Advances in Industrial Control

Pablo Cano Marchal
Juan Gómez Ortega
Javier Gámez García

Production Planning, Modeling and Control of Food Industry Processes

AIC
Springer

Advances in Industrial Control

Series editors

Michael J. Grimble, Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, UK
Antonella Ferrara, Department of Electrical, Computer and Biomedical Engineering, University of Pavia, Pavia, Italy

Advances in Industrial Control is a series of monographs and contributed titles focusing on the applications of advanced and novel control methods within applied settings. This series has worldwide distribution to engineers, researchers and libraries.

The series promotes the exchange of information between academia and industry, to which end the books all demonstrate some theoretical aspect of an advanced or new control method and show how it can be applied either in a pilot plant or in some real industrial situation. The books are distinguished by the combination of the type of theory used and the type of application exemplified. Note that "industrial" here has a very broad interpretation; it applies not merely to the processes employed in industrial plants but to systems such as avionics and automotive brakes and drivetrain. This series complements the theoretical and more mathematical approach of Communications and Control Engineering.

Indexed by SCOPUS and Engineering Index.

Publishing Ethics: Researchers should conduct their research from research proposal to publication in line with best practices and codes of conduct of relevant professional bodies and/or national and international regulatory bodies. For more details on individual ethics matters please see:
https://www.springer.com/gp/authors-editors/journal-author/journal-author-helpdesk/publishing-ethics/14214

More information about this series at http://www.springer.com/series/1412

Pablo Cano Marchal · Juan Gómez Ortega
Javier Gámez García

Production Planning, Modeling and Control of Food Industry Processes

Springer

Pablo Cano Marchal
Departamento de Ingeniería
Electrónica y Automática
Universidad de Jaén
Jaén, Spain

Javier Gámez García
Departamento de Ingeniería
Electrónica y Automática
Universidad de Jaén
Jaén, Spain

Juan Gómez Ortega
Departamento de Ingeniería
Electrónica y Automática
Universidad de Jaén
Jaén, Spain

ISSN 1430-9491 ISSN 2193-1577 (electronic)
Advances in Industrial Control
ISBN 978-3-030-01372-1 ISBN 978-3-030-01373-8 (eBook)
https://doi.org/10.1007/978-3-030-01373-8

Library of Congress Control Number: 2018959258

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To our families.

Series Editors' Foreword

The series *Advances in Industrial Control* aims to report and encourage technology transfer in control engineering. The rapid development of control technology has an impact on all areas of the control discipline. New theory, new controllers, actuators, sensors, new industrial processes, computer methods, new applications, new philosophies…, new challenges. Much of this development work resides in industrial reports, feasibility study papers, and the reports of advanced collaborative projects. The series offers an opportunity for researchers to present an extended exposition of such new work in all aspects of industrial control for wider and rapid dissemination.

Perhaps unsurprisingly, the classification and structure of control in the food industry are remarkably similar to that of process control in the broad spectrum of industrial and manufacturing processes. Industrial processes can be divided into "primary" and "secondary" industries. Primary industrial processes convert raw materials into materials ready for further processing or transformation. An example is mineral processing where iron, copper, and aluminum ores are converted into steel, copper, and aluminum, respectively. Even a raw material such as the water held in lakes, reservoirs, or the sea has to be processed into drinking water.

Similarly in the food industries, raw materials such as sugar cane, wheat grains, and olives need to be converted to sugar concentrate, flour, and olive oil, respectively, all prepared for further processing. Such first-stage processes form a class of "primary" food transformation industries. As with the primary industrial processes, control adaptation to the variable quality of raw materials is one of the key issues. Another key issue for these large-scale food processing plants is maintaining consistent performance in the face of equipment wear and tear and component failure. There is also an element of batch control as different batches of the raw foodstuff input may have different physical properties. Some areas of food industry processes are extremely labor-intensive with much lower levels of automation than is found in inanimate material processing. Processes that deal with animals are subject to a whole range of animal welfare regulations, and the scope for automation is much reduced. All these factors and concerns make the Production Planning and control of primary food industries very complex.

In "secondary" food industries, the refined outcomes of the primary transformation industries enter a more straightforward field of food product manufacture. For example, sugar concentrate, flour and olive oil, become the input ingredients to say, the manufacture of confectionery, biscuits and bread, and sauces. Even here the generic features of control are in evidence. Plant-wide planning and scheduling, often with batch processing, forms a top level for the control hierarchy, while the loop control of food-baking ovens and highly automated food production lines prevails at the lower levels of the hierarchy. A typical Production Planning constraint is that in the manufacture of biscuits where the lighter-colored biscuits are made first, followed by the darker-colored biscuits, with the darkest colored biscuits being made last before the production line is cleaned thoroughly and the sequence begins all over again.

What is a little surprising to the Series Editors is how infrequently they have had the opportunity to publish a control monograph or attend a control conference presentation for these very important and extensive food industry control application areas. Thus, this monograph *Production Planning, Modeling and Control of Food Industry Processes* by authors Pablo Cano Marchal, Juan Gómez Ortega, and Javier Gámez García is a very welcome addition to the *Advances in Industrial Control* series. The monograph opens with an overview and survey of the global importance of food industries. The following three chapters examine model identification and then the control needed in the top and lower levels of the processing control hierarchy. Chapter 5 focuses on the particular issues for Production Planning in the food industry, while Chap. 6 presents a case study of these control concepts and designs applied to olive oil production. The monograph is well focused, and the *Advances in Industrial Control* Series Editors hope that there will be more such food-industry-related monographs in the series in the future.

Glasgow, Scotland, UK

Michael J. Grimble
M. A. Johnson
Industrial Control Centre

Acknowledgements

The authors would like to thank the members of the Robotics, Automation and Computer Vision Group of the University of Jaén for their support and the excellent working environment that they provide.

We would also like to thank the funding institutions that have helped to carry out the different research projects in the last few years, and in particular, the Ministry of Economy and Industry of Spain for the funding of the project DPI2016-78290-R. We make extensive our gratitude to the different partners that have collaborated with us in these projects.

Finally, we would like to thank our families for their patience during the time of the writing of this book. P. Cano Marchal would like to specially thank his wife Beatriz for her support, love, and understanding.

Contents

Chapter 1
Introduction

The food and drink industry is a major sector in the world economy. The worldwide turnover of the industry in 2012 amounted more than 2,700 billion (Europe 2014), which represents a 2.7% of world GDP. The main actors in the industry are the European Union, USA, and China, roughly accounting for 69% of the total, being followed by Japan, Brazil, Mexico, Russia, and Australia (Europe 2014).

The objective of this chapter is twofold: First, it serves as an introduction to the application of automatic control to food processes, by briefly exposing the typical characteristics of food processing operations. Second, it introduces the framework that unites the different chapters included in the book.

The following section briefly exposes the characteristics of food processing operations, with some emphasis on those that characterize these processes as somewhat particular for the application of feedback techniques. In turn, Sect. 1.2 deals with the convenience of the consideration of hierarchical control schemes to handle the implications posed by these characteristics, along with the role of each of these layers in the global process control. Finally, Sect. 1.3 presents an example of food transformation processes and provides a real context for the ideas introduced in this chapter.

1.1 Characteristics of Food Processing Industry

Food industry is quite broad, as it encompasses notably diverse raw commodities and final products, which consequently entails different transformation processes. Food processing involves transforming a set of raw inputs coming from agriculture, stockbreeding, or fishing into finished food products, and the degree of transformation of the products may range from a selection and packaging of the raw materials to sophisticated processes involving different physical or chemical treatments of the inputs.

P. Cano Marchal et al., *Production Planning, Modeling and Control of Food Industry Processes*, Advances in Industrial Control,
https://doi.org/10.1007/978-3-030-01373-8_1

Table 1.1 Divisions (two-digit codes) and classes (four-digit codes) related to food industry according to the International Standard Industrial Classification of All Economic Activities (ISIC) (UN 2008)

10-Manufacture of food products
1010-Processing and preserving of meat
1020-Processing and preserving of fish, crustaceans, and molluscs
1030-Processing and preserving of fruit and vegetables
1040-Manufacture of vegetable and animal oils and fats
1050-Manufacture of dairy products
1061-Manufacture of grain mill products
1062-Manufacture of starches and starch products
1071-Manufacture of bakery products
1072-Manufacture of sugar
1073-Manufacture of cocoa, chocolate, and sugar confectionery
1074-Manufacture of macaroni, noodles, couscous, and similar farinaceous products
1075-Manufacture of prepared meals and dishes
1079-Manufacture of other food products n.e.c.
1080-Manufacture of prepared animal feeds
11-Manufacture of beverages
1101-Distilling, rectifying, and blending of spirits
1102-Manufacture of wines
1103-Manufacture of malt liquors and malt
1104-Manufacture of soft drinks; production of mineral waters and other bottled waters

The diversity of food transformation industry can be properly appreciated examining the classification of economic activities that different governmental and public institutions publish, such as North America's NAICS (ECPC 2012), European Union's NACE (EU 2008), or United Nation's ISIC classification (UN 2008). These classifications primarily attend to the nature of the raw commodities or the final products, which grants their utility to visualize the wide range of activities included in the industry.

Table 1.1 presents the groups and classes related to food processing industry according to the ISIC classification. Packaging of fresh fruits (1030), pickling of vegetables(1030), production of olive oil (1040), manufacturing of flour (1061), refining of sugar (1072), baking of cookies (1071), manufacturing of oven-ready pizza (1075), or brewing of beer (1103) are examples of activities that illustrate the aforementioned diversity of the industry.

On the other hand, food itself can also be classified, and it is of interest to consider its classification according to the degree of transformation or processing it has been subject to. The World Health Organization (WHO) and the Food and Agriculture Organization of the United Nations (FAO) classify food commodities and animal feedstuffs into five classes, distinguishing between primary food commodities—classes

Table 1.2 Proposed food classification systems according to the degree of processing of the food, (adapted and reprinted from CODEX Alimentarius: List of standards, World Health Organization and FAO, ©1993 World Health Organization and Food and Agriculture Organization of the United Nations, reproduced with permission, and also reprinted with permission from Springer, Current Obesity Reports, Food Classification Systems Based on Food Processing: Significance and Implications for Policies and Actions: A Systematic Literature Review and Assessment, Moubarac et al., ©2014)

FAO-WHO

A. Primary Food and Commodities of Plant Origin
B. Primary Food and Commodites of Animal Origin
C. Primary Feed Commodities
D. Processed Food of Plant Origin
E. Processed Food of Animal Origin

Europe (IARC-EPIC)

1. Non-processed
2. Modestly or moderately processed
   - 2.1 Industrial and commercial foods involving relatively modest processing and consumed with no further cooking
   - 2.2 Foods processed at home and prepared/cooked from raw foods or moderately processed foods
3. Processed
   - 3.1 Processed staple/basic
   - 3.2 Highly processed

US (IFIC-Joint Task Force)

1. Minimally processed
2. Food processed for preservation
3. Mixtures of combined ingredients
4. Ready-to-eat processed

Global (NOVA)

1. Unprocessed and minimally processed foods
2. Processed culinary ingredients
3. Ready-to-consume products
   - 3.1 Processed food products
   - 3.2 Ultra-processed products

A, B and C—and processed food—classes D and E—(World Heath Organization and FAO 1993). Recent efforts have been carried out to develop new classification systems that discriminate additional degrees of food processing, mostly motivated by its application in the classification of diets and the study of their healthiness (Moubarac et al. 2014). Table 1.2 includes several of these recently proposed classification systems, together with the WHO/FAO one.

Following the guidelines suggested by these classifications, for our purposes it is convenient to consider the following three classes of food:

1. Unprocessed commodities: These are the raw basic goods as delivered by the primary activity responsible for their production. Examples of this class include beet, sugar cane, grain, raw meat, raw fish.
2. Moderately processed products: This category includes basic transformed goods that may be final products themselves or may constitute inputs of other food processing operations. Olive oil, refined sugar, flour, or quality-selected vegetables are examples of this class.
3. Processed products: This category comprises complex consumer-ready products that require several ingredients from the former classes to be produced. Products such as cookies, canned food, oven-ready pizza, or beer are included in this group.

Now, using the categories defined above, we can roughly classify food processes attending to their class of inputs and outputs:

1. Primary food transformations: These are the processes that take raw commodities as inputs and deliver moderately processed products. Examples of this class are the production of sugar from beet and the production of olive oil.
2. Secondary food transformations: These processes transform moderately processed products into processed ones. The production of cookies from flour and sugar and the production of pizza from flour, cheese, and toppings are activities included in this category.

The wide scope of the industry naturally leads to processes that show different particularities. However, the following distinctive characteristics are often found in this industry:

- The variability of the properties of the inputs is usually high and constitutes a major source of process disturbances that affect the process outputs of interest, such as the quality of the final product or some measure of process yield.
- Food products are usually complex substances, whose properties of interest are difficult to measure online.
- The processes are frequently multivariable, with couplings between different process variables, and disturbances that are susceptible of being propagated through the process.

Figure 1.1 shows a Venn diagram representing these features, emphasizing that, depending on the particular food transformation process considered, some or all of them can be encountered.

These three features pose specific challenges that must be addressed for the automatic control of this type of processes, thus requiring somewhat particular techniques to deal with them. The following sections further elaborate on each of these characteristics.

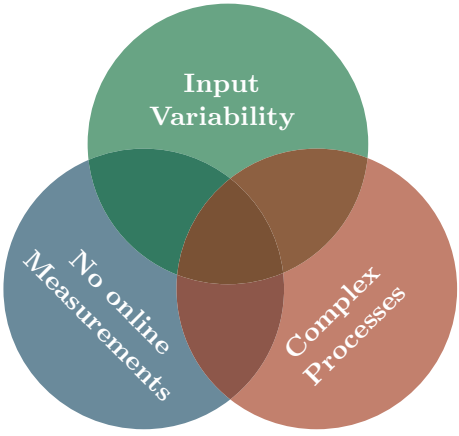


Fig. 1.1 Venn diagram representing the three most relevant characteristics usually found in food processing processes, showing that not all processes necessarily exhibit all the characteristics

1.1.1 Variability of Raw Inputs in Food Industry

As introduced above, the purpose of food processing operations is the transformation of natural products coming from agriculture, stockbreeding, or fishing into final products apt for their consumption.

Consumers require for the quality of the products they consume to be constant, particularly if the product is marketed associated with a brand. This quality of the final products is often dramatically influenced by the features of the inputs employed in the production process. Thus, one of the main objectives of food processing operations is to *damp* the variability of the inputs, so that consistent quality products are finally obtained.

This feature is particularly relevant for primary transformation processes, since they employ unprocessed commodities whose variability is usually high, particularly for agricultural goods (Perrot et al. 2006). On the other hand, secondary transformation processes employ moderately processed products as inputs, which show much lower variability, since they are themselves already the output of a transformation process where delivering consistent output quality is one of its objectives.

The variability in stockbreeding products is caused both by preslaughter and postslaughter factors. The breed of the animals, the feed commodities and the handling conditions supplied during their breeding, the gender and age, and the biochemical dynamics of the of the postmortem period are important factors that influence the quality (Nollet and Toldra 2006). For fish, factors are similar: The type of species, the physiological condition of the fish, the temperature and salinity of the water, killing procedures, and the handling during the postmortem period greatly affect the quality of the product (Rehbein and Oehlenschlager 2009).

The causes for the variability of agricultural raw inputs are diverse. First, it is not strange that a food processing factory deals with raw inputs coming from different varieties of the same species, with the corresponding different typical characteristics of the product. It is sometimes the case that the different varieties are received independently, with the grower stating the type of product, particularly for the fresh fruit and vegetables packaging industry. However, for some other industries, it is not uncommon to receive different varieties mixed, with the corresponding challenge of determining the variety each element belongs to. Even when the different varieties are determined beforehand, the acceptable standards for each feature may be different for each variety, or even the relevant discriminating features may not be same for all the varieties.

Another source of variability is the fact that most food industries receive raw products from different growers, which may follow different agronomic practices and may have their farmlands in locations with slightly different soil, topography, or climate conditions, which provoke differences in the properties of the incoming raw products (Forbes and Watson 1992).

In turn, the evolution of the properties of the products caused by their ripening along the harvesting season introduces yet another source of variability. On the one hand, the mentioned different conditions for different farmlands may provoke the

reception of goods in different ripening stages. On the other hand, meteorological variations from season to season may induce variations on the evolution of the ripeness index of the fruits, inducing changes in the production scheduling.

Finally, the reception of potentially damaged goods due to plagues or adverse meteorological phenomena, such as hail, is another source of variability for industries that receive the raw products directly as they are harvested, with no previous selection carried out by the growers.

The possible actions to reduce the effect of this input variability on the final product depend on the particular process at hand. Traditionally, two main approaches are widespread:

- Classification of the incoming raw products according to their characteristics, and separate processing of each of these categories. This approach is adopted in processes where the input features influence very heavily the output characteristic.
- Adjustment of the values of other process variables to counteract the effect of the disturbances induced by the inputs. This approach is applicable in multivariable processes where the input features are not the only variables affecting the output characteristics, and their effect is mild enough to be compensated by the effect of other process variables.

1.1.2 Difficulty of Online Measurement of Food Properties

A common feature of unprocessed commodities, moderately processed, and processed food is that obtaining fast accurate measurements of their properties of interest is usually not an easy task. Well-established laboratory methods of reference are usually available to determine with sufficient accuracy the values of these properties—otherwise that property could not be considered at all! However, being able to robustly and accurately obtain values of these properties online is usually a quite challenging problem (Huang et al. 2008).

Considered as a matrix for laboratory analysis, food is usually a complex one and requires plenty of previous operations to prepare the sample before it is actually analyzed with the corresponding equipment. Moreover, typically, the time required to perform the analysis is significantly higher than the desirable sampling time for the online monitoring of the process. The need for these previous operations and the time required to perform the analysis make it difficult to adapt the reference laboratory methods for the online measurement of the products. Thus, different analytic methods, usually based on indirect measurements, must be considered for the online analysis.

Food, be it the finished ready-to-market product or the raw inputs, is often a complex mixture of different components. The already mentioned natural variability of the product, together with the complexity of its composition, renders it sometimes difficult to develop accurate indirect measurement methods, due to the interaction and composition variation of the different components, yielding methods that might

be quite population-specific and require frequent addition of new samples to the calibration, along with a tight supervision of their performance.

These difficulties hold for the development of both indirect at-line and online sensors. The adaption of the at-line measuring principles and equipment to online devices provides challenges of its own. Practical implementation considerations, such as the possible accumulation of substance in the sensor, or the ratio between the required time for the analysis and the flow velocities of the substances subject to analysis, impose further constraints that must be considered to successfully build or use an online sensor.

Moreover, food properties such as aroma or flavor are the macroscopic result of the composition of many chemical compounds that are present at quite low concentrations. These quality characteristics are very difficult to assess using instrumental methods and are usually determined by expert tasters. Assessing these properties off-line is usually a difficult task, even without the constraints imposed by the requirements of online measuremente, so performing these analysis online is a major challenge.

Despite the difficulty of the task, considerable research effort has been devoted to building and improving sensors and application methods capable of providing at-line and online measurements of key variables of food processing operations, and many successful applications have been reported (Sun 2004; Huang et al. 2008; Woodcock et al. 2008; Wu and Sun 2013; Loutfi et al. 2015). Among the different technologies of sensors applied, three may be highlighted for the extensive research and success in commercial application: machine vision, infrared spectroscopy, and *electronic noses*.

Machine vision has been used in applications such as monitoring of both unprocessed commodities, such as grain (Majumdar and Jayas 2000), rice (Wan et al. 2000), or potatoes (Tao et al. 1995), and processed food, such as corn tortillas (Mery et al. 2010) and pizzas (Sun 2000). The review papers, Davies (2009), Wu and Sun (2013), Zhang et al. (2014), and Śliwińska et al. (2014), provide further examples of applications.

In turn, infrared spectroscopy is probably one of the best known technologies for the construction of indirect sensors for the food industry. This technology has been employed in the measurement of meat properties (Savenije et al. 2006; Prieto et al. 2006), grain (Welle et al. 2003), dairy products (Kawasaki et al. 2008), or olive oil (Jiménez et al. 2005). Additional examples can be found in the book Sun (2008) and review papers Huang et al. (2008) and Śliwińska et al. (2014).

Electronic noses are an emerging technology, and its routine use is not really spread in food industry yet (Loutfi et al. 2015). However, the reports of applications at the research level have been increasing over the last years, and the time for its adoption in industry is closer. Examples of its applications include the detection of rancidity of milk (Capone et al. 2001) and olive oil (Aparicio et al. 2000), estimating the shelf life of meat (Amari et al. 2009) or discriminating the quality of black tea (Tudu et al. 2009a, b). Again, further examples are included in the review papers Loutfi et al. (2015) and Śliwińska et al. (2014).

Despite the advances in the application of these technologies, many of the reported applications are for at-line analysis. At-line devices improve the sampling rate and

delay of the analysis results over the external laboratory alternative, but the sampling rate is usually limited by the need of an operator to sample the process and perform the analysis in the at-line device and usually still constitutes a limiting factor for the use of these data in a feedback control structure.

Besides, plenty of food processes do not enjoy the possibility of performing online or at-line analysis of the food products involved in the process, either because no such analyzers are available in the market or because their price is prohibitively high for the company carrying out the operations. In these circumstances, it is often the expert operators that play the role of at-line sensor (Allais et al. 2007), assessing the quality of the final products based on their experience, again, at a sampling rate that constitutes a handicap for control applications (Allais et al. 2007; Caldwell 2012).

1.1.3 Complex Nature of Food Processes

As observed in the introduction, the degree of transformation of raw inputs in food processes greatly varies from one industry to another. Consequently, so could be expected from the complexity of the involved processes.

However, beyond those cases in primary processes where no transformation of the inputs is intended at all, such as the packaging of fresh products, the processes that drive the transformation of the inputs into the final products are usually complex, with many biochemical reactions and matter transference between phases taking place, even in primary processes.

Fermentation and cooking processes are clear examples where these types of reactions take place. Still, some other processes where, at first sight, only physical interactions occur also contain this type of reactions that greatly influence the quality characteristics of the final products (Birle et al. 2013).

This intrinsic complexity often involves that the processes are not fully understood, and even when the process is fully characterized, the values of the intervening variables may not be easy to measure, making them difficult to be controlled.

The second source of complexity for food processing operations is their multivariable nature. Typically, several conflicting process objectives are considered in the activity of the plant, with process variables causing opposite effects on them, which leads to intertwined process dynamics.

At it mildest expression, a careful design in the pairing between controlled and manipulated variables must be considered, if the system is to be controlled as a series of isolated single variable control loops. In cases where the coupling between variables is severer, full multivariable control schemes might be required to achieve acceptable performance of the plant.

Another implication of this multivariable nature of the processes is that, as is the case with process industry, in food industry the operating set-points are usually chosen as the result of a trade-off between different effects of the process variables in the product properties, technological constraints that limit the values of some variables and cost considerations. Consequently, the economic facet of the problem

highly encourages the selection of good process set-points, along with a good control capable of maintaining the process variables as close as possible to these specified set-points.

1.2 Characteristics of the Automatic Control of Food Processing Industry

The features introduced in the previous section confer food processing operation's particular characteristics from the automatic control perspective. This section introduces a conceptual model of food transformation processes and discusses a hierarchical approach for its automatic control.

1.2.1 Types of Variables and Process Dynamic Layers

In food industry processes, the relevant involved variables can usually be classified into five categories, according to their role in the process:

- Process output variables (y_h): These are the variables whose value conforms to the final objective of the process. They can be quality characteristics of the final product, or properties of the process, such as transformation or recovery yield.
- Intermediate process variables (y_l): These are the variables that are not directly manipulable, but whose values influence the process output variables.
- Manipulable variables (u): These are the variables whose value can be set by the process operators.
- Raw product properties (f): These are the properties of the input products and are considered disturbances to the process, since their values cannot be changed.
- Fast process disturbances (d): These are the disturbances that affect the intermediate process variables (y_l), and whose effect on the process output variables (y_h) is considered to be exerted exclusively via these variables.

  The relations between these variables can be grouped into two layers:

  – Lower-level dynamics (G_l): These are the process relations that associate manipulable variables (u) with the intermediate process variables (y_l), subject to the action of the fast process disturbances (d).
  – Higher-level dynamics (G_h): These relations link the raw input properties (f) and the intermediate process variables (y_l) with the final process outputs (y_h).

Figure 1.2 depicts a block diagram showing the defined relations between the different types of variables considered.

Two reasons motivate such a classification of the process variables and dynamics: the diverse generality of the models for the low- and high-level dynamics, and the different ease of measurement of the output variables of each layer.

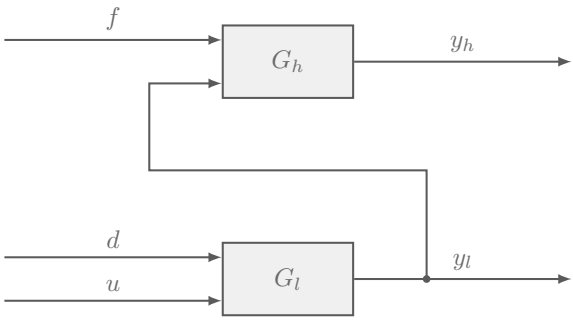


Fig. 1.2 Block diagram showing the different groups of variables considered in a food transformation process and the two layers of dynamics that relate them

Intuitively, the lower-level dynamics can be considered as the implementation details of the process, while the higher-level layer regards recipe-like relations, i.e., relations among variables that are true in general for the process at hand and essentially independent of the particularities of the concrete implementation of the process.

The lower-level dynamics depend heavily on the particular characteristics of the equipment used to carry out the process, and thus, models constructed for this layer are only valid for a specific processing plant.

On the other hand, the higher-level layer regards the core relations that, being specific to the process at hand, hold true for every instance of the process, irrespective of the particular equipment employed. Thus, the models for this layer have a broader range of validity. Indeed, it is this knowledge of this layer that allows an expert operator to operate a plant with different low-level characteristics.

The often different ease of measurement of y_l and y_h is explained by the magnitudes that are typically included in each group. Intermediate process variables (y_l) are usually variables such as temperatures, tank levels, or flows, for which standard online sensors are available. For these variables, the technically achievable minimal sampling time is usually small compared to the characteristic time constant of the process and does not constitute a limitation. Besides, the cost of obtaining a new datum is usually irrelevant, once the required equipment is available.

In turn, process output variables and raw product properties, which are relevant to the higher-level layer, are usually food characteristics much harder to measure. As discussed in the previous section, its online measurement is usually quite challenging, and the products are often analyzed either at-line or in a laboratory. This lack of online measures naturally leads to having long sampling times for this type of variables and a significant cost per obtained datum.

1.2.2 Convenience of Hierarchical Control

The consideration of two layers of process dynamics, together with the different sampling times of each type of variable, naturally introduces the convenience of considering a hierarchical control structure for the global control of food processes.

The availability of fast online measurements of y_l allows to devise control loops with small sampling times that employ u as manipulated variables, to assure that the elements of y_l remain close to their specified set-points r_l, despite the effect of d.

However, the output variable of real interest of the process is y_h, which includes the variables that characterize the quality of the produced product and other economically relevant magnitudes. Essentially, the interest on y_l is just due to its influence on y_h.

If only the feedback control loops that consider y_l as their output variables were to be considered in the system, y_h would be under open-loop control. Thus, disturbances provoked by the input product characteristics (f) and errors in the Modeling of G_h that may lead to selecting values of r_l that do not yield the desired y_h would not be compensated. Therefore, although the difficulty in measuring y_h and f restricts the viable approaches, the inclusion of a higher-level control loop, explicitly accounting y_h as its controlled variable, is advisable.

However, none of the two variables affecting y_h through G_h are directly manipulable. The characteristics of the raw products (f) are considered disturbances, since their values are considered to be given and cannot be chosen, while y_l effectively depends on the values of u and d and, thus, cannot be fixed arbitrarily.

Therefore, the inclusion of the lower-level controller on y_l provides the basis to assume that the reference values provided to this controller (r_l) will eventually be the values of y_l despite the effect of d. Thus, the upper-level control layer can employ r_l as its manipulated variables, effectively constituting a master controller over the lower-level feedback controllers on y_l.

Given the fundamental influence of f on y_h, the inclusion of feedforward action based on the values of measurable components of f is usually appropriate. In cases where the disturbances induced by f can be counteracted by the effect of other variables involved in the process, this feedforward action can be exerted via variations of r_l, complementing the action of the high-level feedback controller. In situations where the dependence of y_l on f is such that the induced disturbances practically impede reaching the specified r_h, a feedforward action updating r_h to a feasible objective is usually useful. As discussed in the previous section, the production objective of the process is often chosen as the result of a trade-off between competing effects; if the previously selected objective is not achievable, it is of interest to re-evaluate the decision and consider a new objective, optimal under the current conditions.

The fundamental dependence of y_h on f entails that some values of y_h might not be reached if the values of f are not adequate. If we assume that the target values of y_h are given by business-related considerations, we are implicitly setting a constraint on the acceptable values of f, since those values that do not allow to reach y_h should be rejected.

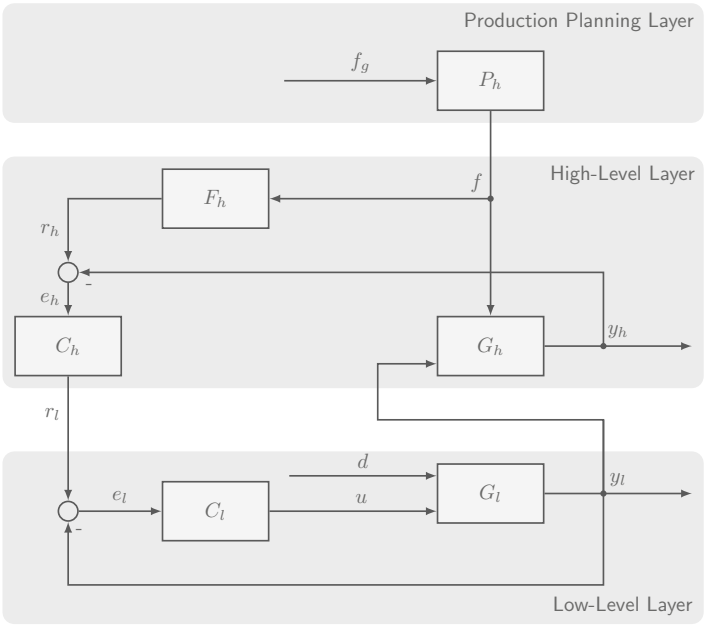


Fig. 1.3 Block diagram showing the proposed hierarchical control structure. Here, C_l represents the low-level controller, C_h the high-level controller, P_h the Production Planning algorithm, and f_g the value of the raw product properties in the grove

Since typically f varies along the season, and so does the cost of inputs possessing different values of f, a relevant question to address is when to produce so that the value and cost of f are optimal for the given production objective. To answer this question, aspects such as restrictions on the processing capacity of the facility and on the availability and price of raw inputs with different values of f should be considered. This problem effectively constitutes a Production Planning optimization for the whole season and comprises a third layer on the global control of the process.

The control structure discussed is depicted in Fig. 1.3, showing the separate dynamic blocks and the three control layers. The next section discusses in more detail the characteristics of each of these layers.

1.2.3 Characteristics of Each Controller Layer

Each of the three control layers devised in the global control diagram has a different objective. The low-level control layer intends to maintain the values of the low-level dynamic layer output variables (y_l) close to some reference values (r_l) that, as far as this control layer is concerned, are already given.

In turn, the high-level control layer is responsible for providing appropriate values of r_l to the underlying layer, so that the high-level outputs of the process (y_h) effectively attain the established objectives r_h, or updating these r_h in case that they are not attainable, given the current available product characteristics (f).

Finally, the Production Planning layer deals with selecting when to produce each product, attending to the possibly varying characteristics of the input products during the season, and their cost. This layer, if market-related considerations are also included, may also define the specific product to be produced in each considered time period.

The diversity in the objectives of each of the layers is reflected on the different aspects of the production process that are of interest for each of the controllers. The low-level control layer deals with the relations between the manipulable variables, low-level disturbances, and intermediate process variables y_l. The high-level control layer is focused on the relations between r_l, f, and y_h, while the planning layer attends to how the evolution along the season of f may affect the production, mainly from an economic point of view. The major characteristics of each of the three layers are further discussed in the following subsections.

1.2.3.1 Low-Level Control Layer

The main concern of this layer is to improve the dynamic response of the plant and reject the relatively fast disturbances that may affect y_l, so that it effectively reaches r_l. In this layer, the dynamics of the involved systems are important, and the characteristics of this lower layer of the process provide a fairly standard framework for feedback control.

Since this layer is concerned with fairly low-level relations of the process, most of the time simple SISO control schemes should be enough to achieve the required performance and fulfill the objective of this control layer. In cases where the multivariable nature of the process is apparent at this layer, basic MIMO techniques could be applied to improve the performance. However, the focus is on dealing with the different parts of the process as independently from each other as possible, as the multivariable nature of the process is dealt with in the high-level control layer.

The availability of online sensors for the output variables of interest (y_l) allows to include traditional system identification techniques as useful tools for Modeling the different involved systems, as a previous step to the design of the controllers.

The kind of challenges presented by this control layer is completely analogous with those faced when dealing with the low-level control layer of process industry: existence of process delays, impact of disturbances, interaction between control loops, etc.

1.2.3.2 High-Level Control Layer

The objective of this control layer is to assure that y_h effectively reaches the predefined values r_h. The fulfillment of the objective requires that y_h is measured, so that the deviations from the prescribed values are detected, and corrective actions are proposed to counteract these deviations.

Even though the two actions mentioned above are common to every feedback control scheme, they are explicitly stated since they represent the core challenges found in this layer.

As already commented, the measurement of food properties is not an easy task, and this constitutes the most representative characteristic of this layer. The sampling rate at which the involved variables can be measured fundamentally conditions the properties of the feedback loop to be implemented. For instance, an implication is the inability to reject disturbances whose bandwidth is higher than the available sampling rate.

The second requirement, that is, the computation of corrective actions based on the measures of the output variables, is hindered by two aspects: the already mentioned often inherent complexity of the relations between the variables of this layer and the additional barrier that the difficulty of measuring the involved variables represent. If the measurement of the involved variables is complex and expensive, the experiments required to perform an identification of the system of interest can be expected to be, consequently, also complex and expensive.

Moreover, the fundamental influence of f on y_h, and the fact that they are disturbances, that is, variables whose value cannot be chosen or modified, represents an additional difficulty for the understanding of the system relations, since the identification experiments must be carried out with the available values of f. It is regularly the case that a particular set of values of f is only available during a very short time period of the season, introducing further practical constraints to the task. Moreover, if such is the situation, raw inputs exhibiting these values are usually appreciated, and therefore, expensive, increasing the cost of the experiments.

In cases where the above difficulties are sufficiently mild, system identification experiments can be performed, and models of the system can be constructed. However, it is not rare that carrying out such experiments is unfeasible, and some other approaches must be sought to gather the required knowledge of the system.

The first concession is usually to settle with static models, as opposed to dynamic models. That is, experiments to model the steady-state relations between the process variables can be planned following factorial design procedures or other approaches.

Another common approach is to aim at obtaining useful information of the system without performing experiments, resorting to the application of fuzzy techniques to gather knowledge from expert process operators. This approach may intend to directly elicit feedback control rules used by the expert to operate the system or to obtain a fuzzy model of the system, which, in turn, would be used to design and implement a model-based controller relying on this model.

The use of static models for this control layer is usually acceptable due to two factors: The variations of f typically represent low-frequency disturbances, and the

available sampling frequency of the involved variables found in practice is not fast either, so the use of static models is usually good enough.

Usually, the key added value of this controller is not the improvement of the dynamic response of the plant, but rather the explicit consideration of the multivariable nature of the system, and the capacity to modify the values of the process set-points (r_l) so that the output variables eventually reach the desired values, while considering the process-specific trade-offs.

Thus, the type of controllers of interest for this layer should take into account the multivariable nature of the system and compute the control action optimizing the trade-offs of the process. Hence, Model Predictive Control is the perfect candidate for the layer, as it allows to include multivariable models and take into account constraints, and the control action is computed as the solution of an optimization problem whose objective function can be designed to consider the trade-offs of the process. Moreover, since the sampling time for this layer is usually quite high, requiring a relatively long time to solve the optimization problem is usually not a problem. A special feature of this layer is that, as discussed above, the models employed by the controller are usually not traditional dynamic first-order plus dead-time ones, but rather static or fuzzy ones. However, the fundamental idea of measuring the current output, computing a sequence of control actions for the considered control horizon, and applying just the first of those actions applies.

1.2.3.3 Production Planning Layer

The objective of this upper layer is to take into account that, although the properties of the raw inputs are considered fixed once that they arrive at the production facilities, the evolution through the season of the properties of the natural raw products usually means that, depending on the current stage of the season, certain values of the properties of these raw inputs, or certain cost of the inputs of a specific quality, are expected.

The general idea is to decide when to produce so that the properties and cost of the raw inputs are optimal for the final product intended to be produced. Thus, if models of the evolution of these characteristics of the raw products are available, together with the models that relate f with y_h—which are often already available from the previous control layer—a planning problem can be set up to decide what amounts of which product quality to produce at each considered time period. If the available models that relate f and y_h are dynamic, they can be simplified to take into account just the steady-state relations of the variables, since the dynamics of the relations offer no extra insight or information to the problem, as the utility of the model is just to provide a mapping from raw product properties (f) to final product characteristics (y_h).

The interest of the problem is greater in cases where different properties of the raw products evolve in opposite directions; i.e., some properties may increase their fit for the process production objective, while others decrease theirs; and if different final

products can be produced with different requirements on the raw inputs properties for each of them.

Market- and business-related aspects of the company carrying out the operations are also of interest for the problem, since market prices and sales previsions for each product quality are required to contemplate which product is more profitable, along with the production limits for each of them.

Finally, capacity constraints due to the production facilities and supply limits, related to a limited amount of available raw product for each time period, are also required to pose the optimization problem.

Thus, the output of this layer is the solution of an optimization problem whose objective is to maximize the profit of the whole season, and provides a production plan for the season, that is, a sequence of the amounts of each final product to produce at each time period.

1.3 Food Processing Examples

This section presents an example of a food transformation process and comments its relevant features from the Modeling, control, and Production Planning perspective in order to provide some real context for the ideas presented in the previous sections. The Virgin Olive Oil production process is further treated in the case study presented in Chap. 6.

1.3.1 Virgin Olive Oil Production Process

The Virgin Olive Oil production process (VOOPP) is an industrial process whose objective is the extraction of the oil out of the olives using mechanical means exclusively. The process is composed of two main stages: the paste preparation and the effective oil separation. The paste preparation encompasses the crushing of the olives to form the so-called olive paste, and the heating and stirring of the paste in the thermomixer. The oil separation is carried out feeding the paste to a solid bowl centrifuge, where the oil is separated from the rest of the paste as a result of the different density of these components. The oil that comes out of the decanter usually shows an unacceptably high content of moisture and impurities, so a further separation step is held, with the purpose of reducing the content of these undesirable components in the oil prior to its storage. Figure 1.4 shows a block diagram with the phases of the process.

The relevant global output variables of the VOOPP are the quality (q) of the obtained Virgin Olive Oil (VOO) and the industrial yield (μ). The quality of the oil (q) is determined by a series of physicochemical and organoleptic parameters included in the European Norm 2568/91, while the industrial yield is the ratio between the extracted oil and the total amount originally contained in the olives.

Fig. 1.4 Block diagram showing the main stages of the Virgin Olive Oil elaboration process

The paste preparation phase exerts a great influence on the quality of the oil, while also imposing an upper bound on the achievable yield. In turn, the oil separation stage finally dictates the obtained yield, with its influence on the quality being scarce.

The main variables that govern the paste preparation phase are the size of the sieve employed in the mill used to crush the olives, the temperature of the olive paste in the thermomixer, and the time that this paste remains there. Of these variables, the sieve size is directly manipulable, while the kneading time and temperature are not.

Since these two variables are not directly manipulable, and they exert an important influence on the global output variables—quality and yield—it is advisable to include feedback loops to assure that their values remain close to those desired.

The temperature of the paste inside the thermomixer depends on the temperature and flow of the incoming paste, the temperature and flow of the heating fluid, the heat exchange surface, the room temperature, and the outflow of paste. In turn, the residence time of the paste inside the thermomixer depends on its volume and the flows of paste entering and leaving it.

As can be seen, only variables directly related to the physical phenomena are pertinent for these control loops, with the previous and subsequent steps of the process being irrelevant. The typical manipulable variables used to control the kneading time and temperature are the paste inflow to the thermomixer and the position of a valve installed in the heating fluid pipe, respectively.

These loops constitute two examples of low-level control loops for the process: The output variables of the loops are not the main variables of interest of the process, they are easily measured using standard equipment, the achievable sampling times do not impose any constraints on the control loop, and a very reduced number of variables is of interest for the design of the controllers.

On the other hand, the control of the global output variables presents fairly different characteristics. To begin with, the measurement of the variables is not trivial. Although the measurement of the yield can be carried out using near-infrared spectroscopy (NIR) equipment, devices capable of providing online measurements of this variable are fairly expensive and still not a too mature technology, and their use is not widespread in the industry at all (Cano Marchal et al. 2011). At-line equipment is far more common, but its use introduces a severe constraint on the practically achievable sampling time.

As for the quality, the situation is even less favorable for its online measurement: There are many parameters that are relevant for the quality of the oil, but none of them is currently being routinely measured online in the industry (Cano Marchal et al. 2011). For some of the physicochemical parameters, NIR equipment is a promising technology, while for organoleptic parameters, the current research lines point to the

use of electronic noses for their determination (Escuderos et al. 2007; Esposto et al. 2008). In any case, a similar sampling rate constraint is found for these parameters as well.

Another key characteristic to consider for the control of the global outputs is the high number of variables that play a role in the process, and their intertwined effects on the output variables. Even disregarding the details of the low-level loops, and just considering their set-points as representative variables, the number of variables affecting the outputs is large enough to give rise to encountering a *fat plant*, with more inputs than outputs. Of these inputs, some are manipulable variables and some are to be considered as disturbances, but there are still extra degrees of freedom for the control of the plant.

These extra degrees of freedom provide the opportunity to consider additional objectives beyond that of the outputs reaching their specified set-points, such as minimizing the production cost. Thus, dealing with the system as a whole, in a multivariable framework, entails great interest.

Another argument that further supports the convenience of considering the system in a multivariable framework emerges when contemplating the selection of set-points for the global outputs. The opposite influence of some of the process variables on the outputs effectively results in the appearance of a trade-off relation between these outputs. Thus, the selection of optimal set-points for these variables requires regarding the constraints that fixing the value for one output imposes on the other.

Once that the interest for a multivariable model of the system has been established, the questions of what type of model and what variables should be included arise. The determination of the most convenient set-points for the output variables does not require a dynamic model, since a model comprising just the steady-state relations of the variables is adequate for this purpose. On the other hand, if the model is to be used in a model-based controller to close a feedback loop on the outputs, the dynamics of the relations between variables are, in principle, relevant, so a dynamic model is called for.

However, the typical large time constants of the systems involved, plus the limitation on the sampling time to be used in practice due to the difficulties imposed by the measurement of the variables, effectively limit the interest of dynamic models. Furthermore, the development of dynamic models of the relations is usually prohibitively expensive, due to the amount of samples required to perform an identification of the subsystems involved. Taking into account the cost and the benefit of using dynamic models, it is usually better to settle for static models, which will be able to take into account the relations between the steady states of the variables, although not their dynamic evolution.

This effectively means that the sampling frequency must be low enough to assure that the variables are in steady state, thus limiting the bandwidth of the disturbances that can be rejected by this high-level control loop. Here, the hierarchical control approach shows its convenience: Due to the characteristics of the process, the high-frequency disturbances of the process usually affect the intermediate variables, which are under feedback loops with a much higher bandwidth that, therefore, should be capable of rejecting them. In turn, the disturbances affecting directly the output

variables, such as the properties of the incoming olive, typically show low-frequency characteristics.

As for the variables to be included in the model, both process variables and raw input properties should be introduced in the model, as their effects on the outputs are important to be considered. Here, the hierarchical control structure allows to obviate the low-level dynamics of the plant and just consider the output variables of the low-level feedback loops, together with the manipulable variables that directly affect the high-level outputs. In practice, this means that only the kneading time and temperature must be considered along with the sieve size of the mill, allowing to obviate the position of the heating liquid valve, the inflow of paste, etc. In turn, the olive properties that affect the high-level outputs are the variety, the fat content, the moisture content, the ripeness, and the health of the olives, i.e., if they are damaged due to hail, plagues, or low temperatures.

The discussion about the characteristics and variables included in the model of the high-level layer of the VOOPP was motivated by the question of closing a feedback loop on the high-level outputs of the system. However, before we address how to tackle this feedback loop, it is desirable to know what reference values of the high-level output variables must be maintained.

Since there is a trade-off relation between quality and yield, a sound method to select what specific values of these variables to aim for is to set up an optimization problem whose solution could provide these values.

Assuming that the maximization of the profit is the main objective of the VOOPP, an objective function that encourages quality and yield and penalizes cost is appropriate. This optimization problem is necessarily constraint by the relations existing among the different variables involved in the VOOPP, a model of which is the high-level layer model already discussed. Thus, this model provides the knowledge of the system required to select the most appropriate set-points for the high-level output variables.

The production cost is related to the value of some of the process variables, but the value of these variables is decision variables of the optimization problem, and thus, their values are provided by the solution of the problem. The extra parameters required for the model are the unitary cost of each process variable, and the different market prices for each VOO quality.

The solution of this optimization problem provides both the reference values of the high-level process outputs and the set-points to feed to the low-level control loops. Furthermore, this optimization problem constitutes the base for the feedback control strategy: A model-based feedback controller for the high-level outputs can be devised considering a similar optimization problem and introducing an observer to estimate the disturbance affecting the process. This way, feedback is introduced in the control law, and disturbances and errors in the Modeling of the process models can be compensated.

Finally, an important remark is that the values of the properties of the incoming olives are fixed once that they arrive to the factory, but that they evolve along the season in the groves. Thus, if a model of the evolution of these properties in the

groves is available, we can introduce an additional degree of freedom to the problem by considering the production date as a decision variable.

The introduction of this additional decision variable transforms the problem into a Production Planning one, which requires, at least, to consider the trade-off relations existing between quality and yield, and may be further refined to consider the implications on the process costs of the varying raw input properties. That knowledge of the process is gathered in the models of the high-level layer of the process, so these models, or simplified versions of them, can be used for this purpose.

References

Allais I, Perrot N, Curt C, Trystram G (2007) Modelling the operator know-how to control sensory quality in traditional processes. J Food Eng 83(2):156–166

Amari A, Barbri NE, Bari NE, Llobet E, Correig X, Bouchikhi B (2009) Potential application of the electronic nose for shelflife determination of raw milk and red meat. In: AIP conference proceedings, vol 1137. AIP Publishing, pp 457–460

Aparicio R, Rocha SM, Delgadillo I, Morales MT (2000) Detection of rancid defect in virgin olive oil by the electronic nose. J Agric Food Chem 48(3):853–860

Birle S, Hussein M, Becker T (2013) Fuzzy logic control and soft sensing applications in food and beverage processes. Food Control 29(1):254–269

Caldwell D (ed) (2012) Robotics and automation in the food industry: current and future technologies, 1st edn. Woodhead Publishing, Philadelphia

Cano Marchal P, Gómez Ortega J, Aguilera Puerto D, Gámez García J (2011) Situación actual y perspectivas futuras del control del proceso de elaboración del aceite de oliva virgen. Revista Iberoamericana de Automática e Informática Industrial RIAI 8(3):258–269

Capone S, Epifani M, Quaranta F, Siciliano P, Taurino A, Vasanelli L (2001) Monitoring of rancidity of milk by means of an electronic nose and a dynamic PCA analysis. Sens Actuators B Chem 78(1–3):174–179

Davies ER (2009) The application of machine vision to food and agriculture: a review. Imaging Sci J 57(4):197–217

ECPC (2012) North American Industry Classification System (NAICS) main page

Escuderos ME, Uceda M, Sánchez S, Jiménez A (2007) Instrumental technique evolution for olive oil sensory analysis. Eur J Lipid Sci Technol 109(5):536–546

Esposto S, GianFrancesco M, Roberto S, Ibanez R, Agnese T, Stefania U, Maurizio S (2008) Monitoring of virgin olive oil volatile compounds evolution during olive malaxation by an array of metal oxide sensors. Food Chem

EU (2008) Statistical classification of economic activities in the European community, Rev 2

Europe F (2014) Data and trends of the European food and drink industry 2013–2014

Forbes JC, Watson D (1992) Plants in agriculture. Cambridge University Press, Cambridge

Huang H, Yu H, Xu H, Ying Y (2008) Near infrared spectroscopy for on/in-line monitoring of quality in foods and beverages: a review. J Food Eng 87(3):303–313

Jiménez A, Molina A, Pascual MI (2005) Using optical NIR sensor for on-line virgin olive oils characterization. Sens Actuators B 107:64–68

Kawasaki M, Kawamura S, Tsukahara M, Morita S, Komiya M, Natsuga M (2008) Near-infrared spectroscopic sensing system for on-line milk quality assessment in a milking robot. Comput Electron Agric 63(1):22–27

Loutfi A, Coradeschi S, Mani GK, Shankar P, Rayappan JBB (2015) Electronic noses for food quality: a review. J Food Eng 144:103–111

Majumdar S, Jayas DS (2000) Classification of cereal grains using machine vision: IV. Combined morphology, color, and texture models. Trans ASAE 43(6):1689–1694

Mery D, Chanona-Pérez JJ, Soto A, Aguilera JM, Cipriano A, Veléz-Rivera N, Arzate-Vázquez I, Gutiérrez-López GF (2010) Quality classification of corn tortillas using computer vision. J Food Eng 101(4):357–364

Moubarac J-C, Parra DC, Cannon G, Monteiro CA (2014) Food classification systems based on food processing: significance and implications for policies and actions: a systematic literature review and assessment. Curr Obes Rep 3(2):256–272

Nollet LML, Toldra F (2006) Advanced technologies for meat processing. CRC Press, Boca Raton

Perrot N, Ioannou I, Allais I, Curt C, Hossenlopp J, Trystram G (2006) Fuzzy concepts applied to food product quality control: a review. Fuzzy Sets Syst 157(9):1145–1154

Prieto N, Andrés S, Giráldez FJ, Mantecón AR, Lavín P (2006) Potential use of near infrared reflectance spectroscopy (NIRS) for the estimation of chemical composition of oxen meat samples. Meat Sci 74(3):487–496

Rehbein H, Oehlenschlager J (2009) Fishery products: quality, safety and authenticity. Wiley, New Jersey

Savenije B, Geesink GH, van der Palen JGP, Hemke G (2006) Prediction of pork quality using visible/near-infrared reflectance spectroscopy. Meat Sci 73(1):181–184

Śliwińska M, Wiśniewska P, Dymerski T, Namieśnik J, Wardencki W (2014) Food analysis using artificial senses. J Agric Food Chem 62(7):1423–1448

Sun D-W (2000) Inspecting pizza topping percentage and distribution by a computer vision method. J Food Eng 44(4):245–249

Sun D-W (2004) Computer vision-an objective, rapid and non-contact quality evaluation tool for the food industry. J Food Eng 61(1):1–2

Sun D-W (2008) Infrared spectroscopy for food quality analysis and control. Academic Press, Amsterdam

Tudu B, Jana A, Metla A, Ghosh D, Bhattacharyya N, Bandyopadhyay R (2009a) Electronic nose for black tea quality evaluation by an incremental RBF network. Sens Actuators B Chem 138(1):90–95

Tudu B, Metla A, Das B, Bhattacharyya N, Jana A, Ghosh D, Bandyopadhyay R (2009b) Towards versatile electronic nose pattern classifier for black tea quality evaluation: an incremental fuzzy approach. IEEE Trans Instrum Meas 58(9):3069–3078

UN (2008) United Nations statistics division - classifications registry

Wan YN, Lin CM, Chiou JF (2000) Adaptive classification method for an automatic grain quality inspection system using machine vision and neural network. American Society of Agricultural Engineers, pp 1–19

Welle R, Greten W, Rietmann B, Alley S, Sinnaeve G, Dardenne P (2003) Near-infrared spectroscopy on chopper to measure maize forage quality parameters online. Crop Sci 43(4):1407

Woodcock T, O'Donnell C, Downey G (2008) Review: better quality food and beverages: the role of near infrared spectroscopy. J Near Infrared Spectrosc 16(1):1

World Heath Organization and FAO (1993) CODEX Alimentarius: list of standards

Wu D, Sun D-W (2013) Colour measurements by computer vision for food quality control - a review. Trends Food Sci Technol 29(1):5–20

Tao Y, Heinemann PH, Varghese Z, Morrow CT, Sommer HJ III (1995) Machine vision for color inspection of potatoes and apples. Trans ASAE 38(5):1555–1561

Zhang B, Huang W, Li J, Zhao C, Fan S, Wu J, Liu C (2014) Principles, developments and applications of computer vision for external quality inspection of fruits and vegetables: a review. Food Res Int 62:326–343

Chapter 2
Modeling and System Identification

In order to *alter* or *influence* the way a process operates, one should previously have a proper understanding of the relations between its variables, its natural behavior, and what is reasonable to expect of the controlled system. This understanding is usually expressed using *models*, which are mathematical objects that represent the behavior and features of the system that is of interest to us. The decision of which features to include and to what degree of accuracy strongly depends on the purpose of the model. Typically, models aimed at simulation require a higher level of detail than those to be used strictly for control purposes. On the other hand, models to be used for classical feedback control need to capture the dynamics of the system, while static models are typically enough for production planning purposes.

As introduced in Chap. 1, the low- and high-level control layers of food process industry processes exhibit different requirements from a control point of view, so the models needed vary accordingly. On the one hand, the low-level layer demands dynamic models that capture the transient behavior of these systems. Output variables for these systems as usually easily measured online, so System Identification techniques can be applied. Furthermore, these systems are sometimes simple enough to employ first principles in order to obtain an appropriate model structure whose parameters can be identified using experimental data. Section 2.1 introduces these crisp Modeling techniques.

On the other hand, the complexity of the systems involved in the high-level layer and the difficulty of measuring the relevant output variables lead to considering alternative techniques for the construction of appropriate system models. In this scenario, fuzzy logic emerges as a convenient tool and its fundamentals and some particularly apt techniques are presented in Sect. 2.2.

P. Cano Marchal et al., *Production Planning, Modeling and Control of Food Industry Processes*, Advances in Industrial Control,
https://doi.org/10.1007/978-3-030-01373-8_2

2.1 Crisp Modeling

Crisp Modeling techniques are applicable when either a good understanding of the physics of the process or plenty of experimental data are available, as is typically the case for the low-level layer of food manufacturing processes.

As discussed above, dynamic models provide more information than static ones and they are typically indispensable for low-level control applications. Dynamic models can be expressed using different mathematical formulations, each with different merits, that can be transformed from one into another quite easily. The main formulations of interest to us are:

- Ordinary differential equations (ode): This is the typical formulation in which physical and chemical laws are expressed, so it is a natural approach to constructing models using first principles. This formulation is expressed as relations between the derivatives of the variables involved. For a single-input, single-output (SISO) system where u and y denote the input and output variables, respectively, a general linear differential equation model is expressed as:

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = b_m \frac{d^m u}{dt^m} + b_{m-1}\frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u \tag{2.1}$$

- State-space models: These models express the relations between the variables using two sets of equations and the notion of *state variables*. State variables capture the status of the system, in the sense that knowing their value at a given instant and the inputs from that point on allows to know the evolution of the system. These models are constituted of two sets of equations: state transition equations and output equations. The state transition equations relate the derivatives of state variables with the state variables themselves and the inputs and describe the evolution of the system. The output equations relate the output of the system with the state variables. These models can be used for SISO systems and are specially convenient for multiple-input, multiple-output (MIMO) systems.
  A general nonlinear space-state model is expressed as

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}), \\ \mathbf{y} &= h(\mathbf{x}, \mathbf{u}). \end{aligned}$$

  Linear models can also be expressed very compactly using matrix notation as

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u}, \\ \mathbf{y} &= C\mathbf{x} + D\mathbf{u}. \end{aligned}$$

  Here, A, B, C, and D are matrices of appropriate dimensions.

- Transfer function models: These models are typically used for SISO systems and relate the Laplace transforms of the input and output variables. They are widely used in control, as they provide valuable insight of the frequency response of the system. The transfer function representation of the system modeled in Eq. (2.1) is:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} \tag{2.2}$$

  Here, s represents the Laplace variable.

These models assume that the independent variable is time and that it is continuous. When a controller is implemented in a digital computer, the data are collected only at certain time instants and the computer effectively *sees* a collection of points. It is sometimes desirable to directly model the relation between these number sequences directly, particularly for System Identification purposes and when the sampling time is relatively large compared to the process typical time constant. In process control, it is not usually the case that fast dynamics arise; however, many of the available System Identification techniques are developed using sampled systems, as is the natural way of dealing with the data available to perform the System Identification.

The analog of differential equations is difference equations, where the values of past time instants of the variables of interest are related:

$$y_t + a_1 y_{t-1} + \cdots + a_n y_{t-n} = b_0 u_t + b_1 u_{t-1} + \cdots + b_m u_{t-m}. \tag{2.3}$$

Discrete-time state-space models can also be constructed. These relate the value of the state variables in the next time step with the current and past values of the inputs and themselves:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots, \mathbf{x}_{t-n}, \mathbf{u}_{t-1}, \ldots, \mathbf{u}_{t-m}) \tag{2.4}$$

$$\mathbf{y}_t = h(\mathbf{x}_t, \mathbf{u}_t) \tag{2.5}$$

Finally, discrete alternative of the transfer function models are also available via the z-transform. A typical model is:

$$G(z^{-1}) = \frac{Y(z^{-1})}{U(z^{-1})} = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} \tag{2.6}$$

Linear versus nonlinear versus linearization. In control engineering practice, it is common to work with linear models of the systems. Systems in real life are rarely linear. However, for control purposes, we usually want to have the system operating close to a specified set-point. Under these circumstances, a linear model valid only in the vicinity of the set-point is very useful. For simulation purposes, nonlinear models are preferable, as a higher degree of accuracy and a higher range of validity of the model are usually required.

2.1.1 First Principles Modeling

First principle Modeling means using fundamental physical or chemical laws to model the system of interest. These techniques are applicable when a good understanding of these relations is available and the system is simple enough for the task to be tractable. Frequently, these laws provide the structure of the models, but the specific values of some of the parameters may be unknown or difficult to estimate. Under these circumstances, the first principle Modeling is used to provide a specific model structure (order of the equations, existence an approximate order of magnitude of delays, existence of integrators in the model, etc.) whose parameters can be determined using System Identification techniques.

Next, we briefly introduce the type of systems most frequently found in the low-level layers of food processing and the laws that govern their behavior.

