Improvement in the Performance of Online Control Applications via Enhanced Modeling Techniques

Nunzio Bonavita[♣], *Riccardo Martini, Ted Matsko*
ABB Automation

Keywords: **Process Control, MPC, Neural Networks, SPC, Plant Optimisation**

1 Introduction: Role of Modeling in Process Control

Modern process control is based on process modeling. Advanced process control (APC), real time optimization (RTO), process monitoring, operator training simulation, abnormal situation management (ASM) and fault detection and isolation (FDI) are all based on some kind of process modeling. Models are a very effective way to embed "knowledge" in process automation, which has increased its "autonomy" level, growing more and more from "reactive" to "proactive" [1], [2].

Before entering any discussion about models, it is probably appropriate to define what is meant by the word in this context. We will refer to the following, generally accepted definition, taken by Denn's book [3]: "*A mathematical model of a process is a system of equations whose solution, given specific input data, is representative of the response of the process to the corresponding set of inputs*". It is possible to identify two main approaches to model building:

a. Fundamental or first principle models where a description of the process (and the related automation) is created starting from fundamental laws
b. Empirical models (or data-driven) where the models are developed through a fitting-like procedure over the actual plant live data.

Modelling is a goal-oriented activity: it is performed to answer specific questions, so it should be always taken into account that there is not a "always winning" technique and that no one model is appropriate for all situations [4].

First principle models always provide a causal relationship, while an empirical model may not. The empirical model may just imply that the same driving forces move both the input and output variables, and that the underlying theoretical model provides the relationship. So the user must insure that the underlying process does not change behaviour if an empirical model is used. In practice, full theoretical models are very expensive to derive and only used in full-scale optimisation projects. Process control applications usually employ empirical models [5].

♣ Author to whom all the correspondence should be sent. Address is:
Nunzio Bonavita, ABB Solutions, Via Hermada, 6, 16154 Genova, *Nunzio.Bonavita@it.abb.com*

The two approaches do not compete: not only do they supplement each other very efficiently (for example, using empirical models to determine unknown parameters in rigorous, equation-based models), but they also work better in different areas.

Although the landscape is quite varied, it is possible to classify project applications in the process industry:

1. Equation-based models are more common in "plant-wide" applications, where the interest is related to the description of the plant behavior as a whole;
2. Data-driven models are the reference technology where there is a need for an accurate description of process units with less emphasis on the interactions among them and much stronger requirements on specific details or custom conditions that are not easily known a-priori (just think of equipment wear and tear).

Essentially class 1 includes engineering and training simulators, and real-time closed loop plant optimizers (RTCLO), while class 2 includes most of the typical advanced applications from multivariable process control to inferential measurements, from fault detection and isolation to quality and process performance monitoring.

Empirical models are not new to APC. In fact they are quite common and well accepted, having faded into the background of current interest and development activities. There is still fertile ground for making improvements in the modeling technology of APC applications though.

This paper deals with empirical models and with the possibility to exploit recent technical improvements to overcome some of the present drawbacks.

First we will look at the role of the model in the most common APC technique, Model Predictive Control. The most common techniques used today were selected for their compatibility with the computers of the mid-1980's. Alternate formulations developed in the 1960's, but never commercially exploited, can now play a part bringing better performance to APC.

In the second section, we will examine tools to exploit the abundance of process and laboratory data archived in plant information systems for prediction and monitoring purposes. Excellent commercial tools are available to build highly accurate empirical inferential product quality models, but they lack practical features needed for heavy-duty online application, thus resulting in a much less pervasive presence of process models than it could and should be.

The third part, §4, will describe a suite of products aiming to push a little further the envelope in the APC market . Combining a set of pre- and post- model building tools, with the actual regression engines, provides an overall increase in model utility and robustness. Finally some details on two applications will be given in §5.

2 Improvements in Modeling for Model Predictive Control

2.1 Some Background Information on MPC

Multivariable control first came into common use in the 1980's when several independent sources all began converging on a basic architecture. The key to this architecture is the use

of an internal linear dynamic model in the controller calculation. The algorithm computes an estimate of process disturbances acting on the process variables being controlled. The disturbance estimate, the process variable setpoints and feedforward signal levels become inputs to the controller calculation. With these inputs and the process model, the controller is able to calculate the required values for the independent, manipulated variables. This structure is illustrated in Figure 2.1

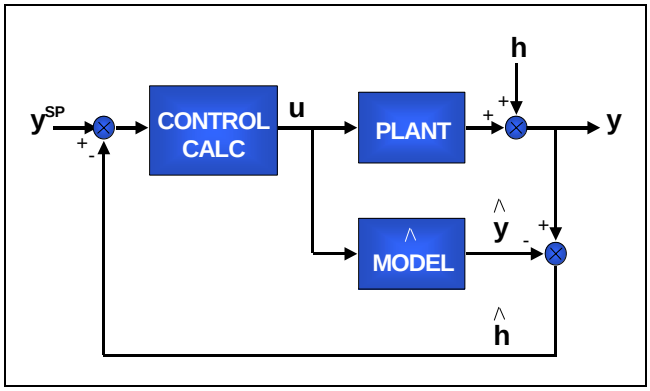

Figure 2.1- Standard MPC Structure

In the controller error minimization calculation, the model helps predict future values of the process variables. This led to the names Internal Model Control (IMC) and Model Predictive Control (MPC). The MPC algorithms became the first large-scale deployment of computer based multivariable process controllers ([6], [7]). To make the calculations efficient and convenient, the algorithms use discrete impulse response models. These models can predict the values of future process outputs through the discrete convolution equation. The equation is fairly simple to program and lends itself to incorporation in the optimisation algorithms needed to calculate the values of future MVs, while minimizing process variable deviations.

MPC algorithms have become the dominant method for dealing with interactive chemical process control problems. They supplanted a technique of using control function blocks. This technique combined feedback PID controllers with feedforward control and decoupling controllers. The basic elements of this method were easier for plant personnel to understand, but the overall schemes were very difficult to manage, especially when constraints come into play. Constraints could enter the problem when actuators saturate or when a process limit is active through a high/low selector function block. The schemes fell apart under the complexities of larger systems and under the conditional behaviour required by varying sets of active constraints. MPC on the other hand, has proven to be very flexible in expanding to large systems and in handling complicated constraint scenarios. An enhancement that first appeared in the early 1990's posed the controller optimisation problem as a multi-objective optimisation, where each stage of the optimisation problem added a new constraint while adhering to the optimal solution for previously solved higher ranked constraints [8]. This innovation made tuning the controllers with varying sets of active constraints much easier.

After the move to multi-objective algorithms, there has been little else to point to as an improvement in MPC or as an alternative to MPC. Advances in computing technology have lead to wider availability of dynamic process modelling tools for the chemical and petroleum related industries. The increased availability of rigorous high fidelity models has not lead to a generally accepted way to use these models for process control. One reason is difficulty in getting the dynamic model to match the plant. There are many details in the model that are needed for process control, which are not necessarily needed for design work, but design work is often the original intent of the model. For instance, details about the valves are often inaccurate and need to be gathered from plant operating data. A second reason is selection of the control architecture. In a linear system, an output disturbance could be modelled by a bias. In a nonlinear system, that choice is not so obvious. The mathematical approaches to

solving the problems also vary. Brute force integration of Open Equation physical systems, as the inner layer of a gradient algorithm computing control moves, is a difficult procedure. The alternate formulation combining the physical, time and control equations into a single objective via collocation produces daunting numerical problems.

Thus MPC is the algorithm of choice for most control problems. This does not mean that there are not shortcomings. There are actually several problems that have not been addressed by the market, as development has tended to be stagnant.

2.2 Problems that Occur with MPC Models

As stated earlier, the common MPC packages employ impulse response models. When they display the models to engineers, it is common to integrate the impulse response models, converting them to step response models, as illustrated in Figure 2.2. These models contain a great number of coefficients. The model identification software packages have many degrees of freedom to use in calculating the models. This in fact does not represent the real system very well, because the real system would have only a few parameters in a linearized set of differential equations. If the data collected from the plant was perfect, this abundance of coefficients would be no problem, but that is not the case. Process data collected from plant step tests always carries process noise and the effect of unmeasured disturbances.

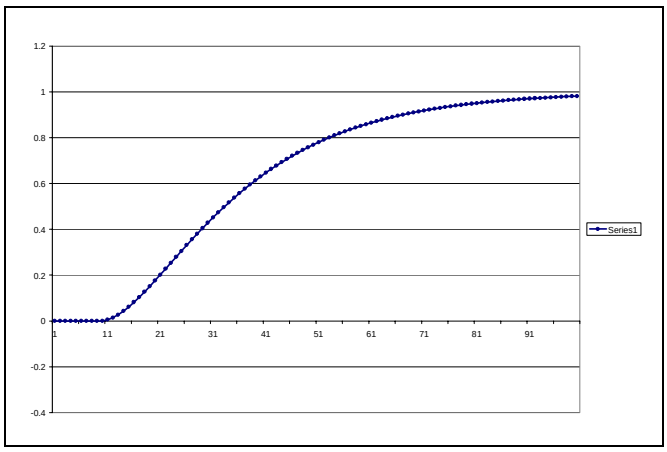

Figure 2.2 - Typical Step Response Model

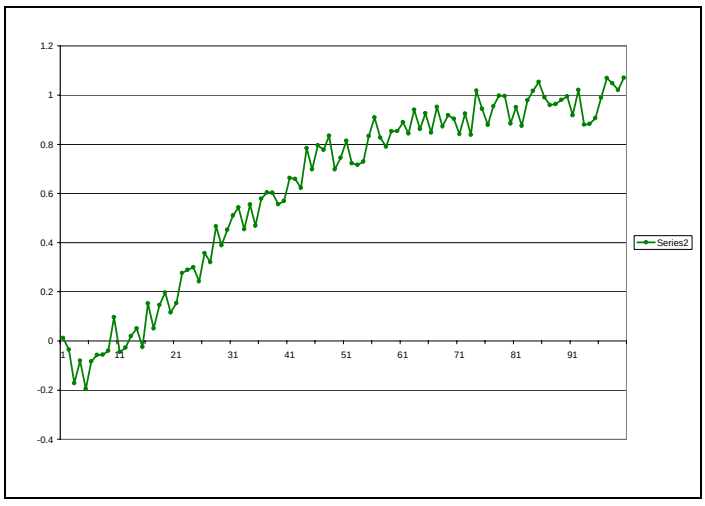

Figure 2.3 - Actual Step Response Identified

The general result is that these extra coefficients begin to fit the noise, as shown in Figure 2.3. The variation in these coefficients affects the MPC algorithm, resulting in unnecessary manipulation of the controller outputs. The engineer must now make a tradeoff, detune the controller, or fix the model. Since detuning is undesirable, most modelling algorithms provide a smoothing function. This is not cost-free because filtering usually tends to attenuate the model gain, see Figure 2.4, and in extreme cases can change the dynamics. The resulting filtered model will not match the real plant, requiring detuning of the MPC algorithm that the smoothing was supposed to avoid.

