FOPAM Workshop Module 2

Latent Variable Analytics and Nonlinear and Dynamic Extensions

S. Joe Qin

In collaboration with
Leo Chiang and Richard Braatz

Raleigh, NC, Aug. 4-5, 2019

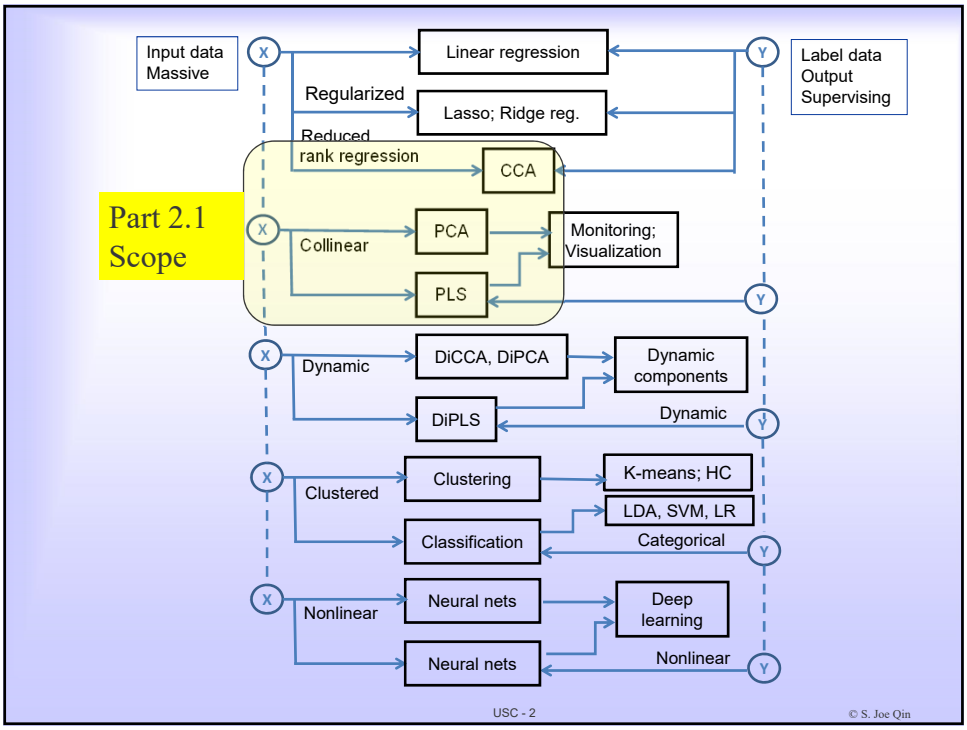
Content in Module 2

1. PRINCIPAL COMPONENT ANALYSIS, PARTIAL LEAST SQUARES, AND CANONICAL CORRELATION ANALYSIS
2. PROCESS MONITORING USING PRINCIPAL COMPONENT ANALYSIS
3. DYMANIC COMPONENT MODELING FOR MULTIVARIATE TIME SERIES
   – DYNAMIC PCA, PLS, CCA
4. Classification Methods: Discriminant Analysis and Support Vector Machines
5. Nonlinear and Neural Network Extensions to "Latent Variables"

MODULE 2 – PART 1
PRINCIPAL COMPONENT ANALYSIS, PARTIAL LEAST SQUARES, AND CANONICAL CORRELATION ANALYSIS

S. Joe Qin

For the FOPAM Workshop in collaboration with Leo Chiang and Richard Braatz



OUTLINE

- Principal component analysis (PCA)
  - History
  - Objective
  - [Workshop]
- Partial least squares (PLS)
  - Performs regression with collinearity in the input variables
  - Objective
  - [Workshop]
- Canonical correlation analysis (CCA)

Historical Perspective of PCA

Karl Pearson, 1901

[559]

LIII. *On Lines and Planes of Closest Fit to Systems of Points in Space.* By KARL PEARSON, *F.R.S., University College, London* *.

K. Pearson, *"On lines and planes of closest fit to systems of points in space"*, The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, Sixth Series, 2, pp. 559-572 (1901)

(1857-1936)

Harold Hotelling

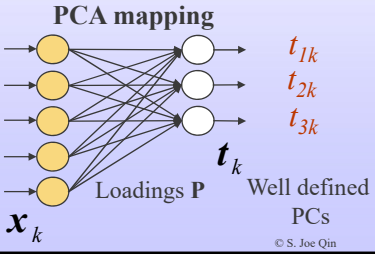
Harold Hotelling formulated PCA in the modern form.

- Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology, 24*(6), 417-441.

Historical Perspective

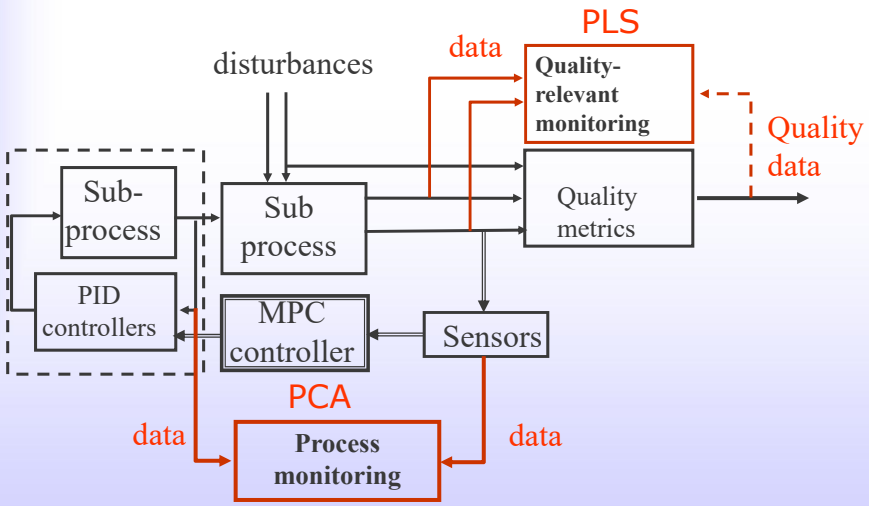
PCA and Machine Learning

- PCA (Pearson, 1901)
- Hebbian learning (1940s), unsupervised, to alter the weight between two neurons
- ADALINE (Adaptive Linear Neuron, 1960) with sign nonlinearity
- Multilayer neural networks (sigmoidal, late 1980's)
- PCA is linear neural networks (Oja, 1992)



E. Oja (1992). Principal components, minor components, and linear neural networks, Neural Networks, 5, 927-935.

Data-driven process monitoring



Principal Components Analysis

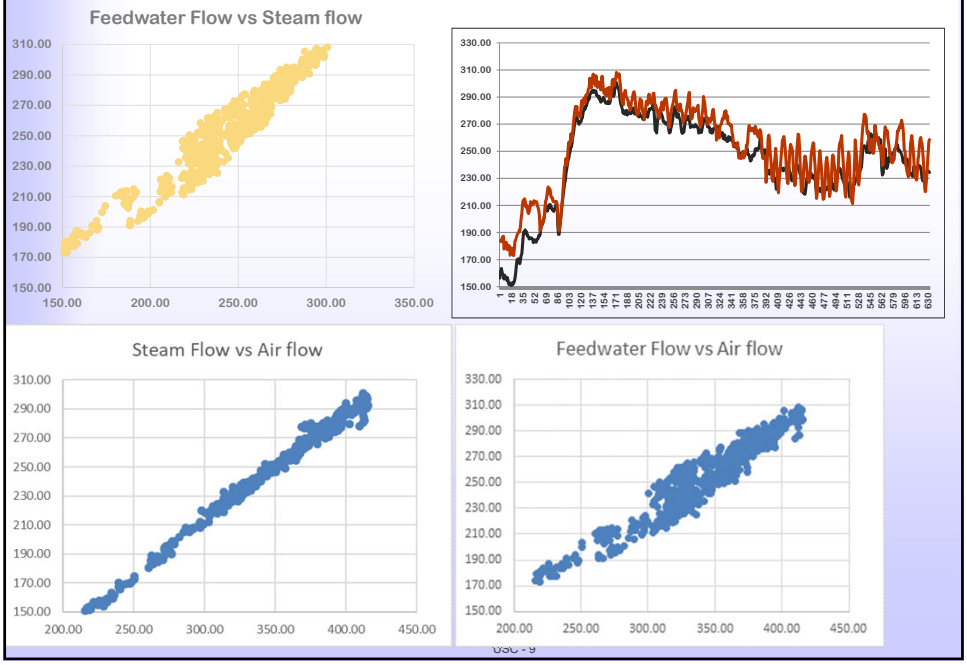
PCA produces a low-dimensional representation of a dataset.

- It finds a number of linear combinations of the variables that
  - have maximal variance, and
  - are mutually uncorrelated.
- PCA is a dimension reduction tool, thus useful for data visualization and exploratory analysis
- Process monitoring: detecting abnormal situations reflected by process data.

Correlations/collinearity in Process Data

- Collinearity can come from
  - Mass balance, energy balance
  - Control-induced correlations, e.g.,
    - Ratio control of air flow and fuel flow
    - Feedback/feedforward control makes steam flow and water flow correlated
  - Operational restrictions and safety constraints
    - Excessive stack oxygen is necessary for safety, but too much excess is a waste of energy

Data Correlation: feedwater, air flow, steam flow

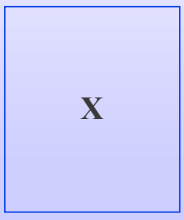


Data Matrix

- $x_1, x_2, \ldots x_M$ --- M variables or features
- N observations:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_M \end{bmatrix} = \begin{bmatrix} x_{11} & \vdots & \vdots & x_{1M} \\ x_{21} & \vdots & \vdots & x_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & \vdots & \vdots & x_{NM} \end{bmatrix}$$

- Block notation:



USC - 10

Correlation Analysis

- Correlation coefficient tells one-to-one correlation

$$r_{ij} = \frac{1}{N-1}\sum_{k=1}^{N}\frac{(x_{ki}-\bar{x}_i)(x_{kj}-\bar{x}_j)}{s_i s_j}$$

where $0 \le r_{ij} \le 1$

- Correlation matrix for all pairs of variables

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ r_{21} & r_{22} & \cdots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{M1} & r_{M2} & \cdots & r_{MM} \end{bmatrix}$$

Often x_i is centered to zero mean and unit standard deviation, i.e., s_i=1. Then

$$r_{ij} = \frac{1}{N-1}\sum_{k=1}^{N} x_{ki}x_{kj} = \frac{1}{N-1}\left[x_{1i}x_{1j} + x_{2i}x_{2j} + \cdots + x_{Ni}x_{Nj}\right]$$

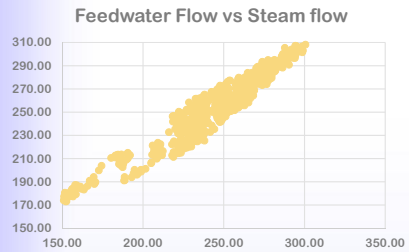
$$\frac{1}{N-1}\mathbf{x}_i^T\mathbf{x}_j$$

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ r_{21} & r_{22} & \cdots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{M1} & r_{M2} & \cdots & r_{MM} \end{bmatrix} = \frac{1}{N-1}\left\{\mathbf{x}_i^T\mathbf{x}_j\right\} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}$$

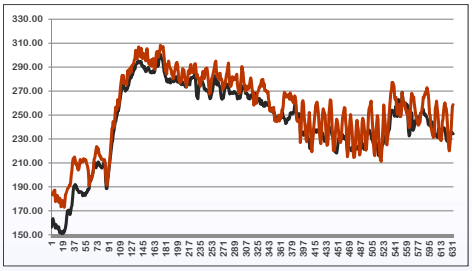
We often say that $\mathbf{X}^T\mathbf{X}$ 'is' the sample correlation/covariance matrix.

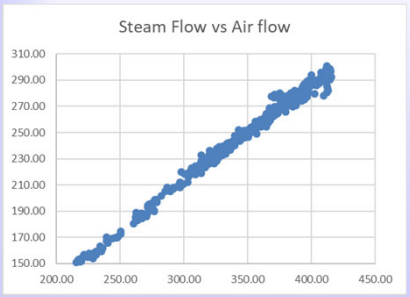
The population covariance is $\mathbf{R}$ when $N \to \infty$

Correlation coefficients

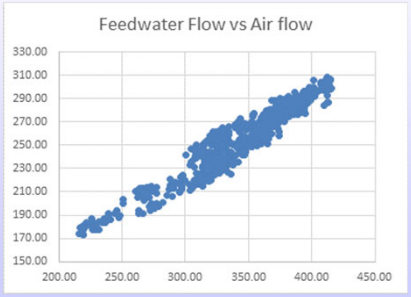


Correlation coefficient: 0.9579





Correlation coefficient: 0.9955

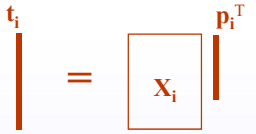


Correlation coefficient: 0.9573

Principal Component Analysis (PCA)

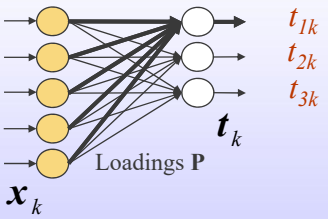
1. Scale X to zero mean and unit variance.
2. Initialize: $\mathbf{X}_1 := \mathbf{X}$
3. Project or combine:

$$\mathbf{t}_i = \mathbf{X}_i \mathbf{p}_i \quad \text{and} \quad \mathbf{p}_i^T \mathbf{p}_i = 1$$



4. PCA objective:

$$\max \mathbf{t}_i^T \mathbf{t}_i$$

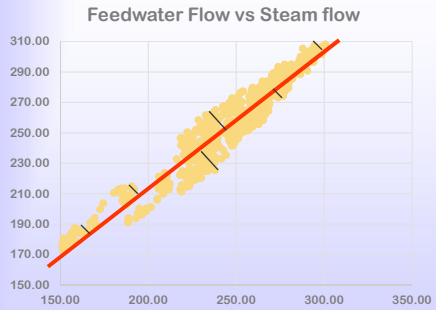


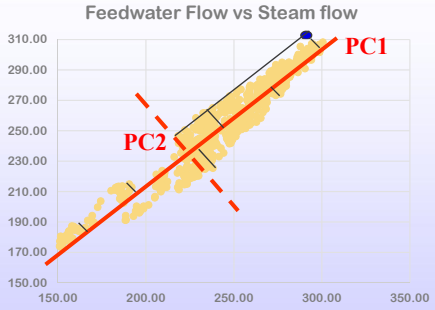
5. Residuals: $\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{t}_i \mathbf{p}_i^T$
6. Set *i*:=*i*+1 and return to 3 for the next component

$$\mathbf{P} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_l] \qquad \tilde{\mathbf{P}} = [\mathbf{p}_{l+1} \quad \mathbf{p}_{l+2} \quad \cdots \quad \mathbf{p}_M]$$

Example: water and steam flow

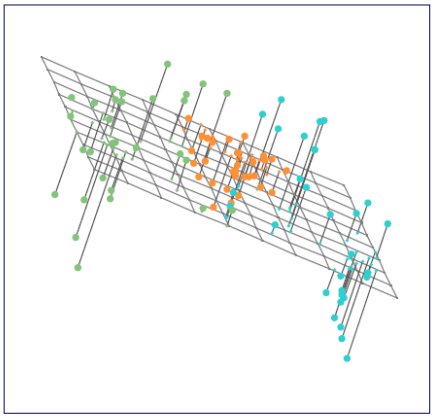
- Large variance direction is easy to **visualize** features in the data
- PCs are orthogonal
- PCs are **features** from rotating the coordinates

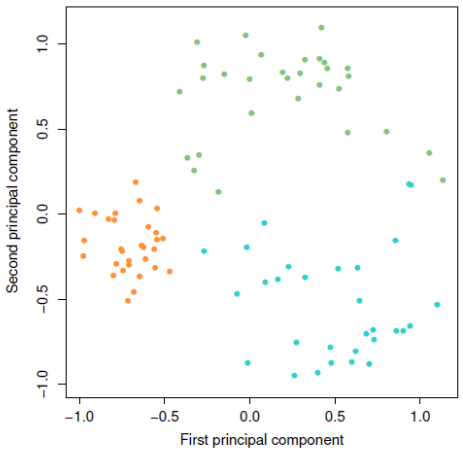




PCA achieves

- Largest variance, the view is better
- Minimum perpendicular distance

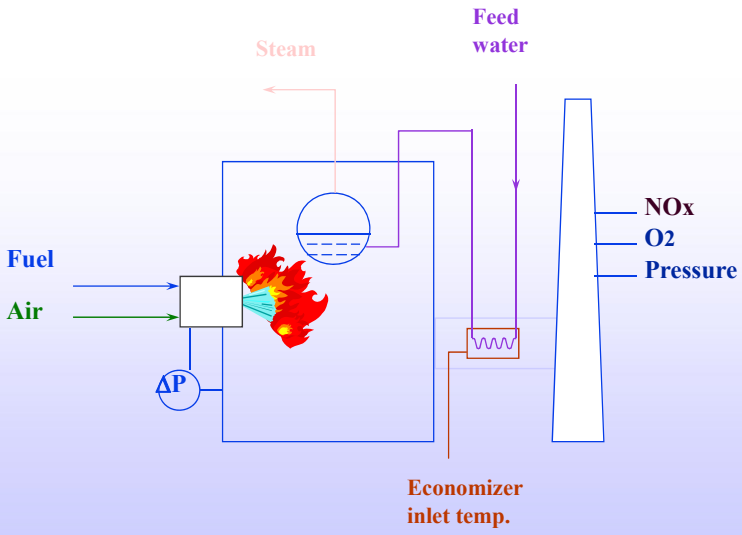




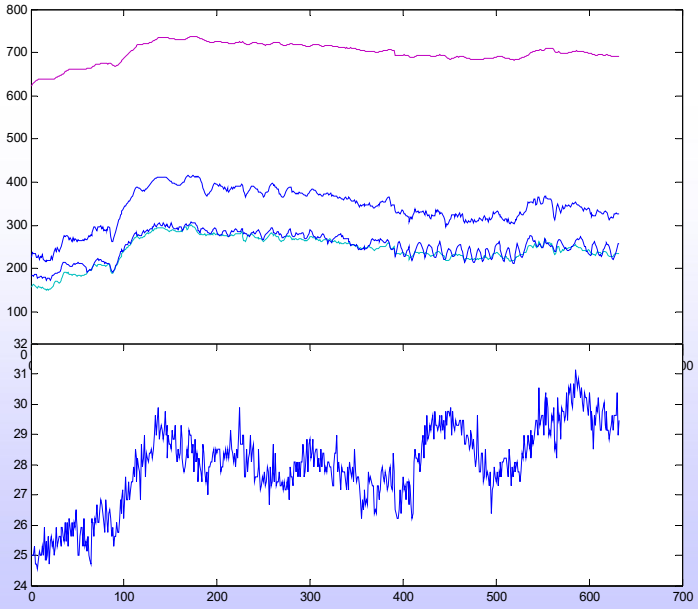
PCA Summary

- The PCs capture the maximum variance in the data
- No output variables/labels – unsupervised learning
- All variables respond to the PCs which are the latent, unmeasured variables

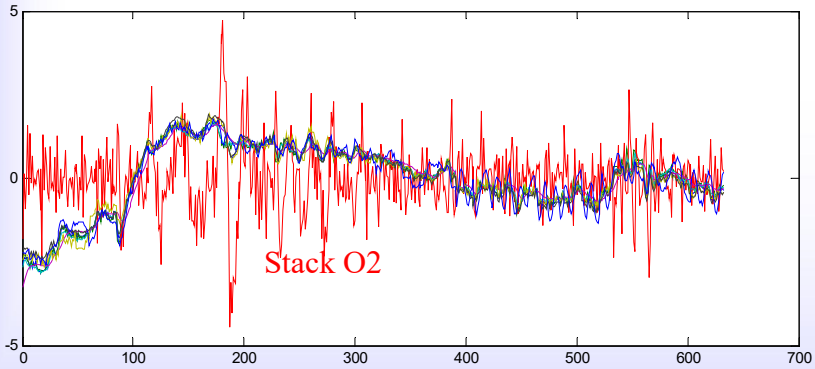
Boiler Data Example



Input and Output (NOx) data



Scaled to zero mean and unit variance



PCA on the Boiler Data

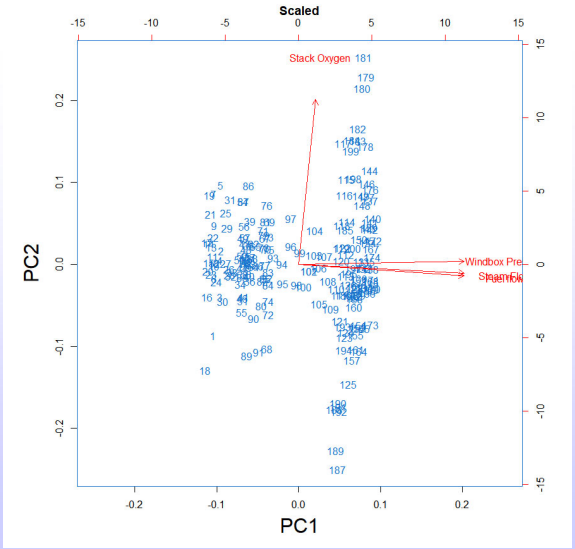
- PCA on Fuel Flow; Windbox Pressure; Steam Flow; and Stack Oxygen (first 200 observations)
  - Show the loadings in a table
  - Plot the scatter biplots (scores & loadings)
  - Compare the biplots with and without variance scaling
  - Plot the variances explained by each PC, Percent variance explained (PVE)

PCA loadings

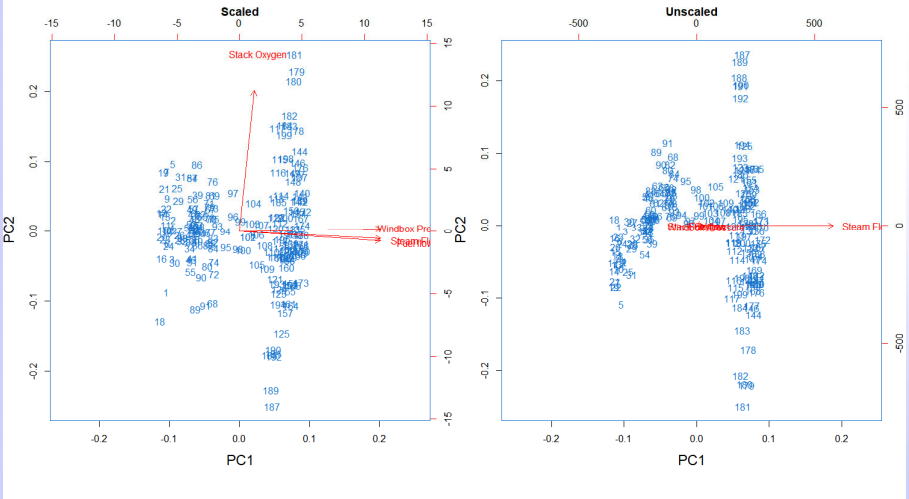
	p1	p2	p3	p4
Fuel Flow	0.5760	-0.0673	0.2007	0.7896
Stack Oxygen	0.0595	0.9962	0.0576	-0.0269
Steam Flow	0.5763	-0.0526	0.5809	0.5725
Windbox Pressure	0.5767	0.0169	0.7868	0.2193

Note: the loading vectors are unit norm, orthogonal, and signs don't matter

PCA biplot: show scores and loadings together



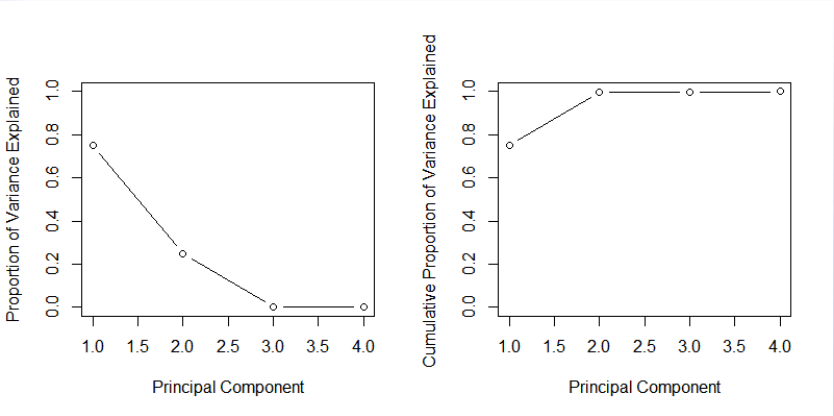
Scaling of the variables matters



Cumulative Percent Variance with l PCs

Select the number of PCs to achieve 90% CPV:

$$\mathrm{CPV}(l) = \sum_{i=1}^{l} \lambda_i \Big/ \sum_{i=1}^{M} \lambda_i$$



How many PCs should we use?

