Hybrid Modelling/ Machine learning for soft-sensing and process modelling

Alexandre José Gomes da Silva
alexandre.g.da.silva@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2018

Abstract

It is common to have systems within processes that are poorly understood and cannot be easily represented by first-principle models. A prototype tool was developed, in Python, aiming to soft-sense parameters belonging to these systems. This tool allows for the creation of predictive models using input and output data. These models were subsequently integrated in dynamic simulations in gPROMS®.

For this purpose, a partial least squares regression (PLSR) algorithm was implemented. This algorithm has the particularity of needing the models dimensions to be chosen prior to creating the model, therefore a 10-fold cross-validation (CV) method was implemented along with other tools to help judge the model's quality. The Distance to the model and Hotelling's T^2 statistical tests were also implemented for outlier detection.

In order to study the models behaviour when integrated in a dynamic simulation, data from two different cases studies, a granulation and a compression case, was used. Furthermore, data from a solid oxide fuel cell case was used to study the creation of a model capable of predicting several response variables. These cases led to the implementation of a 2nd order polynomial transformation and a reciprocal transformation.

The simulations showed the high importance of choosing training data that covers the operative conditions of the simulation. Otherwise the results obtained will be unreliable since the model is being used to predict data outside the range it was meant to be used in.

Keywords: Hybrid Modelling, Partial Least Squares Regression, Soft-sensing, Distance to the model, Hotelling's T^2

1. Introduction

In the process industry, a large number of sensors are commonly implemented with the purpose of delivering data for process monitoring and control. Approximately two decades ago, researchers started to use the large amounts of data being measured and stored by them to build predictive models. These models are referred to as soft-sensors.[1]

Two main classes of soft-sensors exist: model-driven and data-driven. Model-driven soft-sensors, also called white-box models, are based on first-principle models and therefore have full phenomenological knowledge regarding the process background. Data-driven soft-sensors, also called black-box models, are based on the data measured from a process and therefore the model itself has no knowledge of the process. Given that model-driven sensors are primarily used for planning and development of process plants and inferential control, the focus of soft-sensing is usually concentrated in data-driven models.[1]

As a consequence of being based on data measured directly from the process, data-driven sensors describe the real process conditions. Their usefulness is derived from the fact that they can build multivariate features related to different process variables that have a direct influence in the process's output quality. Several modelling approaches are used for these models, being the PLSR one of the most popular in chemical engineering and chemometrics, among others.[1]

The inclusion of a data-driven soft sensor in dynamic simulations implies its use alongside first-principle models. This combination of white and black box models constitutes another type of modelling strategy: grey-box modelling, also referred to as hybrid modelling.[2]

These types of models are specially appealing when a process is non-linear. When that is the case, hybrid modelling combines white and black box modelling, compensating for the shortcomings of the standalone implementation of these strategies. As a consequence, hybrid modes offers a higher degree of flexibility, enabling them to accomplish the modelling task based on precision, reliability and relevancy of the process knowledge and process data.[2]

Motivated by the fact that many systems integrated in chemical processes are poorly understood or too complicated to model trough first-principle models, a tool able to create predictive models from process data was developed, in Python. This was accomplished with a focus on model quality, which led to the implementation of several methods to judge the predictive capability of the model, and on model validity, which was assessed trough statistical tests. The model generated by this tool was then integrated into dynamic simulations, for the purpose of soft-sensing parameters belonging to the systems in question.

2. Background

Machine learning has been introduced steadily into the process industry in recent years, composing the main type of approach used for modelling soft-sensors. Even tough many different machine algorithms are available, for this work, the PLSR machine learning algorithm was deemed the most suitable.[1]

2.1. Machine Learning in industrial processes

Machine learning refers to a set of tools for understanding data, more specifically, it is an algorithm based method that allows the estimation of an unknown dependency between a systems inputs and outputs from the available data. Four types of machine learning algorithm exist: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. However, in the process industry, most problems call for either supervised or unsupervised learning, being that this work focus solely on supervised learning. Furthermore, most problems in this industry are regression problems i.e. problems with a quantitative response.[3][4][5]

Supervised learning consists on fitting a model that relates the output variables (response variables) to the input variables (predictor variables), using datasets with responses associated to each observation of the predictors. It either aims to accurately predict the response for future observations (prediction) or to better understand the relationship between the inputs and outputs (inference).[3]

Machine learning is widely used in different areas of the process industry. It has helped with the increasing complexity and high dimensionality of the manufacturing systems. This tendency leads to the prediction of an expanded use of machine learning in this field, specially trough hybrid approaches[6]. For sensor fault detection, machine learning is also applied. It is used for the identification of the sensor's or process's faults effectively, also allowing to find which particular sensor or set of sensor is responsible for the fault.[1]

2.2. Partial Least Squares Regression

In multivariate problems it is common to use the widely understood multiple linear regression (MLR), a method applicable when there are few input variables, they are not significantly collinear and there is some understanding of how they relate with the output variable. However, in most of the problems tackled through machine learning, in the field of engineering, it is highly common to come across a problem with a high number of input variables, which present a higher degree of collinearity. This fact renders MLR inefficient and thus the need to use another type of regression arises. The PLSR presents a useful alternative, especially when there is the need to obtain a predictive model for the output variable(s).[7] [8]

The PLSR relates the input variables (x) with the output variables (y) and it will create a multivariate linear model with predictive capabilities. When modelling through PLSR, it is assumed that the process in question is influenced by just a few underlying variables - latent variables (LV). It is not known how many of them there are, being one of the aims of PLSR analysis to estimate their number. The X and Y variables are hypothesized to be realizations of these latent variables and, therefore, are not assumed to be independent. The LV assumptions approximately correspond to the application of microscopic concepts, making PLSR suitable for the modelling of chemical data. Whenever the number of LV's equals the number of input variables, the latter is independent and thus PLSR yields the same results as MLR.[8]

2.2.1 Partial Least Squares Regression Algorithm - PLS1 and PLS2

Even though there are several types of PLSR algorithms , only the PLS1 and PLS2 algorithms are implemented in the scikit-learn package. They differ from each other in the sense that PLS1 is particular case of PLS2, applied when there is only one response variable.

The algorithm starts with the treatment of the input data matrices, – X of dimensions (N,K) and Y of dimensions (N,M) – in order to make their distributions be fairly symmetrical. They are centred, through the subtraction of the mean of each variable to their respective column in the input matrix and, optionally, they can also be scaled by diving each variable by its standard deviation. Once the treatment has been applied, an iterative loop begins, the outer loop, whose first step begins with another loop, the inner loop.

