Correlation Analysis Use Case

Description of required AF functions

Main Author: Gerrit Stapper (Gerrit.Stapper@covestro.com)

COV-IO-S&A-GPCT-PA

Coauthors / Review: Achim Kuepper (Achim.Kuepper@covestro.com)

COV-IO-S&A-GPCT-APC

Document status: final

Document Version History

Version	Modification Date	Name (Role)	Change
0.1	2016-11-24	Gerrit Stapper (PI Application Coordinator)	Initial creation
1.0	2016-12-02	Dr. Achim Kuepper (Methodology Expert)	Review
1.1	2016-12-20	Dr. Achim Kuepper (Methodology Expert)	Edited for public release

Correlation Analysis Use Case
Description of required AF functions

Content

1. Basic use case description

The PI Application “Correlation Analysis” is currently a PI-ACE application which analyses correlations between data series of process data (e.g. two data series) and tries to search for discrepancies of the different data sources.

2. Required Functions

2.1. Covariance calculation

2.1.1. Code

The covariance calculation is required to get the rate of similarity between two value arrays. The covariance calculation is required to be implemented as following (C# code):

```
/// <summary>
/// Calculates the covariance between two double arrays.
/// </summary>
/// <param name="inputDataX"></param>
/// <param name="inputDataY"></param>
/// <returns>Returns double.NaN if the arrays haven´t the same length.
             Otherwise returning the calcuted covariance.</returns>
private double cov(double[] inputDataX, double[] inputDataY)
{
    if (inputDataX.Length == inputDataY.Length)
    {
        double meanX = mean(inputDataX);
        double meanY = mean(inputDataY);
        double ret = 0;
        for (int i = 0; i < inputDataX.Length; i++)
        {
            ret += ((inputDataX[i] - meanX) * (inputDataY[i] - meanY));
        }

        ret = ret / (inputDataX.Length);
        return ret;
    }
    else
    {
        return double.NaN;
    }
}
```

2.1.2. Parameter description

```
double[] inputDataX, double[] inputDataY
```

These are data series data arrays having double values. Since there are no arrays currently available in AF Analytics, these can be also PI Tags with a data horizon. Example: "Covariance('sinusoid', 'sinusoidu', '*', '*-300s')"

The code line

```
"if (inputDataX.Length == inputDataY.Length)"
```

needs to be adapted ofcourse, since the number of events for each tag can be different.

2.1.3. Result description

The calculated result is the covariance.

2.1.4. Example

See chapter 3, Attachment: Excel sheet "Covariance".

2.2. Max Covariance calculation

2.2.1. Code

The Max covariance calculation is more or less a for-next-loop executing multiple covariance calculations. The code is starting with two different value arrays coming from different data rows (PI Points). It shall be checked if there is a maximum covariance across a defined data horizon in such a case that the relative position between those data rows is shifted back and forth.

The result is a rate how big the delay between the two data rows is at that point where the covariance (the similarity) is at its maximum.

The covariance calculation is required to be implemented as following (C# code):

```
/// <summary>
/// Calculates the maximum covariance between two double arrays in a
/// given time horizon.
/// </summary>
/// <param name="val1"></param>
/// <param name="val2"></param>
/// <param name="search_length "></param>
/// <param name=" array_length "></param>
/// <returns></returns>
private int max_cov(double[] val1, double[] val2, int search_length,
                int array_length)
{
       double cov_max;
       double cov_val12;
       double[] val1_cov = new double[array_length];
       double[] val2_cov = new double[array_length];

       int foundSearchInterval;
       foundSearchInterval = 0;
       cov_max = 0;
       for (int ll = - search_length; ll <= search_length; ll++)
       {
              Array.Copy(val1, search_length, val1_cov, 0, array_length);
              Array.Copy(val2, search_length + ll, val2_cov, 0, array_length);
              cov_val12 = this.cov(val1_cov, val2_cov);
              if (cov_val12 > cov_max)
              {
                     cov_max = cov_val12;
                     foundSearchInterval = ll;
              }
       }
       return foundSearchInterval;
}
```

2.2.2. Parameter description

```
double[] val1, double[] val2, int search_length, int array_length
```

“val1” and “val2” are time series data arrays having double values. Since there are no arrays currently available in AF Analytics, these can be also PI Tags with a time horizon.

“search_length” is specifying how much val2 shall be shifted in relation to val1 to search the maximum covariance (either in terms of time or in terms of number of archive events).

“array_length” is specifying how many values are used from arrays “val1” and “val2”.

Example: “CovarianceMax(‘sinusoid’, ‘sinusoidu’, ‘*’, ‘*-300s’, 30,20)”

2.2.3. Result description

The calculated result is the iteration number of the maximum covariance found during the for-next-loop.

2.3. Archive Delete Function or temporary Array Function

During the calculations, the raw data coming from the data source (via PI-Tag data reference) needs to be filtered with some certain formula (called filtered raw data) and also saved with a history across some hundreds data points. This new PI-Tag again is required to calculate some more values, e.g. the Standard Deviation. After that, the filtered raw data is no longer used, only the standard deviation is used for later analyses.

There shall be a possibility to either create temporary arrays saving double-values on a mid-term basis (in terms of hours and days) or to provide a possibility to delete the history of PI Tags within the PI AF Asset Analytics.

3. Attachment

Attachment-Examples.xlsx