- If we use PCs as a summary of our data, how many PCs are sufficient?
  - No simple answer to this question
- The "scree plot" on the previous slide: Look for an "elbow that touches the floor".
  - 2 PCs is a good choice.
- NOTE: Sometimes the largest PC is not a feature of interest – domain knowledge helps here

Advanced Material

Properties of PCA

1) Scores are linear combinations of variables: $\mathbf{t}_i = \mathbf{X}\mathbf{p}_i$

2) Loadings are eigenvectors of the covariance matrix:

$$\mathbf{X}^T\mathbf{X}\mathbf{p}_i = \lambda_i \mathbf{p}_i \qquad \text{with } \mathbf{p}_i^T\mathbf{p}_i = 1$$

3) Both loadings and scores are orthogonal

$$\mathbf{p}_j^T\mathbf{p}_i = 0 \text{ and } \mathbf{t}_j^T\mathbf{t}_i = 0 \quad \text{for } j \neq i$$

4) Collect all loadings and scores into matrices

$$\bar{\mathbf{P}} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_M] \qquad \bar{\mathbf{T}} = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_M]$$

then from 1) to 3), $\bar{\mathbf{T}} = \mathbf{X}\bar{\mathbf{P}}$ $\qquad \bar{\mathbf{P}}^T\bar{\mathbf{P}} = \mathbf{I}_{M\times M}$

We have $\bar{\mathbf{T}}^T\bar{\mathbf{T}} = \bar{\mathbf{P}}^T\mathbf{X}^T\mathbf{X}\bar{\mathbf{P}} = \bar{\Lambda} \equiv \text{diag}\{\lambda_1 \quad \cdots \quad \lambda_M\}$

or $\mathbf{R} \leftarrow \mathbf{X}^T\mathbf{X} = \bar{\mathbf{P}}\bar{\Lambda}\bar{\mathbf{P}}^T = \mathbf{P}\Lambda\mathbf{P}^T + \tilde{\mathbf{P}}\tilde{\Lambda}\tilde{\mathbf{P}}^T$

PCA Relations per Observation

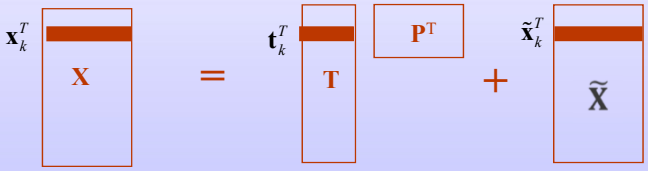
where $\bar{\mathbf{P}} = \left[\underbrace{\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_l}_{\mathbf{P}} \,\middle|\, \underbrace{\cdots \quad \mathbf{p}_M}_{\tilde{\mathbf{P}}} \right]$

$$\bar{\mathbf{T}} = \left[\underbrace{\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_l}_{\mathbf{T}} \,\middle|\, \underbrace{\cdots \quad \mathbf{t}_M}_{\tilde{\mathbf{T}}} \right]$$

then, $\mathbf{X} = \bar{\mathbf{T}}\bar{\mathbf{P}}^T = \underbrace{\mathbf{T}\mathbf{P}^T}_{\hat{\mathbf{X}}} + \underbrace{\tilde{\mathbf{T}}\tilde{\mathbf{P}}^T}_{\tilde{\mathbf{X}}} = \hat{\mathbf{X}} + \tilde{\mathbf{X}}$

where $\tilde{\mathbf{X}} = \tilde{\mathbf{T}}\tilde{\mathbf{P}}^T$ is the residuals.

For the k-th observation,



PCA Scores and Estimates

- PCA 'projection' for the k-th sample:

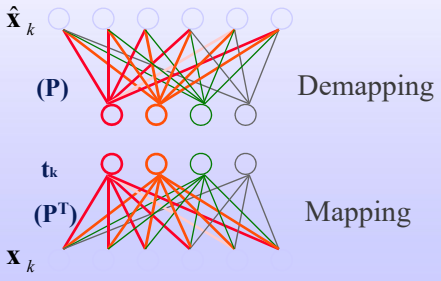
$$\hat{\mathbf{x}}_k = \mathbf{P}\mathbf{t}_k = \mathbf{P}\mathbf{P}^T\mathbf{x}_k$$

- A sample at *k* is mapped to the scores:

$$\mathbf{t}_k = \mathbf{P}^T\mathbf{x}_k$$

- And the PCA residual:

$$\tilde{\mathbf{x}}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k = (\mathbf{I} - \mathbf{P}\mathbf{P}^T)\mathbf{x}_k$$

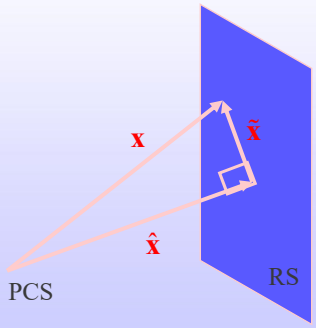


Subspace Projection

- PCS: Principal Component Subspace
- RS: Residual Subspace

$$\mathbf{x} = \hat{\mathbf{x}} + \tilde{\mathbf{x}}$$

and $\hat{\mathbf{x}} \perp \tilde{\mathbf{x}}$



[WORKSHOP] on PCA

Partial Least Squares (PLS)

- We know that ordinary least squares (OLS) suffers from collinearity in X data
- PLS reduces dimensions to avoid the negative impact of collinearity in X data
- Unlike PCA, PLS uses Y to supervise the extraction of X components
- PLS attempts to find the factor directions to explain both Y and X simultaneously.

Partial Least Squares (PLS)

- Initializing: $\mathbf{X}_1 := \mathbf{X}$ $\mathbf{Y}_1 := \mathbf{Y}$
- Projections: start from $i := 1$

- **PLS objective:** maximizing covariance

$$\max \mathbf{t}_i^T \mathbf{u}_i \quad \text{where } \mathbf{t}_i = \mathbf{X}_i \mathbf{w}_i \quad \text{and} \quad \mathbf{u}_i = \mathbf{Y}_i \mathbf{q}_i$$

- Inner regression: $\mathbf{u}_i = b_i \mathbf{t}_i + \mathbf{r}_i \quad \Rightarrow b_i = \mathbf{u}_i^T \mathbf{t}_i / \mathbf{t}_i^T \mathbf{t}_i$
- Residuals:

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{t}_i \mathbf{p}_i^T$$
$$\mathbf{Y}_{i+1} = \mathbf{Y}_i - b_i \mathbf{t}_i \mathbf{q}_i^T$$

Inner model where $\mathbf{p}_i = \mathbf{X}_i^T \mathbf{t}_i / \mathbf{t}_i^T \mathbf{t}_i$

- Let $i := i+1$, PLS then **iterates** for the next factor
- PLS extracts components by the size of the covariance

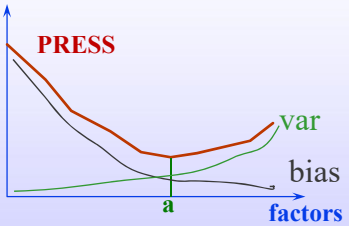
Selecting the number of PLS factors: Cross-Validation or Train/test validation

- If all data are used for training the PLS model, the training error will decrease as more factors are used in the model
  - Train/test validation (with lots of data) – use a portion of the data for training and the rest for testing
  - **Cross-validation** (limited data OK) – split the data in s-fold; keep one of them for testing and the rest for training. This is done in turn. The smallest test error for a given number of factors leads to the right model

Cross-Validation Procedure

- CV is used to select the number of PLS factors

- Data are split into s-fold. Build a model on (s-1) blocks, and calculate the predicted error sum of squares (PRESS) on the left-out block



- Final PLS Model: build PLS again with (a) factors using all data

Train/test Validation

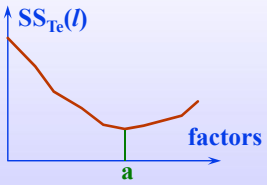
- Divide all data into s=2 subsets

X_1	X_2
Y_1	Y_2

Test on Set 2 with a model built on Set 1

- Calculate the test error sum of squares

$$SS_{Te}(l) = \sum_{j=1}^{M_y} \sum_{i=1}^{n_{test}} (y_{ij} - \hat{y}_{ij}(l))^2$$



where $\hat{y}_{ij}(l)$ is the predicction using a model with l factors

- Final PLS Model: run PLS again using all data

Cross-Validation example: s=3

Divide all data into s=3 subsets and calculate the PRESS on the test set(i):

Test on 1 with a model built on 2 and 3

X_1	X_2	X_3
Y_1	Y_2	Y_3

$$\rightarrow PRESS(l, s=1) = \sum_{j=1}^{M_y} \sum_{i=1}^{n_1} (y_{ij}^{(1)} - \hat{y}_{ij}^{(1)}(l))^2$$

Test on 2 with a model built on 1 and 3

X_1	X_2	X
Y_1	Y_2	Y

$$\rightarrow PRESS(l, s=2) = \sum_{j=1}^{M_y} \sum_{i=1}^{n_2} (y_{ij}^{(2)} - \hat{y}_{ij}^{(2)}(l))^2$$

Test on 3 with a model built on 2 and 1

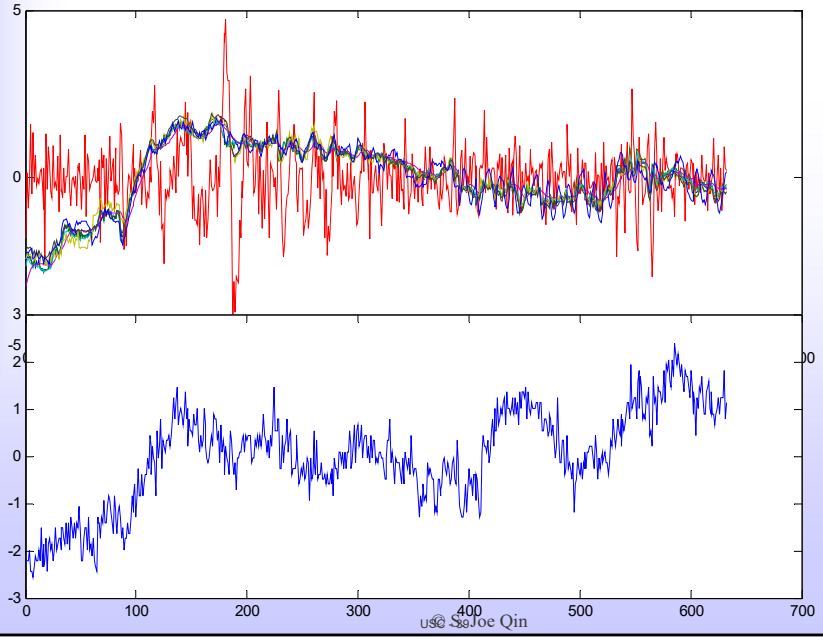
X_1	X_2	X_3
Y_1	Y_2	Y_3

$$\rightarrow PRESS(l, s=3) = \sum_{j=1}^{M_y} \sum_{i=1}^{n_3} (y_{ij}^{(3)} - \hat{y}_{ij}^{(3)}(l))^2$$

$$\text{Total: } PRESS(l) = \sum_{i=1}^{s=3} PRESS(l, i)$$

$$\text{for } l = 1, 2, \cdots a, \cdots$$

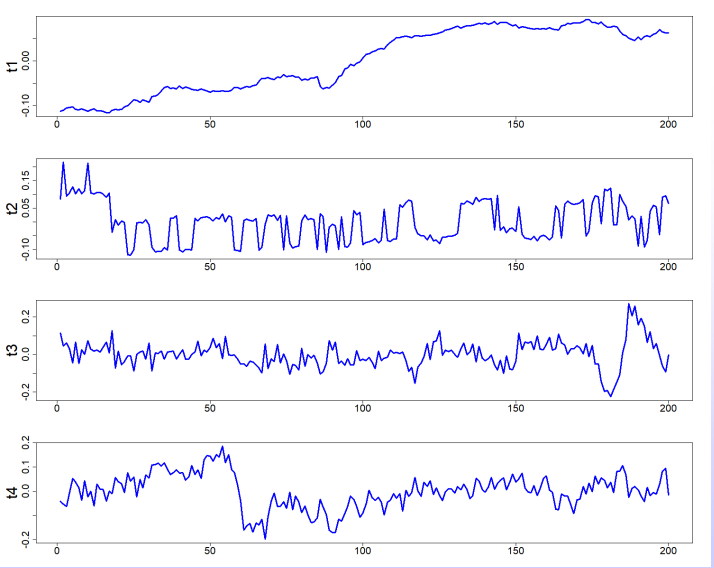
Boiler Example: Scaled data



Boiler Example for PLS

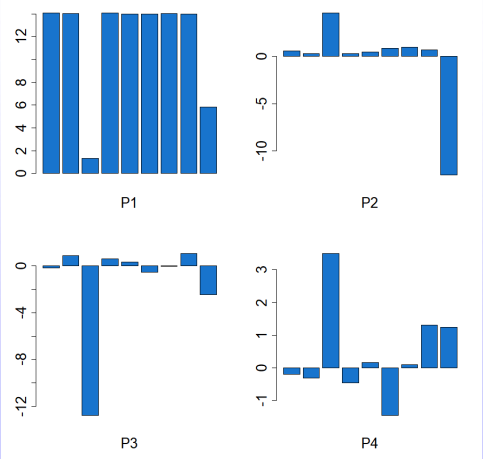
- Using the first 200 data points and all input variables to model NOx
  - Show the scores plots for first 4 factors
  - Show the loadings plots for first 4 factors
  - Use Data [201:350] as test set. Plot the test error of NOx vs. the number of factors
  - Determine the number of factors to use
  - Use the selected number of factors, predict data in [1:200], [201:350] and [351:end].

PLS Scores – Boiler Data



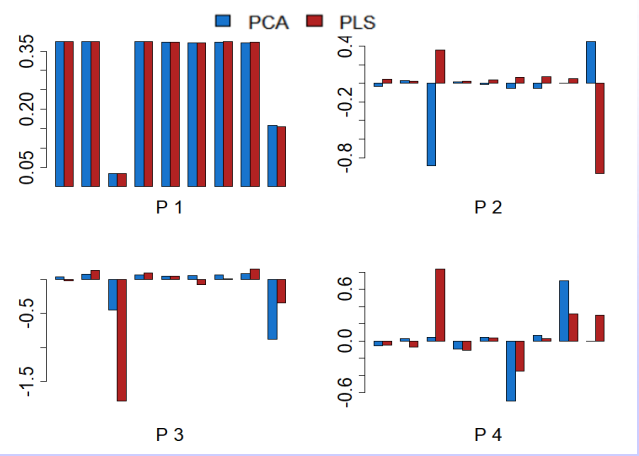
PLS Loadings – Boiler Data

Variable Names		
1. AirFlow	2. Fuel flow	3. Stack Oxygen
4. Steam Flow	5. Economizer Inlet Temp	6. Stack Pressure
7. Windbox Pressure	8.Feedwater Flow	9. Ambient Temp

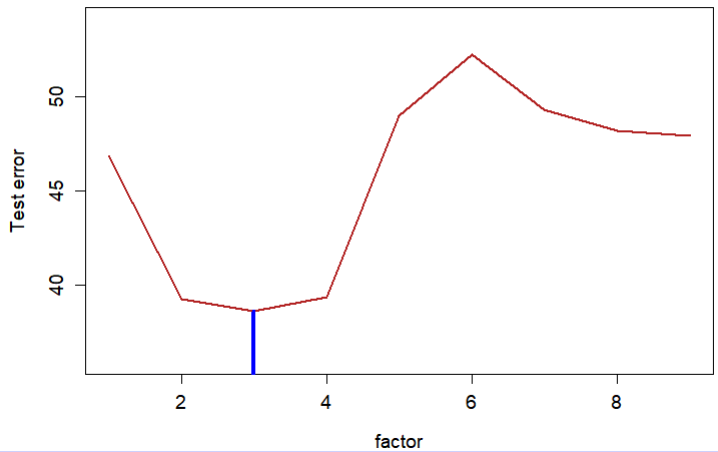


PLS differs from PCA loadings

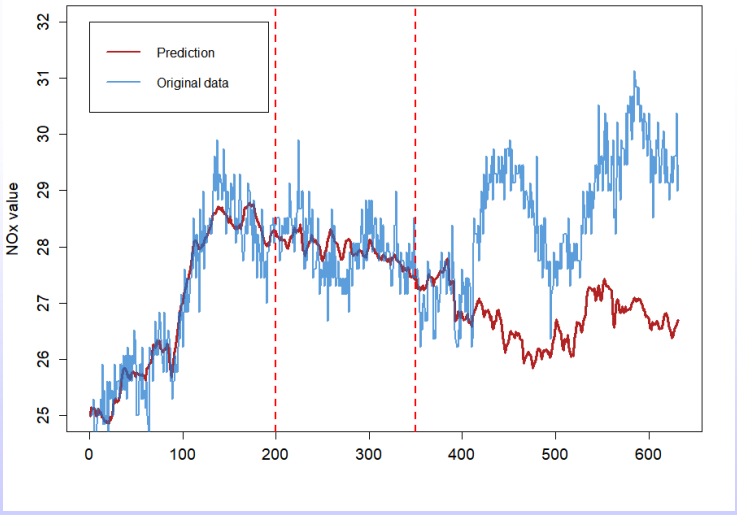
- The first PC for the boiler data is similar, but the other PCs differ more
- Sign differences don't matter



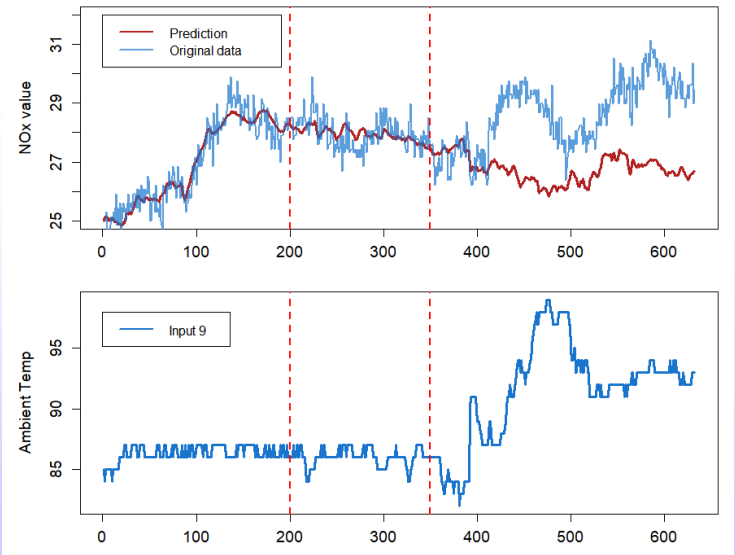
Test Error (PRESS if cross-validated)



PLS prediction



Diagnosis (ambient temp not air-conditioned; data sampled at 5 min sampling rate)



PLS Inner regression and correlation coefficients

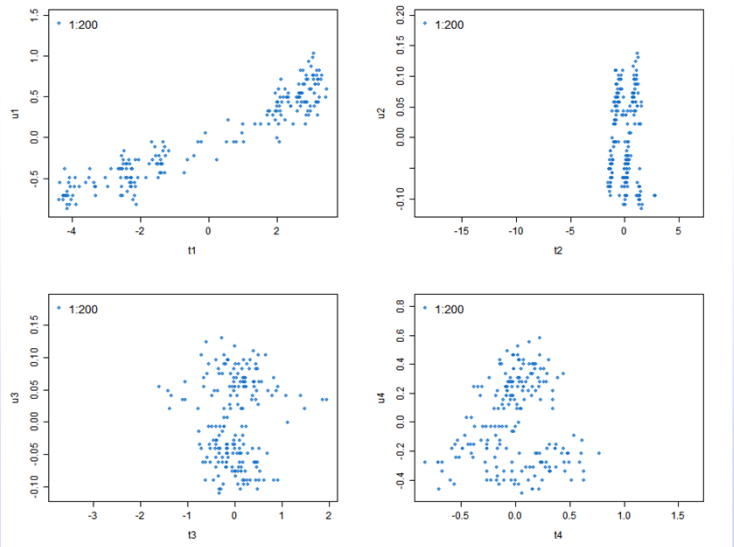
- Inner regression coefficients (b_i):

0.5234	0.0928	0.0888
0.3999	1.4084	0.4440
1.2486	0.5954	0.2869

- Inner correlation coefficients

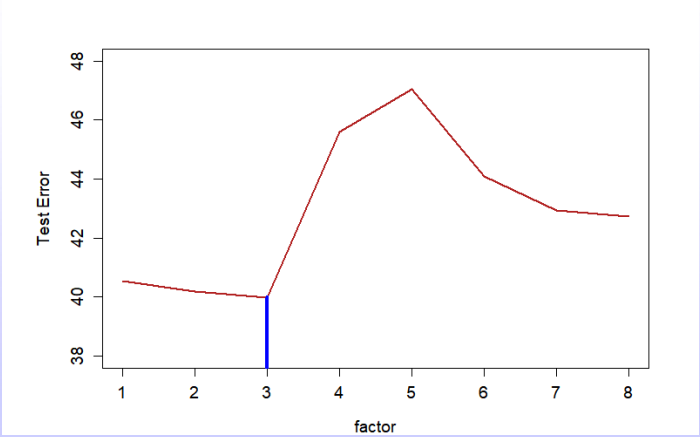
0.9464	0.1820	0.0983
0.2599	0.1992	0.0953
0.1492	0.0445	0.0293

PLS Inner relations –visualization and diagnosis: many points in [351:end] extrapolated



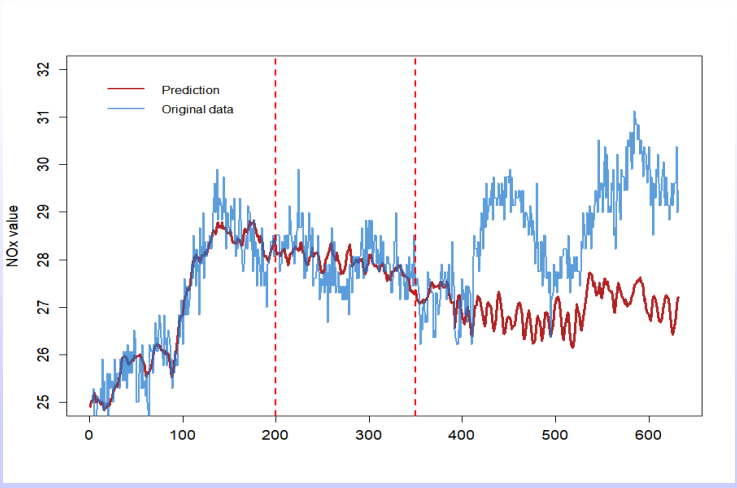
What if we remove input 9 (ambient temp) since it has little variation when A/C is on, and redo the PLS model?

