USING MACHINE LEARNING FOR PUMP DIAGNOSTICS

Aiming at top-class performance for its new range of ultra-high-performance liquid-chromatography (UHPLC) pumps, Spark Holland engaged Controllab for the model-based design of the pump control. Together, they also developed a machine learning algorithm that can automatically diagnose pump data and indicate potential faults.

CORNELIS TUMP AND CHRISTIAN KLEIJN

Introduction

Spark Holland is a world-class provider of innovative sample introduction, extraction and separation technology for analytical systems. Their systems are used in laboratories all over the world. A key component of Spark is the ultra-high-performance liquid-chromatography (UHPLC) pump (Figure 1). This instrument can pump mobile-phase solvents with a very accurate flow rate over a large pressure range. For their new range of UHPLC pumps, the company wanted to achieve a level of performance that would be top-class. They asked Controllab for help on the control system of the pump. Together they not only succeeded in building this pump, but they also developed a machine-learning algorithm that can automatically diagnose pump data and indicate potential faults.

UHPLC pump

In analytical chemistry, UHPLC is a technique used to separate, identify and quantify components in a mixture; for instance, blood. It works by passing pressurised solvent containing the sample mixture under high pressure through a column with a solid adsorbent material. Each component interacts slightly differently with the adsorbent material, causing different flow rates and leading to the separation of the components as they run out of the column. The separated components can then be identified using an analyser.

To successfully separate the components, a UHPLC pump is used to pressurise the liquid solvent and maintain a constant flow rate and pressure at the outlet. A short drop in pressure or a small deviation in flow rate can lead to incorrect identification of the components in a sample. In 2012, Spark wanted to develop a new range of UHPLC pumps. Controllab was asked to help develop the control system.

Figure 2 shows a schematic of a UHPLC pump. Two pistons (primary piston PP and secondary piston SP) and two pressure chambers (primary and secondary chamber) are used to maintain a constant flow rate and outlet pressure. By using a specific stroke pattern for both pistons, the pressures inside both chambers can be controlled and the pressure of the secondary chamber can be kept constant. As this secondary chamber is directly connected to the outlet, this results in a constant flow rate and pressure for the outlet. Spark wanted the pump to operate in the pressure range of 0 to 1,300 bar and deliver a flow rate of 1 µl/minute to 5 ml/minute.

Model-based design

Controllab created a simulation model to investigate the pump design and assess the requirements. The model included the electric drives, brushless DC motors, lead screws, pistons, fluids and all the nonlinearities involved. The model was developed in 20-sim, a modelling and simulation package well suited for physics and control software. Test set-ups were made in Spark's laboratory to measure the nonlinearities of the components and to verify the components of the simulation model. Figure 3 shows the top view of the model.

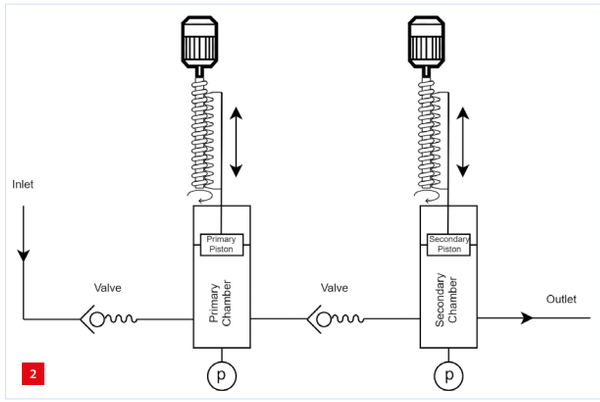
Analysis of the simulation data showed that micrometer accuracy of piston movement was needed to meet the flow requirements. In addition, the nonlinearities of the components and the properties of the fluid had to be incorporated in the control system to achieve the desired accuracy of the pressure.

The UHPLC pump of Spark Holland.