Material balances. Mass conservation is the physical principle that enables the construction of material balance equations. Simply stated, the principle affirms that whatever matter is introduced into the system either outputs it or is accumulated. This can be expressed using an ordinary differential equation as:

$$q_{in}(t) = q_{out}(t) + \frac{dm(t)}{dt}. \tag{2.7}$$

Here, q_{in} is the mass flow that inputs the system, q_{out} is the mass flow that leaves the system and $\frac{dm(t)}{dt}$ is the accumulation term. This is a dynamic equation that is useful for understanding how fast is mass accumulating in or leaving the system. In steady state, that is, when variables do not change with time, all derivatives are zero, so the input flow must be equal to the output flow and the mass inside the system will not change.

This equation is applicable anywhere we have a flow of material entering and leaving a system. Three aspects must be taken into account when writing the equations. The first is that the principle states that *mass* is conserved, not *volume*. That means that if the material we are dealing with does not have constant density, mass flows must absolutely be used; if the density is constant, volumetric flows are equivalent to mass ones, so either one can be employed. The second consideration regards flows that are compounded by different components that leave the system at different stages. In this case, mass conservation can be applied to each of the components individually. The final warning deals with systems where chemical reactions take place. In this case, the mass balance must consider each of the elements individually.

Energy balance and heat transfer. Analogously to mass conservation, the first principle of thermodynamics that states the conservation of energy is the driving principle of energy balance equations. This principle can be stated as:

$$\frac{dH(t)}{dt} = Q(t) - W(t). \tag{2.8}$$

Here, $H(t)$, $Q(t)$, and $W(t)$ represent enthalpy, heat, and mechanical work, respectively. For these balances, $H(t)$, $Q(t)$, and $W(t)$ need to be particularized to reflect the specifics of the process at hand, typically taking into account constitutive equations of the materials and the physical phenomena involved. For instance, the energy balance of a heat exchanger where there are not any phase transitions is much simpler than the case when phase transitions indeed take place; however, both cases ultimately respond to the fundamental law expressed in Eq. (2.8).

Heat transfer is the transmission of thermal energy between systems due to their difference in temperature. Heat transfer laws typically need to be considered in thermal systems to model the rate at which the energy transference is taking place.

There are three mechanisms of heat transfer: conduction, convection, and radiation. Conduction is caused by the exchange of kinetic energy between particles of adjacent systems; convection is provoked by the transportation of energy in moving fluids, and radiation is the transmission of energy performed by photons in electromagnetic waves.

Any given system typically exhibits all the three different heat transfer mechanisms, however with varying relative degrees of relevance. It is common practice for the Modeling of systems, thus, to simplify the analysis and just consider those mechanisms that exert the greatest influence in the overall heat transfer.

The laws that govern the heat transfer are different for each mechanism. Conduction is governed by a diffusion law and can be modeled as a term that is proportional to the difference of temperature of the systems

$$\frac{dQ}{dt} = \frac{k\,A}{x}(T_1 - T_2).$$

Here, k is the heat conductivity, A is the area, and x is the width of the material. Convection is also proportional to the difference of temperature of the systems

$$\frac{dQ}{dt} = h\,A(T_1 - T_2).$$

Here, A is the area and h is the convection coefficient that takes into account the type of fluid and the average velocity of the fluid. Radiation, in turn, is proportional to the fourth power of the temperature

$$Q = \varepsilon\sigma(T_1^4 - T_2^4).$$

Here, Q is the heat flux, ε is the emissivity, and σ is the Stefan–Boltzmann constant.

Mass transfer. Mass transfer is a phenomenon caused by a net movement of particles from one location to another. Typical examples of this phenomenon include evaporation, membrane filtration, distillation, or precipitation. The driving force of this phenomenon is the difference in concentration between the different media that compose the system.

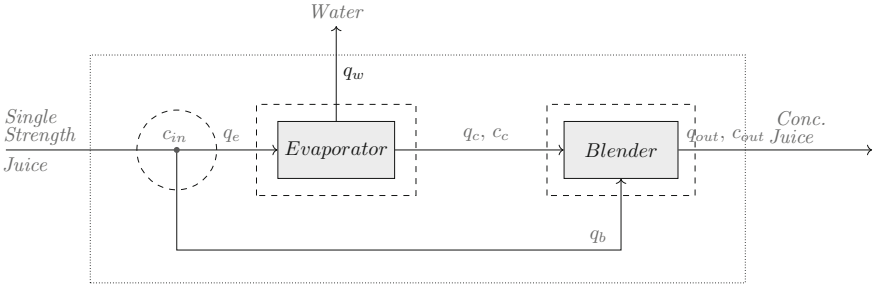


Fig. 2.1 Block diagram of the orange juice concentration example

For a one-dimensional system, Fick's Law expresses the relation between the involved variables

$$J = -D\frac{dc}{dx}. \tag{2.9}$$

Here, J is the diffusion flux, D is a diffusion coefficient, c is the concentration of the component, and x represents the spatial dimension considered.

Hydraulic systems. Hydraulic systems can be used to model variation of level in reactors and tanks. These systems are basically mass conservation systems that include a constitutive equation that somehow relates the output flow of the tank or reactor with its current height.

For laminar flow, the output can be considered to be proportional to the height of liquid, so the differential equation that models a tank of constant section is:

$$\frac{dh(t)}{dt} = q_{in}(t) - \frac{1}{R}h(t). \tag{2.10}$$

Here, R is a constant that takes into account aspects such as the diameter of the output nozzle, etc.

For turbulent flow, Torricelli's law states that the output velocity of the fluid is proportional to the square root of the height of fluid above; thus,

$$\frac{dh(t)}{dt} = q_{in}(t) - B\sqrt{h(t)}. \tag{2.11}$$

Again, B is a constant that takes into account different aspects, such as the geometry of the output nozzle.

Let us consider, as an example, an orange juice concentration process as defined in Fig. 2.1. Let us begin assuming that the mass flows coming into the blender are constant, so the total mass inside the blender remains constant as well. Under these circumstances, the blender can be modeled as a simple tank with perfect mixture, where the output concentration is the one in the tank:

$$M_b \frac{dc_{out}(t)}{dt} = q_c\, c_c(t) + q_b\, c_{in}(t) - q_{out}\, c_{out}(t). \tag{2.12}$$

If the incoming flows into the blender are allowed to vary as well, then we have to set up two equations: one for the total mass of the system and a second for the solid components.

The total mass conservation states that:

$$\frac{dM_b(t)}{dt} = q_c(t) + q_b(t) + q_{out}(t). \tag{2.13}$$

In turn, now balance for the solids now states:

$$\begin{aligned} \frac{d}{dt}(M_b\, c_{out}) &= q_c(t)\, c_c(t) + q_b(t)\, c_{in}(t) - q_{out}(t)\, c_{out}(t), \\ M_b(t)\frac{dc_{out}(t)}{dt} + c_{out}(t)\frac{dM_b(t)}{dt} &= q_c(t)\, c_c(t) + q_b(t)\, c_{in}(t) - q_{out}(t)\, c_{out}(t) \end{aligned}. \tag{2.14}$$

Using Eq. (2.13), this can be rewritten as:

$$\begin{aligned} M_b(t)\frac{dc_{out}(t)}{dt} &= q_c(t)\, c_c(t) + q_b(t)\, c_{in}(t) - q_{out}(t)\, c_{out}(t) - c_{out}(t)\left[q_c(t) + q_b(t) + q_{out}(t)\right] \\ M_b(t)\frac{dc_{out}(t)}{dt} &= q_c(t)\,(c_c(t) - c_{out}(t)) + q_b(t)\,(c_{in}(t) - c_{out}(t)). \end{aligned} \tag{2.15}$$

The equations that model the evaporator, assuming that we can simply choose q_w as desired, are equivalent to the ones in the blender:

$$\begin{aligned} \frac{dM_e(t)}{dt} &= q_e(t) - q_w(t) + q_c(t) \\ M_e(t)\frac{dc_c(t)}{dt} &= q_e(t)\,(c_{in}(t) - c_c(t)) + q_w(t)\, c_c(t) \end{aligned} \tag{2.16}$$

The positive sign in the term regarding $q_w(t)$ means that the concentration in the evaporator should rise if the flow of water increases, which makes intuitive sense.

So, if we assume constant flows and levels, the whole system can be modeled by a second-order linear state-space model using the concentrations in the evaporator and the tank as states. However, if we suppose that the levels and flows may vary, then we have a fourth-order nonlinear system.

2.1.2 Black-Box Versus Gray-Box Versus White-Box Modeling

First principle Modeling requires a fair understanding of the relations that exist between the different variables of interest of the process. It is not rare that the process

of interest is complex enough that the task of deriving a first principle model of it is very arduous or even unfeasible. System Identification is an alternative approach that can be considered if experimental data are available.

The idea behind System Identification is to excite the system using appropriate inputs, record the response of the system outputs, and select, out of a set of possible candidates, the model that best describes or reproduces the recorded experimental behavior of the system.

Models that do not assume any knowledge of the system are called *black-box* models, in the sense that the model is considered a box that simply provides the correct outputs for the given inputs, without much regard of the inner workings behind that relation. If some knowledge of the system is considered for the selection of the model structure but its parameters are identified using experiments, the model is referred to a *gray-box* model. Continuing the analogy, if a model structure and parameters are derived using first principles exclusively, the model is labeled as a *white-box* model, in the sense that all its inner workings are known.

There are very good software packages that implement the routines required to perform a black-box System Identification, so the practitioner does not need to dwell on the mathematical implementation details of the System Identification techniques. Particularly fit for these purposes is the excellent System Identification Toolbox of MATLAB (MATLAB System Identification Toolbox 2014). One drawback of these packages is, precisely, their orientation toward black-box Modeling, which makes using gray-box Modeling a more challenging task, since it might not be easy to extract the value of the parameters of a custom model structure using standard System Identification tools.

It is sometimes the case that some physical intuition can be used to select a specific model structure, such as the order of the model or the existence of an integrator or a time delay. In these cases, the models can also be referred to gray-box models, as some knowledge has been put into them. It is usually good practice to reflect a bit on these issues, as the selection of the system model and the disturbance dynamics may have considerable influence in the final behavior of the system.

An indirect approach to gray-box Modeling is to identify a black-box model with a prespecified structure and to find the relation between its parameters and those of the white-box model, and consequently to identify the values of the parameters of the latter. Depending on the structure of the models, it might be the case that only ratios of some of the parameters can be known, and not the values of the independent elements.

This arises the question of identifiability, that is, when is it possible to identify a certain model using a certain set of experimental data. Or, in other words, what characteristics should an experimental data set include to allow the identification of a certain type of model.

The identification of a black- or gray-box model can be performed using historical process data or by performing a System Identification experiment. This issue of when historical data can be used to identify the system or what is the best way to perform a System Identification experiment is not trivial and will be further discussed in Sect. 2.1.5.

The type of models and the approximations used for its identification have implications in their validity. First Principle models typically can be used in a wide range of process situations, while black-box models, being linear approximations of the real process, necessarily have a much narrower application range. A viable approach to overcome this limitation is to identify different models for different working regimes, however at the expense of additional work.

The core elements required to identify a system are the set of experimental data, commonly denoted as Z and a set of candidates models. The idea is to choose from that set of candidates the one that better fits the experimental data set. For this selection, there are two main methodologies: On the one hand, prediction error methods use an optimization approach to find the parameters of the model, while subspace identification methods project the data into appropriate subspaces to find these parameters. The next sections further detail these approaches.

2.1.3 Prediction Error Methods

Prediction error methods aim to identify the system model minimizing a function that expresses some measure of the expected difference between a predicted system output and the actual one. This predicted output is computed using a prespecified model structure—whose unknown parameters are the optimization variables—and the inputs and observed outputs up to a certain time instant. Mathematically, this objective function can be expressed as:

$$J(\theta) = \sum_{t=1}^{T-\tau} \left\| y_{t+\tau} - \hat{y}_{t+\tau|t}(\theta) \right\|. \tag{2.17}$$

Here, $\hat{y}_{t+\tau|t}(\theta)$ denotes the predicted value of the output y at time $t+\tau$ computed using the information available at time t for a model with parameters θ.

The fundamental model used for prediction error methods supposes that the system output is composed of two terms: the effect of the measurable input signal u_t and the influence of unmeasured noise or disturbances e_t. Mathematically,

$$y_t = G(z^{-1})u_t + H(z^{-1})e_t \tag{2.18}$$

Different types of models are considered in the literature depending on the structure of $G(z^{-1})$ and $H(z^{-1})$, each making distinct assumptions about the relations between the model and the noise and providing different characteristics to the identified model. The most common way to generically express $G(z^{-1})$ and $H(z^{-1})$ is to use discrete-time transfer functions:

$$G(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})F(z^{-1})}, \quad H(z^{-1}) = \frac{C(z^{-1})}{A(z^{-1})D(z^{-1})}. \tag{2.19}$$

The coefficients of these polynomials are the parameters generically referred to θ in Eq. (2.17) above. Identifiability considerations advocate for considering these polynomials to be co-prime—i.e., supposed not to have common factors—and the denominators to be monic—i.e., having 1 as the leading coefficient. The notion of identifiability will be further discussed in Sect. 2.1.5; for now, it is enough to see that a special structure of these polynomials is required.

Out of this general structure, the most common models usually employed are:

1. ARX models: This family of models are called *auto-regressive exogenous* and can be obtained from the general expression setting $C(z^{-1}) = D(z^{-1}) = F(z^{-1}) = 1$; thus,
$$y_t = \frac{B(z^{-1})}{A(z^{-1})}u_t + \frac{1}{A(z^{-1})}e_t. \tag{2.20}$$
In this model, $G(z^{-1})$ and $H(z^{-1})$ share all their poles and the coupling of the input is defined through polynomial $B(z^{-1})$.
2. ARMAX models: These are the *auto-regressive moving average eXogenous* models and generalize ARX models by allowing e_t to be colored by a polynomial $C(z^{-1})$:
$$y_t = \frac{B(z^{-1})}{A(z^{-1})}u_t + \frac{C(z^{-1})}{A(z^{-1})}e_t. \tag{2.21}$$
3. Output error models: These models suppose that the noise directly affects the output, without being filtered through the dynamics of the system:
$$y_t = \frac{B(z^{-1})}{F(z^{-1})}u_t + e_t. \tag{2.22}$$
Although playing the same role as $A(z^{-1})$, the denominator of $G(z^{-1})$ is denoted $F(z^{-1})$ to emphasize that there are no common dynamics with the noise model, as $A(z^{-1})$ is usually reserved to precisely denote common dynamics between $G(z^{-1})$ and $H(z^{-1})$.
4. Box–Jenkins models: These are the most general models and consider independent dynamics of $G(z^{-1})$ and $H(z^{-1})$:
$$y_t = \frac{B(z^{-1})}{F(z^{-1})}u_t + \frac{C(z^{-1})}{D(z^{-1})}e_t. \tag{2.23}$$

For simplicity, we will focus on the ARX model to highlight the key ideas related to prediction error method system identification. The objective is to find an expression to compute a prediction of the value of y using the model in Eq. (2.20). For that, we express it as a difference equation and reorganize the terms:

$$\begin{aligned} A(z^{-1})y_t &= B(z^{-1})u_t + e_t \\ y_t + (A(z^{-1}) - 1)y_t &= B(z^{-1})u_t + e_t \\ y_t &= \left(1 - A(z^{-1})\right) y_t + B(z^{-1})u_t + e_t \\ y_t &= -a_1 y_{t-1} - \cdots - a_n y_{t-n} + b_0 u_t + b_1 u_{t-1} + \cdots + b_m u_{t-m} + e_t \end{aligned}$$

The key observation is that all the terms in the right member regarding y get up to $t-1$. This means that we can compute an estimate of the current value of y, namely y_t, if we know y up to the previous instant, along with the corresponding values of u and e.

$$\hat{y}_{t|t-1} = -a_1 y_{t-1} - \cdots - a_n y_{t-n} + b_0 u_{t|t-1} + b_1 u_{t-1} + \cdots + b_m u_{t-m} + e_{t|t-1} \tag{2.24}$$

The above equation includes the term $u_{t|t-1}$, that is, in order to compute our estimate we should know the value that u is to have the next time instant. There are two main reasons why this is not a huge hurdle. The first is that u is the input variable, and as such, its value is decided by the plant operator, so we can reasonably assume that we know what value we are going to use in the following time instant. The second consideration is that given the digital implementation of control systems, it is very unlikely that the coefficient b_0 is different from zero. That is, real control systems usually show a time delay of at least one sampling instant, so the term requiring knowledge of u_t at time $t-1$ simply vanishes due to its associated coefficient.

A greater challenge is posed by the term $e_{t|t-1}$. Since e is considered to be unmeasured, it is not only that we do not know what value will e have at the next time instant, it is that we can not even know its value at the current time instant! We can, however, characterize e statistically and make an educated guess of its most likely value. Since e is usually supposed to have zero mean, the best guess is precisely zero. If the model structure is other than ARX, terms like $e_{t-1|t-1}$ appear in the predictor equation. These terms, unlike $e_{t|t-1}$, are not zero in general, but their value can be estimated using past observations of y, so the predictor can also be computed with the available experimental data.

Now that we know how to compute $\hat{y}_{t+\tau|t}$, it is time to comment on the most frequent choices for the particular measure of the prediction error used for J in Eq. (2.17). Choosing the square of the differences gives rise to the least squares estimation method, which is the default option for most applications. If the probability density function of the e_k is known, the maximum likelihood (MLE) or the maximum a posteriori Bayesian (MAP) methods can be used. The solution of the optimization method provides the parameters θ that together with the model structure already specified comprise the final model of the system identified from the available experimental data.

An initial educated guess of appropriate model structures can found using spectral analysis or correlation methods. Also, subspace methods can be used to obtain an initial estimation of appropriate dimensions of the model.

The literature on System Identification is rich, and plenty of good references can be found. We refer the reader to Tangirala (2014), Ljung (1998), and Johansson (1993) for a deep treatment of the subject. These methods are implemented in the excellent System Identification Toolbox of MATLAB (MATLAB System Identification Toolbox 2014).

2.1.4 Subspace Identification

Subspace identification methods take a different approach from prediction error methods. In this case, the aim is to identify a space-state model of the system without prespecifying any particular model structure. This lack of requirement of specifying a structure is an important advantage of subspace methods over their prediction error counterparts. For SISO systems, this property is handy but probably not a fundamental one, as it is not hard to propose a set of possible model structures, check their fit and select the best one. This property is fully leveraged when we combine it with the intrinsic suitability of space-space models to deal with multivariable systems. In effect, prediction error methods have two drawbacks for multivariable systems: On the one hand, transfer function models are less adequate for this type of systems; on the other hand, if the exact structure of the models is not known, testing different structures is much more time consuming due to the essentially combinatorial nature of the task. Testing three-order models and three delays requires nine combinations for a SISO system, while it requires 36 for a two-input, two-output MIMO system and 81 for a three-input, three-output one.

One drawback of subspace methods is that standard methods can not accommodate previous knowledge of the model structure into the identification procedure. This restriction also implies that the realization of the identified space-state model may not be easy to interpret physically.

Before delving into subspace identification, it is convenient to devote some attention to the notions of *controlability* and *observability*, which are key properties of state-space models. Controllability is concerned with whether we can drive the system to any desired state, provided that we employ appropriate inputs, while *observability* deals with whether it is possible to obtain an estimate of the initial state vector given sufficient observations of the output variables. For simplicity, we will assume a SISO system, although the ideas are easily applied to MIMO systems.

Let us consider a discrete-time state-space model of the type:

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + Bu_t \tag{2.25a}$$

$$y_t = C\mathbf{x}_t + Du_t \tag{2.25b}$$

with $A \in \mathscr{R}^{n\times n}$, $B \in \mathscr{R}^{n\times 1}$, $C \in \mathscr{R}^{1\times n}$ and $D \in \mathscr{R}$.

Let us further suppose that A, B, C, and D are known, we have access to u_t and y_t for $t \geq 0$, and our objective is to ascertain what the value of $\mathbf{x}_0$ was. Just knowing y_0 and u_0 is not sufficient for recovering $\mathbf{x}_0$, as $x \in \mathscr{R}^n$ and the output equation of the state-space model (Eq. 2.25b) just provides us with a single relation between the variables. We must find $n-1$ additional equations so that we have n equations for n unknowns.

The first idea is noting that we can use the state transition equation (Eq. 2.25a) recursively to express any subsequent vector state x_t in terms of the initial one $\mathbf{x}_0$ and the sequence of inputs exerted $u_0, u_1, \ldots u_t$:

$$\begin{aligned}
&\mathbf{x}_1 = A\,\mathbf{x}_0 + B\,u_0;\\
&\mathbf{x}_2 = A\,\mathbf{x}_1 + B\,u_1 = A\,(A\,\mathbf{x}_0 + B\,u_0) + B\,u_1 = A^2\,\mathbf{x}_0 + \begin{bmatrix}AB & B\end{bmatrix}\begin{bmatrix}u_0\\u_1\end{bmatrix}\\
&\mathbf{x}_3 = A\,\mathbf{x}_2 + B\,u_2 = A\left(A^2\,\mathbf{x}_0 + \begin{bmatrix}AB & B\end{bmatrix}\begin{bmatrix}u_0\\u_1\end{bmatrix}\right) + B\,u_2 = A^3\,\mathbf{x}_0 + \begin{bmatrix}A^2B & AB & B\end{bmatrix}\begin{bmatrix}u_0\\u_1\\u_2\end{bmatrix}\\
&\mathbf{x}_t = A^t\mathbf{x}_0 + \begin{bmatrix}A^{t-1}B & A^{t-2}B & \cdots & AB & B\end{bmatrix}\begin{bmatrix}u_0\\u_1\\\vdots\\u_{t-2}\\u_{t-1}\end{bmatrix}
\end{aligned} \tag{2.26}$$

This expression provides some additional insight into the notion of controlability. The *extended controlability matrix* $\mathscr{C}_t \in \mathscr{R}^{n\times t}$ is defined as

$$\mathscr{C}_t = \begin{bmatrix}A^{t-1}B & A^{t-2}B & \cdots & AB & B\end{bmatrix}. \tag{2.27}$$

Equation (2.26) affirms that the states that can be reached must essentially lie in the subspace spanned by the columns of $\mathscr{C}_t$. If $\mathscr{C}_t$ spans the whole $\mathscr{R}^n$, then any desired state is reachable, as we could always find appropriate values for u_0, u_1, …, u_t such that we *hit* that point. This means that we need at least n time steps to span the whole state space and that the matrix $\mathscr{C}_n$ must be *invertible*—i.e., its columns must be linearly independent—for the system to be controllable.

Having Eq. (2.26) to compute any state in terms of the initial one and the inputs, we can relate $\mathbf{x}_0$ not only with y_0, but also with the subsequent output values available $y_1, y_2, \ldots, y_t$:

$$
\begin{aligned}
y_0 &= C\,\mathbf{x}_0 + D\,u_0 \\
y_1 &= C\,\mathbf{x}_1 + D\,u_1 = C\;(A\,\mathbf{x}_0 + B\,u_0) + D\,u_1 = CA\,\mathbf{x}_0 + \begin{bmatrix} CB & D \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} \\
y_2 &= C\,\mathbf{x}_2 + D\,u_2 = C\,\left(A^2\,\mathbf{x}_0 + \begin{bmatrix} AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix}\right) + D\,u_2 = CA^2\,\mathbf{x}_0 + \begin{bmatrix} CAB & CB & D \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \\
y_{t-1} &= C\,\mathbf{x}_{t-1} + D\,u_{t-1} = CA^{t-1}\,\mathbf{x}_0 + \begin{bmatrix} CA^{t-2}B & CA^{t-3}B & \cdots & CB & D \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{t-1} \end{bmatrix}
\end{aligned}
\tag{2.28}
$$

We can express the set of equations above in a matrix equation:

$$
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{t-2} \\ y_{t-1} \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{t-2} \\ CA^{t-1} \end{bmatrix} \mathbf{x}_0 + \begin{bmatrix} D & 0 & 0 & \cdots & 0 & 0 \\ CB & D & 0 & \cdots & 0 & 0 \\ CAB & CB & D & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ CA^{t-3}B & CA^{t-4}B & CA^{t-5}B & \cdots & D & 0 \\ CA^{t-2}B & CA^{t-3}B & CA^{t-4}B & \cdots & CB & D \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{t-2} \\ u_{t-1} \end{bmatrix}
\tag{2.29}
$$

Here, the only unknown is $\mathbf{x}_0$. Let us introduce some further notation and rewrite the above equation more compactly, using $\mathbf{y}_{0,t-1} \in \mathscr{R}^t$, $\mathbf{u}_{0,t-1} \in \mathscr{R}^t$, $\mathscr{O}_t \in \mathscr{R}^{t \times n}$ and $\mathscr{G}_t \in \mathscr{R}^{t \times t}$ as:

$$
\mathbf{y}_{0,t-1} = \mathscr{O}_t\,\mathbf{x}_0 + \mathscr{G}_t\,\mathbf{u}_{0,t-1}. \tag{2.30}
$$

Rearranging the terms, we have:

$$
\mathbf{x}_0 = \mathscr{O}_t^{-1}\,\left(\mathbf{y}_{0,t} - \mathscr{G}_t\,\mathbf{u}_{0,t}\right). \tag{2.31}
$$

Here, $\mathscr{O}_t$ is termed as the *extended observability matrix*. This equation shows that the condition for the system to be observable is that $\mathscr{O}_t$ be invertible, so we need n samples and the columns of $\mathscr{O}_t$ must be linearly independent.

With this discussion, we have fulfilled our objective of recovering the value of $\mathbf{x}_0$, given full knowledge of the system matrices A, B, D, and D. The objective now is to estimate the values of these matrices supposing that they are unknown.

Equation (2.30) is the key relation for subspace identification methods. Let us begin noting that the same relation can be written if we move forward one step in time; namely,

$$
\mathbf{y}_{1,t} = \mathscr{O}_t\,\mathbf{x}_1 + \mathscr{G}_t\,\mathbf{u}_{1,t}. \tag{2.32}
$$

We can, of course, continue advancing time steps as long as we have experimental data available:

$$
\mathbf{y}_{s-1,t+s-2} = \mathscr{O}_t\,\mathbf{x}_{s-1} + \mathscr{G}_t\,\mathbf{u}_{s-1,t+s-2}. \tag{2.33}
$$

These equations above can be expressed more compactly as:

$$\left[\mathbf{y}_{0,t-1}\ \mathbf{y}_{1,t}\ \cdots\ \mathbf{y}_{s-1,t+s-2}\right] = \mathscr{O}_t\left[\mathbf{x}_0\ \mathbf{x}_1\ \cdots\ \mathbf{x}_{s-1}\right] + \mathscr{G}_t\left[\mathbf{u}_{0,t-1}\ \mathbf{u}_{1,t}\ \cdots\ \mathbf{u}_{s-1,s+t-2}\right]. \tag{2.34}$$

This way, we can define the following matrices:

$$Y_{t,s} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{s-1} \\ y_1 & y_2 & \cdots & y_s \\ y_2 & y_3 & \cdots & y_{s+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{t-2} & y_{t-1} & \cdots & y_{s+t-3} \\ y_{t-1} & y_t & \cdots & y_{s+t-2} \end{bmatrix}, \quad U_{t,s} = \begin{bmatrix} u_0 & u_1 & \cdots & u_{s-1} \\ u_1 & u_2 & \cdots & u_s \\ u_2 & u_3 & \cdots & u_{s+1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{t-2} & u_{t-1} & \cdots & u_{s+t-3} \\ u_{t-1} & u_t & \cdots & u_{s+t-2} \end{bmatrix}, \quad X_s = \left[\mathbf{x}_0\ \mathbf{x}_1\ \cdots\ \mathbf{x}_s\right], \tag{2.35}$$

which allow us to express Eq. (2.33) as:

$$Y_{t,s} = \mathscr{O}_t X_s + \mathscr{G}_t U_{t,s}. \tag{2.36}$$

Here, $Y_{t,s} \in \mathscr{R}^{t\times s}$ and $Y_{t,s} \in \mathscr{R}^{t\times s}$ are known, since they are just the experimental data stored in a particular matrix form, but $X_s \in \mathscr{R}^{n\times s}$, $\mathscr{O}_t \in \mathscr{R}^{t\times n}$ and $\mathscr{G}_t \in \mathscr{R}^{t\times t}$ are unknown.

The key idea behind subspace identification methods is to project the available experimental data into appropriate subspaces so that some terms of Eq. (2.36) vanish. We restrict our attention to a brief presentation of the multivariable output error state-space (MOESP) method for noise-free systems to provide a feeling of the type of reasoning and operations involved in subspace methods.

The first step is performing a LQ factorization of a composition of the $Y_{t,s}$ and $U_{t,s}$ matrices:

$$\begin{bmatrix} U_{t,s} \\ Y_{t,s} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{12} & L_{22} \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix}. \tag{2.37}$$

This factorization allows to find orthonormal basis for the column space of $U_{t,s}$ and its orthogonal complement $U_{t,s}^{\perp}$, so that $Y_{t,s}$ can be expressed using these two bases as:

$$Y_{t,s} = L_{12}\, Q_1^T + L_{22}\, Q_2^T \tag{2.38}$$

We can multiply Eq. (2.37) by the projection matrix onto the row space of $U_{s,t}$:

$$L_{12}\, Q_1^T\, \Pi_{U_{s,t}^{\perp}} + L_{22}\, Q_2^T\, \Pi_{U_{s,t}^{\perp}} = \mathscr{O}_t\, X_s\, \Pi_{U_{s,t}^{\perp}} + \mathscr{G}_t\, U_{t,s}\, \Pi_{U_{s,t}^{\perp}} \tag{2.39}$$

Now, due to orthogonality between the factors, the first and last terms of the equation vanish to zero, resulting:

$$L_{22}\, Q_2^T = \mathscr{O}_t\, X_s\, \Pi_{U_{s,t}^{\perp}} \tag{2.40}$$

Next, we multiply both terms by Q_2 and note that $Q_2^T Q_2 = I$ and $\Pi_{U_{s,t}^{\perp}} Q_2 = Q_2$, since $\Pi_{U_{s,t}}$ is precisely the projection matrix to the space spanned by Q_2, so we get:

$$L_{22} = \mathscr{O}_t \dot{X}_s Q_2, \tag{2.41}$$

At this point, given some technical conditions that guarantee that L_{22} is rank n, we can recover the extended observability matrix by performing a singular value decomposition on L_{22}:

$$L_{22} = \mathscr{O}_t X_s Q_2 = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = U_1 \Sigma_1 V_1^T. \tag{2.42}$$

This equation states that up to a similarity transformation:

$$\mathscr{O}_t = U_1 \Sigma_1^{1/2}. \tag{2.43}$$

The system matrices A and C can be obtained easily once we have $\mathscr{O}_t$. Let $\overline{\mathscr{O}_t}$ denote the first $t-1$ rows of the matrix $\mathscr{O}_t$, while $\underline{\mathscr{O}_t}$ represent the last $t-1$ rows. According to Eq. (2.29), the first row of $\mathscr{O}_t$ (for SISO systems, block for MIMO systems) provides C, while A can be recovered noting that $\underline{\mathscr{O}_t} = \overline{\mathscr{O}_t} A$, so

$$A = \overline{\mathscr{O}_t}^{\dagger} \underline{\mathscr{O}_t}. \tag{2.44}$$

Recovering B and D is more involved and requires using Eqs. (2.36) and (2.37) to set up a matrix equation which can be solved using least squares (see Tangirala for details), but is also possible.

Other subspace methods, such as N4SID and CVA, take different approaches in the projections they convey, but the general idea is the same as the MOESP algorithm. Moreover, they can be shown to fit in a general frame that characterize each of them as a particular choice of weight. Further details on subspace identification can be found in Tangirala (2014) and Johansson (1993).

2.1.5 Design of Experiments

The design of experiments is a topic of great practical importance for System Identification. It is usually the case that performing a System Identification experiment conveys some sort of additional costs besides the usual operation of the plant, usually in terms of production of substandard products or reduced plant yield. It is thus important to reflect on the future use of the model, so that we build the simpler possible that fits our purposes and analyze the experimental data acquired during the normal operation of the plant to ascertain if experiments are needed at all. However, normal operation is typically carried out under closed-loop control, which conveys

additional challenges to the System Identification task, such as correlation between inputs and disturbances and low excitation in both inputs and outputs.

The key concept in experiment design is the idea of an *informative experiment*. Informally, an informative experiment is one that allows to obtain a satisfactory model of the process by means of producing an informative data set. This depends on two conditions that are related to each other: On the one hand, the system model must be *identifiable*; on the other hand, the input must be *persistently exciting*.

Identifiability of a model is a concept that is related to the possibility of obtaining an estimate of the value of each of its different parameters. In Sect. 2.1.3, we briefly commented that the polynomials in the denominators of transfer functions in Eq. (2.19) should be monic. The reason behind this is that we should fix the value of the coefficient of the leading term of the denominator; otherwise, we would be able to identify only the ratio between the rest of the parameters of the model and that particular coefficient. There is, so, some constraints to be considered when defining the structure of a gray-box model for identification.

There is, however, another facet of identifiability of a model that is closely related to the characteristics of the input signal used in the corresponding experiment. To see this, we can think of the simplest—and close to absurd—case: We have a stable dynamic system that has reached its steady state, and we want to obtain a model simply observing its steady-state response without any input. Under these circumstances, the output of the system will remain constant at whatever value it had reached, thus providing no information whatsoever about what type of system we are facing. This extreme example clearly shows that the possibility of discerning between candidate models depends on the characteristics of the input signal applied during the data gathering step.

This leads to the concept of persistently exciting input which, under more technical terms, states that the input should contain a sufficiently rich frequency content in order to provide enough excitation to the system so that we can discern between different model alternatives. The more complex the model, the richer the frequency content required to carry out the experiment. A typical choice of input is pseudorandom binary signals (PRBSs), which are deterministic signals that fluctuate between two fixed values at pseudorandom time instants. Figure 2.2 includes a plot of pseudorandom input and its associated output for a well damped linear second-order system. Pseudorandom binary signals allow to choose their frequency content by selecting appropriate clock sampling rate for its generation, so that it can be focused on the frequency range where higher accuracy of the model is required. The main drawback of these type of signals is that the switching between two fixed values does not allow to detect nonlinearities of the system. For these systems, some variants of the PRBS have been proposed, such as multivalued PRBS and amplitude-modulated PRBS.

Another very important consideration is what should be considered an input when performing a System Identification experiment. Clearly, the manipulated variable of the system at hand should be considered an input; however, other signals which are commonly considered disturbances should also be considered as inputs if they are measurable. If there are known disturbances affecting the process that are not

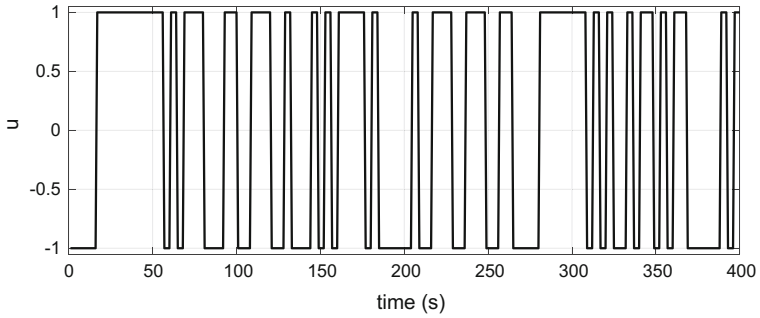


Fig. 2.2 Pseudorandom binary signal

measurable, the amplitude of the input signals should be such that these disturbances affect the process as mildly as possible.

2.2 Fuzzy Modeling

Fuzzy Modeling is an interesting alternative for situations where experimental data are scarce or expensive to obtain, or the system of interest is complex and highly nonlinear. At the core of fuzzy Modeling techniques is the concept of fuzzy set, an extension to the concept of classical sets that allows for members to have a *degree* of membership to the set, which will be presented in detail in the next section.

The most widely used fuzzy Modeling techniques are the rule-based fuzzy logic systems (RBFLS). This type of models are constructed using fuzzy if-then rules that relate the value of the input variables to the value of the outputs. These models can be built in a *white-box* approach, using the knowledge of experts in the process of interest, or in a more data-driven approach using available experimental data. These two alternatives in the construction of the system are related to the two major approaches for RBFLS: Mamdani and Takagi–Sugeno–Kang (TSK). Their difference lies in the way that the value of the output variable is expressed, being Mamdani more natural for the expert-based approach, while TSK is better suited for the data-driven alternative.

The most common approach of RBFLS is *flat* RBFLS, meaning that all the rules are defined over the same set of inputs and outputs. This method is appropriate for systems with a moderate number of variables and is particularly apt for the construction of rules based on experimental data. However, the number of possible rules grows exponentially with the number of variables, so systems having a large number of variables require ways to handle the additional complexity inherent in the high number of potential relations. Hierarchical RBFLS (Torra 2002) are a way to handle this complexity by Modeling the complete system as a composition of smaller subsystems. A Modeling methodology that makes use of this hierarchical approach

and is particularly fit for Modeling complex food processes is Fuzzy Cognitive Maps (FCM), which are presented in Sect. 2.2.3.

Neuro-fuzzy systems are fuzzy systems equivalent to RBFLS that are formulated as neural networks and provide the means to leverage all the existing tools for neural network training into data-driven construction of fuzzy logic systems. They are presented in Sect. 2.2.4.

Finally, some further comments on how to employ an hybrid approach to the construction of FLS are included in Sect. 2.2.5, presenting some ideas to fully leverage expert knowledge to provide a system structure and available experimental data to fine-tune the parameters of these models.

2.2.1 Introduction to Fuzzy Logic

The key idea of fuzzy logic is the extension of the concept of membership of an element to a set. In traditional crisp logic, an element either belongs or does not belong to a given set, and thus, its membership value is either 1 or 0, respectively. In fuzzy logic, this membership value is allowed to have any value in the [0, 1] interval, thus allowing *degrees of membership* of an element to a set.

A classical example here is to consider the *set of tall people*. We can define such a set as $T = \{x | x \geq 1.90\}$. This expresses that all the people whose height is 1.90 m or more belongs to the set. This definition implies that a person being 1.89 m does not belong to this *set of tall people* T; however, it makes intuitive sense to consider that even if it does not exactly meet the requirements defined to belong to the set, some degree of membership could be assigned to that element. Another person being 1.87 m could also be considered to somewhat belong to that set, but to a lesser degree than the 1.90 m one. Figure 2.3 shows a plot showing the definition of the crisp set T along with a fuzzy set T.

The set of crisp values that a given variable can have is called the *universe of discourse*. In the previous example, the universe of discourse can be considered to be a subset of the real numbers, say [0, 3]. Over this universe of discourse, we can also define other fuzzy sets, such as the *set of short people* or the *set of average height people* (Fig. 2.4).

The idea of assigning a partial degree of membership of an element to a set is appealing; however, it is of limited practical interest on its own. The real advantage of fuzzy logic comes into play when we try to establish relationships between variables when there is some sort of uncertainty present, be it lack of knowing the precise value of the variable or uncertainty on the precise relations existing between the variables. This is where fuzzy inference systems (FIS) come into play.

Let us suppose that we want to build a system that helps us in brewing a cup of tea. We can relate the temperature of water with how nice a cup of tea we can brew. If temperature is not high enough, then most of the flavor will not be appreciated; conversely, if temperature is too high, then maybe the flavor will not be as pleasant as if the temperature is just the appropriate one.

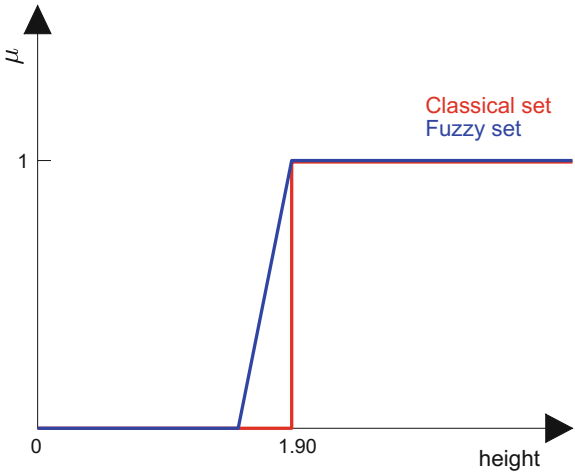


Fig. 2.3 Comparison of a classical and a fuzzy set for the variable *height*

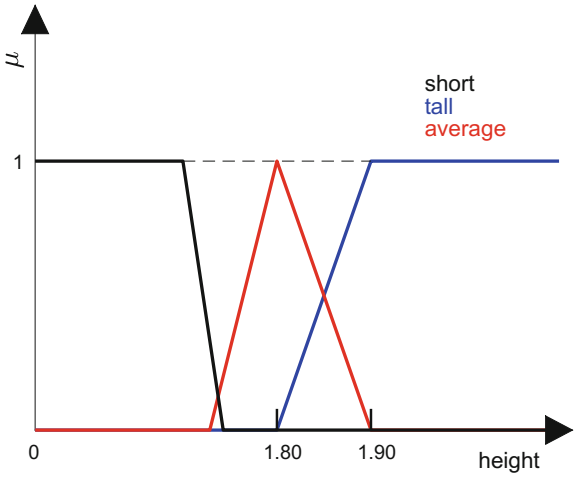


Fig. 2.4 Definition of fuzzy sets in the universe of discourse of the variable *height*

We can now consider the temperature of water as a fuzzy variable. In this case, the universe of discourse can be the [0, 100] interval, if we express the temperature in °C. Now, we can define the fuzzy sets of *low*, *moderate*, *high*, and *very high*.

Furthermore, we can define a variable called *tea flavor* and define some fuzzy sets on its universe of discourse, for instance, *unpleasant*, *nice*, and *excellent*. An important detail to notice here is that for this variable, although it is quite clear for a human what these terms mean, there is no clear candidate set to be the universe of discourse. We can, therefore, just choose an arbitrary set—say [1, 10]—and spread these fuzzy sets over this universe of discourse.

A natural way of expressing our knowledge about the relations between these variables is using if-then rules. Let us suppose that if the temperature is high, then

the tea will be excellent, while if it is either not high enough or a bit too high, it will be nice. Finally, if the temperature is way too low, then the tea will be unpleasant. This can be expressed using fuzzy rules as:

- If temperature is $VERYHIGH$, the tea flavor is *NICE*.
- If temperature is $HIGH$, then tea flavor is *EXCELLENT*.
- If temperature is *MODERATE*, the tea flavor is *NICE*.
- If temperature is LOW, the tea flavor is $UNPLEASANT$.

In these rules, the terms $HIGH$, $VERYHIGH$, *MODERATE*, LOW can be interpreted easily by a human and convey a explicit mathematical meaning that allow a computer to calculate what degree of flavor we are going to enjoy in the cup of tea.

We can further elaborate this model and consider that not only the temperature of the water is important, but the time allowed for the infusion also influences the final flavor of the tea. Now, we can consider how these two variables influence the flavor of the tea and express:

- If temperature is $VERYHIGH$ and time is $HIGH$, the tea flavor is $UNPLEASANT$.
- If temperature is $VERYHIGH$ and time is *MODERATE*, the tea flavor is *NICE*.
- If temperature is $VERYHIGH$ and time is LOW, the tea flavor is *NICE*.
- If temperature is $HIGH$ and time is $HIGH$, then tea flavor is *NICE*.
- If temperature is $HIGH$ and time is *MODERATE*, then tea flavor is *EXCELLENT*.
- If temperature is $HIGH$ and time is LOW, then tea flavor is *NICE*.
- If temperature is *MODERATE* and time is $HIGH$, the tea flavor is *NICE*.
- If temperature is *MODERATE* and time is *MODERATE*, the tea flavor is *NICE*.
- If temperature is *MODERATE* and time is LOW, the tea flavor is $UNPLEASANT$.
- If temperature is LOW and time is $HIGH$, the tea flavor is $UNPLEASANT$.
- If temperature is LOW and time is *MODERATE*, the tea flavor is $UNPLEASANT$.
- If temperature is LOW and time is LOW, the tea flavor is $UNPLEASANT$.

We can see that the number of rules required increases promptly with the number of variables considered. If we were to consider three or more input variables, we can see that the number of rules required to describe all the possible combination of values pretty quickly provides a system that is hard to manage. In these cases, the use of hierarchical fuzzy systems, such as the one presented in Sect. 2.2.3, is a good way of dealing with this extra complexity by means of systematically considering smaller subsystems and their relations.

2.2.2 *Types of Fuzzy Logic Systems: Mamdani Versus Takagi–Sugeno–Kang*

The most common type of fuzzy logic systems (FLS) is rule-based FLS, and they present the structure shown in the previous section: a collection of rules that relate the values of input and output variables. The FLS select the relevant rules out of the

pool of available ones and combines their outputs to provide the values of output variables for the scenario at hand. The purpose of this section is to look in some detail into the different steps that are required to carry out this computation, which can be split into three main stages:

1. Decide what rules are relevant for the scenario at hand, that is, which rules are best suited for the combination of input values considered.
2. Compute the values of the output variables provided by each rule.
3. Merge the proposals of each rule into a single final value for each output variable.

Similar to the notion of partial membership of an element to a set, the antecedent part of the rule may be considered to be not completely true or completely false, but true to some degree. This degree, in turn, should somehow affect the confidence we have that the outputs will show values similar to the ones prescribed by the rule. This degree of truth is usually called the *firing strength* of the rule.

The computation of this firing strength is easy for rules that involve a single-input variable, as we may use its degree of membership to the FS included in the antecedent of the rule as the firing strength. When we have more than one variable in the antecedent clause, such as when we considered both time and temperature in the tea brewing example, we need to combine the membership values of the inputs to obtain the firing strength of the rule.

Let us suppose that both inputs are connected using the logical relation OR: This means that if one input or the other has the value prescribed in the rule, then the rule applies. In this case, it is clear that we can just split the rule in two: one considering the first input and one considering the second input and just compute the firing strength of each rule employing the membership value of the corresponding antecedent variable.

If the two inputs are connected by AND, then we cannot just simply split the rule, as we need to consider both inputs at the same time to check to what degree the rule is true in the given scenario. For Boolean logic, an AND operation implies multiplying both membership values, as the result will only be 1 if both inputs are. For fuzzy logic, there are two main ways of computing this combination: product and minimum. In both cases, the extreme membership values coincide with their crisp counterparts; for any other values, each method provides different results. However, both are used in practice, and the decision of which one to use is usually on practical considerations, such as the computation burden or some others—the reader can find further details in Mendel (2001).

Once that we have looked into how to compute the firing strength of the rules, we need to address how to effectively compute the final value of the different output variables present in the system. FLS almost always have more than one rule in their rule set for each output variable. How do we combine the prescriptions of each of these rules into a single value for each output variable?

The first point to consider is how is the information about the output provided in the rule. There are two major approaches to this: Mamdani FLS provide fuzzy sets as outputs of their rules, while Takagi–Sugeno–Kang (TSK) systems provide a crisp value as the output, typically given as some function of the crisp values of the inputs.

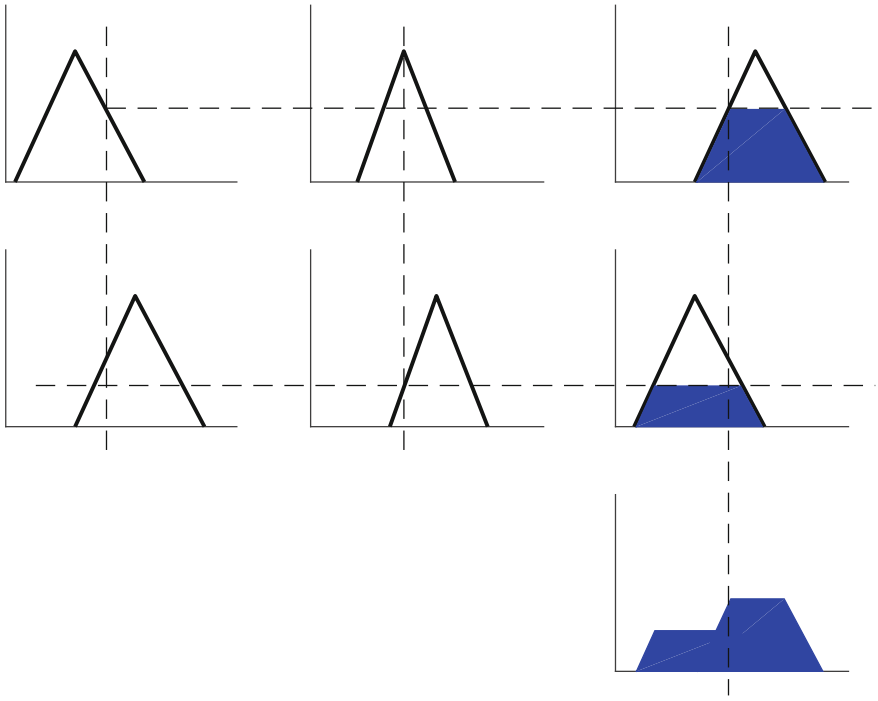


Fig. 2.5 Mamdani inference

For Mamdani FLS, the output of the fuzzy inference process is a FS, obtained by combining the FS provided by each of the different rules. This combination is typically carried out using sup-min combination and is graphically depicted in Figure 2.5. The first step is to compute the firing strength of each of the involved rules. Let us assume that we choose the *minimum* implementation of the AND operator. Then, the firing strength of the rule is the minimum of the membership values of the inputs. That is depicted in the left side of Fig. 2.5. The output FS is computed computing the minimum between the firing strength of the rule and the membership value to the output FS for each element of the universe of discourse. This can be visualized in the right side of Fig. 2.5.

The output FS is not yet a crisp value, so a further step called *defuzzification* needs to be carried out. There are several ways of accomplishing this defuzzification step: centroid (COG), mean of maxima, height, etc. These methods have varying levels of computational complexity and generally provide different crisp values for the same FS; the decision of which to use is, again, mostly based on practical considerations and preference of the system designer.

In turn, in TSK systems, each rule provides directly a crisp value for the output variable, usually given as a function of the crisp values of the inputs. In this type of FLS, no fuzzification step is required, since all the consequents are already crisp

numbers. The final value of a given output variable is computed simply as a weighted average of the values provided by each rule, using the firing strengths as the weights.

We have so far only considered rules that provide a single-output variable. What about rules that specify more than one output? It is easy to see that considering single-output rules do not suppose a lose of generality, as similar to what we did for rules where the inputs were combined using OR operators, we can simply separate each of the output variables into its own rule. These rules will simply have identical antecedents but different consequents. The interested reader can find more details in the excellent book Mendel (2001).

2.2.3 *Fuzzy Cognitive Maps*

Flat FLS are convenient for systems that have a limited number of involved variables or for systems where plenty of experimental data is available and a data-driven approach is sought. For scenarios where there are many different relevant variables and scarce experimental data, flat FLS are usually not the best option, as the number of rules grows very rapidly with the number of variables and systems become hard to manage very quickly. For these scenarios, hierarchical FLS are a better candidates for the job.

Fuzzy Cognitive Maps (FCM) are a type of hierarchical FLS that are convenient for systems with profuse variables, as they ease the elicitation of knowledge from experts allowing a systematic approach to this elicitation process. There are many different types of FCM proposed in the literature; the particular Modeling method presented in this section is based on simplified dynamic cognitive networks (sDCN).

The FCM model is a graph composed of nodes that represent the variables of the system and arcs that symbolize the relations between these variables. Formally, the model is defined as a tuple:

$$\mathbf{M} = \langle \mathbf{V}, \mathbf{A} \rangle, \tag{2.45}$$

where $\mathbf{V}$ designates the set of nodes and $\mathbf{A}$ stands for the set of arcs. Figure 2.6 shows a generic multi-input, single-output FCM with a single layer.

The following properties are defined for each node $v_i \in \mathbf{V}$ of the network:

- U_{v_i}: the universe of discourse of the node, that is, the set of possible crisp values of the variable.
- H_{v_i}: the collection of FS $L^k_{v_i}$ defined in U_{v_i}:

$$L^k_{v_i} = \{\langle x, \mu_{L^k_{v_i}}(x)\rangle \ : \ x \in U_{v_i}\}, \tag{2.46}$$

$$H_{v_i} = \{L^k_{v_i}, \ k = 1, 2, \ldots, K_i\}. \tag{2.47}$$

- $S_c(v_i)$: the crisp value of the node. Depending on the role of the node in the network and the computation process, this value can be known—if the node is an input node—or be the result of the inference process.

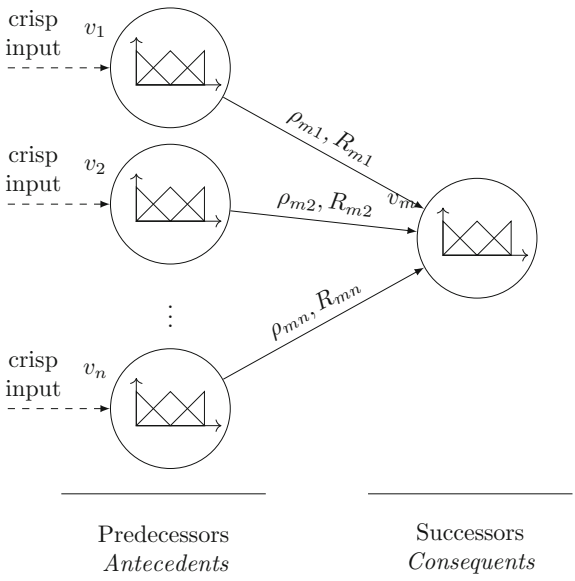


Fig. 2.6 Generic multi-input, single-output model for traditional, singleton FCM. A more complex model can be built by composition of this elementary structure

- $S_f(v_i)$: an array containing the degree of membership to each FS L_{v_i} of the crisp value of the node $S_c(v_i)$ (see Fig. 2.6).

$$S_f(v_i) = [\mu_{L^1_{v_i}}, \ldots, \mu_{L^{K_i}_{v_i}}]^T. \tag{2.48}$$

In turn, for each arc a_{ij} the following properties are defined:

- ρ_{ij}: strength of the relation between the nodes v_i and v_j.
- R_{ij}: causal relationship matrix. It defines the relationship between the FS of the predecessor and the successor nodes connected by the arc. The size of the matrix is $K_i \times K_j$, with K_i and K_j being the number of labels in H_{v_i} and H_{v_j}, respectively.

The properties of the arcs emulate the rules in a rule-based FLS, as they encode the relationships between the variables of the system. In the R_{ij} matrix, each row is associated with a FS defined in U_{v_i}, while each column is associated with a FS defined in U_{v_j}. Each entry of R_{ij} can be considered a rule relating the value of the predecessor (node v_j) with the successor (v_i), where the rule weight is given by the value of the entry times the weight associated with the arc. For example, let $L^k_{v_i}$ be triangular fuzzy sets, and let $\mathbf{q}_i = [q_i^1 \; q_i^2 \; \cdots \; q_i^{K_i}]^T$ denote the peaks of these fuzzy sets. Then, each entry is associated with a rule of the type:

`If` v_j `is` $L^b_{v_j}$ `Then` v_j `is` q_i^a, `with weight` $\rho_{ij} R_{ij}^{ab}$.

As an example, consider the following matrix:

$$R_{ij} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \tag{2.49}$$

The shape of the matrix denotes that three fuzzy sets have been defined in both U_{v_i} and U_{v_j}, and the associated rules are:

If v_j is $L^1_{v_j}$ Then v_i is q^3_i, with weight ω_{ij}.
If v_j is $L^2_{v_j}$ Then v_i is q^2_i, with weight ω_{ij}.
If v_j is $L^3_{v_j}$ Then v_i is q^1_i, with weight ω_{ij}.

The distribution of the nonzero entries in the relation matrices R_{ij} allows to easily encode certain predefined types of relations among variables, which eases the elicitation of knowledge from the experts. Positive and negative relations are considered, along with three types of relations:

- Bivalent relations: For a positive (negative) bivalent relation, a low value of the input variable tends to decrease (increase) the value of the output, and a high value of the input tends to increase (decrease) the value of the output. The following matrices R are examples of positive and negative, respectively, bivalent relations for a system with three defined labels per node:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

- Univalent relations: These are asymmetric relations, in the sense that some level of values of the predecessor exert some influence on the successor, while others exert none. Besides, the influence of the node always tends to increase (or decrease) the value of the successor.

  The following matrices are examples of univalent relations that always tend to decrease and increase, respectively, the value of a node:

$$\begin{bmatrix} 0 & 0.5 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 1 \end{bmatrix}.$$

  The arranging of the entries of the matrices in the first or last rows guarantees that the influence of the corresponding predecessor will be that of increasing (or decreasing) the value the node would have had if this node was not to exert influence.

  It should be noted that, in order to make sense, nodes exerting this influence on a successor are required not be the only predecessors, since that would either leave the value of the node undefined (if the incidence of the node is zero) or always having a extreme (maximum or minimum) value. However, this requirement is in

line with the intuition that it is natural to think of a relation that moves the value of a variable only when there is some other relation that establishes a reference value.

For the matrices presented above, higher values exert higher influence on the successor. The following two matrices illustrate analogous behavior, but the influence being exerted by lower values of the input:

$$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0.5 & 0 \end{bmatrix}.$$

- Sweet-spot relations: A certain value of the input provokes the maximum (minimum) value of the output, with higher and lower values of the input resulting in lower (higher) values of the output. The following matrices exemplify a minimum and maximum sweet-spot relations:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Level dependent and saturation effects can also be easily expressed using these matrices, as exemplified in the following two matrices:

$$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.50}$$

Here, the matrix on the left expresses a relation where the strength of the effect is magnified as the value of the input increases, while the matrix on the right shows a saturation for high values of the input that results in the same effect on the output as the one provoked by the moderate level.

The value of a successor node is computed according to the following steps:

1. The impact received by the node i is defined as:

$$\mathbf{w}_i = \sum_{j=1}^{n_i} \rho_{ij} \, R_{ij} S_f(v_j) = [w_i^1 \; w_i^2 \; \cdots \; w_i^{K_i}]^T \tag{2.51}$$

2. The computation of the crisp value $S_c(v_i)$ of the node is performed using a weighted average combination of the value of the peak of each fuzzy set using the impact received by the node:

$$S_c(v_i) = \frac{\sum_{k=1}^{K_i} w_i^k \; q_i^k}{\sum_{k=1}^{K_i} w_i^k}. \tag{2.52}$$

3. Finally, the fuzzy state vector of the node $S_f(v_i)$ is evaluated to capture the membership values for each FS as follows:

$$S_f(v_i) = [\mu_{L^1_{v_i}}(S_c(v_i))\ \mu_{L^2_{v_i}}(S_c(v_i))\ \cdots\ \mu_{L^{K_i}_{v_i}}(S_c(v_i))]^T. \tag{2.53}$$

Equation (2.52) performs the inference and defuzzification steps simultaneously and is very similar to the zero-order Takagi–Sugeno–Kang (TSK) model computation of the crisp value of the output of a set of rules.

It is important to remark the difference between the normalized impact $\bar{\mathbf{w}}$ and the fuzzy state $S_f(v_i)$. Even though the defuzzification of both arrays renders the same crisp value, both arrays are in general different.