- Again three factors are needed.



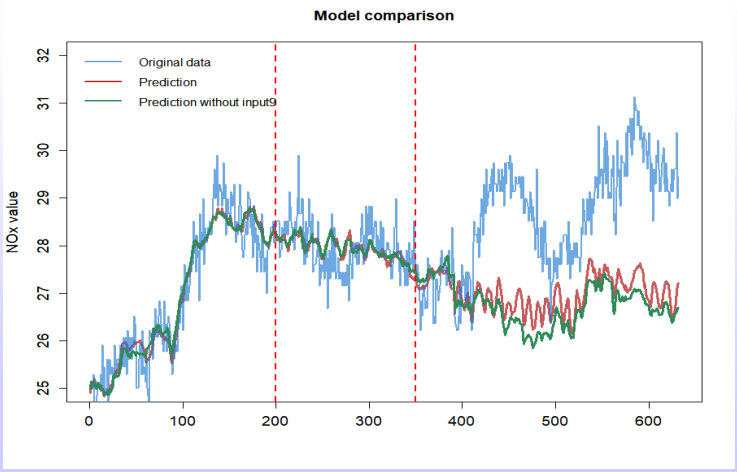
PLS prediction (without Input 9)

We still see that the model is incapable of predicting the excursions in NOx.



Comparison of two PLS models

- No much difference from using 9 inputs.
- Training data need to cover all ranges. [e.g., add the last 100 points in training]

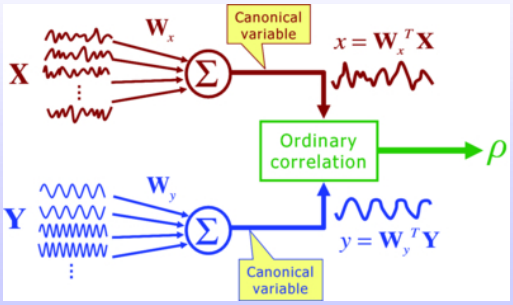


OUTLINE

- Principal component analysis (PCA)
  - History
  - Objective
  - [Workshop]
- Partial least squares (PLS)
  - Performs regression with collinearity in the input variables
  - Objective
  - [Workshop]
- What is canonical correlation analysis (CCA)?

Canonical correlation analysis (CCA)

- Again, it's due to Hotelling!
  - Hotelling, H., 1936: Relations between two sets of variants. Biometrika, 28, 321-377.
- Basic idea: maximizing correlation



Canonical correlation analysis (CCA)

- Projections:

$$\mathbf{X}_i \mathbf{w}_i = \mathbf{t}_i \qquad \mathbf{Y}_i \mathbf{q}_i = \mathbf{u}_i$$

- **CCA objective:** maximizing correlation

$$\max \frac{\mathbf{t}_i^T \mathbf{u}_i}{\|\mathbf{t}_i\| \|\mathbf{u}_i\|} \quad \text{where } \mathbf{t}_i = \mathbf{X}\mathbf{w}_i \text{ and } \mathbf{u}_i = \mathbf{Y}\mathbf{q}_i$$

- Solution:

$\mathbf{w}_i$ is an eigenvector of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{X}$

$\mathbf{q}_i$ is an eigenvector of $\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$

CCA vs. PLS

- Unlike PLS, CCA does not need to deflate data matrices. It is a (generalized) eigenvector solution
- Since CCA involves inverting matrices, it suffers from collinearity as least squares does
- The solution is problematic when the number of data points is less than the number of variables
  - Note: PLS was invented later (1977, 1982) by H. Wold

H. Wold, in A Second Generation of Multivariate Analysis, ed. by C. Fornell, p. 325, Preager, New York (1982)

More on CCA vs. PLS

- The number of non-zero CCA components is equal to the smaller dimension of the input and output variables.
  - When there is only one output, CCA needs one component only. Therefore,
  - CCA does not explore the X covariance structure as PLS does.
  - CCA is not recommended to use for process monitoring of X; use CCA-PCA for concurrent monitoring (Zhu et al., 2017)
- CCA can be used as inferential sensors to predict Y, but regularization is needed

Qinqin Zhu et al. (2017). Concurrent Quality and Process Monitoring with Canonical Correlation Analysis. *Journal of Process Control*, 60, 95-103.

SUMMARY

- PCA is a dimension reduction tool that extracts **features** with the most variance
  - Unsupervised learning
  - Visualization of loadings and scores with biplots
- PLS maximizes covariance between two sets of variables
  - Robust to collinearity; supervised learning
  - Needs more than one factor for one output
- CCA maximizes correlation between two sets of variables
  - Suffers from collinearity; supervised learning
  - Needs only one factor for one output

[WORKSHOP] on PLS and CCA

- PLS - Model on boiler data [1:350]
  - How many factors do you need? [hint: cross-validation]
  - Plot inner relations for the first four factors
  - How good are the predictions on [351:end]?
- CCA - Model on boiler data [1:350]
  - How many factors can you extract? [hint: only one since there is only one output.]
  - How good are the predictions on [351:end]?

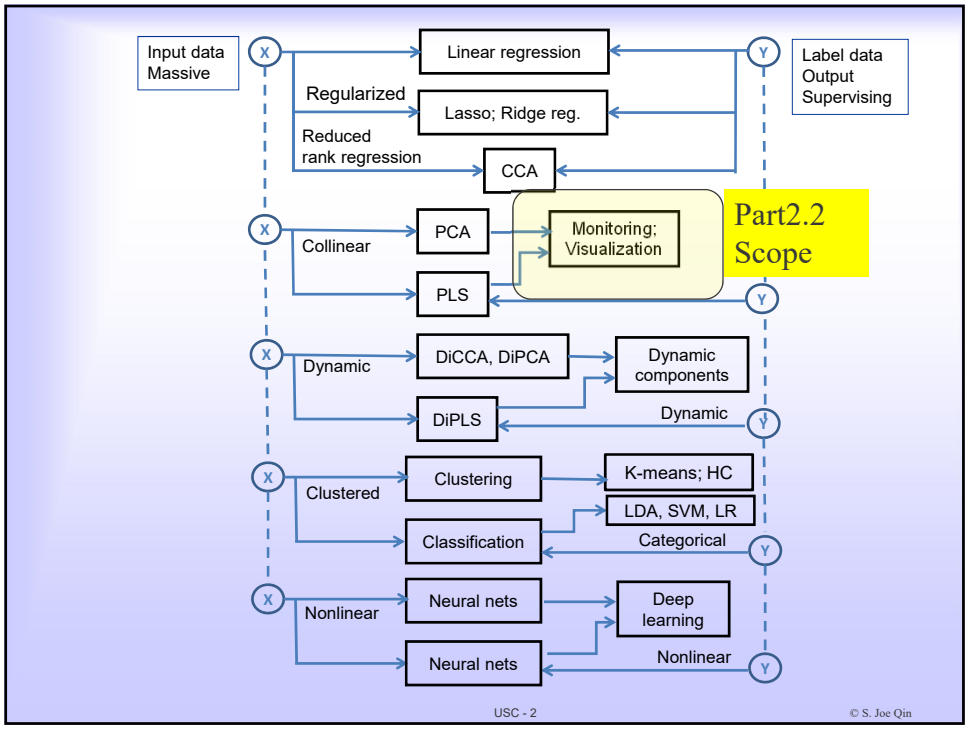
MODULE 2 – PART 2

PROCESS MONITORING USING PRINCIPAL COMPONENT ANALYSIS

S. Joe Qin

For the FOPAM Workshop in collaboration with Leo Chiang and Richard Braatz

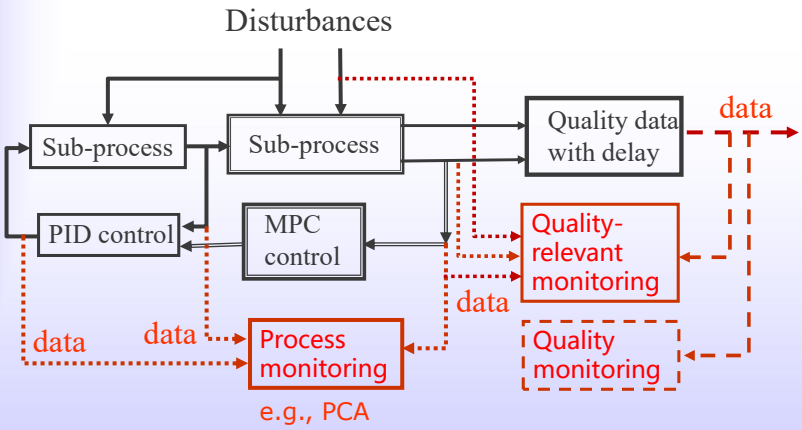
Reference: S.J. Qin (2003). Statistical process monitoring: basics and beyond, *J. Chemometrics*, 17, 480-502.



OUTLINE

- Process Monitoring
- Fault detection
  - Squared prediction error (SPE) or Q
  - T^2 index
- Fault Contribution Analysis

Multi-rate Control and Monitoring

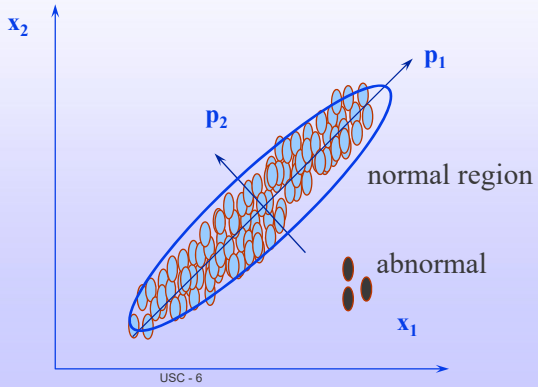


Characteristics of Process Operations

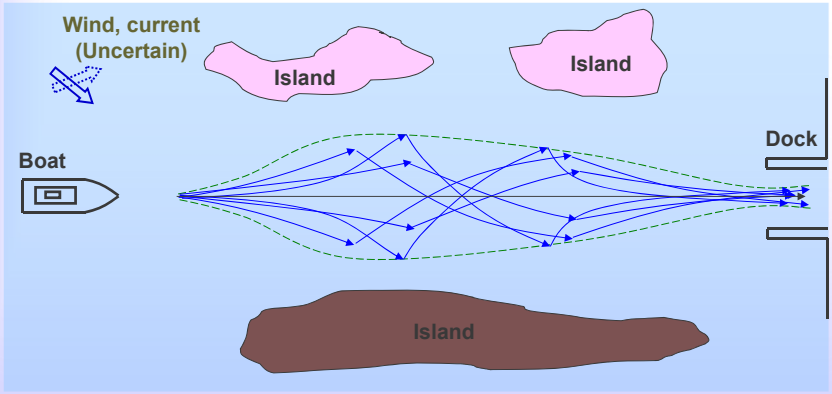
- Data rich, with various time scales and sources
- Multi-level objectives for control and operations co-exist
- Soft, operational faults happen more often and costly
- Few operators responsible for large operations

Statistical Process Monitoring

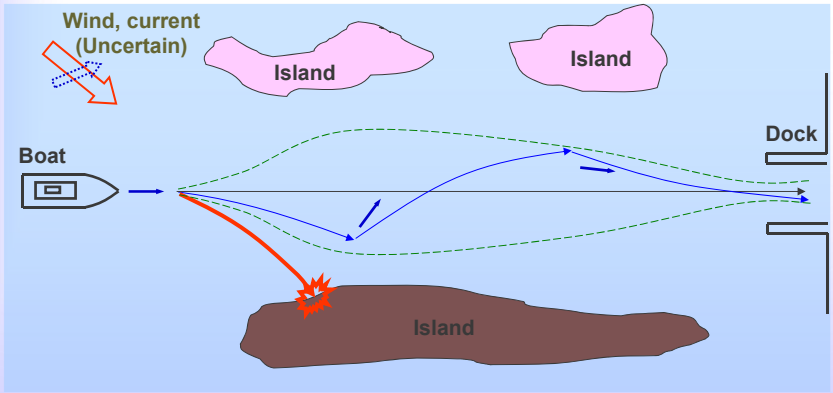
- Use PCA to extract principal components residuals from **normal** data.
- Derive 'normal' control regions for the principal component and residual spaces.
- Detect and diagnose anomalies in new data



Data-driven Methods - A Simple Illustrative Example



Fault Detection - A Motivating Example



PCA Scores per Observation - recap

- A sample at *k* is mapped to the score space:

$$\mathbf{t}_k = \mathbf{P}^T \mathbf{x}_k \qquad (\text{similarly, } \tilde{\mathbf{t}}_k = \tilde{\mathbf{P}}^T \mathbf{x}_k)$$

- PCA projection for the k-th sample:

$$\hat{\mathbf{x}}_k = \mathbf{P}\mathbf{t}_k = \mathbf{P}\mathbf{P}^T \mathbf{x}_k$$

- PCA residual:

$$\tilde{\mathbf{x}}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k = (\mathbf{I} - \mathbf{P}\mathbf{P}^T)\mathbf{x}_k$$

$$\because \quad \begin{bmatrix} \mathbf{P} & \tilde{\mathbf{P}} \end{bmatrix} \begin{bmatrix} \mathbf{P} & \tilde{\mathbf{P}} \end{bmatrix}^T = \mathbf{P}\mathbf{P}^T + \tilde{\mathbf{P}}\tilde{\mathbf{P}}^T = \mathbf{I}$$

$$\therefore \quad \tilde{\mathbf{x}}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k = (\mathbf{I} - \mathbf{P}\mathbf{P}^T)\mathbf{x}_k = \tilde{\mathbf{P}}\tilde{\mathbf{P}}^T \mathbf{x}_k$$

USC - 9

PCA based Fault Detection

- PC scores and residuals are monitored with different statistics
- Hotelling's T^2: monitor the PC scores for the k-th observation,

$$T_k^2 = \mathbf{t}_k^T \Lambda^{-1} \mathbf{t}_k = \mathbf{x}_k^T \mathbf{P} \Lambda^{-1} \mathbf{P}^T \mathbf{x}_k = \sum_{i=1}^{l} \frac{t_{ki}^2}{\lambda_i}$$

Λ contains the major eigenvalues of Cov(x).

- Q index, a.k.a. squared prediction error, SPE: monitor the residuals,

$$Q_k = \tilde{\mathbf{t}}_k^T \tilde{\mathbf{t}}_k = \mathbf{x}_k^T \tilde{\mathbf{P}}\tilde{\mathbf{P}}^T \mathbf{x}_k = \mathbf{x}_k^T (\mathbf{I} - \mathbf{P}\mathbf{P}^T)\mathbf{x}_k$$
$$= \tilde{\mathbf{x}}_k^T \tilde{\mathbf{x}}_k$$

USC - 10

Hotelling's T² Control Limit

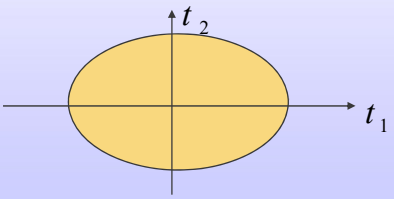
Assuming $\mathbf{t} \sim \mathbb{N}(\mathbf{0},\boldsymbol{\Lambda})$, the $(1-\alpha)\times 100\%$ control limit is

$$T_k^2 = \mathbf{t}_k^T \boldsymbol{\Lambda}^{-1} \mathbf{t}_k = \sum_{i=1}^{l} \frac{t_{ki}^2}{\lambda_i} \le \tau_\alpha^2 = \chi_\alpha^2(l)$$

For example, $\mathbf{t}_k^T = \begin{bmatrix} t_1 & t_2 \end{bmatrix}, \boldsymbol{\Lambda} = diag\{\lambda_1 \quad \lambda_2\}$,

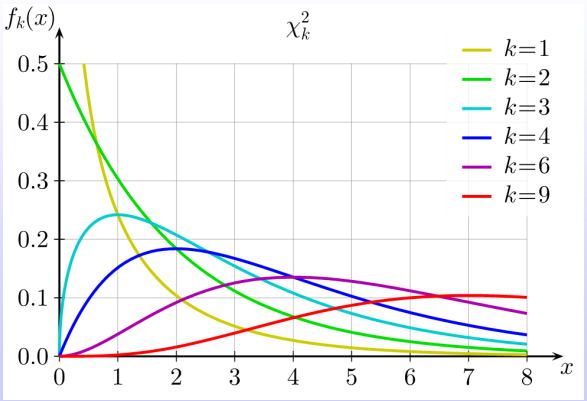
$$T_k^2 = \frac{t_1^2}{\lambda_1} + \frac{t_2^2}{\lambda_2} \le \tau_\alpha^2$$

which is an ellipse.



Chi-square distribution

The chi-square distribution (χ2-distribution) with k degrees of freedom is the distribution of a sum of the squares of k independent standard normal random variables.



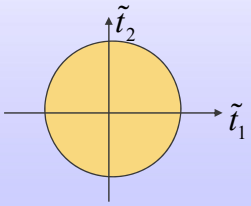
A Simple SPE Control Limit

Assuming $\tilde{\mathbf{t}} \sim \mathbb{N}(\mathbf{0}, \tilde{\mathbf{\Lambda}})$, the SPE control limit with $(1\text{-}\alpha)$ confidence is

$$\text{SPE} = \tilde{\mathbf{t}}^T\tilde{\mathbf{t}} = \tilde{\mathbf{x}}^T\tilde{\mathbf{x}} \le \delta_\alpha^2, \quad \text{where } \delta_\alpha^2 = g\chi_\alpha^2(h)$$

$$g = \frac{\sum_{i=l+1}^{M}\lambda_i^2}{\sum_{i=l+1}^{M}\lambda_i} \; ; \; h = \frac{(\sum_{i=l+1}^{M}\lambda_i)^2}{\sum_{i=l+1}^{M}\lambda_i^2} \text{ round up to an integer}$$

Note that $\tilde{\mathbf{t}}^T\tilde{\mathbf{t}} = \delta_\alpha^2$ is a circle.



USC - 13

Box Theorem for Control Limits

Box (1954) gives an approximate distribution

Assuming $\mathbf{x}$ is multi-normal, $\mathbf{x}^T\mathbf{M}\mathbf{x}$ approx. $\sim g\chi^2(h)$, and the control limit with $(1\text{-}\alpha)$ confidence is

$$\mathbf{x}^T\mathbf{M}\mathbf{x} \le \delta_\alpha^2, \quad \text{where } \delta_\alpha^2 = g\chi_\alpha^2(h)$$

$$g = \frac{tr(\mathbf{RM})^2}{tr(\mathbf{RM})} \quad \text{where } \mathbf{R} = \text{cov}(\mathbf{x})$$

$$h = \frac{[tr(\mathbf{RM})]^2}{tr(\mathbf{RM})^2}$$

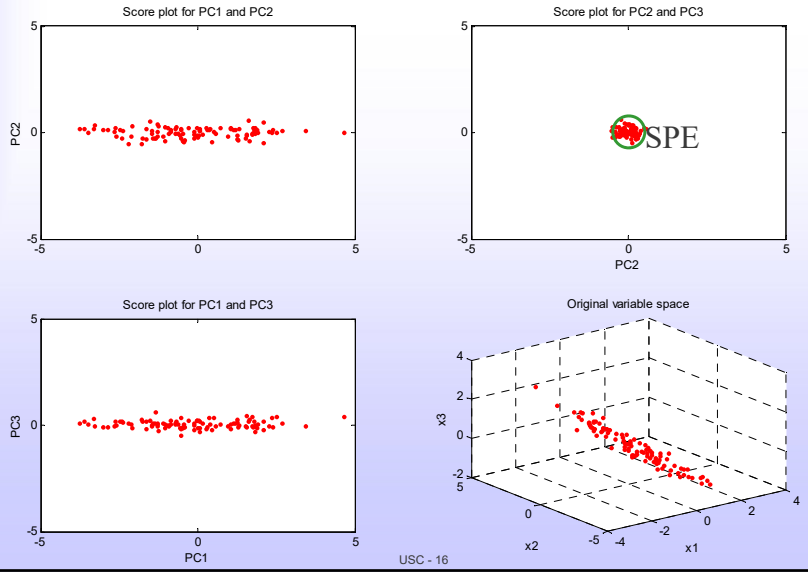
USC - 14

Different treatments of SPE and T2

- The variances of the PC scores are large and different from one another
  - The T2 index control region is an ellipse; it involves inverses of the PC variances
- The variances of the residual scores are tiny and some are close to zero
  - T2 index would invert the tiny variance, making it too sensitive to uncertainty in the variance estimation.
  - Therefore, no inverse is used in SPE.