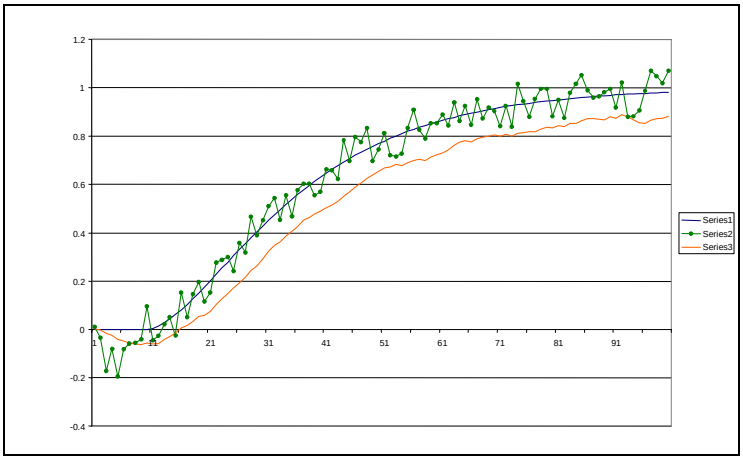

Figure 2.4 - Attenuation of Model

One reason this step response looks ragged is that the model identification does not have the ability to deal with process noise. This type of model does not include parameters to estimate the effect of the noise, see Figure 2.5. There is nothing explicit in the model formulation or the identification algorithm that allows some of the prediction error to be explained by disturbances.

All disturbances or noise added to the process (v) then bias the model coefficients. When you have a lot of model coefficients, there is plenty of opportunity to try and explain the noise with the model, instead of accepting a smoother model and some prediction error. To mitigate this problem, large data sets are needed, which require greater engineering effort collecting plant data.

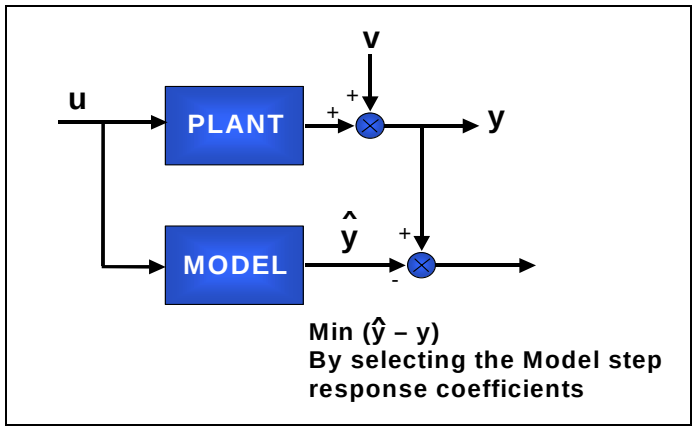

Figure 2.5 - Model Identification

2.3 State Space Models - An Alternate to Impulse and Step Response Models

For a long time the control literature has described modern control algorithms based on a flexible type of multivariable model. The model was based on linear differential equations that mapped the relationships between process inputs and process outputs through use of intermediate variables, called the *state vector*. This type of model was called a state space model. MPC algorithms came along after state space models were introduced, but did not use this type of model.

State space models became linked to optimal control theory for aerospace applications and did not include many of the practical control objectives that were part of the design basis of MPC. The result was that state space models were ignored for a long time by the process industries, but recent enhancements in new algorithms have changed that [10].

The equations that represent a discrete-time state space model are presented in equation 2.1

eq. 2.1

$$x(k+1) = Ax(k) + B_u\,(u(k) + w(k)) + B_d\,d(k)$$
$$z(k) = Cx(k)$$
$$y(k) = Cx(k) + v(k)$$

where x – is the state vector
u – is the process input or control effort vector
d – is a vector of measured disturbance variables, or feedforwards
w,v – are noise vectors
z – is the vector of process variables
y – is the vector of process variables with measurement noise
A, Bu, Bd, and C – are gain matrices
k – is the time in number of sampling intervals

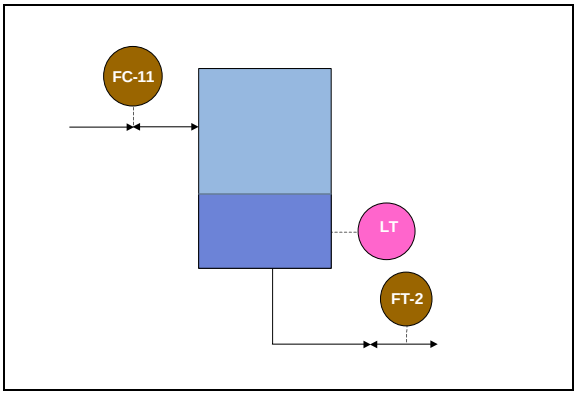

Figure 2.6 - Tank Level Model

This form of a model may not be familiar to many chemical engineers that work with MPC. A simple example can draw parallels to other model representations. In Figure 2.6, the inlet and outlet flows affect the level. The inlet flow, F_1, is manipulated by a flow controller. The outlet flow is not controlled, but is set by a hand valve. The differential equation defining this system is given in equation 2.2

eq. 2.2
$$dL/dt = (F_1 - F_2)/a$$
$$F_2 = b*L$$

where: L – is the level
F_1 – is the flow into the tank
F_2 – is the flow out of the tank
a – is the area of the cross-section of the tank
b – is a constant related to the hand valve position

This is the continuous time differential equation, but we are interested in discrete-time state space models. To get there, first substitute for F_2 in the first equation, then a first order approximation could be used to represent dL/dt as (L(k+1) – L(k))/T, where T is the sampling time. The result is equation 4.3.

eq. 2.3 $$L(k+1) = (1-T*b/a)*L(k) + (T/a)* F_1(k)$$

This is exactly the form of equation 2.1, where the model matrices are A= (1-T*b/a), Bu= (T/a), C = 1 and the state variable is the level, L. This is a scalar example, but this generalizes to the multivariable case. Here, we started with the equations of the process and computed the state space model. In an application to a real process unit, it is not typical to have a fundamental model available. In that case we need to identify the coefficients in the A, B and C matrices from plant step test data.

2.4 Model Identification with State Space Models

Identification with state space models was difficult in the past, but now a technique called sub-space identification makes the task easier [9]. Looking back at equation 2.1, we can see that the equation can be applied repeatedly to predict the values of *x* over a given time period if x(0), u(k), d(k) are available. Subspace identification first constructs the state data from the input output data, then uses regression equations to select the best values for the individual elements of the matrices. If the algorithm tried to directly find A, B and C, there would be some difficulties because there is not a unique solution to the problem. Given one set of A, B and C that work, a transformation could find A', B' and C' that work equally well. The sub space method uses a two step method. The first computes a regression of the states, x, as a function of earlier values of y and u. Once the state sequence is known, ordinary linear regression is used to estimate the A, B and C matrices.

Because the states are not measured directly (unless the C matrix has rows that are all zeros and a single 1), the states are not real model output variables as the level is in the above example. There has always been a question of how many states should be used to describe a process. Sub space identification solves this problem by sweeping over a range of the number of states. Within a commercial implementation of the sub space identification algorithm, the user can select several models and compare their prediction ability, then select the best one for use in the controller.

There is sometimes a concern that state space models built through identification will become very large. As an example, the sub space identification algorithm used in the ABB controller was tested with the Shell Heavy Oil fractionator model. This is a simple model of a fractionator tower with 3 draws and 2 pumparounds as process inputs and 5 temperatures and 2 analysers as process outputs, to make a 7x5 system [11]. Each transfer function has deadtime. The resulting state space model fit the data very well, using only 15 states. That is less than one state per original transfer function, including deadtime. The simulated step responses for this example are shown in Figure 2.7.

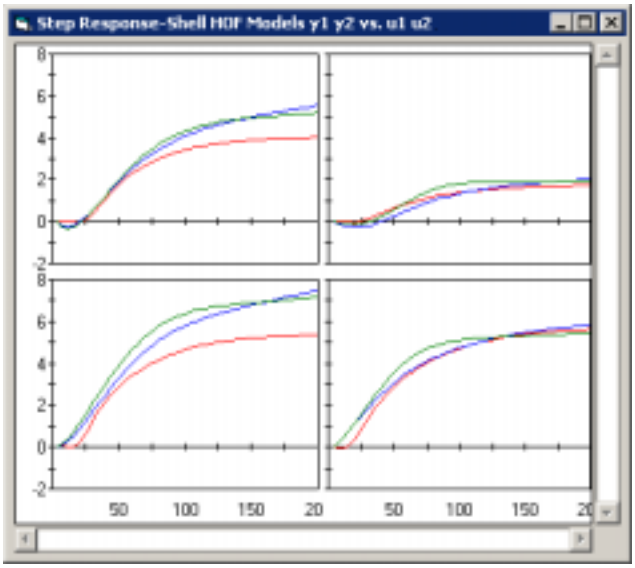

Figure 2.7 – State Space Model Step Responses

What is happening is that the dominant harmonics are being picked up by a small A matrix and are mapped to the process variables through the C matrix. This is not unlike the hidden layer in a neural network.