The entries of $\bar{\mathbf{w}}$ will, in general, be less sparse than those of $S_f(v_i)$, since the former ones are the result of the impacts received by the node and, in general, may have several nonzero elements. In particular, if univalent relations are present, nonzero entries are expected in the first or last elements of $\bar{\mathbf{w}}$, which does not mean that the resulting crisp value of the node necessarily presents nonzero membership to the fuzzy associated with those elements.

For the propagation of the computations from node to node, it is important to use $S_f(v_i)$ instead of $\bar{\mathbf{w}}$, since if asymmetric or level-dependent relations are present, $\bar{\mathbf{w}}$ might activate spurious contributions that should not be activated according to the resulting value of the node. As an example, suppose that values for a node are given by:

$$\bar{\mathbf{w}} = [0.5\ 0\ 0.5], \quad S_f(v_i) = [0\ 1\ 0].$$

Then, for a relation matrix for a successor node to present a marked nonlinear behavior, such as:

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.54}$$

the impact given by $\bar{\mathbf{w}}$ would be [2.5 0 0.5], while $S_f(v_i)$ gives [0 1 0]. The different contributions based on each of these vectors is noticeable.

The main constraint of the methodology is the fact that rules can only have a single variable in the premise clause, practically supposing that the effect of the inputs is independent of each other, which may not always be true. However, this constraint is also a desired property of the approach, as it eases the curse of dimensionality by demanding that the influence of a single variable is considered at a time and limiting the number of possible rules in the system. This constraint simplifies greatly the work burden of the knowledge elicitation and provides visual representations of the models that domain experts, without an engineering background, can understand easily.

2.2.4 Neuro-Fuzzy Modeling

Neural networks are well-known tools widely employed in artificial intelligence and pattern recognition applications. Neural networks consist of a number of nodes that represent computation entities and arcs that connect the nodes and show the flow of information in the network. The major interest in these type of models lies in its ability to *learn* from the data; i.e., there exist algorithms that can adjust the values of the parameters of the network to minimize the discrepancy between the values predicted by the network and the existing experimental data. The most well-known and widely used algorithm is backpropagation.

As neural networks are a widespread technology, there are plenty of tools that allow to efficiently implement them in practice. However, one of the main drawbacks of neural networks is its black-box nature, meaning that it is usually difficult to provide a physical meaning to the structure of the resulting model, thus hindering the possibility of analyzing the inner computations to have some insight of possible improvement routes if the performance does not fulfill the expectations.

One of the main strengths of FLS resides precisely in its interpretability, as the fuzzy rules that compose the system can be understood by humans. Thus, the natural step of merging the relative merits of both methodologies gave rise to adaptive neuro-fuzzy inference systems (ANFIS), which are a type of adaptive neural network with a particular topology that are functionally equivalent to TSK FLS.

Figure 2.7 depicts the typical five-layer structure of an ANFIS, where each layer performs a specific step of the fuzzy reasoning algorithm. The nodes in the first layer represent linguistic labels, and the output of this nodes is the membership value of the input value to the corresponding FS. The type of membership function is fixed and specified by the designer, while the parameters that define the specific shape of the FS are adaptive, meaning that their value will be provided by the learning algorithm. Fig. 2.7 shows an ANFIS with two FS defined for each crisp input.

The second layer contains fixed nodes that perform the multiplication of the incoming signals and play an analogous role to the computation of the firing strength of a fuzzy rule. In effect, this multiplication can be viewed as performing an AND combination of the membership values passed to them using the multiplication rule.

The third layer normalizes the firing strengths so that they add up to one. This step can be interpreted quite intuitively, as we can think of these normalized firing strengths as a relative measure of how closely each rule fit the scenario defined by the current inputs.

The fourth layer simply computes the consequent part of each fuzzy rule multiplied by its corresponding normalized firing strength. If we are implementing a first-order TSK system—i.e., the consequents of the rules are linear combinations of the values of the inputs—the computation can be expressed as:

$$O_{4,i} = \bar{w}_i(p_i\, x + q_i\, y + r_i), \tag{2.55}$$

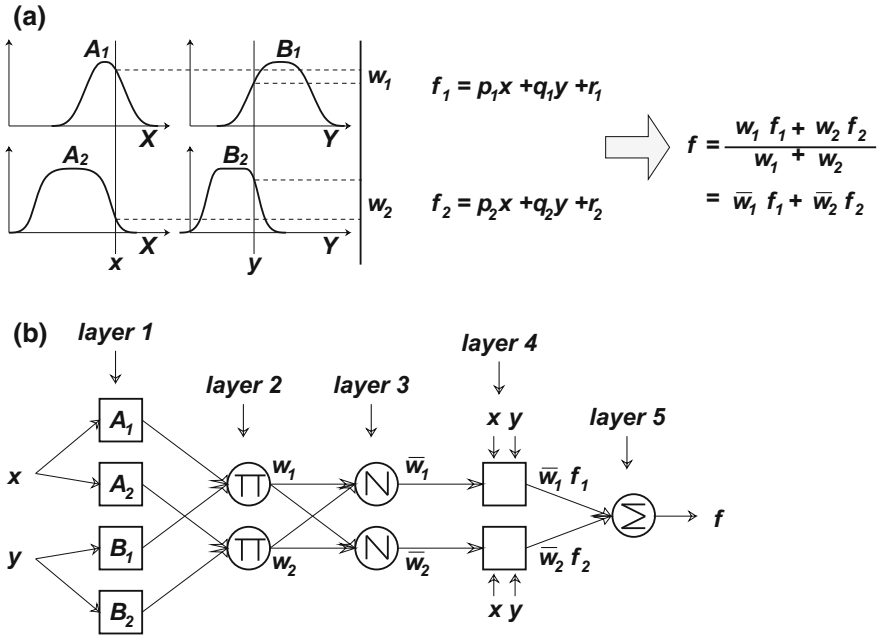


Fig. 2.7 Structure of a five-layer adaptive neuro-fuzzy inference system (ANFIS)

where $\bar{w}_i$ is the normalized firing strength and p_i, q_i, r_i are adaptive parameters called *consequent parameters*.

Finally, the fifth layer does not have tunable parameters and provides the crisp output value of the computation simply adding the signals coming from the previous layer.

The discussion above shows that ANFIS is a convenient formalism that allows to build TSK systems where the specific value of the parameters that define the membership functions of the antecedents and the weights of the linear functions of the consequents can be adjusted using experimental data.

2.2.5 *Identification of Fuzzy Models Parameters*

FLS have the interesting theoretical property of being universal approximators, meaning that any sufficiently smooth function can be approximated using them with arbitrary precision. This property provides the theoretical basis that supports the interest on this type of models.

If experimental data is available, there are different methods for constructing FIS using a data-driven approach. The first family of methods is based on performing a grid partition of the input space and using an optimization-based algorithm to adjust

the parameters of the partition and the consequents of the fuzzy rules. A second approach is clustering the experimental data points into homogeneous groups and defining rules associated with each group. In this method, the fuzzy sets defined in the input space are not shared by the rules, but different FS are defined for each rule. ANFIS systems, as commented above, are also a popular method of constructing FIS with a data-driven approach.

A well-known feature of Fuzzy Cognitive Maps that is of interest for the application of the proposed methodology is the capability of adjusting the parameters of the model based on experimental data. In order to study the inclusion of mechanism to allow the adjustment of the parameters of the system based on eventually available process data, we focus our attention to a model consisting of n inputs and one output. A more complex network can be built connecting different subnets with the same structure as the one analyzed.

For simplicity, we further suppose that all the nodes in the net have the same number of fuzzy labels defined in their universe of discourse. This assumption does not affect the generality of the analysis and simplifies the notation. Figure 2.6 shows the graph being analyzed.

The formula for the computation of the state of the consequent node according to the FCM formulation is:

$$S_c(y) = f(\sum_i \omega_i R_i S_f(u_i)),$$

with $f(\cdot)$ being the function that maps the impact received by the node $\mathbf{w}_y$ to its final crisp state $S_c(y)$.

Let us define:

$$\mathbf{p}_i = \omega_i R_i S_f(u_i) = [p_{i1}\ p_{i2}\ \cdots p_{il}] \tag{2.56}$$

as the impact exerted by node u_i on the node $\mathbf{y}$. Then, the total impact on y can be computed as:

$$\mathbf{w} = \sum_i \mathbf{p}_i. \tag{2.57}$$

Let μ_{ij} denote the membership grade of the i input to its j label, and F_{ij} the function that maps the crisp value of the node i to its membership value for the j label, i.e.,

$$S_f(u_i) = [\mu_{i1}\ \mu_{i2}\ \ldots\ \mu_{il}].$$

The first consideration is noting that we may split each node in the net into as many nodes as labels are defined in their universe of discourse and construct a net that computes the state of node y based on these disaggregated nodes. Figure 2.8 shows the graph that implements these calculations, along with Fig. 2.9, which further details the computation of p_i from $S_f(u_i)$ and the properties of the relation ω_i and R_i, defining r_{kj} as the elements of the matrix R_i.

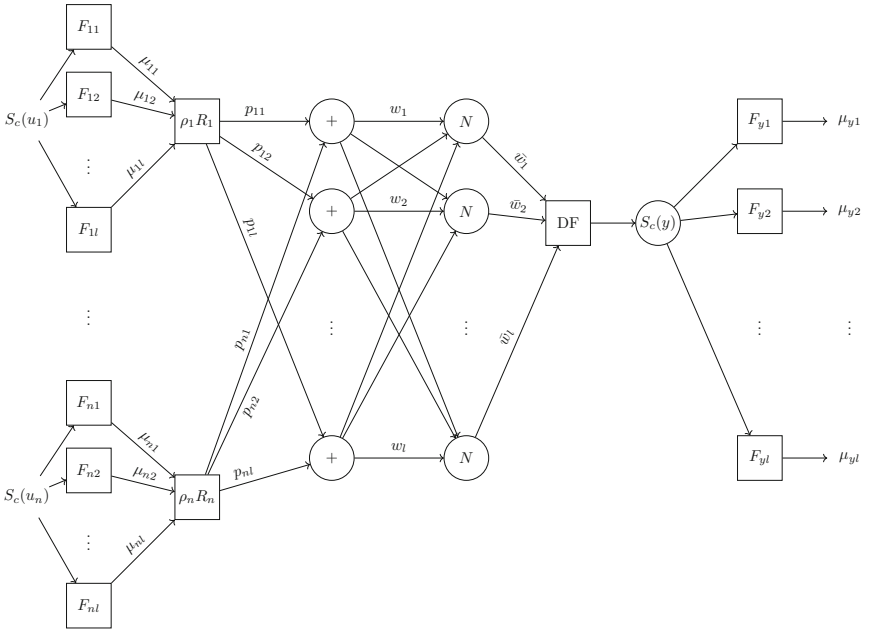


Fig. 2.8 Detailed expanded nodes graph. (© 2017 IEEE. Updated and reprinted, with permission, from Cano Marchal, P., García, J. G., and Ortega, J. G. Application of Fuzzy Cognitive Maps and Run-to-Run Control to a Decision Support System for Global Set-Point Determination. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(8):2256–2267)

In the graph in Fig. 2.8, p_{ik} represents the impact of node i to the k label of node y. Here, $i \in \{1, 2, \ldots, n\}$, with n being the number of input nodes, and $k \in \{1, 2, \ldots, l\}$, with l being the number of labels defined.

In turn, w_k represents the impact of *all* the input nodes to the k label of y. Again, $k \in \{1, 2, \ldots, l\}$.

The nodes N normalize the components of the impact, i.e.,

$$N(w_i) = \bar{w}_i = \frac{w_i}{\sum_{k=1}^{l} w_k}, \tag{2.58}$$

and play a similar role to the functions that normalize the firing strength of a rule to the sum of all rules' firing strength in layer 3 of the ANFIS model.

Finally, the node `DF` computes the crisp value of y based on the normalized impact and the kernels of the labels as defined in the previous section:

$$S_c(v_i) = \sum_{k=1}^{l} \bar{w}_k^i \; m_k^i. \tag{2.59}$$

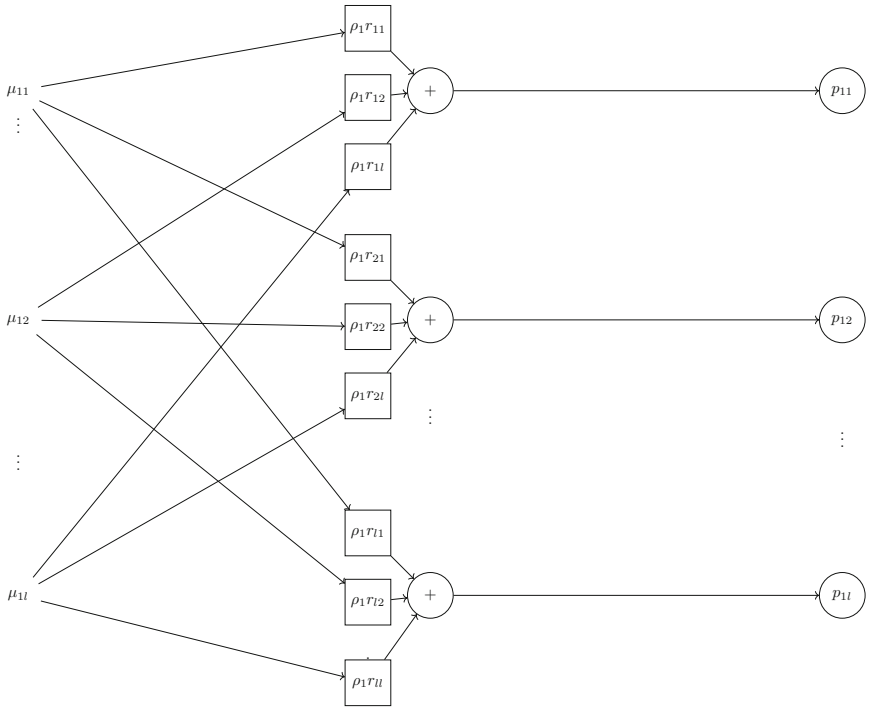


Fig. 2.9 Detailed computation of p_{il}. Updated from (Cano Marchal et al., 2017), ©2017 IEEE, used with permission

Once the crisp value of y is computed, it can be used to compute the fuzzy state of the node $S_f(y)$, which in turn may be the input to a subsequent node in a net.

The net in Fig. 2.8 shows a similar structure to the ANFIS system depicted in Fig. 2.7. This structure allows the entries of the relation matrices can be computed to fit eventually available data from the process using the backpropagation algorithm, with the only limitation being the piecewise continuity of the membership functions to the fuzzy labels Jang (1993).

This approach enables the incorporation of a data-driven approach to the building or refining of the models. The relevant variables along with the type of relations among the variables could be provided by experts, while the concrete values of the entries of the matrices R_{ij} along with the weights ρ_{ij} could be computed applying the backpropagation algorithm to the resulting graph.

References

Cano Marchal P, García J G, Ortega JG (2017) Application of fuzzy cognitive maps and run-to-run control to a decision support system for global set-point determination. IEEE Trans Syst Man Cybern: Syst 47(8):2256–2267

Jang J-S (1993) ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23(3):665–685

Johansson R (1993) Modelling and system identification. Prentice-Hall, Englewood Cliffs

Ljung L (1998) System identification: theory for the user, 2nd edn. Prentice Hall, Upper Saddle River

MATLAB System Identification Toolbox (2014) Matlab system identification toolbox

Mendel JM (2001) Uncertain rule-based fuzzy logic systems: introduction and new directions. Prentice Hall, Upper Saddle River

Tangirala AK (2014) Principles of system identification: theory and practice, 1 edn. CRC Press, Boca Raton

Torra V (2002) A review of the construction of hierarchical fuzzy systems. Int J Intell Syst 17(5):531–543

Chapter 3
Control of Lower-Level Dynamic Layers

As discussed in Chap. 1, the lower-level layer deals with the implementation details of the process, and the objective of the control layer is precisely to make sure that the low-level process variables actually attain and remain at the values prescribed by the higher-level controller of the plant.

In this layer, we can generally assume that we have appropriate online sensors for the variables, so that lack of data does not constitute an impediment for the control task. The only exception to this assumption will be considered in Sect. 3.2.2 when we talk about Run-to-run control, which is precisely used for situations in batch production where certain measurements are scarce.

Two types of processes are considered here: continuous operation processes, where there is an unending operation and materials are constantly being fed into and removed from our system; and batch operation processes, where the output product comes out in groups at certain instants of time and the operation can be divided into a sequence of different steps that take place one after the other, i.e., feeding of materials, mixing, heating, and emptying.

The control ideas that are applied to continuous process control can also be applied to the operation of batch processes, as set-point tracking and disturbance rejection are relevant problems for these processes as well. However, the iterative nature of batch processes may allow to *learn* something from one batch and incorporate that knowledge into the next batch. That is the idea behind *Iterative Learning Control*, which will be covered in Sect. 3.2.1.

P. Cano Marchal et al., *Production Planning, Modeling and Control of Food Industry Processes*, Advances in Industrial Control,
https://doi.org/10.1007/978-3-030-01373-8_3

3.1 Control of Continuous Processes

The control of the lower layer of continuous processes for food industry is very similar to the problems and challenges posed by process control in the chemical industry. Indeed, the process objectives of obtaining consistent quality products with high productivity and at minimum operating costs are shared for these industries, as are the existence of delays and measured and unmeasurable disturbances.

We begin presenting the PID controller which is reported to be responsible for more than 90% of the low-level control loops in industry. Then, the basic trade-off between reference tracking and disturbance rejection motivates the consideration of a more general control structure that allows more freedom to choose the response of the system to changes in the set-point. Thus, two-degree-of-freedom control is introduced and its advantages over simple error feedback controllers are discussed. Finally, the last section presents feedforward control as a technique to make use of information provided by additional sensors to counteract the effect of disturbances that may have a large influence in the output in the process.

3.1.1 Basic PID Control

Proportional-Integral-Derivative (PID) control is the bread and butter of control engineering. The PID controller is relatively simple to understand as its actions have a very clear intuitive interpretation, yet it presents a remarkable effectiveness for processes with benign dynamics.

A basic PID controller can be expressed as:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt}. \tag{3.1}$$

The denomination *Proportional-Integral-Derivative* controller is quite obvious in the light of Eq. (3.1), which shows that the control action is computed simply applying these operations to the error and just adding them. The first addend is called the *proportional* action and is used mainly to increase the speed of response of the controlled system; the second term is called the *integral* action and provides a means to completely eliminate the steady-state error for step inputs in the reference and disturbances; the third element is the *derivative* action and helps to reduce the oscillations that may arise from the inclusion of the integral action or due to relatively high values of the proportional gain K_P.

There are different procedures that help to select appropriate values for each of the parameters of the controller—K_P, K_I, and K_D—according to the degree of knowledge about the process that the control engineer may have. The main strategies can be classified into three major approaches: heuristic, experimental, and analytical.

Heuristic in-line tuning of the parameters is fairly common and can be carried out by expert operators for benign plants. The idea is just to perform an educated trial-and-error procedure with the controller parameters having in mind the influence that each of them have in the control signal. The typical procedure begins setting K_I and K_D to zero and selecting a value of K_P that provides roughly the required bandwidth without having excessive oscillations. Then, if the output is not too noisy, some derivative action can be added to dampen these oscillations. Once that the response is satisfactory, the integral gain (K_I) is increased until the steady-state error is removed at a convenient pace. The influence of the parameters will be revisited when the analytic design methods are presented, and some further insight into the influence of each of the control will be acquired.

Experimental methods, on the other hand, rely on the identification of some features of the plant to provide the values for the parameters of the controller. The autotuning algorithms that are usually available in commercial PID controller typically use this approach. The most famous method is the one proposed by Ziegler and Nichols; however, it is far from being the only one—or even the one to provide the best parameters—as many variations and alternatives have been proposed in the control literature. These methods all share the advantage of requiring a modest amount of knowledge of control system theory to tune a satisfactory controller and a moderate experimental burden. These methods typically require to perform a simple experiment on the plant in order to identify some specific parameters upon which the value of the controller parameters is computed. Some of these methods will be presented in the next sections.

Finally, analytic methods take a more general approach to the controller design problem and are also applicable to more general structures than the one imposed by the PID. For these methods, the PID controller is seen as an integrator and two zeros to be placed so that certain performance and robustness specifications are met. The advantages of these methods are that they are very flexible and allow to design the controller based on constraints derived from the target closed-loop behavior of the system. The disadvantages are that they are more complex than the previous two approaches and require a better hold of the concepts and theory of automatic control and that a somewhat more detailed model of the process might be required, thus demanding a higher workload in the identification of the system.

3.1.1.1 Experimental Tuning Methods

The Ziegler–Nichols tuning rules are probably the best known tuning algorithms for undergraduate students having their first course on control engineering; their practical interest, however, is not as high as their academic popularity. These tuning rules typically provide poor transient responses to set-point changes, adequate disturbance rejection capability, and poor robustness margins. Yet they provide a quite good framework to develop and explain some ideas fundamental to control engineering and encompass all the aspects that an experimental tuning method have, so we will begin presenting them.

Ziegler and Nichols proposed two different methods. The first is based on the open-loop step response of the plant and requires the identification of the parameters of a first-order plus dead-time (FOPDT) model, so this method is only applicable to systems that exhibit a monotonic step response. The second method is based on the determination of the so-called *ultimate gain* and the *ultimate period*. The ultimate gain is the value of gain that provokes sustained oscillations of the closed-loop system, while the ultimate period is precisely the period of those oscillations. The experimental determination of these parameters requires to perform a closed-loop experiment with a proportional controller whose gain is progressively being turned up until the oscillations are sustained.

The Ziegler–Nichols tuning methods are sometimes regarded to as the *open-loop* and *closed-loop* methods, respectively. It is important to stress that *open loop* and *closed loop* refer to the conditions of the experiment that needs to be performed; the controllers are always implemented in a closed-loop control scheme, independently of the method used to obtain their parameters.

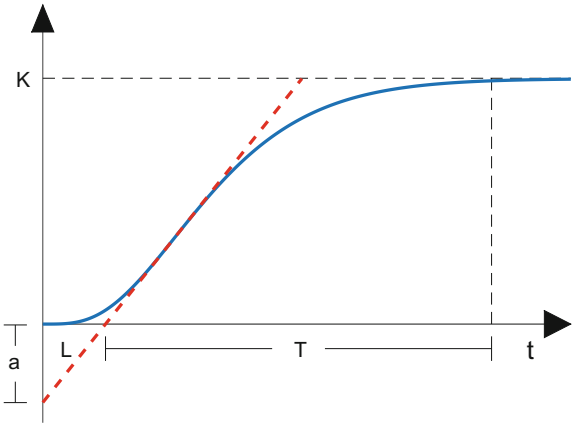
Figure 3.1a shows the sigmoidal step response and how to graphically determine the parameters of the FOPDT model. Table 3.1 includes the values of the parameters for the different types of controllers according to the experimental values of K, T, and L obtained. In turn, Fig. 3.1b shows how to graphically determine the ultimate period (T_{cr}), with the corresponding tuning parameters included in Table 3.2.

As commented before, the performance of the Ziegler–Nichols PID controllers is not completely satisfactory—although they are reported to still be widely used in industry. Another popular tuning rule that provides a slightly improved robustness and good disturbance rejection is the Cohen-Coon method. The method is also based on the identification of a FOPDT model, and the corresponding parameters are included in Table 3.3.

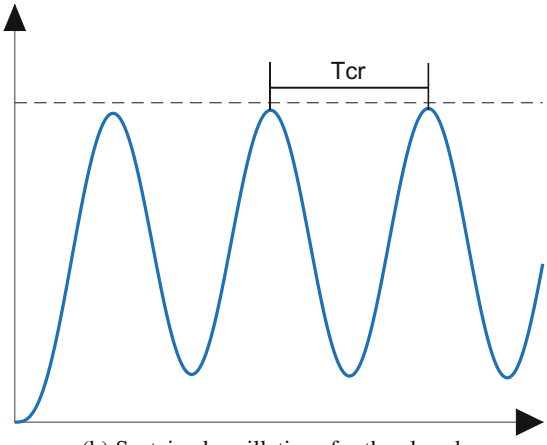
It is worth noting that both the Ziegler–Nichols and the Cohen-Coon provide a better behavior for load disturbance rejection than for set-point changes. To further illustrate this point, we can mention the tuning methods proposed by Chien, Hrones, and Reswick that explicitly provided different parameters depending on whether the designer assigns a greater importance to load disturbance rejection or stress a gentle transient response to changes in the set-point—the proposed parameters, again based on a FOPDT model, are included in Tables 3.4 and 3.5.

The reason for this explicit distinction is that there is a fundamental trade-off in error feedback controllers that does not allow to have good disturbance rejection and gentle set-point tracking at the same time. The fact that disturbance rejection is almost always more important for process control than set-point tracking, as the set-points change only sporadically, justifies the emphasis on disturbance rejection usually found in experimental tuning rules. In order to obtain good performance for both disturbance rejection and set-point tracking, we need a more general control structure: the so-called two-degree-of-freedom control that will be presented in Sect. 3.1.2.

Figure 3.2 shows the behavior of three PID controllers tuned using the open-loop Ziegler–Nichols (ZG), Cohen-Coon (CC), and 20% overshoot regulation Chien–Hrones–Reswick (CHR) methods for a second-order plant with a small delay. As



(a) Response of an step test used for the open loop Ziegler-Nichols tuning method.



(b) Sustained oscillations for the closed loop Ziegler-Nichols tuning method.

Fig. 3.1 Experimental plots for the Ziegler–Nichols tuning rules

Table 3.1 Open-loop test Ziegler–Nichols tuning rules

Controller	K_p	T_i	T_d
P	$\frac{1}{K}\frac{T}{t_m}$	∞	0
PI	$\frac{0.9}{K}\frac{T}{t_m}$	$3.33t_m$	0
PID	$\frac{1.2}{K}(\frac{T}{t_m})$	$2t_m$	$0.5t_m$

depicted in the figure, the response of the controllers is similar, yielding quite large overshoots but satisfactory disturbance rejection capabilities.

The weak robustness margins offered by the experimental tuning rules are due to an aggressive tuning based on a specification aiming for a quarter decay ratio for

Table 3.2 Closed loop test Ziegler–Nichols tuning rules

Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$T_{cr}/1.2$	0
PID	$0.6K_{cr}$	$0.5T_{cr}$	$0.125T_{cr}$

Table 3.3 Cohen-Coon tuning rules, where $r = \frac{t_m}{T}$

Controller	K_p	T_i	T_d
P	$\frac{1}{K\,r}\left(1+\frac{r}{3}\right)$	∞	0
PI	$\frac{1}{K\,r}\left(0.9+\frac{r}{12}\right)$	$t_m\,\frac{30+3r}{9+20r}$	0
PID	$\frac{1}{K\,r}\left(0.75+\frac{r}{4}\right)$	$t_m\,\frac{32+6r}{13+8r}$	$t_m\,\frac{4}{11+2r}$

Table 3.4 Chien–Hrones–Reswick tuning rules for disturbance rejection

	No overshoot				20% overshoot		
Controller	K_p	T_i	T_d	Controller	K_p	T_i	T_d
P	$0.3\,\frac{T}{K\,t_m}$	∞	0	P	$0.7\,\frac{T}{K\,t_m}$	∞	0
PI	$0.6\,\frac{T}{K\,t_m}$	$4t_m$	0	PI	$0.7\,\frac{T}{K\,t_m}$	$2.3t_m$	0
PID	$0.95\,\frac{T}{K\,t_m}$	$2.4t_m$	$0.42t_m$	PID	$1.2\,\frac{T}{K\,t_m}$	$2t_m$	$0.42t_m$

Table 3.5 Chien–Hrones–Reswick tuning rules for set-point tracking

	No overshoot				20% overshoot		
Controller	K_p	T_i	T_d	Controller	K_p	T_i	T_d
P	$0.3\,\frac{T}{K\,t_m}$	∞	0	P	$0.7\,\frac{T}{K\,t_m}$	∞	0
PI	$0.35\,\frac{T}{K\,t_m}$	$1.2t_m$	0	PI	$0.6\,\frac{T}{K\,t_m}$	t_m	0
PID	$0.6\,\frac{T}{K\,t_m}$	t_m	$0.5t_m$	PID	$0.95\,\frac{T}{K\,t_m}$	$1.4t_m$	$0.47t_m$

the amplitude of the effect of the disturbances. Next section presents some analytic methods for the tuning of PID controllers and introduces in greater detail the notion of robustness, its measures, and the associated trade-offs worth taking into account the controller design process.

3.1.1.2 Robustness and Frequency Analysis

A fundamental idea in feedback control is that the models that can be obtained from the real plant are accurate only up to a certain point and there is always a mismatch between the real process and our model of it. Furthermore, different phenomena such as wear, dirt accumulation, or fouling can alter the dynamics of the process, so

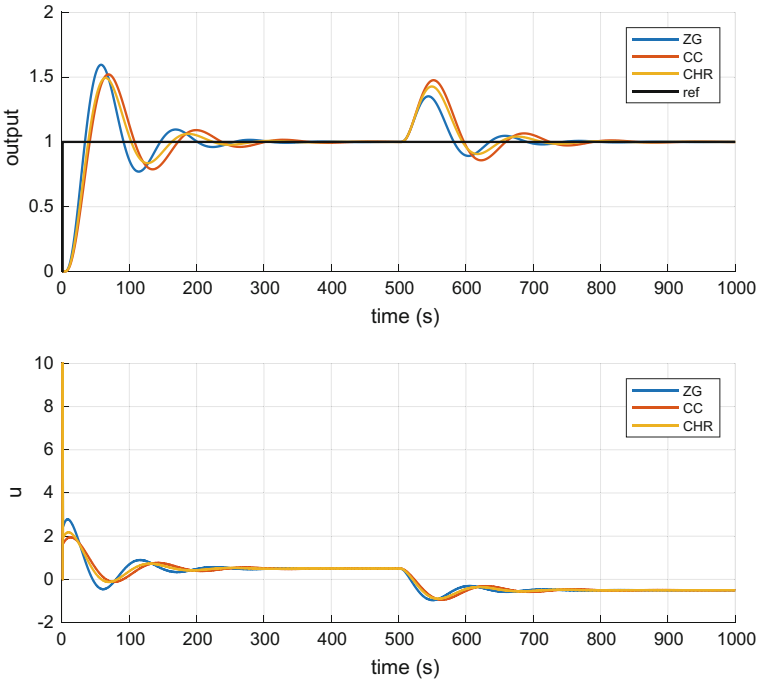


Fig. 3.2 Response of PID controllers tuned with the open-loop Ziegler–Nichols (ZG), Cohen-Coon (CC), and 20% overshoot regulation Chien–Hrones–Reswick methods (CHR). At $t = 500$, a load disturbance enters the system

having an accurate model at the identification stage does not necessarily guarantee having an accurate model some time into the normal operation of the plant.

With this idea in mind, it is natural to ask to what extent the controller that we design will perform properly in the real plant. This is precisely the notion behind robustness. Informally, robustness is concerned with what margin we have for the actual plant to be different from the model that we used to design the controller, usually called the *nominal plant*. The first concern is whether the controller will guarantee a stable closed loop system, and the second is whether the performance specifications that are achieved in the nominal plant will be also obtained in the actual plant. The idea is so appealing that specific methods, commonly known as *robust control* methods, have been developed to design controllers that guarantee explicit margins given a nominal plant and an uncertainty model. The interested reader can find more on this topic in the books (Skogestad and Postlethwaite 2005; Doyle et al. 2009; Zhou and Doyle 1997). Here, we will restrict our attention to some measures of stability robustness.

In order to introduce these stability measures, we need to present frequency analysis. A somewhat intuitive idea as to why frequency analysis is useful is provided by considering that any sufficiently smooth function can be, using Fourier series,

decomposed into a sum of sine function. If we know how each of those sine function is *treated* by our plant, we can know how a function composed of those elements will be treated, as we are assuming that our plant is linear and the superposition principle applies.

The response of a LTI system to sine function can be studied analyzing the value of the transfer function that models the system $G(s)$ for points in the imaginary axis, i.e., $s = j\omega$. Two very common plots that are used to analyze the frequency behavior of LTI system are the Bode and Nyquist plots. Both plots represent the complex number $G(j\omega)$. The Bode plot is composed of two graphs with a common abscissa that represents the frequency ω, usually in a logarithmic scale. The upper plot represents the magnitude of $G(j\omega)$, usually in decibels, while the lower plot represents its phase. In turn, the Nyquist plot uses a polar representation of $G(j\omega)$, so the abscissa represents the real part of $G(j\omega)$ and the ordinate represents its imaginary part. Each point of the Nyquist plot corresponds to a certain frequency ω. Figure 3.3 shows the Bode and Nyquist plots.

An interesting property of the frequency analysis is that it allows to infer properties of the closed-loop system analyzing the loop transfer function $L(s) = C(s)P(s)$. The theory support for the conditional stability margins is provided by the Nyquist theorem. The theorem states that for stable open loop plants, the Nyquist plot of the loop transfer function $L(s)$ must not have any encirclements of the point -1, usually known as the *critical point*. There is also a more general theorem applicable for unstable open-loop plants; however, since most of the plants found in food transformation processes are open-loop stable, we restrict our attention to this case.

Figure 3.3 shows Bode and Nyquist plots with a graphical depiction of the three major stability margins: the phase margin ϕ_m, the gain margin g_m, and the stability margin s_m. The phase margin ϕ_m measures how much we need to rotate the Nyquist plot, keeping the magnitude constant, to touch the critical point. It is related to what amount of extra delay the system can tolerate without losing the stability. The gain margin g_m specifies how much can the open-loop gain be increased while leaving the phase unchanged, so it quantifies the room for errors in the estimation of the static gain of the plant. Finally, s_m provides the shortest distance from the plot to the critical point and captures in a single number the robustness of the design. Typical desired values for these margins are 45–60° for ϕ_m, 2 for g_m, and 0.5 for s_m.

Similar to the discussion on the effect of the actions of the PID on the temporal response of the closed-loop system, we can provide some rules of thumb on the effect of the actions on the stability margins of the system. For this analysis, it is more convenient to express the PID controller using an alternative factorization of the parameters of the controller:

$$u(t) = K_P(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_D\frac{de(t)}{dt}. \tag{3.2}$$

A straightforward comparison of this equation with Eq. (3.1) shows that the relation between the parameters are simply $K_I = \frac{K_P}{T_i}$ and $K_D = K_P\, T_d$. Figure 3.4 shows the Bode diagrams of PI and PID controllers with different parameters and is useful

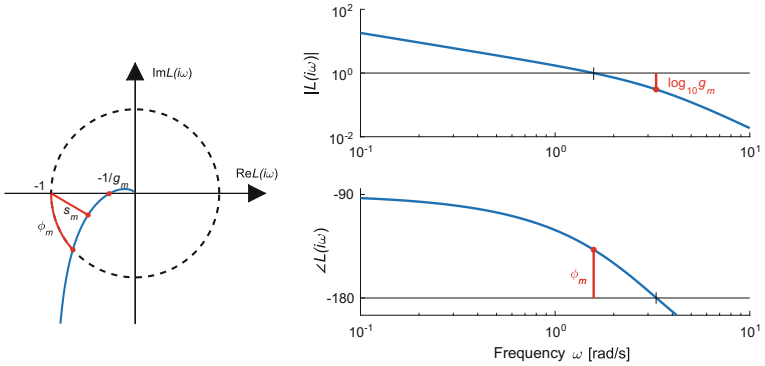


Fig. 3.3 Nyquist and Bode plots showing the phase, gain, and stability margins

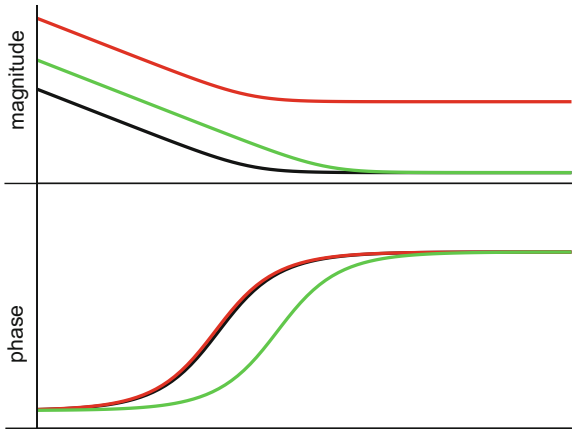
for visualizing the following discussion. The general idea is that a more aggressive tuning of the controller means poorer stability margins. An increase of the proportional gain K_P typically reduces all the margins, as it *pushes upward* the magnitude plot of the Bode diagram while leaving the phase plot unchanged. This means that ω_c—the frequency where the magnitude is 1—is moved to the left, where there is typically a lower value of phase, thus decreasing the phase margin ϕ_m. As the phase does not change, ω_{pc}—the frequency where the phase is $-180°$—remains constant; however, the magnitude $|G(j\omega_{gc})|$ is increased, so the gain margin g_m will also be reduced.

The inclusion of integral action implies including an integrator in the loop transfer function, which means that the phase at low frequencies will be reduced by 90°. The lower the value of T_I, the further to the right that the phase gain induced by the zero begins to counteract the effect of the integrator, thus acting negatively on the phase margin ϕ_m.

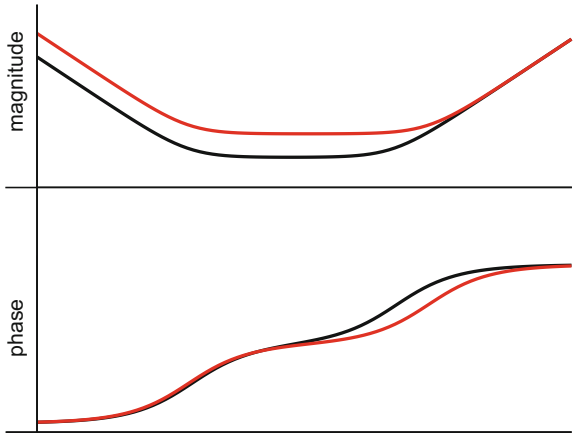
The inclusion of derivative action, on the other hand, provides another *boost* for the phase, thus potentially increasing the phase margin of the system. For this parameter, the higher T_d, the further to the right that the zero begins to influence the system, thus providing a smaller increase of the phase margin. The other side of the coin is that having the zero further to the right means that its associated magnitude increase begins later, thus providing a lower increase of the high frequency gain, which is a desirable feature.

3.1.1.3 Analytic Design Methods

The starting point for analytic methods is having a model of the plant. This model may have been obtained using a simple step response test, similar to the ones used for the experimental tuning methods, or it may arise from a more complex and accurate System Identification procedure like the ones presented in Chap. 2.



(a) PI controller.



(b) PID controller.

Fig. 3.4 Bode plots of a PI and a PID controller, with different parameters. An increase of the gain moves the magnitude upward while leaving the phase constant (plot (**a**), red line). A decrease of Ti moves the phase to the right (plot (**a**), green line). An increase in Td moves the phase to the right (plot (**b**), red line)

Many of the analytic design methods to be presented below provide controllers whose complexity is directly related to the complexity of the model. Since the number of parameters and the complexity of a PID is limited, so should be the models employed in these methods. Furthermore, other methods assume a certain specific type of model for their procedure to be applicable, typically FOPDT or SOPDT—second-order plus dead-time —models. In order to use any of these approaches, it is important to perform a model reduction procedure if our initially available model of the system is too complex. Having an initial complex model, however, is not harmful; on the contrary, having an accurate description of the plant helps being aware of the limitations of what is reasonable to expect of the closed-loop system, thus providing useful insight for making an educated decision when fixing its desired

performance specifications. Many of the methods employ these specifications as the design parameters, so choosing them wisely is capital for a successful design.

The fundamental common idea of analytic methods is to choose a target closed-loop behavior and compute the required parameters of the controller using the available model of the plant. The differences between the methods mostly lie in how they choose that target loop. Next, we present the pole placement, lambda tuning, and Internal Model Control methods.

Pole Placement. Pole placement is a simple method that contains most of the common features of the analytic design approach and is useful to illustrate the design trade-offs. The idea is that the closed-loop behavior of the system is strongly influenced by the location of the poles of $T = \frac{PC}{1+PC}$, so the parameters of the controller are chosen so that these poles lie in desired locations.

Let us start considering a first-order model of the plant and a PI controller:

$$P(s) = \frac{K}{\tau s + 1},$$
$$C(s) = K_P\left(1 + \frac{1}{T_i s}\right) = \frac{K_P(s + 1/T_i)}{s}.$$

Then, the closed-loop transfer function T is given by:

$$T(s) = \frac{KK_P(s + 1/T_i)}{s(\tau s + 1) + KK_P(s + 1/T_i)} = \frac{\frac{KK_P}{\tau T i}(T_i s + 1)}{s^2 + \frac{1+KK_P}{\tau}s + \frac{KK_P}{\tau T_i}}$$

We can equal the characteristic polynomial with a standard second-order characteristic polynomial expressed as $s^2 + 2\xi\omega_n s + \omega_n^2$ and get:

$$K_P = \frac{2\xi\omega_n\tau - 1}{K},$$
$$T_i = \frac{2\xi\omega_n\tau - 1}{\omega_n^2\tau}.$$

These expressions provide explicitly the parameters of the controller as functions of ω_n that controls the speed of the response and ξ that influences the shape of the response. Note, however, that we are not free to fully specify the closed-loop behavior because T has a zero in $-1/Ti$ that will also influence the shape of the response to changes in the reference. A large value of ω_n also improves load disturbance rejection; however, a limit on the realistically achievable ω_n is set both by physical limitations of the actuator, restricting the maximum value of K_P and the effect of unmodeled dynamics.

Since the loop transfer function is

$$L(s) = \frac{KK_P(s + 1/T_i)}{s(\tau s + 1)} = \frac{\frac{KK_P}{T_i}(T_i s + 1)}{s(\tau s + 1)}$$

we could use the parameter $T_i = \tau$ to cancel the pole of the plant, which would result in a first-order closed-loop transfer function, where we could choose K_P to place the closed-loop pole according to the required speed of response:

$$T(s) = \frac{1}{\frac{T_i}{KK_P}s + 1}.$$

This election provides a full control of the closed-loop transfer function to a step response without overshoot. However, a closer look at the transfer function from the load disturbances to the output shows that the pole of the plant is not canceled. This is not troublesome if this pole is fast; however, if it is a slow pole, then the rejection of load disturbances will be very sluggish.

The fact that the poles of the closed-loop transfer function could be located arbitrarily is due to the fact that the PI controller has two parameters, which is enough to freely choose the roots of a second-order polynomial. Since the PID controller has three parameters, it will allow to arbitrary choose the location of the roots of a third-order polynomial, thus allowing to freely choose the closed-loop poles for a second-order plant—note that the controller includes an integrator, thus increasing in one the order of the characteristic polynomial of the closed-loop system.

The most general plant whose closed-loop poles can be arbitrary placed, thus, can be expressed as

$$P(s) = \frac{b_1 s + b_2}{s^2 + a_1 s + a_2}$$

This plant can be controlled with a PID controller expressed as

$$C(s) = k + \frac{k_i}{s} + k_d s$$

to yield the following closed-loop transfer function

$$T(s) = \frac{(b_1 s + b_2)(k_d s^2 + ks + k_i)}{(s + \alpha\omega_n)(s^2 + 2\xi\omega_n s + \omega_n^2)}.$$

Here, the design parameters are α, ξ, and ω_n. Under this setup, the parameters of the controller are given by the following expressions:

$$k = \frac{a_2 b_2^2 - a_2 b_1 b_2(\alpha + 2\xi)\omega_n - (b_2 - a_1 b_1)(b_2(1 + 2\alpha\xi)\omega_n^2 + \alpha b_1 \omega_n^3)}{b_2^3 - b_1 b_2^2(\alpha + 2\xi)\omega_n + b_1^2 b_2(1 + 2\alpha\xi)\omega_n^2 - \alpha b_1^3 \omega_n^3}$$

$$k_i = \frac{(-a_1 b_1 b_2 + a_2 b_1^2 + b_2^2)\alpha\omega_n^3}{b_2^3 - b_1 b_2^2(\alpha + 2\xi)\omega_n + b_1^2 b_2(1 + 2\alpha\xi)\omega_n^2 - \alpha b_1^3 \omega_n^3}$$

$$k_d = \frac{-a_1 b_2^2 + a_2 b_1 b_2 + b_2^2(\alpha + 2\xi)\omega_n - b_1 b_2 \omega_n^2(1 + 2\alpha\xi) + b_1^2 \alpha\omega_n^3}{b_2^3 - b_1 b_2^2(\alpha + 2\xi)\omega_n + b_1^2 b_2(1 + 2\alpha\xi)\omega_n^2 - \alpha b_1^3 \omega_n^3}$$

Again, we can choose the poles, but the location of the zeros will be fixed. Two-degree-of-freedom control allows to maintain the good load disturbance rejection properties while improving the response to step commands and will be presented in Sect. 3.1.2.

Lambda Tuning. Lambda tuning is a common design method for process control and can be thought of as a special case of pole placement. The plant model used is FOPDT:

$$P(s) = \frac{K}{\tau s + 1} e^{-Ls}. \tag{3.3}$$

The idea of the method is to approximate the delay term and design a PI or PID controller—depending on the approximation used for e^{-Ls}—that cancels the plant pole and sets a time constant the closed-loop system given by λ, which is the design parameter of the method. The comment about the unsuitability of canceling slow poles due to their effect on load disturbance rejection should be considered for this method as well.

For the design of a PI controller, the delay is approximated using the first term in a Taylor series expansion: $e^{-Ls} \approx 1 - Ls$. Choosing $T_i = \tau$ and requiring the closed-loop pole to be at $-1/\lambda$ yield the tuning rule:

$$\begin{aligned} K_P &= \frac{1}{K} \frac{\tau}{L + \lambda}, \\ T_i &= \tau. \end{aligned}$$

The design of a PID controller uses the Padè approximation of the delay:

$$e^{-Ls} \approx \frac{1 - \frac{L}{2}s}{1 + \frac{L}{2}s}. \tag{3.4}$$

This way the plant has two poles that will be canceled with the zeros of the controller. Let the PID controller be expressed as:

$$C(s) = K_P \frac{(1 + sT_i)(1 + sT_d)}{T_i s},$$

then the tuning rules are given:

$$\begin{aligned} K_P &= \frac{1}{K} \frac{\tau}{\frac{L}{2} + \lambda}, \\ T_i &= \tau, \\ T_d &= \frac{L}{2}. \end{aligned}$$

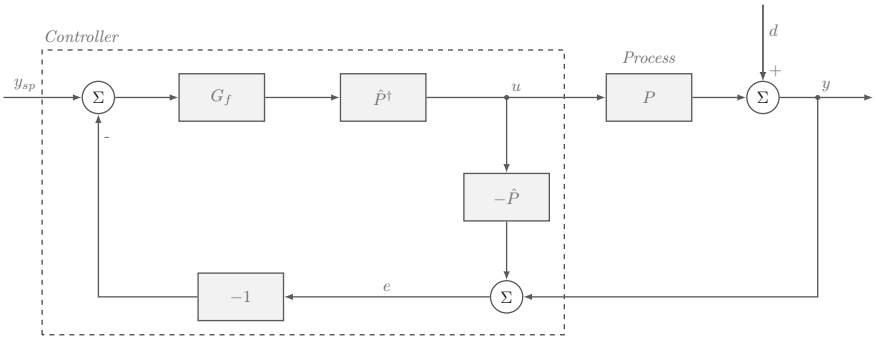


Fig. 3.5 Internal Model Control

Internal Model Control. Internal Model Control (IMC) is a control design approach that explicitly incorporates a model of the plant in the controller structure. As commented above, the complexity of the controller is directly related to the complexity of the model of the plant, so although this control design technique typically provides controllers whose structure is more complex than a PID, it can also yield PID-like controllers if appropriate approximations of the plant model are carried out.

Figure 3.5 depicts the block diagram for the IMC paradigm. The controller is composed of three different transfer functions: $\hat{P}$, $\hat{P}^\dagger$, and G_f. $\hat{P}$ is simply a model of the plant, with the overbar notation explicitly stating that the model will have some mismatch with the actual plant, while $\hat{P}^\dagger$ is an approximate inverse of $\hat{P}$. G_f, in turn, is a low-pass filter that allows to explicitly take into account the robustness of the design. The convenience of the inclusion of this filter is easily seen considering the frequency response of $\hat{P}^\dagger$. $\hat{P}$ is a model of a real physical plant, and as such will have very low values of gain in the high frequency band. On the other hand, $\hat{P}^\dagger$ is an approximate inverse of this model, so it will show high gain in these same high frequencies. In order to have a control signal u that does not have too much energy in these high frequencies, it is wise to filter those components from the input to $\hat{P}^\dagger$, thus the adequacy of including G_f.

Although innocent at first sight, the computation of an approximate inverse of a plant is a sensitive topic that requires some care in order to arrive at a solution that is both physically plausible and sufficiently accurate. A simple example to illustrate the difficulties of finding a plant inverse is to consider a FOPDT model. The time delay element cannot be exactly inverted, as it would require to foresee the future value of the input. The next step might be to consider a Padè approximation of the delay term and try to invert it. This approximation, given in Eq. (3.4), contains a zero in the right half-plane, so its inverse would be unstable, as it would contain a right-half-plane pole. It is not a good idea to have signals in the process that grow unbounded, so it is not desirable to consider an unstable inverse.

For nonminimum phase systems, the usual approach is to use a stable pole that is the specular image of the unstable zero. If we used this criterium for the Padè

approximation of the delay term, we would end up with a term that is simply 1, as the pole in the Padè approximation is precisely located at the specular reflection of the pole. For a FOPDT model expressed as Eq. (3.3), a commonly used inverse is

$$\hat{P}^{\dagger} = \frac{1}{K} \frac{1 + \tau s}{1 + \frac{\tau}{N} s}, \tag{3.5}$$

with N providing the frequency range where the approximation is accurate. As can be seen, no term is included to account for the delay. The reason to include the denominator term is to achieve a strictly proper transfer function that can be actually physically deployed.

For the IMC control paradigm, $\hat{P}^{\dagger}$ is in series with G_f, so a simpler inverse

$$\hat{P}^{\dagger} = \frac{1 + \tau s}{K} \tag{3.6}$$

can be considered, as the transfer function $G_f \, \hat{P}^{\dagger}$ will indeed be at least strictly proper if G_f is of order 1 or more.

The IMC control structure can be shown to be equivalent to a controller with the expression

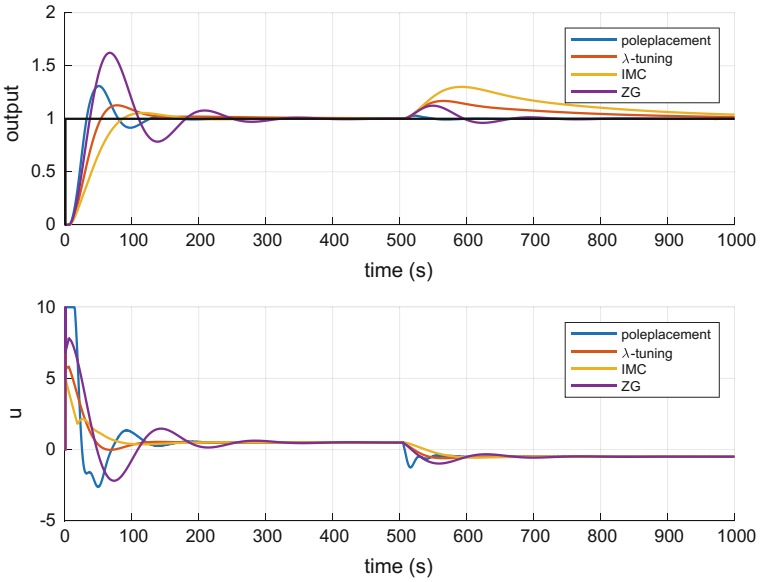
$$C(s) = \frac{G_f \, \hat{P}^{\dagger}}{1 - G_f \, \hat{P}^{\dagger} \, \hat{P}}. \tag{3.7}$$

In fact, if the plant $\hat{P}$ is a FOPDT model, $\hat{P}^{\dagger}$ is taken as Eq. (3.6) and G_f is chosen as a first-order filter

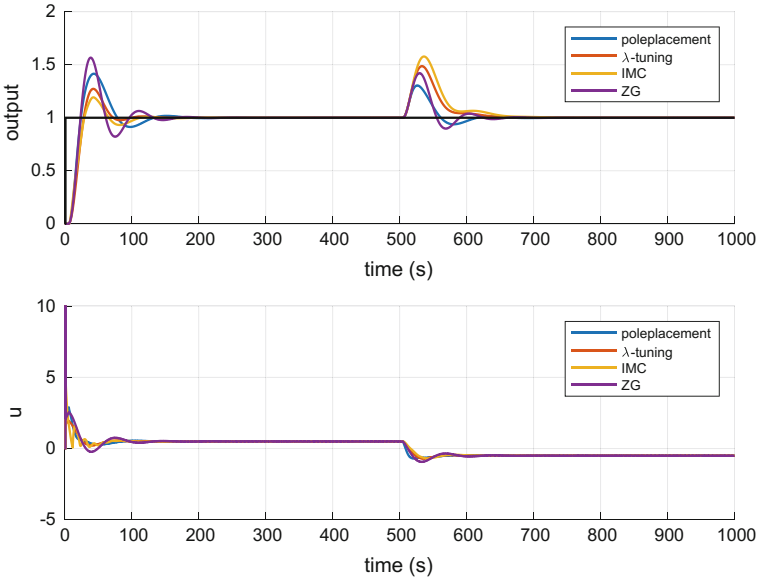
$$G_f = \frac{1}{1 + \tau_f s}, \tag{3.8}$$

the controllers provided by this method coincide with those proposed by the lambda tuning method if $\tau_f = \lambda$, and the delay term in $\hat{P}$ is approximated accordingly. However, this method provides a more general framework that can tackle more complex models of the plant at the expense of providing more complex controllers. Finally, this method includes an implicit cancelation of the poles of the plant, so poor rejection of load disturbances is expected if there are open-loop poles that are much slower than the closed-loop poles.

Figure 3.6 shows the response of different controllers tuned according to the rules presented in the section for a SOPDT plant. The upper plot shows the scenario when the plant contains a pole that is substantially slower than the time constant of the controlled system. As commented above and depicted in the plot, the cancelation of this pole induced by the lambda tuning and IMC methods provokes a very sluggish rejection of load disturbances. This effect is not present in Figure 3.6b, where the plant poles and the time constant of the controlled system are of the same order of magnitude. The reader can find more details of PID tuning techniques in the excellent books (Astrom and Hagglund 2006; Visioli 2010).



(a) The plant contains a pole that is much slower than the closed loop time constant.



(b) The plant does not contain any pole substantially slower than the closed loop time constant.

Fig. 3.6 Comparison of the step responses and load disturbance rejection for controller using pole placement, lambda tuning, Internal Model Control, and open-loop Ziegler–Nichols methods

3.1.1.4 Integrator Windup, Derivative Kick, and Bumpless Transition

The previous sections provide guidelines to design PID controllers for varying degrees of knowledge about the plant. There are, however, some practical details that need to be considered when actually implementing a PID controller in an industrial facility. The most important phenomena that should be born in mind in the implementation stage are known as integrator windup, derivative kick, and bumpless transition.

Integrator windup. A basic premise of the theory underlying the design of feedback controllers is that the plant is a linear time-invariant (LTI) system. Although this is rarely the case for real plants, it is a very good approximation for most systems under feedback control, since the control precisely deals with keeping the output of the system close to the set-point, thus maintaining the system close to an operating condition where a linearized model can be identified.

However, the fundamental limitation on the range of values actually achievable by the manipulated variable due to the limited size of real actuators conveys a nonlinear behavior that must be considered for the actual implementation of PID controllers.

The phenomenon is called *integrator windup* and is provoked by the increase of the integral term of the controller when the control signal is beyond the saturation limit. In these circumstances, the controller output is greater than the value that the actuator can actually apply to the physical plant, which is limited precisely by the saturation limit. While the error keeps the same sign, the integral action keeps increasing and so does the controller output; however, the actual actuation to the plant is constant, as it is limited by the saturation. When the error finally changes sign, the integral term starts to decrease and so does the controller output; however, the actual control signal applied by the actuator is kept constant until the controller output reaches a value that is lower than the saturation limit. This behavior introduces a delay between the intent of the controller to decrease the manipulated variable and its actual decrease, thus provoking a sluggish and oscillatory response of the controlled variable.

There are many different techniques proposed in the literature to counteract this behavior. Maybe the simplest to understand is limiting the increase of the integral term when the output is saturated. This can be easily achieved in the digital implementation of the controller simply including an appropriate *if-then* clause that checks if the controller output is to provide a value that is beyond the saturation limit of the actuator. If that were the case, then the integral accumulation term should not be further increased, but kept constant.

Figure 3.7a shows the behavior of a PI controller with and without an antiwindup scheme. The control signal reaches the saturation level quickly after the step in the reference; however, the controller with the antiwindup scheme returns to the linear regime earlier and yields a lower overshoot, noticeably improving the system response.

Derivative kick. Another implementation detail that is typically overlooked at the controller design stage is the so-called *derivative kick* phenomenon. This effect is fairly simple to understand and is provoked by the sudden change in the error that

is provoked by a change in the set-point, thus providing a very large instant value of the derivative of the error. If derivative action is used in the controller, this means that a very large spike is expected due to this high value of the derivative. This is not a desirable behavior, as it might damage the physical actuators of the plant. The solution to this problem is actually quite simple: It is enough to use the derivative of the output instead of the derivative of the error to drive the derivative action. Since the error is given by

$$e(t) = r(t) - y(t)$$

if $r(t)$ is constant, as is usually the case for food transformation processes, then the derivative of $e(t)$ is equal to the negative derivative of $y(t)$

$$\frac{de(t)}{dt} = \frac{dr(t)}{dt} - \frac{dy(t)}{dt} = -\frac{dy(t)}{dt}.$$

Figure 3.7b shows the derivative kick and the behavior of the system for a PID controller that uses the output to compute the derivative term of the controller. As depicted in the plot, the control signals show a significantly slower spike when the PID controller is used.