Example: -x1=x2=x3 plus noise:

SPE control region is often tight, more sensitive to faults



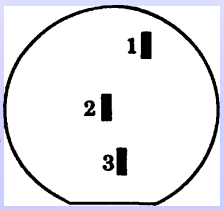
Summary of PCA for Fault Detection

- A sample of normal data with good coverage of the normal variations
  - Pre-processing, missing data and outlier treatment, scaling, etc., are necessary
- Perform PCA with a chosen number of PCs
- Calculate control limits for SPE, T2
- For new observations, use control charts to detect faults
- Fault diagnosis follows after detecting a fault

[DEMO] T2 region is an ellipse

- Use the wafer critical dimension (CD) data for 3 sites, from M. Joshi and K. Sprague (1997). It's a photo process in semiconductor manufacturing. We perform
  - PCA – preliminary analysis using all data
  - Recalculate control limits using normal data
  - Show control charts after deleting bad wafers
  - Show contribution analysis for wafers detected with T2 and SPE

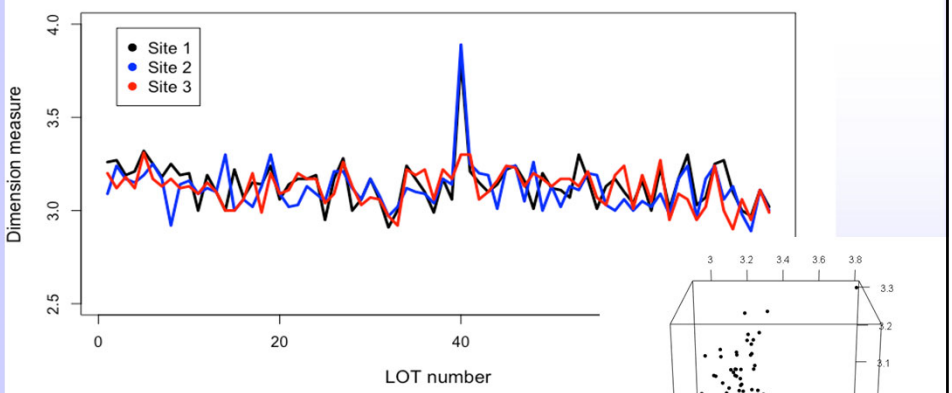
Data Sites on a Wafer

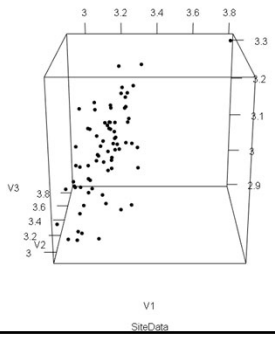


The Photo Problem

- In mid-80's, managers at DEC asked process engineers to bring SPC to semiconductor manufacturing. When engineers tried using the text book approach, the results were disastrous: too many charted points outside of the apparent SPC limits
- Semiconductor manufacturing has numerous batch processes. Consequently, statistical methods listed in the vast majority of SPC publications did not apply directly.

Variables (CD) vs. sample number

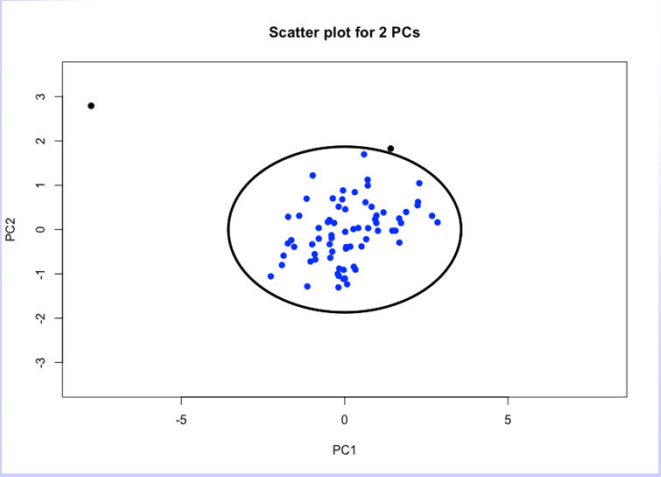




PCA – preliminary analysis

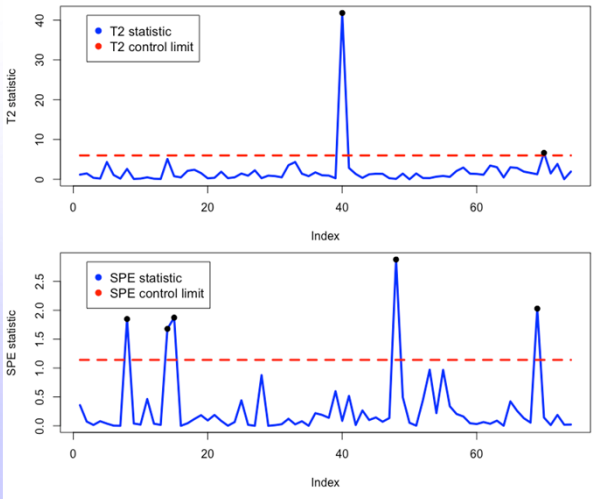
Number of PC is 2 (90% variance)

T2 plot for 2 PCs



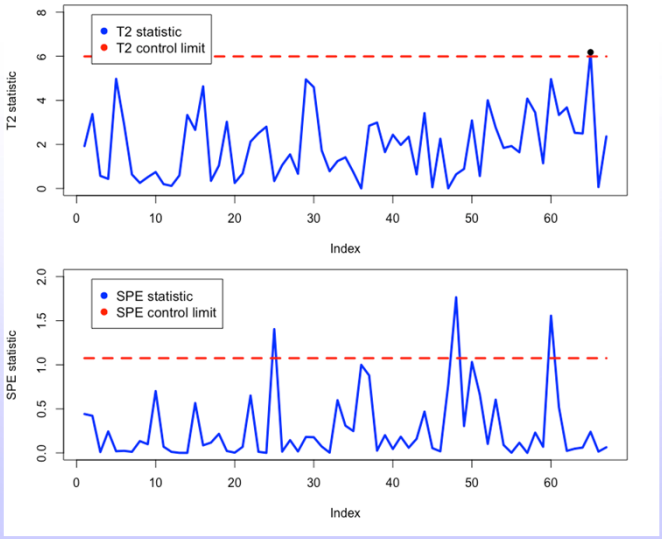
T2 and SPE control charts

The control limits are OK, but we should recalculate them using normal data only



Only 1 PC in the SPE

Recalculate T2 and SPE control limits after removing bad wafers



OUTLINE

- Recap of PCA
- Fault detection
  - Squared prediction error (SPE)
  - T2 index
  - Combined index
- **Fault Contribution Analysis**

SPE Contribution Plots

- **SPE contributions:**

$$\text{Since } SPE = \tilde{\mathbf{x}}^T\tilde{\mathbf{x}} = \sum_{i=1}^{M} \tilde{x}_i^2,$$

$$SPE_i = \tilde{x}_i^2$$

is the contribution to SPE from the $i\text{-}th$ variable

- Variables with large contributions are likely the causes of the fault
- Contributions 'smear' a fault in one variable to other variables, but are often effective
- Contributions are sensitive to scaling, so it is important to scale variables appropriately

Contributions for T-square

- **T2 contribution is not easy to define**
  - Nomikos (1997) defines a T2 contribution that expands one side of the index only,

$$T^2 = \mathbf{x}^T\mathbf{P}\Lambda^{-1}\mathbf{P}^T\mathbf{x} = \sum_{i=1}^{M} \underbrace{x_i\mathbf{p}_i^T\Lambda^{-1}\mathbf{P}^T\mathbf{x}}_{T_i^2}$$

T_i^2 is defined as the contribution of Variable x_i.

$\mathbf{p}_i^T$ is the i-th row of $\mathbf{P}$, $i = 1,\ldots,M$.

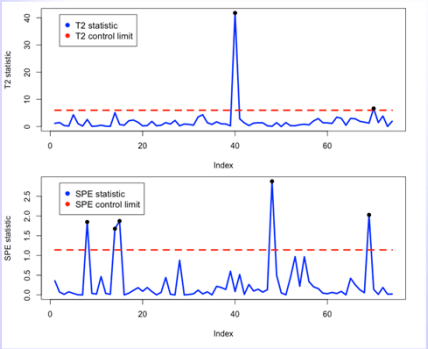
- **Reconstruction-based contributions (RBC) is an alternative (Alcala and Qin, 2009)**

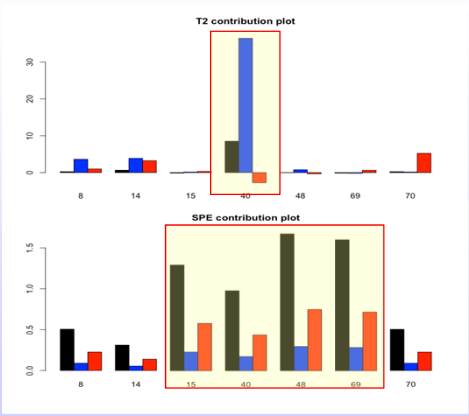
Carlos Alcala and S. Joe Qin (2009). Reconstruction-based Contribution for Process Monitoring, Automatica, 45, 1593-1600.

Example: T2 and SPE contributions

T2 out of control wafer: 40 – Site 2 is the problem

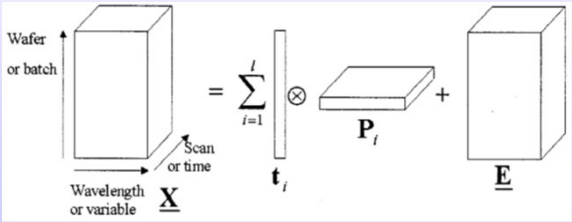
SPE OOC wafers: 15, 40, 48, 69 – Site 1 stands out in the residuals





Multi-way or batch data monitoring

- **Unfold to matrices to perform PCA**
  - **Be aware that the measurement dimension is very high after unfolding, requiring a large number of PCs**
  - **The confidence level is usually chosen very high accordingly**



PCA for Monitoring

- Check the link

https://weiusc.shinyapps.io/contributions-reconstruction-8/

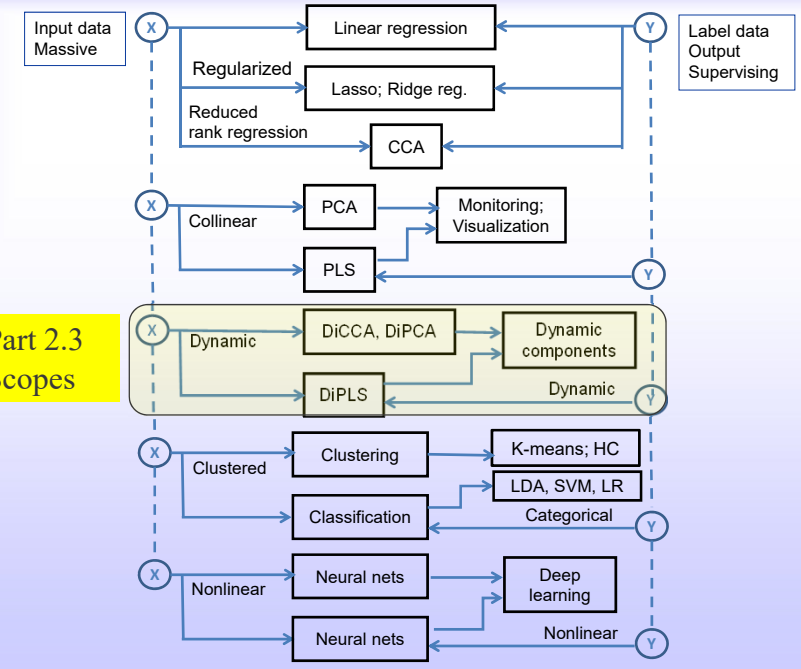
Module 2 – Part 3

DYMANIC COMPONENT MODELING FOR MULTIVARIATE TIME SERIES – Dynamic PCA, PLS, CCA

S. Joe Qin

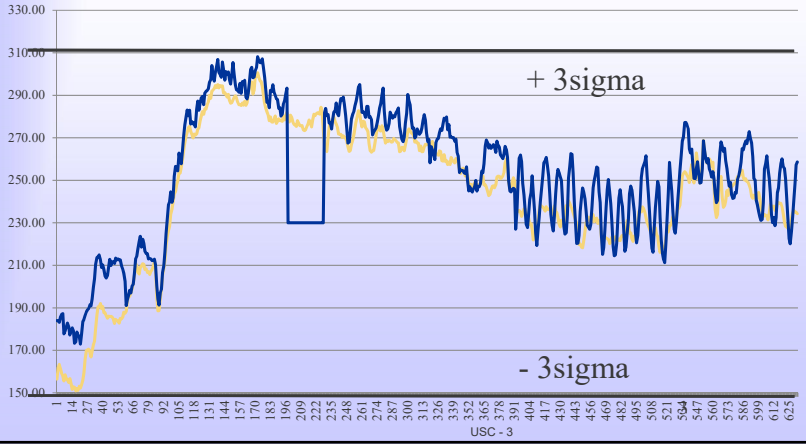
For the FOPAM Workshop in collaboration with Leo Chiang and Richard Braatz

Part 2.3 Scopes



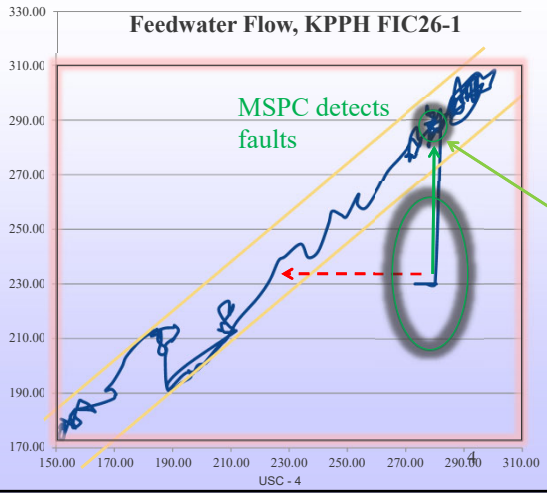
Detect faults in dynamic data

- PCA based monitoring can detect a fault that breaks cross-correlation.
- dynamic time-correlation can reveal more information



Dynamic data: identify causes

- Dynamic PCA extracts auto-correlations



Univariate time series modeling

Auto-regressive (AR) model

$x_k = \sum_{i=1}^{s} a_i x_{k-i} + e_k$; where $\{e_k\}$ is white noise with variance σ^2.

We define $x_{k-i} = q^{-i} x_k$ using a backward shift operator,

the AR model is

$x_k = \sum_{i=1}^{s} a_i q^{-i} x_k + e_k$; or $(1 - \sum_{i=1}^{s} a_i q^{-i}) x_k = e_k$

which means the series $\{x_k\}$ is filetered achieve white noise $\{e_k\}$

or $x_k = (1 - \sum_{i=1}^{s} a_i q^{-i})^{-1} e_k$

which means the series $\{x_k\}$ is 'driven by' white noise $\{e_k\}$

Auto-regressive moving average (ARMA) model

$x_k = \sum_{i=1}^{s} a_i x_{k-i} + c_i e_{k-i} + e_k$;

where $\{e_k\}$ is white noise with variance σ^2.

Using the backward shift operator, the ARMA model is

$(1 - \sum_{i=1}^{s} a_i q^{-i}) x_k = (1 - \sum_{i=1}^{s} c_i q^{-i}) e_k$

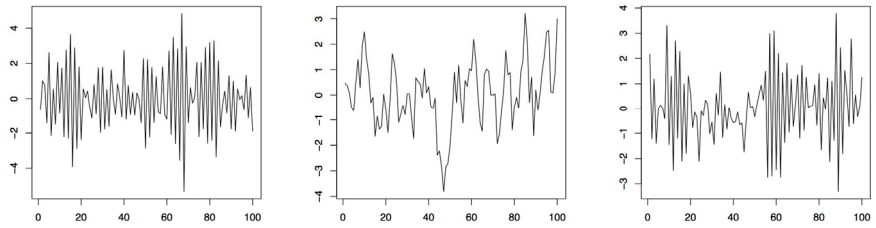
or $x_k = (1 - \sum_{i=1}^{s} a_i q^{-i})^{-1} (1 - \sum_{i=1}^{s} c_i q^{-i}) e_k$

which still means the series $\{x_k\}$ is 'driven by' $\{e_k\}$

ARIMA function in R

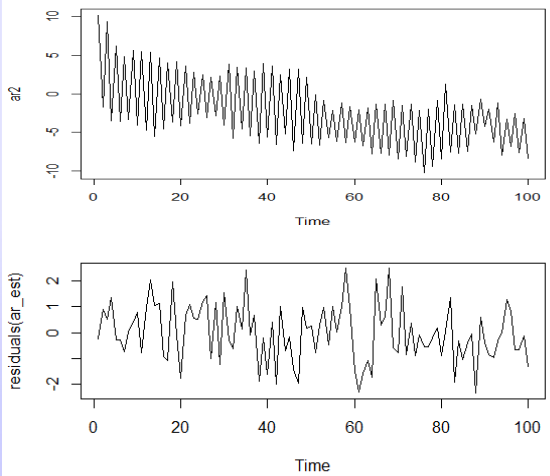
- Realizations of three autoregressive processes, two AR(1) and one AR(2)

```
>ar1neg = arima.sim(list(order=c(1,0,0), ar=-.9), n=100)
>ar1pos = arima.sim(list(order=c(1,0,0), ar=.8), n=100)
>ar2 = arima.sim(list(order=c(2,0,0), ar=c(-.5,.3)), n=100)
```



An example: a dynamic signal and its residual

```
> ar2 = arima.sim(list(order=c(2,0,0), ar=c(.0,.99)), n=100)
> plot(ar2)
> ar_est = arima(ar2,order=c(2,0,0))
> plot(residuals(ar_est))
```

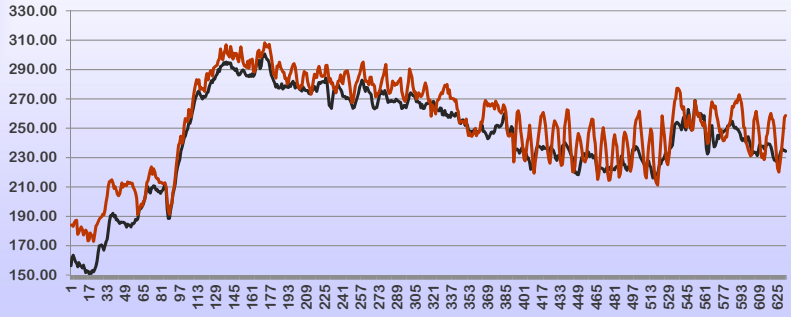


Dynamic time series has predictable content

The residuals are white and appropriate for monitoring

Dynamic Component Modeling

- With auto-correlated time series , extract a latent variable that is most auto-correlated, or predictable
- The latent variable is a linear combination of the original variables

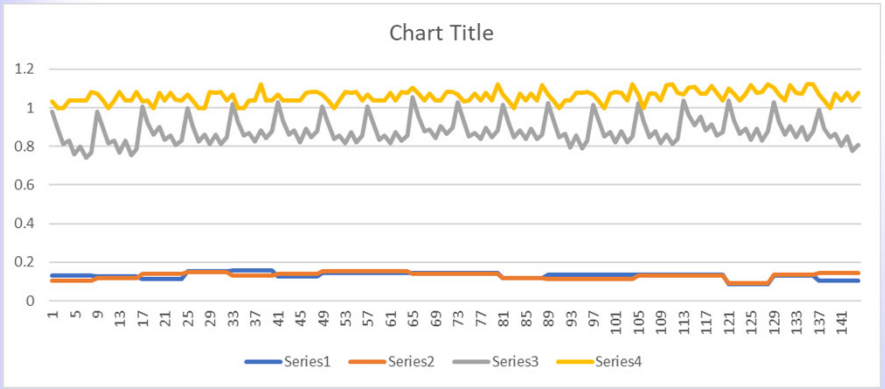


9

USC - 9

Example: Oxide thickness data

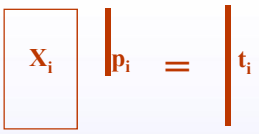
- Multiple scans per wafer



USC - 10

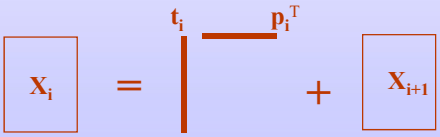
Principal component analysis focuses on variance in the component only

Find a projection to maximize the variance of the scores



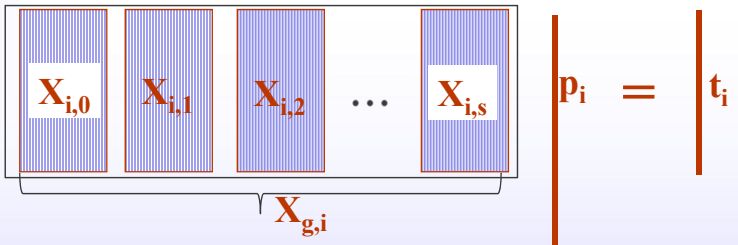
$$\max \mathbf{t}_i^T \mathbf{t}_i \quad \text{where } \mathbf{t}_i = \mathbf{X}_i \mathbf{p}_i \quad \text{and} \quad \mathbf{p}_i^T \mathbf{p}_i = 1$$

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{t}_i \mathbf{p}_i^T$$



Dynamic PCA: Simple augmentation

- Augment X with lagged measurements



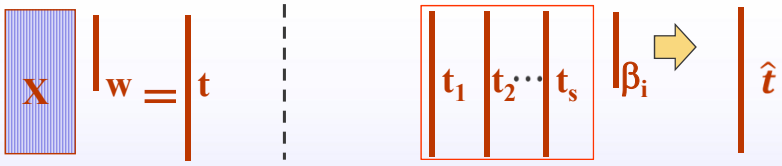
- Perform PCA on $X_{g,i}$
- Dimension increases dramatically
- The PC scores have the most variance, but not necessarily dynamic

* Ku, W., Storer, R., & Georgakis, C. (1995). Disturbance detection and isolation by dynamic principal component analysis. Chemometrics and Intelligent Laboratory Systems, 30, 179.

