output*: a row/column of variables with associated gains
    - *Why*: to examine how PV_1 is related to others in the model
        - Discover $\rightarrow$ Locate $\rightarrow$ Compare
    - *How*: Encode
4. Find all variables associated with PV_1 in the gain matrix (i.e. variables with **nonzero gains**)
    - *Input*: row/column in the gain matrix

- *Output*: a set of MPC variable identifiers with nonzero gains
- *Why*: to narrow down the possible variables that are causing a problem
  - Produce
- *How*: rearrange the gain matrix based on the gain values in the row/column
  - Select + Arrange

5. Examine each of those variables to find which one is exhibiting abnormal behaviour

- *Input*: set of MPC variable identifiers with nonzero gains
- *Output*: single MPC variable that is likely to be the cause of the problem
- *Why*: to identify the variable that is causing the problem, and to set up the process of digging further
  - Discover $\rightarrow$ Lookup $\rightarrow$ Identify for each variable
  - Compare $\rightarrow$ Summarize when variable characteristics have been identified
- *How*: Lookup the variable and present its data; Analyze its behaviour; Decide if the behaviour is 'abnormal'
  - Encode + Introduce $\rightarrow$ Import $\rightarrow$ Manipulate $\rightarrow$ Navigate + Filter

6. Repeat this process until a reasonable root cause of the problem is identified

The task analysis definition can be summarized as a flowchart. This representation is shown in figure 3.6.

3.2 *Visualization development process*

Each layer of the nested model mentioned in section 2.1 is addressed.

Characterizing the domain problem consists of framing the problem statement in a way that outlines *why* the visualization is necessary, using domain-specific jargon. The domain problem here is that engineers[4] working with MPC systems need to monitor and diagnose MPCs using process data and process knowledge.

[4] Specifically, those involved in *operational monitoring*.

Data and task abstraction involves identifying data sources and tasks, and expressing them in a generic, computer science-based language. There are three sources of data for this project:

1. Time-series process data, with both continuous and discrete values;

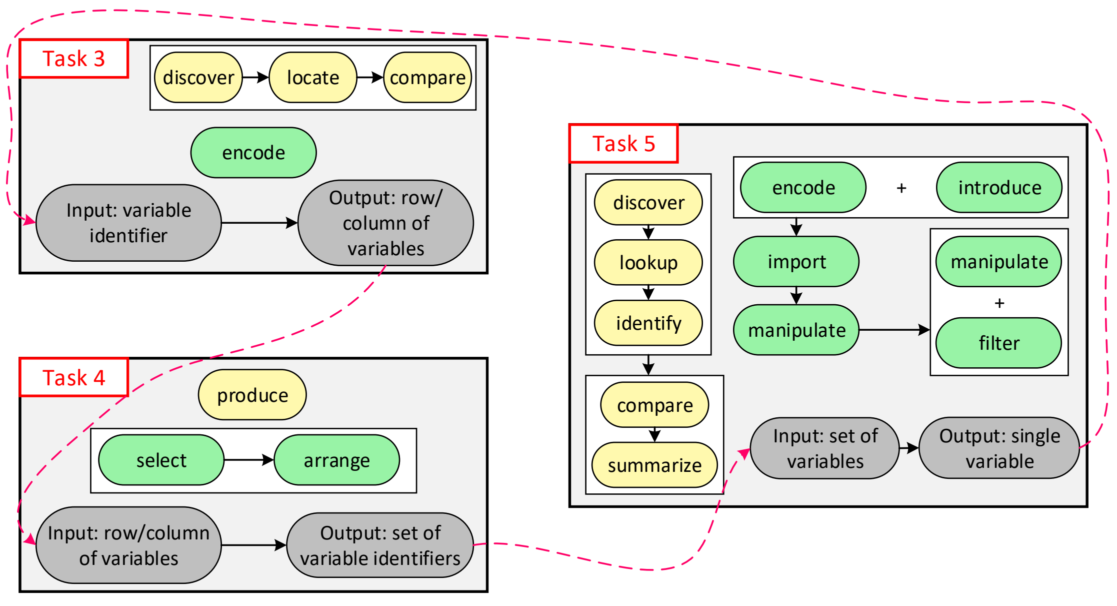


Figure 3.6: Defined task analysis based on the multi-level typology. Yellow blocks indicate the *why* of a task, grey ones indicate the *what*, and green ones indicate the *how*. Red dashed arrows represent transitions from one task to the next; note that, as defined in step 6, this process involves repetition of steps. Steps 1 and 2 are not defined in this way.

2. MPC variable metadata (i.e. tag names, descriptions, variable types, etc.);

3. Gain matrix data.

Abstracting the data into generic computer-science based data sources was not a trivial task, and required in-depth understanding of the principles of Object-Oriented Programming (OOP). OOP is based on the idea of designing programs to manipulate user-defined 'objects' rather than functions and logic. These objects are data structures that each have their own attributes and methods, allowing large data sets like the ones used here to be broken down into more intuitive and workable chunks. OOP was used to create an MPC variable 'class' that contains its measurements (raw value, steady-state target, engineering limits) as well as its metadata (description, type, LP cost, etc.)[5]. The process of creating these classes and ensuring that each dataset is in the correct form was particularly tedious, since there were some discrepancies between variable names in the metadata and the process data. The final class structure included all relevant data to the specific MPC variable, as well as functions for plotting, filtering, and normalizing that dataset. The end result is a set of MPC variable objects, each representing a physical variable

[5] For instance, the process data from Parkland was provided as a single tabular file with the tag name, timestamp, and value as columns. Each MPC 'variable' has numerous tag names that represent the measured value, engineering limits, etc.

in the plant, that have their respective datasets attributed to them. This is similar to how process variables are structured in Supervisory Control and Data Acquisition (SCADA) systems, where each process variables has a set of tags that each represent a type of measurement (value, SS, UL, etc.). The task abstraction is discussed in section 3.1.

Visual encoding and interaction are especially important because the project itself is based on the lack of visual encoding and interactivity in existing tools for MPC visualization. There are two principal datasets for which the design of visual encoding and interaction is especially relevant: the gain matrix and the variable constraint data. The chosen encoding for both of these datasets is a colour-coded heat map, and these decisions are justified in section 3.2.1.

Algorithm design is the creation of code that accomplishes the goals set out by the previous layers. As mentioned in the section discussing data abstraction, MPC variables are expressed as objects belonging to a user-defined class, each containing all necessary attributes and functions. Python code was used to accomplish this, using numerous libraries such as Dash, Pandas, and Numpy. Jupyter notebooks were used extensively in testing the created classes and in creating the data structures to be presented in the final visualization tool. The chosen form of the final tool is a web application hosted through the Dash library in Python, which combines Plotly objects and HTML/CSS components to create an interactive interface. The reason why Dash and Python were chosen to create this tool is because of my familiarity with Python. An initial attempt was made to leverage existing open-source tools like Clustergrammer [31] to eliminate the need for creating visualization tools from scratch. However, due to my limited fluency in Javascript and relative proficiency in Python, I used Plotly and Dash to create these tools.

3.2.1 *Design decisions and justifications*

Decisions regarding visualization design are mostly relevant to the 'visual encoding and interaction' layer of the nested model. The first dataset discussed in section 3.2 is the time-series process data for each variable. These are represented as line plots, with the various tags relevant to each MPC variable represented in different colours. Figures 3.1 - 3.5 show this colour scheme.

Gain matrix

As shown in section 3.1.2, the gain matrix plays a crucial role in aiding controller diagnosis. Hence, its representation and interactivity are essential components of the visualization tool.

The gain matrix is a 2-dimensional matrix containing numerical values in each cell. The raw gain data are continuous numerical values, making it a quantitative dataset. The chosen representation for this is a heat map, since it is, by definition, a matrix. Additionally, the gain matrix uses nominal labels, rather than quantitative ones. The choice to use a heat map was also verified by the extensive use of heat maps to represent gene expression data in the area of bioinformatics, which involves similar datasets. The first iteration of the heat map representation used a *continuous* colour scale to represent raw gains, with negatives in red, positives in green, and zeros in white. Some additional background is required in order to justify this choice of colours.