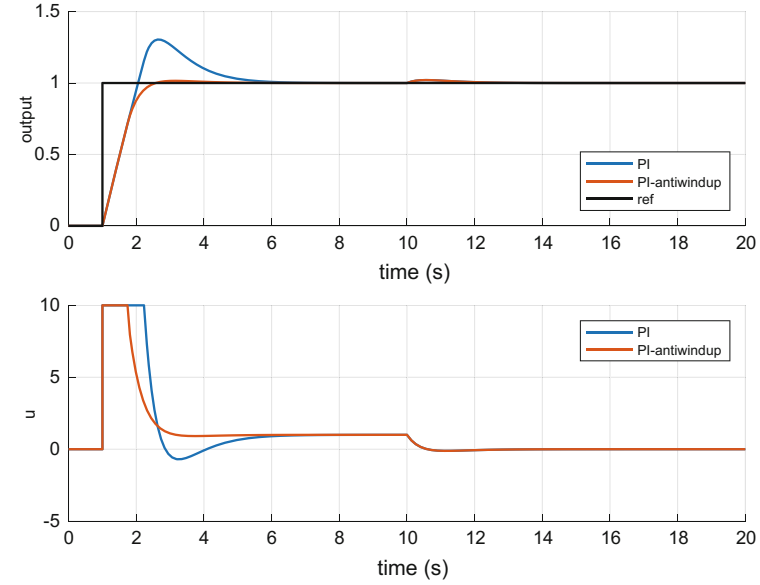
Bumpless transition. Real process plants are sometimes operated using the controllers in manual mode; that is, the control input value is decided directly by the operator. When the controller is changed from manual to automatic mode, some care must be taken so that an undesirable transient is not induced in the plant. The idea is that the output of the PID controller should match the value of the manipulated variable at the transition instant t_c. In order for this to be so, the integrator term should be recomputed just before the transition is made. Let us assume a PI controller for simplicity, then we have that the controller output is given by

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau = K_P e(t) + K_I I(t).$$

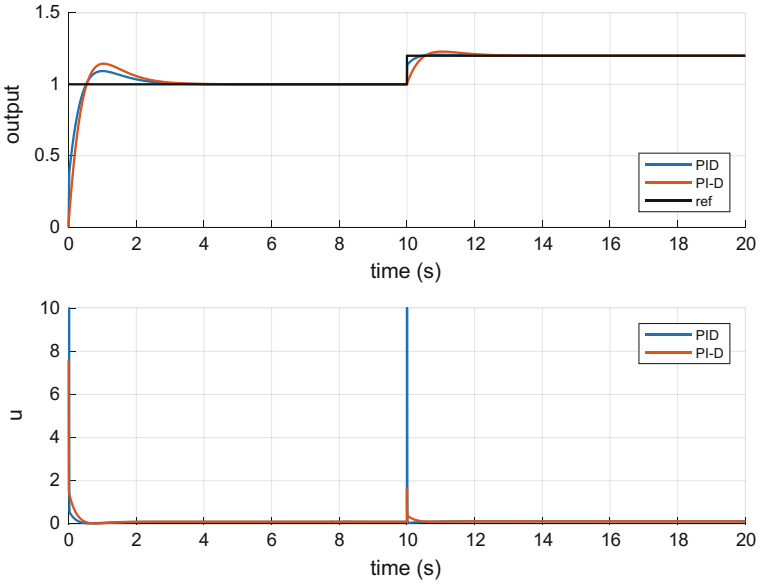
We can easily force the controller to provide an output $u(t_c) = u_c$ simply adjusting the term $I(t)$ which provides the value of the integral of the error

$$I_c = \frac{u_c - K_P e(t_c)}{K_I}.$$

This forces the output of the controller to match the value of the manipulated variable and assures that no transient due to the internal state of the controller will affect the plant. Figure 3.8 shows the output of a plant for a manual–automatic switch for a case when no precautions have been made in the commutation and when the presented bumpless commutation strategy is employed.



(a) Effect of an antiwindup scheme in the behavior on a PI controller.



(b) Effect of the implementation of a PI-D controller for the reduction of the derivative kick effect.

Fig. 3.7 Integrator windup and derivative kick effects in PI controllers

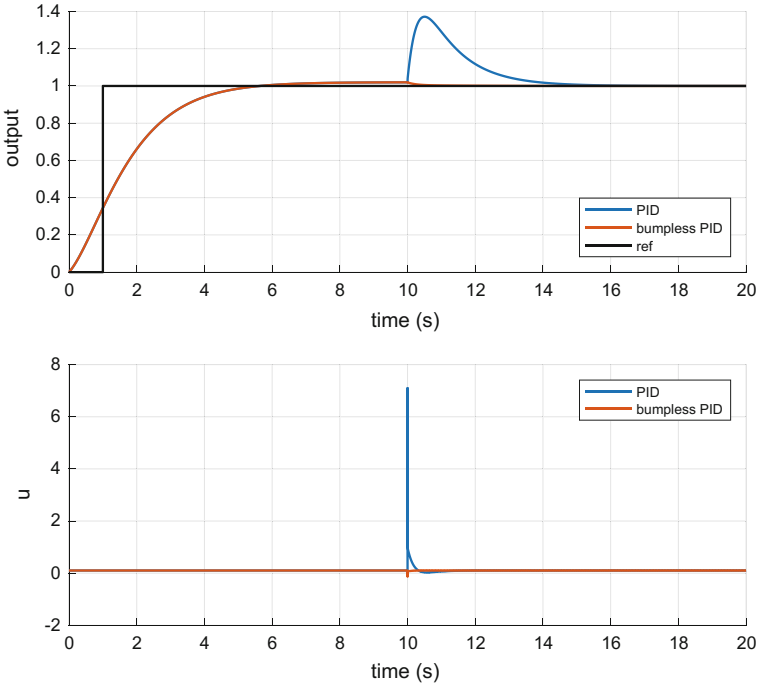


Fig. 3.8 Effect of the inclusion of a bumpless commutation strategy to a PI controller

3.1.1.5 Effect of Measurement Noise

The inclusion of feedback has many advantages; however, some tolls have to be paid in order to reap the benefits. The first is a purely economic one, as an investment in a sensor is required to properly close a feedback loop on the variable of interest of the plant. The second is inherent to the ability of having different poles in the closed-loop system than the ones in the plant. This allows to stabilize open-loop unstable plants, but also opens the door for the converse case: Open-loop stable can be made unstable if the controller is not properly designed. This concern motivated the robustness analysis presented in the previous sections.

The last issue provoked by the inclusion of feedback is connected with the fact that every measurement inherently includes some level of noise. Feedback control employs the measurements of the output variable to make decisions about the appropriate value of the manipulated variable, so the measurement noise will exert some influence on the behavior of the manipulated variable. It is thus important in the design of the controller to analyze the impact of the measurement noise on the control signal and assure that it is maintained within acceptable levels.

Frequency analysis is again the most useful tool to analyze the implications of this noise in the control signal. Measurement noise typically concentrates its energy

content in the high frequency bands, so it is important that the transfer function that relates the noise with the control signal does not have large gain values for the high frequency bands. This transfer function is

$$G_{un} = \frac{C}{1 + PC}. \tag{3.9}$$

Given the typical high-frequency roll-off present in real process plants, this transfer function can be approximated as $G_{un} \approx C$ for the high frequencies, which suggests that the high frequency gain of the controller should be limited. This is indeed the case and provides support for the necessity of filtering the derivative action in a PID controller.

Figure 3.9 shows the Bode diagrams of G_{un} and C for a plant $P =$ and a PID controller when the controller is and is not filtered. The time response plots to a simulated high-frequency noise show the convenience of including the filtering in the controller.

3.1.2 Two-Degree-of-Freedom Control

In continuous operations of food processes, the set-point is typically held constant for long periods of time and the main control objective is to reject the disturbances that may enter the process. Set-point tracking is, thus, not usual in these processes; batch processes, on the other hand, do typically require variables to follow certain trajectories until a steady condition is reached.

Several discussions in the previous section pointed out that there is a fundamental trade-off between having good transient response to changes in the set-point and rejecting load disturbances satisfactorily. In fact, the Chien, Hrones, and Reswick experimental tuning rule explicitly provided two different rules depending on whether set-point transient response or load disturbance rejection is given more importance.

The good news is that this trade-off applies to error feedback systems, but not to a more general control structure called two-degree-of-freedom control. The main idea is to filter the reference before the error signal is constructed, with this filter F providing an extra degree of freedom that can be used to improve the response of the system to set-point changes. There are cases where this control scheme cannot be used because the reference and the plant output are not available and only the error signal is accessible; however, these cases are extremely rare in food transformation processes. Another possible reason that may prevent from using two-degree-of-freedom control is that there is already available dedicated control equipment does not allow it. In any other case, there is no real reason not to use this control approach, as it requires no additional hardware, does not complicate too much the control system, and allows to decouple the set-point tracking and load disturbance rejection problems. When two-degree-of-freedom control is used, the first step in the design process is to select a feedback controller C that provides good load disturbance rejection capabilities and

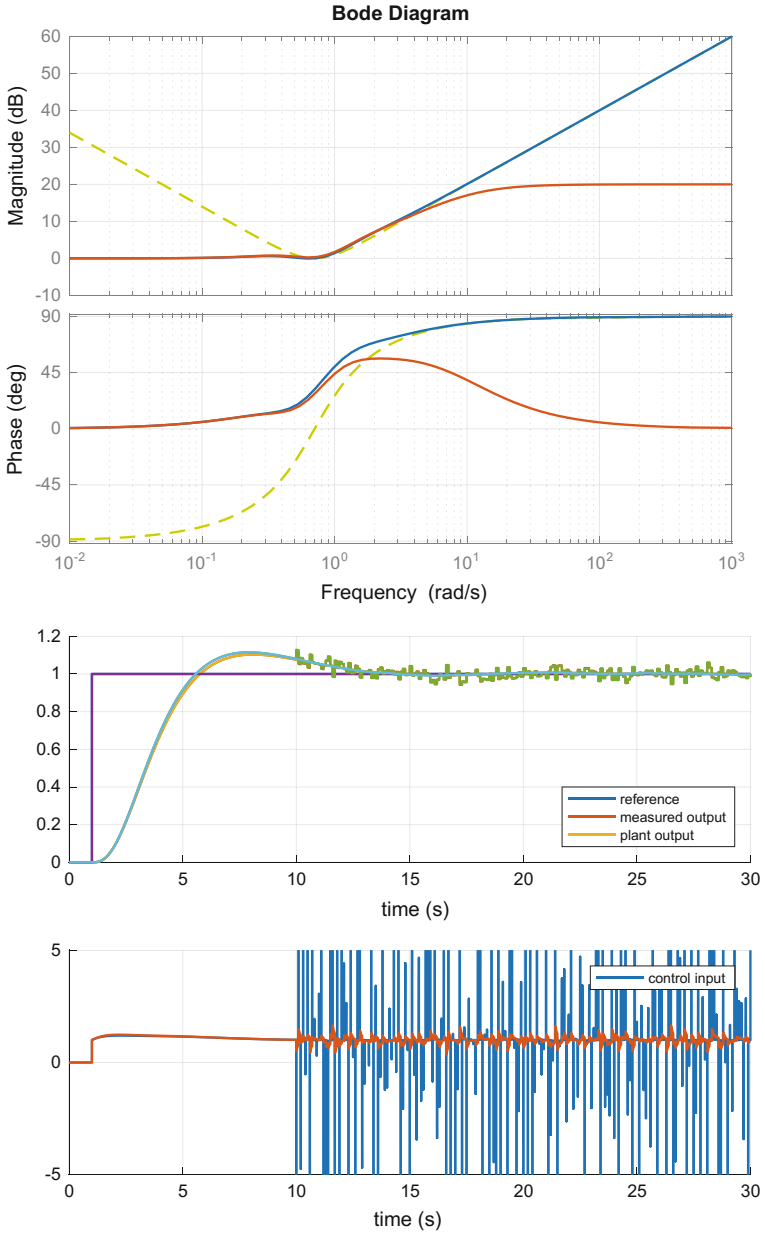


Fig. 3.9 Effect of the inclusion of high-frequency filtering in the control signal when measurement noise is considered

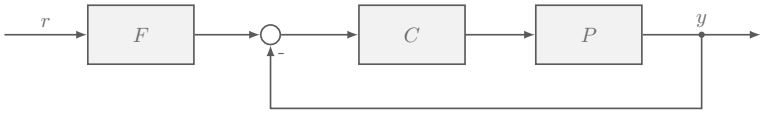


Fig. 3.10 Basic block diagram for the inclusion of two-degree-of-freedom control. The block F filters the reference before the error signal is computed

appropriate robustness margins. Then, the filter F is selected to provide appropriate response to set-point changes.

Figure 3.10 shows a block diagram implementing two-degree-of-freedom approach. Under this control scheme, the transfer function that relates the output with the reference is

$$G_{yr} = \frac{PCF}{1 + PC}.$$

Let the plant P, the controller C, and the filter F be given by

$$P(s) = \frac{n_P(s)}{d_P(s)};\ C(s) = \frac{n_C(s)}{d_C(s)};\ F(s) = \frac{n_F(s)}{d_F(s)},$$

then G_{yr} can be expressed as

$$G_{yr} = \frac{n_C\, n_P\, n_F}{(n_C\, n_P + d_C\, d_P) d_F}.$$

This representation of G_{yr} provides some insight into how to use the filter F to improve the transient response to set-point changes. The denominator of F—namely, d_F—appears as a factor of the denominator of G_{yr}, so it could be used to cancel the terms of the numerator of G_{yr} that are inconvenient for meeting the performance specifications. In a way, the filter F can be thought of as a tool to move the zeros of G_{yr} to some desired locations provided by n_F. There is, however, a subtle but important difference between the cancelation of factors of n_C and n_P: The transfer function of the controller C is chosen and implemented by the control engineer, so there is precise knowledge of the exact locations of its zeros. This means that the cancelations of these factors using d_F will be exact. On the other hand, n_P is only part of a model of a real plant, so the cancelation of its factors can only be carried out approximately, as the exact location of the plant zeros is not exactly known. It is important, thus, to have some care when trying to cancel slow process zeros, as some residual slow dynamics may appear in the resulting closed-loop transfer function if the cancelation is not precise enough.

A basic form of two-degree-of-freedom control is provided by the set-point weighing capability available in many commercial PID controllers. The control law implemented by this type of controllers can be expressed as

$$u(t) = K_P(br(t) - y(t)) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt},$$

where b is the set-point weighing parameter to be tuned. This configuration is equivalent to the block diagram shown in Fig. 3.10 having a controller C and a filter F given by

$$C(s) = \frac{K_D s^2 + K_P s + K_I}{s}, \quad F(s) = \frac{K_P\, b\, s + K_I}{K_D s^2 + K_P s + K_I}.$$

According to the discussion above, this effectively means that we can use b to choose the location of the closed-loop zero given by the controller. Using this controller for the pole placement design approach presented in Sect. 3.1.1.3 for the plant

$$P(s) = \frac{b_1\, s + b_2}{s^2 + a_1\, s + a_2}$$

yields a closed-loop transfer function given by

$$G_{yr}(s) = \frac{(b_1 s + b_2)(K_P\, b\, s + K_I)}{(s + \alpha\omega_n)(s^2 + 2\xi\omega_n s + \omega_n^2)}. \tag{3.10}$$

Here, the values of K_P, K_I, and K_D can be computed using the same formulas presented as functions of α, ξ, and ω_0; however, now there is freedom to locate one of the closed-loop poles of G_{yr} choosing b accordingly. Figure 3.11 shows the behavior of the system for different values of b. As can be seen, having lower values of b means less aggressive responses with less overshoot at the expense of a higher rise time.

A more general two-degree-of-freedom scheme is shown in Figure 3.12, and it includes a block that explicitly computes the steady-state value of u corresponding to r. The closed loop transfer function is given by

$$G_{yr} = \frac{P(CM_y + M_u)}{1 + PC} = M_y + \frac{PM_u - M_y}{1 + PC}. \tag{3.11}$$

This equation provides another point of view to design the transfer functions M_y and M_u. The second equality shows that G_{yr} can be made to behave similar to M_y if the fraction term is made small. This fraction can be made small either by having a large denominator, i.e., $1 + PC$ being *large* or by choosing M_u so that the numerator is small, i.e., having

$$PM_u \approx M_y. \tag{3.12}$$

Having a small denominator is achieved by choosing C appropriately and has been discussed extensively in the previous sections as is the key principle behind feedback control. Making the numerator of the fraction small is a different approach that requires a better knowledge of the behavior of P in order to include an approximate inverse of it in M_u. This approach is very tightly connected with the feedforward

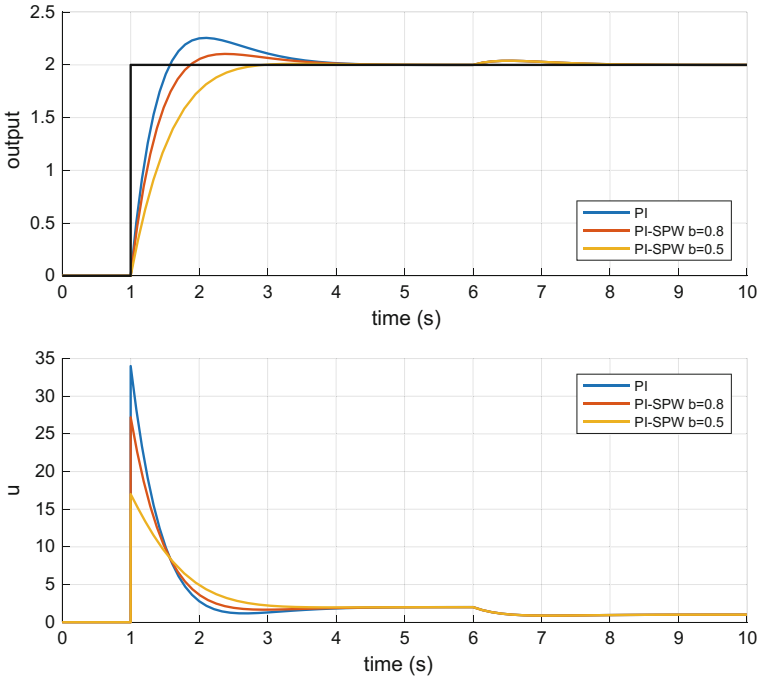


Fig. 3.11 Inclusion of set-point weighting in the transient response of PI controllers

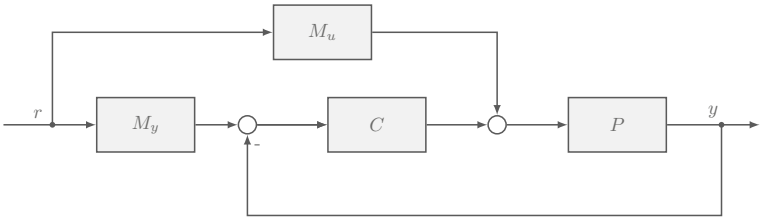


Fig. 3.12 Complete block diagram for the inclusion of two-degree-of-freedom control. The block M_y filters the reference before the error signal is computed, while the block M_u computes the steady-state value for u according to r

control approach to be presented in the next section, so further comments on how to choose appropriate values for M_u are delayed until the presentation of those ideas.

A final question of interest is to consider that, since it is typically desired that the output variable y reaches the reference value r *quickly*, what could be done to guarantee that this happens in the lowest possible time given the available actuators and plant dynamics. This type of questions is addressed by *optimal control* and will be covered in Chap. 4 when Model Predictive Control (MPC) is presented. For linear systems, a well-known result is that the optimal control strategy to have

a minimum time response is given by the so-called *bang–bang* controller, which commutes between the extreme values of the control signal according to the dynamics of the process model. If the plant controller is implemented using a digital computer, this type of controllers can be easily included; commercial PID controllers, on the other hand, do not typically provide this type of features.

3.1.3 Feedforward Control

Feedback controllers are the cornerstone of industrial control systems; however, they act only when there is already a measurable error in the output. If the dynamics of the system are such that the recovery from load disturbances is too slow, then it makes sense to wonder if there is some other techniques that can be used to improve the response of the system to these disturbances.

The idea of feedforward action is fairly simple, as it simply tries to account for the influence that we expect that some disturbance will have on the output and correct the value of the manipulated variable before this effect is actually seen on the output. In order to implement this strategy, we need to know the magnitude of the disturbance acting on the process; thus, additional sensors are required. The second requirement is that, since we need to anticipate our response, we must have a good idea of how the system is to behave due to the disturbance; thus, a reasonably good model of the effect of the disturbance on the process must also be available. Notice here that, although the model of this influence will never be perfect, the inclusion of feedback action also helps to counteract the influence of these Modeling errors, thus easing the precision requirements for this model.

Figure 3.13 depicts a feedforward control scheme where the plant P has been factored into two components

$$P = P_1 \; P_2$$

in order to provide more flexibility for Modeling the inclusion of the measurable load disturbance d in the control loop. Under this scheme, the closed-loop transfer function is given by

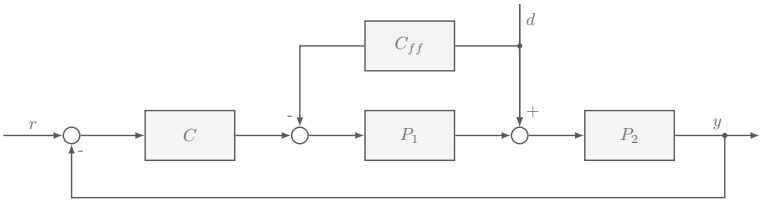


Fig. 3.13 Block diagram of the system including feedforward action. The plant P is factored into two components P_1 and P_2 to allow a more general model of the inclusion of the disturbance d. The block labeled C_{ff} is the feedforward controller responsible for computing the feedforward action u_{ff} to compensate the effect of the measurable disturbance d

$$G_{yr} = \frac{P_1 P_2 C}{1 + P_1 P_2 C} \tag{3.13}$$

and the transfer function from the disturbance to the output is given by

$$G_{yd} = \frac{P_2(1 - P_1 C_{ff})}{1 + P_1 P_2 C}. \tag{3.14}$$

Notice the similarity of this equation with Eq. (3.11). Analogously to what was commented in Sect. 3.1.2, there are two ways to make this transfer function *small*: forcing $1 + P_1 P_2 C$ to be large via designing C appropriately—i.e., the feedback control approach—and having

$$P_1 C_{ff} \approx 1 \tag{3.15}$$

so that the numerator is small. Equation (3.15) suggests that C_{ff} should be an inverse of G_1 in order to properly reject the influence of d. In Sect. 3.1.1.3, some comments were included about the convenience and difficulties of finding an inverse of a FOPDT plant that can be actually implemented in practice. In general, plants that contain nonminimum phase zeros are problematic, since their exact inverses are unstable systems. The most common approach for these cases is to use the mirror reflection of the unstable zero across the imaginary axis so that a stable inverse is obtained. Another common issue with the construction of approximate inverses is causality, as having a transfer function with a higher degree in the numerator than the denominator cannot be implemented. The usual solution to this problem is to include additional fast poles so that the approximate transfer function is strictly proper.

As commented in the previous section, feedforward control ideas can also be applied to two-degree-of-freedom controllers, as the concept is essentially the same: using knowledge about the value of an external signal—the measurable load disturbance d for feedforward control and the reference r for two-degree-of-freedom control—and a model of its influence on the output y to counteract its undesirable effect before it can be actually measured via precomputing a supplementary control action to be added to the feedback action.

The feedforward action is complementary to the feedback action, and it is wise to use them together if the investment in extra equipment makes economic sense supported the faster rejection of load disturbance and its influence on the quality of the produced goods.

3.2 Control of Batch Processes

Many of the challenges found in the automatic control of batch processes are common to those in continuous process control. Batch processes are also subject to plant uncertainty and are affected by disturbances that must be rejected, so feedback controllers with appropriate robustness margins must be designed and implemented. The time-

limited nature of batch processes induces a special relevance to the set-point tracking problem, as every batch will have a transient period where the system must evolve from its initial condition to the corresponding steady-state operation values, usually following a well-defined desired trajectory. Consequently, two-degree-of-freedom control schemes are usually required to assure that these trajectories are followed conveniently without compromising the load disturbance rejection capabilities of the controlled system.

There is, however, a relevant distinctive feature of these processes that is not shared with their continuous operation counterparts and that is the repetitive nature of batch processes. Usually, once a batch operation is finished, an identical one is prepared to be carried out. This provides the opportunity to examine the performance of the batch that was just finished and find some guidance about how to improve the next one; i.e., it offers the possibility to *learn* from past experiences in order to improve the behavior of future batches.

There are two main approaches to systematically use these past experiences: Iterative Learning Control and Run-to-run control. Iterative Learning Control is applicable when the outputs of interest of the process are easily measured, and we can have real-time measurements of their value, such as when dealing with temperatures or other standard process variables. On the other hand, Run-to-run control is applicable in situations where measurements are scarce and only available once that the batch is already finished, as is usually the case when some complex quality feature is the output variable of the process. The following sections detail these approaches.

3.2.1 Iterative Learning Control

Iterative Learning Control (ILC) is a control strategy that makes use of the repetitive nature of a task to improve the system performance in future batches using information about past iterations. The basic idea is to use a feedforward approach to modify the reference signal or the manipulated variable so that errors that were observed in previous iterations are compensated.

The analysis of ILC is helped by considering an additional dimension that accounts for which iteration we are currently on. Typically, this dimension is denoted by an iteration index k. The first time that the task is performed is assigned the index 0, using 1 for the first time that ILC is actually applied, since before this iteration there is no previous data that can be used to *learn* from.

In order to implement an ILC approach, the error and the applied control input signals of previous iterations must be stored. Since these controllers are implemented in digital computers, these signals need to be sampled and the data are only available at given time instants. This setup is very naturally formulated using discrete-time transfer functions, so those will be used in this section instead of the continuous time transfer functions used in the previous section.

Figure 3.14 depicts two common alternatives for the deployment of an ILC scheme. The upper configuration directly updates the control signal applied to the

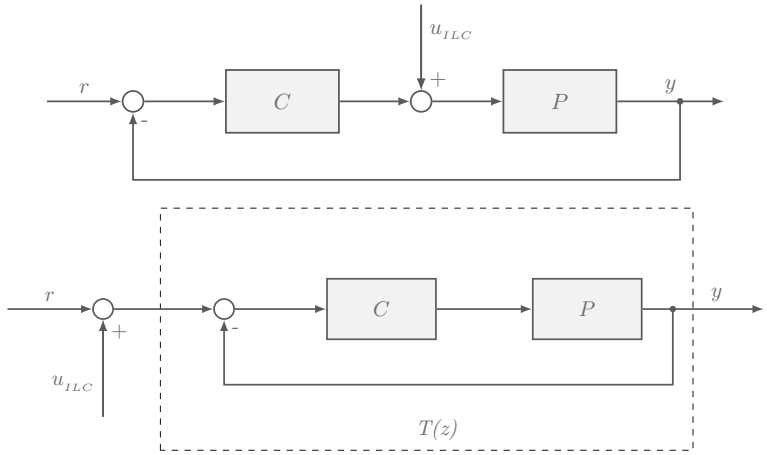


Fig. 3.14 Possible ILC schemes. The second is preferred, as it allows to be plugged into any configuration

plant, while the lower configuration employs the ILC input to modify the actual reference fed into the controller. Both configurations are equivalent; however, it is often more useful to consider the second approach as there is no need to have direct access to the manipulated variable. This allows to implement an ILC approach even in cases where a nonaccessible commercial low-level controller is used, since only the references that are passed to this controller need to be iteratively updated, and those are typically more accessible.

Let $P(z)$ designate the plant and $C(z)$ the feedback controller. Let $r(t)$ denote the reference, $y(t)$ the output, $u(t)$ the manipulated variable computed by the controller $C(z)$, and $u_k(t)$ the signal generated by the ILC scheme. This way, according to Fig. 3.14 we can write

$$y(t) = T(z)\big(r(t) + u_k(t)\big). \tag{3.16}$$

With this setup, the initial error is given by

$$e_0(t) = r(t) - y_0(t).$$

Note that $r(t)$ does not include an iteration subindex, as it is implied that the reference is kept constant during iterations. Indeed, the idea is to try to follow $r(t)$ better with each iteration using information about how we performed the previous iterations; in order for this to be so, $r(t)$ should not change between iterations. On the other hand, for the first iteration it is common to take $u_0(t) = 0$, as there is still no information available that may suggest using any other values.

A simple analysis of the values of $e_0(t)$ can provide some initial insight into how to use it for incrementally improving the tracking accuracy. The time instants where the signal $e_0(t)$ is positive mean that output did not reach $r(t)$, so $u_1(t)$ should be

increased; conversely, if $e_0(t)$ is negative, $u_1(t)$ should be decreased so that $y(t)$ approaches $r(t)$. This suggests using a simple proportional rule to update $u_1(t)$ for the next iteration

$$u_1(t) = u_0(t) + Le_0(t),$$

where L is the tuning parameter to be adjusted that determines how aggressively is $u_1(t)$ modified. Following this approach, the general updating rule is given by

$$u_{k+1}(t) = u_k(t) + Le_k(t). \tag{3.17}$$

Using this equation, we can derive the following relations for the error e_{k+1}:

$$\begin{aligned} e_{k+1}(t) &= r(t) - y_{k+1}(t) \\ &= r(t) - T(z)\, u_{k+1}(t) \\ &= r(t) - T(z)\Big(u_k(t) + L\, e_k(t)\Big) \\ &= r(t) - T(z)\, u_k(t) - T(z)\, L\, e_k(t) \\ &= r(t) - y_k(t) - T(z)\, L\, e_k(t) \\ &= e_k(t) - T(z)\, L\, e_k(t). \end{aligned}$$

The last line can be rewritten as

$$e_{k+1}(t) = (1 - T(z)\, L)e_k(t), \tag{3.18}$$

which provides a recurrence relation between e_{k+1} and e_k that can be used to derive the conditions for the convergence of the algorithm. If $\|1 - T(z)\, L\| < 1$, then the error is guaranteed to decrease from one iteration to the next and eventually converge to zero. If a model of $T(z)$ is known, it can be used to choose an appropriate value of L that guarantees the convergence of the algorithm.

Some valuable additional insight is provided by considering Eq. (3.18) in the frequency domain. Let us suppose that P is modeled using a continuous time transfer function $T(z)$; then, we can generalize L to also be a transfer function $L(z)$ instead of a simple static gain. The convergence condition then states that

$$\|1 - T(z)L(z)\| < 1. \tag{3.19}$$

Furthermore, a closer look at Eq. (3.18) suggests that choosing $L(z)$ as the inverse of $T(z)$ will provide perfect tracking in just one iteration. The difficulty of actually implementing this theoretical result in practice is that, as with feedforward control, $T(z)$ may only be inverted approximately. There are, however, two advantages of using an ILC scheme over just applying a feedforward filter. The first is that the tracking error will decrease with each iteration if $L(z)$ is chosen so that Eq. (3.18) is fulfilled even if $L(z)T(z) \neq 1$; i.e., there is no need to have a perfect inverse of $T(z)$ for the ILC approach to yield satisfactory results.

The second is that the transfer function $L(z)$ does not have to be causal, since the error signal is completely known before the next iteration is started, so *future* values are available for computation. This allows more freedom to compute the approximate inverse of $T(z)$ and helps to mitigate or completely remove the phase delay inherent in filtering operations. However, obtaining an exact inverse of the model $T(z)$ might still not be a good idea, since the typical very low gains in the high frequency band for real process plants would require very high gains of $L(z)$. The reason that advocates for limiting the gain of $L(z)$ in the high frequency band is the effect of measurement noise. Equation (3.17) shows that the high-frequency noise content in e_k would be amplified according to $L(z)$ when computing u_{k+1}.

A more general control update rule that explicitly takes into account the convenience of limiting the frequency band where the algorithm *learns* can be stated as

$$u_{k+1}(t) = u_k(t) + Q(z)L(z)e_k(t). \tag{3.20}$$

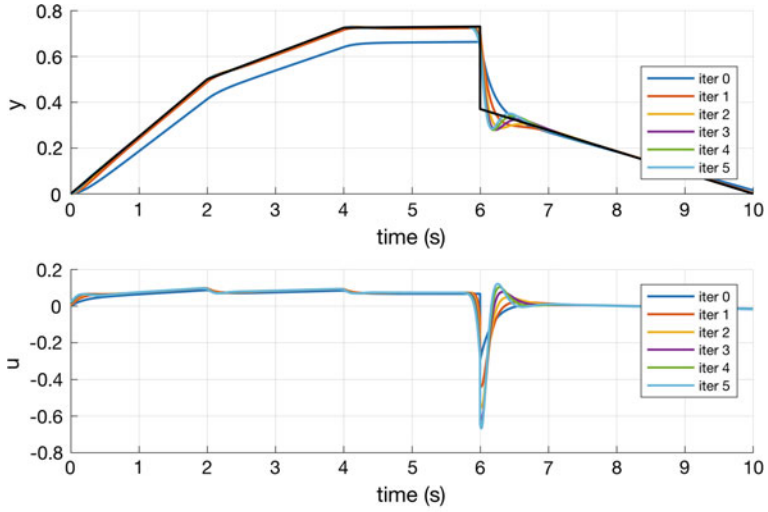
With this formulation, $L(z)$ is taken as an approximate inverse of $T(z)$ as faithful as possible, while $Q(z)$ is a low-pass filter that explicitly limits the influence of the high-frequency content of e_k in u_{k+1}.

Another topic of interest in ILC is the influence of random disturbances that do not act in a predictable manner in all the iterations. The algorithm cannot discern which parts of the error $e_k(t)$ were caused by the process dynamics and will be encountered in every iteration or were caused by a random disturbance that will not be replicated in subsequent iterations. Both types of errors are regarded as errors to be compensated by the algorithm, so the manipulated variable for the next iteration is adjusted accordingly to this available information. If this particular random disturbance does not appear in the next iterations, then the control input would be compensating for something that did not occur and would drive the output away from the reference. This discussion motivates the convenience of filtering these errors so that random disturbances do not affect too much the plant output.

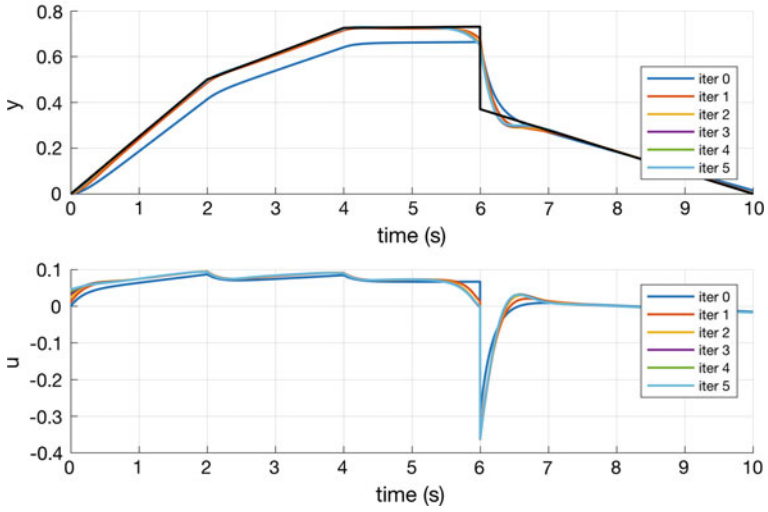
Two comments are relevant to this concern. On the one hand, the already commented systems that operate under ILC usually already have a PID-like controller (C) that takes care of the rejection of random load disturbances, so their effect on the output should already not be too high. On the other hand, there are extended ILC schemes that consider more than one iteration for the computation of $u_{k+1}(t)$, allowing a reduction of the effect of these random disturbances. A second-order ILC update scheme can be expressed as

$$u_{k+1}(t) = u_k(t) + Q(z)T(z)\Big(\omega\, e_k(t) + (1-\omega)\, e_{k-1}(t)\Big). \tag{3.21}$$

Here, the parameter $\omega \in [0, 1]$ governs how fast is new information considered in the algorithm. An extreme value of $\omega = 1$ is equivalent to just considering the last iteration, while a value of $\omega = 0.5$ entails averaging the errors of the two previous iterations. Using values lower than 0.5 is not advised, as it does not provide any further robustness and just assigns more weight to the *older* iteration, thus simply



(a) High bandwidth Q filter.



(b) Low bandwidth Q filter.

Fig. 3.15 Convergence of an ILC controller and the effect of the filter Q

adding some delay in the inclusion of information. The extreme example would be using a value of $\omega = 0$ that entirely disregards e_k and just uses e_{k-1} to compute u_{k+1}.

Figure 3.15 shows the behavior of a ILC controller and the influence of the filter Q for a plant controlled with just a proportional controller. As depicted in the figure, the initial response of the system shows a significant offset of the output that is completely removed even with the first iteration of the ILC controller. The upper

plot shows the behavior of the system with a filter Q with higher bandwidth than the one employed in the bottom figure. The major difference in the behavior of the controllers is highlighted with the abrupt step-like change in the reference at $t = 6$ that introduces high-frequency content in the error signal. As expected, the upper plots show a brisker response with a larger overshoot, while the lower plots show a more conservative behavior. The interested reader is pointed to Moore (2013) and Xu and Tan (2008) for further details on ILC.

3.2.2 Run-to-Run Control

The basic idea of Run-to-run control is exactly the same as that of ILC, i.e., using information gathered from previous iterations to improve the performance of future ones; however, the major difference is that the output variable can only be measured once that the batch is finished, so there is not an error signal available that provides the deviations of the output at each time instant of the progress of the batch, but just one final measurement to assess the whole iteration.

This lack of information means that the ILC approach of modifying the reference based on the actual error observed at each time instant cannot be implemented, as we only have one measurement to update the control signal for the whole batch. Furthermore, since the data of the evolution of the output during the batch are not available, dynamic models cannot be developed and the relevant process variables need to be related using static models exclusively.

These constraints are usually dealt with using the set-point of some process variables as the output of the controller; i.e., the Run-to-run controller provides updated set-points to variables such as the time duration of the batch, the set-point temperature of a reactor, or some other relevant variables whose effective value will in turn be controlled by a PID-like low-level controller.

In order to infer how to update these references, a solid notion of the influence of these variables on the output is required. Accordingly, Run-to-run controllers typically are model-based controllers that include an observer capable of estimating the disturbance acting on the system. The basic idea is to use the model of the system to compute the control action that allows to compensate for the estimated disturbance.

The best known Run-to-run controller is the exponentially weighted moving average (EWMA) controller. This controller assumes that the process is governed by a linear relation between the input and the output and considers a disturbance acting on the system according to the structure

$$y_k = \beta\, u_k + v_k. \tag{3.22}$$

Here, u_k denotes the process input, y_k stands for the process output, and v_k designates the disturbance acting on the k batch. Note that the variables do not depend on time, but only on the iteration index k, as there is only one value of those variables available per iteration. The process is modeled as

$$\hat{y}_k = b\, u_k + \hat{\nu}_k, \tag{3.23}$$

while the observer is given recursively by

$$\hat{\nu}_{k+1} = \omega\, \hat{\nu}_k + (1-\omega)(y_k - b\, u_k). \tag{3.24}$$

It is customary in the Run-to-run literature to denote as T the target of the system, that is, the desired value of the output. Adopting this notation and using Eqs. (3.23) and (3.24), the control action can be computed inverting the model of plant as

$$u_{k+1} = \frac{T - \hat{\nu}_{k+1}}{b} \tag{3.25}$$

Note that since the process model is static, we can perform an exact inversion of the plant without needing to take into account the comments presented in Sect. 3.1.1 regarding the computation of inverses of dynamic models.

An interesting theoretical property of this controller is that if the disturbance acting on the plant is an integrated moving average (IMA) process driven by a zero mean random variable ε_k with variance σ, given by

$$\nu_k = \nu_{k-1} - \theta\varepsilon_{k-1} + \varepsilon_k, \tag{3.26}$$

and the value of ω matches θ, then the observer provides the minimum mean square error estimate of the disturbance. However, the interest of this property is purely theoretical, as even in the case that the actual disturbance was effectively an IMA process, it is very unlikely that value of θ was known. In practice, ω is usually used as the tuning parameter for the controller, as it influences how fast the new estimation of the disturbance is used to modify u_{k+1}. The role of this parameter is analogous to the one played by the one in Eq. (3.21), and similar comments can be made; namely, low values of ω produce a more aggressive response of the controller that is useful when the uncertainty of the measurement is low, while lower values provide a gentler behavior and better performance if the uncertainty of the measurement is substantial.

Note that even though the models used in Run-to-run control are static, the inclusion of Eq. (3.24) causes the controller to be a dynamic system in the iteration dimension and, as such, introduces the possibility of having an unstable controlled system. The stability conditions for this controller can be derived casting the EWMA controller as an IMC controller and can be expressed as

$$0 < \omega\frac{\beta}{b} < 2. \tag{3.27}$$

This relation shows that overestimating the value of the process gain, i.e., having ratios $\beta/b < 1$, is conservative for the stability of the controller. It also supports the role of ω as a tuning parameter of the controller, as it also influences the stability of the controlled system, with lower values of ω being on the side of safety.

Different Run-to-run controllers can be implemented following the basic guideline shown in the EWMA controller simply modifying the process or disturbance model according to the situation at hand. The EWMA controller is well suited to compensate errors in the estimation of the model parameters and step-like disturbances. If the disturbance affecting the process is similar to a drift, then the observer can be modified to reflect it.

If a more complex and accurate process model is available, it is also quite straightforward to introduce it in a Run-to-run scheme, as all that is required is to update Eq. (3.23) and recompute Eq. (3.25) accordingly. Moreover, multivariable systems can also be dealt with in this general framework and will be presented in greater detail in Sect. 4.2.2. Further details about Run-to-run control can be found in Adivikolanu and Zafiriou (1997), Adivikolanu and Zafiriou (2000), Campbell et al. (2002), Good and Qin (2006).

References

Adivikolanu S, Zafiriou E (1997) Internal model control approach to run-to-run control for semiconductor manufacturing. In: American control conference, 1997. Proceedings of the 1997, vol 1, pp 145–149

Adivikolanu S, Zafiriou E (2000) Extensions and performance/robustness tradeoffs of the EWMA run-to-run controller by using the internal model control structure. IEEE Trans Electron Packag Manufact 23(1):56–68

Astrom KJ, Hagglund T (2006) Advanced PID control. ISA, Research Triangle Park

Campbell W, Firth S, Toprac A, Edgar T (2002) A comparison of run-to-run control algorithms. In: American control conference, 2002. Proceedings of the 2002, vol 3, pp 2150–2155

Doyle JC, Francis BA, Tannenbaum AR (2009) Feedback control theory. Dover Publications Inc., Mineola

Good R, Qin SJ (2006) On the stability of MIMO EWMA run-to-run controllers with metrology delay. IEEE Trans Semicond. Manufact. 19(1):78–86

Moore KL (2013) Iterative learning control for deterministic systems. Springer, London. (Softcover reprint of the original 1st ed, 1993)

Skogestad S, Postlethwaite I (2005) Multivariable feedback control: analysis and design, 2nd edn. Wiley, Chichester

Visioli A (2010) Practical PID control. Springer, Berlin. (Softcover reprint of hardcover 1st ed, 2006)

Xu J-X, Tan Y (2008) Linear and nonlinear iterative learning control. vol 291. Springer, Berlin. (Edition 2003)

Zhou K, Doyle JC (1997) Essentials of robust control. Pearson, Upper Saddle River. (Edition 01)

Chapter 4
Control of Higher-Level Dynamic Layers

The higher-level layer of the hierarchical control structure deals with the relations between those process variables that constitute the core of the process, i.e., the relations that define how the features of the final product y_h are determined according to the raw product characteristics f and the values of the set-points of the low-level process variables r_l. In a sense, the knowledge and experience in dealing with these relations are the key features that define an expert operator of a process, so the task of this layer is to somewhat assist these experts and easen their job by making educated decisions that assure that the final product features y_h will effectively reach the target values especified by r_h if the values of f make it possible, or suggest and achieve the best reachable value of y_h if the current objective is not feasible.

In order to accomplish this task, the controller needs to know in some detail how the different raw input features and process variables influence the final product features y_h, that is, a somewhat detailed and precise model of the relations between the variables of this layer is fundamental for the presented approach. Once that this knowledge is available, the key idea is to use these models to infer what decision to make according to the practical constraints that the two different types of variables of interest impose, as the set-points of process variables r_l can be adjusted during the operation of the plant but the characteristics of the raw inputs f cannot be changed once that they arrive at the plant so, for all practical purposes, they need to be regarded as disturbances. If these input properties can be measured, then that information can be used to adjust the process to those conditions, exactly like measurable disturbances are used in feedforward control of the lower-level control layer. It is precisely this feedforward nature of the approach that requires the models to be of an acceptable precision, as the inherent robustness induced by feedback is not available in feedforward methods.

Food transformation operations are typically complex multivariable processes, where there is usually more than one possible set of values of the process variables r_l that allow to attain a specific output target y_h for a given set of input characteristics

P. Cano Marchal et al., *Production Planning, Modeling and Control of Food Industry Processes*, Advances in Industrial Control,
https://doi.org/10.1007/978-3-030-01373-8_4

f. Conversely, different values of y_h can typically be achieved for a given set of f by choosing appropriate values of r_l. This additional freedom can be used to try to achieve the target output while considering additional factors, such as economic efficiency in the operation.

A simple yet interesting idea is that optimizing the operation of a food transformation plant for a given batch of raw inputs can be thought of as two independent problems: on the one hand, find the values of the technological variables that optimize the production of each product that can be produced with that raw input; on the other hand, once that the optimal operation conditions are known for each product, find which of those products offer the greatest profitability. Note that these two problems are in fact independent, in the sense that choosing which product offers the greatest profitability from a set of candidates improves the operation of the plant even if the production of each of the products is carried out suboptimally, meaning that it could be completed at a higher efficiency level. In turn, finding how to carry out the production of a given product as efficiently as possible is independent of considering whether producing that particular product is indeed the most profitable action for the given set of f.

The distinction between these two problems is of great practical interest due to the different complexity of the models required to solve them: setting up a problem to find the most profitable product for a given set of f can be accomplished with simple models based on aggregate historical data, while finding the optimal parameters for the production of a target y_h requires a full model of the process.

The ability to manipulate the behavior of the process and dampen the effect of f on y_h is provided by the set-points of the lower layer outputs r_l; however, given the frequently capital influence of the raw input features f on the final product characteristics y_h, the control authority provided by r_l might not be enough to assure that *any* desired value of y_h can be reached. Typically, a sensible approach is to consider the range of possible values of y_h offered by f and to choose one of those according to some criteria. This approach is equivalent to the finding which product is most profitable and is presented in Sect. 4.1.1.

Once that the production objective is considered known, the next task is to effectively operate the process, so that this objective is actually achieved successfully. At this stage, proper values of the set-points of the low-level variables (r_l) must be used. As commented above, it is typically the task of expert process operators to provide these values and supervise that the production objectives are actually met. If a proper model of the process is available, this set-point proposal step could be assisted by an expert system so that the burden laying on expert operators could be easened. This scenario is equivalent to finding the values of r_l that optimize the production of y_h for a given f, and Sect. 4.1.2 presents some ideas and methods that tackle this situation.

The feedforward nature of the ideas presented so far requires precise models for them to be implemented successfully in practice. Moreover, if unmeasured disturbances affected the process and prevented it from effectively fulfilling the production objective, the controller would not be able to update its proposed set-points. Even though a typical feature of the final product characteristics is that they are hard to measure, the inclusion of some feedback based on the available measurements or

indications from expert operators is very advisable in order to add robustness to the system. Section 4.2 presents how Model Predictive Control and Run-to-run control can be applied to this layer to introduce feedback from the process and robustify the control system, providing a means to counteract Modeling errors and unmeasured disturbances.

4.1 Feedforward of Input Characteristics

The fact that the characteristics of the raw inputs deeply influence the achievable features of final products is common to most food transformation processes and well known in the industry. These sections present the feedforward approaches that employ the knowledge about the input characteristics to improve the operation of the plant.

4.1.1 Production Objective Selection Based on Input Properties

The most straightforward way to initially address the influence of the raw input properties (f) on the final product is to explicitly consider what final product features (y_h) are reasonable to expect obtaining in the process for those input properties. From a more formal point of view, this is equivalent to considering a mapping from f to y_h, i.e.,

$$y_h = \Phi(f). \tag{4.1}$$

This mapping $\Phi(\cdot)$ can be obtained with varying degrees of detail depending on the data and time available. If historical data are accesible, a simple yet useful approach is to use these data to build a statistical input—output model that completely disregards the influence of the process. If the use of historical data is not viable, the knowledge of expert operators can be used to construct a fuzzy model relating just these sets of variables. Since the size of these sets is typically not too large, a standard rule-based, single-layer, fuzzy logic system (FLS) could be used for this purpose.

If this relation Φ is one-to-one, meaning that for each point in the raw input space there is only one possible reachable point in the product output space, then the problem would already be solved, as measuring the values of f would immediately provide the corresponding objective value r_h for y_h, provided that Φ is known. This means that the properties of the inputs completely determine the output characteristics, with the process variables not having any possibility to influence the value of the output features; in this scenario, all that can be done is to detect which level of quality of inputs is available and select the production objective directly evaluating the mapping Φ with those values of f. Notice that for this section, we assume that

the values of f are fixed; how to use this mapping to potentially improve the overall operation of the plant by defining what values of f would be adequate to have is the topic of Chap. 5.

However, having a one-to-one Φ is rare, so the more general case of having a one-to-many Φ must be considered. In this scenario, the state of the raw inputs conditions—but does not uniquely define—the properties of the outputs, leaving some freedom for choosing which production objective to aim for. A systematic way of dealing with this additional freedom is by defining an optimization problem whose solution should provide the desired production objective.

Defining an optimization problem involves establishing an objective function, assigning costs to the different variables, and including the relevant relations of the system as constraints, so that only physically plausible values are provided. For our purposes, the optimization problem can be generally defined as

$$\begin{aligned} \underset{y_h}{\text{minimize}} \quad & C(y_h) \\ \text{subject to} \quad & \Phi(f) = y_h, \end{aligned} \tag{4.2}$$

where $C(y_h)$ is the function that computes the total costs and revenues associated to the values of y_h. In this problem, y_h is the variable whose values are modified in order to find the point that provides the lowest possible value of $C(y_h)$ and f, as commented above, is a fixed value. In general, we will consider both y_h and f to be vectors, i.e., $y_h \in \mathscr{R}^{n_{yh}}$ and $f \in \mathscr{R}^{n_f}$, with n_{yh} and n_f being the dimension of y_h and f, respectively.

The nature of the objective function and the constraints define the difficulty of solving the problem. Linear problems, i.e., problems where both the objective function and the constraints are affine functions of the variables, can be solved very easily even for really large problems—hundreds of variables and thousands of constraints. The inclusion of nonlinearities in the objective function or the constraints complicates the solution of the problem, practically limiting the size of tractable problems and usually providing solutions that are not guaranteed to be the truly absolute optimum, but just locally optimum in some area of the space of variables.

Although somewhat obvious, it is important to remark that the solution of an optimization problem is optimal only for the objective function that was defined in the problem, the costs assigned to the variables and the constraints included, so in order to obtain useful insight from the solution of the problem, it must be carefully set up to capture the essential relations of the system, with the costs assigned to the variables being faithful representations of their actual expenses and remunerations and the objective function being aligned with the actual economic objectives of the operation.

Since the solutions of the optimization problem will be different depending on the cost coefficients included in the function $C(y_h)$, it is appropriate to consider how to define these values. There are different ways of tackling this task; the first is to use historical data of previous operations to estimate the profit obtained per type of

product produced and relate it to the components of y_h, ignoring as many details regarding the specifics of the process as possible.

Let us look in more detail at this approach with a simple example that will be also useful in concretizing the general optimization setup. We have defined y_h to be a vector variable that includes information regarding the features of interest of the final product. It is typically convenient to define the components of y_h taking into account how this final product is marketed. If there is a quality feature of the product whose value affects the price continuously, then y_h should include a component which is a continuous variable that keeps track of the value of that particular feature. If, on the other hand, our product is classified into a countable set of prespecified quality levels and remunerated accordingly, then it makes sense to define discrete variables that encode what quality level the product belongs to, ignoring to some extent the underlying features that qualify the product as apt for that particular quality level.

Let us initially suppose that our product is classified into two quality levels—say A and B—and the product is remunerated exclusively based on the quality level it belongs to. With this simple setup, a convenient way to define y_h is as a vector having two components, each one encoding whether the product belongs to the corresponding quality level or not

$$y_h = \begin{bmatrix} y_h^A \\ y_h^B \end{bmatrix}.$$

Each of the two components of y_h are binary variables, i.e., $y_h^A, y_h^B \in \{0, 1\}$. Furthermore, a product cannot belong to the two quality levels simultaneously, so we must enforce this condition including the following constraint in the optimization problem

$$y_h^A + y_h^B = 1. \tag{4.3}$$

This constraint effectively models our requirement because each of the components of y_h can only be either zero or one, so forcing their sum to equal 1 naturally induces the value of exactly one of them to be 1, while the other will be 0.

The conditions imposed by f on y_h can be modeled defining a function Φ^k for each quality k that determines whether this particular quality level can be produced with the current value of f – for this simple example, $k \in \{A, B\}$. This function can be built to return a 1 if the product can be produced and 0 otherwise. Note that although $\Phi^k(f)$ depends on f, since the value f is fixed at the time that we need to solve the optimization problem, $\Phi^k(f)$ will be a constant parameter in that instance of the optimization problem. We can make this more explicit by defining a parameter

$$\delta_f^k = \Phi^k(f) \tag{4.4}$$

so that the constraint can be expressed as

$$y_h^k \leq \delta_f^k. \tag{4.5}$$

Let p^A and p^B denote the selling price of the product for quality A and B, respectively, and c^A and c^B denote the corresponding production costs obtained from historical records. Then, the objective function can be defined as

$$C(y_h) = (c^A - p^A)\, y_h^A + (c^B - p^B)\, y_h^B, \tag{4.6}$$

which simply states that the total cost incurred is the production cost minus the selling price for each quality.

The optimization problem, thus, can be written as

$$\begin{aligned} \underset{y_h^A, y_h^B}{\text{minimize}} \quad & (c^A - p^A)\, y_h^A + (c^B - p^B)\, y_h^B \\ \text{subject to} \quad & y_h^A \leq \delta_f^A, \\ & y_h^B \leq \delta_f^B, \\ & y_h^A + y_h^B = 1, \\ & y_h^A, y_h^B \in \{0, 1\}. \end{aligned} \tag{4.7}$$

Let us analyze the solutions provided by the problem for some scenarios. If both products can be produced with the current value of f, i.e., both δ_f^A and δ_f^B equal 1, then the first two constraints, which define which products can be produced, do not have any influence in the optimization problem and the product that offers a better profitability is selected, as we have included a constraint that forces to provide just one product.

If the value of f is such that we can only produce one type of product, it will be selected, as the first two constraints will force the other product to have a value of 0 and we have explicitly stated that one product must be selected by defining (4.8) with an equality constraint. Note that this also implies that if f does not qualify to produce any of the two products, i.e., $\delta_f^A = \delta_f^B = 0$, then the problem is unfeasible and the solver would throw an exception. If, instead, we prefer the problem to remain feasible, we can relax (4.8) into

$$y_h^A + y_h^B \leq 1. \tag{4.8}$$

This way, we guarantee that we produce at most one product, but we are no longer requiring that one of the two components of y_h should be 1. Note that as long as producing a product is profitable, that is, as long as

$$c^k - p^k < 0 \tag{4.9}$$

for some k, the variable y_h^k will be driven to be 1 by the optimization problem if that is compatible with the rest of constraints. There is another small implication of relaxing (4.8) into (4.9); if the equality constraint is used, production will be assigned to a product—if the quality constraints allow it—even if it operates at a loss, while using the inequality will only assign production if the product is profitable.

Another interesting remark is that, as long as

$$c^A - p^A \neq c^B - p^B \tag{4.10}$$

that is, as long as the products are not exactly equally profitable, we can relax the condition requiring y_h^k to be binary into just requiring it to be between 0 and 1

$$0 \leq y_h^k \leq 1, \tag{4.11}$$

since the optimal solution will always be to produce the product with the highest profitability, thus the corresponding y_h^k will always have the maximum possible value, which is 1. This substitution changes the nature of the problem from a Integer Linear Problem to a Linear Problem, which is much easier to solve. This idea of replacing a constraint by another that does not change the feasible set and provides the same optimal value at a lower computational complexity level is ubiquitous and important to keep in mind when defining optimization problems, particularly when their sizes begin to be considerable.

If we take a step back and analyze the discussion above about the solutions that this optimization problem provides, we can realize that the problem just supplies the answer that common sense applied to the particular situation would, which is to produce the most profitable product possible. For such a simple setup, there is a product that is always more profitable than the other, and the only reason not to produce it is if it is not possible because of the value of f. The difficulty for setting up the problem is in determining accurately the production costs c^A and c^B, as these parameters deeply influence the solution of the problem. Note that assigning a constant value to these parameters means that the production costs are not influenced whatsoever by the value of f, which is typically a quite crude simplification.

What is remarkable and useful of the presented approach is that it can handle a large variety of situations in a very compact way that allows to easily include other characteristics into the problem. Let us illustrate it by slightly increasing the complexity of the problem supposing that our product, besides being classified into two quality levels, also has a feature that is remunerated linearly at two different prices depending on the quality assigned to the product. To model this new scenario, we can extend y_h with a component to contain the value of the remunerated quality feature

$$y_h = \begin{bmatrix} y_h^A \\ y_h^B \\ y_h^Q \end{bmatrix}.$$

Under this setup, the cost function can be expressed as

$$C(y_h) = c^A\, y_h^A - p^A\, y_h^A\, y_h^Q + c^B\, y_h^B - p^B\, y_h^B\, y_h^Q. \tag{4.12}$$

Note that the cost function is no longer linear, as there are products of variables in it. Note also that, under this new setup, there might no longer be a *more prof-*

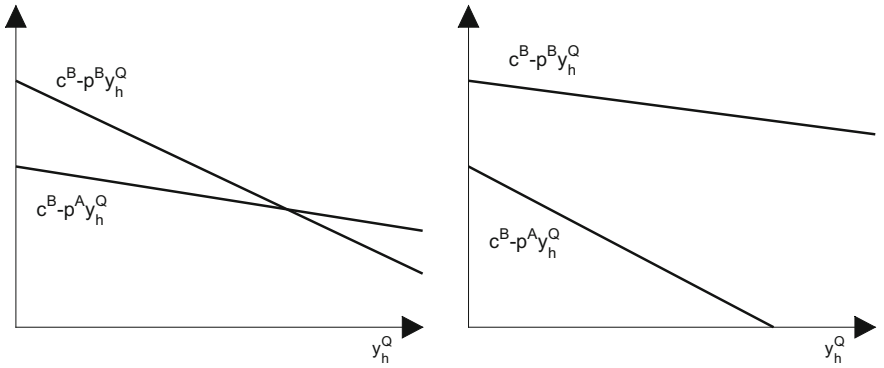


Fig. 4.1 Profitability of products A and B as a function of y_h^Q. Depending on the values of c^k and p^k there may be a more profitable product for every value of y_h^Q or it may depend on y_h^Q

itable product, but that it may depend on the value of y_h^Q, even assuming that the same level of y_h^Q is reached for both levels of quality. Figure 4.1 depicts an example where y_h^Q determines the profitability (left) and an example where the profitability is independent of y_h^Q (right). In the latter case, it is not necessary to include y_h^Q in the optimization problem as it does not influence the solution of the problem; a model like (4.7) that considers only the two quality levels would provide the same result with lower complexity.