In the inner loop the weight matrices W of dimensions (K, A) and C of dimensions (M, A) for X and Y respectively are calculated. Said calculation is accomplished trough equations (1) and (2). In both equations, it is possible to observe that $X^{(r)}$ and $Y^{(r)}$ are used. This happens because they represent the residual matrix of the iteration number r of the outer loop, except for the first iteration, where they represent the treated input data matrices. The subscript i represents the number of the iteration of the inner loop.[8][9]

$$W_i = \frac{X^{(r)T}U_{i-1}}{U_{i-1}^T U_{i-1}} \qquad (U = Y^{(r)} \quad for \quad i = 1) \tag{1}$$

$$C_i = \frac{Y^{(r)T}T_{i-1}}{T_{i-1}^T T_{i-1}} \qquad (T = X^{(r)} \quad for \quad i = 1) \tag{2}$$

Upon being calculated, the X weight matrix (W) is normalized trough equation (3).

$$W_i = \frac{W_i}{\sqrt{W_i^T W_i}} \tag{3}$$

After normalization, the X-score – T (N, A) – and Y-score – U (M, A) – matrices will be calculated trough equations (4) and (5). These matrices will be used to calculate the weight matrices of the next iteration within the inner loop.

$$T_i = X^{(r)}W_i \tag{4}$$

$$U_i = Y^{(r)}C_i \tag{5}$$

At the end of each iteration the constrain in (6) is tested. The inner loop will run as long as this condition is not met.[9]

$$(W_i - W_{i-1})^T(W_i - W_{i-1}) < 10^{-6} \tag{6}$$

Once the inner loop is finished, the scores will once again be calculated trough equations (4) and (5), using the weights obtained at the end of the inner loop. These matrices are of particular importance since they act as estimators of X and Y. With them, it is also possible to calculate the loading matrices: P (K, A) for X and Q (N, A) for Y.[10][8]

$$P = \frac{X^{(r)T}T}{T^T T} \tag{7}$$

$$Q = \frac{T^{(r)T}U}{U^{(r)}U} \tag{8}$$

At this point, it is possible to make estimations of the X and Y data through the calculated matrices, as shown in equations (9) and (10).

$$\hat{X} = TP^T \tag{9}$$
$$\hat{Y} = UQ^T \tag{10}$$

This estimations will be subtracted to the X and Y matrices, in order to get the residual matrix for the next iteration, after which the algorithm will go back to the inner loop step. There will be A iterations, being A the number of components of the model i.e. the number of latent variables (model dimensions). Once the algorithm is finished, it is possible to get predictions of Y through X, since the X-scores have the property of being predictors of Y. This will be done through equation number (12), where W* represents a transformed weight matrix and a centred and scaled (possibly) response matrix is obtained - Y*.[8]

$$W^* = \frac{W}{P^T W} \tag{11}$$

$$Y^* = XW^*C^T \Leftrightarrow Y^* = XB^* \tag{12}$$

Through the equation shown above, it is possible to get the coefficient matrix B*, however, it cannot be applied directly to an

X matrix without previous treatment of the latter. Therefore, this matrix needs to be treated, using the standard deviation (std) of both the input and output matrices, in order to be used with an untreated X matrix.

$$B = B^* \frac{STD(Y)}{STD(X)} \tag{13}$$

Once treated, prediction of new values will be accomplished though equation (14). It is worth noting that, in the equation displayed bellow, both means refer to the mean of the training matrices.

$$Y = XB + (Y_{mean} - X_{mean}B) \tag{14}$$

PLS2 has the disadvantage that, by modelling all the response variables at the same time, the number of components of the model will be the best compromise between the optimal value for each variable as obtained by CV. PLS1 presents the advantage of being able to eliminate some of the effects of interactions between response variables, which, in some cases, is a cause of nonlinearity.[11]

Regarding the other partial least squares (PLS) algorithms, PLS1 and PLS2 differ from them in the step referring to the update of the residual matrix after each iteration. The advantage of this method of updating the residual matrix is more evident in cases with just one response variable since, when using other algorithms, it is easier to get a null residual matrix, terminating the algorithm before the loop has gone through A iterations. [10]

2.2.2 Choosing the Model's Dimensions

The model's dimensions will determine the number of iterations, therefore it is important to determine this parameter correctly, avoiding overfitting i.e. getting a well fitted model with very little to none predictive power. This will be done by applying CV to models with different numbers of components and then choosing the one with the higher score.[8] [3]

Cross-validation consists on splitting the data into validation and training data. Then, the model is fitted to the training data and predictions are made with the validation data. The predictions are then compared with the actual values and the overall prediction error sum of squares (PRESS) (equation (15)) is calculated, estimating the predictive capacity of the model. The higher the prediction capability of the model, the lower the PRESS will be.

$$PRESS = \sum_{i}^{N} (Y_{predicted} - Y_{measured})^2 \tag{15}$$

Although this parameter is a good indicator of the overall prediction capability of the model, in this work, at the end of the cross-validation process, the mean squared error of cross-validation (MSE CV) - the PRESS score divided by the number of samples - will be the score displayed, making it easier to compare models with different number of samples in the input data matrix. Additionally, the goodness of prediction (Q^2) - equation (16) - will also be computed and displayed. This scoring method yields a score between 0 and 1 (it is possible for the score to be negative, however said case will correspond to a model which cannot predict any data correctly), that always presents itself as being equal or lower than the goodness of fit (R^2), being that a model is good at predicting whenever it scores above 0.5. This scoring method allows for a better and easier interpretation of the model's quality, especially in multiple output cases, since the order of magnitude of the outputs will vary.[12]

$$Q^2 = 1 - \frac{PRESS}{\sum_i^N (y_i - \bar{y})^2} = 1 - \frac{PRESS}{TSS} \tag{16}$$

As a result of using centred and scaled matrices when applying cross-validation, the calculation of total sum of squares (TSS) will be simplified, since this parameter will correspond to the number of samples in the matrix.