Visualization literature commonly defines a *colour ramp* as an encoding of ordinal or continuous numeric data associated with a specific colour scheme, either using sequential or diverging colours. The specific colour ramp chosen should effectively encode the numerical data in the gain matrix, and present the data in a way that is easily comprehensible and aesthetically pleasing. The chosen colour map is meant to adhere to the concept of *perceptual uniformity*: "the concept that colors should convey the differences between the values they are representing." [32]. Smart et al (2020) also outline other guidelines for colour maps:

> "Perceptual guidelines emphasize ways to make ramps intuitively mirror the underlying data. Sloan & Brown [70] stress that colors in a ramp should be maximally distinguishable and follow an easily remembered order." [32]

Green and red are traditionally complementary colours, so it follows that they can be sufficiently distinguishable. The chosen colours also maintain similar saturation and luminosities, making for a smooth transition between them. Additionally, using a 'neutral' colour like white to represent zero values allows the heat map to blend with the visualization background more effectively, which makes for a nicer-looking visualization.

Representing the gain matrix as a continuous heat map provided an improvement to the raw display, but presented a new issue. The real FCC gain matrix is fairly sparse and exhibits a large spread of values[6]. The large spread of values, combined with the sparsity of the matrix, cause most of the cells in the heat map to be white or near-white in colour. Choosing a different colour to represent zero values presented the same problem, where one neutral colour takes up most of the visualization and a handful of cells are distinguishable. The issue here is that a colour-coded heatmap representing raw gains did not actually improve the usability of the display; rather, this made the display unusable. The continuous representation of the raw FCC gain matrix is shown in the leftmost heat map of figure 3.7. Even though this issue may seem specific to this context, this

[6] 'Real' is used here because the gain matrix needed to be normalized in order to comply with data distribution agreements. The gains shown here range from 0 to 1, but their spread is proportional.

is likely not the case. In any industrial process, large multivariable control systems often have sparse gain matrices [23], so it is likely that the issue of having indistinguishable white/near-white cells would persist.

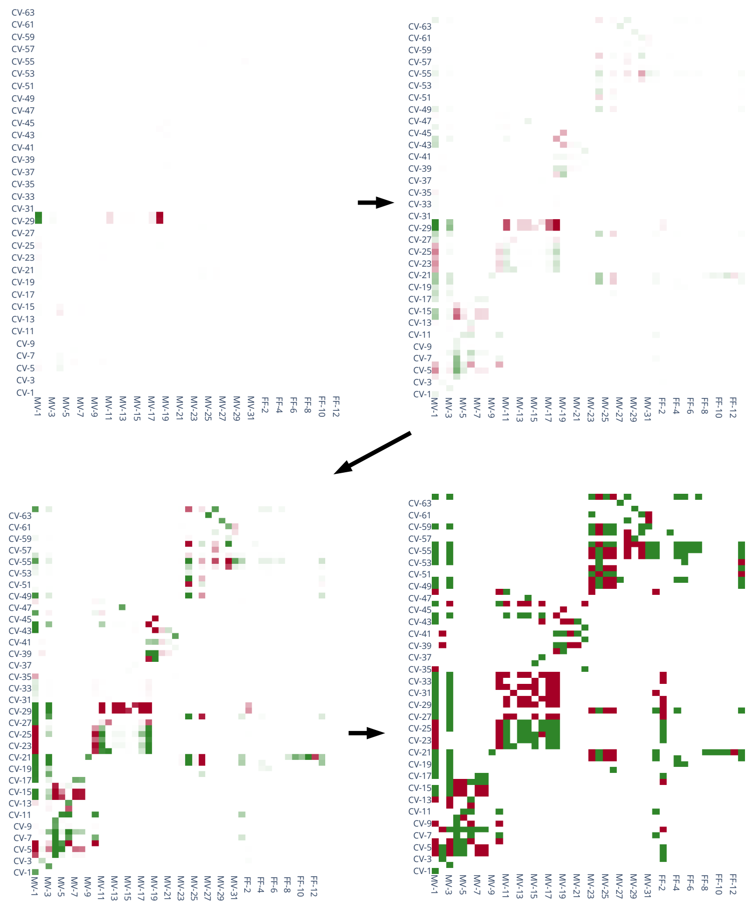


Figure 3.7: Heat map representations of the FCC gain matrix as the project progressed. Following the arrows (left to right, top to bottom) the heat maps represent the following forms of the FCC gain matrix: raw FCC gain matrix, log-transformed matrix,, tanh-transformed matrix, discrete matrix. Note the variability in distinguishable cells as the heat maps progress.

As seen in figure 3.7, encoding the raw FCC gain matrix using a continuous colour scale does not provide a more interpretable representation of the system. Rather, it now requires the user to hover over each cell in order to obtain the gain value, which undermines the original purpose of the heat map representation. Looking at the data, this issue was found to be caused by an unusually skewed data distribution. It is important to understand the raw gain data distribution in order to obtain a transformation that is suitable to the dataset. The raw gain data range from around -1000 to $+2500$

with around 13% nonzero values and a very small proportion of gain values with magnitudes above 50[7]. Consequently, the heat map encoding of the *raw* gain matrix shows most cells as either white or near-white. This issue of having an unusable heat map due to numerous indistinguishable values in the heat map may be solved by performing a nonlinear transformation on the dataset, such that the distribution is 'squashed'. Bringing the distribution closer to normality in this way results in data that do not skew the colours being used, thereby making cells in the heat map more easily distinguishable. This transformation must effectively compress the data such that near-zero values are distinguishable from zero, while extreme values are not emphasized. Eisemann et al (2011) discuss various transformations common to non-normal distributions:

[7] The raw gain data have a standard deviation of 71.50, and only 0.46% have magnitudes greater than 50.

> "However, real data of a variety of fields present a non-normal behavior [Mic89]. There is a great variety of transformations that are used to improve normality of variables, e.g. adding or multiplying constants, taking the square root or converting to logarithmic scales... [The logarithmic transformation] can be used to transform variables that are right skewed and generate pleasing visualizations with an otherwise bad resolution distribution. In such visualizations, a few large values take up most of the color map scale and the rest of the data points are squashed into a small part of the scale with low resolution." [33]

In their discussion, Eisemann et al (2011) propose an alternative method for data transformation that is specifically geared towards the data being encoded. Their method involves sorting and normalizing the data, then projecting each data point onto a vector with a user-specified angle; the choice of angle decides which points are to be emphasized in the transformation. This method may provide an effective data transformation; however, due to relative simplicity and time constraints, we first chose to investigate a logarithmic transformation. In this case, the log transformation is not simply taking the logarithm of each data point, because there are negative values in the gain matrix. The relation used to transform the gain matrix is shown in equation 3.1.

$$K'_{ij} = \text{sign}\left(K_{ij}\right) \times \log_{10}\left(\left|K_{ij}\right| + 1\right) \tag{3.1}$$

Where:

- K_{ij} = the gain matrix entry in row i and column j
- K'_{ij} = log-transformed gain
- $\text{sign}(x)$ =the sign function, which returns either -1, 0, or 1

1 is added to the gain value in the logarithm term so that any zero gain values result in a log-transformed value of 0 instead of $-\infty$.

After transforming the gain matrix using a logarithm function, the heat map representation becomes marginally clearer. The modified heat map is shown at the top-right of figure 3.7. The logarithmic transformation reveals many more nonzero gain values compared to the raw gain heat map, but the number of distinguishable cells is still insufficient for the presentation of underlying patterns in the data. The logarithmic display improves discriminability of cells in the lower-left section of the gain matrix, but the top-left and top-right sections exhibit poor discriminability. We can confirm this by comparing the log-transformed gain matrix to the discretized gain matrix in the bottom-right heat map in figure 3.7. While the logarithmic transformation is a useful nonlinear computation, it does not appear to effectively squash the spread of data.

One other data transformation that is useful in squashing datasets with unpredictable distributions is the hyperbolic tangent function, or tanh. The hyperbolic tangent is defined in equation 3.2.

