Chemical Process Fault Detection using Principal Component Analysis - A Short Course Project for Process Control

Joseph K. Scott*

June 28, 2023

Introduction. In complex engineered systems, including chemical processes, component malfunctions, process upsets, or other abnormalities are frequent and unavoidable. These events are referred to as *faults*. Detecting faults quickly and accurately is essential for maintaining safety, product quality, and profitability. As a result, *fault detection* has become an integral part of modern control systems. Fault detection methods continuously monitor the process output measurements and attempt to determine whether or not they are consistent with the expected output values given the control actions that have been taken. If the real outputs are significantly different than the expected values, then a fault has likely occurred. However, anomalous measurements can also result from many other factors, such as disturbances and process transients. Therefore, a key challenge in fault detection methods is to properly distinguish between disturbances, which require no special action, and true faults, which require immediate operator attention. Effective fault detection algorithms must detect faults rapidly in order to minimize the negative effects of the fault, but should also have a low false alarm rate. A *false alarm* is when the method declares that a fault has occurred, when in fact there is no fault in the real process.

In this assignment, you will implement and test a fault detection method based on principal component analysis (PCA) that is widely used in the chemical process industry. A key advantage of PCA is that it does not require a process model. The analysis is based entirely on process data and involves computations that can be scaled up to very large data sets, which is

*School of Chemical and Biomolecular Engineering, Georgia Institute of Technology (jscott319@gatech.edu)

important for large chemical processes. On the other hand, the PCA method is derived using a number of limiting assumptions, so it does not always work as expected in real processes. Moreover, there are fundamental limitations to what can be inferred from data alone, with no process model to use for guidance. Much academic research has been done, and continues to be done, to generalize this approach by using more sophisticated data analysis techniques (including machine learning) and/or incorporating process models/physics. However, these methods are beyond the scope of this assignment.

PCA Background. Principal component analysis is widely used for fault detection in industrial processes. The standard PCA-based fault detection method is described in [1] and will be followed here. This approach does not assume a system model or any information about the disturbances and measurement noises that affect the system. Instead, it assumes that we have a historical dataset. Assuming that n_y different process variables are measured, the historical data set can be represented by a matrix $\mathbf{Y}^\circ \in \mathbb{R}^{m \times n_y}$ composed of m measured output vectors $\mathbf{y}_k^\circ \in \mathbb{R}^{n_y}$ collected during some period when the process was operating at steady-state without faults. The PCA method is derived based on the assumption that these outputs obey a multivariate Gaussian distribution. This holds for linear systems when the disturbances and measurement noises are serially uncorrelated Gaussian sequences, but not necessarily otherwise.

The PCA method involves an offline calculation, where $\mathbf{Y}^\circ$ is analyzed to understand the normal behavior of the system and develop a fault detection test, and on online portion that is used in real time to test each new process measurement for evidence of a fault. In the offline calculation, $\mathbf{Y}^\circ$ is first normalized so that each column has zero mean and unit standard deviation (i.e., compute the mean and standard deviation of each column, subtract the mean from each element of the column, and then divide each element of the column by the standard deviation). Denote this scaled matix by $\mathbf{Y}$ and the scaled data points by $\mathbf{y}_k$. Next, the sample covariance matrix of the data is computed as

$$\mathbf{S} \equiv \frac{1}{m-1}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}. \tag{1}$$

The eigenvalue decomposition of $\mathbf{S}$ is computed to obtain

$$\mathbf{S} = \mathbf{V}\mathbf{D}\mathbf{V}^{\mathrm{T}}, \tag{2}$$

where $\mathbf{D}$ is a diagonal matrix with the eigenvalues of $\mathbf{S}$ in descending order on the diagonal, and $\mathbf{V}$ is an orthogonal matrix whose columns are the eigenvectors of $\mathbf{S}$ corresponding to each eigenvalue in $\mathbf{D}$ (This can be done with the Matlab command `eig(S)`).