In the scenario shown in the left plot of Fig. 4.1, the features of the raw inputs condition the production in two different ways: on the one hand, analogously to the previous example, they may prevent one of the products to be produced due to the quality threshold encoded by δ_f^k; on the other hand, they influence the profitability via their effect on y_h^Q. Let us assume that the final value of y_h^Q depends on which type of product is produced but is fixed for the product, i.e., every time that product k is produced, the value of y_h^Q is γ_f^k. This can be modeled as a linear equality

$$y_h^Q = \gamma_f^A \, y_h^A + \gamma_f^B \, y_h^B, \tag{4.13}$$

which yields the optimization problem

$$\begin{aligned} \underset{y_h^A, y_h^B}{\text{minimize}} \quad & c^A \, y_h^A - p^A \, y_h^A \, y_h^Q + c^B \, y_h^B - p^B \, y_h^B \, y_h^Q \\ \text{subject to} \quad & y_h^A \leq \delta_f^A, \\ & y_h^B \leq \delta_f^B, \\ & y_h^A + y_h^B = 1, \\ & y_h^Q = \gamma_f^A \, y_h^A + \gamma_f^B \, y_h^B, \\ & y_h^A, y_h^B \in \{0, 1\}. \end{aligned} \tag{4.14}$$

Note that y_h^Q is linked to y_h^A and y_h^B via a linear equality constraint; this means that the problem can be rewritten eliminating y_h^Q from the problem. Substituting y_h^Q using Eq. (4.13) in the objective function yields

$$C(y_h) = c^A\, y_h^A - p^A\, y_h^A \left(\gamma_f^A\, y_h^A + \gamma_f^B\, y_h^B\right) + c^B\, y_h^B - p^B\, y_h^B \left(\gamma_f^A\, y_h^A + \gamma_f^B\, y_h^B\right).$$

This expression can be greatly simplified using the binary and mutually exclusive nature of y_h^A and y_h^B. These properties allow to write

$$\begin{aligned} y_h^A\, y_h^A &= y_h^A, \\ y_h^B\, y_h^B &= y_h^B, \\ y_h^A\, y_h^B &= 0. \end{aligned}$$

Using these expressions and removing the constraint containing y_h^Q, the optimization problem is written as

$$\begin{aligned} \underset{y_h^A, y_h^B}{\text{minimize}} \quad & (c^A - p^A\, \gamma_f^A)\, y_h^A + (c^B - p^B\, \gamma_f^B)\, y_h^B \\ \text{subject to} \quad & y_h^A \leq \delta_f^A, \\ & y_h^B \leq \delta_f^B, \\ & y_h^A + y_h^B = 1, \\ & y_h^A,\, y_h^B \in \{0, 1\}. \end{aligned} \tag{4.15}$$

This problem is remarkably similar to (4.7), the only difference being the substitution of p^k with $p^k\, \gamma_f^k$. Note that both p^k and $p^k\, \gamma_f^k$ are constant parameters whose values are known and fixed before the optimization problem is solved, so for the practical purposes of solving the optimization problem there is absolutely no difference between problems (4.7) and (4.15); there are, however, important differences in the assumptions used to derive these models and the meaning of the coefficients in the objective function.

The key aspect that explains that we arrive at problems with the same structure is the fact that we assume that f is fixed and we are taking into account the influence of the process via the parameters c^k that account for the production expenses. At the end of the day, the optimization problem, as commented before, will just provide the most profitable product that is possible to produce. The profitability is entirely determined by the coefficients of the objective function; it is how these coefficients are computed where the difference between the models resides, but how these coefficients are found is transparent to the optimization problem itself.

Note, for instance, that the requirement that γ_f^k does not depend on f, but only on the product k can be generalized to consider that γ_f^k does in fact depend on the value of f without having to alter the optimization problem at all. There would be, of course, a difference when applying the technique to a real plant, as for the case

when γ_f^k does not depend on f those parameters need to be estimated only once, while having γ_f^k varying with f requires to update these parameters for every batch of raw inputs received.

Another interesting remark is that the optimization problem (4.15) can also model a scenario where the production of the product is subject to a certain yield that may vary depending on the value of f. This yield can be captured in a parameter γ_f^k, and it would provide exactly the same objective function and constraints.

The substitution of variables and simplification of optimization problems with the intention of transforming problem (4.14) into (4.15) is interesting as provides a simpler problem that can be solved more easily and faster. However, these benefits come at the expense of the interpretability of the problem; it is easier to visualize which assumptions are made in the former optimization than in the condensed latter. Consequently, it is often more interesting to work with relatively verbose optimization problems in terms of variables and constraints but that are easier to interpret, rather than reducing the number of constraints and variables since the beginning. Most commercial and open-source solvers already include some preprocessing steps where they actively use this type of substitution techniques to effectively reduce the sizes of the problems in order to solve them faster and more efficiently, so the price paid in terms of solution time is usually not worth the additional effort required to interpret the solutions provided, particularly when initially setting up and debugging the problem.

4.1.2 Set-Point Selection of Intermediate Process Variables

One of the common features between food and process industry is that their production objectives are typically the result of a careful consideration of different competing effects to provide an overall optimum from an economic point of view. As presented in the previous section, the features of the raw inputs condition the set of feasible production objectives;however, it is the process variables that ultimately define what specific values of y_h are achieved out of those that could in principle be attained. Furthermore, these process variables often have a large influence on the operation cost of the production process, so choosing adequate values for these variables is important both to actually achieve the production objective and to do so efficiently from a economic point of view.

Food transformation processes typically have more inputs than outputs, so they are typically *fat* plants where the same output y_h can be achieved using different sets of values of r_l. Analogously to what was commented for the selection of the production objective for one-to-many relations between f and y_h, this additional freedom can be used to try to achieve the target output while considering additional factors, such as economic efficiency in the operation. These additional considerations can again be formulated into an objective function, so that it can be included in an optimization problem whose solution provides the corresponding optimal values of the process variables.

Note that, unlike the optimization problem in the previous section, the variables being optimized in the problem are no longer the production objective y_h, but the process variables r_l. Consequently, the set of constraints included in this optimization problem must contain a model that relates the outputs with the input features *and* the process variables, that is, this optimization problem requires a full model of the process relations. The problem presented in the previous section did not require this full model of the process to determine which values of y_h to aim for because all the influence of the process variables was condensed into precomputed parameters that were considered to be available before solving the problem instance—the solution of the optimization problems presented in this section is precisely one way to provide these coefficients to the objective selection optimization problem.

The optimization problem whose solution provides the optimal process variables to obtain a target output y_h^* for a given input characteristics f^* can be defined as

$$\begin{aligned} \underset{r_l}{\text{minimize}} \quad & J(r_l, y_h) \\ \text{subject to} \quad & y_h = \Phi(r_l, f), \\ & f = f^*, \\ & y_h = y_h^*. \end{aligned} \tag{4.16}$$

Here, $J(r_l, y_h)$ is the objective function and $\Phi(r_l, f)$ denotes the model of the process. Note that for this problem to be feasible, y_h^* should be compatible with f^*; that is, there should exist some value of r_l such that y_h^* could be obtained from f^*. If that were not the case, then the problem would be unfeasible and the solver would throw an exception when trying to solve it. Note that solving this optimization does not imply that obtaining y_h^* from f^* is the optimal decision, the solution to this problem simply provides the values of r_l that allow to carry out that production plan optimally; it might be the case that aiming to produce some other y_h might offer a higher rentability.

The decision of producing y_h^* from f^* would typically not be arbitrary, but would came from the solution of an optimization problem similar to the ones proposed in the previous section, like problem Eq. (4.2). However, since problem (4.16) is more general than Eq. (4.2), it can be modified to choose the target value for y_h with little extra effort, simply by including terms in the cost function that model the revenue obtained based on the values of y_h and adding y_h into the set of variables being optimized

$$\begin{aligned} \underset{r_l, y_h}{\text{minimize}} \quad & J(r_l, y_h) \\ \text{subject to} \quad & y_h = \Phi(r_l, f), \\ & f = f^0. \end{aligned} \tag{4.17}$$

The main caveat of this approach is, precisely, the necessity of having a reasonably accurate model of the process relations $\Phi(r_l, f)$ to set up the optimization problem. This is also the reason why the approach presented in the previous section is useful.

The number of variables included in this type of models is typically much larger than those required to construct a direct mapping from f to y_h, so the complexity of constructing this type of models is much higher.

However, once that the $\Phi(r_l,\ f)$ is known, the fact that a model that relates all the relevant variables in the plant is available offers great possibilities for optimizing the process. For instance, it is also possible to suppose that f is not fixed and solve a modified version of (4.19) that theoretically also provides the optimal value of f

$$\begin{aligned} \underset{r_l,\ y_h,\ f}{\text{minimize}} \quad & J(r_l,\ y_h) \\ \text{subject to} \quad & y_h = \Phi(r_l,\ f). \end{aligned} \tag{4.18}$$

By allowing f to be optimized as well, the solution to this problem would provide the absolutely most profitable—according to $J(r_l,\ y_h)$—production scenario. Note that the solution of this problem, although theoretically appealing, provides indications for just one production scenario; in practice, the values of f will not always be the optimal ones, so it is of interest to consider a more general approach that offers insight for a wider set of production conditions.

One optimization technique that can be used to take full advantage of the process model $\Phi(r_l,\ f)$ to obtain valuable insight is multiobjective optimization. Multiobjective optimization is a generalization of the optimization problems presented so far that allows to deal with different competing objectives in a systematic way. The key difference between scalar optimization and multiobjective optimization is that the objective space induced by the competing objectives is no longer an ordered space, meaning that it is not always clear if a point represents a better alternative than another. The solution of this type of problems is given by a so-called Pareto frontier that includes the optimal alternatives for the different competing objectives.

Let us illustrate the technique with an example. Suppose that the final product possesses two desirable features whose relation represents a trade-off: Increasing the value of one is achieved at the expense of the other. This a very common situation that reflects the typical trade-off between some notion of quality for the product and some notion of productivity or yield. In this example, the vector y_h can be defined to have two components

$$y_h = \begin{bmatrix} y_h^A \\ y_h^B \end{bmatrix}.$$

Here, $y_h^A,\ y_h^B \in \mathscr{R}$, i.e., the components of y_h are continuous variables. Let us further suppose that y_h is defined so that we our competing objectives are to minimize the value of each of the components of y_h

$$F = \begin{bmatrix} \underset{r_l}{\text{minimize}} & y_h^A \\ \underset{r_l}{\text{minimize}} & y_h^B \end{bmatrix}.$$

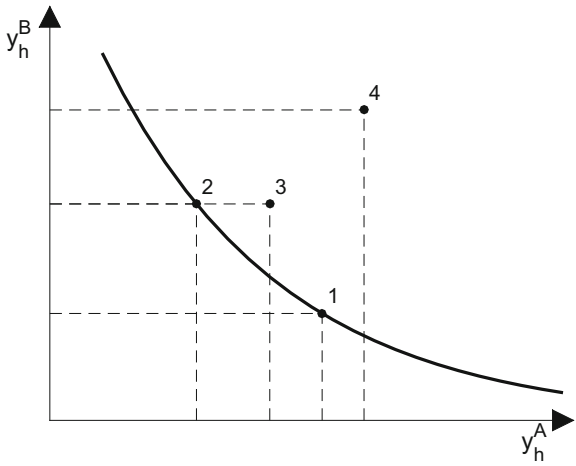


Fig. 4.2 Objective space for the multiobjective optimization example. The points labeled 1 and 2 are non-dominated, while the points 3 and 4 are dominated

The optimization problem, assuming that we consider the raw input features to be fixed, can then be expressed as

$$\begin{aligned} \underset{r_l}{\text{minimize}} \quad & F \\ \text{subject to} \quad & y_h = \Phi(r_l, f), \\ & f = f^0. \end{aligned} \tag{4.19}$$

A key notion in multiobjective optimization is the idea of *non-dominated point*. In general, non-dominated points are points where, in order to improve in one of the objectives, at least one of the others must decline. For our example, since we only have two objectives, a non-dominated point is a point where we must diminish one objective in order to enhance the other.

This concept can be visualized nicely, as the components of y_h can be plotted jointly in a plane as depicted in Fig. 4.2. In this figure, the points labeled 1 and 2 represent non-dominated points, while 3 and 4 are dominated points. The point labeled 1 has a lower value of y_h^B than point 2 at the expense of having a larger value of y_h^A. The point labeled 3 shows the same value of y_h^B than point 2, however, the value y_h^A could still be decreased more, so it is not an optimal point. Non-dominated points are also usually called *Pareto-optimal* points, and the line where these points lie is called the Pareto frontier.

Note that the one of the keys ideas in multiobjective optimization is that all the objectives are given equal relevance, the Pareto frontier offers the set of points that are achievable given the problem constraints, offering no preference for neither of those. This approach is interesting, for instance, when the prices associated with the output features are not fixed; this technique offers the set of possible alternatives, so that educated decisions can be made.

Another important remark is that the solution to the optimization problem provides not only the set of Pareto-optimal alternatives, but also the values of the process variables r_l that are required to achieve each of those points. Since the process costs are usually tightly related to the values of these variables, this information can be used to estimate these costs and offer more insight for the decision-making process regarding product prices commented in the paragraph above.

The problem (4.19) cannot be fed as is to a solver for its solution, as solvers can only deal with scalar objective functions. There are different techniques that allow the construction of the Pareto frontier, and they typically require solving a set of optimization problems that have scalar objective functions derived from the vector F. A common and simple approach is called weighted sum scalarization, which requires finding the solution to the following problems

$$\begin{aligned} \underset{r_l}{\text{minimize}} \quad & \sum_{k=1}^{K} \omega_k \, F_k \\ \text{subject to} \quad & y_h = \Phi(r_l, f), \\ & f = f^0, \end{aligned} \tag{4.20}$$

for different combinations of scalarization weights ω_k. Here, $k = 1, \ldots, K$ indexes the objectives.

Lexicographical optimization is another useful approach to the solution of multi-objective optimization problems. The key idea is to prioritize the elements of F and solve the problem for one objective at a time, including the values of the previous objective functions as constraints for the following optimization problems. For our example, since we only have two objectives, we should choose which one to prioritize. Let us suppose that we want to optimize y_h^A first, so the first problem to solve would be

$$\begin{aligned} \underset{r_l}{\text{minimize}} \quad & y_h^A \\ \text{subject to} \quad & y_h = \Phi(r_l, f), \\ & f = f^0. \end{aligned} \tag{4.21}$$

Let us denote the optimal value of y_h^A as y_h^{A*}, then the second optimization problem is given by

$$\begin{aligned} \underset{r_l}{\text{minimize}} \quad & y_h^B \\ \text{subject to} \quad & y_h = \Phi(r_l, f), \\ & f = f^0, \\ & y_h^A = y_h^{A*}. \end{aligned} \tag{4.22}$$

Note the inclusion of the last constraint. Note also that the constraint is set for y_h^A, not for r_h. The reason for this is that we assuming a *fat plant*, so there may potentially

be another set of values of r_h that provide y_h^{A*} while yielding a better value for y_h^B than the one provided by the solution of problem (4.21).

The lexicographical optimization approach typically offers very different solutions depending on the order of solution of the problems, i.e., depending on the ordering of the elements of the objective vector. The reason is that each subproblem completely disregards the following objectives, so it tends to offer extreme values of the Pareto frontier. This behavior is typically not wanted in cases like the one presented in the example, where the two objectives can be considered very similar to each other in relevance. However, this approach can be used, for instance, for minimizing the operation costs once that a production objective has been defined, making sure that the solution of the problem minimizing the costs yields the very same objective as computed without consideration of the costs and not just a close one.

4.2 Feedback of Final Product Properties

One of the defining features of the output variables of the higher-level control layer y_h is their difficulty to be measured; however, the inclusion of feedback is so beneficial that it is worth analyzing what can be done with the measurement possibilities available in each case.

If the measurement of the outputs can be carried out on-line or with a large enough sampling rate, Model Predictive Control (MPC) is a well-known approach that is widely used in process industry for tasks similar to the one considered here.

If the available measurements are scarce, then it is not really possible to implement a full classical MPC approach; however, the inclusion of some modifications under the point of view of Run-to-run control offers a viable approach that, although less powerful than the full MPC, still offers an increase of the robustness of the control system versus a pure feedforward approach.

4.2.1 Model Predictive Control

Model Predicitive Control is a control paradigm that is extensively employed in the higher layers of hierarchical control schemes for the process industry. It is a quite powerful approach that allows to take into account easily aspects, such as the existence of limits in the control action or the inclusion of explicit objective functions, that are difficult to handle in the classical design of feedback controllers. Furthermore, its approach makes it very easy to work with multivariable systems, which is one of the key aspects of the higher layers of the hierarchical control scheme.

The basic idea of the approach is to make explicit use of a model of the system to find a sequence of inputs that drives the output from its current value to the desired one in a certain period of time. This sequence of inputs is found solving an optimization

problem that includes this model of the system as one of its constraints. This problem is solved at each sampling time and provides the whole sequence of values for the control input; however, only the first value is actually applied to the plant, as the next sampling instant the problem is solved again using the updated information obtained from the sensors, so that an updated value of input can be applied to the plant. This idea of applying only the first computed input signal and redoing all the computations the next sampling instant is called *receding horizon* and is one of the key concepts of MPC and the one that provides its closed-loop nature, as opposed to open-loop feedforward approaches where the control signal is computed once and applied irrespective of the actual evolution of the plant.

This discussion on MPC is presented in the context of the higher-level layer of the control hierarchical control structure, where the control signal is denoted by r_h—as it is these values that constitute the manipulated variables in the control setup—y_h designates the output and the reference is represented by r_h. However, throughout this section we will use standard notation typically used in the MPC literature in order to simplify the work for readers interested in further reading about MPC using the excellent resources available. This way, the control signal will be denoted u, the reference r, and the output y.

Many different flavors of MPC exist in the control literature, with their major differences being the type of model and objective function employed. For this section, we will assume that the system is modeled using a discrete state-space representation, however, most of the comments in the section carry out with no modifications if the type of model employed is of different nature. Let the model of the plant be given by a standard state-space formulation without the feedforward term as

$$\begin{aligned} x(t+1) &= A\,x(t) + B\,u(t), \\ y(t) &= C\,x(t). \end{aligned} \tag{4.23}$$

It is very frequent in MPC to employ the difference of the control input as the manipulated variable instead of its absolute value, so we can define

$$\Delta u(t) = u(t) - u(t-1). \tag{4.24}$$

In order to employ $\Delta u(t)$ as the control signal in the formulation of the model, an additional variable needs to be included in the state to keep track of the absolute value of $u(t)$, yielding the augmented model

$$\begin{aligned} \begin{bmatrix} x(t+1) \\ u(t) \end{bmatrix} &= \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x(t) \\ u(t-1) \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u(t) \\ y(t) &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ u(t-1) \end{bmatrix}. \end{aligned} \tag{4.25}$$

This system can be rewritten more compactly as

$$\begin{aligned} \tilde{x}(t+1) &= \tilde{A}\,\tilde{x}(t) + \tilde{B}\,\Delta u(t), \\ y(t) &= \tilde{C}\,\tilde{x}(t), \end{aligned} \tag{4.26}$$

with the definitions of $\tilde{x}(t)$, $\tilde{A}$, $\tilde{B}$, *and* $\tilde{C}$ being easily deduced by direct comparison of Eqs. (4.25) and (4.26). The predicted output j time instants ahead from a given time t_0 can be computed using this process model and is given by

$$\hat{y}(t_0 + j \mid t_0) = \tilde{C}\tilde{A}^j \tilde{x}(t_0) + \sum_{i=0}^{j-1} \tilde{C}\tilde{A}^{j-i-1}\tilde{B}\Delta u(t_0 + i). \tag{4.27}$$

The hat included in the notation $\hat{y}(t_0 + j \mid t_0)$ is used to emphasize that $\hat{y}$ is a prediction of y, not necessarily its actual value, while the *conditioning* that appears in the argument is used to accentuate the fact that this prediction is made with information available up to time t_0. Let us assume that we can measure the whole state vector at time t_0 $\tilde{x}(t_0)$, then note that the first addend of the right term is a fixed known value. The second addend, in turn, is not fixed and depends on the inputs to be applied to the plant from time t_0 until time $t_0 + j - 1$. This equation can be particularized for the different time instants of interest to us and each of those instances included as constraints for the optimization problem to be solved, as they relate the corresponding values of Δu, which can be chosen by us, with the values of $\hat{y}$, which are our outputs of interest. Particularizing Eq. (4.27) for the first few time instants yields

$$\begin{aligned} \hat{y}(t_0 + 1 \mid t_0) &= \tilde{C}\tilde{A}\tilde{x}(t_0) + \tilde{C}\tilde{B}\Delta u(t_0) \\ \hat{y}(t_0 + 2 \mid t_0) &= \tilde{C}\tilde{A}^2\tilde{x}(t_0) + \tilde{C}\tilde{A}\tilde{B}\Delta u(t_0) + \tilde{C}\tilde{B}\Delta u(t_0 + 1) \\ &\vdots \\ \hat{y}(t_0 + k \mid t_0) &= \tilde{C}\tilde{A}^k\tilde{x}(t_0) + \tilde{C}\tilde{A}^{k-1}\tilde{B}\Delta u(t_0) + \tilde{C}\tilde{A}^{k-2}\tilde{B}\Delta u(t_0 + 1) + \cdots + \tilde{C}\tilde{B}\Delta u(t_0 + k - 1). \end{aligned}$$

This construction is analogous the one presented in Chap. 2 in the context of subspace identification, and can be rewritten in matrix form as

$$\begin{bmatrix} \hat{y}(t_0+1 \mid t_0) \\ \hat{y}(t_0+2 \mid t_0) \\ \vdots \\ \hat{y}(t_0+k \mid t_0) \end{bmatrix} = \begin{bmatrix} \tilde{C}\tilde{A} \\ \tilde{C}\tilde{A}^2 \\ \vdots \\ \tilde{C}\tilde{A}^k \end{bmatrix} \tilde{x}(t_0) + \begin{bmatrix} \tilde{C}\tilde{B} & 0 & \dots & 0 \\ \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{C}\tilde{A}^{k-1}\tilde{B} & \tilde{C}\tilde{A}^{k-2}\tilde{B} & \dots & \tilde{C}\tilde{B} \end{bmatrix} \begin{bmatrix} \Delta u(t_0) \\ \Delta u(t_0+1) \\ \vdots \\ \Delta u(t_0+k-1) \end{bmatrix}$$

This equation can be written more compactly as

$$\mathbf{y} = \mathbf{F}\,\tilde{x}(t_0) + \mathbf{H}\,\mathbf{u}, \tag{4.28}$$

where the terms are defined as

$$\mathbf{y} = \begin{bmatrix} \hat{y}(t_0+1\,|\,t_0) \\ \hat{y}(t_0+2\,|\,t_0) \\ \vdots \\ \hat{y}(t_0+k\,|\,t_0) \end{bmatrix}, \ \mathbf{u} = \begin{bmatrix} \Delta u(t_0) \\ \Delta u(t_0+1) \\ \vdots \\ \Delta u(t_0+k-1) \end{bmatrix},$$

$$\mathbf{F} = \begin{bmatrix} \tilde{C}\tilde{A} \\ \tilde{C}\tilde{A}^2 \\ \vdots \\ \tilde{C}\tilde{A}^k \end{bmatrix}, \qquad \mathbf{H} = \begin{bmatrix} \tilde{C}\tilde{B} & 0 & \dots & 0 \\ \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{C}\tilde{A}^{k-1}\tilde{B} & \tilde{C}\tilde{A}^{k-2}\tilde{B} & \dots & \tilde{C}\tilde{B} \end{bmatrix}.$$

Equation (4.28) is just a linear system of equations that relate $\mathbf{y}$ and $\mathbf{u}$ with $\tilde{x}(t_0)$ using coefficients that are derived from the known model of the process. Note that although $\mathbf{y}$ is completely determined by $\tilde{x}(t_0)$ and $\mathbf{u}$, its value is not known, as $\mathbf{u}$ is not fixed. Indeed, we intend to solve an optimization problem to provide us with the values of $\mathbf{u}$ that supply the values of $\mathbf{y}$ that minimize a certain objective function.

The objective function most frequently used in MPC is comprised of two terms: one that penalizes the deviations of the output from the reference and another intended to limit the control effort. An example of such a function is given by

$$J(\hat{y}(t), \Delta u(t)) = \sum_{j=N_1}^{N_2} \delta(j)\Big(r(t+j) - \hat{y}(t+j\,|\,t)\Big)^2 + \sum_{j=1}^{N_u} \lambda(j)\Big(\Delta u(t+j-1)\Big)^2. \tag{4.29}$$

This objective function includes the parameters N_1, N_2, and N_u that exert a fundamental influence in the behavior of the controller. N_1 and N_2 determine the time span where it is important that the output follows the reference. N_2 is the maximum prediction horizon, and delimits how *far* into the future we are predicting the evolution of $\hat{y}(t)$, while N_1 determines when to start to penalize deviations of $\hat{y}(t)$ from $r(t)$. If N_1 is chosen to be 1, it means that we are immediately penalizing these deviations, while having a larger value allows for the system to evolve without considering those deviations in the objective function. Having values larger than 1 for N_1 is quite common for systems with time delays or inverse responses, since for these systems it is known that the system will take some time to start approaching the reference, and it makes sense to leave some room for these dynamics to take place and begin considering the deviations after this time.

N_u, in turn, is called the *control horizon* and is used to influence the shape of the control input. Having a value of N_u lower than N_2 is typically attached to forcing that, from that point on, the change of the control signal should be 0

$$\Delta u(t+j) = 0, \ \ j > N_u.$$

Finally, the factors $\delta(t)$ and $\lambda(t)$ can be used for two things; on the one hand, if they are taken as constants $\delta(t) = \delta$ and $\lambda(t) = \lambda$, they are used to define the relative importance assigned to deviations from the reference and variations of the control input. If they are allowed to change with time, they can be further employed to assign different relevance to errors happening at different times, which in turn can be used to induce a more conservative or more aggressive control action.

Joining all the elements of the previous discussions, the optimization problem whose solution provides the sequence of inputs can be written

$$\begin{aligned} &\underset{\Delta u(t)}{\text{minimize}} \quad \sum_{j=N_1}^{N_2} \delta(j)\Big(r(t+j) - \hat{y}(t+j \mid t)\Big)^2 + \sum_{j=1}^{N_u} \lambda(j)\Big(\Delta u(t+j-1)\Big)^2 \\ &\text{subject to} \quad \mathbf{y} = \mathbf{F}\,\tilde{x}(t_0) + \mathbf{H}\,\mathbf{u}. \end{aligned} \tag{4.30}$$

This is just an equality constraint optimization problem whose solution can be computed analytically, as we can use the relations defined in the constraints to eliminate the variables $\hat{y}(t+j \mid t)$ of the objective function and just minimize a function of the variables $\Delta u(t)$. For $\delta(t) = 1$ and $\lambda(t) = \lambda$, this solution is given by

$$\mathbf{u} = \big(\mathbf{H}^T\mathbf{H} + \lambda I\big)^{-1}\mathbf{H}^T\big(\mathbf{r} - \mathbf{F}\,\tilde{x}(t_0)\big). \tag{4.31}$$

If the control and maximum prediction horizons, i.e., N_u and N_2, respectively, tend to infinity, this solution coincides with the well-known LQR controller, which provides the grounds for some theoretical analysis of the convergence and stability of MPC controllers.

The inclusion of additional constraints to the problem, such as limiting absolute value of the control action or restricting its rate of change, lead to it not having a closed-form solution any more. However, depending on the type of constraints included, this may not impose any practical difficulty. The problem (4.30) is not a Linear Optimization Problem because the objective function presents quadratic terms, however, it belongs to a special class of nonlinear optimization problems called Convex Optimization Problems. These problems have a special structure that make them almost as easy to solve as Linear Problems, so they can be solved really fast in practice, and the specific nature of MPC, where different but very similar problems are to be solved repeatedly, can be exploited to make the solution of these problems even more efficient.

Once that we no longer require a closed-form solution of the optimization problem, different types of convex objective functions can be used without significantly complicating the solution of the problem, which opens the door to tailoring the design of this function to better fit the actual control objectives. For instance, using absolute values—l_1 norms—instead of squared differences in the objective function is known to induce sparsity in the solution of the problem, meaning that many of the values of $\Delta u(t)$ provided will be zero, which might be a desirable feature.

Furthermore, the inclusion of additional constraints on the optimization problem is typically good for shaping to some extent the control action provided. For instance, the already mentioned limits on the change of the control action, besides Modeling an actual limitation of many real actuators, can be used to obtain gentle control actions that do not include undesirable high-frequency components.

So far we have supposed that all the state vector is measurable. This strong assumption can be relaxed employing an observer. An observer makes use of the model of the system and of the available measurements to construct an estimation of the value of the vector state. A full Luenberger state observer is given by

$$\hat{x}(t+1) = A\hat{x}(t) + Bu(t) + L\Big(y(t) - C\hat{x}(t)\Big). \tag{4.32}$$

The construction of this observer supposes having a dynamical system for the error of prediction whose eigenvalues can be located by appropriately choosing the value of L. Let the estimation error be given by

$$e(t) = x(t) - \hat{x}(t), \tag{4.33}$$

then the error at the next time instant can be computed using Eq. (4.32) as

$$\begin{aligned} e(t+1) &= Ax(t) + Bu(t) - A\hat{x}(t) - Bu(t) - L\Big(y(t) - C\hat{x}(t)\Big) \\ e(t+1) &= Ax(t) - A\hat{x}(t) - L\Big(Cx(t) - C\hat{x}(t)\Big) \\ e(t+1) &= \big(A - L\,C\big)e(t) \end{aligned} \tag{4.34}$$

This last line shows that the dynamics of the estimation error are indeed governed by the eigenvalues of $A - L\,C$, which, as commented, can be located appropriately choosing an adequate value of L using the Ackermann formula or other more numerically robust alternative. If noise is affecting the measurements, then it could be of interest to design a Kalman filter to construct the estimation of the state vector. The reader can find a very thorough treatment of MPC in Camacho and Bordons (2004).

4.2.2 Run-to-Run Control

In situations where measuring the outputs of interest of the plant is difficult and cannot be made online, it is difficult to obtain a dynamic model of the system, so Model Predictive Control cannot be applied. If a static model of the system is available, then this knowledge can be used, together with the accessible measurements of the output, to include feedback into the higher-level layer applying Run-to-run control ideas to these multivariable systems.

The main reason to apply feedback to the higher-level layers is to correct eventual deviations of the high-level outputs from their desired values. These deviations

may arise from different origins: lack of precision of the model employed in the feedforward computations, errors in the estimation of the properties of the inputs or existence of unmeasured disturbances acting on the system. In effect, if the models used to compute the set-points of the process variables are not precise enough, the feedforward approach presented in Sect. 4.1.2 will provide values that will be slightly off, thus yielding an output value that will not exactly match the target value. In turn, even if the models were perfectly accurate, errors on the estimation of the properties of the raw inputs may also drive the feedforward system to provide reference values that again miss the target. Finally, without the inclusion of some sort of feedback on the system, the effect of unmeasured disturbances on the ouptut cannot be counteracted. Note that in practice, it is typically quite difficult to find out the actual reason that drives the output away from its target, however, all of these effects can be compensated for with the inclusion of a Run-to-run controller in the high-level layer.

Run-to-run controllers are typically composed of a model of the plant and an observer used to estimate the disturbances affecting the plant – note that both errors in the estimation of the properties of the inputs and Modeling errors can be modeled as a disturbance acting on the system. The basic idea is to use the model of the system to calculate what control action to apply, so that it counteracts the effect of the estimated disturbance.

For SISO systems, the implementation of this idea is quite simple, as all that is needed is to use the model to compute how much to increase or decrease the value of the only input of the system. The EWMA controller presented in Chap. 3 is a classical Run-to-run controller that applies this idea. For MIMO systems, however, the scenario is a bit more complex, as there is more than one input in the system and it might not be evident which of those inputs should be adjusted to drive the outputs close to their references.

Let us initially suppose that we do not explicitly consider the action of a disturbance on the system and the static model that relates the inputs and outputs of the MIMO system is linear and given by:

$$y = Bu$$

with $y \in \mathscr{R}^n$, $u \in \mathscr{R}^m$ and $B \in \mathscr{R}^{n \times m}$. If the model is accurate and the plant does not vary its parameters, this relation is true for all the iterations that are carried out, so we can particularize it for the batches characterized by indexes k and $k+1$:

$$\begin{aligned} y_k &= B\, u_k, \\ y_{k+1} &= B\, u_{k+1}. \end{aligned} \tag{4.35}$$

Analogously to what was done in the MPC formulation, we can define the change of control action Δu_{k+1} as

$$\Delta u_{k+1} = u_{k+1} - u_k. \tag{4.35a}$$

Using this definition and subtracting the expressions in Eq. (4.35), we can express the value of the output in the iteration $k+1$ in terms of its value in the previous iteration and the change of control action performed

$$y_{k+1} = y_k + B\,\Delta u_{k+1}. \tag{4.36}$$

Let us assume that we denote as T the target value for y, then we can find the value of Δu_{k+1} we need to apply using (4.36) directly

$$\Delta u_{k+1} = B^{\dagger}\big(T - y_k\big), \tag{4.37}$$

where $B^{\dagger}$ denotes a pseudo inverse of B. If B has full row rank, then this inverse is given by

$$B^{\dagger} = B^T\,(B\,B^T)^{-1} \tag{4.38}$$

and provides the value of Δu_{k+1} with minimum Eucledian norm (Greville 2003). Of course, in order to apply this method in practice, the value of u_k must be recorded, so that u_{k+1} can be reconstructed using Δu_{k+1} and Eq. (4.35a).

If we define the error incurred in each iteration as

$$e_k = T - y_k,$$

we can use Eq. (4.35) and Eq. (4.35a) to analyze the convergence conditions of the algorithm, leading to the equation

$$e_{k+1} = \big(I - B\,B^{\dagger}\big)e_k. \tag{4.39}$$

Given that this is a matrix equation, the vector error is guaranteed to converge to zero if all the eigenvalues of the matrix $I - B\,B^{\dagger}$ have a norm that is less than 1. Note that this convergence depends only on the accuracy of the model B; if B were perfect, then $B\,B^{\dagger} = I$ and the error reaches zero in one iteration. However, this approach does not offer any parameter that can be tuned to modify the behavior of the controller.

This parameter can be included in the system explicitly considering that a disturbance is acting on the system and introducing an observer to estimate it. This way, the model of the system is now

$$y_k = Bu_k + \nu_k,$$

where ν_k is the disturbance acting on the system. The observer is given by

$$\hat{\nu}_{k+1} = \omega\,\hat{\nu}_k + (1-\omega)(y_k - B\,u_k). \tag{4.40}$$

Again, the fact that food processing plants are typically *fat* opens the door to using optimization problems that help to make these decisions based on minimizing

an objective function that encode additional preferences on the system behavior. This way, the Run-to-run control approach for multivariable systems is very similar to the MPC control principle, in the sense that the control action is computed solving an optimization problem each time that we have fresh data about the problem. The major difference is that, since the models are static, only one value of the control action is computed, as opposed to the whole trajectory of control actions provided by MPC.

As presented in the previous section, a classical objective function is to consider a term that penalizes the deviations of the outputs and another that limits the control signal. For the Run-to-run approach, the optimization problem can be defined as

$$\min_{u_k} \; J = \left(T - \hat{y}_k\right)^T Q \left(T - \hat{y}_k\right) + u_k^T R u_k + \Delta u_k^T S \Delta u_k \tag{4.41}$$

$$\text{s.t.} \;\; \hat{y}_k = B u_k + \hat{v}_k. \tag{4.42}$$

Here, Q weights the importance of the deviations from the target, R penalizes the value of the inputs, and S restricts the changes in the input values. If no constraints on the values of the inputs are considered, then the solution of the optimization problem is Good and Qin (2006):

$$u_k = \left(B^T Q\, B + R + S\right)^{-1} \left(S u_{k-1} + B^T Q\, (T - \hat{v}_k)\right).$$

If restrictions on the values of the inputs are considered, then the problem no longer has an analytic solution, and a numeric solution should be found.

The conditions for the stability of the controller for the unconstrained case and $S = 0$ can be found in Good and Qin (2006). The closed-loop computation formula of the observed disturbance is:

$$\hat{v}_{k+1} = (I - (I - I\omega)\xi)\hat{v}_k + (I - I\omega)(\xi T + v_k), \tag{4.43}$$

with

$$\xi = (\beta - B)(B^T Q B + R)^{-1} B^T Q + I. \tag{4.44}$$

If we denote the eigenvalues of ξ as $\lambda_j = a_j + b_j i$, the stability criterion can be expressed as:

$$\frac{a_j^2 + b_j^2 - 2a_j}{a_j^2 + b_j^2} < \omega < 1. \tag{4.45}$$

This way, the introduction of the parameter ω, analogously to the SISO case, allows to tune the behavior of the controller and be more aggressive for scenarios where the models of the system are known to be accurate, and more conservative if the uncertainty about the models is larger.

References

Camacho EF, Bordons C (2004) Model predictive control. Springer, Berlin
Good R, Qin SJ (2006) On the stability of MIMO EWMA run-to-run controllers with metrology delay. IEEE Trans Semicond Manuf 19(1):78–86
Greville AB-ITNE (2003) Generalized inverses: theory and applications, 2nd edn. Springer, Berlin

Chapter 5
Production Planning for Food Transformation Processes

The higher-level control layer presented in Chap. 4 can be used to improve the operation of a food processing plant from two different perspectives: on the one hand, it can be used to decide which product is most profitable to produce given the current features of the raw inputs; on the other hand, it can provide which values of the process variables can be used to produce the target product so that a certain objective function is optimized.

The discussion introduced in Chap. 4 made some important assumptions: All the required input materials for the production are already available, their properties are fixed, and, once that we have decided which product is the most profitable one, that is the only product that we have to produce. The first and the last assumptions disregard aspects of Production Planning that may have an impact on the profitability of the operation, while the second assumption leaves an important question unanswered: Given that the properties and prices of agricultural products typically vary through the year, it is relevant to consider when to schedule the production of the final goods so that the profit of the whole year is maximized.

This chapter presents how these facets can be modeled and included as constraints in optimization problems to optimize an operational criteria formalized in the corresponding objective function. Section 5.1 considers the application of classical ideas regarding batch Production Planning to food processing operations, while Sect. 5.2 deals with the formalization and solution of an optimization problem to provide a season-wide production plan taking into account the variations of raw input price and properties throughout the year.

5.1 Batch Production Planning

This topic deviates a bit from the hierarchical control structure presented in Chap. 1 that constitutes the backbone of the book; however, the direct applicability of these ideas and the fact that the mathematical techniques and abstractions required to apply

P. Cano Marchal et al., *Production Planning, Modeling and Control of Food Industry Processes*, Advances in Industrial Control,
https://doi.org/10.1007/978-3-030-01373-8_5

them are essentially the same as those required for the season-wide optimization of the production advocated for their inclusion in the book.

Batch Production Planning is a classical tool for the manufacturing industry. Fortunately, many of its ideas can be readily applied to food processing operations that present similar problems to the ones faced by general manufacturing operations. The simplest example where the utility of these techniques can be easily appreciated is food packaging operations; here, the typical issues that need to be addressed include when to order the incoming raw goods so that the demand is met without having inputs rot due to excessive storage time and scheduling the production of different final products limiting their stock but also assuring that excessive time is not wasted in changing between products.

The following sections introduce different models that take into account this type of problems with progressively increasing levels of detail, along with some ideas regarding their solutions.

5.1.1 mrp Model

The simplest classical model that deals with batch Production Planning is the *materials requirement planning* (mrp) model. This model was not originally formalized to be included as a constraint for an optimization problem like the ones we have considered in Chap. 4; instead, the production plan was provided using a dedicated algorithm to find the solution that allows to produce the items as late as possible while meeting the requirements of the demand. In this section, we will present a version of the problem that takes into account the same elements as the original problem, but that is formalized to be solved using the already familiar optimization structure.

The purpose of the model is to provide a production plan for a set of final goods, whose demand is considered known, that may share some of their components. These components may be bought from external suppliers or produced in-house. In the mrp literature, the term *stock keeping unit* (SKU) is employed to denote the different products and materials that are involved in the production process, be it raw inputs, semi-processed goods, or final products.

In order to establish a production plan, we must specify both the time span that the plan covers and its granularity, i.e., whether the considered time periods represent days, weeks, months. The selection of the appropriate granularity and time span depends on the purpose and level of detail of the model: If we are providing a rough estimate of the production plan for the year, then a biweekly or monthly schedule is appropriate; if, in turn, we are planning the detailed production schedule for the following month, maybe using days as the time period of the model is the appropriate choice.

Let us suppose that we have I different SKU and that we are planning the production for an horizon consisting of T time periods. Associated with each SKU and time period, we shall define a variable $x_{i,t}$ that represents what amount of SKU i is to be set for production at time period t. This way, we have T variables for each

SKU, totaling $I \times T$ variables in the whole model. Conversely, the production for each time period t is encoded in I variables, one for each of the SKU in the model.

The basic law that relates these $x_{i,t}$ variables is a simple conservation equation: The elements already available plus the produced ones must equal the demand of goods. For a final product that is sold to external parties, this can be modeled as:

$$S(i,0) + \sum_{\tau=1}^{t-LT(i)} x_{i,\tau} \geq \sum_{\tau=1}^{t} D(i,\tau). \tag{5.1}$$

Here, $S(i,0)$ refers to the stock of the SKU i that is available at the beginning of the time period considered and $D(i,t)$ is the demand for the SKU i at time t. Both $S(i,0)$ and $D(i,t)$ are external pieces of data that are required to be known in order to set up the optimization problem.

An important remark about Eq. (5.1) is that this expression actually implies T constraint equations for the optimization problem, one for each time period t. That is, in order to fulfill the demand, we must enforce that this constraint is met at each time period, not just in the final step when $t = T$. Note that the expression is an inequality, meaning that it is OK if more items of SKU i than the ones demanded at time t are available at that time. Changing this expression into an equality would imply that the number of elements of SKU i should match exactly its demand at each time instant which, besides being a much stronger constraint that greatly restrict the set of feasible solutions, does not reflect the practical situation we are Modeling.

Another important remark about Eq. (5.1) involves the limits for the summation term of $x_{i,t}$. Note that the superior limit is $t - LT(i)$. The term $LT(i)$ is the lead time for SKU i, that is, the amount of time that elapses between the placing of a supply order and the receiving of the goods. If we want to have a certain amount of SKU i at time t, then the production needs to be scheduled at time $t - LT(i)$; that is the reason of this somewhat strange summation limit. This is easily seen if we particularize the expression with a very simple example. Suppose that we are considering the initial time instant, i.e., $t = 1$ and that the lead time for the SKU i is $LT(i) = 0$; that is, the element is available in the same time period as scheduled. Then, Eq. (5.1) provides

$$S(i,0) + \sum_{\tau=1}^{1-0} x_{i,\tau} \geq D(i,1),$$
$$S(i,0) + x_{i,1} \geq D(i,1).$$

This equation means that the demand of SKU i in the first time step can be covered with the initial stock $S(i,0)$ and whatever is set to production in the first time step $x_{i,1}$.

Let us consider now that the lead time for SKU i is $LT(i) = 1$, meaning that the items are available the next time period after they are set to production. Then, Eq. (5.1) yields

$$S(i,0) + \sum_{\tau=1}^{1-1} x_{i,\tau} \geq D(i,1),$$
$$S(i,0) \geq D(i,1).$$

Here, the upper summation limit is 0, which is lower than the initial $\tau = 1$. We adopt the convention that if the upper limit of the summation is greater than the lower, then we consider that are no terms in the summation. This means that the only way that we can supply the demand at time $t = 1$ is if we have enough initial stock, as we do not have time to produce any items of SKU i because of its lead time.

Equation (5.1) applies for elements whose demand is exclusively external, meaning that they are not used to produce any other SKU. This equation can be generalized to cover the case of SKU that are used in the production of other SKU by simply including a term that accounts for this internal demand. In order to do so, we need to know what elements are required to produce each SKU. This information is provided by the *Bill of Materials* (BOM), which is a list that contains this information. In order to include this information into the corresponding constraint, the BOM is encoded into a matrix R, whose element $R(i, j)$ represents the number of elements of SKU i needed to produce an item of SKU j. This way, the amount of SKU i required to cover the demand due to its usage in other SKU at time t is given by

$$\sum_{j=1}^{I} R(i, j)\, x_{j,t},$$

that is, the sum of the elements that are required for each SKU. Note that it is not necessary to remove index i from the summation of the above equation as $R(i, i)$ can be simply considered to be 0. Including this demand term in Eq. (5.1) provides

$$S(i,0) + \sum_{\tau=1}^{t-LT(i)} x_{i,\tau} \geq \sum_{\tau=1}^{t} \left(D(i,\tau) + \sum_{j=1}^{I} R(i, j)\, x_{j,\tau} \right). \tag{5.2}$$

This equation covers all the possible cases for all the SKU in the system, simply assigning appropriate values to $R(i, j)$ and $D(i, t)$.

A fairly common situation in practice is for SKU to have a minimum production quantity, that is, a minimum amount of SKU that are available when an supply order is created, be it the minimum amount provided by an external supplier or the minimum batch size for in-house production. This lower limit is usually denominated *minimum lot size* and designated by $LS(i)$.

The inclusion of this constraint means that the minimum amount of produced goods of SKU must be, at least, $LS(i)$. Let us consider the constraint

$$x_{i,t} \geq LS(i).$$

This equation forces the production of $x_{i,t}$ to be larger than the minimum lot size *for each time instant*. If we were to schedule production in the time instant t, then this constraint correctly models our requirement; however, if we were not to schedule production for that period, this constraint still enforces that the production is at least $LS(i)$, which is undesirable. The behavior that we really want to model is that whenever we produce SKU i, we need to produce at least $LS(i)$; however, we may very well not produce any amount of SKU i at a given time period t, so $x_{i,t}$ may be 0.

In order to correctly model this behavior, we need to introduce the so-called *indicator variables* that encode whether there is production scheduled for SKU i at time t or not. This variables are denoted $\delta_{i,t}$ and are binary. The associated constraints are

$$\begin{aligned} x_{i,t} &\geq \delta_{i,t}\, LS(i), \\ \delta_{i,t} &\geq \frac{x_{i,t}}{M}. \end{aligned}$$

Here, M is a constant such that it is greater than the total demand for SKU i for the whole production plan. This way, if $x_{i,j}$ is greater than 0, then $\delta_{i,t}$ needs to be 1, thus enforcing the constraint that $x_{i,t}$ must be greater than $LS(i)$. If, on the contrary, $x_{i,t}$ is 0, then $\delta_{i,j}$ is also 0, thus effectively disabling the minimum lot size constraint. The inclusion of these indicator variables, however, makes the problem not being linear any more, thus substantially increasing the complexity of its solution.

In order to formalize a proper optimization problem, we need to establish the objective function to be minimized. In the mrp model, the original objective was to produce the goods as late as possible while meeting the demand. This objective can be modeled using the following optimization function

$$\sum_{i=1}^{I}\sum_{t=1}^{T}(T-t)\, x_{i,t}.$$

Note that this is a linear objective function that assigns more cost to elements produced in earlier periods, as lower values of t correspond to higher values of $T-t$, which is the factor that represents the cost of $x_{i,t}$. Since this objective function is to be minimized, the values of $x_{i,t}$ have to be constrained into being positive to assure that the problem is not unbounded.

Gathering all the discussion above, the mrp optimization problem can be defined as

$$\begin{aligned} \underset{x_{i,t}}{\text{minimize}} \quad & \sum_{i=1}^{I}(T-t)\, x_{i,t} \\ \text{subject to} \quad & S(i,0) + \sum_{\tau=1}^{t-LT(i)} x_{i,\tau} \geq \sum_{\tau=1}^{t}\left(D(i,\tau) + \sum_{j=1}^{I} R(i,j)\, x_{j,\tau}\right), \\ & x_{i,t} \geq \delta_{i,t}\, LS(i), \end{aligned} \tag{5.3}$$

SKU	Name	Type
1	Lettuce Salad	Final Product
2	Mixed Salad	Final Product
3	Chopped Lettuce	Intermediate Product
4	Chopped Carrot	Intermediate Product
5	Chopped Red Cabbage	Intermediate Product
6	Lettuce	Raw Input
7	Carrot	Raw Input
8	Red Cabbage	Raw Input

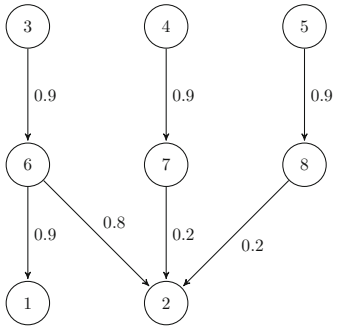


Fig. 5.1 Definition of SKUs and bill of materials for the salad manufacturing example

Table 5.1 Demands for the salad manufacturing example

SKU	1	2	3	4	5	6	7	8	9	10	11	12
1.0	10.0	30.0	10.0	20.0	30.0	20.0	30.0	40.0	20.0	10.0	20.0	10.0
2.0	10.0	20.0	50.0	20.0	80.0	40.0	10.0	90.0	20.0	60.0	70.0	20.0

$$\delta_{i,t} \geq \frac{x_{i,t}}{M},$$
$$\delta_{i,t} \in \{0, 1\},$$
$$x_{i,t} \geq 0.$$

Depending on the specific application, it is a common requirement to have $x_{i,t}$ assigned integer values. Enforcing that condition complicates the solution of the optimization problem; however, it is usually not really necessary to actually include this integrality constraint, as if the value of $x_{i,t}$ is large enough, it is usually acceptable to just round it to the closest integer without introducing errors that are substantially larger than the uncertainty in the values of the parameters of the problem.

In order to illustrate the method with an example, let us consider a facility that produces packaged salads out of raw agricultural products. Let us further assume that it produces two final products: lettuce salad and mixed salad that include lettuce and two other ingredients: carrot and red cabbage. The production process consists of washing and chopping the incoming products and packing the ingredients that constitute the final products. Figure 5.1 includes a table with the definition of the different SKUs considered in the process and a graph that depicts the BOM, while Table 5.1 contains the demand for each output SKU and each of the 12 time steps consideredin the planning.

Tables 5.2 and 5.3 show the solutions to the Production Planning problems for two scenarios; Table 5.2 shows a case where the minimum lot requirements for each SKU are low (scenario I), while Table 5.3 shows the planning when LS has much larger values for certain SKUs (scenario II). As seen in the tables, the scenario I provides a

Table 5.2 Production Plan for the salad manufacturing example for scenario I

SKU	LS	LT	S	1	2	3	4	5	6	7	8	9	10	11	12
1	10	1	100	0	0	0	0	20	30	40	20	10	20	10	0
2	10	1	100	0	0	0	80	40	10	90	20	60	70	20	0
3	10	1	0	0	0	640	520	500	1000	360	580	760	260	0	0
4	10	1	0	0	0	160	80	20	180	40	120	140	40	0	0
5	10	1	0	0	0	160	80	20	180	40	120	140	40	0	0
6	10	2	0	640	520	500	1000	360	580	760	260	0	0	0	0
7	10	2	0	160	80	20	180	40	120	140	40	0	0	0	0
8	10	2	0	160	80	20	180	40	120	140	40	0	0	0	0

Table 5.3 Production Plan for the salad manufacturing example for scenario II

SKU	LS	LT	S	1	2	3	4	5	6	7	8	9	10	11	12
1	50	1	100	0	0	0	0	50	0	50	50	0	0	0	0
2	50	1	100	0	0	0	80	50	0	110	0	60	90	0	0
3	10	1	0	0	0	640	900	380	1000	500	480	720	0	0	0
4	10	1	0	0	0	160	100	0	220	0	120	180	0	0	0
5	10	1	0	0	0	160	100	0	2220	0	120	180	0	0	0
6	40	2	0	640	900	380	1000	500	2480	720	0	0	0	0	0
7	40	2	0	160	100	0	220	0	2120	180	0	0	0	0	0
8	40	2	0	160	100	0	220	0	2120	180	0	0	0	0	0

Production Plan that is more evenly spaced, while the constraints of scenario II force production to be more concentrated in certain time instants.

5.1.2 Other Optimization Problems

The mrp model introduced above does not include some aspects that are usually important to consider when planning the production of a facility, being the most relevant one capacity. Indeed, capacity constraints are an important factor to take into account as they may prevent a production plan to be actually implemented.

In order to consider capacity constraints, it is interesting to introduce the notion of *resource*. A resource is any element of the production facility that needs to be used to produce some SKU. Machines are the paradigmatic example, but the concept can also be applied to available labor hours, etc. Since the concept of resource encompasses many different elements, the units used to measure their capacity will be heterogeneous. In order to simplify dealing with this heterogeneity, it is useful to introduce the concept of *utilization fraction*, which is the fraction of the total available capacity that is required to produce one item of SKU. This way, the parameter used to encode the capacity is dimensionless.

Along with the notion of resource comes the notion of *routing*, which is the list of resources required to produce a given SKU. Analogously to the approach used for the BOM, the routing can be encoded into a matrix $U(i,k)$ that contains the utilization fraction of resource k required to produce a unit of SKU i. This way, the capacity constraint for a given resource k can be expressed as

$$\sum_{i=1}^{I} U(i,k)\, x_{i,t} \leq 1. \tag{5.4}$$

This equation provides $K \times T$ different constraints, one for each resource and time instant. If the availability of the resources is not fixed through the time horizon considered in the planning, the matrix U can be generalized into a hypermatrix $U(i,\ k,\ t)$ including one index to keep track of the time instant. Note, however, that the number of constraints is not changed by this generalization of U, as there are still the same number of resources and time steps; the only change is that the factor that multiplies $x_{i,t}$ for a given resource k is no longer the same in all time instants.

The model that is built introducing constraint (5.4) into the mrp problem given in Eq. (5.3) is called *manufacturing resources planning* and usually abbreviated as MRPII, to emphasize its difference with mrp. Note that the inclusion of these capacity constraints increases substantially the difficulty of employing a custom heuristic algorithm for finding the optimal production plan; however, from the optimization point of view, it does not complicate substantially the solution of the problem, as the capacity constraints are linear.

The mrp problem considered a minimum lot size constraint. Sometimes, this constraint is an exact model of the actual business operations, such as dealing with minimum orders from suppliers; however, some other times this constraint is used as a crude model for situations where there is no real hard bound on the minimum number of elements in a lot, as is typically the case when using this constraint to force internal production to have a minimum quantity. The reason for this is that changing the production from one product to another— usually known as *changeover*— typically requires to change some tools or to clean some machine, thus rendering the involved resources unavailable for some time; also, starting the production of a product typically involves having some substandard products while the process is being adjusted. Both phenomena imply that there is a cost associated with changeovers, thus the requirement of having a minimum lot size in order to limit them.

The constraint of having a minimum lot size can be suppressed if these costs associated with changeovers are explicitly included in the model. The simplest way of considering them is defining two new parameters: $V(i,k)$ that accounts for the amount of resource k required to start producing SKU i; and $W(i,\,j)$ that accounts for how many items of SKU i are wasted when changing over to SKU j. Note that the elements of this matrix will almost all be zero except for the diagonal that represent the elements of SKU i that are wasted when the production of that very SKU i is produced. Note that the off-diagonal entries of $W(i,\,j)$ do not have to reflect the same data as $R(i,\,j)$, as the SKUs that are required to produce wasted SKU i are already accounted for in the material requirement constraint; the off-diagonal entries of $W(i,\,j)$ should only contain additional elements that are wasted.

Introducing these new elements, the material requirement constraint can be written as

$$S(i,0) + \sum_{\tau=1}^{t-LT(i)} x_{i,\tau} \geq \sum_{\tau=1}^{t} \left(D(i,\tau) + \sum_{j=1}^{I} \left(R(i,\,j)\, x_{j,\tau} + W(i,\,j)\, \delta_{j,\tau} \right) \right),$$

while the capacity constraint is given as

$$\sum_{i=1}^{I} \left(U(i,k)\, x_{i,\,t} + V(i,k)\, \delta_{i,t} \right) \leq 1. \tag{5.5}$$

Here, the variable $\delta_{i,t}$, just as in the mrp model, encodes whether there is production of SKU i assigned to time instant t.

The changeover Modeling presented here is very simple and does not consider effects such as variable values of $V(i,k)$ depending on what product was being produced before or what happens when the production of SKU i expands several time steps. These and other advanced topics are out of the scope of this book; however, there is plenty of literature dealing with this topics. We refer the reader to the book Voß and Woodruff (2010) for further reading.

5.1.3 Solution to the Optimization Problems

The difficulty of solving an optimization problem is intimately related with the nature of its variables, constraints and objective function, so having a broad perspective on how difficult it is to solve a particular type of problem is important to make decisions about whether the inclusion of some constraints is worth the additional complexity introduced for its solution.

A general common form of expressing an optimization problem is

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \qquad i = 1, \ldots, m, \\ & h_j(x) = 0, \qquad j = 1, \ldots, p. \end{aligned} \tag{5.6}$$

From a mathematical point of view, optimization problems can be classified into classes according to the nature of their objective functions, constraints, and variables. These classes indicate the complexity of solving the problem or, equivalently, the typical size of the problem that can be solved in a reasonable amount of time. The classes of optimization problems that we are going to discuss are Linear Problems, Mixed Integer Linear Problems, Convex Problems and General Nonlinear Problems.

Linear Problems (LP). These problems have affine constraints, and objective functions, and the variables are real numbers. This class of problems can be generically expressed as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & a_i^T x \leq b, \qquad i = 1, \ldots, m. \\ & x \in \mathscr{R}^n \end{aligned} \tag{5.7}$$

Here, $c^T \in \mathscr{R}^n$ represents a vector containing the cost assigned to each element of the vector variable x and $a_i \in \mathscr{R}^n$ and $b_i \in R$ are vectors and scalars, respectively, that contain the coefficients that define the affine constraints of the problem. Note that expressing the constraint using $\leq$ does not constitute a loss of generality, as the coefficients of a_i and b can be adjusted in case that the constraint was originally expressed as $\geq$, and equality constraints can be imposed simply introducing corresponding $\geq$ and $\leq$ constraints.

The solution to these problems can be found using the simplex or interior point methods Boyd and Vandenberghe (2004), of which many reliable implementations, both commercial and open source, exist—see, for instance, Gurobi, CPLEX and GLPK. This class of problems is the simplest to solve and constitute a fundamental base for the solution of more complex optimization problems, whose solution strategy typically implies solving a series of related linear problems.

These problems are relatively simple to solve, with a standard desktop computer being able to solve general problems with hundreds of variables and thousands of

constraints in a few seconds. If the problem shows some special structure, then the size of the problem can be increased even more.

Mixed Integer Linear Problems (MILP). These problems, like Linear Problems, have affine objective functions and constraints; however, their variables are no longer exclusively real numbers, but also include binary or integer variables. This type of problems can be expressed as

$$\begin{aligned} \underset{x,\delta}{\text{minimize}} \quad & c^T x + d^T \delta \\ \text{subject to} \quad & a_i^T x + l_i^T \delta \leq b_i, \qquad i = 1, \ldots, m. \\ & x \in \mathscr{R}^n \\ & \delta \in \mathscr{Z}^p. \end{aligned} \tag{5.8}$$

If the variable δ is binary, the last constraint can be changed to $\delta \in 0, 1$. This class of problems, although innocent looking at first sight, are actually quite hard to solve, requiring the use of branch-and-bound algorithms for their solution. These algorithms require to solve a number of Linear Problems that is theoretically exponential in the number of binary or integer variables. This means that extending the number of real variables does not increase significantly the time required to solve the problem, while including additional binary or integer variables do have a large impact on the solution time. This way, it is important to formulate the models of the problem limiting the number of integer of binary variables (Pochet and Wolsey 2006).