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.2}$$

A graphical representation of the hyperbolic tangent, along with the logarithmic transformation, is shown in figure 3.8.

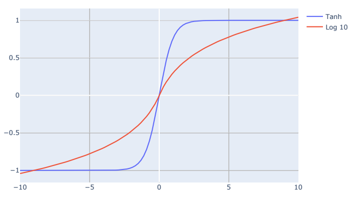


Figure 3.8: Transformation functions for the hyperbolic tangent (blue) and base-10 logarithm (red). The hyperbolic tangent transformation is the same as its function, whereas the logarithmic transformation is described by equation 3.1. Note the larger magnitude of the tanh function for small nonzero values compared to that of the logarithm function.

The hyperbolic tangent is more useful than the logarithm, in this case, because the range of its output is $[-1, 1]$, while that of the logarithm is $(-\infty, +\infty)$. Both transformations are approximately linear at zero; this is important because most gain values are at or near zero. At zero, the approximately linear hyperbolic tangent has a higher slope than the logarithmic transformation. The combination of these factors means that near-zero values are emphasized, while extremely large values are squashed[8]. For instance, the hyperbolic tangent would yield a larger transformed value for a gain of 1, but for a gain of 10 or higher, the transformed value would be 1. On the other hand, a gain of 1 would yield a smaller transformed value through the log-transform, while a gain of 10 or higher would be greater than 1.

There are myriad other nonlinear transformations that may effectively present all nonzero gain values. The limitation of using the logarithmic transformation, along with the alternative transformation methods, are discussed in chapter 4.

An alternative solution to this issue is not through finding a different visual encoding or data transformation, but by changing the type of dataset used. The gain matrix contains quantitative data, but this form is not necessarily the most effective. Discussions with industry partners revealed that, for understanding the relationships between variables in the context of the LP optimization, users are

[8] For this reason, hyperbolic tangent functions and similar functions (such as the sigmoid function) are sometimes used in machine learning as 'activation functions', especially in neural networks. Inputs to the activation functions are unknown because they are calculated iteratively through the 'learning' process, so having this bounded output can be useful.

often more concerned with the gain direction than with the actual gain value. The direction refers to the sign of the gain in a particular MV/CV pairing, where all possible direction values are either positive (+1), negative (-1), or zero. Note that, for the general controller diagnosis procedure in section 1.3.6, example gain values are classified as either positive, negative, or zero. The raw gain matrix can therefore be transformed using a sign function to reflect the direction of each pairing. Transforming the gain matrix in this way changes it from a quantitative dataset to an ordinal dataset, and greatly simplifies the visualization while retaining much of the information being conveyed. The resulting heat map presents process gains as one of three colours: red represents negative values, white represents zero, and green represents positive, as shown in figure 3.9.

Comparing figure 3.9 to the original gain matrix display and to the other displays in figure 3.7, some benefits of implementing this type of design study appear. It is apparent that by simply transforming the dataset using a sign function and encoding the values as colours, end users can perform required tasks without an intense cognitive load. Sorting the discrete heat map causes zero and nonzero gains to be sufficiently distinguishable, as there would be a very clear boundary in the colours of the sorted cells. Correll et al (2018) provide a comparison of discrete and continuous colour mappings:

> "While continuous color maps afford greater fidelity in presenting values, non-linearity in human color perception introduces errors in extracting numeric values from continuous colors. Quantizing a color map is therefore an exercise in balancing perceptual error and quantization error. Discrete maps offer finer control over this balance, which can result in better performance in tasks involving heatmaps." [34]

In presenting continuous forms of the gain matrix - with both raw data and transformed data - it is difficult to predict exact values or even approximate values just by looking at the colour. Even knowing where the maximum, minimum, and zero lie on the colour scale, predicting gain values accurately from the colour scale is difficult. This is made more difficult in the logarithmic scale, where the colour ramp encodes some nonlinear transformation of the original data. Overall, the continuous colour ramps shown here are the cause of this *perceptual error*, and may actually increase the cognitive load required of the users when coupled with complex, nonlinear data transformations. Using a discrete colour ramp to encode an ordinal version of this dataset eliminates much of this perceptual error, as the boundaries between different values become much clearer. However, discretizing the gain matrix in this way also means that all information regarding the magnitude of gains is lost (*quantization*

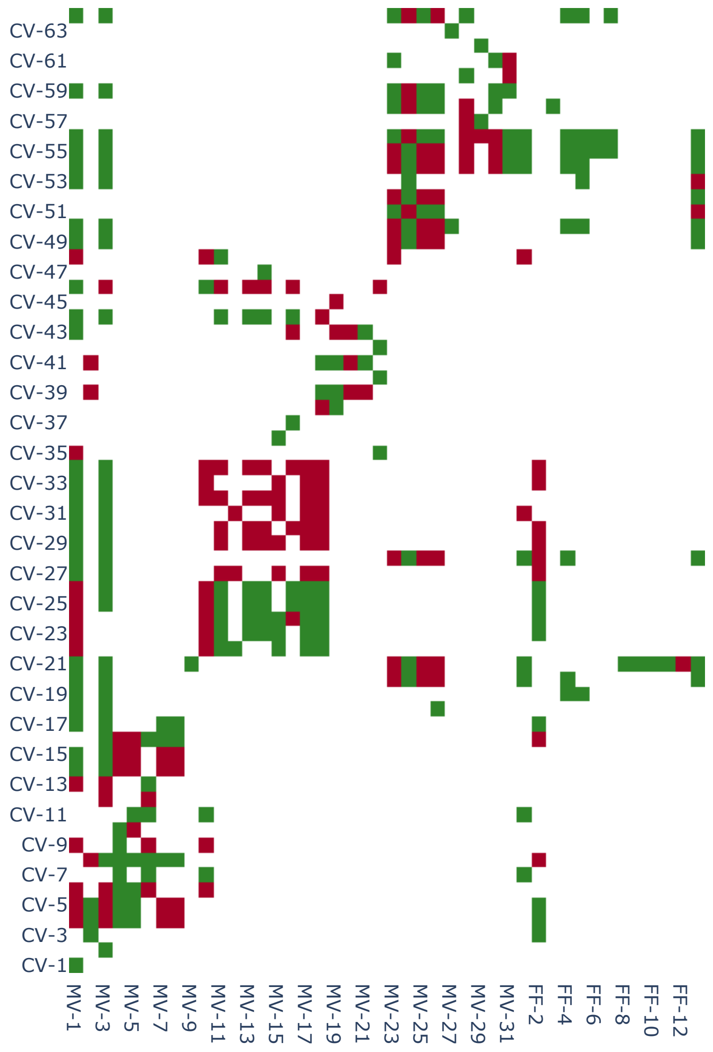


Figure 3.9: Discrete heat map showing the gain matrix as an ordinal dataset. Positive values are in green, negatives in red, and zeros in white.

error). A comparison of the colour ramps in continuous and discrete form is shown in figure 3.10.

This is an important tradeoff to consider; encoding more information (i.e. direction + magnitude of gain values) may reduce the discriminability of points in the display, but reducing the data to only the directions means that the user cannot obtain gain values from the visual encoding of the gain matrix. It may therefore be better to explore alternative data transformations, such that each gain value in the matrix is sufficiently distinguishable while retaining all information present in the raw gain matrix. Another alternative could be to use different visual channels to encode gain magnitude and direction, similar to the Bertin reorderable matrix [28]. For instance, directions can be encoded as a colour (as in the discrete heat map here), while magnitude is encoded by the size of a shape in the display. Siirtola et al (2005) describe this feature of Bertin's matrix:

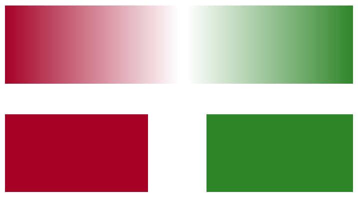


Figure 3.10: Comparison between discrete and continuous colour ramps. From left to right, the colours encode negative (red) to zero (white) to positive (green) values.