This leads to a specific advantage over the problems defined in Section 2.2. Because the model is fit with a small number of states, there are not sufficient degrees of freedom to create the noisy type of step responses shown in Figure 2.3. It is not necessary to filter the data heavily or augment the objective function of the identification algorithm with penalty functions.

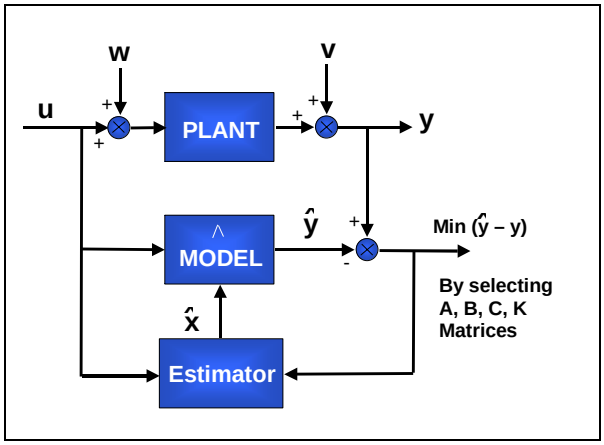

Figure 2.8 - Model Identification with Disturbances

When the disturbances entering the process are not jagged high frequency noise, the step response algorithms can bias the process gain and mis-identify the dynamics. The state space model has an advantage to overcome these problems. The identification can use an estimator to compensate for these disturbances. See Figure 2.8. In this diagram, noise is added to the process as either an input (w) or output (v) disturbance. The estimator predicts the estimated value of x(k+1) from equation 2.1, with one substitution.

eq. 2.4 $$B_u * w(k) = K * (\hat{y}(k) - y(k))$$

where K is the estimator gain matrix (Kalman filter).

The identification algorithm picks values for A, Bu, Bd, C and K to minimize the error between y and ŷ. This error can be made small, by manipulating the values of K, without biasing the A and B matrices.
The result is that state space models are less sensitive to unmeasured disturbances that occur during step testing.

2.5 Large Applications Increase the Problems not Just the Problem Size

There is a tendency for MPC applications to get bigger and cover a larger scope of the process. Part of the justification for this is economic. If the MPC algorithm is trying to maximize feedrate, the complexity of a large controller is offset by the improved economic opportunity. Bigger more complex problems are being shoe-horned into the same technological solution. The multi-objective optimisation techniques giving true ranked constraints help to make tuning easier for large problems, but other areas have not been improved.

With controllers spanning several unit operations, the intermediate storage between units is now part of the control problem. This includes distillation tower vessel bottoms and reflux drums, flash drums between reactors and separation sections, and hold-up in reactor vessels.
The step response type models are poorly formulated for representing levels. Levels are integrating variables. If a flow in or out of the vessel changes, creating an imbalance in the mass balance, the level variable will ramp up or down. A finite length step response cannot represent this behaviour. This is a problem for model identification and for formulation of the prediction equations within the control algorithm.

To address this, integrating variables receive special treatment in MPC algorithms. The special treatment is really a work-around. Internally the derivative of the level is controlled, which is modelled by using the impulse response model of the level as the step response model for the level's derivative. Then a pseudo cascade loop, hidden from the user, resets the target for the derivative to control the integrating variable. This formulation results in aggressive behaviour, because the level controller makes large moves in the level derivative

target. Additionally, this formulation increases the chances of fitting the level noise with the model parameters. The level's derivative is in fact much noisier than the level.

2.6 Working with Integrators using State Space Models

If we look back at Figure 2.6 and replace the hand valve with a second flow controller FC-2, the equation describing the system is simply

eq. 2.5 $$dL/dt = (F_1 - F_2)/a$$

and there is no simplification for F_2. Using a first order approximation for the derivative again, the difference equation becomes

eq. 2.6 $$L(k+1) = L(k) + (T/a)*(F_1(k) - F_2(k))$$

If we subsitute d/dt = s in equation 2.5 to create the LaPlace transform, we would recognize the form of an integrator.

eq. 2.7 $$L(s) = (F_1(s) - F_2(s))/(a*s)$$

When the A matrix has a value of one on the diagonal, the equations are describing an integrator. State space models can represent integrators naturally. Therefore the control algorithm does not need to create any special structures to handle integrators, unlike the problem described in Section 2.5.

2.7 Large Applications Encompass Feedforward Variables

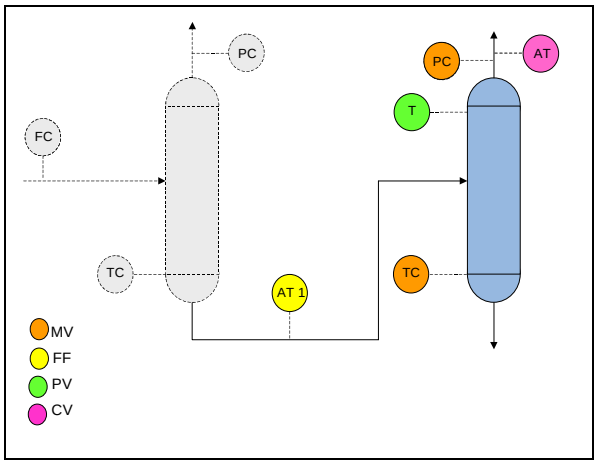

Figure 2.9 - Composition Feedforward

In section 2.5 we discussed integrators as one of the problems with a large controller scope. A second attribute of larger controllers involves the loss of feedforward variables. This can be illustrated by looking at Figure 2.9 and Figure 2.10. In the first diagram, a process unit, possibly a distillation tower, has a feedforward configured from the tower feed analyser. In this case, when the composition of the tower feed changes (AT-1), the controller will make compensating moves with the manipulated variables.

In the second diagram, there are now two unit operations configured in the MPC application. The analyser on the bottom of the first tower is now an MPC CV. If this is configured as a single MPC problem, the MVs on the first tower affect both AT-1 and AT-2. AT-1 is not independent of the MVs, so AT-1 cannot be a feedforward variable for AT-2. AT-1 is a valuable source of disturbance information for AT-2 and now it is lost. It has changed from an independent feedforward variable for AT-1 to an intermediate variable in the model.

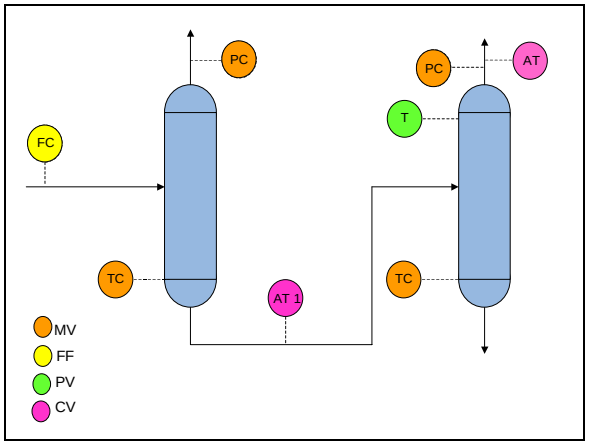

Figure 2.10 - Composition as Controlled Variable

An alternative is to configure two smaller MPC applications, one for each tower. This would work and AT-1 could be a feedforward to AT-2, but any constraints in the second tower cannot be helped by MVs controlling the first tower.

This predicament is due to the SISO (single in single out) nature of step response models used by MPC combined with the inherent assumption of output noise. There is no relationship or mapping between CVs.

2.8 Taking Advantage of Intermediate Variables

Figure 2.10 showed an MPC application that spanned two process units. The intermediate analyzer contains valuable information for modeling the downstream analyser. With a state space model, engineering tools are available to impose structure on the model. Take the process model in Figure 2.11 as an example, where the variable y_2 is an intermediate variable.

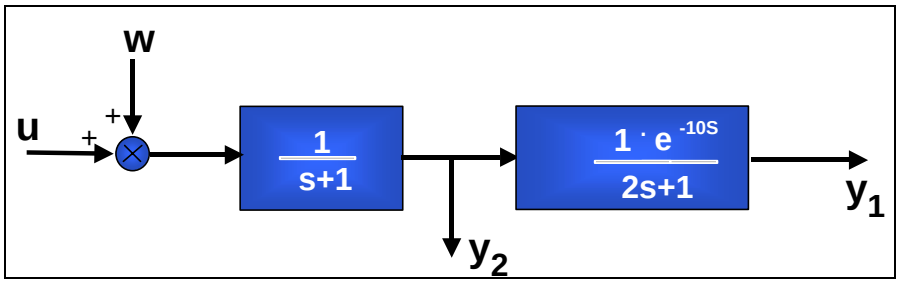

Figure 2.11 – Intermediate variables

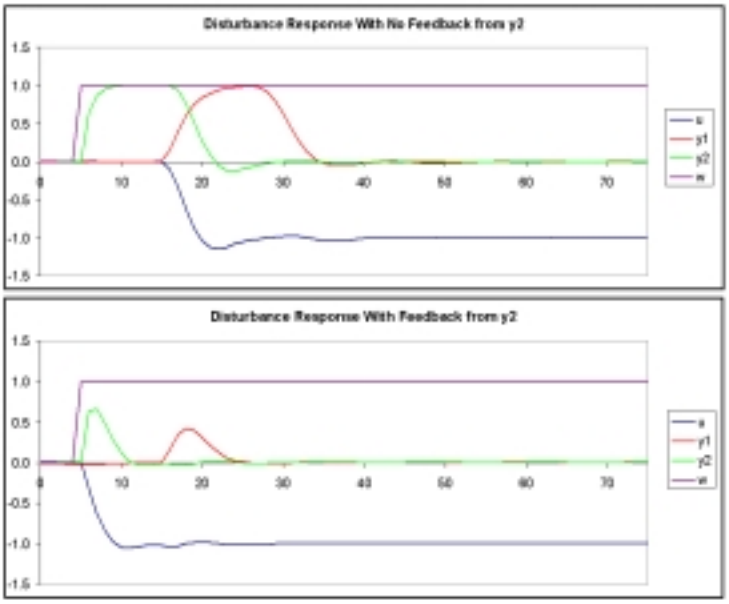

Figure 2.12 – Effect of Intermediate PV

The top half of Figure 2.12 shows the results of closed loop MPC control of y_1 measuring only y_1, while w acts as a step disturbance. When the disturbance input v makes a step change, y_2 will respond before y_1, so this is something to take advantage of. In the second example, y_2 is configured in the controller as a PV or predictive variable. Because y_2 has no deadtime, the estimator immediately sees the disturbance, predicts the effect on y_1 and the controller moves u much quicker, as seen in the lower half of Figure 2.12.