Dynamic inner PCA (DiPCA)

(Dong and Qin, 2016)

Linearly combine the measurements, then augment the past scores to predict the future



- **DiPCA objective**

  The extracted dynamic t scores is most predicted by its past values in terms of

$$\max \mathbf{t}^T\hat{\mathbf{t}} \quad \text{where } \mathbf{t}=\mathbf{X}\mathbf{w}, \qquad \hat{\mathbf{t}} \Leftarrow AR(\mathbf{t}_1,\ldots,\mathbf{t}_s,\boldsymbol{\beta})$$

$$\text{subject to} \quad \mathbf{w}^T\mathbf{w}=1 \ \text{ and } \ \boldsymbol{\beta}^T\boldsymbol{\beta}=1$$

Calculate the residual and extract the next component

Dynamic Component Modeling

- **High dimensional data are driven by a few dynamic components, which are not measured directly.**
- **DiPCA extracts the most dynamic , i.e., most predictable components from data, one after another**
  - **DiPCA distills the multi-dimensional data into dynamic components, in descending order of predictability**

Extensions to Dynamic PLS and CCA

- Initializing: $\mathbf{X}_1 := \mathbf{X}$ $\mathbf{Y}_1 := \mathbf{Y}$
- Projections:

- **PLS objective:**

$$\max \mathbf{t}_i^T \mathbf{u}_i \quad \text{where } \mathbf{t}_i = \mathbf{X}_i \mathbf{w}_i \quad \text{and} \quad \mathbf{u}_i = \mathbf{Y}_i \mathbf{q}_i$$

- Residuals:

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{t}_i \mathbf{p}_i^T$$
$$\mathbf{Y}_{i+1} = \mathbf{Y}_i - b_i \mathbf{t}_i \mathbf{q}_i^T$$

Inner model

*Note: cannocial correlation analysis does

$$\max \frac{\mathbf{t}_i^T \mathbf{u}_i}{\|\mathbf{t}_i\| \|\mathbf{u}_i\|}, \quad \text{where } \mathbf{t}_i = \mathbf{X}_i \mathbf{w}_i \quad \text{and} \quad \mathbf{u}_i = \mathbf{Y}_i \mathbf{q}_i$$

USC - 15

Dynamic-inner PLS: extract an LV in Y that co-varies most with an LV in X

- Initializing: $\mathbf{X}_1 := \mathbf{X}$ $\mathbf{Y}_1 := \mathbf{Y}$
- Projections:

- **Dynamic PLS objective:**

$$\max \mathbf{u}_i^T [g_i(q^{-1}) \mathbf{t}_i]; \quad \text{where } \mathbf{t}_i = \mathbf{X}_i \mathbf{w}_i \quad \text{and} \quad \mathbf{u}_i = \mathbf{Y}_i \mathbf{q}_i$$
$$g_i(q^{-1}) = \sum \beta_{ij} q^{-j}; \quad \sum \beta_{ij}^2 = 1; \quad \|\mathbf{w}_i\| = 1; \quad \|\mathbf{q}_i\| = 1$$

- Residuals:

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{t}_i \mathbf{p}_i^T$$
$$\mathbf{Y}_{i+1} = \mathbf{Y}_i - b_i \mathbf{t}_i \mathbf{q}_i^T$$

Inner model replaced with $g_i(q^{-1}) \mathbf{t}_i$

- The order of $g_i(q^{-1})$ can vary from factor to factor

* Y. Dong, S. Joe Qin (2015). IFAC ADCHEM, Paper MoM3.5.

USC - 16

Dynamic-inner CCA (DiCCA): extract an LV in Y that is most correlated with an LV in X

Linearly combine the measurements, then augment the past scores to predict the future

- DiCCA objective

  The extracted dynamic t scores is most predicted by its past values in terms of

$$\max \frac{\mathbf{t}^T\hat{\mathbf{t}}}{\|\mathbf{t}\|\|\hat{\mathbf{t}}\|} \quad \text{where } \mathbf{t}=\mathbf{X}\mathbf{w}, \qquad \hat{\mathbf{t}} \Leftarrow AR(\mathbf{t}_1,\ldots,\mathbf{t}_s,\boldsymbol{\beta})$$

Calculate the residual and extract the next component

DiCCA (Dong and Qin, 2017), CACE paper

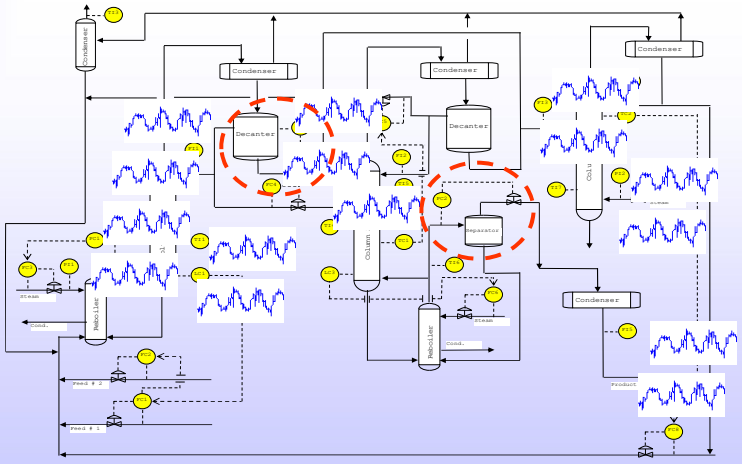
Dynamic-inner CCA (DiCCA) Algorithm

1. Initialize $\mathbf{w}$ with a column of the identity matrix.
2. Calculate $\mathbf{w}$, $\boldsymbol{\beta}$ by iterating the following relations until convergence.
   $\mathbf{t}=\mathbf{X}\mathbf{w}$ and form $\mathbf{t}_i$ from $\mathbf{t}$ according to Eq. (24).
   Form $\mathbf{T}_s=[\mathbf{t}_s \cdots \mathbf{t}_1]$.
   $\boldsymbol{\beta}=(\mathbf{T}_s^T\mathbf{T}_s)^{-1}\mathbf{T}_s^T\mathbf{t}_{s+1}$,
   $\hat{\mathbf{X}}_{s+1}=\sum_{i=1}^{s}\beta_i\mathbf{X}_{s-i+1}$,
   $\hat{\mathbf{t}}_{s+1}=\sum_{i=1}^{s}\beta_i\mathbf{t}_{s-i+1}$,
   $\mathbf{w}=\begin{bmatrix}\mathbf{X}_{s+1}\\ \hat{\mathbf{X}}_{s+1}\end{bmatrix}^{+}\begin{bmatrix}\hat{\mathbf{t}}_{s+1}\\ \mathbf{t}_{s+1}\end{bmatrix}$ and $\mathbf{w}:=\mathbf{w}/\|\mathbf{w}\|$
3. Calculate $J=\lambda=\frac{\hat{\mathbf{t}}_{s+1}^T\mathbf{t}_{s+1}}{\|\hat{\mathbf{t}}_{s+1}\|\|\mathbf{t}_{s+1}\|}$.

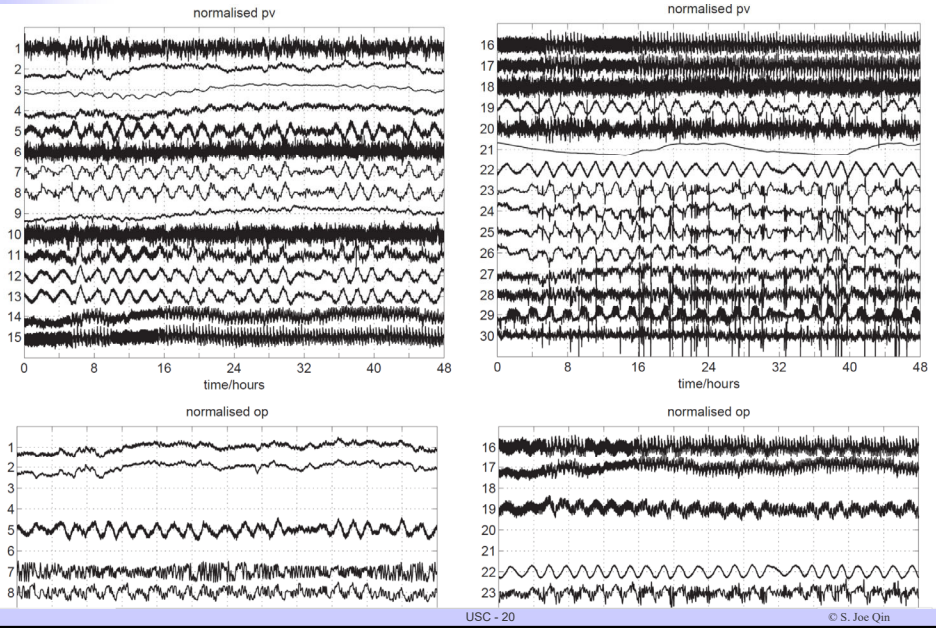
* Y. Dong and S. Joe Qin (2017). CACE

Demonstration on Eastman Data

- Oscillatory dynamics are of concern

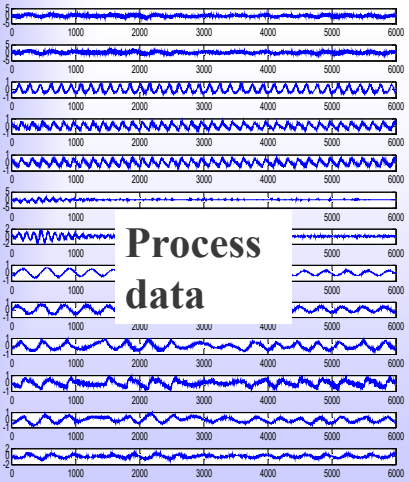


Thornhill et al. (2003)



Dynamic Data Distillation

- DiPCA/DiCCA

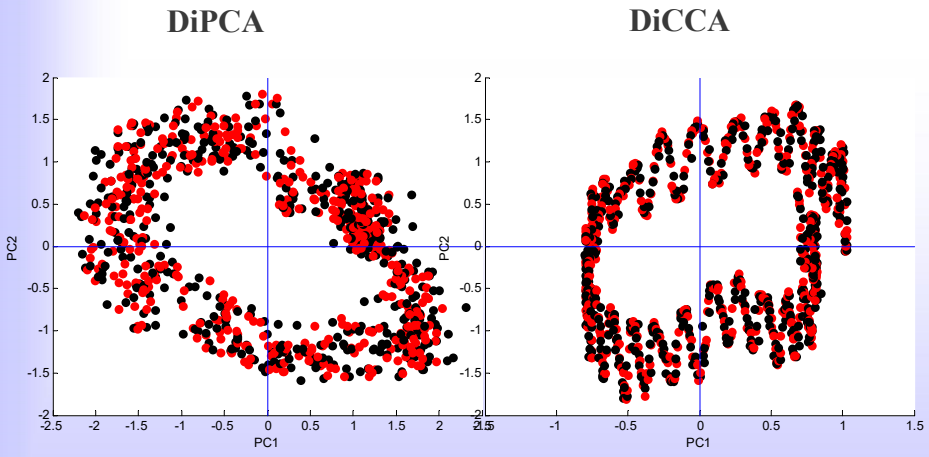


Data Distillation

- Most predictable
- 2nd most predictable
- Last predictable
- Not predictable; White noise; PCA applies

DiPCA

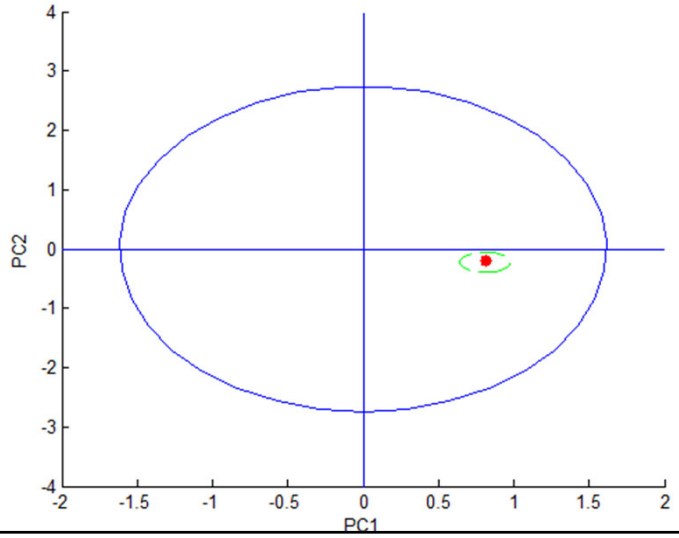
First two factors



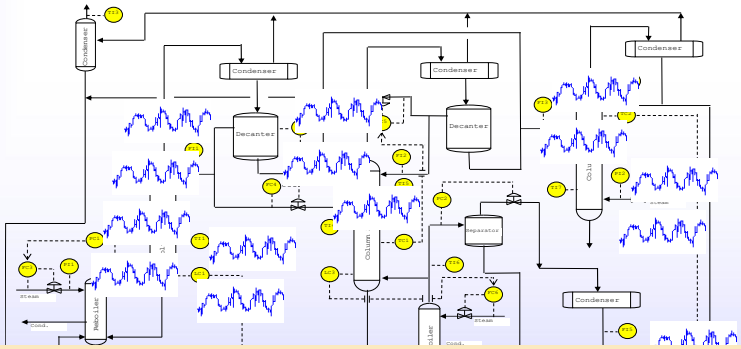
- DiCCA reveals oscillations with two distinct frequencies

DiCCA Monitoring based on Prediction

T2 limit from PCA is not suitable for the circular distribution

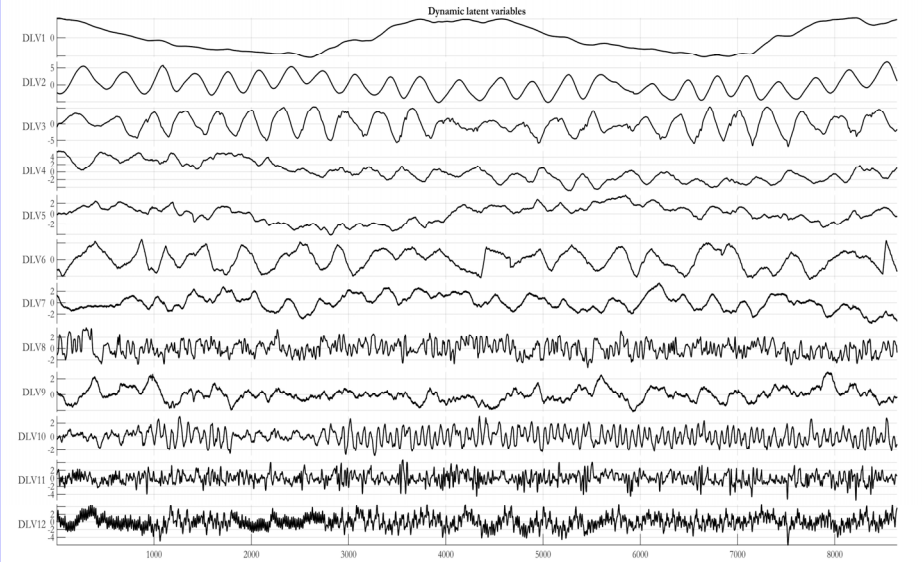


Analysis Details on Eastman Data

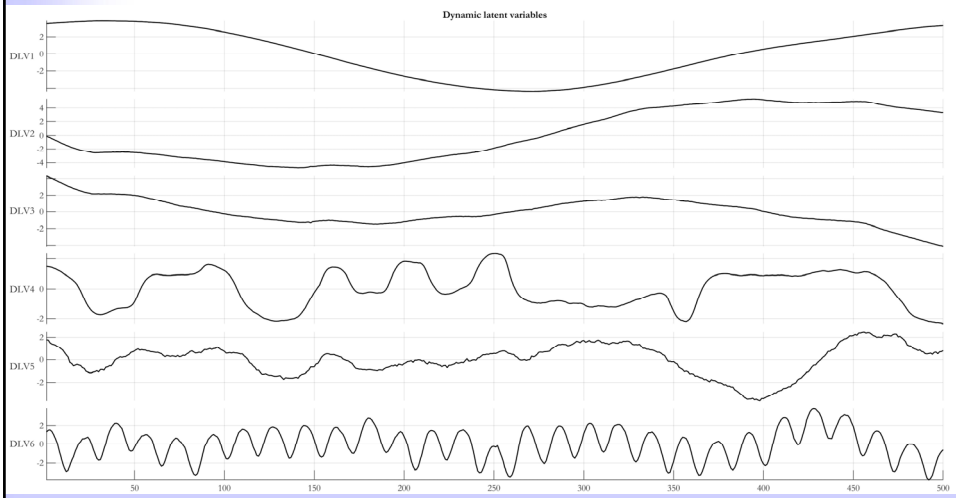


PC1.PV, PC1.OP, FC3.SP, FC3.PV, FC3.OP, TI1.PV, TI2.PV, LC1.PV, LC1.OP, FI1.PV, FC1.SP, FC1.PV, FC1.OP, FC2.PV, FC2.OP, PI1.PV, FC4.PV, FC4.OP, TI5.PV, TI4.PV, TC1.PV, TC1.OP, FC6.SP, FC6.PV, FC6.OP, TI6.PV, PC2.SP, PC2.PV, PC2.OP, LC3.PV, LC3.OP, FI2.PV, FC5.SP, FC5.PV, FC5.OP, FI5.PV, TI3.PV, LC2.PV, LC2.OP, FC8.SP, FC8.PV, FC8.OP, FI4.PV, TC2.PV, TC2.OP, TI8.PV, TI7.PV, PI2.PV, FI3.PV, FC7.OP.

Features in dynamics and time scale are separated - three features: 24 hours, 2 hours, and 6 minutes

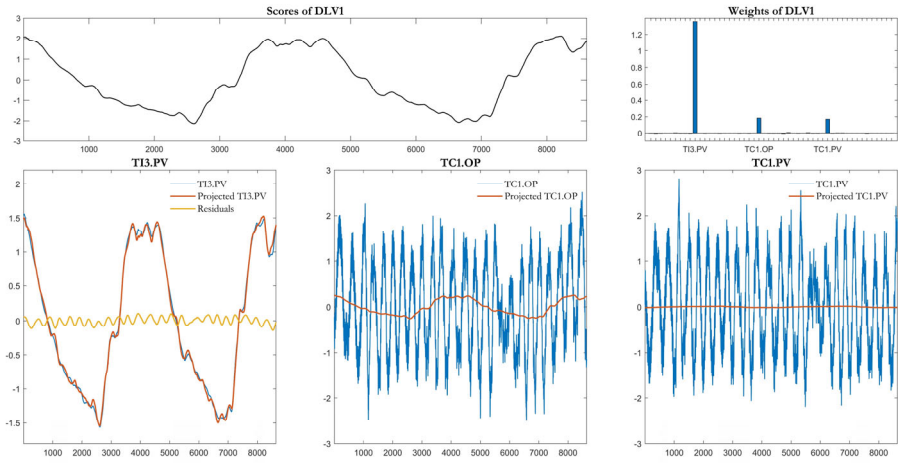


Zoom in on 500 data points (2.3 hrs)



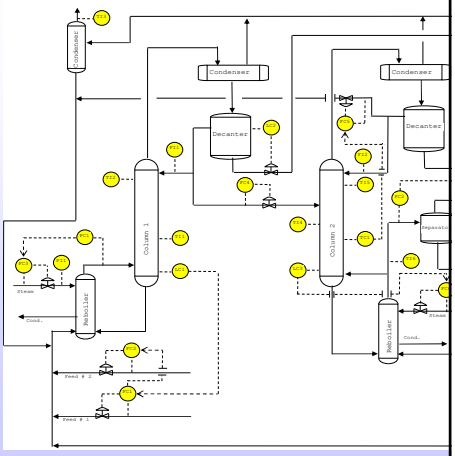
First DLV of 24 hours cycle

- Ti3 – condenser outlet temp
- TC1.OP – controller output of column 1 temp
- TC1.PV – controlled column 1 temp



Process/control loops involved in DLV1

- DLV1 is most weighted with TI3.PV, condenser outlet temperature
- TC1.OP and TC1.PV are equally weighted in DLV1, due to TC1.OP controlling the recycle flow to the condenser
- TC1.PV is weighted in to cancel large oscillations in TC1.OP, but does not have the 24 hour cycle
- The other variables have little weights



Concluding Remarks

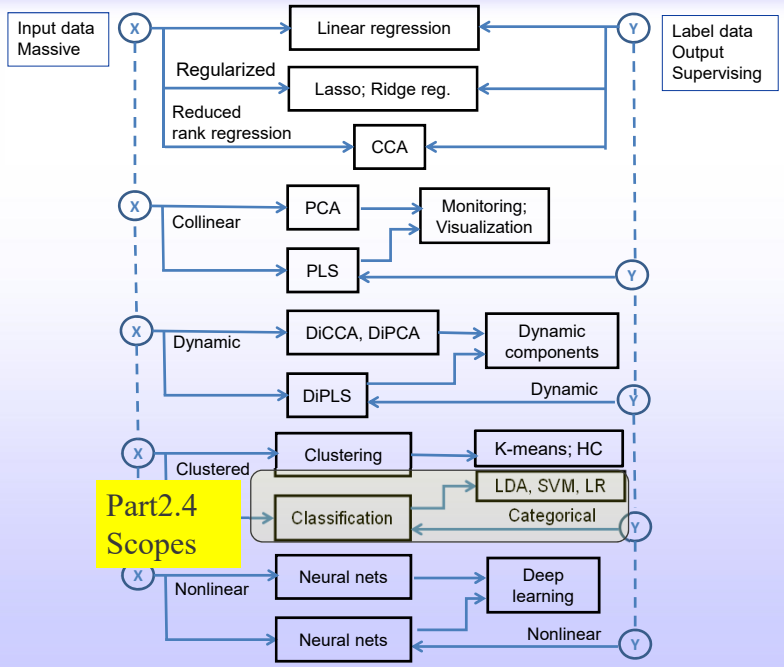
- High dimensional time series data can be handled properly now
  - Dynamics are useful for prediction
  - The residuals facilitate proper inference
- Process/control knowledge is needed
- The control community understands dynamics well to use dynamic data tools
- More applications on the horizon
  - Troubleshooting, on the fly (streaming)
  - Control performance diagnosis, as unstable factors come out first
  - High-D Simulation data visualization

Module 2 – Part 4

Classification Methods: Discriminant Analysis and Support Vector Machines

S. Joe Qin

For the FOPAM Workshop in collaboration with Leo Chiang and Richard Braatz



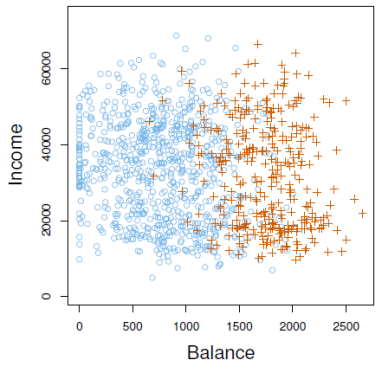
Classification – Supervised learning

- Output variables take qualitative values in a set C, such as

  eye color ∈ {brown,blue,green}

  quality ∈ {pass, fail}

- Given a feature vector *X* and a qualitative response *Y*, the classification task is to predict the category of *Y* (default)



Discriminant Analysis

- Here the approach is to model the distribution of X in each of the classes separately, and then use Bayes theorem to obtain Pr(Y|X).
- When we use normal (Gaussian) distributions for each class, this leads to linear discriminant analysis (LDA).
- LDA can work for more than two classes, and provides low-dimensional views of the data.