> "The actual data values are replaced with symbols, say circles or rectangles, which have a size relative to the actual data value. The smallest value is represented as a 0-sized symbol and the largest value as a symbol filling the whole area available. While interacting with the visual presentation, the user has a chance to detect patterns in the presentation and to gain insight into the data. This kind of pattern recognition is something that human vision is known to do remarkably well." [35]

Using a combination of visual encodings in this way was not implemented in this project, but may prove to be useful in visualizing gain matrix data. Further investigation is needed to evaluate such designs and this is discussed in more detail in chapter 4.

Aside from the encoding strategies used for the gain matrix, the interactivity of the interface must be addressed. The interactivity required from the gain matrix display pertains to steps 3 and 4 in section 3.1.2. Step 3 is not exactly a physical task performed on the matrix; rather, it represents a more cognitive task, where the user identifies a location in the matrix to be used as an input for the next step. Step 4 involves sorting the gain matrix based on gain values paired with the variable being investigated, and filtering out variables that have zero gain. Currently, this functionality is provided by selection of a particular variable using one of two dropdown lists, where each dropdown list represents the labels along one dimension of the gain matrix (in other words, one dropdown for MVs/FFs and one for CVs). It is likely that there are better alternatives to the dropdown setup, and these are discussed in chapter 4. At the moment, no functionality has been implemented to automatically filter out rows/columns with zero gain, due to limitations in the algorithms employed as well as time constraints. Nevertheless, the functionality to sort the gain matrix based on an individual variable is novel in this field, and so is an essential component of this visualization tool.

Constraints

To describe and justify the design decisions made in encoding the constraint data, the structure of the data must first be discussed in detail. The constraint data for a given MPC variable describe the state of the variable with respect to the LP, and are in the form of integers. Each integer is linked to a specified state, as outlined in table 3.1.

Number	Variable state	Colour on heat map
0	Unconstrained	White
1	Constrained at upper limit	Red
2	Constrained at lower limit	Blue
5	Lower SSSTEP limit	Blue-gray
6	Upper SSSTEP limit	Dark blue-gray
8	Out of service	Black
9	Ramp	Grey
11	Not in LP solution	Yellow

Table 3.1: DMC constraint data and the associated variable states with respect to the LP. Each non-FF MPC variable has a constraint tag.

Table 3.1 provides a clear indication that this is a nominal dataset. Despite the states being represented by integers, the ordering of integers has no relation to the information they convey. It is important to consider that this dataset is also a time series. Since the constraint data are not quantitative, there is no real benefit in encoding them in quantitative visual forms, like length or area. Using similar principles as those in gain matrix encoding, a heat map is chosen to encode constraint data. White is chosen to represent unconstrained values, as is red for values constrained at the upper limit, and blue for values constrained at the lower limit. This is to maintain consistency with the colour scheme of the process data plots. Black represents variables that are out of service, and the remaining colours are chosen based on aesthetic preferences; this decision is justifiable because the constraints are at these points (5, 6, 9, 11) for small amounts of time. The final constraint heat map result is shown in figure 3.11.

Constraint data need to be visualized because they may provide an easier method for controller diagnosis and investigation than the method presented in section 3.1. By nature, multivariable control systems are often operating at some intersection of process constraints, as explained in section 1.3.4. Specifically, in this plant, feed rate is often constrained at its upper limit. This can be seen in figure 3.11 near the centre, shown by the 'CV-1' row; this variable shows red cells for the majority of the time span. Consequently, when one CV

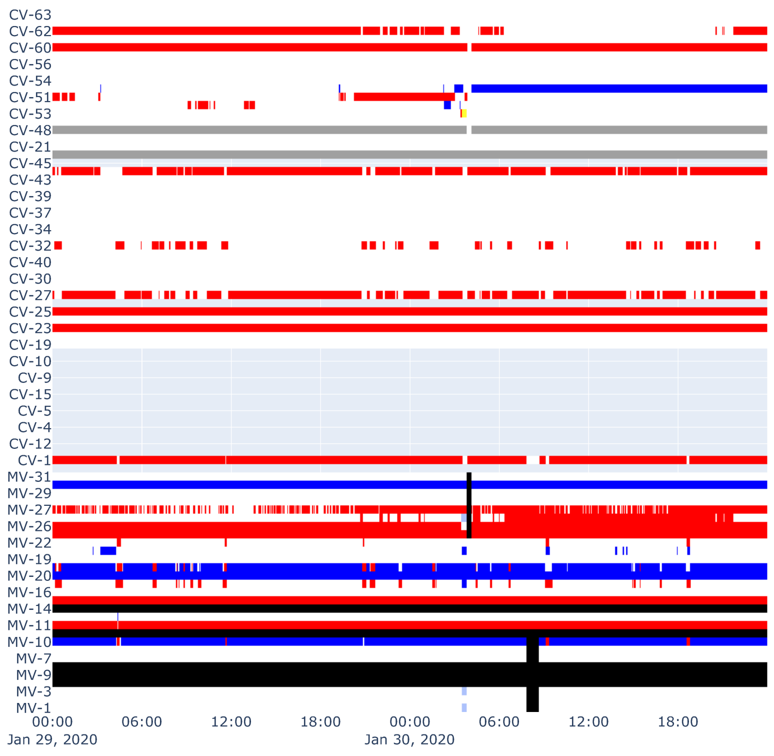


Figure 3.11: Heat map design chosen to encode constraint data. Colours are defined in table 3.1.

goes from being constrained to being unconstrained, that means that one MV has hit its constraint.

The constraint heat map can therefore be used to quickly identify problem areas when diagnosing the controller. From the task analysis in section 3.1.2, when a problematic time period is identified (at the end of step 2), the user can simply highlight that time period in the constraint heat map. From there, the user would need to visually locate variables that go from white to red or blue. This would allow the user to go from step 2 to step 5 in a single task, without the need to go through the gain matrix.

There are some drawbacks to using this method, firstly that the amount of variables the user needs to go through is larger than that of the gain matrix. This is because the constraint heat map has all variables on the y-axis, whereas the tasks presented here involve successive filtering of variable sets. To address this, the constraint heat map can use filtering functionality based on variable type or relation to the problem variable. Secondly and, more importantly, the current form of constraint data does not represent the relationships between variables. For users who are familiar with the process and the physical meanings of each variable, this poses no issue; for those who are not as familiar, they would require a way to relate the variables they are investigating to each other. There is a relationship between the gain matrix and the variable constraints, which is explored in section 1.4, but it has not been incorporated into this visualization tool. One important component of the future work discussed in chapter 4 is the investigation of this relationship, as it may provide more effective methods for performing controller diagnosis. The results of this investigation can also be incorporated into a visualization to provide additional benefits.

3.3 *Final visualization tool*

3.3.1 *Features*

This section does not provide a detailed workflow of the application; rather, it discusses the important features provided in the application.

The application begins with a representation of the gain matrix heat map, along with two dropdown lists and two checklists containing the names or identifiers of each MPC variable (there is one dropdown and one checklist for MVs/FFs and one of each for CVs). The checklists represent variables that are plotted on the left side of the screen, while the dropdown lists represent the variable on which the gain matrix sorting is based. There is also a "Reset" button, which removes any process plots on the screen.

In addition to the reordering functionality, users have the ability to examine plots for specific MV/CV pairs in the gain matrix. Selecting a cell in the gain matrix highlights the gain of that particular MV/CV pairing, and automatically plots the process data of each variable in said pairing on the left side of the screen. This is shown in figure 3.12.

The selection tool is shown by the red arrow in figure 3.12; this highlights which MV and which CV are being selected. The z value of the pairing is the gain, but since this is simply a sign function, its presence may be redundant. Selecting variables to plot in this way automatically updates the checklists below with the variables being plotted.

Selecting a variable from either dropdown list rearranges the gain matrix based on the values in the row/column of that variable. Rearranging using a dropdown list requires the user to know which variable they are looking for, which may not be an effective way to perform this task. This limitation is discussed further in chapter 4.