The intermediate variable y_2 is faster than y_1 based on the structure in Figure 2.11, but it is not necessary that intermediate variables act as inputs to downstream CVs. The Kalman filter estimator is capable of inferring the effect of a disturbance on y_1 in the future, if it is measured on y_2 now.

3 Enhancing Prediction and Monitoring Capability by Merging Statistics and Modeling Techniques

A second common use of empirical models is related to the estimation of unknown or unmeasured quantities and to the assessment of equipment or process conditions.

In this category fall applications like software sensors, used to continuously infer vital parameters like product qualities or pollution content, and fault detection and isolation, where the outcome of models are compared with the actual measurement in order to identify changes early or incumbent failures. Additionally, it includes also statistical techniques like Statistical Process Control (SPC) designed to promptly spot production problems and/or variations.

Different from control applications, it is not common to have devoted experiments or step test data collection campaigns for this type of model. Data is usually acquired from process historians or other data repositories. There are a number of impending difficulties, which will be quickly summarized in the following sections.

3.1 Dealing with large datasets.

It is so easy to extract process data that the engineer is left with much more than he really needs and has to remove data that could be detrimental to the model. Let's consider model inputs. Many plant measurements are often available, but for empirical model building "less is better". So the problem is to find the minimum number of inputs able to maximize both model performance and model robustness. What is the number? Which inputs should be selected?

Luckily the statisticians have created proven methodologies to facilitate answering these questions. Probably the most important technique is Principal Component Analysis (PCA). PCA is a very effective methodology, able to separate the signal from the noise in a process, revealing how many (and possibly which) variables are responsible for the former and how many for the latter [12].

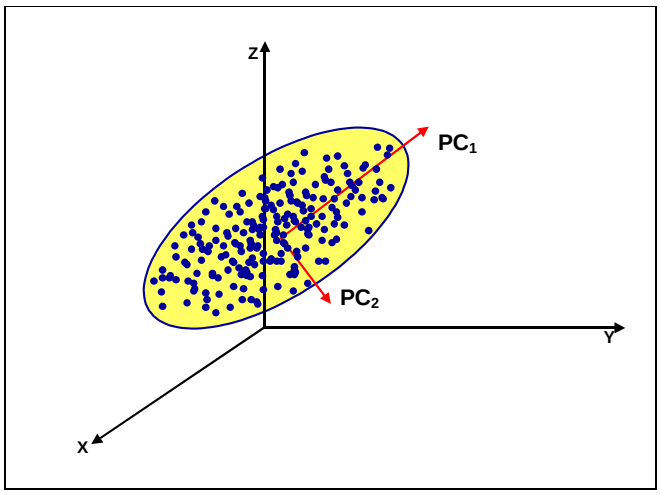

Figure 3.1 – Example of PCA of a 3D dataset

From a mathematical standpoint, it is a projection-based technique able to reduce the dimensionality of a problem through the construction of orthogonal latent variables (named

principal components – PC) that are a weighted linear combination of the original variables. They are computed so that the first PC provides the direction of largest variation in data. The second PC indicates the largest one in a direction which is orthogonal to the first, and so on. (for more details on PCA see [13]). Figure 3.1 provides a simplified view of the first two PCs in a 3d example. Given *n* records, each one made up of *k* variables, the (*n*x*k*) correlation matrix of the standardized variables[1], **X**, can be calculated and decomposed into the *p* eigenvectors (with $p=\min\{n,k\}$). Usually only the first *r* eigenvectors (corresponding to the *r* largest eigenvalues) should be retained for the model, forming the loading matrix **P**, while the remaining (*p*-*r*) eigenvectors will form the residual matrix **E**, assumed to explain the process noise, as described in equation 3.1:

eq. 3.1
$$\mathbf{X} = \mathbf{t}_1\mathbf{p}_1^T + \mathbf{t}_2\mathbf{p}_2^T + ... + \mathbf{t}_r\mathbf{p}_r^T + \mathbf{E}$$

An example showing how a pre-processing utility embedded into the modeling toolkit may help the process engineer to quickly understand how many degrees of freedom exist in the problem is given in § 5. Analysis of the most important PC tells a lot about which variables should be selected as model inputs. Remaining uncertainties in model structure could be resolved by comparing several models built with the most promising candidates. Automatic model building and effective comparison metrics would be a desirable feature for the control engineer.

A second issue is related to dataset partitioning. It is important that the available data is properly divided into modeling and validation sets, each set covering all operational cases. This must be done properly or the model assessment will be biased by data that does not represent the process. Figure 3.2 shows a rather extreme example of how splitting the data set incorrectly could lead to wrong conclusions. A model trained on data collected in period A, is not in the position to provide good predictions when plant is operating in period B. The opposite risk happens when the split is made so that the resulting data sets are not independent. A typical example is represented by a 1-1 splitting, when consecutive records are sent alternatively to the modeling and the validation sets. While this seems like a good way to have records from any operating mode in both the datasets, this will result in a poor validation set, which is too close to the modelling set (unless each record is taken with appropriate time between samples).

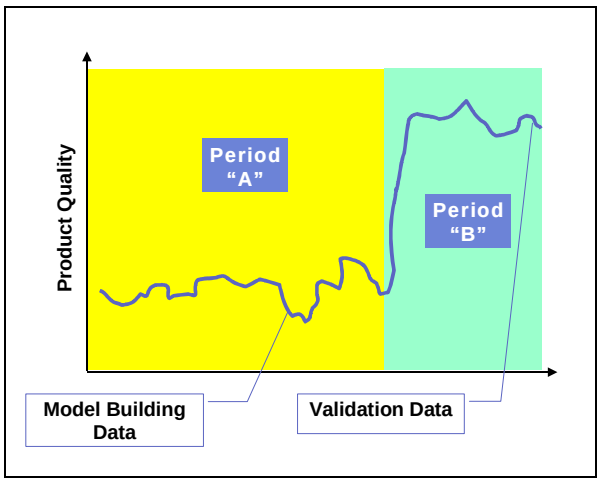

Figure 3.2 – Improper Splitting of a Data Set

In this case, proper help for the engineer would be a statistical tool capable of comparing and assessing the consistency of the different datasets. Once more, the solution is providing simple rules-of-thumb able to spot possible statistical traps and numerical shortcomings.

[1] Y is the standardized variables of y if: $Y = (y-\mu_y)/\sigma_y$

3.2 *The "Garbage-in-Garbage-out" rule.*

A second problem comes, paradoxically, from the effectiveness of the actual number crunching techniques. Modern data mining tools are able to find a solution to the problem "in any case". Unfortunately, very often the solution found is not a really good one for the problem. It is what is known as the "garbage-in-garbage-out" rule. If you feed your software with wrong data, it will provide the wrong answer. Outliers, correlated inputs, auto-correlation, re-arrangement of variables due to process time delays are dangerous occurrences which may spoil the results of the most powerful algorithm.
This moves the emphasis from model development to data analysis and pre-processing. While domain competence and expertise is a must [14], statistical analysis and engineering tools can reduce the burden on the control engineer. A problem is that they require numerical and statistical skills that often exceed the typical process engineer's background.

While a comprehensive treatment of these problems is outside the scope of this paper, their non-triviality is evident in the following short example about outlier identification and removal.
Measurement errors in the context of sampled, measured variables are simply the difference between what is measured and what is the true value of the variable, that is,

Eq. 3.2
$$(y)_k = (x)_k + [(g)_k + (b)_k + (w)_k]$$

where: y is measured; x the true value; e = g + b + w is the error due to the measurement; and k denotes the k-th value of the time series. Equation 3.2 represents an additive error model with:

1. Gross Errors denoted as $(g)_k$
2. Systematic Bias denoted as $(b)_k$
3. Random Errors (noise, normal or uniformly distributed) denoted as $(w)_k$

The trickiest and most dangerous class of error is the Gross Error, which includes the following two subclasses of GEs:

1. Sensor failure related GEs. These result in a sustained GE that is characterized by a non-continuous, dynamic response such as a step function for the measured variable and often a noise-free signal that exhibits steady state behaviour.
2. Sensor fault related GEs (Outliers). These result in short GEs that are characterized by a non-continuous, dynamic response such as an impulse or an impulse-like response. Outliers may be distinguished from noise by the amplitude of the peaks.

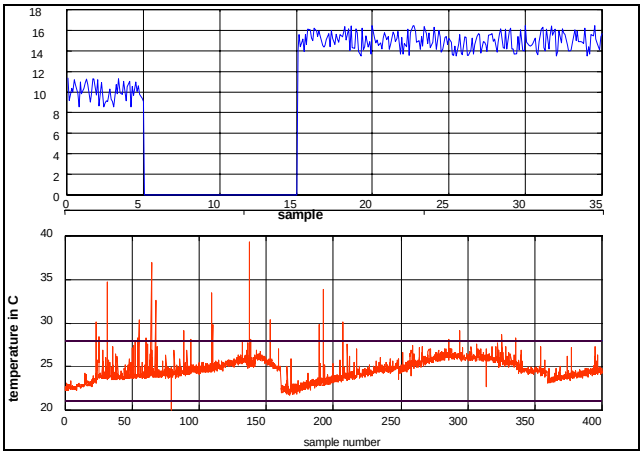

Figure 3.3 – Sensor Failures and Outliers Gross Errors

Figure 3.3 shows the different effects of the two kinds of GEs on plant data: any process engineer is, unfortunately, very familiar with similar charts and knows how this impacts any effort to build reliable models out of such data.