As with linear programming, both commercial and open-source solvers are available for this class of problems; however, commercial solvers usually include a preprocessing step where different proprietary heuristics are applied in order to reduce the size of the problem and formulate it more efficiently, so for these problems there is a considerable difference between the performance of commercial solvers, such as Gurobi, versus open-source alternatives like GLPK.

Convex Problems (CP). These problems allow to have nonlinear functions both in their objective function and their constraints; however, two limitations apply:

- All the nonlinear functions need to be convex.
- The nonlinear convex functions cannot appear in equality constraints.

The general Convex Problem can be expressed as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \qquad i = 1, \ldots, m, \\ & a_i^T x = b_i, \qquad j = 1, \ldots, p. \end{aligned} \tag{5.9}$$

Here, $f(x)$ and $g_i(x)$ are required to be convex functions. A function $f(x) : \mathscr{R}^n \rightarrow \mathscr{R}$ is said to be convex if

$$f\big(\alpha x_1 + (1-\alpha)x_2\big) \leq \alpha f(x_1) + (1-\alpha) f(x_2).$$

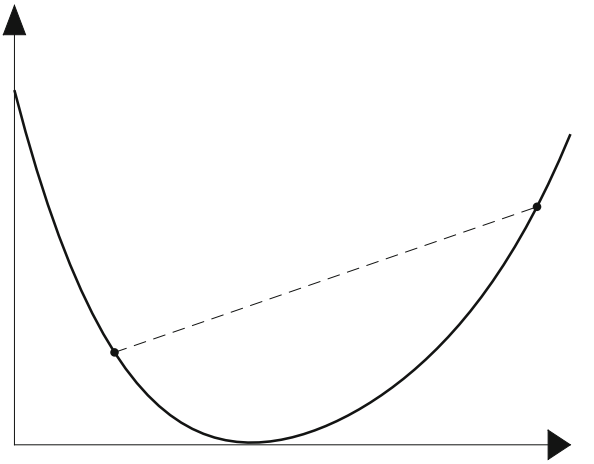


Fig. 5.2 Plot of a convex function. The segment that joins any two points always lies above the curve

The convexity condition is very easily visualized for functions of one variable, and it basically requires the function to be below the segment created joining the images of any two points in its domain. Figure 5.2 depicts this.

Convex Problem is nonlinear; however, somewhat surprisingly, they are much easily solved that MILP problems; in fact, they are only slightly harder to solve than a LP. Loosely speaking, the reasons that provide such benevolent features to this class of problems are two: On the one hand, limiting the inequality constraints to being convex assures that the feasible set is convex; on the other hand, the fact that the objective function is convex means that if we find a local minimum, it is guaranteed to be a global minimum. Note that the sign of the inequality and the fact that we are minimizing the objective function are key for the properties of the problem.

Convex Problems are a quite broad family that include some interesting problems that are well known. For instance, least squares fitting is a Convex Problem, just like (convex) quadratic programming (QP), where the objective function is quadratic and the constraints are affine, and (convex) quadratically constrained quadratic programming (QCQP), that also includes quadratic constraints.

Different software tools specific for Convex Optimization Problems are available, both solvers and Modeling tools. On the solver side, we can mention ECOS (Domahidi et al. 2013), CVXOPT (Andersen et al. 2013) and SCS (O'Donoghue et al. 2017); on the Modeling side, it is worth highlighting CVXPY (Diamond and Boyd 2016), a Modeling tool that allows to write and debug Convex Problems quite easily via its Disciplined Convex Programming functionality, that allows to assure the convexity of the constraints included in the model.

Convex Problems provide more room to tailor the objective function and the constraints of a problem to reflect the features of the system more accurately than using just affine function would without considerably increasing the complexity of the solution procedure, thus representing a very interesting alternative from a practical point of view.

General Nonlinear Problems (GNLP). This class of problems is very broad, as it includes all the problems not included in the classes defined above. These problems are typically very hard to solve, requiring long computation times to find the global optimal solution. Sometimes, however, the advantage of finding the truly global optimal solution of the problem is not significantly greater than finding a *good* solution. In this context, good solutions are local optima, i.e., points that are the optimum of a certain region of the feasible set, although not necessarily the global optimum. Algorithms that find and report local optima are called *local optimization* algorithms, while *global optimization* algorithms actually try to guarantee that the solution provided is the global optimum.

Local optimization algorithms are much faster than global optimization ones, as the global optimization algorithms need to partition the feasible space and make sure that there is no other point that offers a lower value of the objective function. In order to do this, they typically need to solve many local optimization algorithms in different regions of the domain, similar to the approach used by branch-and-bound algorithms.

The formulation of a model that results in a General Nonlinear Problem is typically best avoided if possible, as finding its solution is not an easy task at all. If the approximation of the model so that it fits some of the other classes is not easy or possible, then the modeler should aim to reduce the size of the problem, i.e., the number of variables and constraints, as much as possible. General nonlinear solvers, both local and global, can also be found, for instance, Baron (Tawarmalani and Sahinidis 2005), Couenne (Belotti et al. 2009) and PSWarm (Vaz and Vicente 2007).

Optimization is a very large and interesting topic itself; we refer the interested reader to Vanderbei (2007) for linear programming fundamentals, Boyd and Vandenberghe (2004) for convex optimization and Nocedal and Wright (2006) for general nonlinear optimization.

5.2 Production Planning for Varying Input Products

Many of the raw input products of food transformation processes, particularly agricultural goods, present an evolution in their price and features during the year. Some products are available only in a specific season, and even within that period, their prices and quality features may show significant variations.

The operation of a food transformation facility can be carried out using two different points of view; on the one hand, production-oriented operations assume that any product will be absorbed by the market and decide what products to produce based on the characteristics of the incoming raw inputs; on the other hand, market-oriented operations plan the production according to an estimation of what products are demanded by the market and thus require the raw inputs to fulfill certain standards in order to be able to produce those items demanded by the market.

Both approaches can benefit from a yearly plan that takes into account the variability of the raw inputs. For production-oriented operations, since there are no

constraints regarding what products to produce, it is clearly of interest to consider what products offer the highest return for each condition of the inputs and plan the production accordingly, as the total amount of processed inputs and processing capacity is obviously limited. For market-oriented operations, since the products are already considered fixed, it is interesting to plan the production so that the raw inputs are such that allow to produce those products while minimizing the total cost.

The approach proposed to obtain this production plan is essentially to set up an optimization problem that takes into account the evolution of the properties of the raw inputs, along with the relevant business and operational constraints, and whose solution provides the plan. The following sections detail this approach.

5.2.1 Optimization Problem Definition

In Chap. 4, we addressed the problem of how to optimally choose the production objective y_h for a given particular set of characteristics of the inputs f. We discussed the convenience of separating the problems of finding how to optimally produce a product and choosing which of those products are optimal to produce. We examined how we could define y_h so that it encoded different quality levels that are remunerated differently and that considering a remuneration proportional to a quality feature could be modeled just modifying the computation of the coefficient of the corresponding objective function.

The problem that we consider in this section is a generalization of that problem, since choosing what to produce for the whole season obviously requires choosing what to produce at a given time instant where the properties of the inputs can be considered to be fixed. However, if we assume that the total amount of inputs that the facility can produce is limited, which is a fairly common situation, then the solution of the season-wide problem requires to consider the production of all the year *at the same time* to correctly allocate the limited inputs to the time periods where production is more profitable overall.

Since we need to consider the whole season at once, we need to have an estimation of the values of the input properties for each time slot considered in the optimization problem, as their actual values are obviously not available. These estimations play a fundamental role in the approach, and they may come from simple extrapolations of historical data available at the company or be provided by sophisticated models that take into account meteorological data and other sources of information. These models can be deterministic or may include stochastic components that take into account the inherent variability and unpredictability of the conditions affecting the evolution of the crop.

The approach to formulate the optimization problem takes advantage of the concept of SKU introduced in the previous section and the decoupling of the problems of producing optimally any product and choosing which product to produce. Analogously to Sect. 5.2, let $x_{i,t}$ denote the amount of SKU i at time t. If the SKU is a

raw product, this variable requires having the corresponding stock at the factory; if the SKU is a produced item, the variable represents a production order.

The constraints that restrain the operation can also be expressed quite naturally with this definition of variables. Let us introduce the sets $\mathbb{I}$ and $\mathbb{T}$ that contain the indexes associated with the different SKUs and time steps, respectively. For this purpose, it is useful to consider that the set $\mathbb{I}$ contains two subsets: The set $\mathbb{G}$ that represents the indexes of SKUs that are inputs to the process, and the set $\mathbb{F}$ that represents the indexes of SKUs that are final products. Note that the union of $\mathbb{G}$ and $\mathbb{F}$ is not required to contain all the elements of $\mathbb{I}$, meaning that we can consider $\mathbb{I}$ to also contain intermediate products.

The fundamental constraints that govern the operations are the conservation relations that can be established between the input products and the final goods. These relations, analogously to the approach presented in Sect. 5.2, can be encoded using a matrix $R(i, j)$, whose entries represent how many elements of input SKU i are required to obtain one element of output SKU j. This way, these constraints can be written as

$$\sum_{j \in \mathbb{I}} R(i, j)\, x_{j,t} \leq x_{i,t}, \quad i \in \mathbb{I}. \tag{5.10}$$

This equation is a simplified version of Eq. (5.2) that assumes that the constituent materials are immediately available and transformed into the final goods. This constraint also implies that only the raw inputs *received* at time t can be used to produce the final goods at time t. The rationale behind this constraint is that the properties of the raw inputs *evolve* with time, so we cannot be sure that the storage of those inputs for a given period of time does not imply a change in their properties that may mean that they are no longer apt for use. If we have an estimation of how many time steps these inputs can be considered to remain in proper conditions, it is certainly possible to include these considerations in the model, simply allowing to use raw inputs from earlier time steps

$$\sum_{j \in \mathbb{I}} R(i, j)\, x_{j,t} \leq \sum_{\tau = t - ST(i)}^{t} x_{i,\tau}, \quad i \in \mathbb{I}.$$

Here, $ST(i)$ represents the number of time steps that SKU i maintains its quality.

A difference between Eqs. (5.10) and (5.2) that is worth commenting is the omission of production lead times in Eq. (5.10). The reason for this is the different focus of the optimization problems that provides different relevance to some of the aspects that comprise the production process. The focus of yearly Production Planning is to assign the production of the different products throughout the year, i.e., to obtain the *big picture* of the production of the year, so typically large time periods are considered for the time steps. With this time resolution, it is typically not relevant to consider the constraints imposed by lead times, as they are usually phenomena that takes place at a lower time scale. For the mrp and MRPII, on the other hand, lead times are a fundamental aspect of the problem, as the focus of these problems is precisely to consider this type of details of the production process.

Another set of constraints that need to be considered are the limitations on the yearly demand of the products, both in total and individually. These can be expressed, respectively, as

$$\sum_{t\in\mathbb{T}}\sum_{i\in\mathbb{F}} x_{i,t} \leq D,$$
$$\sum_{t\in\mathbb{T}} x_{i,t} \leq D(i), \quad i \in \mathbb{F}.$$

Note that considering a constraint on the demand for a SKU and time instant, that is,

$$x_{i,t} \leq D(i,t), \quad i \in \mathbb{F},$$

is not really of much practical use for the purpose of the optimization problem unless there are perishable items that need to be sold within a specified time period, thus requiring to limit the production to what is expected to be absorbed by the market at that time.

For the inputs, limitations on the availability of the products impose constraints that are structurally identical to the ones above

$$\sum_{t\in\mathbb{T}}\sum_{i\in\mathbb{G}} x_{i,t} \leq S,$$
$$\sum_{t\in\mathbb{T}} x_{i,t} \leq S(i), \qquad i \in \mathbb{G},$$
$$x_{i,t} \leq S(i,\ t), \quad i \in \mathbb{G}.$$

However, their interpretation is quite different. In this case, the limitation on the availability of a particular input SKU at a time period plays a fundamental role on the problem, as it is a natural way of implementing the implications of the raw input evolution model in the optimization problem.

The connection of the input SKUs with the final goods via the $R(i, j)$ matrix implicitly assumes that the set of features of input SKU i are appropriate to obtain output SKU j. That is, associated with each input, SKU is a known set of values for the features that the raw products need to have in order to consider that those SKUs are available. This way, setting the supply of input SKU i at time t as zero means that the corresponding raw inputs do not possess the appropriate features to qualify for the associated quality level, which, due to the relations contained in Eq. (5.10), prevent the production of the associated output SKUs.

These connections between the features of the raw inputs and the output products require some knowledge about the process, be it just a simple informal model based on the experience of the operators or a sophisticated one. Conversely, the model of the evolution of the input properties may range from a simple criterion based on past experience that determines whether it is realistic to expect a certain SKU at a certain time step, to a full model of the evolution of the properties augmented with threshold functions to determine what input SKUs are available at each time. Whatever the

case, the key idea is that the supply of each SKU are given data for the optimization problem; that is, they are parameters of the optimization problem.

The fact that the parameters of the problem are data that is available before its solution opens the door to introducing more sophisticated effects into the problem without changing its structure. For instance, it is not unusual that the transformation yield from input to outputs may vary in time; this behavior is easily modeled simply augmenting the matrix R to consider time as well, thus having $R(i, j, t)$. The structure and difficulty of solving the optimization problem remain unchanged, as all that changes are that the coefficients of some constraints are no longer the same. The values of these entries, again, may come from experience, historical data, or a model; for the optimization problem, the origin of these values is transparent.

Another important aspect of production, relevant for the focus of the problem, is capacity constraints. These can be included without substantially increasing the complexity of the model using the concept of *utilization fraction* introduced in Sect. 5.2. This way, we may define a matrix $U(i, k)$ that encodes the utilization fraction that the production of SKU i requires of resource k and express the constraint as

$$\sum_{i \in I} U(i, k)\, x_{i,t} \leq 1, \;\; t \in T.$$

Once that we have discussed the relevant set of constraints of the problem, we need to address the definition of the objective function. Let us define $C(i, t)$ as the cost associated with the SKU i at time t and let $P(i, t)$ denote the unitary income of selling SKU $x_{i,t}$. With these elements, the objective function can be written as

$$\underset{x_{i,t}}{\text{minimize}} \quad \sum_{t \in \mathbb{T}} \sum_{i \in \mathbb{I}} \big(C(i, t) - P(i, t)\big)\, x_{i,t}. \tag{5.11}$$

This objective function captures the essence of what we are trying to achieve. SKUs that represent final products have positive values of $P(i, t)$, so the associated negative sign will induce the corresponding $x_{i,t}$ to be as large as possible. Conversely, input SKUs have positive $C(i, t)$, so the positive sign drives those variables toward their minimum possible values. The summation over all the time steps forces to take into consideration the whole season.

Note that both $C(i, t)$ and $P(i, t)$ are allowed to vary with time. The definition of these values is the task of external models that reflect the operational expenses and market conditions that the production and commercialization of the goods face. Regarding the computation of the costs $C(i, t)$, it is interesting to note the differences between input and output SKUs. For input SKUs, $C(i, t)$ should reflect the purchasing and transportation costs associated with the acquisition of the goods, while for output SKUs, $C(i, t)$ should contain the process costs, such as labor or energy, without including cost of the raw materials, as they already accounted for in the corresponding input SKU terms. Commercialization costs can be included in this term $C(i, t)$ as well or directly deduced from the sale prices, which would yield $P(i, t)$ to be interpreted not as the sale price, but as the net income per sold item. Since

$C(i,t)$ and $P(i,t)$ appear subtracted in the objective function, both approaches are absolutely equivalent, and it is up to the modeler and the idiosyncrasy of the company to choose one or another.

Collecting all the elements discussed in the section, the production planning problem can be expressed as

$$
\begin{aligned}
\underset{x_{i,t}}{\text{minimize}} \quad & \sum_{t\in\mathbb{T}}\sum_{i\in\mathbb{I}} \big(C(i,t) - P(i,t)\big)\, x_{i,t} \\
\text{subject to} \quad & \sum_{j\in\mathbb{I}} R(i,j)\, x_{j,t} \leq \sum_{\tau=t-ST(i)}^{t} x_{i,\tau}, \quad i \in \mathbb{I}, \\
& \sum_{i\in I} U(i,k)\, x_{i,t} \leq 1, \quad t \in T \\
& \sum_{t\in\mathbb{T}}\sum_{i\in\mathbb{F}} x_{i,t} \leq D, \\
& \sum_{t\in\mathbb{T}} x_{i,t} \leq D(i), \quad i \in \mathbb{F}, \\
& x_{i,t} \leq D(i,t), \quad i \in \mathbb{F}, \\
& \sum_{t\in\mathbb{T}}\sum_{i\in\mathbb{G}} x_{i,t} \leq S, \\
& \sum_{t\in\mathbb{T}} x_{i,t} \leq S(i), \quad i \in \mathbb{G}, \\
& x_{i,t} \leq S(i,t), \quad i \in \mathbb{G}.
\end{aligned}
\tag{5.12}
$$

This problem, as commented at the beginning of the section, requires to have an estimation of the evolution of the properties of the inputs to be solved. However, as the season evolves, new information may be available that may refine our estimations. It is perfectly possible to take an approach similar to the *receding horizon strategy* applied in Model Predictive Control and solve the problem each time that new information may modify our estimations of the parameters of the problem. This way the optimization problem reflects all the information available up to that moment and optimizes the production from that point on taking into account the actual events that have already occurred.

5.2.2 Models of the Evolution of the Raw Product and the Process

As presented in the previous section, the role of raw input evolution and process models is to provide the values of some the coefficients that appear in the optimization problem, namely $R(i,j,t)$, $S(i,t)$, and $C(i,t)$.

This separation between the optimization problem and the models has two important implications: On the one hand, the mathematical structure of the models used to generate the values of the coefficients for the optimization problem does not have any influence in the ease or difficulty of the solution of the optimization problem; on the other hand, the mathematical structure of the optimization problem is not altered if the type of models used for computing its coefficients is modified.

These observations invite to having a modular approach for the construction of this type of optimization problems. The first effort needs to be devoted to the identification and Modeling of the different relevant SKU and resources involved in the process. Then, the relation between the different SKU needs to be encoded with a matrix R whose entries can be obtained from the assessment of expert operators based on their experience. The same expert-based approach can be used to estimate the availability of raw inputs ($S(i,t)$) and costs $C(i,t)$. This first prototype problem can be solved, and the solutions it provides are analyzed to verify that the formulation captures all the essential effects that characterize the yearly operation of the plant.

Once that this problem is set up, the inclusion of more sophisticated means of providing the values for $R(i,j,t)$, $S(i,t)$, and $C(i,t)$ only needs to modify the files that feed the data to the optimization problem, as the problem structure remains unaltered.

If models of the high-level layer of the process are available, they can be used to compute these coefficients. The idea is to solve a set-point definition problem for each output SKU and time step with the values of the properties of the raw inputs provided by the evolution model. This way, let us denote the product features associated with the output SKU as y_h^* and the raw product characteristics as f^*; then, the solution of the optimization problem

$$\begin{aligned} \underset{r_l}{\text{minimize}} \quad & J(r_l, y_h) \\ \text{subject to} \quad & y_h = \Phi(r_l, f), \\ & f = f^*, \\ & y_h = y_h^*. \end{aligned} \tag{5.13}$$

provides the values of the set-point of the process variables r_l that allow to obtain y_h^* if the raw input features are f^*. With these values of r_l, it is typically an easy task to compute an accurate estimation of the process costs $C(i,t)$.

Furthermore, the solution of this problem can also typically provide enough information for the computation of $R(i,j,t)$. As an example, suppose that one of the elements of the vector y_h contains, besides the quality features of the product, some measure of production yield. In this case, it is convenient to split the vector into two components: one that contains the quality features (y_h^Q) and the rest (y_h^R). The idea is to *force* the quality features to have the required values, while allowing the yield related variables to be optimized once the other requirements are met. This can be expressed as

$$\begin{aligned} \underset{r_l}{\text{minimize}} \quad & J(r_l, y_h^R) \\ \text{subject to} \quad & y_h = \Phi(r_l, f), \\ & f = f^*, \\ & y_h^Q = y_h^*. \end{aligned} \tag{5.14}$$

The solution of this optimization problem, then, provides all the information required to compute the values of $R(i, j, t)$, as the yield related variables are also available.

Finally, if the above problem is unfeasible, then the constraint $y_h^Q = y_h^*$ cannot be met with the available values of f^*; this means that the values of the input properties does not allow to produce the SKU associated with the quality features defined by y_h^*, so the input SKU associated with that product should have a supply of zero for that time period.

5.2.3 Types of Optimization Problems and Solutions

As commented in Sect. 5.1, the difficulty of solving one problem is given by the number and type of constraints and variables that it includes. The basic optimization problem presented in Sect. 5.2.1 is a linear problem, as all the constraints and the objective function are affine and the variables are real numbers.

Depending on the particularities of the factory operation, additional constraints may be required to include some phenomenon that need to be captured. For instance, if contraints regarding minimum lot size need to be included, then binary variables would have to be added to the formulation in a completely analogously way to the considerations presented in Sect. 5.1.1.

Another scenario that requires the addition of binary variables is found when needing to model costs incurred by having the factory *open* that are independent of actually assigning production to that time slot. These costs are convenient when we need to assure that the production is assigned to consecutive time periods, without leaving *empty* slots.

In order to model this effect, two new sets of binary variables (δ_t^A and δ_t^B) need to be defined, along with the constraints:

$$\begin{aligned} \sum_{i \in I} \sum_{\tau \geq t} x_{i,\tau} &\leq M \delta_t^A \\ \sum_{i \in I} \sum_{\tau \leq t} x_{i,\tau} &\leq M \delta_t^B. \end{aligned} \tag{5.15}$$

Here, M stands for a constant such that is much larger than the possible production of all SKUs considered in the problem. The enforcement of these constraints induces δ_t^A to equal one if there is production in or *after* the time instant t, while δ_t^B is one whenever production is assigned during or *before* t.

Four possibilities exist for any given time instant:

- Production has not been started yet. In this case, δ^A equals 1 and δ^B equals 0.
- Production has been assigned to the considered time period. Here, both δ^A and δ^B equal 1.
- Production has already started and has not yet finished, but there is no production assigned to the considered time period. In this case, again both δ^A and δ^B equals 1.
- Production has already finished. In this scenario, δ^A equals 0, with δ^B being 1.

With this definition of constraints and variables, the absolute value of the difference between δ^A and δ^B equals 1 whenever the factory either has not started production yet or has already finished. The value of this difference is 0 when either there is production assigned to the time instant or both before and after the considered period. This way, the number of time instants where the factory is open can be computed subtracting the sum of the absolute value of the difference of δ^A and δ^B to the total number of time steps considered.

It is convenient to introduce two new sets of binary variables (ξ^A and ξ^B), along with the constraints:

$$\begin{aligned} \xi_i^A - \xi_i^B &= \delta_i^A - \delta_i^B, \\ \xi_i^A + \xi_i^B &\leq 1. \end{aligned} \tag{5.16}$$

With these variables, the total number of open time periods can be computed simply by

$$n = T - \sum_i (\xi_i^A + \xi_i^B), \tag{5.17}$$

and the cost function can be augmented with a term penalizing the number of time steps when the factory is open.

The downside to the inclusion of these extra variables and constraints reside in the additional computational complexity of finding the solution to an optimization problem that is no longer linear. Obviously, the discussion presented in the previous section regarding the different types of optimization problems and their relative complexity fully applies to this application. It is up to the modeler to decide whether the additional complexity introduced by the inclusions of these extra constraints is worth the effort and the additional time required to solve the problem.

References

Andersen MS, Dahl J, Vandenberghe L (2013) CVXOPT: a python package for convex optimization, version 1.1. 6. Available at cvxopt. org, 54

Belotti P, Lee J, Liberti L, Margot F, Wächter A (2009) Branching and bounds tightening techniques for non-convex MINLP. Optim Methods Softw 24(4–5):597–634

Boyd SP, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge

Diamond S, Boyd S (2016) CVXPY: a python-embedded modeling language for convex optimization. J Mach Learn Res 17(83):1–5
Domahidi A, Chu E, Boyd S (2013) ECOS: an SOCP solver for embedded systems. In: European control conference (ECC), pp 3071–3076
Nocedal J, Wright S (2006) Numerical optimization, 2nd edn. Springer, Berlin
O'Donoghue B, Chu E, Parikh N, Boyd S (2017) SCS: splitting conic solver. version 2.0.2. https://github.com/cvxgrp/scs
Pochet Y, Wolsey LA (2006) Production planning by mixed integer programming. Springer series in operations research and financial engineering. Springer, Berlin
Tawarmalani M, Sahinidis NV (2005) A polyhedral branch-and-cut approach to global optimization. Math Program 103:225–249
Vanderbei RJ (2007) Linear programming: foundations and extensions. Springer Science and Business Media, Berlin
Vaz AIF, Vicente LN (2007) A particle swarm pattern search method for bound constrained global optimization. J Glob Optim 39(2):197–219
Voß S, Woodruff DL (2010) Introduction to computational optimization models for production planning in a supply chain. Springer, Berlin (edición: softcover reprint of hardcover 2nd ed. 2006 edition)

Chapter 6
Case Study: Virgin Olive Oil Production Process

This chapter details a case study where the ideas and approaches presented in the previous chapters are applied to a real industrial example: the production of Virgin Olive Oil (VOO), which was introduced in Sect. 1.3.1.

The application of regular control techniques to the lower-level layer is illustrated with the physical Modeling and System Identification of the heat transfer process that takes place in the thermomixer in Sect. 6.1. Then, these models are used to tune a PI controller to close a low-level feedback loop for the paste temperature in the thermomixer in Sect. 6.2; then, Sect. 6.3 shows how the application of ILC to the batch operation of the thermomixer can improve the response of the system.

Section 6.4 includes the application of fuzzy Modeling to design a controller for the olive crushing stage from a lower-level layer perspective, while Sect. 6.5 shows the application of Fuzzy Cognitive Maps for construction of a high-level fuzzy model for the whole paste preparation step.

Then, Sect. 6.6 shows how the model developed in Sect. 6.5 can be used to determine appropriate production objectives and references for the lower-level variables using the optimization setup described in Chap. 4 and how to add an observer to implement a Run-to-run approach that allows to include feedback to the higher-level layer of the plant.

Finally, Sect. 6.7 shows the application of season-wide planning to select the olive harvesting dates in order to maximize the profit of the whole year.

6.1 Traditional System Identification: Thermomixer Heat Transfer

The kneading temperature is a key element in the VOOPP, as it affects both the quality of the obtained oil and the extraction yield, so it is probably the variable that is controlled most frequently in a feedback loop in the process. In order to design a

P. Cano Marchal et al., *Production Planning, Modeling and Control of Food Industry Processes*, Advances in Industrial Control,
https://doi.org/10.1007/978-3-030-01373-8_6

Fig. 6.1 Picture of an industrial olive oil production plant

proper controller for this control loop, it is interesting to have a reasonable model of the behavior of the temperature of the olive paste.

6.1.1 Physical Modeling of the Heat Transfer

Figure 6.1 shows a picture of an industrial olive oil production plant. The thermomixer is a machine that is composed of a stainless steel tank equipped with heat exchanging surfaces and a motor that impulses the set of blades used to gently stir the paste. Traditional thermomixers are typically composed of several connected vessels in order to increase the residence time of the paste.

An important skill for Modeling is the ability to focus on the phenomena relevant to the purpose of the model, disregarding other elements that, although might be essential to fully understand all the facets of the process, do not exert a capital influence on the variables of interest. The temperature of the olive paste is determined fundamentally by the heat transference process that takes place in the thermomixer between the olive paste and the heating fluid that traverses its heating pipes; so for the Modeling of this heat transference, all the other biochemical and physical processes that take place in the thermomixer and influence the aroma, etc., can be completely disregarded, as they do not exert significant influence on the temperature of the paste.

A model of the heat transference process can be obtained applying the energy conservation law to the olive paste and the heating fluid. If we consider a single

vessel of a thermomixer, the energy conservation for the olive paste can be written as

$$\frac{d}{dt}\big(m(t)\, c_{pp}\, T(t)\big) = F_i(t)\, c_{pp}\, T_i(t) - F_o(t)\, c_{pp} T(t) + \dot{Q}(t) + \dot{Q}_l(t). \tag{6.1}$$

Here, c_{pp} is the specific heat of the olive paste, F_i and F_o are the paste mass inflow and outflow, respectively, m represents the total paste mass inside the container, and $\dot{Q}$ and $\dot{Q}_l$ stand for the heat transference with the heating fluid and the ambient. In turn, the mass conservation law provides the relation that connects the flows with the total mass in the vessel

$$\frac{d}{dt} m(t) = F_i(t) - F_o(t), \tag{6.2}$$

The expansion of the first term of (6.1) and the substitution of this Eq. (6.2) provide

$$m(t)\, c_{pp}\, \frac{d}{dt} T(t) = F_i(t)\, c_{pp}\, \big(T_i(t) - T(t)\big) + \dot{Q}(t) - \dot{Q}_l(t). \tag{6.3}$$

This equation supplies the relation that links exclusively the change of the temperature with the incoming paste temperature and the heat exchanges with the heating fluid and the ambient. A similar balance can be applied to the heating fluid, providing

$$m^w(t)\, c_p\, \frac{d}{dt} T^w(t) = F_i^w(t)\, c_p\, \big(T_i^w(t) - T^w(t)\big) - \dot{Q}(t) - \dot{Q}_l^w(t). \tag{6.4}$$

The negative sign in $\dot{Q}$ accounts for the convention that heat transfer term will be signed so that $\dot{Q}$ is positive if heat is transferred from the fluid to the paste. The known variables in Eq. (6.3) are the input temperatures of the paste T_i and heating fluid T_i^w, while T, T^w, $\dot{Q}$, $\dot{Q}_l^w$, and $\dot{Q}_l^w$ are unknown. These elements can be related using heat transfer laws if we introduce some additional assumptions on how these processes are carried out.

The most important term is $\dot{Q}$, that is essentially the term driving the temperature increase of the paste. The construction details of thermomixers typically vary for different manufacturers and models; however, a reasonable assumption that has been employed in the literature is to consider that the driving term for the heat exchange is proportional to difference of output temperatures for the water and the paste (Bordons and Núñez-Reyes 2008)

$$\dot{Q} = AU\big(T^w - T\big). \tag{6.5}$$

The ambient loss terms are essentially convection processes, so a simple proportional term is a reasonable model

$$\begin{aligned} \dot{Q}_l &= k(T - T_a), \\ \dot{Q}_l^w &= k^w(T^w - T_a). \end{aligned} \tag{6.6}$$

Here, T_a represents the ambient temperature and k and k^w are parameters that account for the surface and convection coefficients. Equations (6.5) and (6.6) can be used to express (6.3) and (6.4) exclusively in terms of flows and temperatures

$$\begin{aligned} m(t)\, c_{pp}\, \frac{d}{dt} T(t) &= F_i(t)\, c_{pp}\, \left(T_i(t) - T(t)\right) + AU\left(T^w(t) - T(t)\right) - k\left(T(t) - T_a(t)\right) \\ m^w(t)\, c_p\, \frac{d}{dt} T^w(t) &= F_i^w(t)\, c_p\, \left(T_i^w(t) - T^w(t)\right) - AU\left(T^w(t) - T(t)\right) - k^w\left(T^w(t) - T_a(t)\right) \end{aligned} \tag{6.7}$$

These equations, along with the mass balance Eq. (6.2) and the corresponding one the heating fluid, provide a nonlinear state-space representation of the heat transference system. The system is nonlinear due to the product of flow and mass variables with temperature variables; if the flows and masses were to be considered constant, then the system would be represented by a linear model due to the assumptions made in the heat transfer terms.

6.1.2 System Identification Approach

The online measurement of the temperature of the paste does not represent any technical challenge, so a data-driven approach based on traditional System Identification techniques is a viable alternative for the construction of a model of the system.

The nonlinear nature of the system makes it important to select properly the conditions of the System Identification experiments, as the accuracy of the obtained models degrades as the process operates further from the identification conditions.

As presented in the previous section, the nonlinearity of the process is due to the masses and flows acting as multiplicative factors of the temperatures; thus, having these terms constant eliminates the nonlinearities for the model. Taking this into account, it is reasonable to identify the system in a scenario where the mass in the thermomixer is constant and the input flow of mass is held steady as well; furthermore, this is the normal operating condition of the plant. The heating water flow, however, cannot be held constant, as it constitutes the manipulated variable. With this in mind, the operating point for the System Identification is chosen to be that defined by having the heating flow in an intermediate position through its operational range.

The identified system is a thermomixer composed of three vessels of 1500 kg of capacity each. The input signal for the identification experiment was a pseudorandom binary signal with its mean matching the middle of the operation range and an amplitude covering as much of the range as possible.

The models were identified using subspace identification with the `n4sid` function of the System Identification Toolbox of MATLAB. The insight offered by the physical Modeling of the previous section suggests choosing first-, second-, and third-order models for the temperatures of the first (T_1), second (T_2), and third (T_3) vessels of the thermomixer, respectively. The identification of the models with these orders for T_2 and T_3 provided models with zeros and poles very close to each other

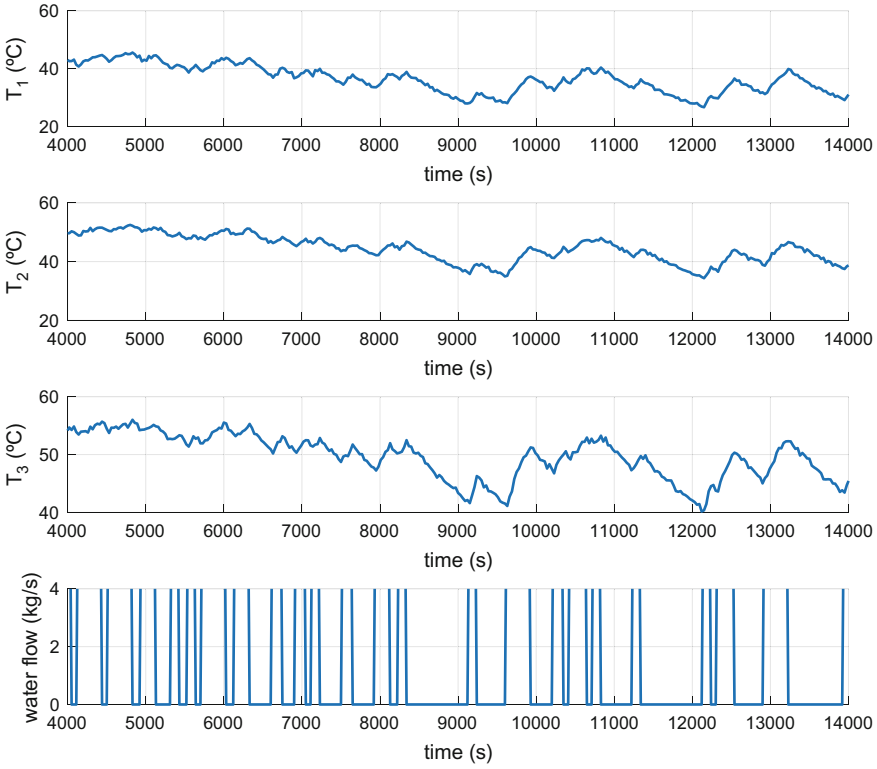


Fig. 6.2 Data for the identification of the paste temperature model

that were even complex for the third-order system. This way, first-order models were identified for all the systems.

Figure 6.2 shows the data obtained for the identification of the system, while Fig. 6.3 depicts the validation set and the output of the identified models. As seen in the figure, the behavior of the models is quite satisfactory, as they capture the fundamental time constant and dynamics of the processes. The identified gains offer a larger error; however, given the nonlinear nature of the plant, this could be expected. The process models identified will be used in Sect. 6.2 to design a PI controller for the temperature loop. The identified models are included in Table 6.1.

Table 6.1 Identified models for the temperature in the three vessels of the thermomixer

Variable	Model
T_1	$G_1(s) = \frac{2.43}{527.08s+1}$
T_2	$G_2(s) = \frac{2.42}{791.15s+1}$
T_2	$G_3(s) = \frac{1.43}{502.28s+1}$

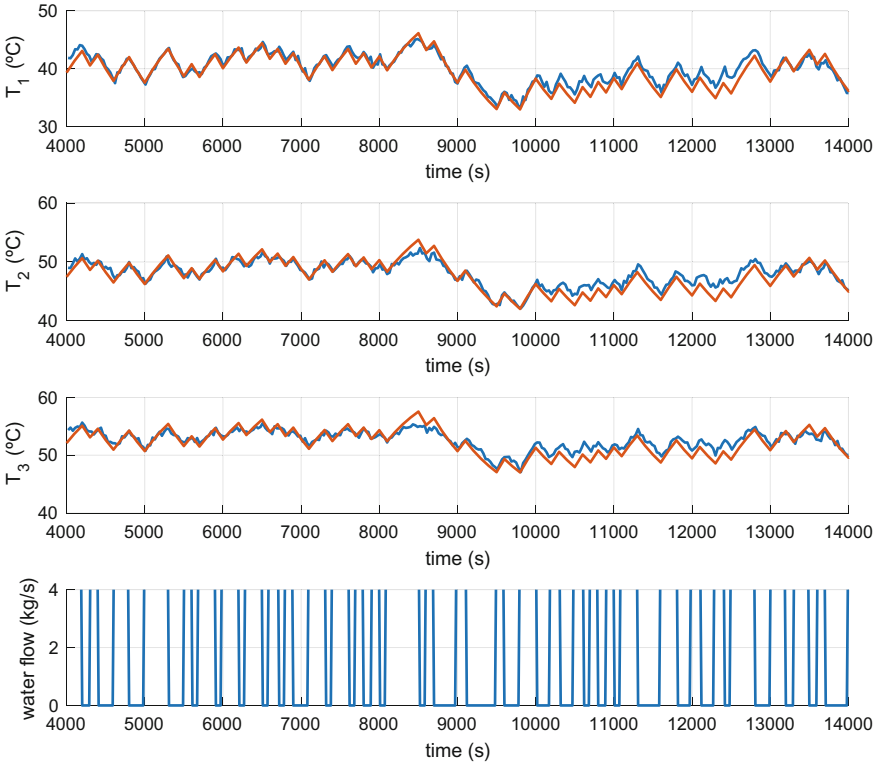


Fig. 6.3 Validation data and model output for the identified models of the paste temperature

6.2 Continuous Process Control: Thermomixer Temperature

The control of the temperature of the paste in the thermomixer is the most common control loop in olive oil factories. Given the slow but simple dynamics of the process, this control loop can be successfully controlled by a PI using the models identified in Sect. 6.1 can be used to tune it. The output variable chosen was T_3, as it is the temperature of the paste just before it is injected into the decanter.

The method employed for the tuning of the controller was the pole placement approach presented in Sect. 3.1.1.3, as it typically offers good disturbance rejection capabilities, which is the main concern for the continuous operation of the thermomixer, as changes in the set-point are not too frequent. For this method, the design parameters are the natural frequency and the damping ratio of the closed loop. Since the time constant of the identified system for T_3 was around 500 s, we selected a natural frequency for the controlled system of $\frac{1}{300}$ and a damping ratio of 0.7.

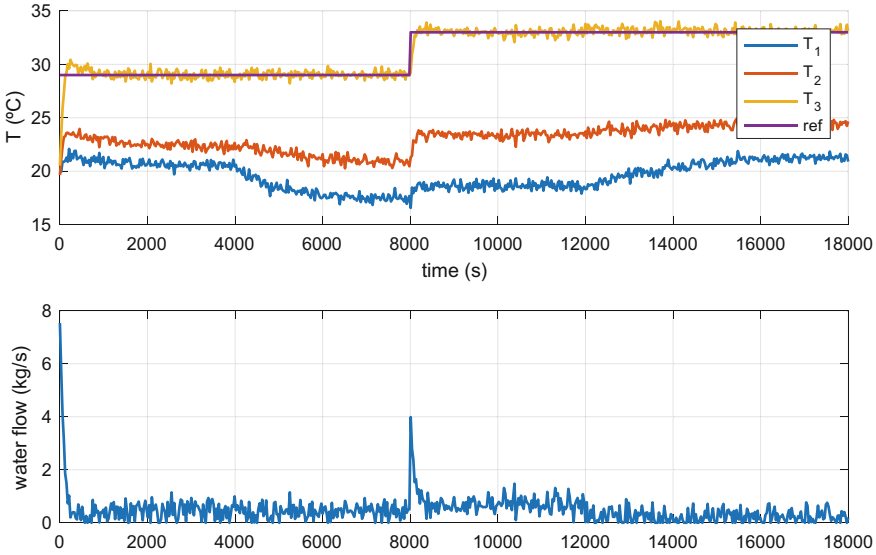


Fig. 6.4 Response of the temperatures in the thermomixer under PI controller

Figure 6.4 shows the performance of the controlled system. As seen in the plot, the response of the controlled variable is quite good, and disturbances acting on the system are rejected. At time $t = 4000$, a sudden drop of 5 °C of the temperature of the input paste flow affects the system, and while T_1 clearly shows a decrease of its value, T_3 remains at its prescribed value without even requiring too much change of the manipulated variable, as the system naturally dampens this type of disturbances. At time $t = 1200$, the inflow of paste is suddenly decreased, again having some influence on T_1 but being completely rejected in T_3 by a slight decrease of the value of the manipulated variable.

The noise affecting the temperature measurements, however, provokes some undesirable relatively high-frequency variations of the manipulated variable that contribute to wearing out the control valve. These variations can be decreased introducing some filtering of the error signal before feeding it to the PI controller. Figure 6.5 shows the behavior of the system when a first-order filter is included in the control loop. The control signals show a lower level of high-frequency variations at the expense of a slightly higher overshoot in the set-point changes. If this overshoot were too high, the inclusion of set-point weighting in the controller could improve this behavior, as demonstrated in the next section.

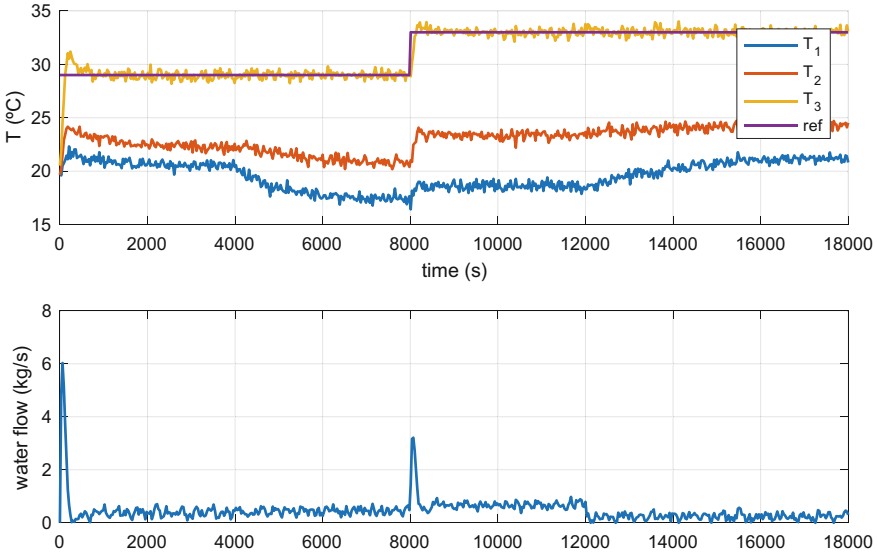


Fig. 6.5 Response of the temperatures in the thermomixer under PI controller when a low-pass filter for the noise is included

6.3 Batch Process Control: Batch Thermomixer

Batch operation of the thermomixer is common in countries like Italy and Argentina, where the olive growers *rent* the facilities of the factory to produce their own olive oil. In Spain, most of the production is carried out with a continuous operation of the plan; however, batch operation in small thermomixers is usually carried out when the objective of the production is obtaining a very high-quality product. Batch production, unlike continuous operation, requires to fill up and completely empty the thermomixer vessel very frequently—in fact, once per batch—which causes the system to be operating at transient regimes regularly, thus requiring more attention to these transient periods than the continuous operation.

Figure 6.6 shows the evolution of the temperature of a batch for two different PI controllers with the same values of K_p and T_i. The blue line shows the performance a plain PI controller tuned for the continuous operation of the thermomixer. As observed in the plot, the system shows a small overshoot that is not helpful if the objective is to obtain high-quality oil. The transient response of PID controllers, as commented in Chap. 3, can be improved using set-point weighting. The red line of Fig. 6.6 shows the response of the system when this technique is included in the controller. As depicted in the figure, the rise time of the controlled system remains basically unaltered, while the settling time is improved and the overshoot

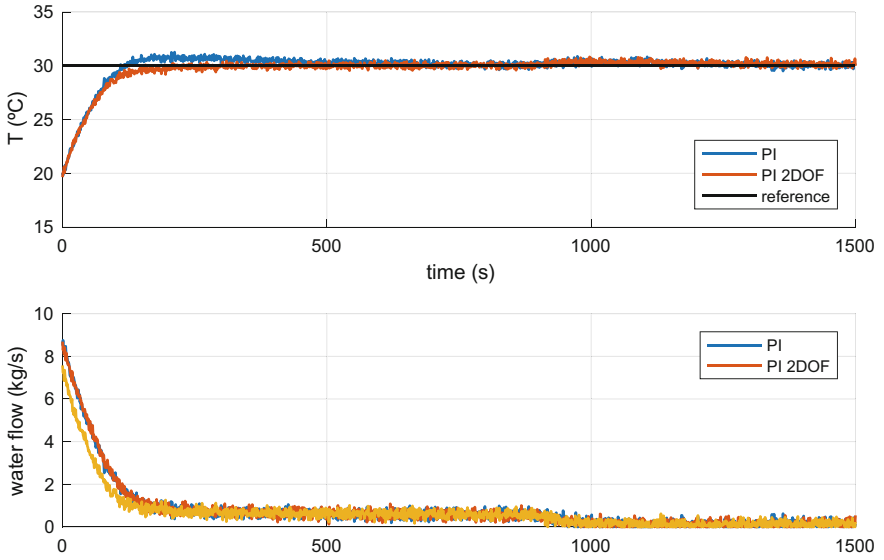


Fig. 6.6 Response of the temperature of a batch thermomixer controlled by a PI controller (blue) and a PI with set-point weighting (red)

is completely eliminated. A close inspection of the plots shows a slight bump in the temperature a bit earlier than 1000 s and a corresponding decrease of the manipulated variable. This effect is due to stopping the inflow of mass into the vessel when it reaches its prescribed level, and shows that the disturbance rejection capabilities of both controllers are identical, as is expected from the theory presented in Sect. 3.

The batch nature of the process naturally provides a setup that allows to apply an alternative approach, based on Iterative Learning Control, to improve the performance of the operation. Figure 6.7 depicts the behavior of the system when an ILC is introduced to modify the references passed to the PI controller without set-point weighting. Surprisingly, the response of the system is progressively degraded in each iteration, as the overshoot that was supposed to be decreasing is actually increasing. The reason for this unexpected behavior can be deduced observing the behavior of the manipulated variable for the successive iterations. As depicted in the plot, the manipulated variable is saturated, and this behavior is due to the integrator windup phenomenon.

Figure 6.8 shows the behavior of the system when an antiwindup scheme is introduced in the PI controller. As observed in the figure, the system quickly converges to a response that is very similar to the one obtained using the two-degree-of-freedom controller but offers a slightly better rise time. The ILC approach has the advantage that is a method that is applicable even if the structure of the main controller of the plant is fixed and does not offer the possibility of using set-point weighting, as all

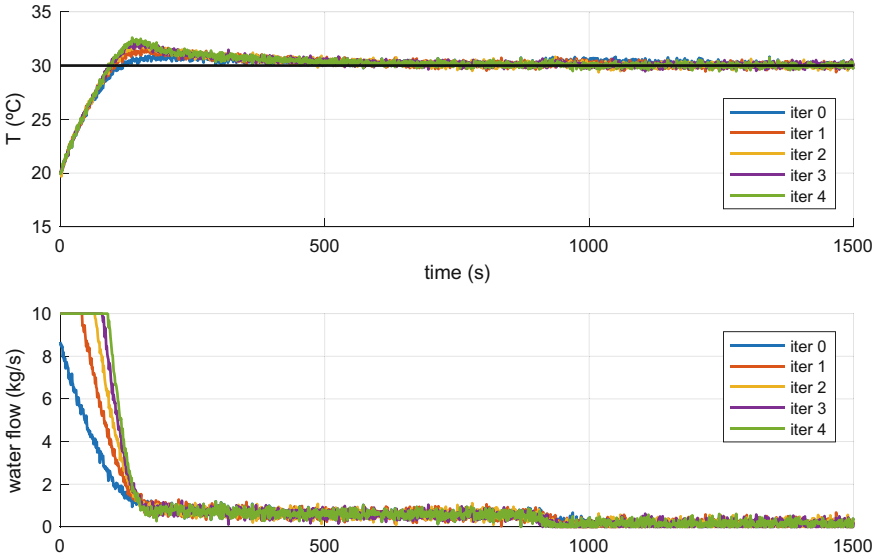


Fig. 6.7 Response of the system when Iterative Learning Control is applied using a PI without antiwindup

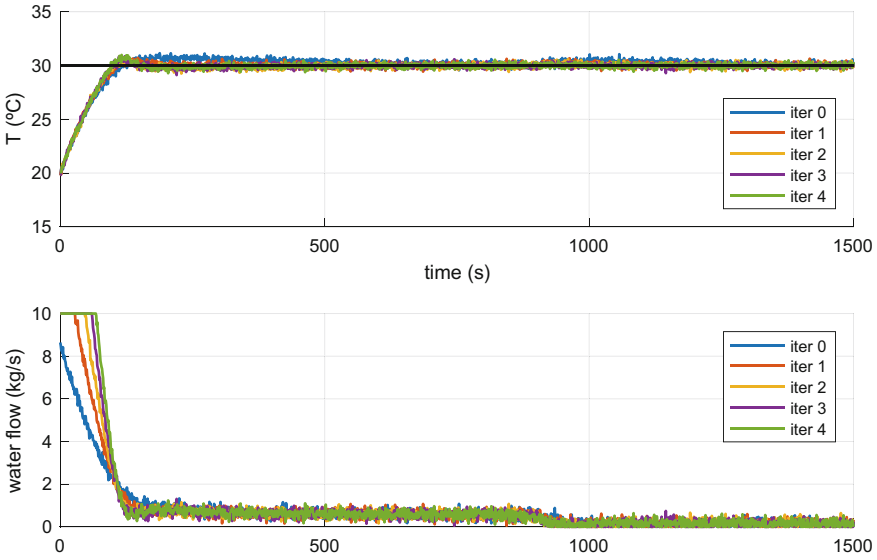


Fig. 6.8 Response of the system with the application of Iterative Learning Control when an antiwindup scheme is included into the PI controller

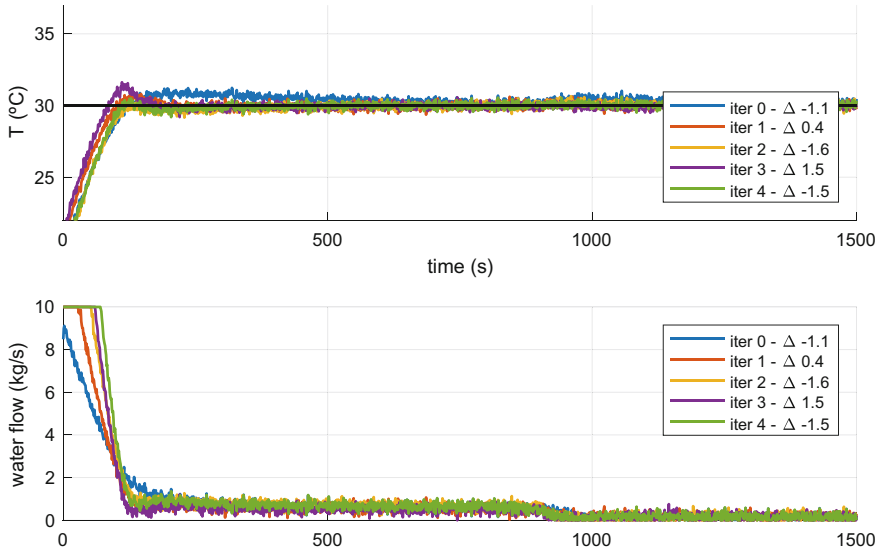
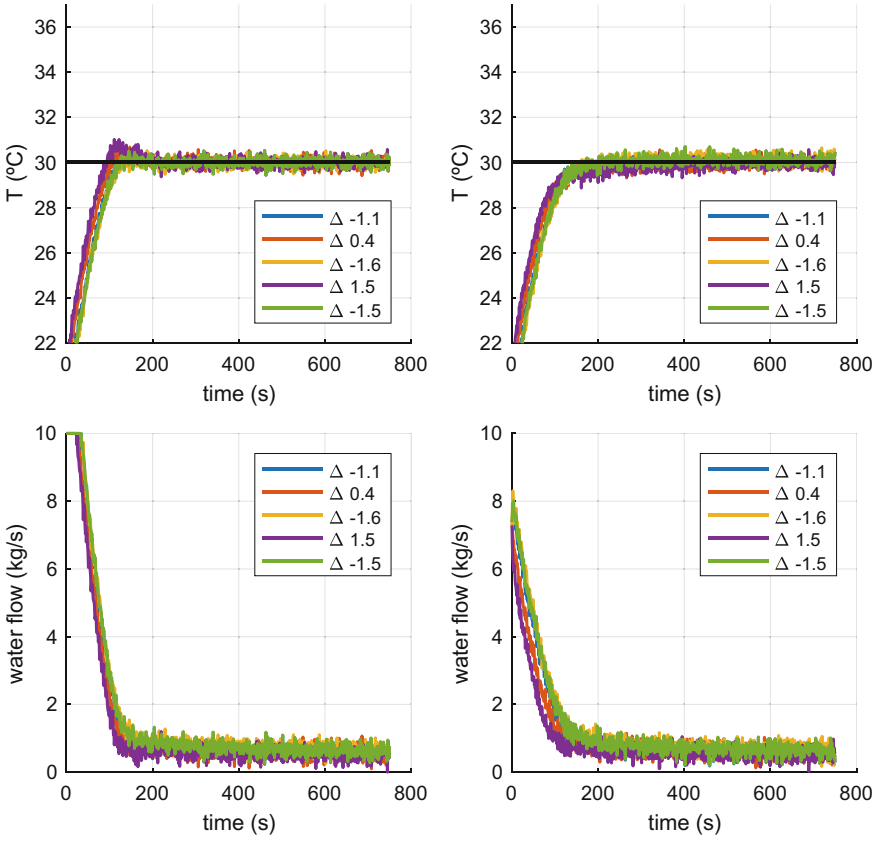


Fig. 6.9 Response of the system when perturbations act during the learning iterations of the Iterative Learning Control approach

that is required to implement it is the possibility of updating the references fed to the controller.

One of the main drawbacks of ILC is that it theoretically requires to have identical initial conditions for each batch, which leads to problems in the convergence of the controller if this condition is not met. Figure 6.9 shows the response of the system when there are variations in the incoming temperature of the paste. A seen in the plot, the convergence of the algorithm is no longer as smooth as the case shown in Fig. 6.8. There is, however, an important practical point that is worth emphasizing: The variations in the incoming temperature affect the convergence of the algorithm if these variations occur while the controller is *learning*, and, as seen in Fig. 6.8, it converges quite quickly, so two or three iterations are enough.

If these variations occur once that the learning mechanism has been switched off, then the response of the system is equivalent to the way a noniterative controller may behave. Figure 6.10 compares the responses of the ILC and the PI controller with set-point weighting. As shown in the plots, both approaches show acceptable behavior, with the ILC approach providing a bit more aggressive response that yields some overshoot but offers better rise and settling times.



(a) ILC. (b) Set-point weighting.

Fig. 6.10 Response of the system under ILC and set-point weighting control when perturbations act after the ILC learning is finished

6.4 Fuzzy Low-Level Modeling: Crusher

The crushing step is an important part of the paste preparation process that, as will be presented with more detail in Sect. 6.5, affects both the yield and the quality of the produced oil. If we focus on the low-level operation of the crusher, the most important aspect to consider is preventing an excessively high power consumption, as operating in this condition provokes a higher wear of the crusher components and may even lead to the total blockage of the crusher.

The power consumption depends on the properties of the olives, mainly ripeness and humidity, the inflow of olives, and the size of the sieve. The size of the sieve cannot be changed without stopping the operation, as changing it requires to disassemble part of the crusher and physically change a specific part. This way, the inflow of olives

and the addition of water are the two variables that are typically used to dampen the disturbances induced by the change of the olive properties.

The control of this step is often manual, with the operators adjusting the damper of the hopper to manipulate the inflow of olives and manually adjusting a valve that regulates the flow of water that showers the olives. The rule of thumb for the operation of the crusher is to add water if the olives are dry and reduce the flow if the power consumption is too high. These premises can be easily included in a fuzzy controller that mimics the behavior of the operators. Note that, unlike the high-level Modeling that will be presented in the next section, where the model reflects the relations between the variables of the plant, the approach taken for this low-level operation is to directly model the behavior of the operator.

The reason for this approach is that the number of variables and rules that are required in the fuzzy system is modest, so it is easier to simply encode the possible situations that may arise during production and the corresponding actions taken by the operators than to model the physical relations of the plant and setting up a model-based controller based on these relations. For the high-level layer, the number of variables is much larger and leads to a prohibitively high number of possible operation scenarios, so the construction of a controller that encodes the responses of the operator to these scenarios is not tractable.

The universe of discourse and the fuzzy sets of the variables of the system are included in Fig. 6.11. As seen in the figure, three terms are considered for the olive and the water addition, five terms for the error, and seven terms for the outputs of the controller. The fuzzy sets of the controller inputs are defined using triangular membership functions, while the outputs are fuzzy singletons. Note that the humidity of the olives, although it is a very relevant parameter that influences the behavior of the system, is not included. The reason for this is that there are no off-the-shelf online sensors that provide this values, so the controller needs to be implemented without this information. The way to circumvent this difficulty is considering the relation between the inflow of olives and the power consumption. If the inflow is relatively low but the power consumption is relatively high, it must be because the moisture content of the olives is low; this logic can be easily included in the controller to use the addition of water when needed. The set of rules defined in the controller are included in Table 6.2, and the defuzzification step is carried out using the center of gravity for singletons method.