The two most widely used types of cross validation are the k-fold cross validation and leave one out cross-validation (LOOCV). In k-fold cross validation, the data is randomly divided into k groups (folds) of approximately equal size. The model is then fitted on k-1 groups and predictions are made using the remaining group, which will act as a validation set. A score is calculated and the procedure will be repeated k times, so that each group will act as a validation set once[3]. In LOOCV, the procedure is similar to the one in k-fold CV, being that k will be equal to the number of samples, so that each sample acts as a validation group once. At first glance, this type of CV poses itself as the most thorough, however this type of cross-validation tends to choose the model of excessive size instead of the optimal model.[3] [13] [8]

In order to choose the number of components that yields the best model, a 10-fold CV will be applied to several models, each of them fitted using a different number of components. This parameter must be an integer number greater or equal to one and lower or equal to the overall number of x variables. Once CV has been applied to the different sized models, the MSE CV scores will be compared and the one that has the lowest one will be the chosen model.

2.2.3 Statistics

Since the point of the model is to predict new values of Y, there is a need to apply statistical tests on the input data, so that some information regarding the applicability of the model can be obtained. In this work, two different tests will be applied: the distance to the model of an observation (DmodX) and the Hotelling's T^2.

PLSR is also a projection method, therefore it creates a projection of the X-matrix (interpreted by the X scores) creating a plane with the same dimensions as the model. The distance to the model test, a moderate outlier detection tool, calculates the distance of a sample point to this plane through equation (17). If the distance to the model is being calculated for a sample point of the training data, equation (17) should be multiplied by a distance to model correction factor (ν), obtained trough equation (18). This factor takes into account the fact that the DmodX is expected to be smaller when calculated for an observation that is part of the training data.

$$DmodX = \sqrt{\frac{\frac{\sum_k e_{ik}}{K-A}}{\frac{\sum_i \sum_k e_{ik}}{(N-A-A_0)(K-A)}}} \tag{17}$$

$$\nu = \frac{N}{N - A_0 - A} \tag{18}$$

As it will be discussed further in this work, the equation for the correction factor (equation (18)) presented in the SIMCA® 15 User guide [14] does not yield satisfactory results and therefore was considered to be wrong. As such, for reasons explained in section 4.3.1, equation (19) was used for this parameter.

$$\nu = \frac{N}{N - 0.542 \cdot A + 0.487} \tag{19}$$

As shown above, the equation for DmodX is dependent on the number of X variables (K), number of samples in a dataset (N) and number of components of PLS model, models dimensions (A). The A_0 parameter takes a value of 1 if the matrix has been centred, and 0 otherwise. The e_{ik} parameter represents the residual for the variable k in the ith sample. It is calculated through the subtraction of the data matrix being evaluated (after centring and/or scaling) with the estimation made with de X-score and loading matrices.

$$\begin{bmatrix} e_{11} & \cdots & e_{1k} \\ \vdots & \ddots & \vdots \\ e_{i1} & \cdots & e_{ik} \end{bmatrix} = X - TP^T \tag{20}$$

When calculating the distance to the model of a prediction set there is the need to calculate a new X-score matrix, since this is the only non-model specific matrix (equation (21)). This can be easily achieved through the rotation matrix, W* (equation (11)) .[8]

$$T_{prediction} = X_{prediction}\frac{W}{P^TW} \tag{21}$$

$$\Leftrightarrow T_{prediction} = X_{prediction}W^*$$

For each sample, the distance to the model will be calculated and compared with the critical distance. Any sample with a distance higher than the critical one, will represent a possible outlier and a sample with a distance higher than twice the critical distance is considered an outlier. This parameter is represented by the squared root of an inverse cumulative F-distribution function for a significance level of 5%. The number of degrees of freedom of the denominator of said F-distribution amount to the number of degrees of freedom of the model.[14] [15]

$$DOF_{model} = \sqrt{(N - A_o - A)(K - A)} \tag{22}$$

The degrees of freedom (dof) of the numerator of the F-distribution in question match with the number of degrees of freedom of the observations in the training set. The way to calculate them will be dependent on the value of the dof of the model, as such:

$$DOF_{observations} = \frac{M + \sqrt{K - DOF_{model}} - A}{\nu} \tag{23}$$

$$K > DOF_{model}$$

$$DOF_{observation} = \frac{M - A}{\nu} \tag{24}$$

$$K < DOF_{model}$$

When calculating degrees of freedom, the M parameter represents the minimum value between K, the DOF_{model} and 100.[14]

Although a data point may be close to the model plane, it is in the centre of the model where has more significance. Therefore, the distance to the centre of the model of the projection of an observation in the model plane will be calculated through Hotelling's T^2. [14]

$$T_i^2 = \sum_K \frac{(t_{ik} - \bar{t_k})^2}{s_{tk}^2} \tag{25}$$

This test is dependent on the X-score value for observation i and number of components k (t_{ik}), as well as the mean ($\bar{t_k}$) and standard deviation (s_{tk}) for the X-score vector with k number of components i.e. the mean and standard deviation of the X-score matrix's column number k.The critical distance for Hotelling's T^2, is calculated through a cumulative F-distribution with a 5% significance level. The dof of the numerator of this distribution match the number of components, while the dof of the denominator correspond to the subtraction between the number of observations and the model's dimensions. This test allows for the exposure of outliers through the following constrain:

$$T_i^2 > \frac{A(N-1)}{N-A}F_{critical}(p) \tag{26}$$

The samples with a T^2 value higher than the critical distance at a significance level of 5% (p = 0.05) represent suspected outliers, while the samples with a T^2 higher than the critical distance at a significance level of 1% represent outliers.[14]

3. Software Tools

3.1. Python™language

For the creation of the tool, Python programming language was used. Python is an interpreted, interactive, object-oriented programming language which incorporates modules, exceptions, and classes[16]. Scikit-learn was the Python module used to apply machine-learning. It integrates a wide range of state-of-the-art machine learning algorithms for supervised and unsupervised problems[9].

3.2. gPROMS software®

For the implementation of the models created in dynamic simulations, the gPROMS® platform was used, more specifically, the gPROMS FormulatedProducts® software, developed by Process Systems Enterprise (PSE). gPROMS FormulatedProducts is used for integrated digital design of robust formulated products and their respective manufacturing processes[17]. The gPROMS platform allows the usage of external software components - foreign object (FO) - which provide a way of importing data. In the present work, a single FO was used, the pyFO, of internal PSE development, responsible for trading of information between the platform and Python [18].