The checklists contain all variable names, sorted as either MVs/FFs or CVs. Selecting a variable in the checklist displays its process data

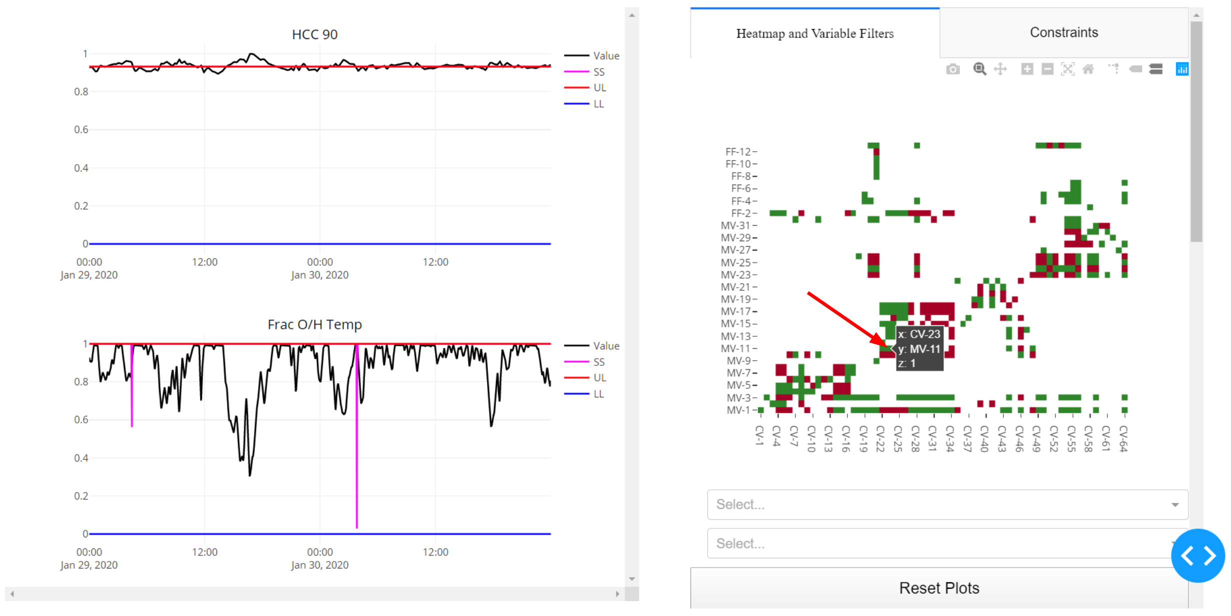


Figure 3.12: Selecting the gain matrix to produce process plots. The red arrow indicates where the cursor was clicked, showing the relevant process plots on the left. Notice the pop-up at the selection point; this indicates which variables are being selected (x and y values), as well as the gain (z value).

on the left side of the screen.

The combination of the gain matrix heat map, checklists, and dropdown lists provide the functionality required to accomplish the tasks presented in section 3.1.2. The methods by which these tasks are accomplished in this display may not be the most effective, but the idea is to introduce this functionality into APC applications.

The next major feature is the constraint heat map, displayed as a secondary tab on the same screen. This heat map is always present and is not currently permutable; due to time constraints, we did not undertake an in-depth task analysis that uses this dataset. A display showing the 'Constraints' tab is shown in figure 3.13.

Presenting the constraint heat map is expected to 'set the stage' for including constraint data in the process of controller diagnosis, as discussed in chapter 4.

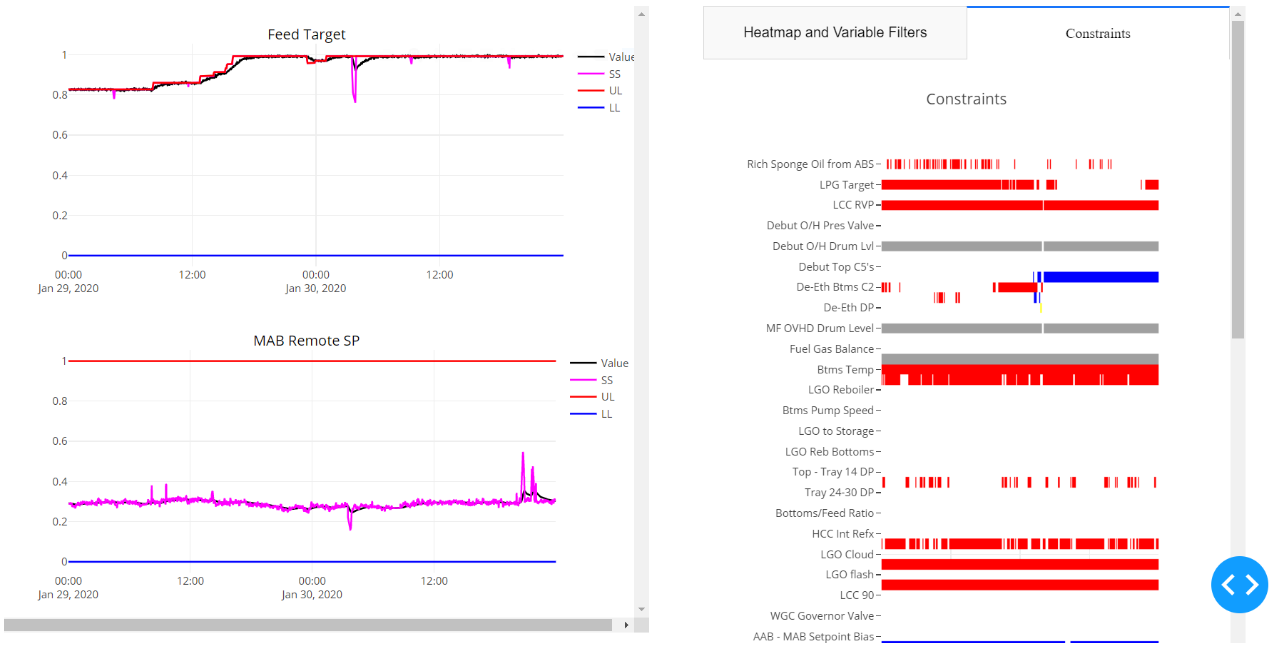


Figure 3.13: Application display showing process plots on the left as selected from the checklists, and the constraint heat map on the right.

4
Limitations and Future Work

The limitations of this research can be split into two broad categories: conceptual, and practical. Conceptual limitations refer to the pitfalls of the ideas behind the research, while practical limitations refer to issues caused by the specific methods and tools employed.

One major conceptual limitation in this project is the use of raw gain values. This does not refer to the usage of raw gains in the heat map; rather, this limitation is related to the presence of the raw FCC gain matrix altogether. Raw gain values are often dependent on the engineering units in the Distributed Control System (DCS), meaning that the gain values scale according to the relevant units of measurement. For instance, if one CV is measured in barrels per day (BPD) and another is measured in thousands of barrels per day (MBPD), the magnitude of their gains would differ by a factor of 1000. In reality, their gains are equal, and so the raw gain matrix may not always provide a good representation of the system. Furthermore, during controller commissioning, control engineers analyze the gain matrix in order to assess how well the model matches the plant. Some of these methods of analysis, such as SVD, scale with the units of measurement chosen. The choice of units can therefore affect how engineers analyze and modify the model, and so the use of a unit-independent representation of the system can provide significant benefits. One such representation uses each variable's 'Typical Move': this is the amount by which the controller typically moves a given MV [36]. Using this to scale gain values should yield a better representation of what the 'real' gains are, because it allows different variables to be compared on an equal scale. Visualizing the raw gain matrix thereby provides little benefit, as shown in figure 3.7.

This limitation was introduced in the discussion of various data transformations to be performed on the raw gain matrix in section 3.2.1. Since the raw FCC gain matrix is far from normal[1], there are numerous options for transforming the dataset to make it more

[1] A histogram of the raw FCC gain matrix exhibits one large peak (around a gain value of zero) and two distant outliers. Most gain values are around or exactly zero, while a small number of MV/CV pairings has extremely high/low gains, making this distribution highly skewed.

normal. Examples of alternative transformations include typical move scaling and the sigmoid transformation. The spread of data was somewhat improved by logarithmic transformation, but this did not result in a much better visualization. This is because the logarithmic scaling greatly reduced the emphasis on outliers, but did not increase its emphasis on small nonzero values. Consequently, the issue of distinguishing between near-zero and nonzero values persists, so an alternative transformation may provide an improvement. One such transformation is that presented by Eisemann et al (2011), which uses the dataset and user-defined parameters to create a transformed dataset that is geared towards the goals of the user. For example, the main design parameter in this transformation is an angle α, which determines how each data point is projected; some values of α favour the emphasis of outliers, while others put less emphasis on outliers and focus on more central data points.