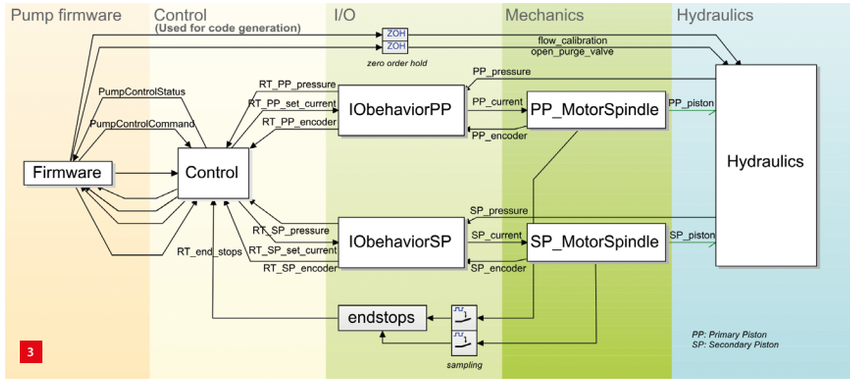
AUTHORS' NOTE

Cornelis Tump is the Research & Development director at Spark Holland, located in Emmen (NL). Spark Holland, a supplier of analytical systems for modern laboratories, is an expert in liquid handling and sample preparation. Christian Kleijn is the CEO and owner of Controllab, located in Enschede (NL). Controllab is the developer of the 20-sim modelling and simulation software package.

sales@sparkholland.com
www.sparkholland.com
info@controllab.nl
www.controllab.nl



UHPLC pump system containing electrically driven pistons.



Top view of the 20-sim simulation model for the UHPLC pump.

A control system was designed by Controllab that separated the pump strokes into different regions (forward-stroke pressure build-up, forward-stroke fluid flow, etc.) and applied a specific controller goal for each region. The control system performed well in several simulations. C-code was generated from the controller and deployed on an embedded board mounted on a test set-up. Various tests were carried out in the laboratory to verify the simulation results. The pump design was able to provide a stable flow under all conditions.

Based on this test set-up, Spark decided to go for it and scale up to detailed design. After a successful period of laboratory testing with a small batch of prototypes, the factory design was ready in 2015. The pump was an instant hit, with large numbers sold to customers all over the world.

Pump maintenance

The UHPLC pump contains parts that can wear out during use and cause failure. Examples are leakage of the primary and secondary seals and leakage of the check valve. Each pump contains firmware that communicates with the internal sensors and actuators and logs data to a memory buffer. This allows service engineers working on the pump (Figure 4) to retrieve the last 240 seconds of data and examine it to determine the cause of failure. The pump does not contain sensors to detect failures directly. The service engineer must therefore examine the data and search for deviations from normal use that may indicate the correct cause of the failure. Spark wanted to develop software that would help the service engineers identify the most likely cause of failure. The software needed to include a pattern recognition algorithm that could identify each type of failure with sufficient accuracy, but how?

Machine learning

Machine learning is a technique that can handle pattern recognition quite well. Large sets of data are used to train a neural network to recognise specific patterns and connect these to a specific failure, such as a primary seal leak. Once the network has been trained, its parameters are frozen. The network can then be used on a real pump to diagnose its performance and indicate the cause of failure.

To find good results with machine learning, large data sets are needed. For each component that can fail, the various stages of failure must be captured in the data sets. Multiple failing components require orthogonal data sets, which leads to an exponential growth of the data sets with the number of components. Spark did not have these data sets. Only a small set of pumps were dismantled in the laboratory to inspect the failed components and record usage data. On the other hand, they had the highly accurate simulation model from Controllab. So, this gave rise to the idea of using the simulation model to create synthetic data sets.

Failures can be easily introduced into a simulation model. For example, primary seal leakage can be introduced as a flow path from the piston chamber to the housing, where the flow is a function of the piston speed and chamber pressure. A degradation parameter such as the laminar flow conductivity can be used to indicate the leakage flow. In this way, failures can be added to the simulation model by increasing the degradation parameter. For each failure, a specific degradation parameter was used.

By running simulations, artificial data sets could be created. Now the first question was: is this data set rich enough to train a neural network? Are the failures leading to changes in behaviour that are distinguishable enough for a neural network to identify them individually? The second question was: will this network give good results in practice? Were the failures modelled sufficiently well to make the network also work well on data from real pumps?

Neural networks

A machine-learning pipeline was used to prepare the generated data sets for training (see Figure 5). In order to automate the data-generation process, degradation parameter combinations were defined in an Excel file, starting with individual non-zero parameters followed by combinations of non-zero parameters.

Service engineer working on the UHPLC pump.

Each failure class had approximately the same number of parameter combinations such that the resulting dataset was balanced. Using 20-sim's Python scripting interface, these parameter values could be entered into the model to run a simulation. The results of the simulation were stored in a .csv file (comma-separated values) containing all model variables against a fixed time step for the duration of one pump stroke.