3.2.1 Global system Analysis

gPROMS has the capability of performing uncertainty analysis with Monte Carlo methods on certain models trough the global system analysis (GSA) tool. This tool was used for a sensitivity analysis - the study of how the variance of the outputs of the model is dependent on the inputs factors affected by uncertainty.[18]

3.3. SIMCA® software

SIMCA is a statistical software used for multivariate data analysis. It allows for the interpretation and visualisation of large amounts of data, batch data and time-series data, among others. By being able to analyse process variations, identify critical parameters and predict final product quality, this software becomes suitable for data mining and process modelling. [19]

4. Granulation Case Study

Like with any other model, there is a need for the validation of the PLS model and, in this case, for the validation of the implemented statistical tests. With this purpose, a PLS model was fitted to data from a granulation case study, which was previously ran trough the umetric's SIMCA® software, creating a PLS model and applying the distance to model and Hotelling's T^2 tests.

4.1. Process description

In the case study in question, data from a granulation process was provided by AstraZeneca (AZ) as a part of the Advanced Digital Design of Pharmaceutical Therapeutics (ADDoPT) project. This project consists of a four year collaboration between pharmaceutical companies, solution providers and academia aiming to make existing and new digital design approaches widely usable within the pharmaceutical industry, increasing efficiency and effectiveness of drug development and manufacture.[20]

The flowsheet referring to the process where the training data for the model was collected (Figure 1) was also provided and assembled by AZ. In this dry granulation process, powder particles are fed to a roller compactor where they are compacted into a ribbon and subsequently milled. The process ends with the compacting of the granules from the mill in the tablet press. Linked to the output stream of the mill there is a particle size distribution sensor (PSD_sensor) that will obtain the results of a particle size distribution for said stream, namely the cumulative diameters (D_{10}, D_{50}, D_{90}) and the volume weighted mean (D_{43}) , which will then act as inputs for the model. This is possible since the data based sensor block (Sensor_data_based), a block that contains the developed model, was added to the flowsheet. This block receives the cumulative and volume mean diameters from the particle size distribution and calculates prediction for the flow-function coefficient (FFC).

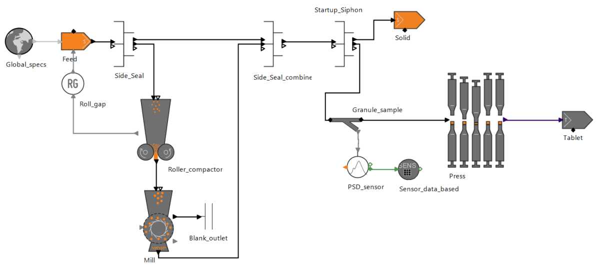


Figure 1: AZ's process flowsheet.

4.2. Model Development and Validation

The above mentioned data refers to several particle size distributions and the parameters obtained from it, i.e the cumulative diameters and the D_{43}. The cumulative diameters amount to the diameter at which a certain percentage of the particle sample has a diameter lower or equal than said value. Both the cumulative diameters and D_{43} are used as inputs for the model, in order to calculate the FFC. This coefficient refers to a function which is of importance in granulation, since it allows for the calculation of the flow performance of the powder. All these parameters were calculated for 19 different samples.

These parameters were loaded into SIMCA, which created a PLS model with 3 components. This software also applies CV to different sized models and chooses the one with the best CV score. It should be noted that a logarithmic transformation was applied to the output data in order to normalize it.[21]

The input data were inserted into Python and, with the help of the scikit-learn package, a PLSR was applied, along with a 10-fold CV in order to choose the optimal number of components. A model with 3 components, a Q^2 score of 0.649 and R^2 score of 0.801 was chosen due to its superior prediction capability (higher Q^2), which is in line with what was obtained in SIMCA.

To help visualise the behaviour of the different models during CV, the plotting of a validation curve was implemented (Figure 2). This curve represents both the training and validation scores obtained during CV for the different numbers of components possible, that is, the scores for the training and test sets. Since in CV 10 different scores are calculated, one for each fold, the errors calculated will have a certain degree of uncertainty. Because of this, a std band (calculated trough Equation (27)) is shown alongside the curves, quantifying the degree of uncertainty for each score.

$$std = \sqrt{\frac{\sum_{i=1}^{N} (y_{predicted} - y_{measured})^2}{N}} \tag{27}$$

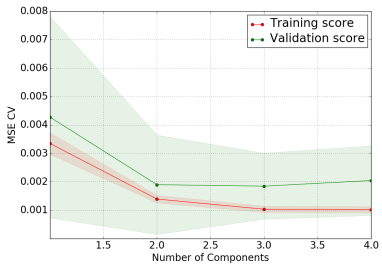


Figure 2: Validation curve for the granulation case.

Having obtained the same results as SIMCA, the tool's capability of generating a PLS model with the optimal number of components was validated. In order to visualize and better judge the prediction capabilities of the model, there is the need to look at the plot of the predicted values against the the output training data (Figure 3). These values are analysed using a y=x plot as reference (the closer the values are to this reference line, the better the predictions) along with a band, which indicates the expected area where the predictions will land, according to the std of the predictions in CV. Additionally, the plots also display the values for the confidence intervals calculated for confidence levels of 95% and 99%. These intervals correspond to the range were the prediction are expected to land with 95% or 99% or certainty.

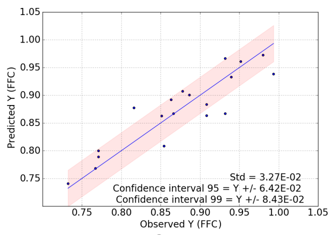


Figure 3: Predicted Y vs. Observed Y plot with reference.

Trough the plot displayed, it can be seen that most of the predictions land close to the reference plot within the std band. This allow us to classify the quality of the predictions as being good. Some deviations are shown, but they are not severe.

Due to the fact that there is only a small number of observations in the training set, there was the suspicion that the observed deviations might have been caused by overfitting. To asses this, a learning curve plot (Figure 4) was implemented. For this plot, a model is fitted for different numbers of observations (training examples) and both the training score of the model and the CV score are plotted alongside uncertainty bands, similar to the ones in the validation curve.

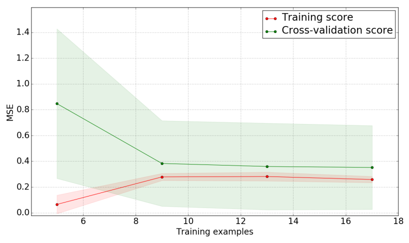


Figure 4: Learning curve for the granulation case.