Luckily today, statistical methods are available to supplement the engineer's judgment in order to

obtain good data for model building purposes. Filtering of high frequencies, de-mean and de-trend, interpolation, manual/automatic substitution in case of bad quality, automatic identification of spikes and outliers, rearrangement of variable values to tackle delays, ability to identify correlations /autocorrelation in variables are some of the tasks that a good tool should provide. A must for this type of tool is a full visual approach. Regardless of how powerful the tool is from a purely mathematical point of view, the engineer needs to be able to visually verify and validate process data and preview any change/filtering to it.

3.3 When should a model be considered a good model?

Deciding when a model is good is an additional challenge. In theory a perfect model should be able to fully reproduce the behaviour of the modelled system. But certainly noise should be out of the picture, because it adds randomness that has to be avoided and the model should not fit it. So how do you tell when the model is improving and when it is just chasing the noise? What is the minimum error squared between modelled and measured output that we would like to see, given that we have just said that the 'zero' is not our target? This is something similar to what described in §2.2. There is a clear need to accept a simpler model with some prediction errors, rather than struggling to obtain more complex ones, which "look better" on training datasets, only because the noise was modelled. Usually avoiding overfitting requires expertise and a lot of iteration on different datasets in order to properly tune the model. A common technique is to perform simultaneous predictions both on a training and on an unrelated test set, checking and comparing the results while gradually increasing model complexity. Figure 3.4 shows a chart of the average squared error on both modeling and test set, against model complexity. When the error on the test set starts to increase, it is a good signal that the algorithm is fitting the noise. The problem is that this procedure requires a lot of iterations and book-keeping in order to properly identify the optimal degree of complexity.

Building data-driven models out of good, meaningful data is pretty quick and easy. This allows the engineer to try several possible structures and approaches with the final result of spending time just "fine tuning" a multiplicity of models, looking for an optimum whose identification may result controversial (different metrics – R^2, mean square error, maximum square error, etc. - may provide different hierarchies). This is made worse by the fact that developing a model is always a three-step approach: build your model on a fraction of your data, assess it against a second data-set, then rebuild your model on the whole dataset.

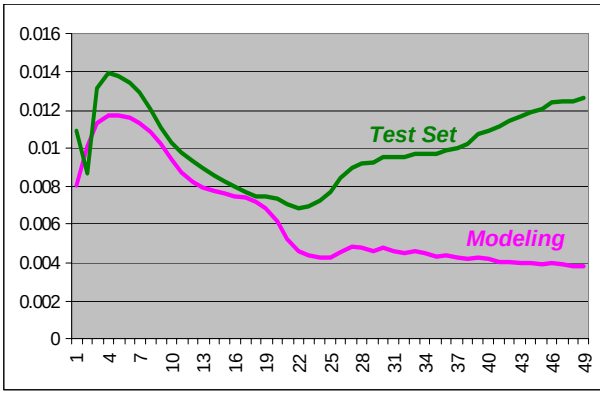

Figure 3.4 – Model Overfitting

Facilities able to assist the user with automatic model building and proper comparison tools could greatly reduce development time and drive the inexperienced user safely towards comfortable results. A highly desirable feature would provide automatic model building using a group of possible inputs, permutation rules and scripting. Smart algorithms can emulate the try-and-test procedure that engineers do when they try using different input combinations. This way, the engineer may use his time for higher level tasks and let the

software generate and train multiple models based on his instructions; then, later he can compare models based on their prediction and select the model with the best performance (also taking into account his process knowledge). This process requires good tools for model comparison, both from a numerical and visual point of view.

3.4 On-line implementation.

Putting a model on-line is not just a matter of reading and writing back and forth to the basic automation system, but to seamlessly integrate it, so that the operator may fully benefit from it. Issues that need to be carefully revised and taken into account are on-line data filtering both of inputs and outputs. Inputs must be filtered to remove noise. Outputs may use a moving average to increase reliability and decouple downstream applications from calculation noise. Other considerations are bad quality management, writing back to the DCS with related issues like proper displays and alarm generation.

But maybe the trickier issue in configuring the model is how to take advantage of incorporate periodically available data. Often empirical models are used to estimate variables available only sporadically from plant laboratories or analysers. Using this information to correct and update the prediction is an obvious improvement, but it is necessary to identify and avoid use of errant lab analysis, preventing the calculation of an incorrect model bias. Implementing this identification and filtering logic may is so complex that many works available in the literature have simply given up this opportunity to improve the accuracy of their application.

To improve the model's estimate using laboratory data or field analysers, the model output value could be corrected by a bias. The bias value should be determined by the difference between the estimated and the measured value. Since analysis results often involve errors that have a Gaussian distribution, the "tails" of this distribution must be avoided or the bias calculation will be faulty. Programming logic must implemented in order to detect and remove statistically unreliable data. Statistical Process Control strategies, which control average and variances of absolute values and differences, are very effective in identifying acceptable input data. The whole system may also require operator acknowledgements and interactions that should be implemented on the standard operator console to make it easier for him/her to accept the technology. Figure 3.5 presents a typical bias update scheme.

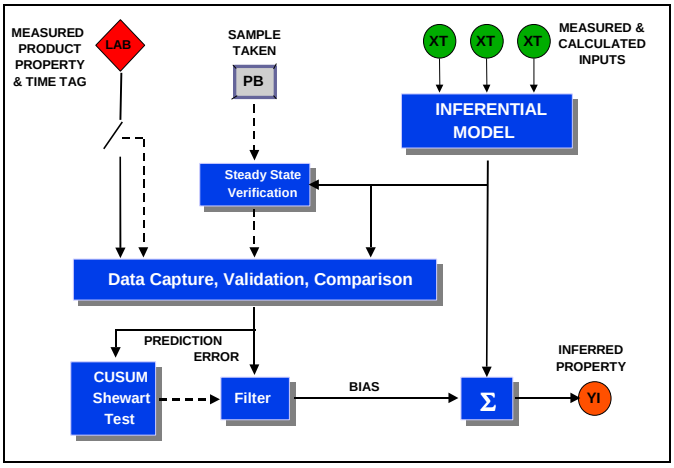

Fig. 3.5 – Typical Bias Update Scheme

3.5 Protecting the Value of Your Modeling Investment

Models are an important asset for plant operation. Sometimes models have been built and validated over the lifetime of the process, representing a significant engineering effort worth protecting. Legacy models that were once on-line, may not have received the proper

maintenance to keep them on-line. This can easily result from upgrades to the DCS or replacement of an old process computer.

Many process engineering departments have some very useful models that have never run on-line, but reside in some spreadsheet or stand-alone VB application. In some cases, many years of operating experience are distilled in these models. In other cases, they are brand new models developed by an enthusiastic young engineer using the latest mathematical modelling software packages. But the tools that were used to build the models may have no connectivity capability. So whether the model building procedure was long and arduous or quick and pioneering, the result is the same: relegation to an off-line status.

The typical process model users are engineers, whose prime interest is not process control or information technology, but process operations and product quality. Present software technology allows removal of most of the barriers among components, languages and equipment. Existing models and custom software can therefore be converted with minimal effort into components (like DLLs and COM objects) that can be easily integrated online using an open platform. So a good modeling package not only allows the engineer to easily build new models, but it will also allow him to host his existing orphaned applications without a great deal of effort and no custom coding. The modeling environment will provide real-time data access for inputs, input validation, calculation scheduling and graphical display of outputs for the operator for the existing model.

4 The Optimize^{IT} APC Control Suite: an Innovative Approach to APC

4.1 Why Another Commercial Product Suite?

The authors of this paper have significant experience in APC project work. Their current company is a large process control company that worked with many of the APC software vendors through license agreements. This experience has allowed first hand experience with these packages, including knowledge of their shortcomings.

The process control community has not taken advantage of the advances in modeling technology. As described in sections 2 and 3, there are approaches to building better models for on-line applications and to avoid these problems. How could integrating these advances in modelling technology affect the control engineer and the plan operator?

1. The improvement in modelling technologies can lead to improvements in process control performance. This should lead to better economic performance on existing applications and should create new opportunities where the ROI was not adequate with the prior technology. Furthermore, the new technologies should make problems feasible that were previously considered technically too difficult to try. So wider application and better performance are the first effect.
2. With software that is easier to use, with prevalent availability of live and historical process data and with computer power no longer an issue for these types of on-line applications, on-line modelling should proliferate for simpler applications. For instance, the control engineer should be able to

build an inferential model whenever he wants or needs one, with a simple toolkit that his plant owns.

With this in mind, the authors have been involved in designing and developing new APC products within the ABB Industrial^{IT} framework, as their company has determined that the APC business is best approached by having the best software solutions. The new products include the technology enhancements previously discussed, as well as including many features to improve ease of use. This comprehensive new suite provides components that work together for all APC project activities.

The suite includes:

- MPC
- Neural networks
- PCA, MLR and PLS Regression
- SPC and MvSPC
- Control Loop Tuning
- Control Loop Auditing

Providing a comprehensive suite allows the application engineer to build control solutions without worrying about writing programs or connecting applications on multiple platforms. Some features of the suite are:

- Support OPC as the I/O standard and ODBC for database import
- Standard project navigation window in all on-line and off-line tools
- Common data editing and pre-processing tools
- Enhanced trend element, common across all programs

These features should decrease the time the engineer spends learning about the package, and increase the amount of time the engineer spends using the package.

The packages are also accommodating. There are a great number of proprietary models running in plants running on old platforms and written in old languages like Fortran. In these cases new modelling technology is not needed, but a user-friendly container to execute an old model is needed. A standard wrapper is provided to integrate these applications into a modern architecture.

Not diminishing the usefulness of Fortran, the driving force in the suite is new technology coming from R&D. ABB has a commitment to R&D through the Corporate Research Centre and much of the new technology inside the APC suite was first developed within the R&D group. An example of the innovations built into the suite is the outlier removal strategy in the data processing tool. The strategy applies wavelet analysis to pinpoint data that contaminates information content of the data set [16]. This type of research is typically only funded in large forward thinking organizations, which may explain why APC technology has not moved along faster.

In the following paragraphs, the two main products of the APC suite are described.

4.2 *Optimize^{IT} Predict & Control (P&C)*

Optimize^{IT} Predict & Control is a multivariable, model predictive control software package. The P&C software suite includes on-line components for control and operator interaction and off-line components for controller configuration, dynamic model identification, tuning and analysis. The controller operates through existing instrumentation and control equipment so there is no major investment or interruption of production required.