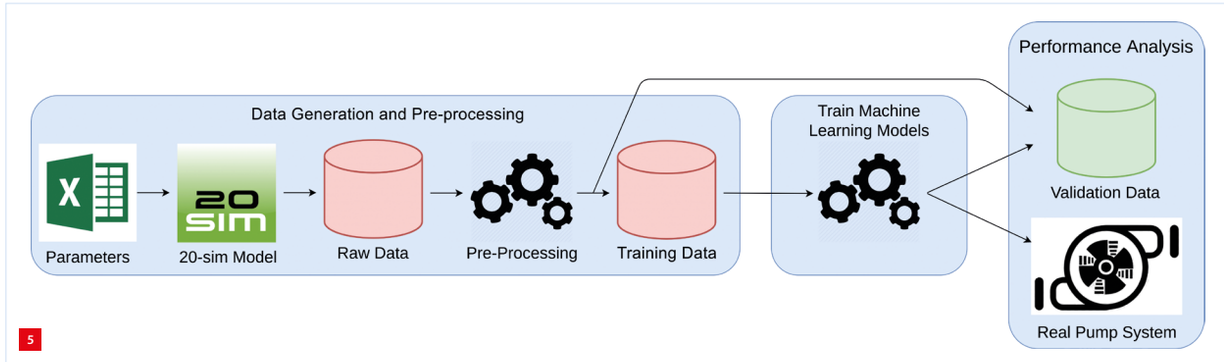
A pre-processing algorithm read these files and turned them into a data set that was suitable for training the neural network. During the pre-processing, only relevant data was selected, such as the pressure in the primary and secondary chambers of the pump and the position of the pistons. The selected variables were normalised using a min-max scaler so that all values were between 0 and 1 and stored in a new data file. This prevented variables with large numerical values from dominating the operation of the neural network.

Next, each data file that was the result of a simulation with one parameter set, was divided in windows based on the logic state variables of the controller. The selected windows were then interpolated to have a fixed window size of 100 samples so that the data was suitable for an LSTM (long short-term memory neural network) model. Finally, the resulting data files were divided in 80% training and 20% validation data.

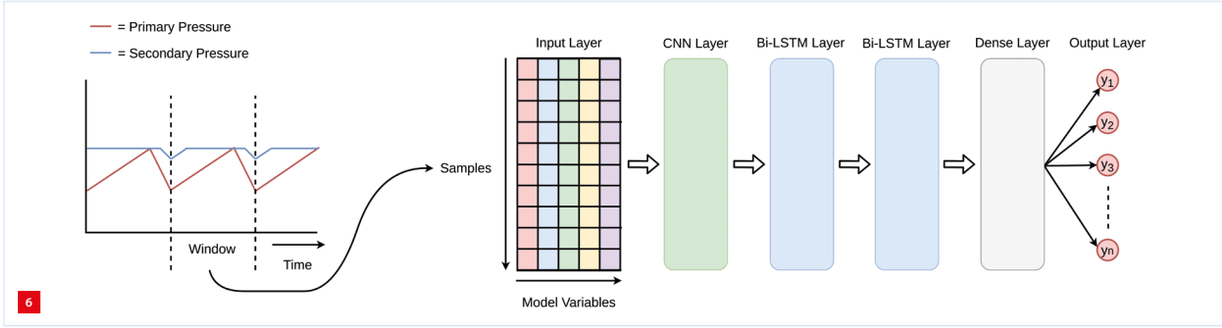
The neural network that was used consisted of a convolutional neural network (CNN) coupled to an LSTM. The CNN was used to perform feature extraction from raw data; features are recognisable parts if the data is plotted (e.g., mean value, maximum value, maximum slope, etc). The LSTM was used to recognise patterns in multivariate time series of data; patterns are combinations of features that indicate a failure and the severity of that failure. Figure 6 shows the neural network model architecture. The output of the network was an indication of the degradation of the pump due to a certain failure. For example, for the primary piston seal the leakage flow was given ranging from zero to severe in a number of steps.

Each signal consisting of 100 samples (each colour of the input layer is a different variable, with vertically the 100 time samples) was directly used as an input for the CNN. Several layers of LSTM read the outputs of the CNN and processed an output; the number of layers and number of neurons used per layer are called the hyperparameters of the neural network. The TensorFlow framework was used to implement the neural network. Using a batch of 64 data sets and Bayesian optimisation, the hyperparameters were determined; for example, the number of layers and the size that will give the best performing network. With these hyperparameter values, the complete data set was used to train the neural network. During this training, the parameters of the neural network were changed until the best performing network was obtained.