Bayes theorem for classication

Bayes theorem

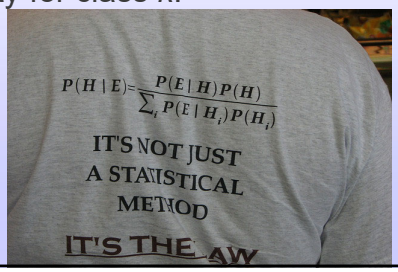
$$Pr(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

where

$f_k(x) = \Pr(X = x \mid Y = k)$ is the density for X in class k.

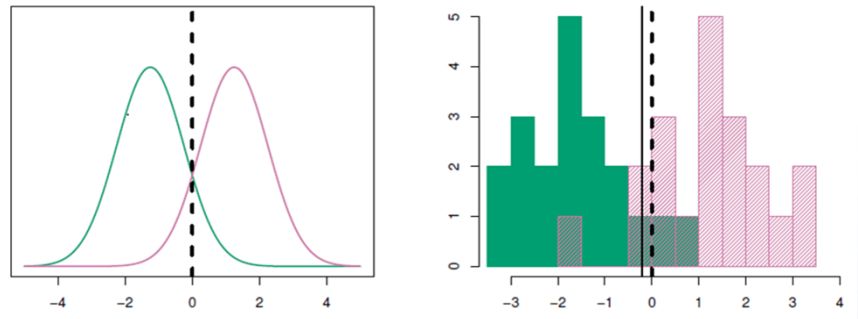
Here we use a normal distribution density for each class.

$\pi_k = \Pr(Y = k)$ is the *prior* probability for class k.



5

Linear Discriminant Analysis when p = 1



Example with $\mu_1 = -1.5$, $\mu_2 = 1.5$, $\pi_1 = \pi_2 = 0.5$, and $\sigma_2 = 1$.
Typically we don't know these parameters; we just have the training data. In that case we simply estimate the parameters and plug them into the rule.

6

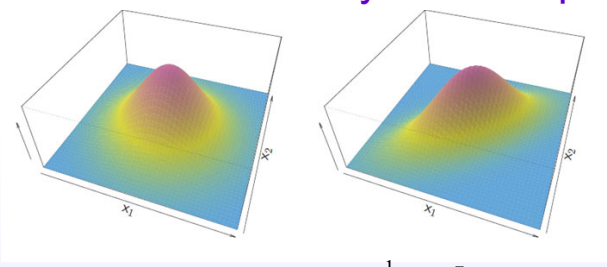
Estimating the parameters

$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-k} \sum_{K=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

Linear Discriminant Analysis when p > 1



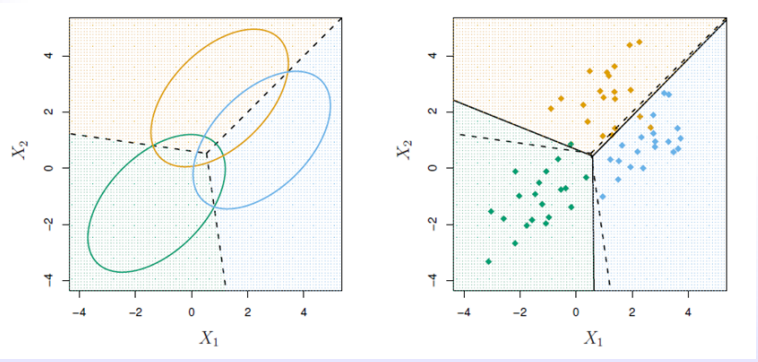
Density: $$f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Discriminant function: $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k \Sigma^{-1} \mu_k + \log \pi_k$

Despite its complex form, it is:

$\delta_k(x) = c_{k0} + c_{k1} x_1 + c_{k2} x_2 + \ldots + c_{kp} x_p$ *a linear function*

Illustration: p = 2 and K = 3 classes

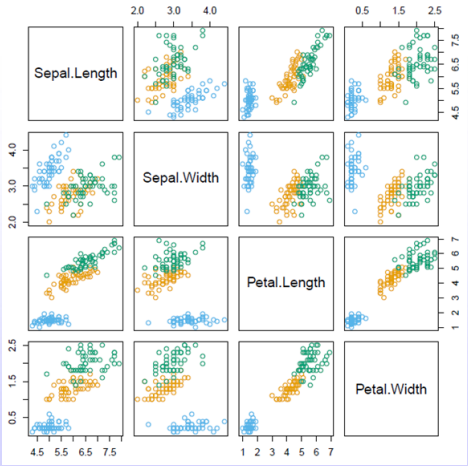


Here $\pi_1 = \pi_2 = \pi_3 = 1/3$.
The dashed lines are known as the *Bayes decision boundaries*.
The dashed lines are based on data.

Fisher's Iris Data

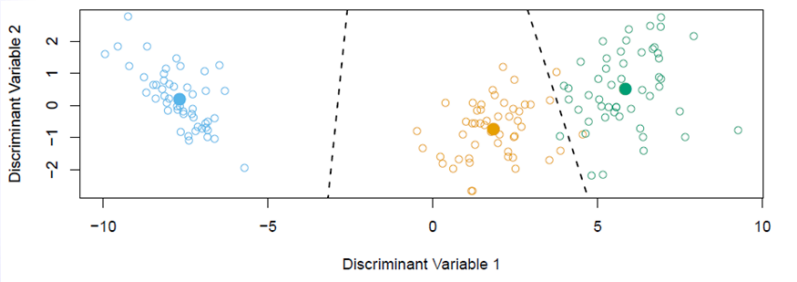
- Setosa
- Versicolor
- Virginica

4 Variables
3 species
50 samples/class



LDA classifies all but 3 of the 150 training samples correctly

Fisher's Discriminant Plot



- When there are K classes, linear discriminant analysis can be viewed exactly in a $K-1$ dimensional plot.
- When $K>3$, we can find the "best" 2-dimensional plane for visualizing the discriminant rule. Like PCA

Types of Errors

False positive (false alarm) rate: The fraction of negative examples that are classified as positive — e.g., 0.2%.
False negative (missed detection) rate: The fraction of positive examples that are classified as negative — e.g., 5.7%.

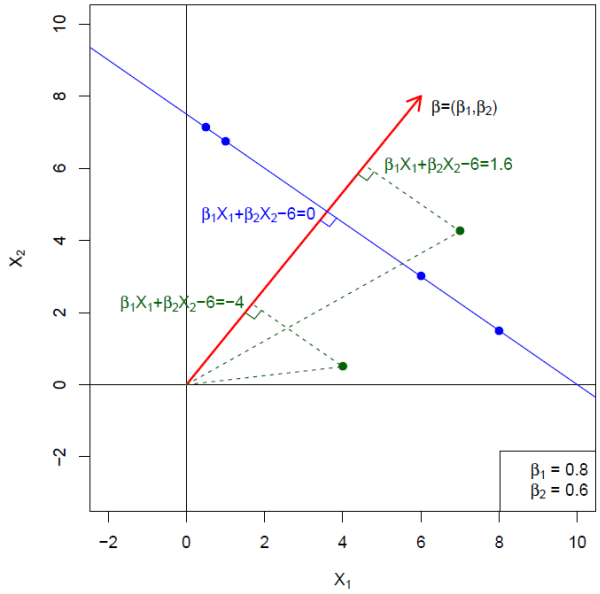
We can balance the two error rates by tuning the threshold

Support Vector Machines

- We try to find a hyperplane that separates two classes in feature space.
- If we cannot, we get creative in two ways:
  - We soften what we mean by “separates”, and
  - We enrich and enlarge the feature space so that separation is possible

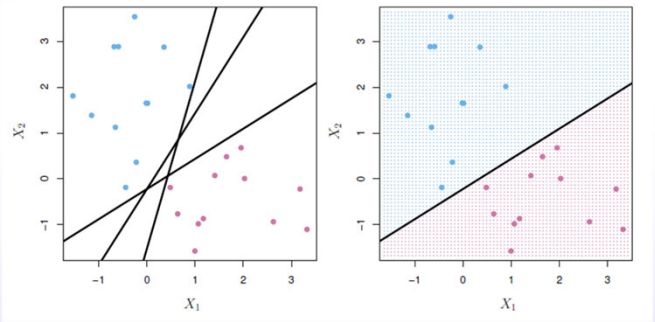
13 © S. Joe Qin

Hyperplane in 2 Dimensions



14 © S. Joe Qin

Separating Hyperplanes

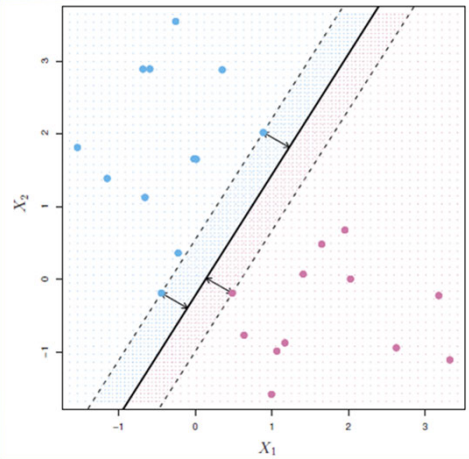


- If $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$, then $f(X)>0$ for points on one sides of the hyperplane, and $f(X)<0$ for points on the other.
- If we code the colored points as $Y_i=+1$ for blue, and $Y_i=-1$ for mauve, then if $Y_i f(X_i)>0$ for all i.

$f(X)=0$ defines a separating hyperplane

Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.

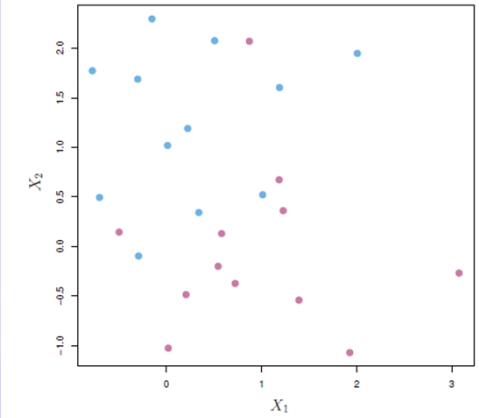


Constrained optimization problem

$$\begin{aligned} &\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M \\ &\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \\ &y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M \\ &\text{for all } i = 1, \ldots, N. \end{aligned}$$

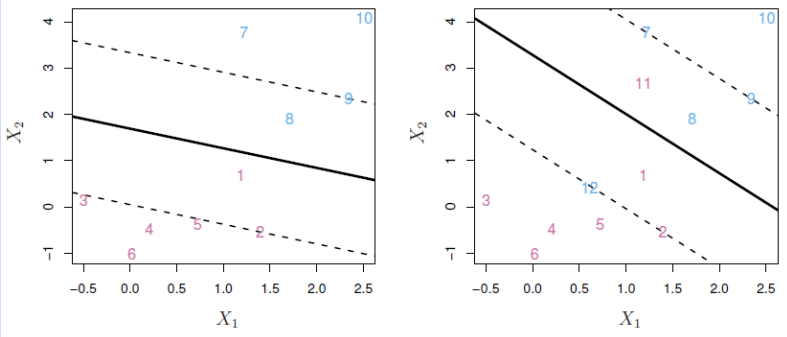
Non-separable Data

The *support vector classifier* maximizes a soft margin.



The data on the left are not separable by a linear boundary.

Support Vector Classifier

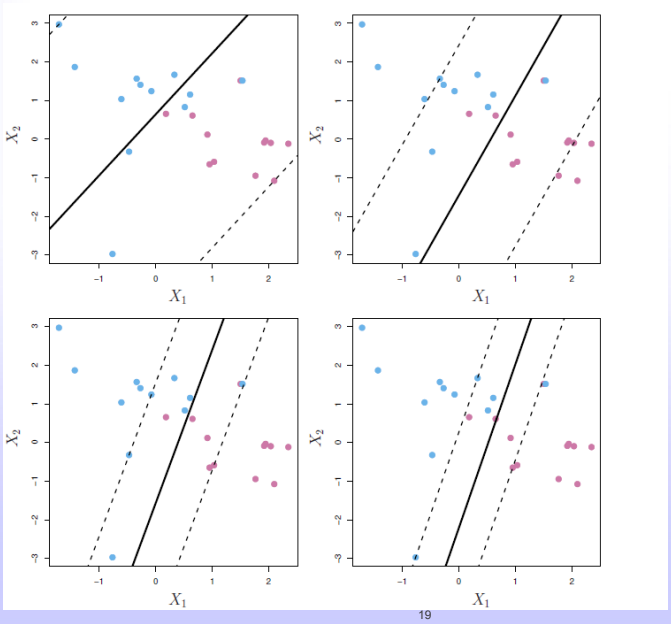


$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \; M \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

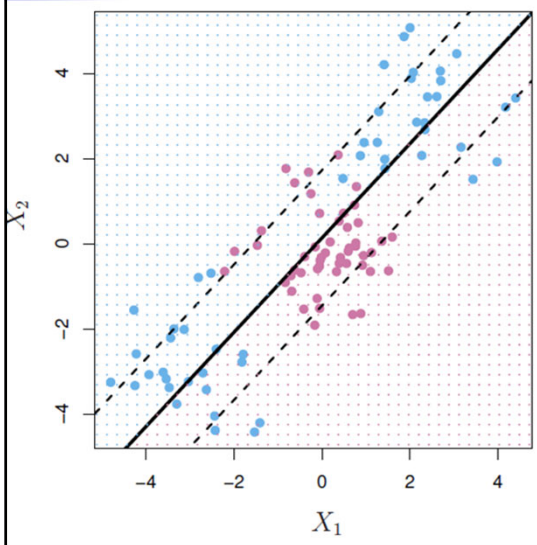
$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

C is a regularization parameter



19

Linear boundary can fail



Sometime a linear boundary simply won't work, no matter what value of *C*.

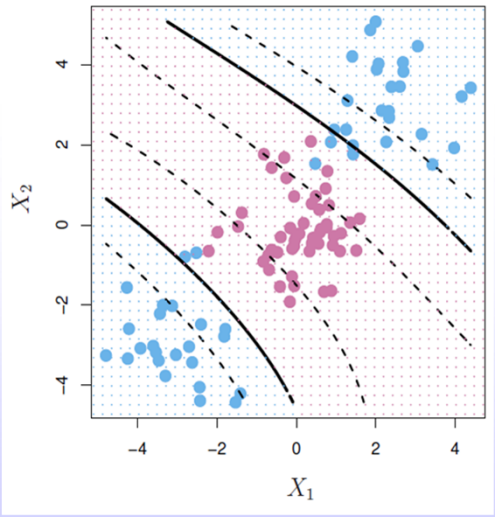
The example on the left is such a case.

What to do*?*

20

Cubic Polynomials

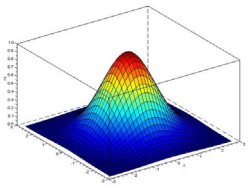
- Here we use a basis expansion of cubic polynomials
- Form 2 variables to 9
- The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space



$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 + \beta_6 X_1^3 + \beta_7 X_2^3 + \beta_8 X_1 X_2^2 + \beta_9 X_1^2 X_2 = 0$$

Nonlinearities and Radial Kernel

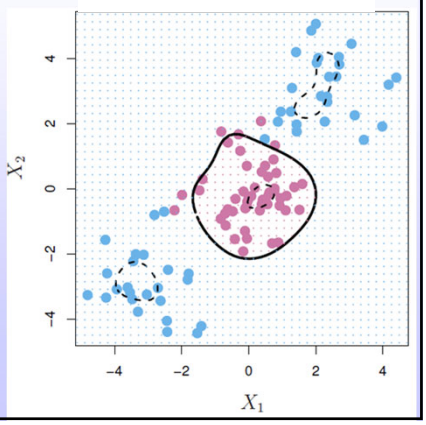
$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$



$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i)$$

Implicit feature space; very high dimensional.

Better than the cubic polynomials, which can get wild.



Summary - Which to Use

- For multi-class problems, LDA is popular.
- For nonlinear boundaries, kernel SVMs are popular.
- For two-class problems,

$$Y = \begin{cases} 0 & \text{if No} \\ 1 & \text{if Yes} \end{cases}$$

can we simply perform a linear regression of Y on X and classify as Yes if $\hat{Y} > 0.5$?

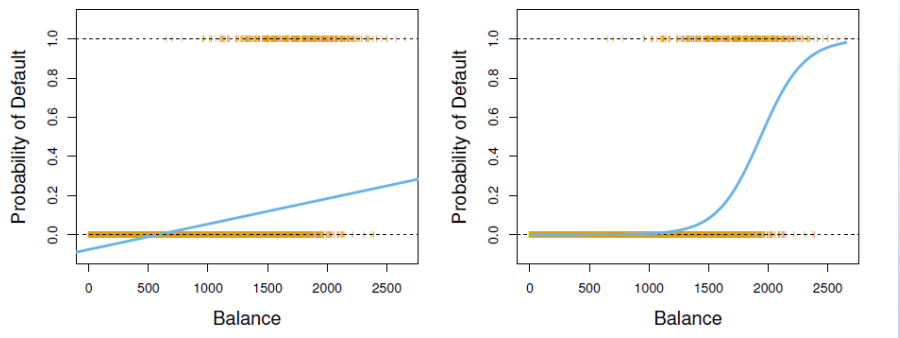
- It's not so bad; PLS-DA does this
- But linear regression can produce probabilities less than zero or bigger than one. *Logistic regression* is more appropriate.

Linear versus Logistic Regression (LR)

- LR uses

$$P(X) = \frac{e^Y}{1+e^Y} = \frac{1}{1+e^{-Y}}$$

- The orange marks indicate the response Y, either 0 or 1. Logistic regression seems well suited to the task.

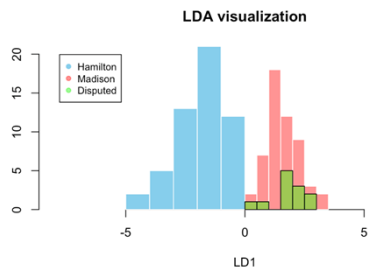


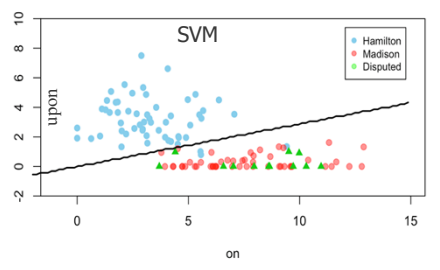
[WORKSHOP] Classify Federalist Papers

- Using SVM and LDA for the Federalist data set

The Federalist Papers were written in 1787-1788 by Alexander Hamilton, John Jay, and James Madison to persuade the citizens of the State of New York to ratify the U.S. Constitution.

The data set has 56 papers written by Hamilton, 50 papers written by Madison, and 12 disputed paper. There are 70 variables that correspond to the relative frequencies (number of occurrences per 1000 words of the text) of the 70 function words. We want to select a pair of words and use them to determine the authors of the 12 disputed papers.