For a small number of training examples, the learning curve shows a large separation between both the scores, indicating that, for up to 8 examples in the training set, the training error is low (there is a good fit to the training data) but the error for the CV tests is high (low quality predictions during CV). This facts paired together constitute overfitting. For higher numbers of samples the errors converge and there is a relatively small difference between both scores, indicating that the number of samples used was appropriate and that there was no overfitting. The high degree of uncertainty in the CV score indicates that it is not possible to claim with total certainty that there is no overfitting. However, having no more data available, it was considered that there is no overfitting.

4.3. Statistical tests validation

An important step in judging the quality of the data set and, therefore, the model's, is outlier detection through statistical tests. As such there is a need to validate the implemented functions and cross reference their results with the ones obtained from the SIMCA report.

4.3.1 Distance to the model

Naturally, the first statistical test applied is the DmodX, measuring the distance of a sample point to the plane created by the model. Both equations (17) and (18) were implemented, as the tests are being applied to the training data.

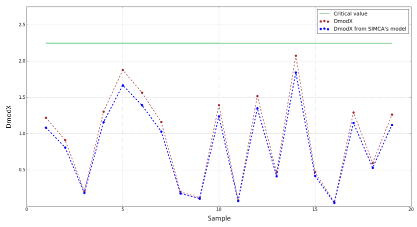


Figure 5: Comparison plot between the DmodX results from the implemented functions and the ones from SIMCA's model.

A clear difference between the results and SIMCA's report can be noted in Figure 5. Upon using some of the training data as a prediction set in SIMCA, the value for DmodX without a correction factor was obtained, presenting the same value as the implemented function based on equation (17), what leads to the conclusion that the deviation in results is caused by an error in the correction factor. Consequently, an hypothesis was created (equation (29)) that the correction factor was only dependent on N and α (a parameter dependent on A).

$$\nu = \frac{N}{N-\alpha} \tag{28}$$

This parameter was then calculated for the correction factors obtained with the unchanged training data set, for a number of components ranging from 1 to 3 and the obtained results were fitted using a linear regression, having shown a high linear relation as such:

$$\alpha = 0.542 \cdot A + 0.487, \qquad R^2 = 0.999 \tag{29}$$

Equation (29) was then promptly implemented, having shown the desired results, displayed in Figure 6

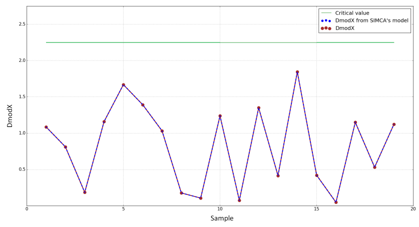


Figure 6: Comparison plot between the DmodX results from the implementation of the new ν and the ones from SIMCA's model.

Since α was obtained for a limited range of the number of components parameter, there is the risk of having a high number of components where the correlation fails. With the validation of equation (29) for high values of A in mind, the first 118 non binary parameters of the dataset presented in reference [22] were inserted into SIMCA and a a model with 118 components was created. The model created was compared with the one obtained in Python using the same data and the value obtained for the correction factor showed a deviation of 0.0097% from SIMCA's. Therefore the hypothesis was considered correct and was applied in the subsequent case studies.

4.3.2 Hotelling's T^2

Regarding the Hotelling's T^2 the implemented function yielded results that match the ones from SIMCA's report (Figure 7) and, therefore, this test was considered validated. From the results it was also concluded that the dataset used to fit the model showed no outliers.

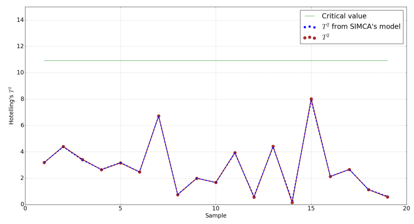


Figure 7: Comparison plot between results from the Hotelling's T^2 implemented and the ones from SIMCA.

4.4. gPROMS Results

Having created the model, the next step was the implementation of the model in a dynamic simulation. The model's implementation is possible because the Python tool allows for the creation of an XML file containing all the relevant information of the model, which will be loaded into the gPROMS software (it is linked to the data based sensor block), that will predict values for the FFC over time. This pairing is only possible trough a FO that allows for the trading of information between Python and gPROMS.

The simulation ran for 1800 s, with the PSD_sensor block registering different values for the particle size distribution. The values predicted for the FFC during the simulation are presented in Figure 8.

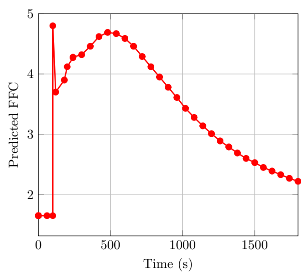
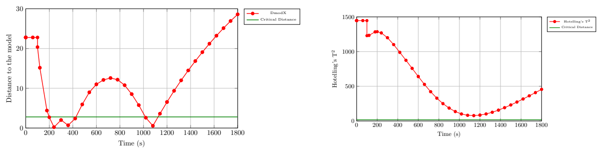


Figure 8: Predicted values for the FFC over time

As it can be seen, the model was able to make predictions when included in a dynamic simulation. However, during the simulation warnings were triggered. Hence, both the results of the statistical tests and the values taken by the model's inputs must be analysed in order to asses what triggered the warning. Figure 9 shows that, even though DmodX not always took a value over the critical distance, the Hotelling's T^2 consistently had values over the critical bound throughout the entire simulation.



(a) DmodX **(b)** Hotelling's T^2

Figure 9: Statistical tests for the predictions of the FFC.

By looking at the comparison plot, presented in Figure 10, it becomes apparent why the statistical tests failed. By having the input parameters assuming values higher than their maximum/ lower than their minimum values in the training data, it is assured that the observation points will present higher distances to the model plane/ its centre.

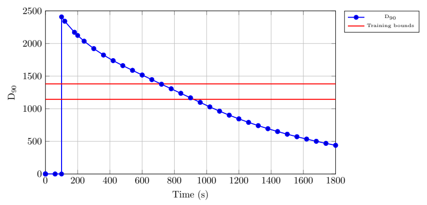


Figure 10: Values of D_{90}, μm, during the simulation and respective training bounds.

The model was fitted with data within certain bounds, however, by exceeding those bounds, there isn't a guarantee that the quality of the predictions will uphold for the new input data. This fact constitutes one of the main limitations of these type of models: no matter how broad the range of the training data, the quality of the predictions made with inputs outside that range can never be assured. Therefore, even though the main goal of soft-sensing the FFC was accomplished, the predictions cannot be considered reliable.