After the training, the parameters were frozen and the resulting neural network could be used to identify faults. The network was first tested on the batch of validation data set aside after the simulations. These tests showed that the network was capable of identifying various faults with an accuracy of 90% or higher. The network was then applied to a small set of test pumps available in the laboratory. Similar good results were obtained. The neural network was able to identify pump failures with sufficient resolution to help a service engineer determine which component needed to be replaced to get a pump working again.



The machine-learning pipeline.



The CNN + LSTM model.

Diagnostics

Spark has implemented the neural network into their diagnostics software. When service engineers connect their PC to a pump, the diagnostics software automatically guides them through the procedure. During a standard test, the UHPLC pump is filled with water and runs through a preset cycle of flow rates and pressures. As it runs, data is automatically logged and sent to the PC for analysis. The neural network is fed with the data and provides results. The software will translate this into instructions for the engineer. Once the faulty component has been replaced, the test is repeated to check that the pump is operating correctly.

The new diagnostics software has a number of benefits. First of all, the engineer has an easier job of troubleshooting the pump and fixing it. For the customer, this means lower costs because only the faulty component needs to be replaced. But the biggest advantage comes from the quantification of the various failures. For example, the neural network returns the leakage of a piston seal varying from zero to large in a number of steps. If the seal has a small leakage, the pump controller will compensate and maintain the desired flow rate and pressure. The customer will not notice but the diagnostics tool will. Moreover, the tool can potentially be used for prognostics; this could become a next step in R&D at Spark Holland.

Patents

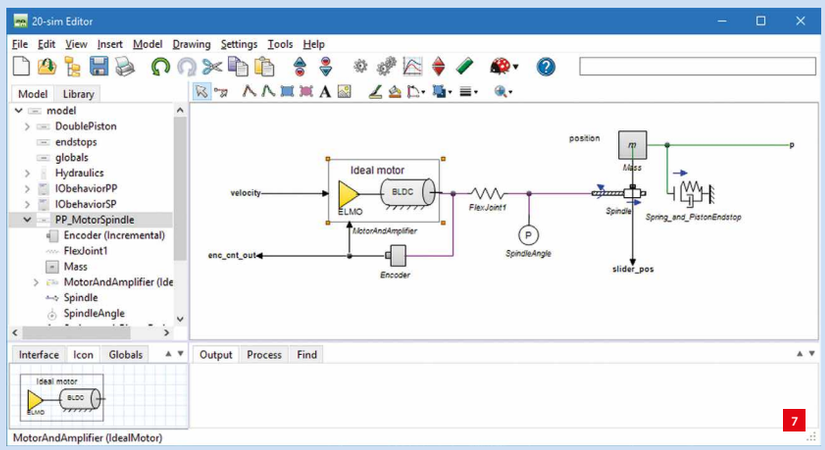
A database search showed that the approach of using model-generated data sets to train a neural network to find defects in a UHPLC pump was sufficiently new for a patent filing. The patent was filed in 2020 and granted in 2022. Also, a European patent about diagnostics is pending.

- US20220128522A1, "Training a neural network processor for diagnosis of a controlled liquid chromatography pump unit".
- EP 3 992 626 A1, patent pending.

20-sim

The 20-sim modelling and simulation software package for mechatronic systems is used by many companies in the precision engineering industry to analyse the dynamic behaviour of machines and develop and test control systems. With 20-sim, models can be entered graphically, similar to drawing an engineering scheme. These models enable simulating and analysing the behaviour of multi-domain dynamic systems and creating control systems.

20-sim comes with a large collection of library components that allows the quick assembly of electro-mechanical models such as the brushless-motor-driven spindle in Figure 7. This section was used as the drive train of the pumps in the UHPLC model. The control library was used to develop a controller for the pump. After successful simulations, the controller block was exported as C-code from 20-sim and deployed on the embedded controller of the UHPLC pump.



Port-based modelling of a motor-driven spindle in 20-sim.

All library components in 20-sim are open, allowing the easy addition of non-standard behaviour by changing the component equations. In case of the spindle (Figure 7), this was done by giving the linear thread small deviations within the tolerances provided by the manufacturer. Using Python scripting, the use of 20-sim can be automated. This allows engineers to change a model parameter, run a simulation and store the results in a .csv file. In this way, the data sets for the neural network were generated.

WWW.20SIM.COM