Another major limitation is the lack of a data pipeline[2]. A data pipeline would allow transfer of data from the plant itself (from the DCS) through databases and IT systems to the designers and users of visual/analytical tools. Consequently, without a data pipeline, it is not possible to build tools for visualization or analysis that use real-time data; such tools would only involve historical data.

[2] A data pipeline is a set of data processing elements that transport data between systems, and that filter and modify data as necessary.

Finally, the lack of a process model gives rise to another limitation, in that there is no information on the dynamics of the process. The gain matrix provides steady-state data, and so the tool cannot provide information regarding how quickly or aggressively one MV would affect a particular CV. This is a minor limitation, and incorporating process dynamics may provide some benefit in the future[3].

[3] AspenTech DMCPlus plots process response curves in the gain matrix, but with large controllers, it is difficult to navigate through specific MV/CV pairings.

The practical limitations of this project are related to the use of Python and Plotly in developing the visualization. One major limitation is the use of dropdown lists to provide sorting functionality on the gain matrix. While the user can sort the gain matrix using dropdown lists, the user needs to keep track of the variables they are investigating in order to do so. The previous attempt to create this functionality in Clustergrammer allowed the user to double-click a variable name on the axes to reorganize the matrix in this way, which was much faster and more effective. Tools like Clustergrammer provide greater customizability; Clustergrammer is especially useful because it is designed specifically to handle heat maps. Because Clustergrammer is built for bioinformatic data, it allows designers to easily provide additional categories to link variables on both axes; for instance, all variables that operate under a specific subcontroller can be grouped together in the heat map. Users can also sort the heat map based on LP cost or variable type. Dash does not provide tools to build this functionality.

There are four important issues that should be addressed in the future. Firstly, we need to test the visual encodings in the tool more extensively, as well as test other types of encodings to ensure that the design decisions made result in an effective interface. This stage will include investigating various data transformations to be performed on the gain matrix, such that the spread of data can be adjusted to make it suitable for a wider variety of visual encodings. This stage will also include testing different visual encodings and strategies to represent gain matrix data, such as using a matrix similar to the Bertin matrix as discussed in section 3.2.1. At the end of this stage, we should be confident in that the chosen visual encoding and interaction strategies effectively represent the data, and that the data have been modified to be suitable for visualization. This form of evaluating the designed tool can be classified as a *formative* evaluation, which is "intended to provide guidance to the designers on how to improve a system and answer the question 'can I make it better?'" [4]. To answer this question requires regular consultation with users, such that their needs are being prioritized in the design of the tool.

Secondly, we need to extensively test the visualization tool to vary the datasets being used, the plants, and the controllers being investigated. Testing with more variable sources of data and more variable use cases will allow us to confirm, with a greater degree of certainty, how applicable this tool is. These sources of data include, but are not limited to: MPC data from different software packages (Shell SMOC, Honeywell RMPCT), data from different plants, and different case studies for controller diagnosis. To summarize, these tasks comprise a process used to evaluate the visualization tool, which can be considered a *summative* evaluation process[4]. This stage would involve more formal consultation with users than that of the first stage, as this summative evaluation is a measure of the tool's performance.

[4] Summative evaluation is "intended to measure the performance of a system and answer the question 'is it right?'" [4]

The third major point that will be addressed is the incorporation of constraint data in the process of controller diagnosis. The tasks provided in section 3.1.2 require users with process knowledge to perform steps 5 and 6. Using the constraint data as part of controller diagnosis may reduce the required amount of process knowledge necessary to diagnose the controller, because the variable constraints and their changes indicate potential problem areas in the plant. Doing so would require investigating the relationships between variable constraints and the gain matrix, which adds another layer of complexity to the problem.

The gain matrix and variable constraints can be correlated in order for operators to deepen their understanding of the process. These correlations can be used to quantify, for example, how much one MV can be changed before the variable becomes constrained and the controller moves to a different operating point. Another example would be how much one MV can be changed before a correlated and constrained CV becomes unconstrained, causing the controller to move to a different set of constraints. These analyses need to be explored extensively, and the data and tasks involved need to be analyzed in order to provide greater analytical capabilities that use the constraint data. For example, in their patent for a solution analysis process for MPC systems, Peterson et al (2011) outline methods by which to calculate how much operators can relax variable constraints [20].

The next major point that will be addressed in future is the investigation of different software packages to create this tool, which was mentioned in chapter 4. Tools like Clustergrammer, which provide greater functionality with heat maps, can be immensely beneficial, as Dash and Python do not provide much customizability in presenting heat maps[5]. For example, the first attempt to create a visualization tool using Clustergrammer allowed the user to double-click a variable name on an axis, causing the matrix to rearrange based on that row or column. This may seem like a trivial difference, but using dropdowns instead of a simple double-click complicates the reordering functionality, thereby undermining the overarching idea of the project. Thus, it is necessary to explore different software packages such that the required functionality is present.

[5] For instance, the Plotly library reserves double-click events for resetting the display after the user zooms or pans any graph display.

There are some additional considerations for future work that may prove relevant. Parallels can be drawn between the diagnosis process and the design process employed by Lim et al (2018) in the creation of Ply, a visual web inspection tool for learning the principles of Cascading Stylesheets (CSS). Akin to our 'pruning' of irrelevant process variables in the final controller diagnosis process, Ply implements pruning functionality that eliminates irrelevant CSS elements from view, in order to "minimize information overload and support novices' visual approach to sense-making" [37]. For novice users, the presence of irrelevant CSS settings in common web inspection tools (such as Chrome Developer Tools, or CDT) imposes a major barrier to learning, as it is unclear which CSS settings are in control of the display. Similarly, we can use such principles to improve the efficacy of our visualization tool, by finding a way to visually prune out irrelevant variables in the controller diagnosis process. Currently, our diagnosis process involves pruning out variables by investigating their constraints, but incorporating this feature in a more visual form can prove useful, especially to novice users of MPC systems. Another feature in Ply that can be useful in aiding controller diagnosis is embedding "expert guidance into the sense-making process to provide missing domain knowledge"[6] [37]. Providing this domain knowledge as part of the tool can yield significant benefits, especially to novice users. As shown in the controller diagnosis, process knowledge allows users to eliminate irrelevant variables effectively, helping users to understand the system in a clearer fashion.

[6] Sense-making refers to "the process of building an understanding of an artifact or example by constructing mental representations of what is known". It shares some similarities with the analysis of controller diagnosis; all representations of the diagnosis process here were all developed by following the example of control engineers in their work.

An additional consideration for future work is making the design more learner-oriented, such that it is easier to learn for novice users. Ko et al (2004) explore this concept by investigating common barriers to learning, using Microsoft Visual Basic (VB) as a case study. Their discussion is focused on the barriers presented in the programming interfaces in VB applications. While a VB programming interface is very different from an APC system, one barrier to learning seems to be especially applicable in the research presented in this thesis. The relevant barriers in their discussion are the 'information barriers', which: "are properties of an environment that make it difficult to acquire information about a program's internal behavior (i.e., a variable's value, what calls what)" [38]. It is easy to see how the information barrier applies in this visualization application; the use of static, dense sheets of raw data that is common in APC systems imposes major difficulties for diagnosis, especially for novice users. Taking these barriers into consideration can therefore result in a more user-friendly product.

5 Conclusion

The over-arching goal of the project is to investigate typical issues involved in diagnosing MPC systems; specifically, issues that relate to the interactions between operators and controllers. A task analysis is performed in order to define the specific workflow involved for controller diagnosis using Munzner's multi-level typology. The defined tasks, along with all sources of data, are compiled, reorganized, and expressed in terms of the nested model for visualization design and validation.