The eigenvalues of $\mathbf{S}$ indicate how much of the variance in the data $\mathbf{Y}$ is explained by each eigenvector, as explained further in Problem 1. Often, there are several eigenvalues λ_i that are near zero, meaning that the data does not have much variance along the direction or the corresponding eigenvector $\mathbf{v}_i$. This indicates that some of the measured variables are not independent of the others. Eliminating these redundancies can reduce the dimensionality of the data, which is helpful for processes with a large number of measured variables, and can help to detect certain kinds of faults more clearly. Thus, the next step in PCA is to choose some number $p \leq n_y$ such that the eigenvalues λ_i for $i > p$ are small in magnitude relative to the others. Let $\bar{\mathbf{D}}$ be a $p \times p$ diagonal matrix with the largest p eigenvalues on its diagonal, and let $\bar{\mathbf{V}}$ be the $n_y \times p$ matrix whose columns are the eigenvectors corresponding to the largest p eigenvalues (i.e., the first p columns of $\mathbf{V}$). Since $\lambda_i \approx 0$ for $i > p$, it follows that

$$\mathbf{S} \approx \bar{\mathbf{V}}\bar{\mathbf{D}}\bar{\mathbf{V}}^{\mathrm{T}}. \tag{3}$$

This provides a much more compact representation of the data in many cases. The eigenvectors remaining in $\bar{\mathbf{V}}$ are called the *principal directions*. These directions are orthogonal to one another, and hence they describe a new coordinate system in which to view the data that makes it easier to detect certain kinds of faults. The matrices $\bar{\mathbf{V}}$ and $\bar{\mathbf{D}}$ are the final result of the offline part of the PCA method.

The online part of the PCA method is executed every time that a new output measurement $\mathbf{y}_k^\circ$ is obtained from the process. The first step is to normalize the new measurement $\mathbf{y}_k^\circ$ to obtain $\mathbf{y}_k$ in the same way as the columns of the historical data $\mathbf{Y}^\circ$ were normalized. Next, $\mathbf{y}_k$ is decomposed into its components along each principal direction $\mathbf{v}_i$ in $\bar{\mathbf{V}}$ (recall that these directions describe an orthogonal basis that is aligned with the shape of the

data). This information is contained in the *score vector* $\mathbf{t}_k$, which is computed by

$$\mathbf{t}_k = \bar{\mathbf{V}}^{\mathrm{T}} \mathbf{y}_k. \tag{4}$$

Each element of $\mathbf{t}_k$, say $(\mathbf{t}_k)_i$, describes the component of the vector $\mathbf{y}_k$ along the principal direction $\mathbf{v}_i$. Since λ_i describes the variance of the historical data in the direction $\mathbf{v}_i$, it can be shown that, if the process is fault-free, the i^{th} component of $\mathbf{t}_k$ should obey a normal distribution with mean zero and standard deviation $\sqrt{\lambda_i}$, and should be uncorrelated from the other components. It follows that the components of the scaled vector $\mathbf{D}^{-\frac{1}{2}}\mathbf{t}_k$ obey independent normal distributions with mean zero and standard deviation 1. Since each element of this vector is expected to have mean zero, the PCA method looks at the sum of squares of these variables to detect a fault. This is called the T^2 statistic and is computed as

$$\begin{aligned} T^2 &= \sum_{i=1}^{p} \left[\lambda_i^{-\frac{1}{2}} (\mathbf{t}_k)_i \right]^2, && (5) \\ &= (\bar{\mathbf{D}}^{-\frac{1}{2}} \mathbf{t}_k)^{\mathrm{T}} \bar{\mathbf{D}}^{-\frac{1}{2}} \mathbf{t}_k, && (6) \\ &= \mathbf{t}_k^{\mathrm{T}} \bar{\mathbf{D}}^{-1} \mathbf{t}_k, && (7) \\ &= \mathbf{y}_k^{\mathrm{T}} \bar{\mathbf{V}} \bar{\mathbf{D}}^{-1} \bar{\mathbf{V}}^{\mathrm{T}} \mathbf{y}_k. && (8) \end{aligned}$$

Since we expect T^2 to be near zero, a large value of T^2 would indicate a fault. However, T^2 will never be exactly zero due to normal process variability. Thus, in order to use T^2 to detect faults effectively, a method is needed to calculate how large T^2 is likely to get during fault-free operation. From the discussion above, we know that T^2 is the sum of the squares of p variables that all obey independent normal distributions with mean zero and standard deviation 1. Such a sum obeys a χ^2 distribution with p degrees of freedom. A more careful statistical analysis shows that the distribution of T^2 needs to be adjusted to account for the fact that $\bar{\mathbf{D}}$ and $\bar{\mathbf{V}}$ were estimated from data and are not known exactly, leading to the conclusion that T^2 satisfies Hotelling's T^2-distribution, which is where it gets its name. In any case, since the distribution of T^2 in the fault-free case is known, a confidence threshold for this value can be calculated analytically. To do so, we first specify a confidence

limit η. Then, the T^2 threshold can be computed as

$$T_\eta^2 = \frac{p(m-1)(m+1)}{m(m-p)} F_\eta(p, m-p). \tag{9}$$

Here, $F_\eta(p, m-p)$ is the inverse cumulative distribution function for the F distribution, whose value can be computed in matlab using the function `finv` taking three inputs as η, p, and $m-p$.