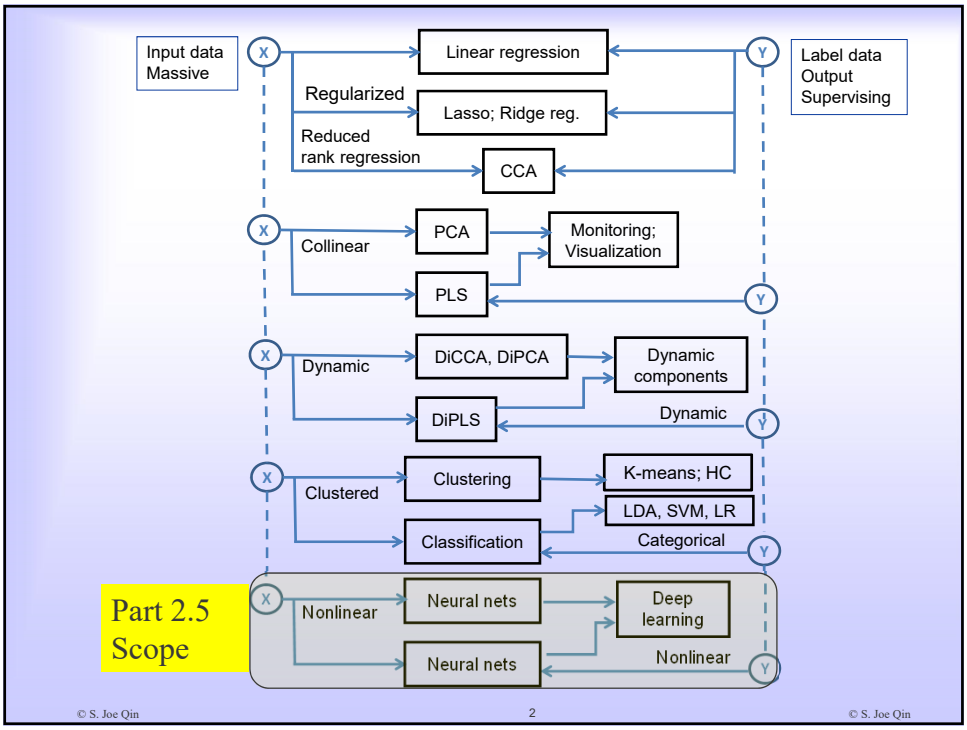
Module 2 – Part 5

Nonlinear and Neural Network Extensions to "Latent Variables"

S. Joe Qin

For the FOPAM Workshop in collaboration with Leo Chiang and Richard Braatz

1 ·

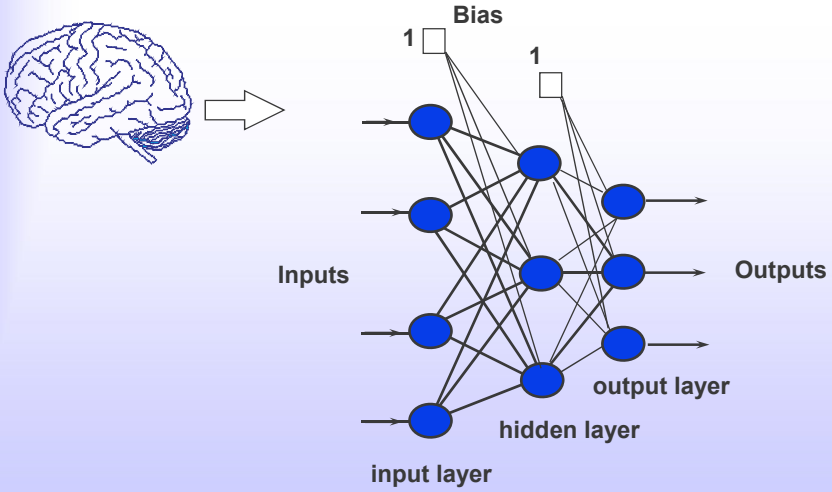


Introduction

- Real world data is usually nonlinear
- When it's nonlinear, use
  - polynomials,
  - splines,
  - local regression, or
  - neural networks

These methods offer a lot of flexibility, but need to watch for overfitting.

Neural Networks – Multilayer Perceptron



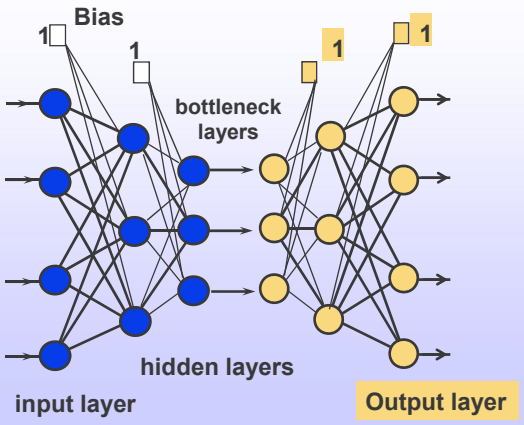
Neural Networks

- Capable of representing any continuous nonlinear functions
- Typically one hidden layer; could be more
  - Deep learning uses much more!
- Transfer functions are typically sigmoidal, radial basis, or ReLU (rectified linear unit)
- If both inputs and outputs represent the same signals, it is known as auto-associative networks
  - nonlinear principal components (Kramer, 1991)
  - Also known as auto-encoder (LeCun, 1987)
  - unsupervised learning

5

Unsupervised, auto-associative networks

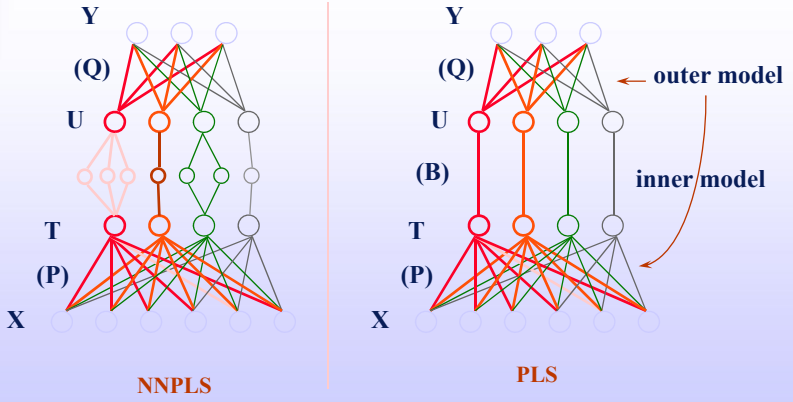
- Kramer (1991). Auto-associative networks
- Dong and McAvoy (1996). Principal curves for NLPCA
- Tan and Mavrovouniotis (1995) input training network



6

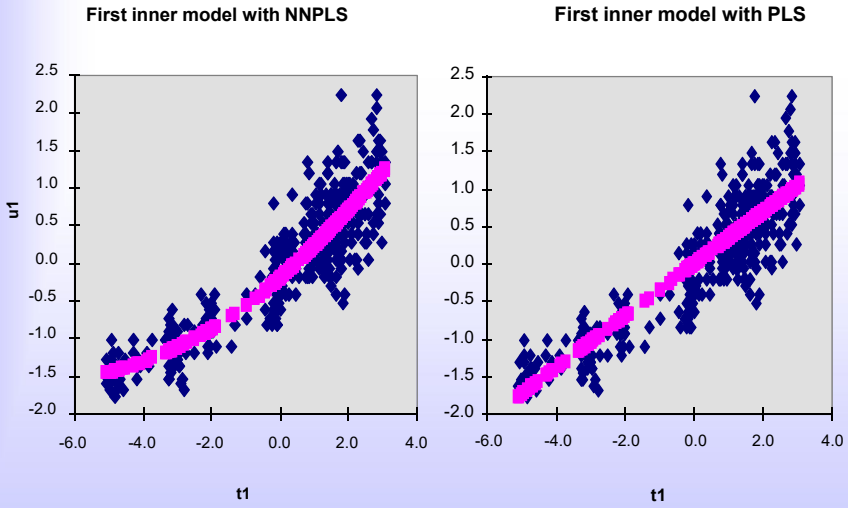
Neural Net PLS

- NNPLS uses neural nets as inner models. Each factor may use a different number of hidden neurons, depending on the nonlinearity (Qin and McAvoy, 1992)



7 © S. Joe Qin

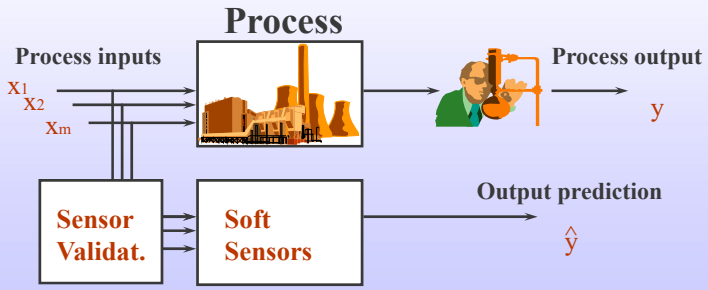
NNPLS vs PLS inner model



8 © S. Joe Qin

Soft Sensors/Inferential Sensors

- Inferential sensors: Neural nets are used to map on-line process measurements to hard-to-measure quality variables
- Sensor validation block is often required to validate and replace faulty sensors



Summary

- Neural Networks found their applications in inferential sensors and nonlinear model predictive control
- Deep learning is an extension of neural net learning techniques, but it uses many layers; difficulty in interpreting
- Convolution neural nets (CNN) – a special structure for image processing to build in a scanning window structure for images, instead of a huge fully connected net
  - Applying CNN to model high dimensional time series data does not make much sense
  - Time series model is 1-D convolution already!

Journal of Process Control 20 (2010) 134–142

Contents lists available at ScienceDirect

Journal of Process Control

journal homepage: www.elsevier.com/locate/jprocont

Stability analysis and optimal tuning of EWMA controllers – Gain adaptation vs. intercept adaptation

Jin Wang [a,*], Q. Peter He [b], S. Joe Qin [c]

[a] Department of Chemical Engineering, Auburn University, Auburn, AL 36849, United States
[b] Department of Chemical Engineering, Tuskegee University, Tuskegee, AL 36088, United States
[c] The Mork Family Department of Chemical Engineering and Materials Science, University of Southern California, Los Angeles, CA 90089, United States

ARTICLE INFO

Article history:
Received 7 April 2009
Received in revised form 11 June 2009
Accepted 11 June 2009

Keywords:
Exponentially weighted moving average (EWMA)
Stability
Optimal weighting
State-space representation
The Kalman filter
Run-to-run control

ABSTRACT

Exponentially weighted moving average (EWMA) controllers are the most commonly used run-to-run controllers in semiconductor manufacturing industry. An EWMA controller can be implemented in two different ways. One way is to keep the process gain as its off-line estimate and update the intercept term at each run, which is termed EWMA with intercept adaptation; the other is to keep the intercept term as its off-line estimate and update the process gain at each run, which is termed EWMA with gain adaptation. Despite the fact that gain variation and adaptation is typical in semiconductor industry, most EWMA formulations are for intercept adaptation and few results exist on the stability and sensitivity of EWMA with gain adaptation. In this paper, we propose a general formulation to analyze the stability of both EWMA controllers. The proposed state-space representation not only reveals the similarities and differences between two types of EWMA controllers, but also explains why the stability conditions for both types of EWMA controllers are independent of process disturbances. In addition, we propose a general framework that unifies the analysis of the optimal control performance for both types of EWMA controllers. The proposed framework is different from existing approaches in that it decouples the state estimation from the control law, and derives the optimal weighting based on the state estimation performance. The proposed framework significantly simplifies the analysis procedure, especially for EWMA with gain adaptation. Using this framework, we derive the optimal EWMA weighting through solving the discrete-time algebraic Riccati equation (DARE) for various process disturbances that are encountered in semiconductor manufacturing industry. Simulation examples are given to illustrate the optimality of the EWMA weighting derived using the framework. Some practical aspects of controller tuning are also discussed based on the simulation results.

1. Introduction

Due to their simplicity and robustness, exponentially weighted moving average (EWMA) controllers are the most commonly used run-to-run feedback controllers in semiconductor manufacturing processes [1–4]. In general, an EWMA controller uses a linear regression model of a process obtained based on off-line experiments or historical data (as shown in Eq. (1)). During on-line estimation and control, one of the model parameters (i.e., intercept or gain) is updated using EWMA statistics with newly observed process data, then the control move for the next run is calculated using the updated process model.

Consider a single-input-single-output (SISO) linear model

$$y[n] = \alpha + \beta u[n] + \epsilon[n] \tag{1}$$

where α and β are model parameters, $\epsilon[n]$ is the process disturbance sequence, $u[n]$ is the input to the process at the beginning of run n, and $y[n]$ is the output of the process at the end of run n.

An EWMA controller with intercept adaptation, denoted as EWMA-I controller, can be formulated as the following:

$$a[n+1] = \omega(y[n] - bu[n]) + (1-\omega)a[n] \tag{2}$$

$$u[n+1] = \frac{T - a[n+1]}{b} \tag{3}$$

where b is the off-line estimate of β, $a[n+1]$ is the on-line estimate of α that is updated after each run, ω is the EWMA weighting and T is the target of the process output. EWMA-I controllers have many applications in the semiconductor manufacturing industry, such as lithography overlay control, and its properties have been studied extensively, e.g., see [1,3,5–7]. Among the existing results, Ingolfsson and Sachs analyzed its stability and sensitivity for white noise and deterministic drift disturbances [5], Del Castillo analyzed its properties for random walk, integrated moving average

* Corresponding author. Tel.: +1 334 844 2020; fax: +1 334 844 2063.
E-mail address: wang@auburn.edu (J. Wang).

0959-1524/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jprocont.2009.06.002

(IMA(1,1)) and deterministic trend drift with random walk disturbances [6]. In the existing analyses, it was shown that for all the disturbances that have been examined specifically, the stability condition for an EWMA-I controller is always $0 < \frac{\omega\beta}{b} < 2$.

On the other hand, in practice an EWMA controller is often implemented in a different way for processes with varying process gains. Specifically, the process gain, instead of the intercept, is updated using EWMA statistics in a run-to-run manner. Examples of such applications include chemical mechanical polishing (CMP) process where the polishing rate is updated [8–10]; chemical vapor deposition (CVD) process where the deposition rate is updated [11]; and etch process where the etch rate is updated [12,13]. For an EWMA controller with gain adaptation (denoted as EWMA-G controller), the control law and the state estimation equations become the following:

$$b[n+1] = \omega\frac{y[n]-a}{u[n]} + (1-\omega)b[n] \tag{4}$$

$$u[n+1] = \frac{T-a}{b[n+1]} \tag{5}$$

where a is the off-line estimate of α and $b[n+1]$ is the on-line estimate of β.

Although EWMA-G controllers have been applied more often than EWMA-I controllers in practice, little research has been conducted to understand their properties. Wang and He [14] analyze the behavior of an EWMA-G controller for white noise and deterministic drift disturbances, and find that there are some interesting similarities between EWMA-G and EWMA-I controllers. For example, the stability condition for EWMA-G controller for both types of disturbances is $0 < \frac{T-\alpha}{T-a}\omega < 2$, which has a similar expression as the stability condition for EWMA-I controller, i.e., $0 < \frac{\beta}{b}\omega < 2$. In addition, the output variance for a process with white noise disturbance controlled by the EWMA-G controller is the same as that controlled by the EWMA-I controller. On the other hand, there are some differences between the two types of EWMA controllers. For example, for unstable cases where the estimated state sequences of both EWMA controllers become unbounded, the process outputs show completely different behaviors: the process output controlled by EWMA-I becomes unbounded, while the process output controlled by EWMA-G converges to the process intercept, and remain bounded.

In this work, we study the properties of the EWMA-G controller systematically by examining its behavior in rejecting different disturbances, and comparing it with that of the EWMA-I controller. We examine four different disturbances, namely, white noise (WN), integrated moving average (IMA), deterministic drift with white noise (DD-WN) and deterministic drift with integrated moving average (DD-IMA). Section 2 gives problem formulation and the mathematical description of the four disturbances. In Section 3, we develop a general formulation to analyze the stability of the closed-loop system controlled by both EWMA-G and EWMA-I controllers. The formulation enables us to reveal and explain the similarities and differences between the two types of EWMA controllers. In Section 4, we propose a general framework based on the Kalman filter formulation to analyze the sensitivity of both types of controllers. By decoupling the state estimation from the feed back control law, the framework unifies the analysis of optimal control performance for both types of EWMA controllers. Optimal EWMA weightings are also derived for different process disturbances. Simulation examples are given in Section 5 to illustrate the results obtained in this work, and Section 6 summarizes the paper.

2. Problem formulation

In this work, the analysis is limited to linear, single-input-single-output processes, as vast majority of the run-to-run controllers in semiconductor industry utilize such a simple process model. However, similar analysis can be extended to MIMO processes, which is not within the scope of this work. To facilitate the comparison of EWMA-G with EWMA-I controllers, we modify the process model Eq. (1) slightly to combine the process disturbances with the corresponding state that is updated at each run.

For EWMA-I,

$$y[n] = \alpha[n] + \beta u[n] \tag{6}$$

For EWMA-G,

$$y[n] = \alpha + \beta[n]u[n] \tag{7}$$

where $\alpha[n]$ and $\beta[n]$ are process disturbances. It is worth noting that measurement noise is not explicitly considered in Eqs. (6) and (7) because it can be combined into the disturbance. The mathematical descriptions of four different disturbances considered in this work are given below, where $w[n]$ denotes a white noise sequence with variance σ^2. The following mathematical descriptions apply to both α and β, i.e., $e[n+1]$ could be either $\alpha[n+1]$ or $\beta[n+1]$.

1. White noise disturbance (WN)

$$e[n+1] = w[n+1] \tag{8}$$

2. Integrated moving average disturbance (IMA)

$$e[n+1] = e[n] + w[n+1] - \lambda w[n] \tag{9}$$

where $\lambda \in [0,1]$ is a constant. Notice that if $\lambda = 0$, an IMA disturbance reduces to a random walk (RW) disturbance.

3. Deterministic drift with white noise disturbance (DD-WN)

$$e[n+1] = (n+1)\delta + w[n+1] \tag{10}$$

where δ is the drifting slope.

4. Deterministic drift with integrated moving average disturbance (DD-IMA)

$$e[n+1] = e[n] + \delta + w[n+1] - \lambda w[n] \tag{11}$$

Similar to the case of IMA disturbance, when $\lambda = 0$, a DD-IMA disturbance reduces to a deterministic drift with random walk (DD-RW) disturbance.

It is obvious that the first two disturbances are special cases of the last two disturbances with the deterministic drift slope $\delta = 0$. In this work we keep them as separate cases because the optimal controllers derived for the first two cases are EWMA controllers while the optimal controllers derived for the last two disturbances are double-EWMA controllers.

3. Stability condition for EWMA controllers

The stability condition for EWMA-I controllers has been well studied. For processes controlled by EWMA-I controllers with white noise or deterministic drift disturbances, Ingolfsson and Sachs [5] expressed the process output as an infinite power series and derived the stability condition based on the convergence of the infinite series. For processes controlled by EWMA-I controllers with RW or IMA disturbances, Del Castilo [6] derived the stability condition by examining the asymptotic mean squared deviation of the process output from the target and obtained the same stability condition, i.e., $0 < \frac{\beta}{b}\omega < 2$. For the stability condition of EWMA-G controllers, Wang and He [14] analyze the processes with white noise and deterministic drift disturbance by deriving the expression of process output as an infinite power series, and find that the same stability condition, i.e., $0 < \frac{T-\alpha}{T-a}\omega < 2$, apply to both disturbances.

In this section, we develop a general formulation to analyze the similarities and differences of the stability conditions of EWMA-I

and EWMA-G controllers by examining the closed-loop pole, instead of the process output under closed-loop control as in existing work. In this way, the specific disturbance dynamics will not be incorporated and the results apply to all types of disturbances.

Without loss of generality, we assume that the process target T is zero in the rest of the paper, as the intercept term α can always be shifted. First, we consider the closed-loop process in Eq. (7) controlled by an EWMA-G controller. By plugging the control law (Eq. (5)) into Eqs. (4) and (7), we can formulate the closed-loop system into the following state-space representation:

$$b[n+1] = \left(1 - \frac{\alpha}{a}\omega\right) b[n] + \omega\beta[n] \tag{12}$$

$$y[n] = -\frac{a\beta[n]}{b[n]} + \alpha \tag{13}$$

which has a linear state equation but a nonlinear output equation. Note that in Eqs. (12) and (13), the estimated process disturbance $b[n]$ is chosen as the state of the closed-loop system, and the process disturbance $\beta[n]$ serves as the input to the closed-loop system and does not affect the stability of the closed-loop system. It is straightforward to see from Eq. (12) that the sufficient condition for the closed-loop stability is,

$$\left|1 - \frac{\alpha}{a}\omega\right| < 1 \tag{14}$$

which is a general stability condition and applies to all types of bounded disturbances. In practice all process disturbances are bounded due to preventive maintenances. Therefore, the stability condition in Eq. (14) is generally applicable. Note that because a closed-loop system controlled by an EWMA-G controller has a nonlinear output equation (i.e., Eq. (13)), the sufficient condition given in Eq. (14) may not be necessary. The stability condition is illustrated in Fig. 1a, and is consistent with that derived in Ref. [14].

For processes controlled by EWMA-I controllers, following the similar procedure, we obtain the following state-space representation of the closed-loop system:

$$a[n+1] = \left(1 - \frac{\beta}{b}\omega\right) a[n] + \omega\alpha[n] \tag{15}$$

$$y[n] = -\frac{\beta}{b}a[n] + \alpha[n] \tag{16}$$

which has linear state and output equations. Similarly, the stability condition can be obtained from Eq. (15), which is

$$|1 - \frac{\beta}{b}\omega| < 1 \tag{17}$$

which is illustrated in Fig. 1b. Because of the linear output equation, this stability condition is sufficient and necessary. The above general stability condition agrees with the condition derived in [5,6] for several specific disturbances considered therein.

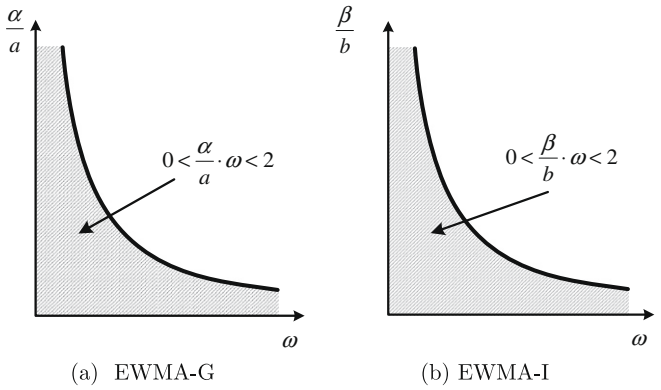


Fig. 1. The stability region for EWMA controllers.