P&C includes four general types of variables:

- Process Inputs
  - (1) Manipulated variables (MVs) which are adjusted by the controller.
  - (2) Feedforward disturbance variables (FFs) which represent measurable disturbances used for feedforward predictions.
- Process Outputs
  - (3) Controlled variables (CVs) which include controlled variables with setpoint targets or constraint variables with min/max limits, and measurable variables that are used to improve the estimates of the controlled process variables.
  - (4) Prediction variables (PVs) that provide additional feedback information that improves disturbance estimation, resulting in quicker, more accurate corrective action

The P&C control algorithm is based on a state space model representation of the process. The state space model is used to predict the effects over time of independent process inputs (MVs and FFs) on dependent process output variables (CVs and PVs). The model allows the controller to account for process dynamics (e.g. deadtimes and lags) between changes in the independent variables and expected changes in the dependent variables, see Figure 4.1.

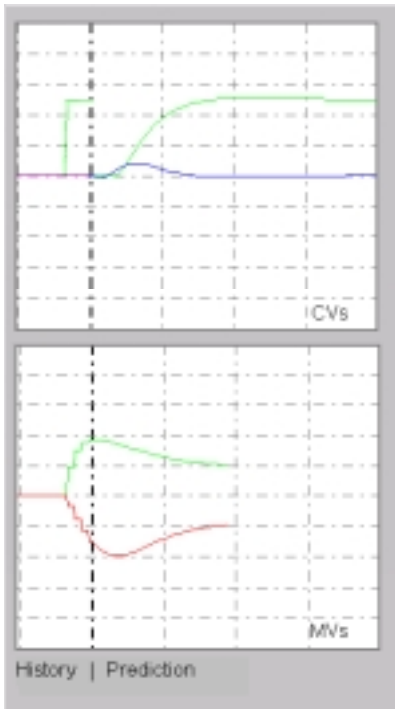

Figure 4.1 - Controller Predictions

P&C provides flexibility in treating a combination of setpoint control and constraint control objectives for multiple variables. Each controlled process variable may include a setpoint target and/or constraint limits (defined as absolute min/max values or min/max deviations from setpoint). Priorities are also assigned to the constraints, so the controller can calculate appropriate moves to prevent or minimize violations of the highest priority constraints, before lower priorities are considered. The controller can accommodate controlled variables that are based on intermittently sampled data, such as analyser or laboratory data.

The P&C algorithm also offers a 3-degrees of freedom control design in which setpoint, feedforward, and feedback control responses can each be tuned independently in the time domain. This provides significant application design flexibility and leads to very robust process handling from more accurate controller commands.

The controller includes a static optimisation stage to drive MVs towards desired targets when the control problem has extra degrees of freedom available. The desired targets for the MVs

may be set by various sources (e.g. plant supervisors, operators, engineers, external software such as RTO). Alternately, an economic objective function can be configured into P&C in terms of the MVs and CVs. The controller then drives to the economic optimum instead of MV targets.

Internal Model and Estimator

P&C includes a state estimation algorithm to dynamically estimate the state variables at each time step by applying the process input measurements (MVs and FFs) to the model and then defining corrections to the states based on the CV prediction errors (the difference between measured value and the value predicted by the model). New state estimates are generated and used to improve the current and future predictions for all the CVs (reducing CV prediction error). Therefore, the state space model allows for earlier detection and faster controller response to unmeasured disturbances, when compared to other competing MPC technologies.

The state space technology also offers the ability to incorporate additional process measurements (PVs) as feedback to improve the estimates of the disturbances. This is a unique feature offered by P&C because the state space methodology provides an integrated model representation. The state estimator utilizes all the current measurements (CVs and PVs) to detect unmeasured disturbances and predict their future effects on the CVs. Notice that the measurements are considered as "feedback" to the state estimator, but the new state estimates are also used in the controller calculations for the future, so the PVs provide a feedforward effect in the controller.

Engineering Tool

The P&C off-line engineering tools are included as a separate software component to support controller configuration, modeling, tuning, analysis, and simulation. The modelling tool includes several methods for developing state space models. Models may be defined from plant test data using a two step procedure of subspace identification followed by prediction error identification. There are many tools available to the user for reviewing the results of the modelling package and for selecting between multiple models that the user may generate. Models may also be defined by the user in terms of transfer function representations or may be imported.

The control tuning and analysis package provides the user with many plots to evaluate controller performance, including closed loop response to setpoint changes and disturbance inputs. Robustness under modelling error is treated. Both time domain and frequency domain tools are available.

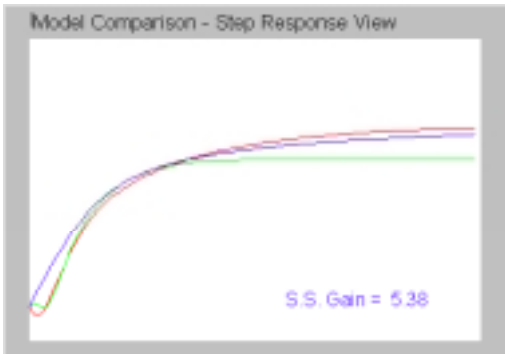

Figure 4.2 Modeling Tools

The online controller comes with a built-in real-time simulator. The simulator may use the model that the controller is using, or the user may specify a different model to test the robustness of his tuning parameters. The simulator may run faster than real-time for engineering studies, or as slow as real-time for operator training sessions.

Operator Interface

The controller may use OPC to link all operating parameters to a DCS console, or the operator can use the P&C Operator Interface client. The client runs on Windows 2000 systems and may be run remotely from the control server on a PC in the control room or on the same computer as the operator's DCS console.

The operator interface provides tabular displays for the operator to control all daily aspects of the application. The displays are configurable. Detailed displays are available for individual variables. User configurable trending is also available, with plotting of future CV predictions and planned MV movement. There are engineering displays for online tuning.

4.3 Optimize^{IT} Inferential Modeling Platform

Optimize^{IT} Inferential Modelling Platform (IMP) is an innovative software package for the development and deployment of data-driven advanced applications. It is based on two separate environments:

- IMP Model Builder for application design and development
- IMP On-line for on-line project deployment and monitoring

IMP features latest generation data analysis and modelling technologies developed in house or selected from technology leaders around the world. The user is able to exploit a rich collection of highly sophisticated tools for data analysis and pre-processing available at his fingertips. Part of them, like basic statistical insight and Principal Component Analysis facilities, comes from a special agreement with InControl Technologies Inc. (Houston, TX).

All the different tools are embedded in an intuitive working environment based on the latest HMI concepts, which remove any hurdles for the inexperienced user.
IMP is designed to be an open modelling environment where new tools may be easily hosted and put to work. However IMP features some latest generation toolkits, which allow building models through several technologies including:

- Neural Networks
- Multiple Linear Regressions
- Calculation Scripts

The Neural Network engine is a customized version of Ward Systems Group's Neuroshell® Predictor, one of the most referenced Neural Network packages available on the market [17]. It contains one of the most sophisticated prediction algorithms available today, yet it is designed to be extremely effective with minimum intervention by the user.
It features two different prediction algorithms:

- a highly sophisticated neural network;
- a statistical estimator driven by a genetic algorithm;

The first modelling technique is a feedforward net, proprietary to Ward Systems Group, which is not based on the classic back propagation algorithm. It dynamically and automatically grows hidden neurons, trains very fast and has excellent generalization capabilities. Hidden neurons are connected to previously added hidden neurons, eliminating the need to pre-specify a distinct number of hidden layers.

One of the most time consuming tasks in developing neural models is the iterative training and testing procedure, needed to identify the model with the best performance, which doesn't

over-fit the data (see [18] and [19] for details on Neural Network training and testing issues). One of the salient advantages of the IMP/Neuroshell neural method is that it can actually be tuned after it is trained in order to provide more or less generalization. This allows the user to decouple the training activity from the testing activity, offering a big advantage both in development time and in the accuracy and reproducibility of the method.

The genetic training method combines a genetic algorithm with the well-known General Regression Neural Net. GRNN is a statistical estimator, originally developed in the statistics literature and known as the Parzen kernel regression [20]. It was subsequently introduced to the neural network community by Donald Specht [21]. It trains everything in an out-of-sample mode, essentially doing a "one-hold-out" technique, also called "jack knife" or "cross validation". This method is therefore extremely effective when you do not have many patterns for training. The genetic training method takes a little longer to execute, but it also reveals the relative importance of each of your inputs.

Common to the two training methods is the capability of removing two of the most frequently heard complaints about prediction systems, i.e. that they are too hard to use, and that they are too slow.

IMP embeds Neuroshell's features in a process control oriented environment, unleashing all the power of neural network modelling, without most of the related drawbacks and nuisances. Highly automated, yet very simple procedures allow the user to simultaneously build several models and then to compare the results.

Let's assume that we have *n* potential model inputs. IMP classifies them as *m* "always used", *p* that "could be used" and *r* that should be never used. Based on this classification the following algorithms are available for automatic development of models:

1. Full permutation of inputs. IMP generates and trains all the models automatically taking into account all the possible permutations. This will produce (2^p) models with a number of inputs ranging from *m* to *m*+*p*
2. Smart identification of useful inputs. IMP starts training the biggest model with all the (*m*+*p*) possible inputs. Based on a computed "input importance factor" and some pre-defined rules and thresholds, it reduces the number of inputs by removing the least significant ones step by step, until it reaches a stopping condition.
3. Iterative improvement. IMP starts training the biggest model with all the (*m*+*p*) possible inputs. It then starts generating all cases using n-1 inputs and evaluates the results in terms of R^2 and average error (both test and train set). Based on performances of the generated n-1 nets, IMP identifies the input that has the least

Fig. 4.3 – Input Selection Facility

impact on the model This input is removed and the procedure is restarted. Rules on thresholds and minimum number of inputs are set to stop the procedure.