5. Compression Case Study

Having the model creation/analysis tools validated, it was now possible to fit a model to new data. Therefore a PLS model was developed for a compression case study. In this case study, with data provided by Pfizer as a part of the ADDoPT project, a model was created for the prediction of the compressibility of a blend, defined as the fraction of volume reduction under pressure[23], from particle size distribution data. Analogously to what happened in the previous case study, the output was logarithmically transformed and predictions were made for this transformation.

5.1. Process description

The flowsheet in Figure 11, provided and assembled by Pfizer, represents the process where the training data for the model was collected. In this process, a fenofibrate is crystallized, milled and then mixed with an excipient. This blend will go trough a particle size distribution (blend_PSD block), whose results will be inserted into the data based sensor (Sensor_data_based), namely the cumulative diameters (D_5, D_{10}, D_{16}, D_{25}, D_{50}, D_{75}, D_{84}, D_{90} and D_{95}), the Sauter mean diameter (D_{32}) and the D_{43} , in order to predict the compressibility of the blend in the roller compactor - parameter that influences the efficiency of this step of the process. The blend will then be milled and tablets will be created due to the usage of a tablet press.

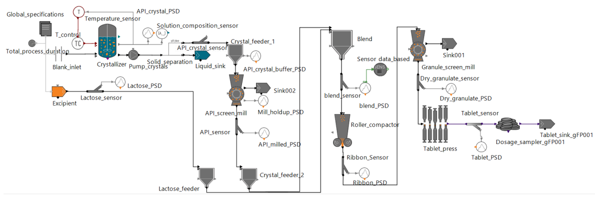


Figure 11: Pfizer's process flowsheet.

5.2. Model Development

This time, the data provided consisted only of the raw data of the particle size distribution, having 100 different samples. With the diameters of the sieves and their respective volume fractions, the cumulative diameters D_5, D_{10}, D_{16}, D_{25}, D_{50}, D_{75}, D_{84}, D_{90} and D_{95}, the D_{32} and the D_{43} were calculated. The data was then loaded to Python and a model with 4 components and a Q^2 and R^2 scores of 0.737 and 0.822, respectively, was obtained.

By interpreting the validation curve (Figure 12) it becomes apparent that there is a high degree of uncertainty prediction wise. For the optimal number of components case, this indicates that albeit the score was the lowest obtained, there is still the possibility to obtain high prediction errors with this model.

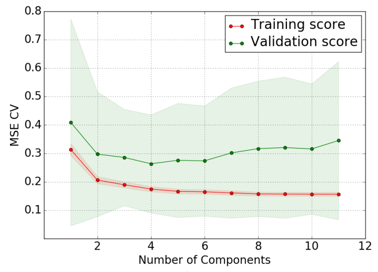


Figure 12: Validation curve for the compression case study.

From the interpretation of Figure 13 it is clear that for high compressibility values, the predictions are close to the reference plot and within the uncertainty band, presenting high quality. However, as the values of compressibility decrease, the deviations from the reference plot increase, with most points going outside the uncertainty band, so much so that a point is reached when the plotted points have non-negligible deviations from the reference and the uncertainty band. Therefore, for the given model, predictions made for lower values of compressibility are not reliable.

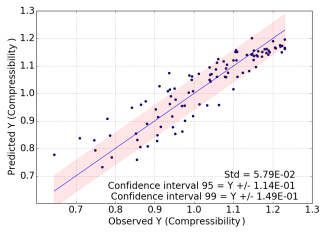


Figure 13: Predicted Y vs. Observed Y plot for the compression case study.

5.2.1 Polynomial Transformation

One possible cause for the bad quality of the predictions for lower values of compressibility is the simplicity of the model. In order to try and correct this issue, a 2nd order polynomial transformation was applied to add complexity to the model. This non-linear transformation adds as input the value 1 (which, in this work, will always have a null coefficient) and the multiplication of each parameter by itself an by the remainder parameters. By training the data on non-linear functions of the input data, it is assured that the PLSR is able to fit to a wider range of data while still generating a linear model. [9]

The transformed input data, now with 78 inputs, was fed to the tool and the result was a model with 13 components, a Q^2 score of 0.795 and a R^2 score of 0.883. The slice of the validation curve in Figure 15 shows that besides having a better score, the model obtained also has a lower degree of uncertainty associated.

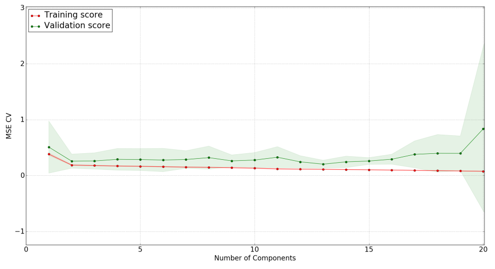


Figure 14: Validation curve for compression case study with polynomial transformation.

Figure 15 shows the new model was able to maintain a good quality of prediction for higher values of compressibility (although some loss in quality is registered), while being able to get better predictions for lower values.

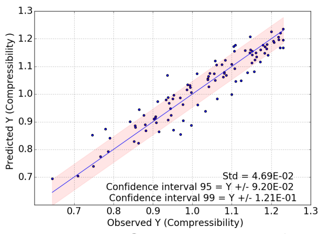


Figure 15: Predicted Y vs. Observed Y plot for the compression case study with polynomial transformation.

The results shown in the aforementioned plot serve to show that, by adding complexity to the training data, the polynomial transformation added robustness to the model, since it is now suitable for predicting values within the range of the compressibility in the training data.

The detection of outliers leads to a critical analysis of the data to try to determine if these observations are explained by the process or not i.e. if the deviations were caused by certain operational conditions. If that is the case, then the observations are not removed from the dataset, if not they are removed. Since the data was provided by a third party, the existing knowledge of the process is not enough to do the required analysis and so it was assumed that the outlier points were explained by the process.

5.3. gPROMS Simulation Results

The relevant information regarding the chosen model was stored into an XML file, which was then loaded into gPROMS with the aim of soft-sensing the compressibility of a blend. The simulation ran for 18000 and predictions for the compression of the blend in the Roller Compactor for each point in time were made, as reported in Figure 16.