The structure chosen for the developed controller is a fuzzy PI, as it provides the *change* of the manipulated variables, thus implicitly including integral action to the system. An important parameter that influences the behavior of the controller is the sampling time used for its implementation. This sample time must be chosen so that the control action is not too sluggish but also taking into account that the system is affected by relatively high-frequency disturbances due to the inhomogeneity of the olive moisture. These disturbances are faster than the typical bandwidth offered by the damper of the hopper, so the controller should not exert too much control action trying to reject them. Another important implication of the selection of the sampling time is that, since the controller provides the *change* of the control signals,

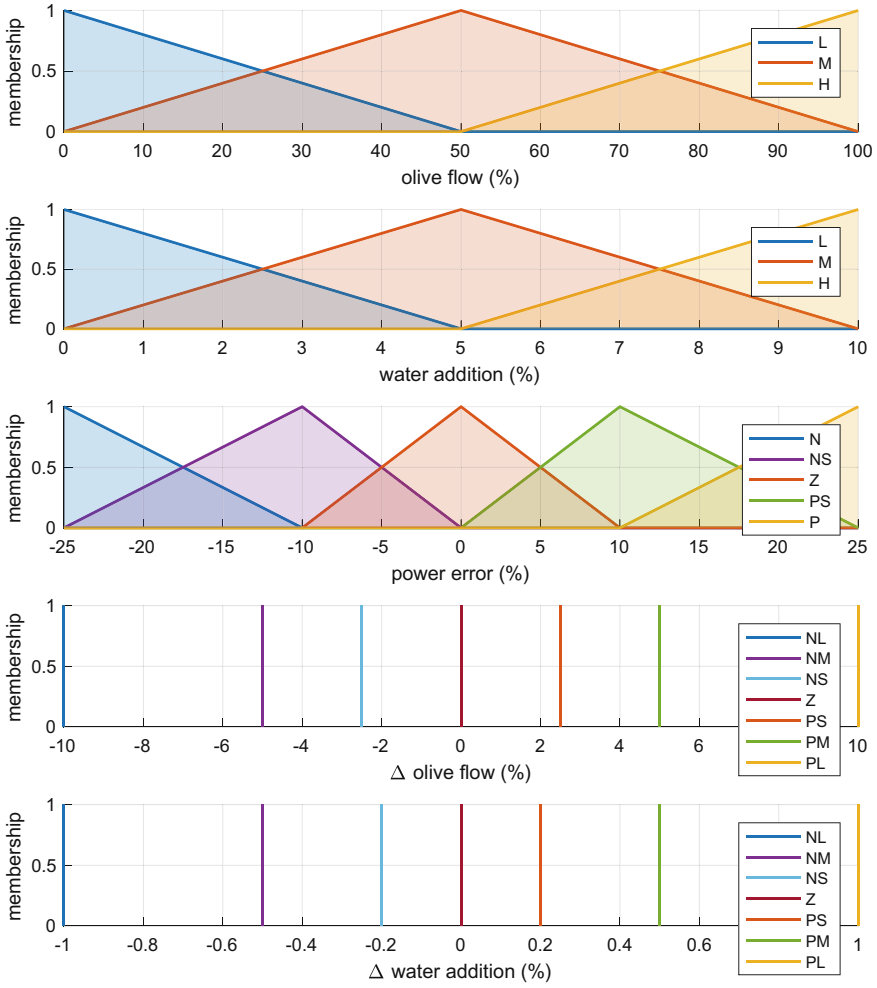


Fig. 6.11 Membership functions and universes of discourse of the variables involved in the crushing stage and included in the fuzzy controller

the selection of the output values of the fuzzy rules needs to be coordinated with this sampling time, as the effective gains of the controller depend on both parameters.

Fuzzy controllers are often designed taking into account both the error and the change of the error, but since the dynamics of the crusher are essentially first order, derivative action is not really needed to obtain an acceptable behavior of the controlled system.

Figure 6.12a shows the response of the system to a step disturbance in the moisture content of the olives fed to the crusher ($t = 200$). As depicted in the graph, this disturbance drives a decrease of the inflow of olives to the crusher and an increase of

Table 6.2 Rules of the fuzzy controller for the crusher

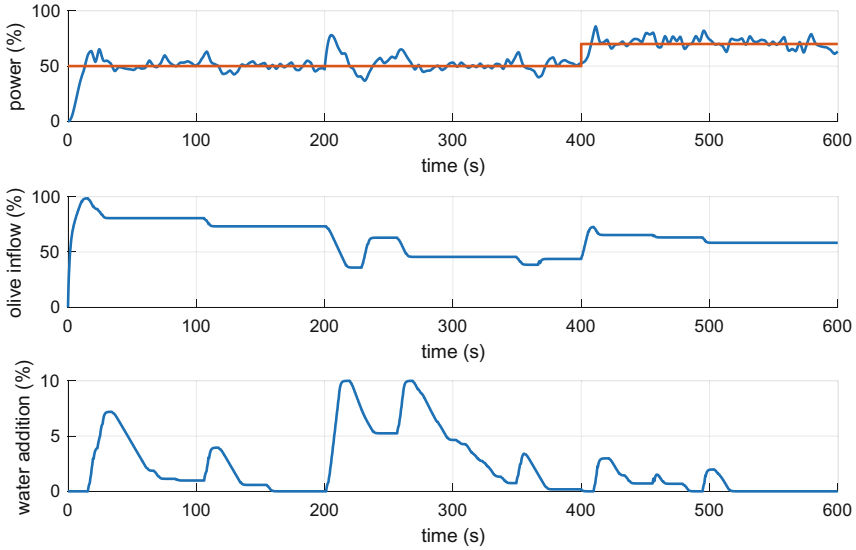
```
If W is L and Q is L and e is N Then DW is PM;
If W is L and Q is M and e is N Then DW is PL;
If W is L and Q is H and e is N Then DW is PL;
If W is M and Q is L and e is N Then DW is PM;
If W is M and Q is M and e is N Then DW is PL;
If W is M and Q is H and e is N Then DW is PL;
If W is H and Q is H and e is PS Then DW is NS;
If W is H and Q is M and e is PS Then DW is NS;
If W is H and Q is L and e is PS Then DW is NM;
If W is H and Q is L and e is PS Then DW is NM;
If W is M and Q is H and e is PS Then DW is NS;
If W is M and Q is M and e is PS Then DW is NS;
If Q is L and e is P Then DQ is PL;
If Q is M and e is P Then DQ is PM;
If Q is H and e is P Then DQ is PS;
If Q is H and e is N Then DQ is NS;
If W is H and Q is M and e is N Then DQ is NS;
If W is M and Q is M and e is N Then DQ is NS;
```

the addition of water, which is progressively decreased afterward. At $t = 400$, a step change of the reference shows an acceptable behavior of the system for the transient response, providing a higher rate of olive inflow and water addition. Figure 6.12b, in turn, shows the rejection of drift disturbances in the moisture content, again yielding an acceptable behavior. It is worth highlighting that the control signals for the actuated variables do not exhibit too much response to the high-frequency variations of the moisture, but are capable to successfully reject the low-frequency disturbances.

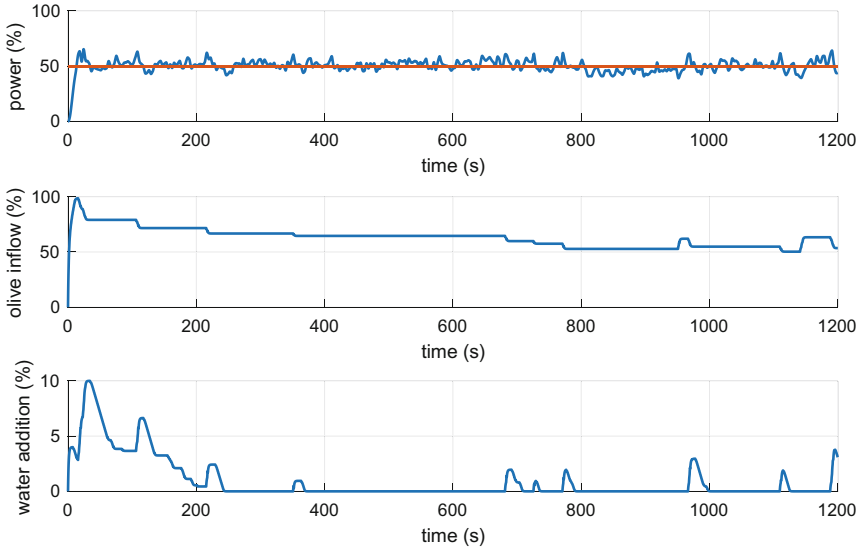
6.5 Fuzzy High-Level Modeling: Paste Preparation

This section details the application of the Fuzzy Cognitive Map Modeling technique presented in Chap. 2 for the construction of the model of the high-level layer of the paste preparation phase of the Virgin Olive Oil production process (VOOPP).

The preparation of the paste is a key step in the VOOPP, as it influences both the extraction yield and the quality of the obtained oil. The paste preparation stage consists basically of two steps: the crushing of the olives in the crusher, where the cells that contain the olive oil are broken, and the kneading that takes place in the thermomixer, where the size of the oil droplets is increased so that it is easier to separate the oil from the rest of the components of the paste.



(a) Response of the controlled system to step disturbances in the moisture content of the olives ($t = 200$) and changes in the reference ($t = 400$).



(b) Response of the controlled system to drifts in the moisture content of the olives.

Fig. 6.12 Response of the crusher with the fuzzy controller designed to control the power consumption

The construction of the system followed the expert knowledge-based approach. Interviews with a reduced number of experts on the VOOPP, along with an extensive review of the published literature on VOOPP and some years of personal professional experience in the industry, were the foundations for the construction of the model.

6.5.1 Definition of Nodes

The nodes involved in the VOOPP model can be divided into two major groups, according to the nature of the value of the variable they symbolize:

- Nodes that represent typical physical variables, such as temperature, time, size. These variables have a natural universe of discourse, namely their typical variation range for the process.
  Since the inference of the system is defined by relations between the different labels of the variables, it is important to have a good definition of the fuzzy sets and the labels defined over this universe of discourse.
  Experts were asked to provide a representative value for each label, and this value was used as the kernel of the membership function defining the label.
  The value of this type variable is typically provided by sensors, so a previous fuzzification step is required before using these data for the computation of the system outputs.
- Nodes that represent variables for which there are typically no available sensors, such as paste preparation (Kneading State (K_s)). These variables may be considered as inherently fuzzy, since it is the expert operator of the factory that determines their value based on indirect measurements, visual inspection, etc. For these variables, the input to the system is already provided fuzzified and the meaningful value of the output is also fuzzy, so the definition of the universe of discourse is arbitrary.

The nodes can also be classified according to the role of the variable they represent in the VOOPP:

- Properties of the incoming olives: this group includes Ripeness (R_f), Incoming Olive Moisture (H_o^I), and Fruit State (E_f). The value of these variables is determined by the evolution of the olives in the grove as influenced by cultivar and meteorological factors, the harvest date and the handling of the olives during the harvesting and transportation.
- Technological parameters: this group comprises all those variables whose set-points are susceptible to be specified by the operator of the factory. Examples of these variables are Kneading Time (t_b) or Sieve Size (C_s).
- Auxiliary parameters: these are parameters whose value depends on other upstream variables, and thus cannot be chosen arbitrarily, but do not represent an output variable of the process. An example of this type of variables is Paste Emulsion (P_E).

- Output parameters: these are the variables that are usually included in the production objective of the process. Examples of this type are Yield (X) and Fruity (F).

This last classification of the nodes will be particularly relevant when addressing the optimization problems to find the production objective and its corresponding set-points.

Five labels were considered for the partition of the universe of discourse of each variable. This number was selected as it represented a balanced trade-off between resolution of the model and complexity. Triangular membership functions that intersect at 0.5 membership grade were used for each term of the model for simplicity reasons and to guarantee that the sum of the elements in $S_f(v_i)$ adds up to one, so that the relative weights defined in the relations are not altered due to this factor.

6.5.2 Definition of Relations

The elicitation of the relations between variables was carried out using a two-step approach:

1. A first characterization of the relations is elicited, and a prototype system is built upon these.
2. The behavior of the system is studied, and the relations of nodes that do not show satisfactory results are fine-tuned.

For the first step, experts were asked to define the type, the sign, and the strength of the relation between the nodes. As is common practice in the construction of FCM from expert knowledge Stach et al. (2010), experts were asked to describe the strength using a linguistic term, which was afterward mapped to a numerical value according to Table 6.3. Also, experts were asked if any nonlinear effect, such as saturation, thresholds, were to be included in the relations, explicitly asking for a mapping from input to output in case these effects were present. Regular relations were translated into R_{ij} according to the structure defined in Chap. 2.

The second step involved studying the values of the nodes as the predecessor nodes swept through their universe of discourse. These obtained values were plotted in contour plots and studied to find regions of odd behavior in order to fine-tune the model accordingly.

Once this lower-level inspection of the system was finished, a more global approach was tackled, defining different scenarios of properties of the incoming olives and checking the output variables for different values of olive properties and process variables. The purpose of this study was to assure that the flow of effects across the model was correct. Some examples of these plots are included in the next section, along with the structure of the models and some comments.

Table 6.3 Definition of the weight levels employed for the VOOPP model

Influence	Value
Very strong	1
Strong	0.75
Moderate	0.5
Weak	0.25

6.5.3 *Fuzzy Cognitive Map Model of the Paste Preparation*

The graph of the paste preparation model is included in Fig. 6.13. The properties of the incoming olives are defined by the nodes Incoming Olive Moisture (H_o^I), Ripeness (R_f), Pit-Flesh Ratio (R_p), Incoming Fruit State (E_f^I), and Olive Illnes (O_I). The values of these parameters depend on their evolution in the orchards, the moment the harvesting is carried out, and the method used for the harvesting and the transportation. However, when olives arrive at the *almazara*, their values are already set, so in the model they are considered fixed value inputs.

Once in the factory, olives are fed into hoppers and the time they remain there effectively alters their properties. This effect is included in the system using the node Storage Time in Hopper (T_s), which reflects the time that olives are stored in the hoppers, and exerts influence on Fruit State (E_f) and Olive Moisture (H_o). These nodes represent the same physical variable as Incoming Fruit State (E_f^I) and Incoming Olive Moisture (H_o^I), respectively, but at the moment the olives are taken from the hoppers and fed to the following stage in the VOOPP. The storage of olives in the hoppers decreases Fruit State (E_f) and Olive Moisture (H_o), as depicted in Fig. 6.14a and b. These effects favor having a low level of Paste Emulsion (P_E), which in turn helps having good Kneading State (K_s). The price to be paid is the increase in Defect (D) and a slight decrease in Fruity (F), which decrease the quality of the obtained oil.

The crushing process is responsible for the breaking of the olive cells and thus freeing the oil. Different crusher technologies are available for the VOOPP, but, by far, the most extended is the hammer crusher. Focusing on just this type of crusher, there are still alternatives in the type and size of the sieve to be used. These alternatives are included in the nodes Sieve Type (S_t) and Sieve Size (C_s), respectively. Sieve Type (S_t) presents only three possible values, one for each of the alternatives existing in the industry.

The effect of these variables is combined into an intermediate node denominated Sieve Size (C_{se}), which in turn exerts its influence on the subsequent nodes.

Crushing Degree (G_m) is the variable that represents the resulting particle size of the olive paste, and it has a strong influence on the final yield. It depends on Sieve Size (C_{se}), as well as on some olive properties, namely Pulp Firmness (P_F) and Pit-Flesh Ratio (R_p). Pulp Firmness (P_F) is a characteristic of the olives, but can be related to Ripeness (R_f) and Fruit State (E_f), and since Fruit State (E_f) is affected

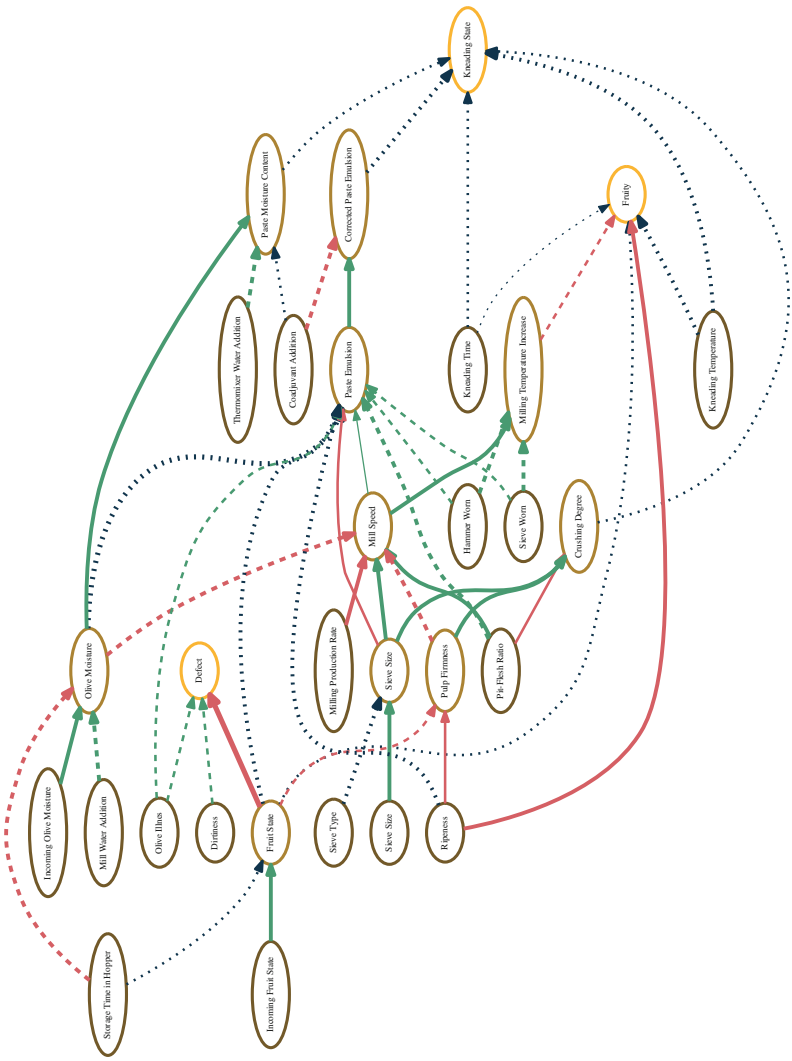
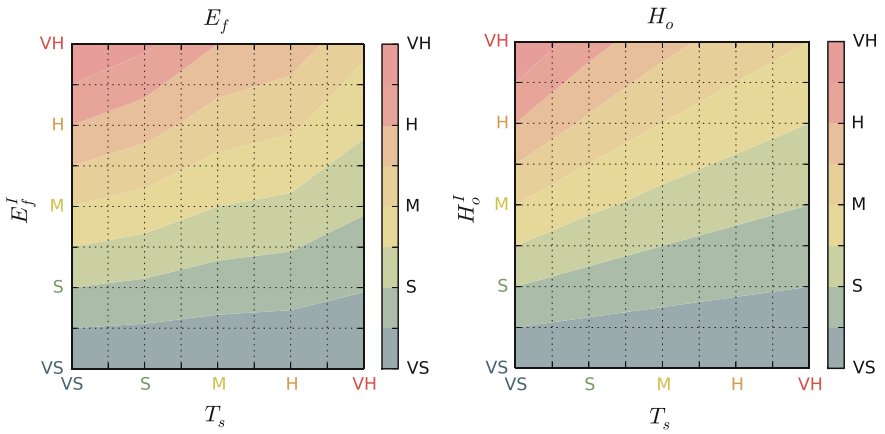


Fig. 6.13 Graph of the VOO paste preparation model. Green (red) arcs represent positive (negative) relations, continuous (dotted) lines stands for bidirectional (unidirectional) relations, and greater width of the line indicates greater value of the relation weight. Blue arcs represent relations defined by matrices of nonstandard form



(a) Fruit State (E_f) (b) Olive Moisture (H_o)

Fig. 6.14 Influence of Storage Time in Hopper (T_s) on Fruit State (E_f) and Olive Moisture (H_o), respectively. The vertical axes represent Incoming Fruit State (E_f^I) (**a**) and Incoming Olive Moisture (H_o^I) (**b**)

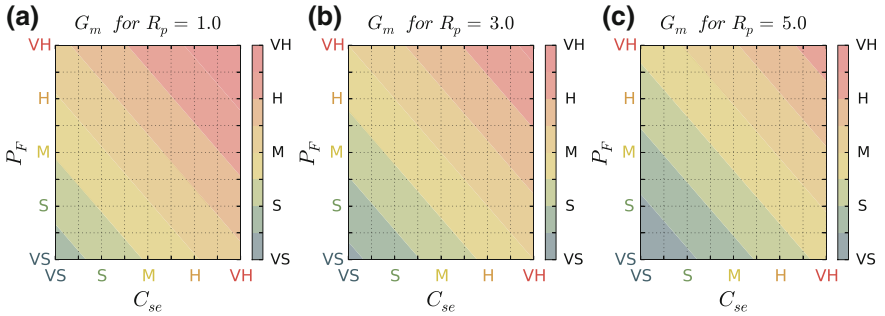


Fig. 6.15 Values of Crushing Degree (G_m) as a function of Sieve Size (C_{se}) and Pulp Firmness (P_F) for three different values of Pit-Flesh Ratio (R_p)

by Storage Time in Hopper (T_s), the value of this parameter is inferred based on these two properties of the olives. In turn, Pit-Flesh Ratio (R_p) is a characteristic of the incoming olives that is defined mainly by their variety, and is another input variable to the system. Figure 6.15 renders the values of Crushing Degree (G_m) as a function of Sieve Size (C_{se}) and Pulp Firmness (P_F) for three different values of Pit-Flesh Ratio (R_p).

Besides Crushing Degree (G_m), Paste Emulsion (P_E) is another important parameter whose value is defined by the crushing process. It is affected by all the parameters that influence Crushing Degree (G_m), plus Olive Moisture (H_o), which plays a major role in its value. Values of Olive Moisture (H_o) below a certain threshold completely inhibit the emergence of emulsions, while higher values of the parameter dramatically

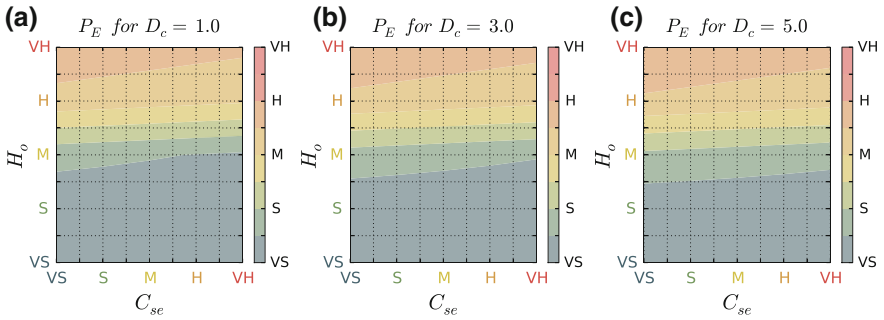


Fig. 6.16 Values of Paste Emulsion (P_E) as a function of Sieve Size (C_{se}) and Olive Moisture (H_o) for three different values of Sieve Worn (D_c)

contribute to their formation (Cert et al. 1996). Besides this, Sieve Worn (D_c) and Hammer Worn (D_h) also exert some influence in the final value of Paste Emulsion (P_E). Figure 6.16 renders the values of Paste Emulsion (P_E) as a function of Sieve Size (C_{se}) and Olive Moisture (H_o) for three different values of Sieve Worn (D_c). This figure highlights the major influence of the Olive Moisture (H_o), while the influence of Sieve Worn (D_c) is very slight.

The final step in the paste preparation process is the kneading of the paste inside the thermomixer. This is probably the most important part in the whole VOOPP, due to the influence it presents on both quality and yield (Clodoveo 2012).

If the value of Paste Emulsion (P_E) is high, then coadjuvants are added to the paste to reduce this value. The node Coadjuvant Addition (A_c) represents the amount of coadjuvant added to the paste, and Corrected Paste Emulsion (P_{EC}) symbolizes the resulting level of emulsions in the paste after the addition of the coadjuvants.

The value of Paste Moisture Content (P_H) is also very important in the process, with values too high and too low affecting negatively the Kneading State (K_s). If Paste Moisture Content (P_H) is low, then some water can be added at this stage of the process, as represented by Thermomixer Water Addition (A_B) node. If Paste Moisture Content (P_H) is too high, then the addition of coadjuvant may moderately attenuate its negative influence on the yield.

Lastly, Kneading Temperature (T_b) and Kneading Time (t_b) are the par excellence parameters that influence the kneading process. Higher values of both variables tend to increase Kneading State (K_s) and penalize Fruity (F), with a stronger influence shown by Kneading Temperature (T_b). Some nonlinear behavior of the parameters is considered, as reflected by the entries of the corresponding relation matrices included in Table 6.4.

Figures 6.17 and 6.18 illustrate the influence of Kneading Temperature (T_b) and Kneading Time (t_b) for a combination of three values of Paste Moisture Content (P_H) and Corrected Paste Emulsion (P_{EC}) on Kneading State (K_s). This figure shows that Kneading State (K_s) is worse for low and high values of Paste Moisture Content (P_H), as well as the negative influence exerted by Corrected Paste Emulsion (P_{EC}).

Table 6.4 Relation matrices for some the arcs of the VOOPP paste preparation model

Predecessor	Olive moisture (H_o)	Kneading temperature (T_b)	Kneading time (t_b)
Successor	Paste emulsion (P_E)	Fruity (F)	Fruity (F)
ρ_{ij}	1	0.75	0.25
$\mathcal{R}^{ij}$	$\begin{bmatrix} 20 & 20 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$	$\begin{bmatrix} 0 & 0.25 & 0.5 & 1.0 & 3.0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0.25 & 0.5 & 1.0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
Predecessor	Ripeness (R_f)	Kneading temperature (T_b)	Kneading time (t_b)
Successor	Fruity (F)	Kneading state (K_s)	Kneading state (K_s)
ρ_{ij}	0.75	0.75	0.5
$\mathcal{R}^{ij}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 2.0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 0.75 \end{bmatrix}$	$\begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0.75 & 0.75 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

Also, the higher weight of Kneading Temperature (T_b) compared to Kneading Time (t_b) is patent in the almost vertical transition lines shown in these plots.

Finally, Fig. 6.18 shows the values of Fruity (F) as a function of Kneading Temperature (T_b) and Kneading Time (t_b) for a combination of three values of Ripeness (R_f) and Milling Temperature Increase (Δ_{Tm}). The figure clearly illustrates the requirement of having an adequate value of Ripeness (R_f) for having high values of Fruity (F), as well as the relative low range of possible process values if very high values of Fruity (F) are to be obtained.

6.6 High-Level Control: Paste Preparation Set-Point Determination

The model developed in the previous section can be used to answer the following questions regarding the process:

1. Which production objectives are possible, given the batch of olives to be processed?
2. Which of those possible objectives should be selected?
3. What set-points of the process variables allow to reach that production objective?

The multi-objective nature of the VOOPP already became apparent in Sect. 1.3.1 during the brief description of its operations: The opposite influence of several process variables on relevant process outputs supposes having to compromise the value

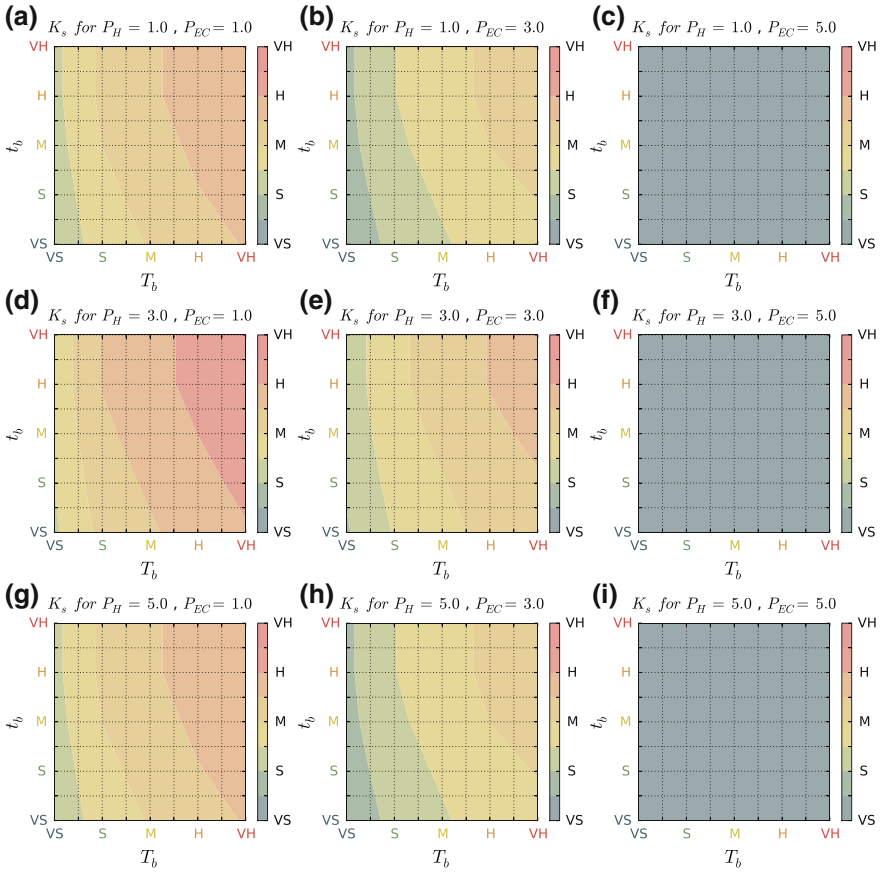


Fig. 6.17 Values of Kneading State (K_s) as a function of Kneading Temperature (T_b) and Kneading Time (t_b) for three different values of Paste Moisture Content (P_H) and Corrected Paste Emulsion (P_{EC})

of one output for the other. This multi-objective characteristic can be formalized by the definition of an objective vector, where each element of the vector represents a desirable characteristic of the output of the process, such as having high fruity, good yield, or low elaboration costs.

The existence of trade-offs between desirable characteristics of the VOO supposes that, in general, there is not a unique set of values of the process variables that concurrently optimize all the elements of the objective vector, but a Pareto boundary of nondominated objective points. These Pareto points represent those situations where an improvement in the value of an objective necessarily means a decrease in the value of another. Finding this boundary answers a slightly improved version the first of the posed questions, namely *which efficient objectives are possible?*

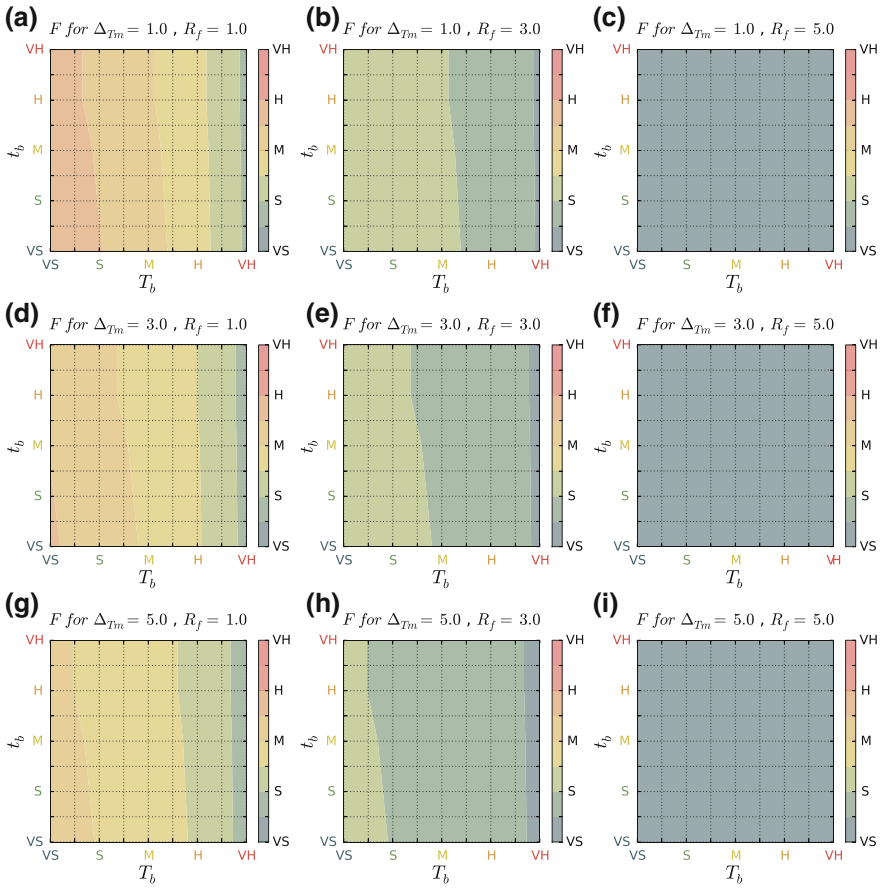


Fig. 6.18 Values of Fruity (F) as a function of Kneading Temperature (T_b) and Kneading Time (t_b) for three different values of Ripeness (R_f) and Milling Temperature Increase (Δ_{Tm})

The selection of just one of these Pareto frontier points as production objective depends on the relevance assigned to each of the components of the objective vector. For instance, depending on the characteristics of the incoming olives and the market, obtaining a high yield might be more important than preserving the fruity of the VOO, or obtaining a VOO without any organoleptic defect might be the first concern in the elaboration. This point is addressed in Sect. 6.6.2 and leads to answering the second question: *Which of the possible objectives should be chosen.*

Finally, the answer to the last question, i.e., *which set-points of the process variable should we use to obtain the defined objective,* emerges naturally from the mathematical structure used to answer the previous questions, as the decision variables in the optimization problems are, precisely, the values of those desired set-points. The

question of obtaining those optimal for a given production objective is discussed in Sect. 6.6.

For the setup of the optimization problems, we classify the VOOPP variables according to their role in the optimization problem as:

- Parameters ($\mathbf{p}$): these VOOPP variables are considered to have a fixed value for the optimization problem at hand. They will usually include the properties of the incoming olives, along with some other process parameters whose value is justified to be fixed for the current problem.
- Decision variables ($\mathbf{u}$): these VOOPP variables are the ones whose value is to be specified by the optimization problem. Usually, they will be a subset of the process variables.
- Objective variables ($\mathbf{y}$): these are the VOOPP variables that are considered the output of the process and are included in the objective vector.

Since the values of the objective variables ($\mathbf{y}$) depend on the parameters ($\mathbf{p}$) and decision variables ($\mathbf{u}$), we may represent these relations as:

$$\mathbf{y} = f(\mathbf{u},\ \mathbf{p}).$$

The VOOPP model obtained in the previous section provides an approximation of this f function, as it relates the values of the output parameters with properties of the incoming olives and the technological parameters.

Following this notation, we may define the objective vector of the multicriteria optimization problem as:

$$F(\mathbf{y}, \mathbf{u} \mid \mathbf{p}) = \begin{bmatrix} f_1(\mathbf{y}, \mathbf{u} \mid \mathbf{p}) \\ f_2(\mathbf{y}, \mathbf{u} \mid \mathbf{p}) \\ \vdots \\ f_n(\mathbf{y}, \mathbf{u} \mid \mathbf{p}) \end{bmatrix},$$

with $f_i,\ i \in \{1,\ 2,\ \ldots,\ n\}$ representing each of the objectives.

The problems we are to solve for the answer of the different posed questions have the general structure:

$$\begin{aligned} \text{minimize} \quad & F(\mathbf{u} \mid \mathbf{p}) \\ \text{subject to} \quad & \mathbf{y} = f(\mathbf{u},\ \mathbf{p}) \\ & \mathbf{p} = \mathbf{p}^0 \end{aligned}$$

with the meaning of *minimize* being properly defined in each particular problem studied.

6.6.1 Achievable Production Objectives

This section covers the determination of the Pareto boundary of possible production objectives given a set of fixed parameters of the process. In this optimization problem, it is natural to consider that $\mathbf{p}$ is composed of the properties of the incoming olives. However, any condition of the process variable whose value was to be regarded as fixed could also be included in the array.

The weighted sum scalarization method was used to obtain the different plots included below. This method requires finding the solutions to the following problems:

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & J = \sum_{k=1}^{c} \omega_k f_k(\mathbf{y}, \mathbf{u}) \\ \text{subject to} \quad & \mathbf{y} = f(\mathbf{u}, \mathbf{p}) \\ & \mathbf{p} = \mathbf{p}^0 \\ & \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max} \end{aligned}$$

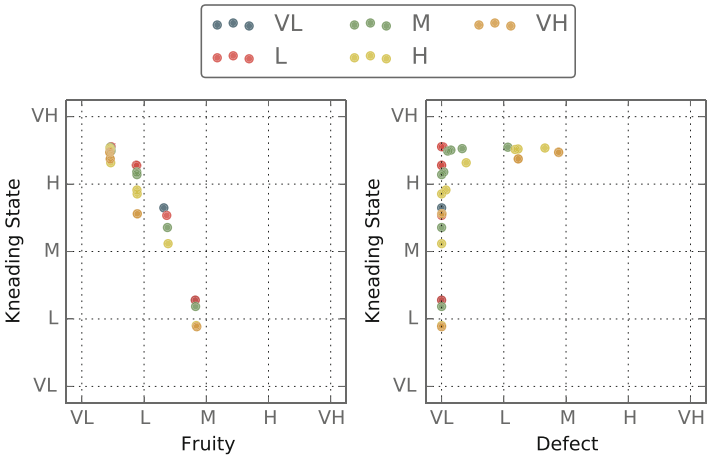
for different combinations of scalarization weights ω_k.

The elements of the objective vector considered were the quality characteristics of the obtained VOO included in the paste preparation model developed in the previous section, namely Fruity (F) and Defect (D), and Kneading State (K_s).

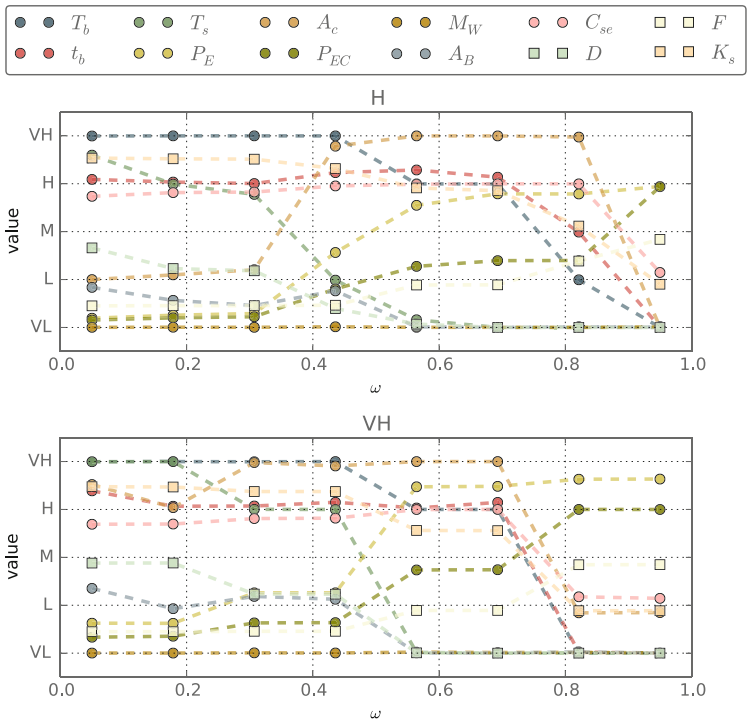
An analysis of the paste preparation model shows that there is no real trade-off between Defect (D) and Fruity (F). The only node that influences both variables is Fruit State (E_f), and the effect of this parameter in both variables shows the same sign. Taking this fact into account, the value of both nodes was combined into a single objective in order to reduce the analysis to a bi-objective problem and facilitate the visualization of the results. As for the weight considering the relative relevance of Fruity (F) and Defect (D), a value of $\omega_q = 0.5$ was chosen to assign the same priority to both parameters. The influence of assigning different priorities to these variables is illustrated in Sect. 6.6.2, when dealing with the selection of a single production objective. This way, the objective vector analyzed was:

$$F(\mathbf{y}, \mathbf{u} \mid \mathbf{p}) = \begin{bmatrix} f_1(\mathbf{y}, \mathbf{u} \mid \mathbf{p}) = \omega_q \; F + (1 - \omega_q) \; D \\ f_2(\mathbf{y}, \mathbf{u} \mid \mathbf{p}) = \qquad K_s \end{bmatrix}.$$

We begin showing how we can analyze the impact on the achievable production objectives and their trade-offs caused by the different properties of the olives. Figure 6.19a shows points in the Pareto front for different values of Incoming Olive Moisture (H_o^I), with the rest of olive characteristics considered defined in Table 6.5. The different starting points for the frontier in the highest achievable Fruity (F) area, showing lower values of Kneading State (K_s) for wetter olives, illustrate the more challenging conditions of obtaining good yields from wet olives while preserving the quality. However, the most noticeable difference is illustrated in the plot relating Defect (D) and Kneading State (K_s), as obtaining high values of Kneading State (K_s)



(a) Points belonging to the Pareto boundary for the different values of Incoming Olive Moisture (H_o^I) specified in the legend, with the rest of the olive properties and fixed process parameters specified in Table 6.5.



(b) Process Set-Points for the points in the Pareto boundary shown in Figure 6.19a. The title of the subplot indicates the value of Incoming Olive Moisture (H_o^I) considered.

Fig. 6.19 Pareto boundary and process set-points for different values of Incoming Olive Moisture (H_o^I)

Table 6.5 Value of the olive properties and fixed VOOPP parameters for Figs. 6.12 and 6.20

	Value
Dirtiness (D_t)	VL
Hammer worn (D_h)	VL
Incoming fruit state (E_f^I)	VH
Incoming olive moisture (H_o^I)	M
Olive illness (O_I)	VL
Pit–flesh ratio (R_p)	M
Ripeness (R_f)	M
Sieve type (S_t)	M
Sieve worn (D_c)	VL

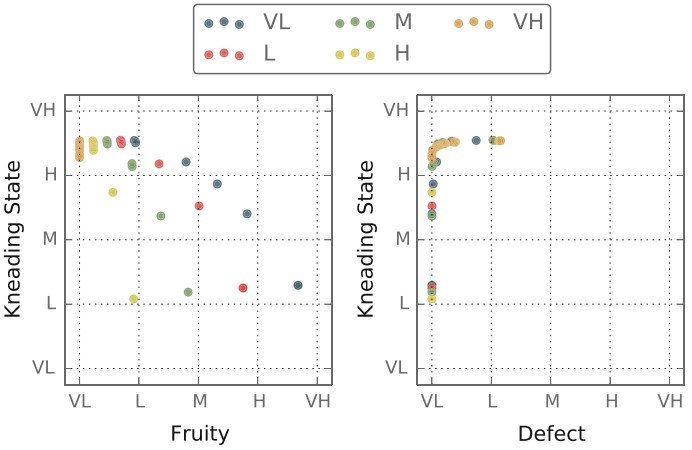


Fig. 6.20 Points belonging to the Pareto boundary for the different values of Ripeness (R_f) specified in the legend, with the rest of the olive properties and fixed process parameters specified in Table 6.5

conveys having a remarkable increase in Defect (D). A plot of the decision variables for the different problems solved when Incoming Olive Moisture (H_o^I) is VH and H is included in Fig. 6.19b. An inspection of this plot shows that main responsible for this behavior is the increase in Storage Time in Hopper (T_s) required to decrease Corrected Paste Emulsion (P_{EC}) beyond the point that Coadjuvant Addition (A_c) can provide.

Another parameter that is interesting to analyze is the influence of Ripeness (R_f) on the different values of Fruity (F) which are achievable. Figure 6.20 shows the Pareto front of this relation, with lower values of Ripeness (R_f) always offering higher values of Fruity (F) for every value of Kneading State (K_s). It is also worth noticing the reduction of choices for interesting process objectives as Ripeness (R_f) increases, patent in the increasing slope of the Pareto frontier showing that decreasing aimed Kneading State (K_s) yields smaller improvements of Fruity (F).

Table 6.6 Value of the olive properties and fixed production parameters for each of the scenarios considered in Fig. 6.21

	November	December	January healthy	January damaged	February	March
Dirtiness (D_t)	VL	VL	VL	VL	VL	VL
Hammer worn (D_h)	VL	VL	VL	VL	VL	VL
Incoming fruit state (E_f^I)	VH	VH	VH	M	M	L
Incoming olive moisture (H_o^I)	VH	H	M	M	L	L
Olive illness (O_I)	VL	VL	VL	VL	VL	VL
Pit–flesh ratio (R_p)	M	M	M	M	M	M
Ripeness (R_f)	VL	L	M	M	H	VH
Sieve type (S_t)	M	M	M	M	M	M
Sieve worn (D_c)	VL	VL	VL	VL	VL	VL

Although the properties of the olives are theoretically independent variables, they usually present some correlation in their values (Hermoso et al. 1997). That is, some combinations of values of the properties are more likely to be found in a real scenario, while others are very unlikely. We analyze next the achievable objectives for some typical combination of the properties of the olives found through the season, as included in Table 6.6.

Figure 6.21 shows the points of the Pareto boundary for each of these scenarios. A first inspection of this figure draws the attention to the decrease of the maximum values for Fruity (F) and increase of the minima for Defect (D) for successive scenarios. As commented in the previous section, the progressive increase of Ripeness (R_f) and decrease of Fruit State (E_f) are responsible for this behavior. This is coherent with the well-known fact in the industry that, in order to obtain good VOO quality, the olive conditions must meet some requirements, with the VOOPP not being able to compensate for the lack of quality of the olives.

The increase in Defect (D) when approaching the higher Kneading State (K_s) values in the two first scenarios can be attributed to the higher Olive Moisture (H_o) considered and the associated increase of Storage Time in Hopper (T_s), a behavior already depicted in Fig. 6.19a. In turn, the vertical disposition of the points in the plot on the right of Fig. 6.21 for the lower-quality scenarios reveals that good yields do not necessarily convey an increase in Defect (D), but a toll is paid in the decrease of Fruity (F).

The values of the set-points for each considered scenario and relative weight of the objectives are commented below:

- **November**: The values of the process variables for low values of ω, which represent the priority of Kneading State (K_s), reflect the traditional recipe for elaborating

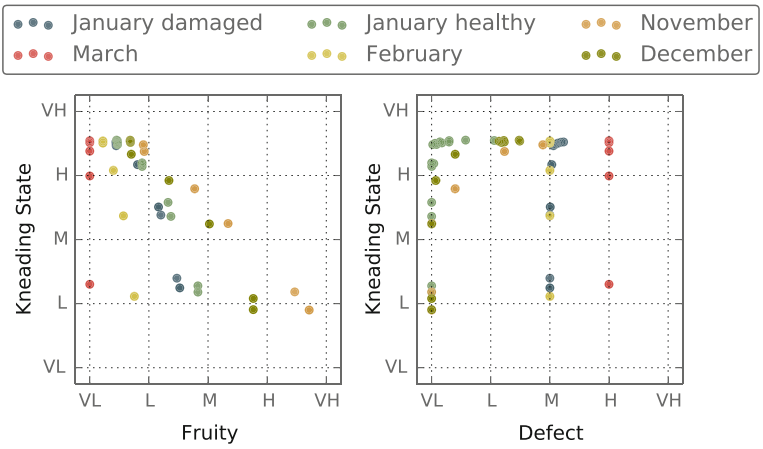


Fig. 6.21 Points belonging to the Pareto boundary for the different scenarios specified in the legend, with the corresponding olive properties and fixed process parameters specified in Table 6.6

VOO without caring about its quality: high Storage Time in Hopper (T_s) to allow for the olives to lose moisture at the expense of increasing Defect (D), and high values of Kneading Temperature (T_b) and Kneading Time (t_b), which offer good Kneading State (K_s) while reducing Fruity (F). The high value of Coadjuvant Addition (A_c) is explained by the existence of emulsions even after the storage in the hopper.

As quality increases its weight in the objective function, the first variables to decrease their values are Storage Time in Hopper (T_s), as it affects both Fruity (F) and Defect (D), and Kneading Time (t_b), which favors a increase in Fruity (F) without affecting too seriously Kneading State (K_s). As ω keeps increasing, Kneading Temperature (T_b) begins to drop, initially being compensated with an increase of Kneading Time (t_b). Finally, when all the relevance is given to the quality, Storage Time in Hopper (T_s), Kneading Temperature (T_b), and Kneading Time (t_b) are assigned their lowest possible values (Fig. 6.22).

- **December and January healthy**: The evolution of the process set-points is very similar in these scenarios to the November scenario. The values of Storage Time in Hopper (T_s) are lower, since Incoming Olive Moisture (H_o^I) is also lower than in the previous scenario, and less time is required for it to be reduced. This explains the lower values of Defect (D) of this scenario for equivalent values of Kneading State (K_s), as visible in Fig. 6.21.

  It is also visible the tendency of requiring higher weights in the objective function for the process variables to adapt to values promoting the quality of the oil, as the final value of this quality decreases from one scenario to the next (Fig. 6.23).
- **January damaged, February and March**: These scenarios represent conditions where olives already present values of Incoming Olive Moisture (H_o^I) which do not provoke problems related to the formation of emulsions. Because of this, the value

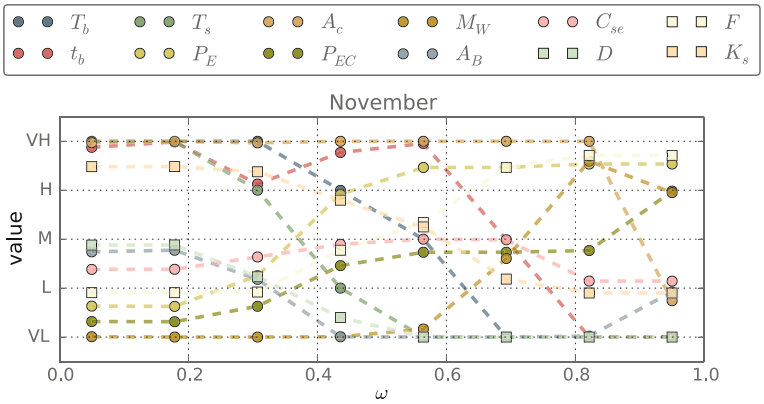


Fig. 6.22 Process set-points for different scalarization weights in the scenario *November*. The corresponding points in the Pareto boundary are shown in Fig. 6.21

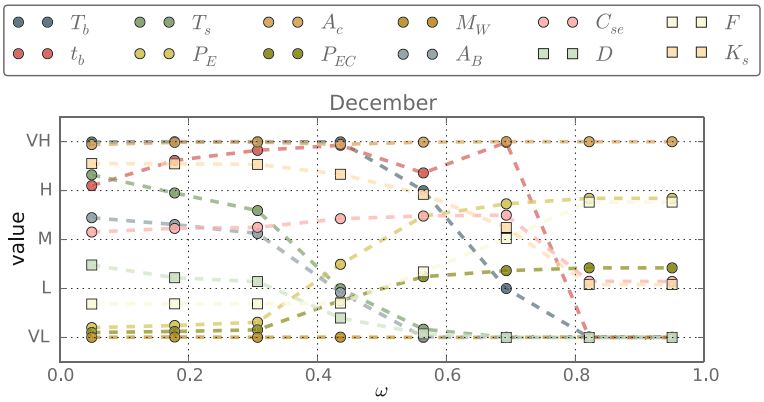


Fig. 6.23 Process set-points for different scalarization weights in the scenario *December*. The corresponding points in the Pareto boundary are shown in Fig. 6.21

of both Storage Time in Hopper (T_s) and Coadjuvant Addition (A_c) is saturated at its minimum value for all points (Fig. 6.24). The low quality of the olives conveys the suggestion of set-points aiming for favoring Kneading State (K_s) for almost every value of ω, what is also reflected in the higher concentration of the points in the Pareto plot of Fig. 6.21.

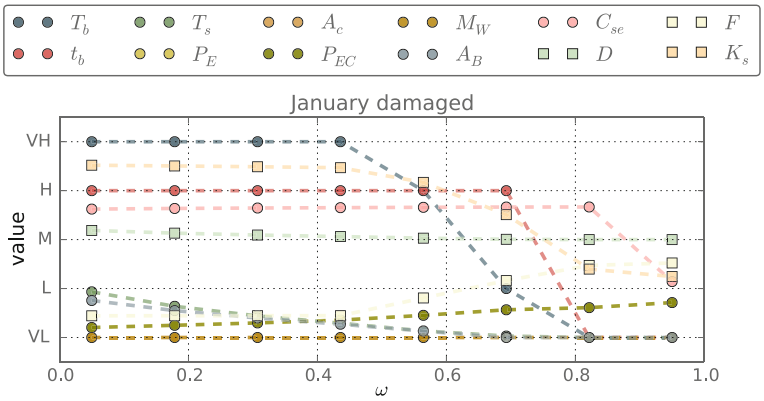


Fig. 6.24 Process set-points for different scalarization weights in the scenario *January damaged*. The corresponding points in the Pareto boundary are shown in Fig. 6.21

6.6.2 Selection of the Optimal Production Objective

Once that the points in the Pareto frontier have been found and the possible production objectives are available, it is for the decision maker to choose which of these points should be aimed at, or to establish criteria for its selection (Fig. 6.23).

The weight assigned to each of the quality elements of the objective function is likely to change during the season. For instance, in the early season, when having defect is not unavoidable, keeping the defect as zero is likely to have the greatest importance for the process, as having a defect immediately means that the quality of the VOO is no longer Extra VOO, and consequently, the price might decrease substantially.

On the other hand, when some defect is unavoidable due to the olive properties, then it might not be so important to prevent increasing the defect if it remains below the bounds of *lampante*, which might leave room for focusing on obtaining better industrial yield.

An obvious approach to take into account the above discussion is to find the production objective that maximizes the income of the produced oil for each production scenario. Two evident factors affect the obtained revenue: the obtained quantity of VOO and its quality, as it affects its market price. A third factor to be considered could be the possibly different production costs of elaborating VOO of different qualities.

To implement this approach, a function that maps the properties of the VOO to a price, a model that provides the industrial yield based on Kneading State (K_s), and a function providing the production costs are required.

The current model of the paste preparation stage is useful when finding the Pareto optimal values, as obtaining good values of Kneading State (K_s) is known to provide

good industrial yields. However, when selecting the optimal elaboration point according to an economic criterion, we need to know the value of industrial yield expected from the operation. This amount can be obtained employing the VOO separation model, assuming that the separation is carried out optimally.

Some comment on the way of considering the industrial yield is required. The total amount of oil obtained per mass unit of processed olives does not depend only on Yield (X) but also on the total amount of oil that the olives convey. It has already been mentioned that there is an evolution of this amount of fat contained in the olives through the season and that variation might be relevant if the additional freedom of when to harvest the olives is granted. In this section, however, we embrace the hypothesis that the olives are already at the factory, which leaves the question of considering how much oil the olives contain irrelevant for the solution of the problem, as this quantity is fixed, and the same irrespective of what quality of VOO we finally produce. The selection of the optimal production objective when that hypothesis is relaxed is also an interesting problem, that is dealt with in Sect. 6.7.

As for the function that maps the properties of the VOO to its price, we may rely on the published bulk market price of the VOO according to its commercial quality, and consider a function that provides this quality based on the VOO characteristics. Such function, applying the European Norm 2568/91, is:

$$q(F, D) = \begin{cases} \text{Extra Virgin Olive Oil} & \text{if } D = 0 \text{ and } F > 0 \\ \text{Virgin Olive Oil} & \text{if } 0 < D \leq 3.5 \text{ and } F > 0 \\ \text{Lampante Olive Oil} & \text{if } D > 3.5 \text{ or } F = 0 \end{cases}$$

Once that the income is available, the production cost should also be taken into account. This production cost can be estimated based on the values the optimal set-points assigned for the objective, which are also available, as commented in the previous section. A simple estimation of the unitary cost per resource allows to include this consideration.

The above discussion suggests the following objective function of the problem to be solved:

$$J = X\ p(q(F, D)) - \sum_j c_j x_j, \tag{6.8}$$

where $p(\cdot)$ denotes the function that maps the commercial quality of the oil to its market price, and c_j is the unitary cost of the process variable x_j, with j indexing all the relevant process variables.

The sale prices have been taken from the average bulk sale prices for the Extra, Virgin, and Lampante qualities from the Poolred system Poolred (2014) from June to July period of 2013 and included in Table 6.7. A fourth quality, namely *Extra superior* not included in the official quality classification, is included in this table. This category regards Extra VOO that possesses high values of Fruity (F) of, and that, although they are technically classified as just Extra VOO, from a

Table 6.7 Sale prices in each scenario (Euros/kg)

PRODUCT	Extra sup.	Extra	Virgin	Lampante
PRICE	4	2.71	2.51	2.36

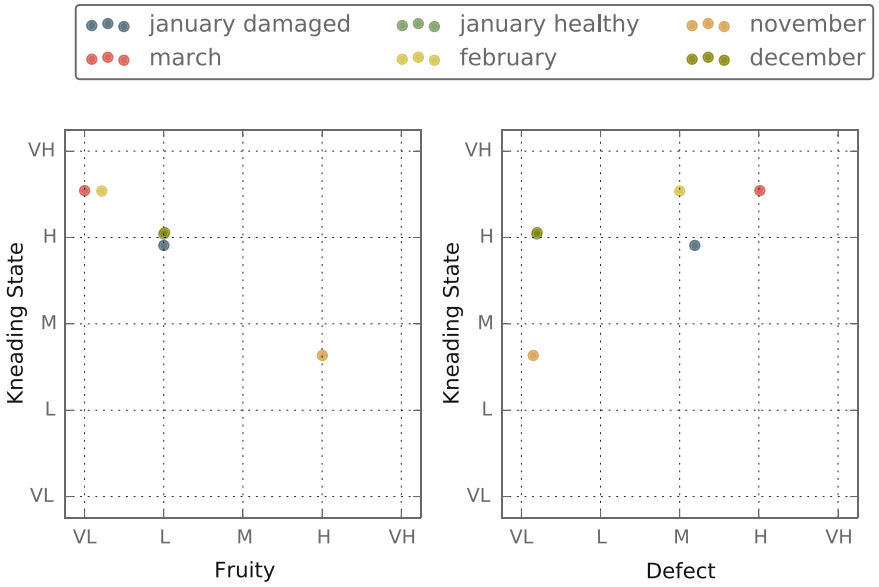


Fig. 6.25 Pareto plot of the optimal production points according to the prices defined in Table 6.7, considering that Extra VOO is obtained with any positive value of Fruity (F)

consumer-oriented quality perspective they are different from regular Extra VOO in that the market is willing to pay a higher price for them.

Figure 6.25 shows the optimal production points according to the application of a strict definition of the quality classification of VOO. Here, the November point aims at obtaining the characteristics for *Extra superior*, defined by requiring a minimum value of Fruity (F) of H. The December point, surprisingly, drops its Fruity (F) aim to L. This is a consequence of applying strictly the technical quality classification, as for a VOO to be Extra it is just required to have *nonzero* Fruity (F) and zero Defect (D). However, from a market-oriented perspective, VOO with such low values of fruity would not be considered as Extra and, thus, would not be paid the corresponding price.

Figure 6.26 shows the production objective for a situation where a market-aware interpretation of the quality of the VOO is applied, considered as requiring minimum values of Fruity (F) for a VOO to be classified as Extra or Virgin. In turn, Fig. 6.27 shows the corresponding set-points of the process variables for those objectives.

