EIGENVECTOR RESEARCH INCORPORATED
Research, Training, and Software

3905 W. Eaglerock Dr.
Wenatchee, WA 98801
www.Eigenvector.com

Deploying your PCA Model Online using PLS_Toolbox with OPC

Key words: OPC Standard, OPC Client, OPC Server

Introduction: OPC is a well accepted standard within the process industries used for easily connecting software and hardware from different vendors. The MathWorks OPC Toolbox provides a powerful way to deploy your multivariate analysis applications online. Working in the MATLAB development environment using PLS_Toolbox and the OPC Toolbox gives you:

- The flexibility and speed of working in MATLAB.
- The power of state-of-the-art Multivariate Analysis (MVA) tools in PLS_Toolbox.
- The ease of OPC integration with the OPC Toolbox.

This document gives a brief introduction to these technologies and an overview of strategies to develop and deploy applications.

Background: OPC is a set of standards for communicating in a "process" environment. OPC compliant devices and software use a single standard for talking to each other, thus avoiding the need for working with thousands of individual drivers and proprietary formats. OPC is based on Microsoft's COM/DCOM (distributed / component object model), which provides the object model for OPC. See the OPC web site, http://www.opcfoundation.org, for further information.

OPC Basics: OPC has several standards but the three core standards are Data Access (DA), Historical Data Access (HDA), and Alarms and Events (AE). The DA standard describes a protocol for data transfer in real-time between clients and servers. Likewise, the HDA standard governs the protocol for accessing historical data. AE is concerned with real-time access to alarm notifications.

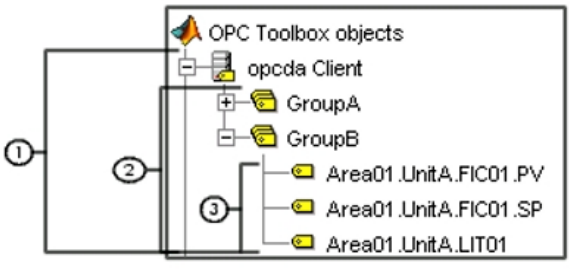
OPC employs a "client-server" network architecture. A server can take many forms and can implement multiple standards (DA, HAD, etc). They typically are shipped as a part of most SCADA, PLC and DCS systems. A client is a requester of an OPC server. Typically you can find HMIs (Human/Machine Interfaces) acting as OPC clients.

Servers are identified in a network by Host Name and Sever ID. A server will contain "tags" (data items) in a "Name Space". An example of a data item might be a sensor, actuator position, or state variable. The server is required to provide this Name Space to the network where each tag has a unique ID within the Name Space. A client can then connect to a specific server and access the items on that server.

OPC Toolbox Organization: OPC Toolbox is a Data Access client conforming to the OPC DA standard. You can read and write real-time data from/to OPC servers from within MATLAB, perform data logging tasks, and interface with Simulink.

Objects in the OPC Toolbox are organized in a hierarchy:

1) OPC Data Access Client (opcda) – Connection to an OPC Server.
   a) Requires a Host (the server computer) and Server ID (Program ID of the server, created when server is installed).
2) Data Access Group (dagroup) – Container for one or more data items.
   a) Manages how often child items are read.
   b) Whether data is logged.
   c) Must have opcda parent.
3) Data Access Item (daitem) – individual data items.
   a) Contains a value, quality, and timestamp collected from an instrument or SCADA.
   b) Data is current as of the last request made to access that instrument.
   c) Data can be logged.



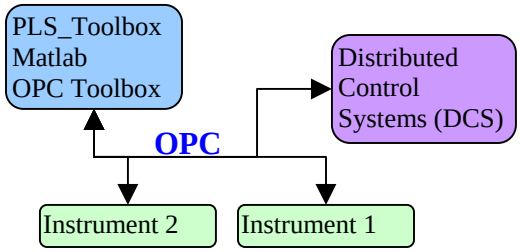
OPC Toolbox can be used in both a command line mode and through a GUI.

Using OPC with PLS_Toolbox: Typically, process models are developed using large historical datasets. During this state of development you would likely be working off-line with PLS_Toolbox to create and

3905 W. Eaglerock Dr.
Wenatchee, WA 98801
www.Eigenvector.com

optimize your specific model(s). Acquiring historical datasets may require the use of database querying tools, third party utilities, or the data logging feature of the OPC Toolbox.

Once you've developed a model you would then design an application to apply the model to new process data. Using the OPC Toolbox this becomes very simple. It's a matter of acquiring data, applying your model to the new data, and then returning/displaying the results or writing the results to an OPC server.



Principal Component Analysis (PCA): Let's assume you've already created a PCA model and you want to apply the model to a single new row of data and examine Q and T^2 contributions in real-time to see if the new sample (represented by the row) is significantly different than what you're expecting with your model.

1) Create a DA Group containing the same variables used to create your model.
2) Create a 'datachange' Callback function that applies new data to the PCA model. The function should provide for the following:
   a) Load the model.
   b) Acquire data from the group object and transform it into a form that can be applied by the model (vector or dataset object).
   c) Apply the model to your new data.
   d) Display or return your results (check Q and T^2 limits calculated when you created the model).
3) Make this function the 'datachange' Callback for the DA Group created in step one.

Exploring Data with PCA: In one example, you might be looking for "oddball" samples in a dataset. This is often accomplished using PCA and looking at how data clusters in a scores-scores plot. Outliers in this plot suggest an anomaly.

Although you could accomplish this using the logging function of OPC Toolbox you may also wish to continuously monitor a process in real-time.

1) Create a DA Group containing variables you wish to explore.
2) Create a 'datachange' Callback function that builds a PCA model on your stored data. The function should provide for the following:
   a) Acquire data from the group object and transform it into a form that can be used by the pca function (double or dataset object).
   b) Append this onto a MATLAB 'persistent' variable, this will retain the variable in memory between calls.
   c) Create a PCA model with your data.
   d) Display the results (e.g., in a scores-scores plot).
3) Make this function the 'datachange' Callback for the DA Group created in step one.

In this scenario you could watch your process in real-time and identify outlier samples. You could also apply this type strategy to create a "trending" analysis of your data.

Conclusion: OPC along with MATLAB, PLS_Toolbox, and OPC Toolbox represent a simple and effective way of integrating your data analysis application into an online environment. MATLAB provides a rapid development environment, PLS_Toolbox gives you advanced MVA tools, and OPC Toolbox provides a simple way of connecting to your OPC network. This paper showed some simple strategies of how you might integrate your process model online with OPC. As you become more familiar with the tools and techniques you'll undoubtedly find more sophisticated and valuable ways to harness this technology. For more information see the resources listed in the next section.

Additional Information:

PLS_Toolbox

- www.software.eigenvector.com

Mathworks Products

- OPC Toolbox
  www.mathworks.com/products/opc/
- Database Toolbox
  www.mathworks.com/products/database/

OPC

- www.opcfoundation.org