4. Guided randomised. IMP starts training the biggest model with all the (m+p) possible inputs. It then removes a randomly chosen input and builds a new model. Model results (R^2 and avg err) are compared to the original model,. If the model is performs better, it is considered as the reference model and the procedure is executed again. Otherwise the new model is deleted, another input is chosen and the procedure starts again.
5. Script-based training. IMP generates and trains models based on a scripting language. Both the inputs and the model details are defined by the engineer through the scripting language and the model generation and training is automatically performed in a unique session

This way, most of the model building activity is completely automatic, even to the point of executing overnight. The engineer needs to check the results and accept the most convenient and best performing models, using the many available comparison facilities.

IMP would be not a true data-driven applications environment if limited only to model building. In fact IMP includes powerful tools for process and quality monitoring, allowing the user to quickly implement SPC control charts and even MvSPC [22]. This is particularly efficient in monitoring complex processes with just a single number, the Hotelling T^2 statistic (refer to [23] and [24] for more details on Hotelling T^2 background). IMP exploits well-referenced statistical calculation routines from InControl, seamlessly embedded into its software architecture.

IMP On-line is designed to quickly and efficiently implement applications involving process models. The engineer only needs to physically connect his PC to the network, browse the OPC Servers available and select the tags he wants to read or write back to the DCS. With no need to write a single line of code, he may specify the preferred options concerning a large number of possible configuration details, including bad quality management, tag limits, engineering units/conversions and tag filtering.

Seamless integration of bias update strategies was given particular attention. Any online implementation of inferential models is usually coupled with a periodic recalibration strategy. This strategy computes the difference between the prediction and available physical measurements (like lab analysis) and treats it statistically to determine the inferential model bias. The bias is then added to the model output, to improve its accuracy and avoid any model drift in case of failure in input sensors.

IMP features built-in routines to allow straightforward implementation of biasing. First of all, it allows connection to external repository of lab data (LIMS) through use of ODBC for automatic collection of lab analysis. It then allows implementation of various bias calculation strategies. Different equations and different timing for bias computation can be set up for any configured prediction; different filtering and data validation strategy can be selected and customized to fit the specific client needs. Straightforward integration at DCS level is possible through use of the built-in OPC connection.

5 First applicative examples

5.1 A Refinery Application

The MPC algorithm discussed above was recently implemented to a refinery application.

The process unit selected for the application was a crude unit. The main objective of the application was quality control of the crude unit product streams. The product quality control layer employed the inferential modeling techniques discussed in this paper, specifically the neural network models. Controlling the product draw temperatures enforced the inferential quality objectives.

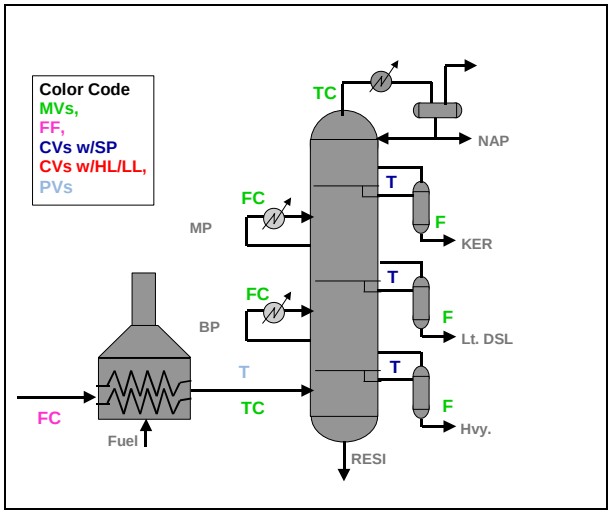

Figure 5.1 - Crude Unit MPC Strategy

There were three temperatures controlled to setpoint, along with additional temperature constraint variables. These variables were controlled using seven manipulated variables, as illustrated in Figure 5.1. The control configuration is summarized in Table 8.1

Controlled Variables	**Manipulated Variables**	**FeedForward**
1. Kerosene Draw Temperature	1. Ovhd Temperature SP	1. Feed Rate
2. Lt. Diesel Temperature	2. Kerosene Draw Rate SP	
3. Hvy. Diesel Temperature	3. Lt. Diesel Draw Rate SP	
	4. Hvy. Diesel Draw Rate SP	
Constraint Variables		
4. Ovhd TC Max Valve Position	5. Middle Pumparound Flow SP	
5–7.Stripper LC Max Valve Positions	6. Bottom Pumparound Flow SP	
8. Heater Max Tube Wall Temp	7 Heater Outlet Temperature SP	
9. Heater Max Tube Wall Temp		

Table 8.1 – Controller Configuration

If there were no constraints active, the three draw temperatures used three of the seven degrees of freedom. The overhead temperature will have an MV target that is a function of the naphtha quality control. That leaves 3 extra degrees of freedom that are used to move the process to a more desirable economic operating point. The pump arounds will maximize heat recovery unless other constraints are encountered that affect quality control. The operator sets the heater MV target.

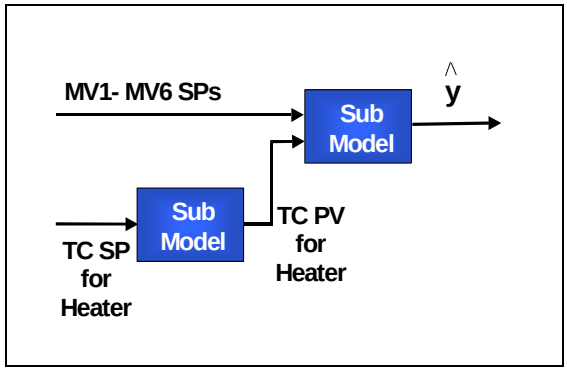

Figure 5.2 Model with Intermediate PV

This is a fairly typical crude unit configuration, but the flexibility of the state space controller was able to provide superior performance to overcome an operational problem. The heater fuel BTU content was

not very stable, causing consistent 3°C to 4°C peak to peak temperature swings in the heater outlet temperature. The period of oscillation was on the order of 30 to 40 minutes and the base controls could not be adjusted to eliminate the disturbance.

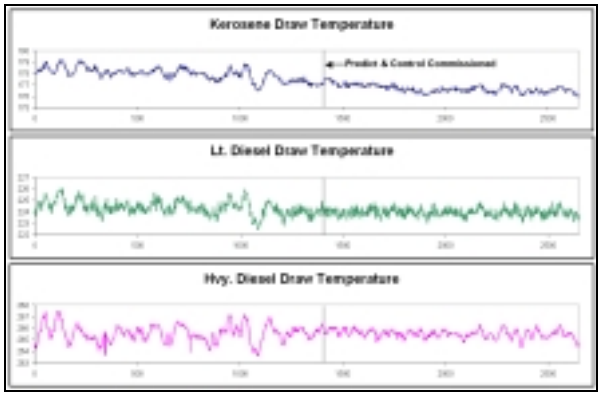

Figure 5.3 – Controller Performance

To improve the temperature control of the draws, the heater outlet temperature was added to the model as a PV. This made the controller structure very good at predicting a process input disturbance. To do this, a special model was built using a model connection tool, resulting in the structure shown in Figure 5.2.
The resulting temperature control is seen in Figure 5.3, showing very good improvement over the pre-existing control.

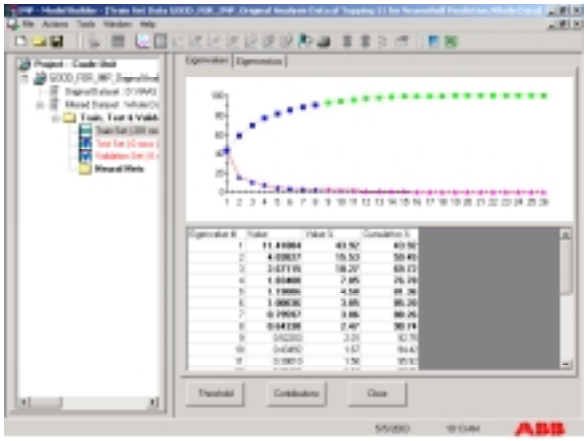

Figure 5.4 – PCA Helps in Identifying Degrees of Freedom

On the same unit, four different inferential models have been built to provide real-time estimation of the ASTM 90% point for the following four products:

- Light Diesel
- Heavy Diesel
- Kero
- Heavy Naphtha

All the models were built using neural network technology. This application may represent a good example to illustrate how merging a statistical data analysis tool and modeling capability in a single environment may greatly simplify model development. Let's consider the Light Diesel (LD) case. A control engineer can easily identify at least 25 different process variables potentially affecting LD production quality. Which one should be chosen? As explained in § 3.1, PCA is an important ally to the engineer. Figure 5.4 shows the results of a PCA on process data, providing evidence that 8 'rearranged' inputs are enough to justify more than 90% of dataset variance. Using this information and the related eigenvector components, it is straightforward to identify the most significant variables for the model. Figure 5.5 graphically illustrates the Neural Network prediction results using this set of inputs.

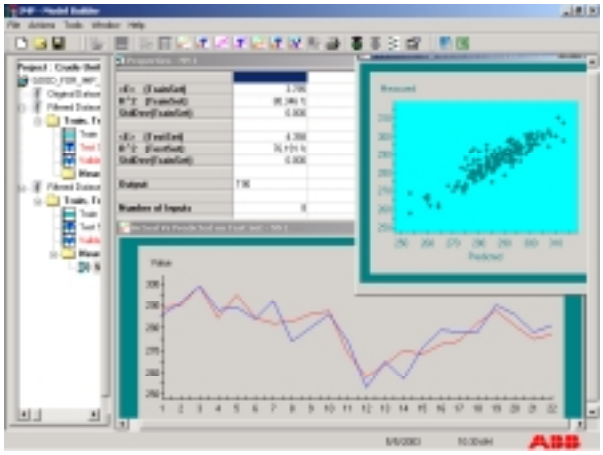

Figure 5.5 – Lower Order Model

5.2 *An Oil&Gas Application*

A second interesting application has been completed on an offshore platform located in Northern Europe, which processes the crude oil coming from several local sub-sea wells and from some submarine oilfields located a few Km from the platform.
The crude oil treatment involves the separation of the feed containing oil, gas and water to obtain oil and gas to be exported. The process can be divided in 3 main parts:

- Crude oil treatment
- Gas treatment and compression
- Condensate treatment

The treated oil, blended with the condensate product coming from the condensate treatment, is stored in cells and offloaded into tankers for the export; the gas is compressed and exported.