There are three principal tasks involved in controller diagnosis. These are combined with the three primary sources of data in order to methodically develop a visualization tools that addresses the issues faced by target users. We iterated through ideas and designs through regular consultation with industry partners.

The interactivity in the gain matrix is an especially important concept here, as it illustrates much of the novelty of this research. The sorting functionality allows users to quickly and easily filter out variables that are not used in the controller diagnosis process.

Extensive user testing is required in order to validate the feasibility of implementing this project in industry. Visualization can be a useful tool to assist control engineers, but testing this tool in a more rigorous way using various datasets and processes is necessary to validate its design.

6 Acknowledgements

I would like to gratefully acknowledge Siang Lim in the Parkland Fuel Corporation for his extensive support and guidance in developing this research. His proficiency in MPC engineering and visualization concepts was an instrumental source of inspiration. I would also like to thank the Parkland Fuel Corporation for the data and process knowledge provided, without which this research would have been impossible to complete.

I would like to also acknowledge my supervisor, Dr. Bhushan Gopaluni, for his consistent guidance and wisdom throughout this project.

Finally, I would like to acknowledge my colleagues in the Chemical and Biological Engineering Department at UBC, who helped me throughout the process of completing this research.

Bibliography

[1] M. G. Forbes, R. S. Patwardhan, H. Hamadah, and R. B. Gopaluni, “Model predictive control in industry: Challenges and opportunities,” *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 531–538, 2015, ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.09.022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405896315011039.

[2] S. Guerlain, G. Jamieson, P. Bullemer, and R. Blair, “The MPC elucidator: A case study in the design for human-automation interaction,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 32, no. 1, pp. 25–40, Jan. 2002, ISSN: 10834427. DOI: 10.1109/3468.995527. [Online]. Available: http://ieeexplore.ieee.org/document/995527/.

[3] T. Munzner, “Process and pitfalls in writing information visualization research papers,” in *Information Visualization*, A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, Eds., vol. 4950, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 134–153, ISBN: 978-3-540-70955-8 978-3-540-70956-5. DOI: 10.1007/978-3-540-70956-5_6. [Online]. Available: http://link.springer.com/10.1007/978-3-540-70956-5_6.

[4] T. Munzner, “A nested model for visualization design and validation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 921–928, Nov. 2009, ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.111.

[5] E. F. Camacho and C. Bordons, *Model Predictive control*, red. by M. J. Grimble and M. A. Johnson, ser. Advanced Textbooks in Control and Signal Processing. London: Springer London, 2007, ISBN: 978-1-85233-694-3 978-0-85729-398-5. DOI: 10.1007/978-0-85729-398-5. [Online]. Available: http://link.springer.com/10.1007/978-0-85729-398-5.

[6] N. Alsop, “When a near collinearity should and should not be made perfect,” Webinar, AspenTech.

[7] B. W. Bequette, “Model predictive control: An overview and selected applications,” Webinar, American Institute of Chemical Engineers, Sep. 30, 2009, [Online]. Available: https://www.aiche.org/academy/webinars/model-predictive-control-overview-and-selected-applications.

[8] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 3rd ed. John Wiley and sons, Dec. 2004.

[9] M. Morari and J. H. Lee, “Model predictive control: Past, present and future,” *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, May 1, 1999, ISSN: 0098-1354. DOI: 10.1016/S0098-1354(98)00301-9. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0098135498003019.

[10] D. A. Hokanson and J. G. Gerstle, “Dynamic matrix control multivariable controllers,” in *Practical Distillation Control*, W. L. Luyben, Ed., New York, NY: Springer US, 1992, pp. 248–271, ISBN: 978-1-4757-0277-4. DOI: 10.1007/978-1-4757-0277-4_12. [Online]. Available: https://doi.org/10.1007/978-1-4757-0277-4_12.

[11] R. C. Sorensen and C. R. Cutler, “Lp integrates economics into dynamic matrix control,” *Hydrocarbon Processing*, vol. 77, no. 9, pp. 57–66, Sep. 1998, ISSN: 0018-8190.

[12] N. O’Mahony, T. Murphy, K. Panduru, D. Riordan, and J. Walsh, “Improving controller performance in a powder blending process using predictive control,” in *2017 28th Irish Signals and Systems Conference (ISSC)*, 2017, pp. 1–6.

[13] K. Brooks, “Linear model predictive control – really not so bad?” South African Council for Automation and Control, Dec. 2017.

[14] S. Qin and T. A. Badgwell, “A survey of industrial model predictive control technology,” *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, Jul. 2003, ISSN: 09670661. DOI: 10.1016/S0967-0661(02)00186-7. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0967066102001867.

[15] B. Roffel and B. Betlem, *Advanced Practical Process Control*. Springer Science & Business Media, Jun. 27, 2011, 317 pp., Google-Books-ID: J1jmCAAAQBAJ, ISBN: 978-3-642-18258-7.

[16] C. M. Alaouze, “Shadow prices in linear programming problems,” New South Wales - School of Economics, 96/18, 1996, Publication Title: Papers. [Online]. Available: https://ideas.repec.org/p/fth/nesowa/96-18.html.

[17] J. Cahill. (Oct. 20, 2008). Improving gas plant throughput and robustness with MPC, Emerson Automation Experts. Library Catalog: www.emersonautomationexperts.com Section: Oil & Gas, [Online]. Available: https://www.emersonautomationexperts.com/2008/industry/oil-gas/improving_gas_p/.

[18] J. Cahill. (Dec. 15, 2008). Perspectives on big matrix and unit-level MPC applications, Emerson Automation Experts, [Online]. Available: https://www.emersonautomationexperts.com/2008/services-consulting-training/perspectives_on/.

[19] C. Lindscheid, A. Bremer, D. Haßkerl, A. Tatulea-Codrean, and S. Engell, “A test environment to evaluate the integration of operators in nonlinear model-predictive control of chemical processes,” *IFAC-PapersOnLine*, Cyber-Physical & Human-Systems CPHS 2016, vol. 49, no. 32, pp. 129–134, Jan. 1, 2016, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2016.12.202. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896316328749.

[20] T. J. Peterson, A. R. Punuru, K. F. Emigholz, R. K. Wang, and D. Barrett-Payton, “Model predictive controller solution analysis process,” U.S. Patent 7949417B2, Library Catalog: Google Patents, May 24, 2011. [Online]. Available: https://patents.google.com/patent/US7949417B2/en.

[21] T. Tao. (Oct. 7, 2015). Sweeping a matrix rotates its graph, [Online]. Available: https://terrytao.wordpress.com/2015/10/07/sweeping-a-matrix-rotates-its-graph/.

[22] D. W. Hoffman, D. L. O’Connor, and K. H. Rasmussen, “Durable model predictive control,” American Institute of Chemical Engineers, Mar. 21, 2010, [Online]. Available: https://www.aiche.org/academy/videos/conference-presentations/durable-model-predictive-control.

[23] D. J. Kozub, "CONTROLLER PERFORMANCE MONITORING AND DIAGNOSIS. INDUSTRIAL PERSPECTIVE," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 405–410, 2002, ISSN: 14746670. DOI: 10.3182/20020721-6-ES-1901.01621. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1474667015400424.

[24] L. Wilkinson and M. Friendly, "The history of the cluster heat map," *The American Statistician*, vol. 63, no. 2, pp. 179–184, May 2009, ISSN: 0003-1305, 1537-2731. DOI: 10.1198/tas.2009.0033. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1198/tas.2009.0033.

[25] P. Dragicevic. (Feb. 12, 2012). List of physical visualizations, [Online]. Available: http://dataphys.org/list/bertins-reorderable-matrices/ (visited on 04/13/2020).

[26] C. Perin, M. L. Goc, R. D. Vozzo, J.-D. Fekete, and P. Dragicevic, "DIY bertin matrix," [Online]. Available: https://aviz.fr/diyMatrix/.

[27] M. Brehmer and T. Munzner, "A multi-level typology of abstract visualization tasks," *IEEE Trans. Visual. Comput. Graphics*, vol. 19, no. 12, pp. 2376–2385, Dec. 2013, ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.124. [Online]. Available: http://ieeexplore.ieee.org/document/6634168/.