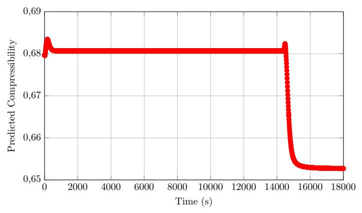
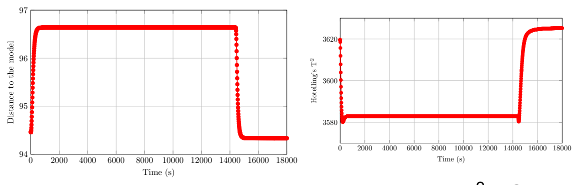


Figure 16: Predicted values for the compressibility of the blend in the Roller Compactor over time

The sensor was able to predict the value for the compression of the blend over time, not registering significant fluctuations, however during the simulation a warning was triggered. To asses the cause of this warning the statistical tests were analysed.



(a) DmodX (Critical Distance = 1.22) **(b)** Hotelling's T^2 (Critical Distance = 27.14)

Figure 17: Statistical tests for the predictions of the compression in the Roller Compactor.

From the results of the statistical tests (Figure 17) it becomes evident that both tests present a distance well above their respective critical distance. This is consequence of the fact that some input parameters go well above the maximum and minimum values established in the training data, as it can be seen in Figure 18.

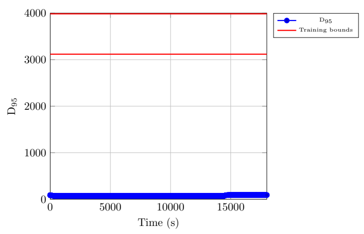


Figure 18: Values of D_{95}, μm, during the simulation and respective training bounds.

Even though the model was able to predict the compressibility of the blend in the roller compactor, it did so using parameters outside the range the model was intended to be used in and, therefore, the predictions are considered unreliable.

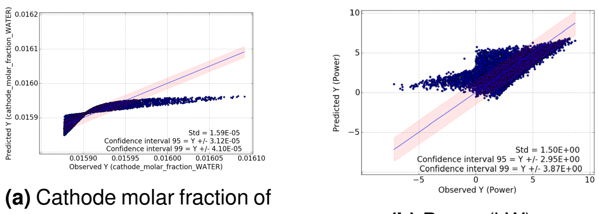
6. Solid Oxide Fuel Cells Case Study

The case study discussed in this chapter had a different objective than the previous ones. Instead of having the goal of creating a predictive model for soft-sensing, the main goal was to asses how the tool would behave when creating a predictive model using the PLS 2 algorithm i.e creating a model with multiple outputs. In this case, training data was obtained from a GSA applied to the operational conditions of a solid oxide fuel cell (SOFC) model from gPROMS AML-FC library, in order to fit a model that could predict the power and voltage of the fuel cell, the molar fractions of O_2, N_2 and H_2O and the out temperature of the cathode and the molar fractions of H_2O, H_2, CO and CO_2 and the out temperature of the anode.

6.1. Model Development

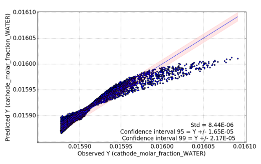
The dataset used to create the model consisted of 10000 samples points of 10 input variables: the current density; the air's flow rate, temperature and pressure; the syngas's CH_4, CO, H_2 and H_2O content and its flow rate and pressure. These data points were directly loaded to the Python tool, without any transformations, and a model with 10 components, a Q^2 score of 0.7726 and a R^2 score of 0.7733 was obtained. Upon applying a 2nd order polynomial transformation, a dataset with 66 input variables was loaded into python and a model with 65 components, a Q^2 score of 0.939 and a R^2 score of 0.940 was obtained.

From the plots of the predicted values against the observed values (Figures 19 and 20) it can be seen that the model is incapable of making good predictions for both cases, even though improvements can be seen after the polynomial transformation. This plots will only be shown for the cathode molar fraction of H_2O and power because these parameters show the worst predictions.

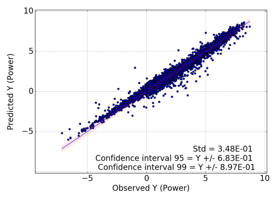


(a) Cathode molar fraction of H_2O. **(b)** Power (kW).

Figure 19: Predicted Y vs. Observed Y plot for the SOFC case study without transformations.



(a) Cathode molar fraction of H_2O.

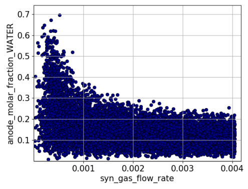


(b) Power (kW).

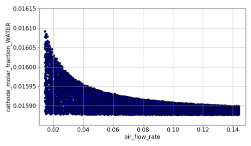
Figure 20: Predicted Y vs. Observed Y plot for the SOFC case study with polynomial transformation.

6.1.1 Reciprocal Transformation

While looking at the plotting of the response variables against the inputs, it was noted that some variables display a reciprocal dependency i.e the outputs were obtained through the inverse of the inputs($\frac{1}{x}$). This is clear in the plots from Figure 21.



(a) Anode molar fraction of H_2O Vs. Syngas flow rate (mol/s).

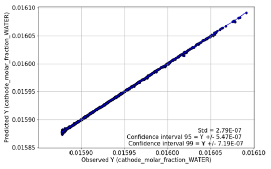


(b) Cathode molar fraction of H_2O Vs. Air flow rate (mol/s).

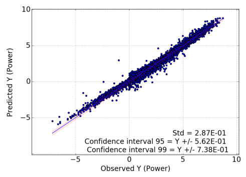
Figure 21: Variables with a clear dependency trough a reciprocal function.

Due to these observations, a reciprocal transformation was applied to the dataset, prior to the polynomial transformation. This transformation consists on adding the inverse of each input variable to the dataset. Upon combining the referred transformation with the polynomial one, a model was fitted to a dataset consisting of 231 inputs. The optimal case obtained consisted of a predictive model with 95 components, a Q^2 score of 0.990 and a R^2 score of 0.994.

The plots of the predicted values against the observed ones (Figure 22) show significant improvements from the previously obtained models. The cathode molar fraction of H_2O now presents all points extremely close to the reference, which is a direct consequence of adding the reciprocal transformation. Albeit the power shows a slight improvement, there are still some points with a significant distance to the reference.



(a) Cathode molar fraction of H_2O.



(b) Power (kW).

Figure 22: Predicted Y vs. Observed Y plot for the SOFC case study with reciprocal and polynomial transformation.

The results obtained show that the tool was able do successfully create a model for the prediction of multiple outputs. However, they also show that, even though the inputs used are known to influence the outputs, the dataset might not have the required complexity for the creation of a good predictive model.