The APC application is related to the implementation of inferential sensors to provide real-time estimation of the quality of the exported products. The two predicted quality parameters are the C_4^- content and the RVP (Reid Vapor Pressure) in the crude oil going to the storage cells.

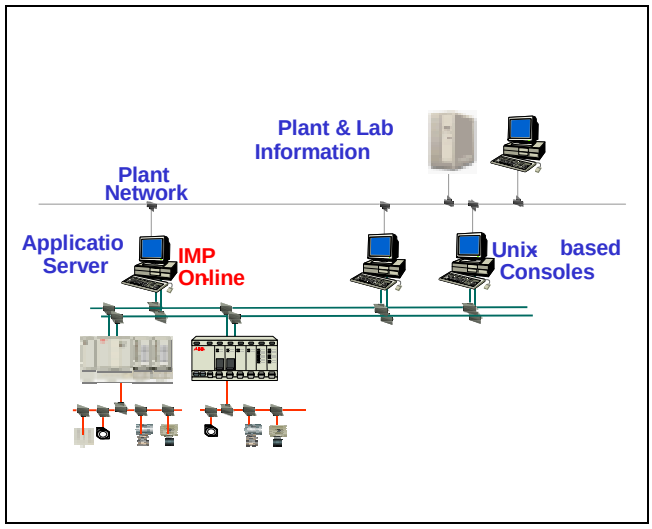

Figure 5.6 – Crude Processing System Architecture

The project was organized in the following steps:

- preliminary process study and target definition
- data collection
- data treatment and elaboration and Neural Network modeling
- off-line validation
- on-line implementation and commissioning

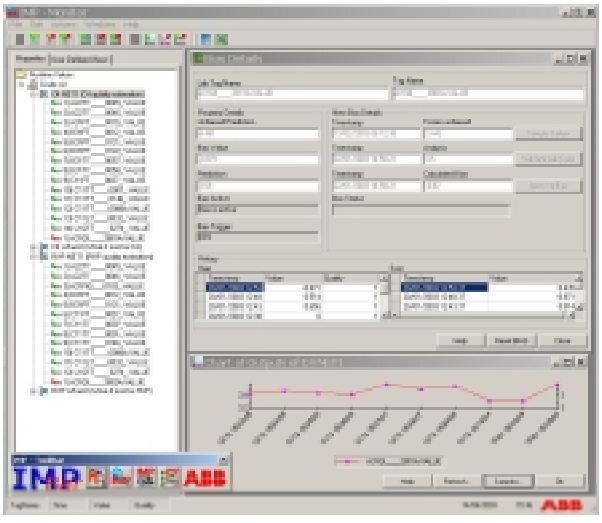

Figure 5.7 – Crude Processing Results

Process data, collected from the plant historian and the LIMS, were re-organized, filtered and pre-processed in order to obtain "clean" and useful training, test and validation datasets, employed for model building and assessment. Finally the best models were shipped back to the plant for final commissioning, realized through the powerful IMP On-line facilities. The target architecture is shown in Figure 5.6 The PC hosting the Neural Network models is connected to the ABB AC 450 DCS through OPC technology; exchange between the Application Server and OPC is performed through use of TCP/IP

functionalities. The client components run on the Application Server and are accessed from PCs using the PCAnywhere application.
The Operators monitor model results through dedicated displays on their Operating Stations, while process engineers manage the application by means of the IMP Monitor software located on the hosting PC. Figure 5.7 shows the typical screen used to check and manage the bias update mechanism.

6 Conclusion and future developments

The paper has highlighted some existing problems, which have prevented wider penetration of model-based techniques into industrial process automation. Analysis of new modeling techniques couched in a modern software environment and supported by the latest statistical methods has shown that these problems no longer stand in the way of model deployment. Software products implementing these techniques were discussed and actual implementation examples reviewed. New modeling solutions lie at the process engineer's fingertips.

As the nightcap, we offer one more example illustrating that on-line empirical process models will proliferate if they are easy to use. Figure 6.1 shows the results of a neural network prediction for solvent emissions in a polymer stripping operation. With new and better tools, there are many possibilities to use multivariable control and process modelling in places that are not traditionally considered.

The ambitious goal is to extend modeling beyond the traditional APC & Optimization domain. This will result in a number of innovative and sometimes unusual applications. Fault detection, process and equipment monitoring, quality control and prediction, predictive maintenance, sensor and analyzer validation, energy optimization are possible areas of interest for advanced modeling-intensive applications. As a final example about smart, smaller scale projects, it's possible to mention an application presently under development at a polymer plant in Europe.
The customer has a problem with the final stage of the process where the finite, extruded product is steam-stripped to remove hexane. Mobile analyzers are used in "monitoring campaigns" to assess the amount of pollution vented into atmosphere with the steam. Obviously this is far from optimum because the analyzers are connected to the plant no more than 20% of the operating time. However using the data stored during these campaigns it has been straightforward to identify a model, which could be easily put on-line for real-time continuous emission monitoring purposes. Figure 6.1 shows the excellent accuracy the Neural Network model is able to provide. Environmental applications are actually one of the most exciting targets for modeling techniques, their use being recommended also by important environmental authorities [25].

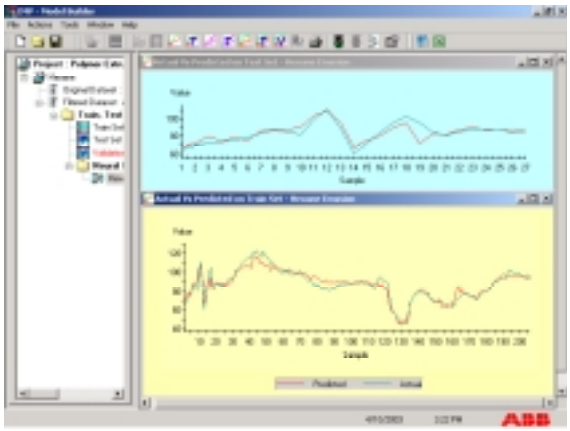

Figure 6.1 – Predictive Emission Monitoring with IMP

Acknowledgement.

The authors would like to thank Mrs. Emanuela Pavan and Mr. Ken Praprost from ABB and Mr. Steve Ward from Ward Systems Group for their highly valuable help towards the writing of this paper.

References

[1] Samad T and Weyrauch J: "*Automation, control and complexity: an integrated approach*", John Wiley and Sons, 2000
[2] Bonavita N, Martini R: "Overcoming the Hurdles in Chemical Process Automation, part I*", Chemical Processing*, Nov. 2002, pp. 31-35
[3] Denn M., "*Process Modeling*", Pitman Publishing, MA, 1986
[4] Marlin, T.E., "*Process Control: Designing Processes and Control Systems for Dynamic Performance*", McGraw-Hill International Editions, 1995
[5] Bonavita N, Matsko T: "Neural Network Technology Applied to Refinery Inferential Analyzer Problems", *Hydrocarbon Engineering*, December 1999, pp. 33 – 38
[6] Garcia C, Prett D, Morari M: "Model Predictive Control: Theory and Practice a Survey", *Automatica*, Vol. 25, No. 3. Pp. 335-348, 1989
[7] Qin SJ, Badgwell TA: "A Survey of Industrial Model Predictive Control Technology", "*Control Engineering Practice*", 11, 2003 pp. 733-764
[8] Idcom-M Multivariable Predictive Control Version 3.1 Control Reference Manual, Setpoint Inc.
[9] Van Overschee P., DeMoor B. "Subspace Identification for Linear Systems: Theory-Implementation-Applications", Kluwer Academic Publishers 1996
[10] Lundh Michael, Molander Mats, "State Space Models in Model Predictive Control", ABB White Paper
[11] Froisy JB, Matsko T. "Idcom-M Application to the Shell Fundamental Control Problem", *AIChE Annual Meeting*, November 1990
[12] Martin E., Morris J., Lane S.: "Monitoring Process Manufacturing Performance", *IEEE Control Systems Magazine*, October 2002, pp. 26 - 39
[13] Jackson J.E. :"*A Users Guide to Principal Component Analysis*", New York, John Wiley and Sons, 1991
[14] Beaverstock, M.C., "It Takes Knowledge to Apply Neural Networks for Control", pp. 335-343, *Proc of ISA*, 1993
[15] Bonavita N and Tomasi R: "Improvements in process control through model-based techniques: a control system vendor's perspective", *Proc. of "1998 IEEE International Conference on Control Applications"* Trieste 1-4 September 1998
[16] Frick A., "Gross Error Detection Using Univariate Signal-Based Approaches", *ABB Technical Report DECRC/A5-TR006*, February 2001
[17] Ward Systems website www.wardsystems.com
[18] Freeman, J: "Neural network development and deployment rules –of- thumb", *Hydrocarbon Processing*, October 1999, pp. 101 – 107
[19] Qin, J: "Neural Networks for Intelligent Sensors and control – Practical Issues and Some Solutions", In "*Neural Networks for Control*", D. Elliott, Ed. Academic Press, 1996 pp. 215 – 236
[20] E. Parzen, "On Estimation of a Probability Density Function and Mode", *Annals of Mathematical Statistics,* Vol. 33 pp.1065-1076, 1962.
[21] D.F. Spetch, "A General Regression Neural Network", *IEEE Trans. On Neural Networks*, Vol. 2, No. 6, Nov 1991, pp. 568-576
[22] Piovoso, M. J., Hoo, K. A., et al. "Multivariate Statistics for Process Control", *IEEE Control Systems Magazine*, October 2002, pp. 8 - 63
[23] Mason, R., Young, J., "*Multivariate Statistical Process Control with Industrial Applications*" ASA-SIAM, 2002
[24] Young, J., Alloway, T, Schmotzer, R., "Introduction to Multivariate Statistical Process Control and its Application in the Process Industry" , Proc. of "*NPRA Computer Conference*", 13-15 November. Chicago, Illinois
[25] "*Continuous Emission Monitoring Systems for Non-criteria Pollutants*". EPA Handbook, August 1997