[28] C. Perin, J.-D. Fekete, and P. Dragicevic, "Jacques bertin's legacy in information visualization and the reorderable matrix," *Cartography and Geographic Information Science*, vol. 46, no. 2, pp. 176–181, Mar. 4, 2019, ISSN: 1523-0406. DOI: 10.1080/15230406.2018.1470942.

[29] J. Mackinlay, "Automating the design of graphical presentations of relational information," *ACM Trans. Graph.*, vol. 5, no. 2, pp. 110–141, Apr. 1, 1986, ISSN: 0730-0301. DOI: 10.1145/22949.22950.

[30] Y. Jansen, "Physical and tangible information visualization," PhD thesis, 2014.

[31] N. F. Fernandez, G. W. Gundersen, A. Rahman, M. L. Grimes, K. Rikova, P. Hornbeck, and A. Ma'ayan, "Clustergrammer, a web-based heatmap visualization and analysis tool for high-dimensional biological data," *Scientific Data*, vol. 4, no. 1, pp. 1–12, Oct. 10, 2017, Number: 1 Publisher: Nature Publishing Group, ISSN: 2052-4463. DOI: 10.1038/sdata.2017.151. [Online]. Available: https://www.nature.com/articles/sdata2017151.

[32] S. Smart, K. Wu, and D. A. Szafir, "Color crafting: Automating the construction of designer quality color ramps," *IEEE Transactions on Visualization and Computer Graphics*, 2020. DOI: 10.1109/TVCG.2019.2934284.

[33] M. Eisemann, G. Albuquerque, and M. A. Magnor, "Data driven color mapping," 2011. DOI: 10.2312/PE/EuroVAST/EuroVA11/005-008.

[34] M. Correll, D. Moritz, and J. Heer, "Value-suppressing uncertainty palettes," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*, Montreal QC, Canada: ACM Press, 2018, pp. 1–11, ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174216. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3173574.3174216.

[35] H. Siirtola and E. Mäkinen, "Constructing and reconstructing the reorderable matrix," *Information Visualization*, 2005. DOI: 10.1057/palgrave.ivs.9500086.

[36] T. J. Peterson, W. P. Snow, J. R. Hind, and K. R. Sheth, "Method of connecting different layers of optimization," U.S. Patent 8620705B2, Library Catalog: Google Patents, Dec. 31, 2013. [Online]. Available: https://patents.google.com/patent/US8620705B2/en.

[37] S. Lim, J. Hibschman, H. Zhang, and E. O'Rourke, "Ply: A visual web inspector for learning from professional webpages," in *The 31st Annual ACM Symposium on User Interface Software and Technology - UIST '18*, Berlin, Germany: ACM Press, 2018, pp. 991–1002, ISBN: 978-1-4503-5948-1. DOI: 10.1145/3242587.3242660. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3242587.3242660.

[38] A. J. Ko, B. A. Myers, and H. H. Aung, "Six learning barriers in end-user programming systems," p. 8, 2004.

[39] D. A. C. Beck, J. M. Carothers, V. R. Subramanian, and J. Pfaendtner, "Data science: Accelerating innovation and discovery in chemical engineering," *AIChE J.*, vol. 62, no. 5, pp. 1402–1416, May 2016, ISSN: 00011541. DOI: 10.1002/aic.15192. [Online]. Available: http://doi.wiley.com/10.1002/aic.15192.

[40] J. B. Waller and K. V. Waller, "Defining directionality: Use of directionality measures with respect to scaling," *Industrial Engineering Chemistry Research*, vol. 34, no. 4, pp. 1244–1252, Apr. 1995, ISSN: 0888-5885, 1520-5045. DOI: 10.1021/ie00043a028.

[41] S. Guerlain, "Human-automation interaction strategies," in *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society*, W. D. Gray and C. D. Schunn, Eds., 1st ed., Routledge, Apr. 24, 2019, pp. 38–38, ISBN: 978-1-315-78237-9. DOI: 10.4324/9781315782379-28.

[42] M. F. Sågfors and K. V. Waller, "Multivariable control of ill-conditioned distillation columns utilizing process knowledge," *Journal of Process Control*, vol. 8, no. 3, pp. 197–208, Jan. 1998, ISSN: 09591524. DOI: 10.1016/S0959-1524(97)00037-1.

[43] M. T. Jimoh, "A vision for MPC performance maintenance," PhD thesis, University of Glasgow, 2013. [Online]. Available: https://eleanor.lib.gla.ac.uk/record=b3001462.

[44] R. Teghtsoonian, "Range effects in psychophysical scaling and a revision of stevens' law," *The American Journal of Psychology*, vol. 86, no. 1, pp. 3–27, 1973, Publisher: University of Illinois Press, ISSN: 0002-9556. DOI: 10.2307/1421845.

[45] S. Carpendale, "Visual Representation from Semiology of Graphics by J. Bertin,"

[46] M. Ellis. (Jun. 21, 2019). RGB vs CMYK: What's the difference? 99designs, [Online]. Available: https://99designs.ca/blog/tips/correct-file-formats-rgb-and-cmyk/.

[47] M. Dubakov. (May 2012). Visual encoding, [Online]. Available: https://www.targetprocess.com/articles/visual-encoding/.

[48] X. Li, D. J. McKee, T. Horberry, and M. S. Powell, "The control room operator: The forgotten element in mineral process control," *Minerals Engineering*, vol. 24, no. 8, pp. 894–902, Jul. 1, 2011, ISSN: 0892-6875. DOI: 10.1016/j.mineng.2011.04.001.

[49] Z. Gu, R. Eils, and M. Schlesner, "Complex heatmaps reveal patterns and correlations in multidimensional genomic data," *Bioinformatics*, vol. 32, no. 18, pp. 2847–2849, Sep. 15, 2016, Publisher: Oxford Academic, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw313.

[50] C. R. Cutler and B. L. Ramaker, "Dynamic matrix control - A Computer control algorithm," presented at the 86th National American Institute of Chemical Engineers, Apr. 1979.

[51] Z. Friedman and S. Park. (May 2006). FCCU advanced control at chevron pembroke refinery, [Online]. Available: https://docplayer.net/55999379-Fccu-advanced-control-at-chevron-pembroke-refinery.html.

[52] J. Cahill. (Oct. 12, 2008). MPC Unsustainable benefits.

[53] S. Mittelstädt, J. Bernard, T. Schreck, M. Steiger, J. Kohlhammer, and D. A. Keim, "Revisiting perceptually optimized color mapping for high-dimensional data analysis," in *EuroVis*, 2014. DOI: 10.2312/eurovisshort.20141163.

[54] J. Jo, S. L'Yi, B. Lee, and J. Seo, "ProReveal: Progressive visual analytics with safeguards," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2019, Conference Name: IEEE Transactions on Visualization and Computer Graphics, ISSN: 1941-0506. DOI: 10.1109/TVCG.2019.2962404.

[55] M. King, *Process Control: A Practical Approach*. John Wiley & Sons, Dec. 13, 2010, 417 pp., ISBN: 978-0-470-97666-1.

[56] N. Instruments. (Mar. 2018). Model-based control design process (advanced signal processing toolkit or control design and simulation module), [Online]. Available: http://zone.ni.com/reference/en-XX/help/371894J-01/lvsysidconcepts/model-based_cd/.

[57] W. S. Cleveland, "Graphical methods for data presentation: Full scale breaks, dot charts, and multibased logging," *The American Statistician*, vol. 38, no. 4, pp. 270–280, 1984, Publisher: [American Statistical Association, Taylor & Francis, Ltd.], ISSN: 0003-1305. DOI: 10.2307/2683401. [Online]. Available: https://www.jstor.org/stable/2683401.

[58] S. El Ferik, "16 - automation technology in hydrocarbon fuel processing plant," in *Advances in Clean Hydrocarbon Fuel Processing*, ser. Woodhead Publishing Series in Energy, M. R. Khan, Ed., Woodhead Publishing, Jan. 1, 2011, pp. 463–495, ISBN: 978-1-84569-727-3. DOI: 10.1533/9780857093783.5.463. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781845697273500160.