Since the data was obtained from a GSA, all points are considered to be explained by the model and, therefore, outlier detection was not necessary.

7. Conclusions

This work aimed to develop a prototype tool, in Python, able to create predictive models for the purpose of soft-sensing parameters in systems that are usually not modelled. This happens either because they are poorly understood and, therefore, first principle models cannot be developed, or because they present non-linear relations between different parameters that make them extremely hard to model trough first principal models.

The literature review showed that the PLSR was the algorithm best suited for the problems tackled in this work, in detriment of the MLR, used for simpler multivariate cases.

The tool's ability to create predictive PLS models was validated against a model for a granulation case, generated by SIMCA®, a statistical software that allows for the creation of PLS models, trough plant data. The validated model was subsequently loaded into gPROMS FormulatedProducts®, trough an internally developed foreign object, being able to successfully soft-sense the intended parameter, the FFC. The simulation's results led to the conclusion that one of the most important steps in creating a predictive model is to choose the training data carefully. Even though the model was able to predict the FFC, the simulation was made with operational conditions different from those used to generate the training data. This leads to the calculation of predictions outside the range the model was intended to be used in, generating unreliable results. Hence, when using this tool, the training data used to fit the model must be generated in a range of operational conditions that cover the ones used in the simulations, thus assuring the reliability of the predictions.

Being that the problems tackled have unknown relations between its parameters, the training data loaded into the tool might not always posses the required complexity to successfully create a good PLSR model, as evidenced by the compression case. Therefore, to widen the range of applicability of the tool, a 2nd order polynomial transformation option was added. With this option, it was possible to increase the complexity of the training data through a commonly used transformation whenever the tool fails to get a good predictive model from the raw input data. The need to apply a transformation is one of the main limitations of this tool, since certain problems may require the application of transformations that aren't commonly used and, therefore, are not implemented.

The decision that the tool required training data with higher complexity was also shown in the SOFC case study. In this case, even after applying the polynomial transformation, the tool failed to get a proper predictive model. This was corrected by adding the option to apply a reciprocal transformation, which can be combined with the previously implemented polynomial transformation, further increasing the tool's applicability to different cases. Additionally, this case led to the conclusion that the PLSR was able to successfully create a single predictive model for several response variables.

7.1. Future Work

In what concerns the tool itself, it still has room for improvement, specially regarding data transformations. As it is applied to other cases, further transformations can be implemented, not only to the predictor variables, but also to the response variables.

Regarding the models integration in dynamic simulations, since it has been proved that it is able to soft-sense certain parameters, it can be integrated into the simulation itself. This would be accomplished by using the calculated predictions as inputs for other models.

References

[1] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven Soft Sensors in the process industry," *Computers and Chemical Engineering*, vol. 33, no. 4, pp. 795–814, 2009.

[2] S. Zendehboudi, N. Rezaei, and A. Lohi, "Applications of hybrid models in chemical, petroleum, and energy

systems: A systematic review," *Applied Energy*, vol. 228, no. December 2017, pp. 2539–2566, 2018. [Online]. Available: https://doi.org/10.1016/j.apenergy.2018.06.051

[3] G. James, D. Witen, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*, 2007, vol. 64, no. 9-12.

[4] M. N. O. Sadiku, S. M. Musa, O. M. Musa, and R. G. Perry, "Machine Learning in Chemical Industry," *International Journal of Advances In Scientific Research and Engineering (IJASRE)*, vol. 3, no. 10, pp. 12–15, 2017.

[5] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data Mining and Analytics in the Process Industry: The Role of Machine Learning," *IEEE Access*, vol. 5, pp. 20 590–20 616, 2017.

[6] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: advantages, challenges, and applications," *Production & Manufacturing Research*, vol. 4, no. 1, pp. 23–45, 2016.

[7] R. D. Tobias, "An introduction to partial least squares regression," *SAS Conference Proceedings: SAS Users Group International 20 (SUGI 20)*, pp. 2–5, 1995.

[8] S. Wold, M. Sjöström, and L. Eriksson, "PLS-regression: A basic tool of chemometrics," *Chemometrics and Intelligent Laboratory Systems*, vol. 58, no. 2, pp. 109–130, 2001.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2012.

[10] J. Wegelin, "A survey of Partial Least Squares (PLS) methods, with emphasis on the two-block case," p. 371, 2000.

[11] T. Hasegawa, "Principal Component Regression and Partial Least Squares Modeling," 2006.

[12] V. Consonni, D. Ballabio, and R. Todeschini, "Evaluation of model predictive ability by external validation techniques," *Journal of Chemometrics*, vol. 24, no. 3-4, pp. 194–201, 2010.

[13] S. Jun, "Linear model selection by cross-validation," *Journal of the American Statistical Association*, vol. 128, no. 1, pp. 231–240, 1993.

[14] S. stedim Biotech, "Simca® 15 User Guide - Multivariate Data Analysis Solution."

[15] G. K. Sofer and A. S. Rathore, *Process Validation in Manufacturing of Biopharmaceuticals*, 3rd ed.

[16] Python Software Foundation, "Python$^{\text{TM}}$," accessed 2018-08-14. [Online]. Available: https://www.python.org/

[17] Process Systems Enterprise Ltd., "PSE: gPROMS - The Platform," accessed 2018-06-12. [Online]. Available: https://www.psenterprise.com/products/gproms/platform

[18] Process Systems Enterprise, "gPROMS ModelBuilder Documentation - Release 4.2.1," Tech. Rep., 2016.

[19] "SIMCA — Umetrics," accessed 2018-08-15. [Online]. Available: https://umetrics.com/products/simca

[20] "ADDoPT – advanced digital design transforming pharmaceutical development and manufacture," accessed 2018-07-09. [Online]. Available: https://www.addopt.org/about_addopt/

[21] C. Feng, H. Wang, N. Lu, T. Chen, H. He, Y. Lu, and X. M. Tu, "Log-transformation and its implications for data analysis." *Shanghai archives of psychiatry*, vol. 26, no. 2, pp. 105–9, 2014.

[22] University of California Irvine, "BlogFeedback Data Set," 2014, accessed 2018-08-08. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/BlogFeedback

[23] A. Michrafy, D. Ringenbacher, and P. Tchoreloff, "Modelling the compaction behaviour of powders: Application to pharmaceutical powders," *Powder Technology*, vol. 127, no. 3, pp. 257–266, 2002.