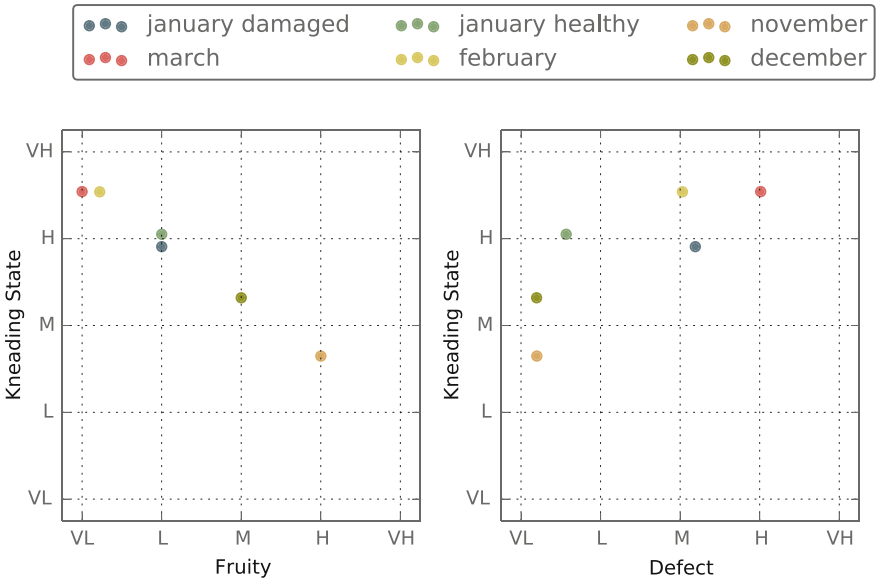


Fig. 6.26 Pareto plot of the optimal production points according to the prices defined in Table 6.7, considering that a small minimum value of Fruity (F) is required for obtaining Extra VOO

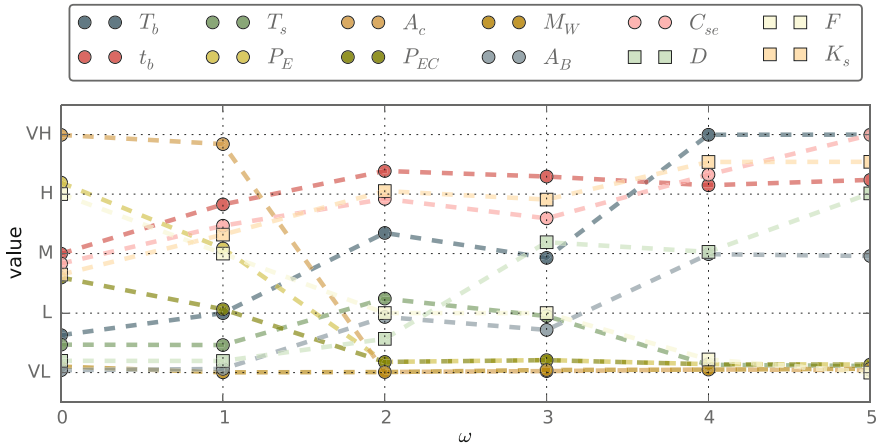


Fig. 6.27 Values of the process set-points for the optimal production points according to the prices defined in Table 6.7, considering that a small minimum value of Fruity (F) is required for obtaining extra VOO. The *November* scenario corresponds to $\omega = 0$, while *March* corresponds to $\omega = 5$

The *November* scenario still aims at Extra superior quality, with prescribed set-points according to this objective: very low value of Storage Time in Hopper (T_s) so that no organoleptic defect is provoked on the oil and use of Coadjuvant Addition (A_c) to reduce the emulsions; low value of Kneading Temperature (T_b) and moderate of Kneading Time (t_b) to preserve the Fruity (F) of the oil.

The target in *December* is Extra VOO, as Fruity (F) would not be high enough to reach the threshold required for Extra superior, as depicted in Fig. 6.21. Since the objective of Fruity (F) is milder, slightly more aggressive conditions are prescribed for the elaboration, increasing slightly both Kneading Time (t_b) and Kneading Temperature (T_b).

The conditions for both scenarios considered in *January* are very similar, being the objective in both producing Virgin VOO. Kneading Temperature (T_b) shows moderate values to preserve Fruity (F) into the limits for the prescribed quality. A slightly slower value of the parameter is set for *January damaged*, as the quality condition of the olives is worse.

Lastly, the objective for *February* and *March* is also very similar, and the process conditions are prescribed identical. In these scenarios, the focus is on obtaining the highest possible quantity of oil, as the quality is already lost; consequently, high values of Kneading Temperature (T_b) and Kneading Time (t_b) are suggested.

6.6.3 Inclusion of Feedback: Run-to-Run Control

The approach taken to implement feedback into the higher layer of the process is, as presented in Chap. 4, to augment the system with an EWMA-like observer and use the nonlinear models developed in the previous section. Using a quadratic objective function, the proposed Run-to-run controller is:

$$
\begin{aligned}
\underset{\mathbf{u}_k}{\text{minimize}} \quad & J = (\hat{\mathbf{y}}_k - T)^T \, Q \, (\hat{\mathbf{y}}_k - T) + \mathbf{u}_k^T \, R \, \mathbf{u}_k \\
\text{subject to} \quad & \hat{\mathbf{y}}_k = f(\mathbf{u}_k, \mathbf{p}) + \hat{\nu}_k \\
& \mathbf{p} = \mathbf{p}^0 \\
& \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max} \\
& \hat{\nu}_k = \omega \, \hat{\nu}_{k-1} + (1 - \omega) \, (\mathbf{y}_{k-1} - f(\mathbf{u}_{k-1}, \mathbf{p}^0)),
\end{aligned}
$$

where Q, R, and ω are their tuning parameters.

Two main types of disturbances have been studied in the Run-to-run literature: steps disturbances and process drifts. Steps disturbances model well sudden changes in the operating condition that remain constant for successive operations, while drifts are better for considering situations where some effect continuously varies from one batch to the next, such as accumulation of dirt (Lee et al. 2008).

This chosen observer structure is known to work well with step disturbances, not being so effective when drift disturbances affect the process. Selecting this observer

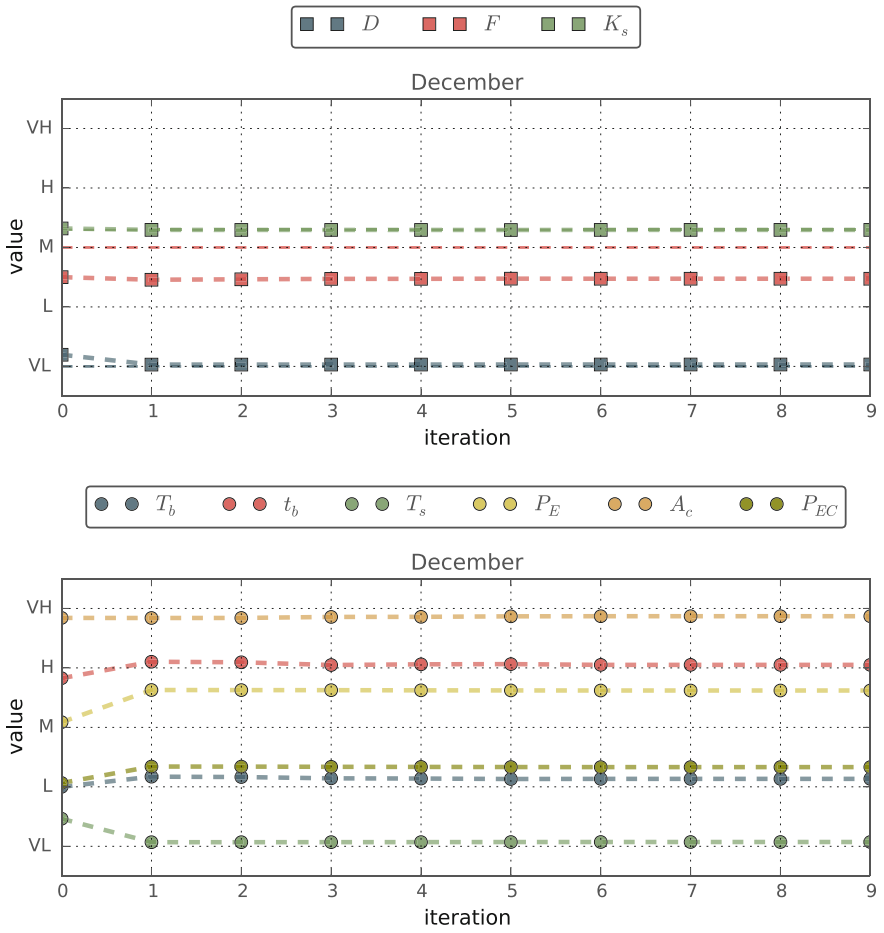


Fig. 6.28 Values of the process outputs and prescribed set-points by the Run-to-run controller for $\omega = 0.2$, when the priority is set to Kneading State (K_s) and a constant disturbance is acting on Fruity (F), for the *December* scenario

structure implicitly assigns more relevance to step disturbances in the process than drifts, that is, indeed, the case for our application of the method to the VOOPP. Since the models are static, a step disturbance is a good approximation for a mismatch of the gain of the process Lee et al. (2008), which is the most appealing use case for the feedback approach. On the other hand, drift disturbances are less likely to occur to the process at the high-level layer. One possible drift could be the progressive moisture loss of the olives in the hoppers as they remain there, but this effect is already considered in the model, so updating the value of Storage Time in Hopper (T_s), this effect is already accounted for. In the proposed scheme, we suppose that the operator measures all the relevant process variables in the VOOPP in each batch,

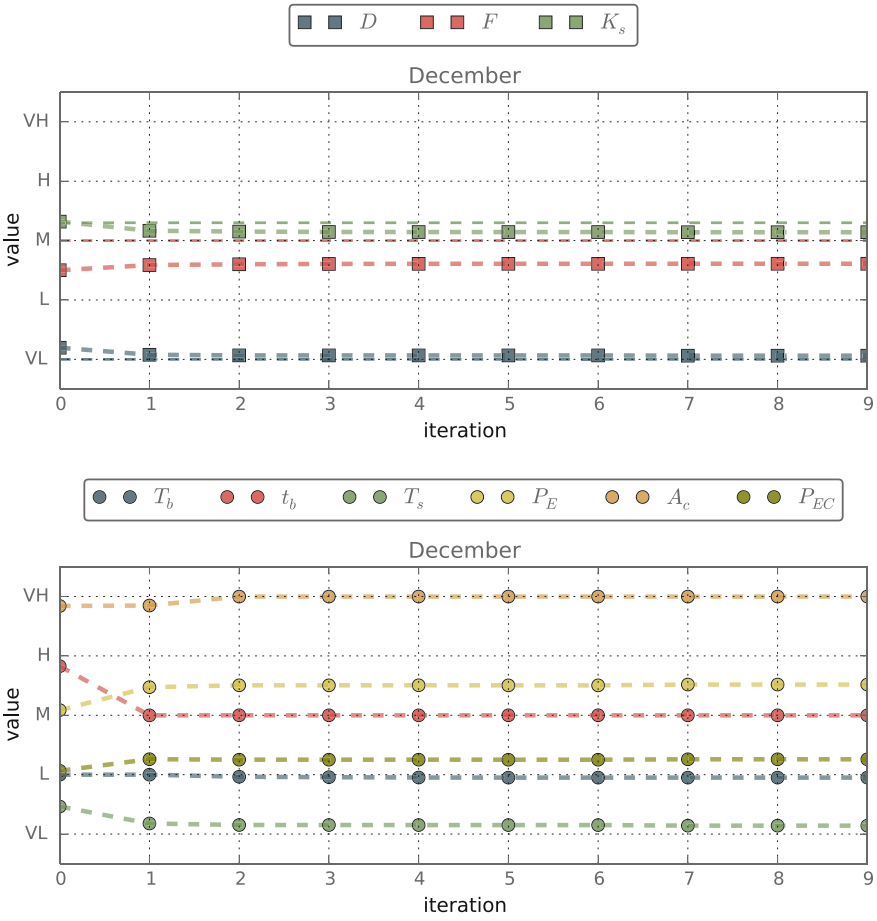


Fig. 6.29 Values of the process outputs and prescribed set-points by the Run-to-run controller for $\omega = 0.2$, when the priority is balanced and a constant disturbance is acting on Fruity (F), for the *December* scenario

and these values are supplied to the system, thus providing some feedforward action for the change of these variables.

The proposed Run-to-run controller has three parameters susceptible of tuning: Q, R, and ω. The first two influence the assignment of relative importance to the errors in the output process variables and the control action, while the latter influences the way the observer estimates the disturbance affecting the system, thus influencing the convergence rate of the controller. This effect can be visualized in the comparison of Figs. 6.28, 6.29, and 6.30.

In order to illustrate the influence of the process parameters, let us analyze in some detail the scenario December, as it presents a balanced situation between achieving good quality and good yield. Let us assume that the production objective for the

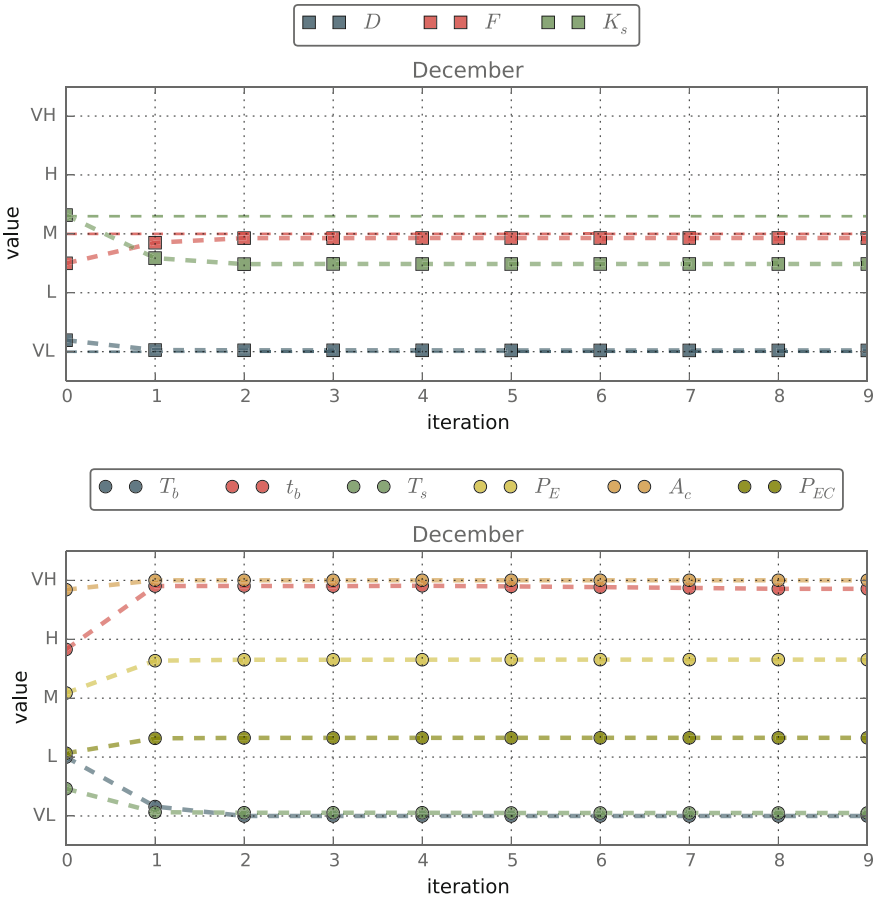


Fig. 6.30 Values of the process outputs and prescribed set-points by the Run-to-run controller for $\omega = 0.2$, when the priority is set to Fruity (F) and a constant disturbance is acting on Fruity (F), for the *December* scenario

scenario is given by the economic criterion presented in Sect. 6.6.2 and that there is a mismatch between the model used for obtaining the optimal production set-points and the actual plant, modeled as an offset affecting Fruity (F).

If we apply the set-points computed according to the method proposed in the previous section, we will find that the process outputs do not exactly match the objective, as Fruity (F) is below the desired values. Iteration 0 of Figs 6.28, 6.29, and 6.30 shows this behavior.

Given the multi-objective nature of the VOOPP, it is expected that in order to reduce the mismatch in Fruity (F), some toll had to be paid in the rest of outputs. So, a preference in the tolerance against deviations in the different outputs should be established, which is exactly the role of parameter Q in the controller.

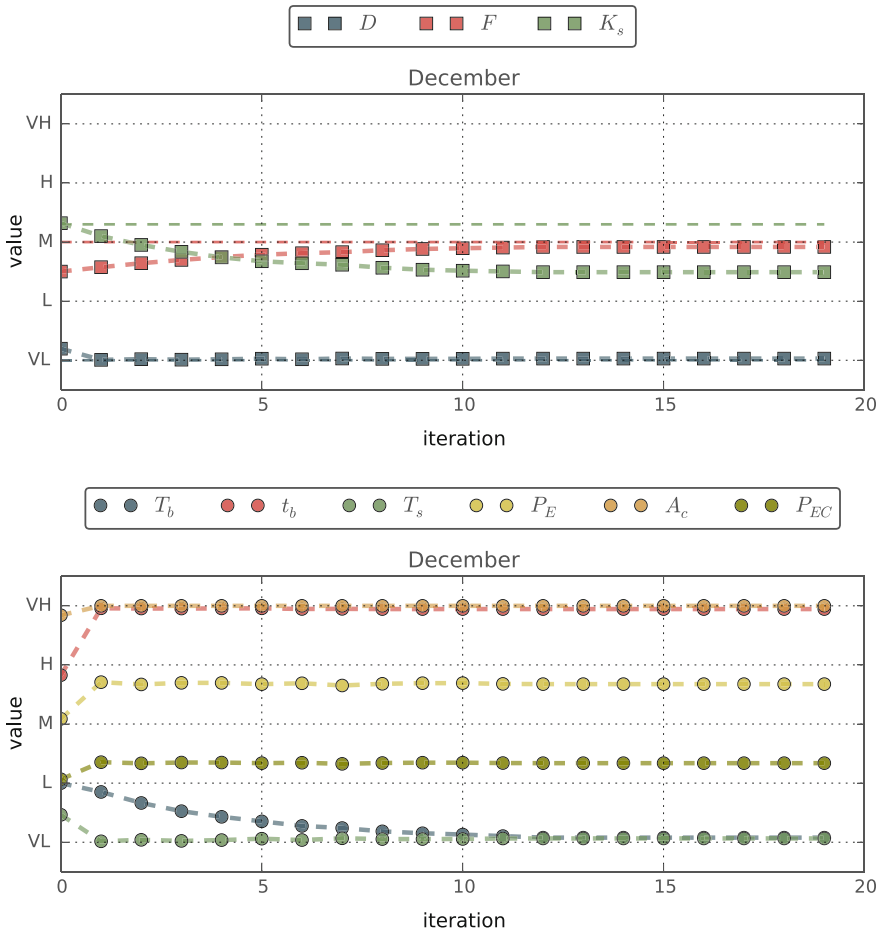


Fig. 6.31 Values of the process outputs and prescribed set-points by the Run-to-run controller for $\omega = 0.8$, when the priority is set to Fruity (F) and a constant disturbance is acting on Fruity (F), for the *December* scenario

Figure 6.28 shows the situation when the preferred output is Kneading State (K_s). Given that the desired value of the parameter is obtained, no major changes are suggested for the process variables, just a slight correction to further reduce Defect (D) which does not imply a change in Kneading State (K_s). The error showed in Fruity (F) is tolerated, and no further actions are prescribed.

In turn, Fig. 6.29 shows the case when the deviations in each of the process variables are equally penalized. As expected, the stipulated process set-points are slightly modified to decrease the error in Defect (D) at the expense of lightly worsening Kneading State (K_s).

To complete the discussion, if the emphasis is put on achieving the prescribed Fruity (F), disregarding the decrease of Kneading State (K_s), the sequence of

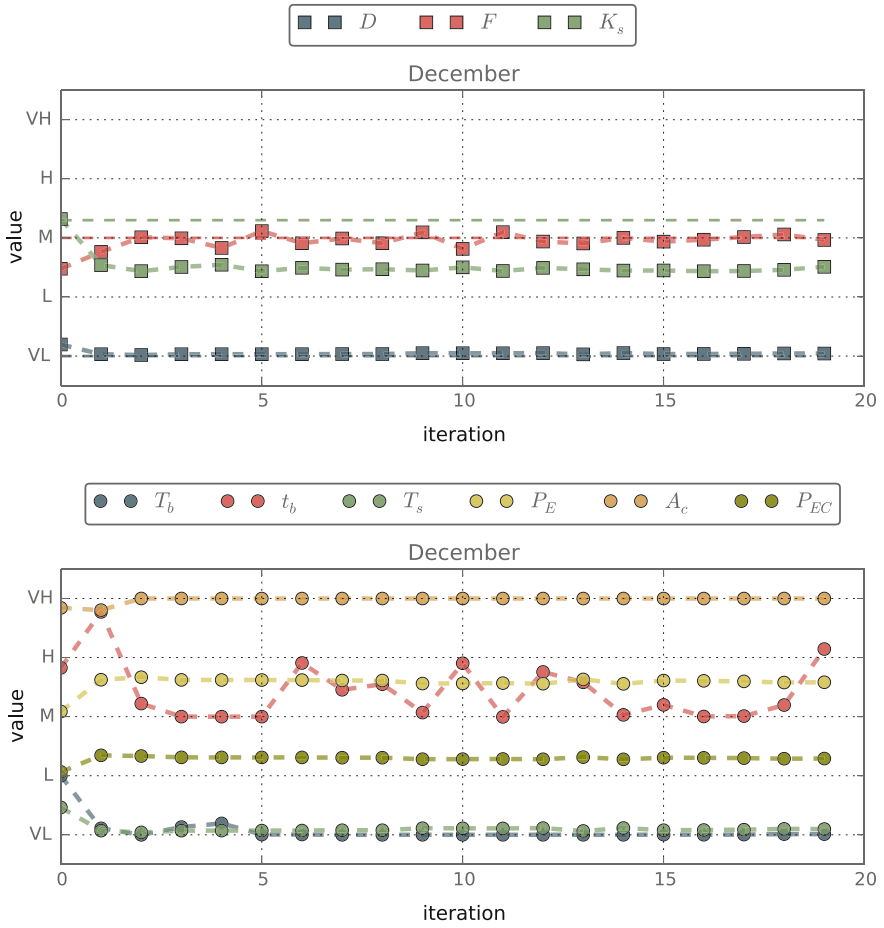


Fig. 6.32 Values of the process outputs and prescribed set-points by the Run-to-run controller for $\omega = 0.2$, when the priority is set to Fruity (F) and a stochastic disturbance is acting on Fruity (F), for the *December* scenario

proposed set-points by the algorithm is portrayed in Fig. 6.30. In this scenario, the desired level of Fruity (F) is almost exactly achieved, at the expense of a higher decrease of Kneading State (K_s).

The influence of ω can be observed comparing Figs. 6.30 and 6.31. In these plots, Q and R have the same value, while ω is set to 0.2 and 0.8, respectively, in each figure. The final values of both the outputs and the process set-points end being equal, but the convergence rate in the first case is much higher than in the second.

However, choosing small values of ω yields higher convergence rate in the case of a fixed deterministic step disturbance, but the algorithm is less robust to the existence of stochastic noise, as the filtering performed by the observer is much milder. To illustrate this effect, Figs. 6.32 and 6.33 show the response of the system

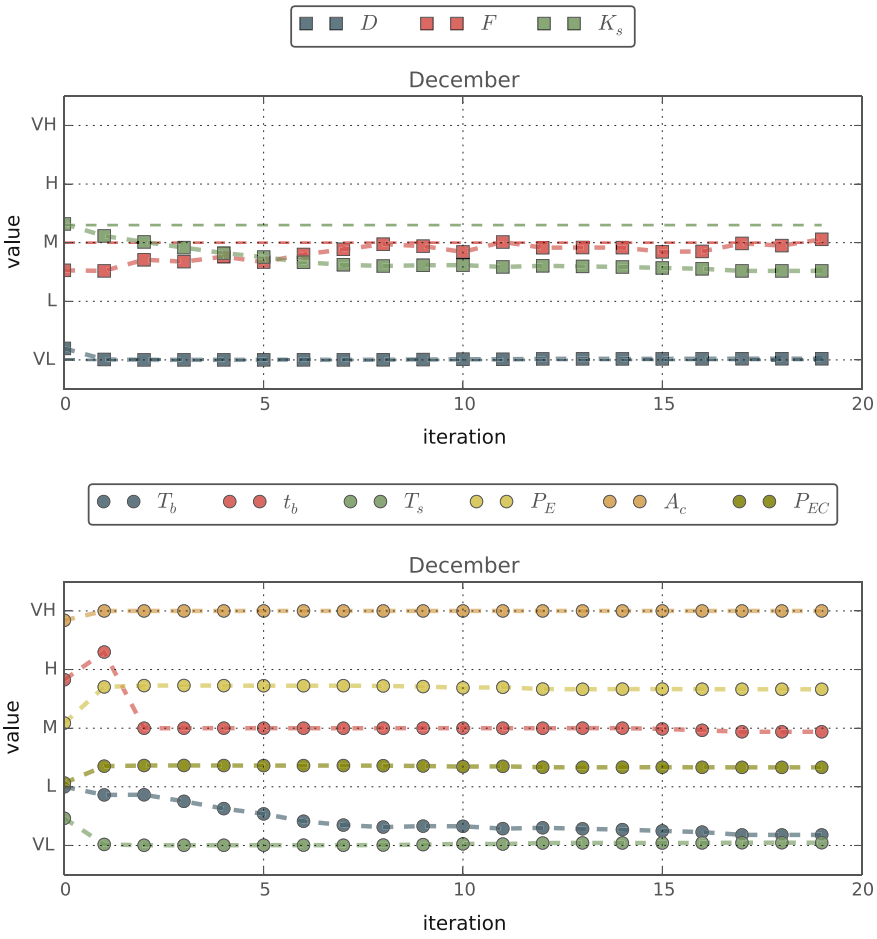


Fig. 6.33 Values of the process outputs and prescribed set-points by the Run-to-run controller for $\omega = 0.8$, when the priority is set to Fruity (F) and a stochastic disturbance is acting on Fruity (F), for the *December* scenario

when independent random Gaussian noise of zero mean and variance 0.25 is applied. The aggressive tuning shows a much greater variability of the prescribed process set-points, while the behavior is smoother for the more conservative one.

Equation 4.44 provides the condition for convergence of the algorithm for the MIMO unconstrained linear case. Using this Equation as guideline, the candidate range of values of ω so that the system converges is the [0, 1] interval. Employing a constant value of 1 for every iteration of the controller is equivalent to actually not including any feedback action at all, since we do not update the estimate of the disturbance and stick to whatever value we considered for the initial iteration.

The other extreme in the range supposes completely disregarding previous estimates of the disturbance, and select the observed mismatch in the last iteration as

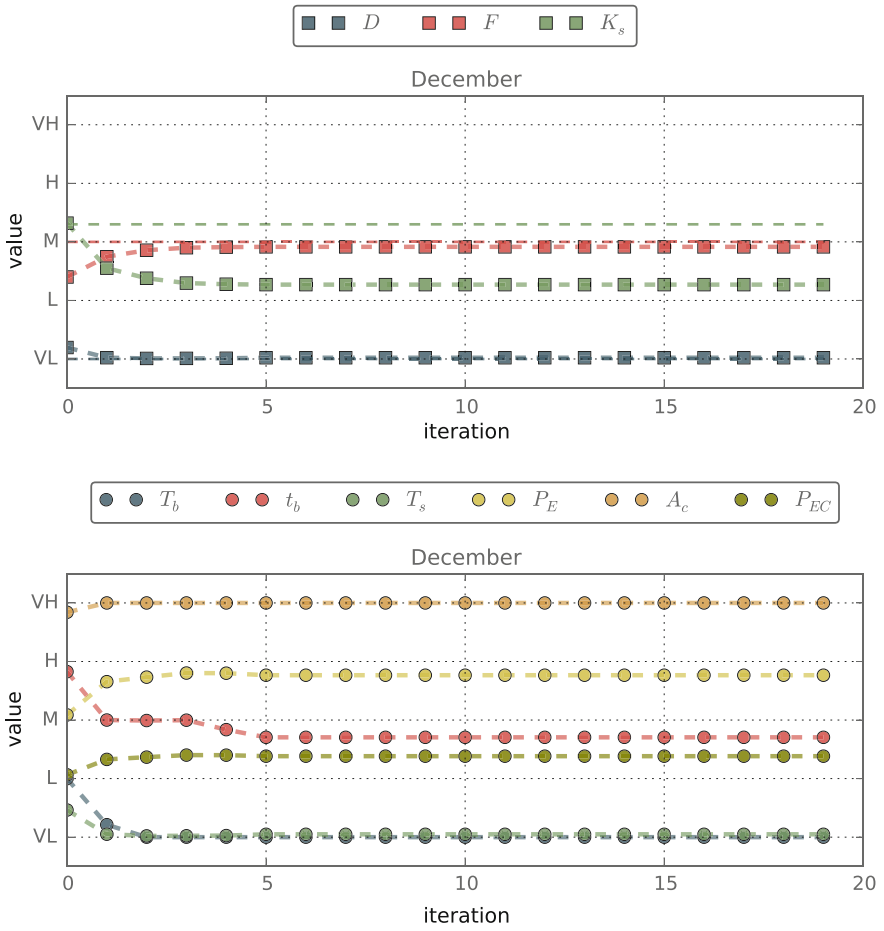


Fig. 6.34 Values of the process outputs and prescribed set-points by the Run-to-run controller for $\omega = 0.2$, when the priority is set to Fruity (F) and a disturbance of the form $\eta_k = \alpha\, \mathbf{y}_k$ is acting on Fruity (F), for the *December* scenario

the estimate. In turn, simulating the system for values of ω higher than 1 effectively conveys the nonconvergence of the algorithm.

Finally, Fig. 6.34 shows the suggested set-points for a scenario where a level-dependent disturbance of the form:

$$\eta_k = \alpha\, \mathbf{y}_k$$

is acting on the system. The plot shows a satisfactory behavior of the system, similar to that obtained for the constant disturbance considered before.

6.7 Production Planning: Selection of Harvesting Dates

The properties of the olives depend heavily on their ripeness, which in turn depends on the moment they are harvested. This way, the assumption that the olives are already harvested conveys that a key decision with a fundamental influence on the properties of the elaborated VOO has already been made. This decision, in turn, influences the profitability of the operation, so a relevant question to address is to find what amounts of what quality of VOO should be produced to maximize the profit for the year, which in turn requires choosing when to harvest the olives so that their properties allow to carry out this schedule.

This production plan can be found applying the methodology proposed in Section 5. According to the guidelines of the proposed method, a SKU can be defined for each final product, considering the product as a combination of the properties of the produced good and the marketing particularities used for its commercialization. For simplicity, let us consider four quality levels, namely *Lampante*, *Virgin*, *Extra*, and *Extra Sup*, and that all the products will be sold in bulk except for the highest quality product that will be sold exclusively bottled. This way, we shall define four SKUs according to four quality levels for the final VOO produced. As VOO is not perishable, the limits on the demand of each SKU ($D(i)$) need to be defined only for the whole season, and these values need to be provided by estimations of the capacity of the company to sell each of the products.

The only raw inputs that are required for the production of VOO are olives, so we define four SKUs to represent olives with properties matching the quality requirements of the output SKUs. The connection between the input and output SKUs is provided by the matrix R, whose entries define how much of each input SKU is required to produce a unit of output SKU. These entries encode key information about the process that influence the solution of the optimization problem, so providing sensible and accurate values for them is important. The other parameters of the problem whose values need to be defined taking into account process considerations are $C(i, t)$ that represents the production cost and may be influenced by the properties of the inputs, and thus change from one time instant to the next, and $S(i, t)$ that represents the supply of the corresponding input SKUs, thus representing the availability of raw inputs with adequate properties for the production of the corresponding output SKU.

Figure 6.35 shows a block diagram of the flow of data for the optimization setup. As depicted in the figure, the origin is a model of the evolution of the olives that provide the values of their key features. These values, in turn, are used by a model of the process that provides, for each output SKU, the values that allow to compute the parameters $R(i, j, t)$, $S(i, t)$, and $C(i, t)$. Finally, these parameters are used in the optimization problem to provide the production plan, i.e., the variables $x_{i,t}$.

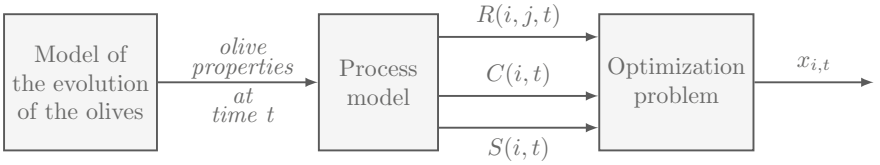


Fig. 6.35 Block diagram of the flow of information for the Production Planning optimization setup

The following sections further elaborate on how the olive evolution model can be constructed and how to compute the parameters $S(i,t)$ and $C(i,t)$ from the available process models.

6.7.1 Model of the Evolution of the Olive Properties

The foundation of the approach is a model that provides the values of the relevant features of the inputs for each time instant considered. In our case, the characteristics of the olives relevant to the problem are the ripeness (R_f), the fat content (X_o), and the humidity (H_o).

As commented in Chap. 5, different levels of sophistication and effort can be devoted to the construction of these models. On the one hand, the simplest approach is to use historical data or published bibliography to obtain average values of the expected properties of the olives. On the other hand, if a large database of historical data that contains additional information, such as pluviometry or other meteorological factors is available, then a bit more sophisticated approach of trying to select the subset of the data that most faithfully resembles the conditions of the current season can be pursued.

In our case, the data provided in Jimenez Herrera et al. (2012) and Gutiérrez et al. (1999) were used for the ripeness and oil content, respectively, and implemented as a lookup table. Quite surprisingly, no data of the evolution of H_o could be found in the published bibliography, so typical evolution data were provided by experts in VOO elaboration and incorporated into the lookup table model.

6.7.2 Process Models

The role of the process models is to provide the values of the parameters $R(i,j,t)$, $C(i,t)$, and $S(i,t)$ for the optimization problem. For this purpose, the high-level models of the process presented in Sect. 6.5 and techniques presented in Sect. 6.6 can be used, as they provide the reference values of the process variables and the expected

values of the output variables, thus providing the basic data for the computation of these parameters.

The entries of $R(i, j, t)$ basically model the relation between the amount of olives processed and oil obtained. The fundamental factors that define this relation are two: the composition of the olives and the amount of oil that the process is capable of extracting. Obviously, the composition of the olives determines how much oil they contain in the first place; however, the moisture content of the olives is also a parameter of interest, as it also influences the extractability. This extractability, as shown in the previous section, also depends on the target quality of the produced oil, so a problem like

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & J = f(\mathbf{y}, \mathbf{u}) \\ \text{subject to} \quad & \mathbf{y} = f(\mathbf{u}, \mathbf{p}) \\ & \mathbf{p} = \mathbf{p}^0 \\ & \mathbf{y}_q = \mathbf{y}_q^0 \\ & \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max} \end{aligned}$$

needs to be solved for each time instant t and output SKU. Here, $\mathbf{y}_q^0$ represents the required quality for the corresponding SKU and $\mathbf{p}^0$ contains the values of the input olives obtained from the evolution model. If this problem is unfeasible, then the quality of the inputs is not enough to produce the corresponding SKU, so $S(i, t)$ must be set to zero. On the other hand, if the problem is feasible, then $S(i, t)$ must be set to the forecasted supply of olives at that time instant and the solution of the problem provides the expected extractability and the values of the process variables that allow to obtain it.

Performing a mass balance on the inputs and outputs of the process, the following relation can be derived:

$$R(i, j, t) = \left(1 - \frac{H_{oi}}{100}\right)\left(\frac{X_{oi} - \gamma_i}{100}\right)\left(1 - \frac{\gamma_i}{100}\right)^{-1}, \tag{6.9}$$

where X_{oi} and H_{oi} account for the composition of the olives and γ_i represents the extractability obtained from the solution of the optimization problem (6.9).

Once the values of the process variables are defined, the computation of approximate process costs can be performed via simple relations. The values of the time that the olives are held in the hopper, and the choice of sieve size does not have much influence in the process costs and can be omitted. The cost associated with the use of microtalc can simply be modeled by:

$$c_{A_t} = A_t \cdot p_{talc}, \tag{6.10}$$

where p_{talc} is the price per kg. of the microtalc employed. The cost of heating the olive paste can be estimated as:

$$c_{T_b} = (T_b - T_{amb}) \cdot c_{paste} \cdot \frac{p_{fuel}}{pci_{fuel}}, \tag{6.11}$$

with c_{paste} being the heat capacity of the olive paste, pci_{fuel} and p_{fuel} the lower heating value and price of the fuel, respectively.

The value of the kneading time t_b does not significantly influence the total processing cost, since it is the rate of flow of paste into the decanter that determines the production rate and, consequently, the amount of time that the factory must be operating in order to process the olives.

Lastly, the man labor cost in the factory is basically independent of the quality that is being produced, and can also be disregarded in this initial overview.

The analysis above finally renders the simple process cost estimation equation:

$$C(i,t) = c_{A_{ti}} + c_{T_{bi}} \tag{6.12}$$

6.7.3 Optimization Problem Definition

Once that the process-related parameters of the problem are available, the next step is to particularize the constraints that need to be considered in the optimization problem. The choices made in the definition of the SKUs imply that there is a one-to-one relation between input and output SKUs, so the associated constraints can be expressed as

$$R(i,\ j,\ t)x_{j,t} \leq x_i, t \in T, i \in F.$$

Here, F represents the subset of I that contains the indexes of the four output SKUs considered, and j is associated with the corresponding value of i. This way, there are four constraints—one per each output SKU—for each time step considered in the problem.

The capacity constraints imposed by the finite processing capacity of the *almazara* can be modeled using a single resource U to represent the processing capacity of the facility, as all the products basically undergo the same routing through the same machines. This way, the capacity constraint can be formulated as

$$U \sum_{i \in I_p} x_{i,t} \leq 1, t \in T.$$

As commented before, since the output SKU is not perishable, then there is no reason to include demand constraints for the individual time steps. Typical market conditions for bulk VOO are such that the demand can be regarded as unbounded for all practical purposes, so the demand constraint needs to be included only for the *Extra sup.* SKU ($i = 4$), which is sold bottled

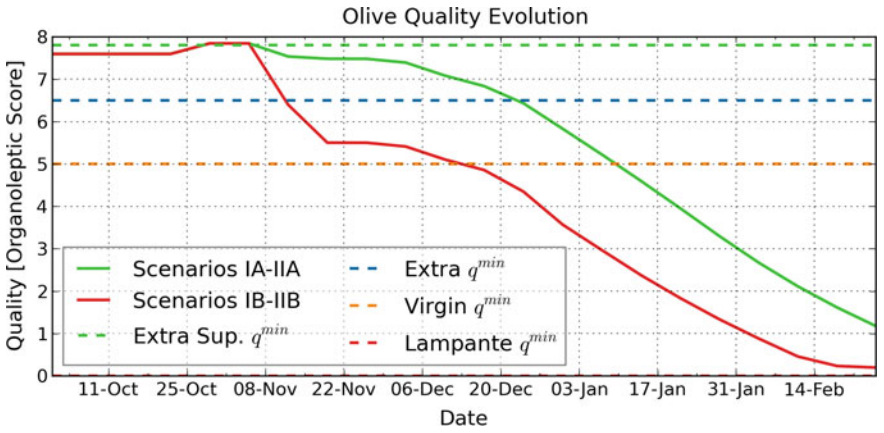


Fig. 6.36 Quality evolution of the olives for the different scenarios considered. (©2014 IFAC. Reprinted, with permission, from Cano Marchal, P., Martínez Gila, D., Gámez García, J., and Gómez Ortega, J. Optimal production planning for the virgin olive oil elaboration process. In IFAC World Congress, volume 19, pages 8921–8926)

$$\sum_{t \in T} x_{4,t} \leq D_4.$$

The raw input supply limitations for each time step can be expressed as

$$x_{i,t} \leq S(i,t), i \in G, \tag{6.13}$$

while the total availability of raw materials for the season can be modeled as

$$\sum_{i \in G} \sum_{t \in T} x_{i,t} \leq S. \tag{6.14}$$

In both cases, G stands for the subset of I that contains the indexes of the raw input SKUs. Putting all these constraints together with the objective function, the optimization problem to be solved can be formalized as

$$\begin{aligned} \underset{x_{i,t}}{\text{minimize}} \quad & \sum_{t \in \mathbb{T}} \sum_{i \in \mathbb{I}} \big(C(i,t) - P(i,t)\big)\, x_{i,t} \\ \text{subject to} \quad & R(i,j,t) x_{j,t} \leq x_i, t \in T, i \in F., \\ & U \sum_{i \in F} x_{i,t} \leq 1, t \in T, \end{aligned} \tag{6.15}$$

Table 6.8 Sale prices in each scenario (Euros/kg)

PRODUCT	Extra sup.	Extra	Virgin	Lampante
SCENARIOS IA-IB	4	2.71	2.51	2.36
SCENARIOS IIA-IIB	3.5	1.75	1.65	1.59

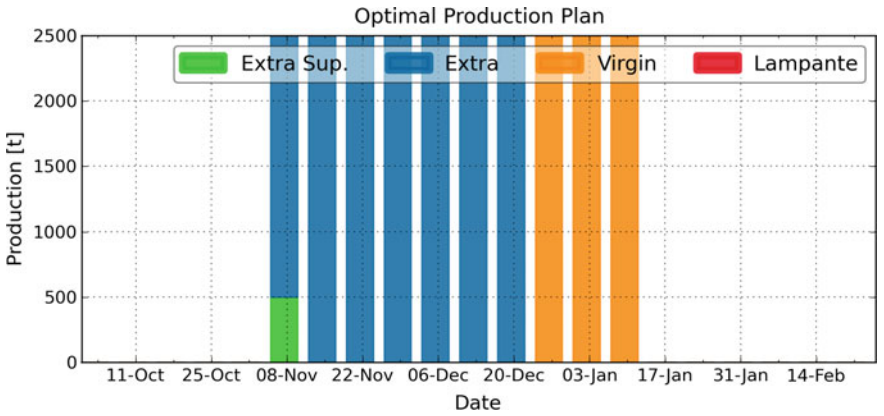


Fig. 6.37 Optimal production plan for scenario IA. Previously published in Cano Marchal et al. (2014) ©2014 IFAC, used with permission

$$\sum_{t \in T} x_{4,t} \leq D_4,$$
$$x_{i,t} \leq S(i,t), i \in G,$$
$$\sum_{i \in G} \sum_{t \in T} x_{i,t} \leq S.$$

In order to illustrate the proposed method, four scenarios have been considered based on two different values for two parameters.

First, two different scenarios of the evolution of the quality of the olives in the orchards are considered. Scenario A considers a regular evolution of the quality, while scenario B considers the situation when some factor, such as a plague or hail, supposed to occur on the first week of November, provokes a substantial decrease of the quality. Figure 6.36 depicts the evolution of the quality for the considered scenarios.

The second parameter considered is the price for the different SKUs. Two different sets of sale prices have been taken from the average bulk sale prices for the Extra, Virgin, and Lampante qualities from the Poolred system (Poolred 2014). Data for

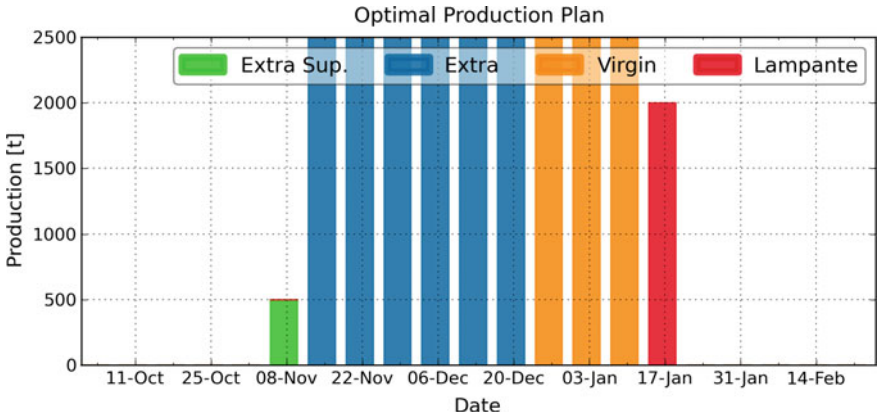


Fig. 6.38 Optimal production plan for scenario IIA. Previously published in Cano Marchal et al. (2014) ©2014 IFAC, used with permission

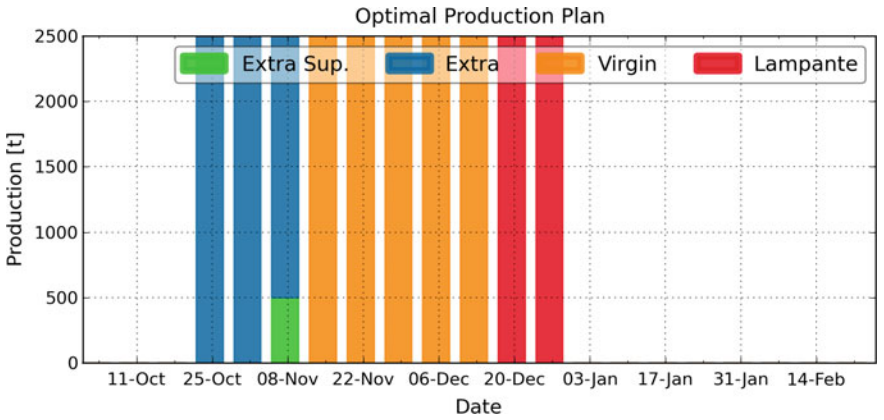


Fig. 6.39 Optimal production plan for scenario IB. Previously published in Cano Marchal et al. (2014) ©2014 IFAC, used with permission

scenario I are taken from June to July period of 2013, while scenario II considers the same period of 2012. For the Extra superior product, since there are no published data, the sale price has been fixed as a typical sale price for that product. The different prices are gathered in Table 6.8, and Fig. 6.37 plots the optimal production plan for scenario IA, while Fig. 6.38 depicts scenario IIA.

As can be seen, both scenarios are quite similar, just implying a small shift in production toward the final part of the harvest season for IIA. The comparison between scenarios IA and IB (Fig. 6.39) shows the convenience of starting to harvest earlier

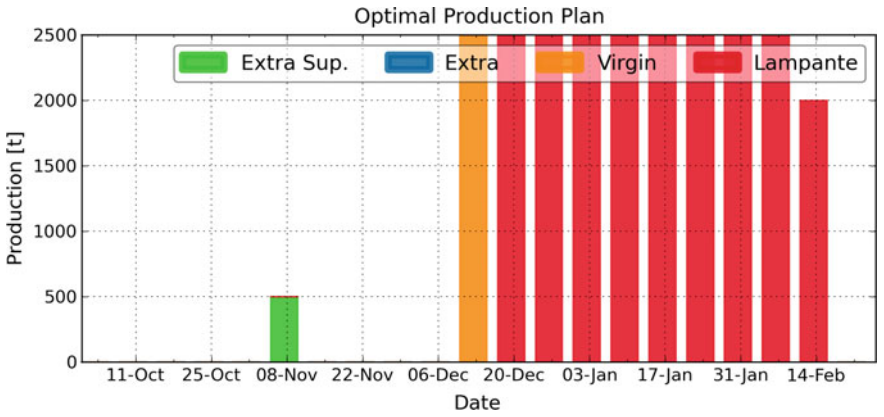


Fig. 6.40 Optimal production plan for scenario IIB. Previously published in Cano Marchal et al. (2014) ©2014 IFAC, used with permission

when the quality drops sharply and the spread between prices for the products is high.

The remarkably different plans provided for scenarios IB and IIB (Fig. 6.40) highlight the fact that if the spread of prices is not high enough, and the base quality is low, it is better to plan the production just aiming to maximize the amount of obtained oil. Finally, it is worth noting that the production of Extra superior remains constant between scenarios, and limited by the selling capacity considered. The fact that it is produced as late as possible is justified by the increasing fat content and extractability due to the evolution of the ripeness of the olives.

References

Bordons C, Núñez-Reyes A (2008) Model based predictive control of an olive oil mill. J Food Eng 84(1):1–11

Marchal C et al, Cano Marchal P, Martínez Gila D, Gámez García J, Gómez Ortega J (2014) Optimal production planning for the virgin olive oil elaboration process. In IFAC world congress, vol 19, pp 8921–8926

Cert A, Alba J, Leon-Camacho M, Moreda W, Perez-Camino MC (1996) Effects of talc addition and operating mode on the quality and oxidative stability of virgin olive oils obtained by centrifugation. J Agricul Food Chem 44(12):3930–3934

Clodoveo ML (2012) Malaxation: influence on virgin olive oil quality. Past, present and future - an overview. Trends Food Sci Technol 25(1):13–23

Gutiérrez F, Jímenez B, Ruíz A, Albi MA (1999) Effect of olive ripeness on the oxidative stability of virgin olive oil extracted from the varieties picual and hojiblanca and on the different components involved. J Agricult Food Chem 47(1):121–127

Hermoso M, Uceda M, Frías L, Beltrán G (1997). Maduración. In: Barranco D, Fernández-Escobar R, Rallo L (eds), El cultivo del olivo, 2 edn. Mundi-Prensa, Madrid

Jimenez Herrera B, Rivas Velasco A, Sanchez-Ortiz A, Lorenzo Tovar ML, Ubeda Munoz M, Callejon RM, de Ouiros EOB (2012) Influence of fruit maturation process on the sensory quality of virgin olive oils from Picual, Hojiblanca and Picudo cultivars. Grasas Y Aceites 63(4):403–410. WOS:000311708500007

Lee S-P, Chou R-J, Tseng S-T (2008) Stability and performance of a double MEWMA controller for drifted MIMO systems. IIE Trans 40(7):690–705

Poolred (2014) POOLred-Sistema de Información de Precios en Origen del Mercado de Contado del Aceite de Oliva

Stach W, Kurgan L, Pedrycz W (2010) Expert-based and computational methods for developing fuzzy cognitive maps. In: Glykas M (ed) Fuzzy cognitive maps. Studies in fuzziness and soft computing, vol 247. Springer, Berlin, pp 23–41

Chapter 7
Conclusions and Future Directions

7.1 Conclusions

In this book, we have delved into the particularities of the application of automatic control techniques to food transformation processes. The broad nature of the food processing industry encompasses processes whose complexity ranges from selection and packaging of fruits and vegetables, to the production of prepared meals, passing through the brewing of beer or the manufacture of dairy products.

This heterogeneity undoubtedly implies that there is a wide spectrum of different situations where the application of automatic control can provide its benefits to the processes, but also means the existence of diverse challenges to overcome for its successful utilization. Many of these challenges are common to the ones faced by process industry: dead-times, disturbances, intertwined dynamics in multivariable systems, etc. However, the key particularity of food industry is that quite often the variables of interest of the final products cannot be easily measured online, so the application of automatic control to these variables needs to be tailored to this special circumstance.

The best way to tackle the complications that arise by this lack of measurements is by leveraging the hierarchical control paradigm to split the plant into two layers: a lower-level one where the measurement of the variables does not suppose any problems, and a higher-level layer where measuring the output variables of interest is restricted. Under this scheme, the lower-level layer can be controlled using standard control techniques, while the higher-level layer can focus its attention to the relations between the references of the lower-level control loops and the product features.

Having appropriate models of the processes is often the first step for the successful design of a control system. The purpose of the model and the availability of online experimental data greatly determine the type and complexity of the model. The tuning of the lower-level feedback loops requires relatively simple process model that can be identified using standard System Identification approaches, such as subspace methods. On the other hand, obtaining models for the upper layer typically requires a

P. Cano Marchal et al., *Production Planning, Modeling and Control of Food Industry Processes*, Advances in Industrial Control,
https://doi.org/10.1007/978-3-030-01373-8_7

different approach, as the System Identification route often is not an option due to the lack of online measurements. For this layer, the application of fuzzy techniques is an interesting alternative, as it allows to employ expert knowledge for the construction of the models and admits to fine-tune their parameters using eventually available experimental data. The precision requirements of the upper-layer models are typically more demanding than those of the lower-level layer, mainly due to the preponderance of feedforward over feedback techniques for the control of these outputs.

As already mentioned above, the challenges of the control of the lower-level layers of the food transformation processes are quite similar to the ones faced by process industry, so many of the control ideas and techniques that are applicable to process industry are useful for the control of this layer. Consequently, PID control constitutes the basic tool for this layer, and the tuning of these controllers can be done using different methods according to the available knowledge about the system.

Beyond the basic PID, feedforward actions are also useful when there are measurable disturbances that affect the controlled output and two-degree-of-freedom control plays an important role for situations where the set-point of the variables is changed frequently or the controlled outputs need to follow a certain trajectory.

Batch operation of food processes is fairly common, so the application of specific techniques for this type of scenarios, such as Iterative Learning Control or Run-to-run control, may also have a positive impact on the behavior of the controlled system as they allow to *learn* from previous operations and use that knowledge to improve the current batch.

The control of the higher-level layer is where the challenges that are more specific to food industry reside. The key idea is that just controlling the lower-level layer does not guarantee that the outputs of the process, considered as the set of features that are important for the final consumer and the economic profitability of the operation, will have the desired values. In fact, if only the lower-level layer is under feedback control, then the relation between the outputs of the lower-level layer and the actual outputs of interest of the process is left in an open loop. This means that if the values chosen for the outputs of the lower layer do not provide the desired process outputs, then there is no mechanism capable of detecting this deviation and correcting it. The key element, again, is the difficulty of measuring online the values of the process outputs, and the sampling rate at which these variables can be assessed fundamentally defines what techniques can be implemented.

This frequent difficulty in obtaining measurements of the outputs implies that the implementation of feedforward schemes is easier to carry out in practice, however, at the cost of requiring models of better quality. Once that a model that relates the set-points of the low-level layer with the process outputs is available, it can be used for different purposes. The general strategy is to set up an optimization problem that includes the process model as a constraint and define an objective function that captures the production goals. Since the features of the incoming raw inputs typically exert a very high influence on the characteristics of the final products, the first utility of this approach is to define what production objective to aim for taking into account the features of the input materials. Furthermore, the solution of this problem also provides what reference values of the lower-level variables should be provided.

Despite the advantages of including these feedforward schemes, the inclusion of feedback in the higher-level layer is also desired, as it provides robustness against Modeling errors, disturbances, and uncertainty in the assessment of the properties of the raw inputs. The idea is to leverage the setup already available and include an observer that incorporates actual information about the process into the optimization problem, thus allowing to update the set-points according to the actual behavior of the process. This constitutes an approach that is very similar to Model Predictive Control and that, depending on the available sampling rate of the output variables, can be seen more as a Run-to-run control approach.

Finally, the seasonal availability of the different inputs makes it interesting to consider when to plan the production so that the profit of the year is maximized. This can be done making use of the models above—or simplified versions of them—defining an optimization problem that decides which products to produce based on an estimation of the properties of the incoming goods.

7.2 Future Directions

As highlighted multiple times in the book, the greatest limitation in the application of feedback ideas to food processes is the difficulty of measuring online the features of interest of the final products. It is the opinion of the authors that the development of sensors capable of performing this task is the key step that may enable a breakthrough in the industry. The amount of research effort devoted to this topic during the last years demonstrates that the authors are not alone in this point of view.

The three most widely researched technologies for this purpose are near-infrared spectroscopy, computer vision, and electronic noses. Plenty of research effort has been devoted to these technologies, as they show features that are necessary conditions for their application online: They don't destroy the samples in the analysis, don't require preparation of the samples and are relatively fast. Unfortunately, these conditions are far from sufficient and many technological challenges still need to be resolved before they are routinely used in every food transformation process in the industry.

One of the essential difficulties for the application of these technologies is the inherent heterogeneity of food materials, which makes it sometimes difficult to obtain robust calibrations, particularly for near-infrared spectroscopy applications. Fortunately, this shortcoming is likely to be superseded by the inclusion of smart functions and Internet of Things (IoT) ideas in the sensors. It is a well-known fact that having a large and heterogeneous set of samples is a key aspect for the construction of robust calibrations. The possibility of having the sensors, in different geographical locations, continually collects and submits data to centralized servers, drastically reduces the costs of obtaining data representative of the different possible states of the product, and offers a scenario where the continuous improvement of the calibrations can be performed at a relatively low cost.

Still, the requirement of having a external method of reference providing values for the update of the calibrations and the consequent necessity of sampling require additional work for the operators of the factory. However, even in absence of these values, nonsupervised techniques could be applied to cluster the data and minimize the number of reference values required to expand or robustify the calibrations.

Another major difficulty is the adaption of the instruments and techniques to the requirements imposed by online measurement. In effect, aspects such as production cadence, physical accessibility of the product, fouling of the sensors or vibrations may complicate substantially the application of an already well-tested method in laboratory conditions. These aspects complicate the acquisition of the primary data, usually providing worse signal-to-noise ratios than the laboratory conditions, thus further complicating the construction of robust calibrations.

One possible alternative is the development of custom automated sampling devices that allow to perform at-line measurements at a relatively high rate, without requiring the intervention of human operators. The drawback of this approach is the additional cost of the sampling devices and the inclusion of additional elements that need to be maintained.

Yet another source of information from the process are evaluations from expert operators. The use of these evaluations for the control of the process has been explored in the literature and is a line of work that is interesting as well. The main drawback of this alternative is the reliance on human operators for the task, which makes it more difficult to assure a consistent source of data for the process, in terms of both sampling time and uncertainty in the provided information.

The use of these evaluations, however, as targets for other sensors in a calibration scenario, again leveraging the IoT capabilities, may result in an interesting research line. The objective is not to have an online sensor capable of providing the information that a reference analysis provides, but a sensor that is capable of reproducing the assessments that an expert operator employs to operate the plant. This type of information can be enough to introduce feedback into the process, as it merges naturally with the fuzzy models of the higher-level layer.

Another foreseeable tendency in the food processing industry is consumer-driven production, considered as explicitly contemplating the preferences and choices of the consumers when selecting the production objectives of process. The practical implementation of this tendency requires an additional model capable of mapping the perceptions and preferences of the consumers to the values of measurable output features of the products. It might be case that these features regarded by the consumers are not classically included in the set of variables usually considered in the process, thus requiring the development and implementation of process models that explicitly relate these new outputs with the corresponding raw input features and process variables. The modular nature of the Fuzzy Cognitive Maps Modeling approach offers advantages in these scenarios, as it is easy to extend existing models with new output variables.

The drastic decrease of cost of sensors and electronics and the pervasiveness of IoT approaches getting into precision agriculture could again set a new ground for the industry, as many of the variability of the incoming raw products could be dampened

by appropriate actions in the groves. Furthermore, the knowledge of these properties may trigger a better decision-making process regarding harvesting times, etc., and also provide data for the season-wide Production Planning problems that would have more reliable information and the possibility, again, of building better models of the evolution of the raw input properties.

Finally, the challenge of sustainability in the food industry can also be helped by the inclusion of automatic control ideas. The use of model-based controllers under an optimization scheme can accommodate terms explicitly accounting for the energy consumption, thus leading to more efficient operations of the processes. Furthermore, the development of precision agriculture may also have a positive impact on the consumption of resources in the groves, as having explicit data may prevent an excessive use of inputs due to uncertainty in the needs of the crops.

Index

P. Cano Marchal et al., *Production Planning, Modeling and Control of Food Industry Processes*, Advances in Industrial Control,
https://doi.org/10.1007/978-3-030-01373-8