Online, a fault is declared whenever the T^2 value for a new measurement exceeds the threshold. The choice of η effectively determines the false alarm rate of the method, since we expect T^2 to fall outside of the threshold with probability $(1-\eta)$ even when the process is fault-free. Therefore, large η values are preferable. On the other hand, choosing η too large will result in a loss of sensitivity. A typical value is 0.95.

Problem 1. Open the file `P1_main`. The provided code in this file generates and plots a set of data consisting of 500 data points for 2 measured variables. Taking this data to be the matrix $\mathbf{Y}^\circ$ in the background above, do the following:

1. Scale the data to obtain $\mathbf{Y}$ as described above. Generate a new scatter plot showing the scaled data instead of the original data.
2. Calculate the covariance matrix $\mathbf{S}$ and its eigenvalue decomposition. What do you notice about the magnitudes of the two eigenvalues? How do you recommend choosing p for PCA?
3. On the same figure as the data from Part 1, plot two lines corresponding to the directions of the two eigenvectors of $\mathbf{S}$. Visually examine how spread out the data is along each eigenvector and compare this to the corresponding eigenvalues. What do you notice?

Problem 2. Open the files `P2_main.m`, `simulateProcess.m`, and `getT2.m`. This problem will guide you through the steps of applying PCA fault detection to a linear system of the form

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{E}\mathbf{w}_k, \tag{10}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{F}\mathbf{v}_k. \tag{11}$$

There are 5 states, 5 outputs, 2 disturbances, and 5 measurement noises. The matrices $\mathbf{A}$, $\mathbf{E}$, $\mathbf{C}$, and $\mathbf{F}$ are defined in `simulateProcess.m`. Although the PCA method does not use a model, we will use this model to simulate the 'real' system so that we can test the PCA method under fault-free and faulty scenarios. The file `simulateProcess.m` simulates the model above (in the fault-free case) over a horizon of 50 time steps using randomly generated disturbances and measurement noises. The distributions of $\mathbf{w}_k$ and $\mathbf{v}_k$ are described in `simulateProcess.m`. Running `P2_main.m` should produce a plot of the outputs of this simulation versus time.

Take some time now to read the rest of the problem statement and then read the provided Matlab files, including all comments, before beginning work. There are many important details in the comments, including parts that you need to add/modify as part of the assignment.

1. The file `historicalData.csv` contains 500 output measurements from the process collected during a time where the process was fault-free. This file is read into a matrix at the beginning of `P2_main.m`. Write code in the first section of `P2_main.m` (offline PCA calculations) to calculate $\bar{\mathbf{V}}$ and $\bar{\mathbf{D}}$. You should find that $p = 4$ is an appropriate choice. Report values for $\mathbf{V}$, $\mathbf{D}$, $\bar{\mathbf{V}}$, and $\bar{\mathbf{D}}$.

2. Review and complete the code in the second section of `P2_main.m` (online PCA calculations). You must provide code in the indicated position for scaling the new measurement. You must also complete the appropriate code for calculating the T^2 value for a given $\mathbf{y}$ inside of the provided function template `getT2.m`. Finally, you will need to compute the threshold for T^2 in order to test the T^2 values you compute. When this is done, running `P2_main.m` should produce a plot of T^2 versus time for the fault-free model. Provide this plot in your solution and comment on the observed false alarm rate.

3. Now that we have tested PCA on the fault-free process conditions, we are going to introduce a fault. Modify the code in `simulateProcess.m` to simulate the case where a fault causes the distribution of $\mathbf{w}_k$ to change after time step 50. Specifically, shift the mean of the distribution of both components of $\mathbf{w}_k$ by 3. Show the T^2 plot. Does PCA detect the fault?

4. Redo Part 3, only this time model a fault where only the mean of the first component of $\mathbf{w}_k$ is shifted by 3, while the second component is unaffected. Show the T^2 plot. How is this plot different than the one produced in Part 3? Does PCA detect the fault?

5. Spend some time experimenting with other possible faults. Find one fault that PCA detects effectively and one that it does not detect effectively and show the T^2 plots. (Hint: Experiment with introducing faults into the matrices $\mathbf{A}$, $\mathbf{E}$, $\mathbf{C}$, and $\mathbf{F}$.)

References

[1] Leo H Chiang, Evan L Russell, and Richard D Braatz. *Fault detection and diagnosis in industrial systems.* Springer Science & Business Media, 2000.