It is worth noting that by formulating the closed-loop system into the state-space representations in Eqs. (12) and (13) and (15) and (16), the stability analysis is significant simplified compared with existing results [5,6,14]. In existing literature [5,6,14], although the same stability condition was obtained for different disturbances, the stability condition derived for one type of disturbances does not directly apply to others as the specific disturbance dynamics is involved in the derivation. In addition, with the state-space formulation, the similar closed-loop stability conditions of the EWMA-G and EWMA-I controllers can be easily explained by their similar linear state equations, while the different output responses of unstable systems can be explained by their different output equations. As shown below, for bounded process disturbances $\alpha[n]$ and $\beta[n]$, when the stability condition is not satisfied the estimated states for both EWMA-I and EWMA-G become unbounded, i.e.,

$$\lim_{n\to\infty} |a[n]| \to \infty \text{ for EWMA-I} \tag{18}$$

$$\lim_{n\to\infty} |b[n]| \to \infty \text{ for EWMA-G} \tag{19}$$

However, the process output controlled by an EWMA-G controller will stay bounded while the process output controlled by an EWMA-I controller will become unbounded. This is illustrated in Fig. 2 with a simulated linear process with $T = 0$, $\alpha = 0.056$ for EWMA-G, $\beta = 0.049$ for EWMA-I and IMA disturbances for the states to be updated at each run. Notice that when the pole of an unstable system is negative, the estimated states, i.e., $a[n]$ for EWMA-I and $b[n]$ for EWMA-G, will oscillate and their limits do not exist. However, for the oscillating case, the absolute value of the estimated states will approach positive infinity as n approaches infinity. The difference in the process output response can be explained as the following. Plugging the estimated states into the corresponding output equations, with bounded process disturbances $\alpha[n]$ and $\beta[n]$, for EWMA-I

$$\lim_{n\to\infty} |y[n]| = \lim_{n\to\infty} \left|-\frac{\beta}{b}a[n] + \alpha[n]\right| \to \infty \tag{20}$$

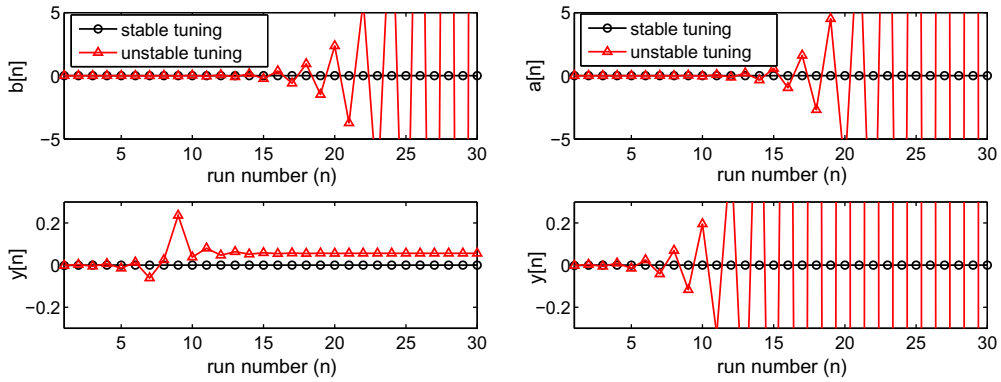
while for EWMA-G

$$\lim_{n\to\infty} y[n] = \lim_{n\to\infty} \left(-\frac{a\beta[n]}{b[n]} + \alpha\right) \to \alpha \tag{21}$$

4. Optimal EWMA weighting

In this section, we examine the sensitivity of EWMA controllers by deriving their optimal weightings for different types of process disturbances. Such analysis has been performed for EWMA-I controllers for several different process disturbances (e.g., see [5,6]), and existing results were all derived by expressing the process output as a function of EWMA weighting and minimizing the output variance or mean squared output error. In this work, we take the characteristics of semiconductor processes into account, which enable us to decouple the state estimation from the feedback control law, and focus on the state estimation performance in deriving the optimal weightings. Without involving the feedback control law, the analysis can be significantly simplified, especially for EWMA-G controllers as the dead-beat control law is nonlinear with respect to the estimated state $b[n]$.

In this work, we consider the special case where model-plant mismatch is absent, and the investigation of the effect of model-plant mismatch is underway. Without model-plant mismatch, the following two characteristics of semiconductor processes indicate that the control performance of the EWMA controller solely



(a) EWMA-G (b) EWMA-I

Fig. 2. Different behaviors of unstable EWMA controllers.

depends on the performance of state estimation. First, as shown in Eq. (1), the deterministic part of the model is static, i.e., previous inputs have no effect on current outputs. Second, dead-beat controllers are applied to compute the control move in both types of EWMA controllers. Therefore, we can derive the optimal EWMA weighting based on the state estimation performance, instead of the closed-loop control performance, which will simplify the analysis significantly.

Three steps are involved in deriving the optimal EWMA weightings. First, we extract the state estimation problem from the overall closed-loop system; next, we formulate different process disturbances into their state-space representations, and finally we derive the optimal weighting from the Kalman filter gain by solving the discrete algebraic Riccati equation (DARE). In this way, the optimality of the EWMA weighting is guaranteed by the Kalman filter which is the optimal state estimator.

We start with decoupling the state estimation from the dead-beat control law. Instead of expressing the process input as a function of the estimated state, i.e., Eqs. (3) and (5), we use the value of inputs directly as they are available at the end of the run.

For EWMA-G controller, the observed process state can be described by

$$\beta[n+1] = f(\beta[n], w[n], w[n-1]) \tag{22}$$

$$\beta_{obs}[n] = \frac{y[n]-a}{u[n]} \tag{23}$$

where Eq. (22) defines the disturbance dynamics, and Eq. (23) describes how the observed value of $\beta[n]$ is obtained through process measurements. Similarly for EWMA-I controller,

$$\alpha[n+1] = g(\alpha[n], w[n], w[n-1]) \tag{24}$$

$$\alpha_{obs}[n] = y[n] - bu[n] \tag{25}$$

In the rest of the paper, we omit the subscript "*obs*" for simplicity.

For different disturbances, we formulate the state estimation component of both EWMA controllers into their corresponding state-space representations,

$$\boldsymbol{\theta}[n+1] = \mathbf{A}\boldsymbol{\theta}[n] + \mathbf{w}[n] \tag{26}$$

$$z[n] = \mathbf{C}\boldsymbol{\theta}[n] + v[n] \tag{27}$$

where θ denotes the state; $z[n]$ denotes the process disturbance to be estimated, i.e., $\alpha[n]$ for EWMA-I and $\beta[n]$ for EWMA-G; $\mathbf{w}[n]$ is a sequence of white noise vector, and v is a sequence of white noise, with

$$\mathrm{E}\left[\mathbf{w}[k]\mathbf{w}^T[j]\right] = \delta_{kj}\cdot\mathbf{Q} \tag{28}$$

$$\mathrm{E}(v[k]v[j]) = \delta_{kj}\cdot R \tag{29}$$

$$\mathrm{E}[\mathbf{w}[k]v[j]] = \mathbf{0} \tag{30}$$

where δ_{kj} denotes Kronecker delta.

Once the state-space representation is obtained, it is straightforward to compute the steady-state Kalman gain and derive the optimal EWMA weightings for different disturbances. The steady-state Kalman filter is given below:

$$\hat{\boldsymbol{\theta}}[n+1] = \mathbf{A}\hat{\boldsymbol{\theta}}[n] + \mathbf{J}\left(z[n] - \mathbf{C}\hat{\boldsymbol{\theta}}[n]\right) \tag{31}$$

$$\mathbf{J} = \mathbf{APC}^T\left(\mathbf{CPC}^T + R\right)^{-1} \tag{32}$$

$$\mathbf{P} = \mathbf{A}\left[\mathbf{P} - \mathbf{PC}^T\left(\mathbf{CPC}^T + R\right)^{-1}\mathbf{CP}\right]\mathbf{A}^T + \mathbf{Q} \tag{33}$$

and the corresponding optimal estimate of the process disturbance is

$$\hat{z}[n+1] = \mathbf{C}\hat{\boldsymbol{\theta}}[n+1] \tag{34}$$

Solving the DARE, i.e., Eq. (33), we can compute the Kalman gain with Eq. (32). Here we only consider the steady-state Kalman filter, because the EWMA controllers use constant tuning parameters. Eqs. (31)–(34) serve as the general framework for our analysis in this section.

Note that the above state-space formulation applies to both EWMA-G and EWMA-I controllers. For the EWMA-G controller, we have $z[n] = (y[n]-a)/u[n]$, while for the EWMA-I controller, $z[n] = y[n] - b\,u[n]$. In the following analysis, we keep the above general formulation without incorporating the detail of $z[n]$ and the analysis applies to both EWMA controllers. When the closed-loop systems are stable, we expect that EWMA-I and EWMA-G have similar sensitivity analysis results for different types of process disturbances due to their similar linear state equations, i.e., Eqs. (12) and (15).

In the rest of this section, we derive optimal weightings for four different disturbances individually, and we categorize them into two groups: disturbances without a deterministic drift component, which result in EWMA controllers, and disturbances with a deterministic drift component, which result in double-EWMA controllers.

4.1. Disturbances without a deterministic drift component

The WN and IMA disturbances belong to this group. In the following we show that the state estimators that provide optimal estimates for these disturbances are equivalent to EWMA controllers with specific optimal weightings.

4.1.1. White noise (WN) disturbance

When the process disturbance is white noise, we obtain the following state-space model,

$$\theta[n+1] = \theta[n] \tag{35}$$
$$z[n] = \theta[n] + v[n] \tag{36}$$

By comparing the above equations with the general formulation (i.e., Eqs. (26) and (27)), we have $A = 1$, $C = 1$, $R = \sigma^2$ and $Q = 0$. It can be verified that the systems described by the above equations is observable, so the Kalman filter can be constructed to provide the optimal state estimation performance. Solving the DARE, we have $P_\infty = 0$, and correspondingly the Kalman gain $J_\infty = 0$. Plugging the Kalman gain into Eq. (31), we see that the optimal estimate of the process disturbance $z[n]$ can be obtained through

$$\hat{\theta}[n+1] = \hat{\theta}[n] \tag{37}$$
$$\hat{z}[n+1] = \hat{\theta}[n+1] = \hat{\theta}[n] = \hat{z}[n] \tag{38}$$

or equivalently it can be formulated into the EWMA estimate,

$$\hat{z}[n+1] = \hat{z}[n] + 0 \cdot z[n] \tag{39}$$

Eq. (39) indicates that the optimal EWMA weighting is $\omega = 0$, which agrees with the existing result, i.e., if the process disturbance is white, open loop operation provides the optimal control performance. In practice, WN disturbance is rare.

4.1.2. Integrated moving average (IMA) disturbance

The state-space representation for an IMA disturbance is given below:

$$\theta[n+1] = \theta[n] + (1-\lambda)w[n] \tag{40}$$
$$z[n] = \theta[n] + w[n] \tag{41}$$

It is straightforward to verify that $z[n]$ is an IMA sequence. In addition, Eqs. (40) and (41) are already in the Kalman filter form, with the Kalman gain $J_\infty = 1 - \lambda$. Therefore, the optimal estimate of $z[n+1]$ obtained from the Kalman filter is

$$\begin{aligned}\hat{z}[n+1] &= \hat{\theta}[n+1] = \hat{\theta}[n] + (1-\lambda)\Big(z[n] - \hat{\theta}[n]\Big) \\ &= \lambda\hat{z}[n] + (1-\lambda)z[n]\end{aligned} \tag{42}$$

Eq. (42) is exactly the EWMA estimate with the optimal weighting $\omega = 1 - \lambda$, which is the well known result for EWMA-I controller with IMA disturbance. Eq. (42) also shows that the same optimal weighting applies to the EWMA-G controller, independent from the nonlinear dead-beat control law, i.e., Eq. (5). This is illustrated using simulation examples in Section 5.

For the special case of RW disturbance where $\lambda = 0$, the optimal EWMA weighting becomes $\omega = 1$, which indicates that if the process noise is a random walk process, using the most recently measurement would provide the best performance for both EWMA-I and EWMA-G controllers.

4.2. Disturbances with a deterministic drift component

While IMA disturbances provide good approximation for many drifting processes, some processes show deterministic drifts, such as the polishing rate decreases due to pad wear in CMP processes. In this work, we consider the deterministic drift combined with white noise and integrated moving average disturbances. We show that the optimal estimate for the two disturbances with a deterministic drift trend are equivalent to double-EWMA estimates with corresponding optimal tuning parameters. For convenience, we first present the formulation of a double-EWMA estimator:

$$\xi_1[n+1] = \omega_1 z[n] + (1-\omega_1)(\xi_1[n] + \xi_2[n]) \tag{43}$$
$$\xi_2[n+1] = \omega_2(z[n] - \xi_1[n]) + (1-\omega_2)\xi_2[n] \tag{44}$$
$$\hat{z}[n+1] = \xi_1[n+1] + \xi_2[n+1] \tag{45}$$

where ξ_1 can be viewed as the estimate of the current state, and ξ_2 is the estimate of the drifting slope, i.e., δ in Eq. (10).

4.2.1. Deterministic drift with white noise (DD-WN) disturbance

The state-space representation for DD-WN is the following:

$$\begin{bmatrix}\theta_1[n+1] \\ \theta_2[n+1]\end{bmatrix} = \begin{bmatrix}1 & 0 \\ 1 & 1\end{bmatrix}\begin{bmatrix}\theta_1[n] \\ \theta_2[n]\end{bmatrix} \tag{46}$$
$$z[n] = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}\theta_1[n] \\ \theta_2[n]\end{bmatrix} + v[n] \tag{47}$$

First we verify that Eqs. (46) and (47) generate a DD-WN sequence. From Eq. (46) we see that θ_1 is a constant, which is also the deterministic drift slope, i.e., δ in Eq. (10). Plugging Eq. (46) repeatedly into (47), we have

$$\begin{aligned}z[n+1] &= \theta_2[0] + (n+1)\theta_1[0] + v[n+1] \\ &= \theta_2[0] + (n+1)\delta + v[n+1]\end{aligned} \tag{48}$$

which is a DD-WN sequence. Next we derive the optimal estimate using the Kalman filter. For the DD-WN disturbance, $\mathbf{A} = \begin{bmatrix}1 & 0 \\ 1 & 1\end{bmatrix}, \mathbf{C} = [0\ 1], \mathbf{Q} = \mathbf{0}_{2\times 2}$ and $R = \sigma^2$.

Solving the DARE, we obtain:

$$\mathbf{P}_\infty = \mathbf{0}_{2\times 2} \tag{49}$$
$$\mathbf{J}_\infty = \mathbf{0}_{2\times 1} \tag{50}$$

and the optimal estimates are

$$\hat{\boldsymbol{\theta}}[n+1] = \mathbf{A}\hat{\boldsymbol{\theta}}[n] \tag{51}$$
$$\hat{z}[n+1] = \hat{\theta}_2[n+1] = \hat{\theta}_1[n] + \hat{\theta}_2[n] = \hat{\delta} + \hat{z}[n] \tag{52}$$

Eq. (51) indicates that for DD-WN disturbance, the optimal estimate of the $\boldsymbol{\theta}[n]$ is obtained based on model prediction only, without any measurement feedback, which is expected as the added noise is white. Consequently, the optimal estimate of the process disturbance is based on the previous estimate only, without measurement feedback as shown in Eq. (52). We further show that Eq. (52) is equivalent to the double-EWMA estimation with $\omega_1 = 0$ and $\omega_2 = 0$. The detailed proof of the equivalency is given in Appendix A.

4.2.2. Deterministic drift with integrated moving average (DD-IMA) disturbance

The state-space representation of a DD-IMA disturbance is given

$$\begin{bmatrix}\theta_1[n+1] \\ \theta_2[n+1]\end{bmatrix} = \begin{bmatrix}1 & 0 \\ 1 & 1\end{bmatrix}\begin{bmatrix}\theta_1[n] \\ \theta_2[n]\end{bmatrix} + \begin{bmatrix}0 \\ 1-\lambda\end{bmatrix}w[n] \tag{53}$$
$$z[n] = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}\theta_1[n] \\ \theta_2[n]\end{bmatrix} + w[n] \tag{54}$$

where the state θ_1 represents the deterministic drift slope δ, and it is easy to verify that

$$z[n+1] = z[n] + \delta + w[n+1] - \lambda w[n] \tag{55}$$

which is a DD-IMA sequence. Similar to the case of the IMA disturbance, Eqs. (53) and (54) are in the form of innovation representation with the Kalman gain $\mathbf{J}_\infty = [0\ (1-\lambda)]^T$. Correspondingly, the optimal estimate of $z[n+1]$ is

$$\begin{aligned}\hat{z}[n+1] &= \hat{\theta}_2[n+1] = \hat{\theta}_1[n] + \hat{\theta}_2[n] + (1-\lambda)\Big(z[n] - \hat{\theta}_2[n]\Big) \\ &= \lambda\hat{z}[n] + (1-\lambda)z[n] + \hat{\theta}_1[n]\end{aligned} \tag{56}$$

which is equivalent to the double-EWMA estimate of the disturbance with $\omega_1 = 1 - \lambda$ and $\omega_2 = 0$. The proof of the equivalency is given in Appendix B.

Again, for the special case of $\lambda = 0$, DD-IMA disturbance reduces to DD-RW disturbance, and its optimal estimates corresponds to the double-EWMA filter with $\omega_1 = 1$ and $\omega_2 = 0$.

It is worth noting that for disturbances with deterministic drift, the optimal EWMA weighting that updates the drifting slope has a value of zero, i.e., $\omega_2 = 0$. This is expected because the model assumes constant, or deterministic, drift slope, therefore the best estimate of the drift slope is simply the average of the differences between two consecutive measurements, i.e., $z[n+1] - z[n]$. Also note that all derivations in this subsection are applicable to both double-EWMA-I and double-EWMA-G controllers, as their corresponding dead-beat control laws are not involved.

5. Simulation examples

In this section, we use simulation examples to illustrate and verify the results we obtained in the previous sections. Eqs. (6) and (7) are used to simulate the processes, Eqs. (2) and (4) are used to estimate process states, and Eqs. (3) and (5) are used to calculate control moves, for processes controlled by EWMA-I and EWMA-G controllers, respectively. Eqs. (8)–(11) are used to simulate different types of disturbances. Model parameters are listed in Table 1, where T represents the process target. For processes controlled by the EWMA-I controller, $a[0] = 5; b = 4$, while for processes controlled by the EWMA-G controller, $b[0] = 5; a = 6$.

Table 1
Simulation parameters.

Parameter	Value	Parameter	Value
α	6	β	4
σ	0.5	δ	0.005
λ	0.7	T	10

5.1. Simulation set-up

For each type of disturbance, in order to verify that the optimal state estimation performance is equivalent to the optimal control performance, we conduct exhaustive search and use mean squared error (MSE) of the process output prediction as the performance index to determine the optimal EWMA weighting. For each setting of ω, 10,000 runs are simulated. To eliminate the effect of initial state, only the last 9000 runs are used to evaluate the MSE. For disturbances with deterministic drift, to avoid the deterministic drift becoming dominant in the process state as the run number becomes large, we reset the process state to its initial value, i.e., α or β, every 500 runs. When the process state is reset, the estimated state will be changed by the same amount to allow the state estimator continuously converge to its steady state.

5.2. Optimal EWMA weightings

We first examine the optimal weightings for EWMA controllers in rejecting WN, RW and IMA disturbances. We determine the optimal weightings for both EWMA-I and EWMA-G controllers using simulation. The results are shown in Fig. 3, where the optimal weightings are indicated by stars. Fig. 3 confirms that for WN disturbance, the optimal control performance is achieved at $\omega = 0$ for both EWMA controllers, which agrees with the weighting that derived from the optimal state estimation performance. Similarly for RW and IMA, the corresponding optimal weightings are $\omega = 1$ and

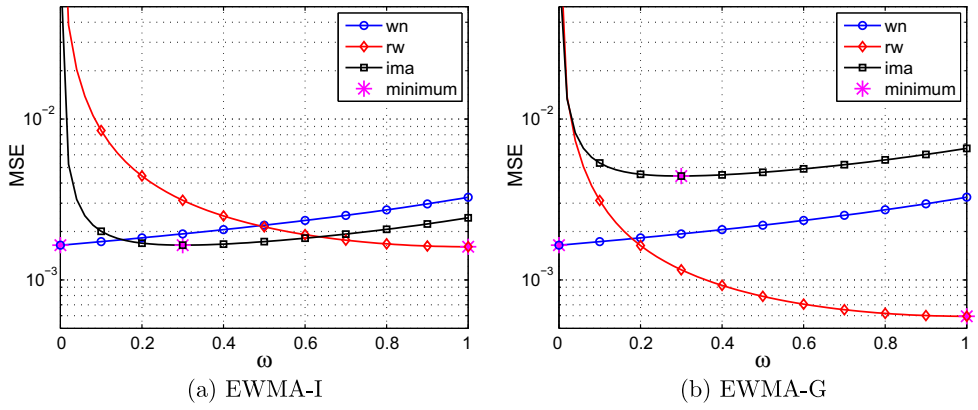


Fig. 3. Effects of EWMA weighting for WN, RW and IMA disturbances.

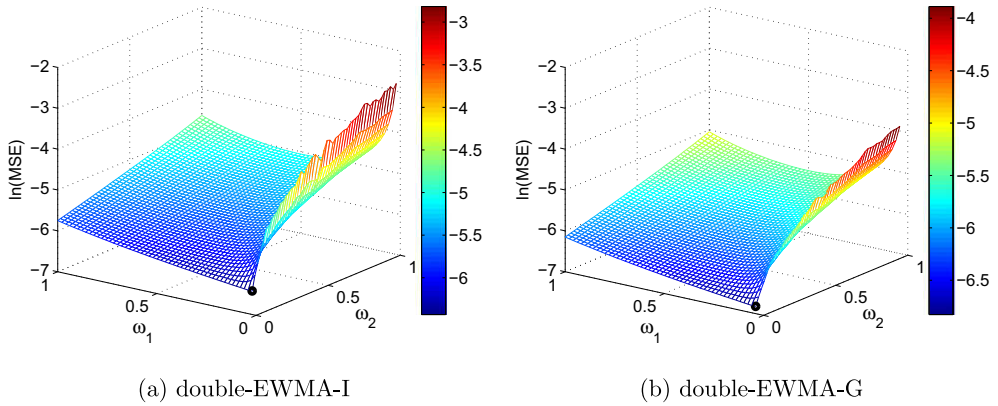
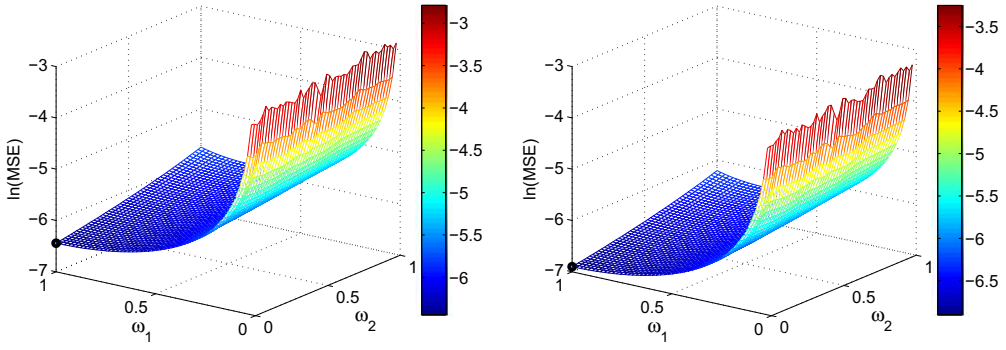


Fig. 4. Effects of double-EWMA weightings in rejecting DD-WN disturbances (minimum MSE, indicated by the circles, obtained when $\omega_1 = \omega_2 = 0$).

$\omega = 1 - \lambda$, respectively, for both EWMA controllers. These results agree with the theoretical optimal weightings obtained in Section 4 by considering state estimation performance only.

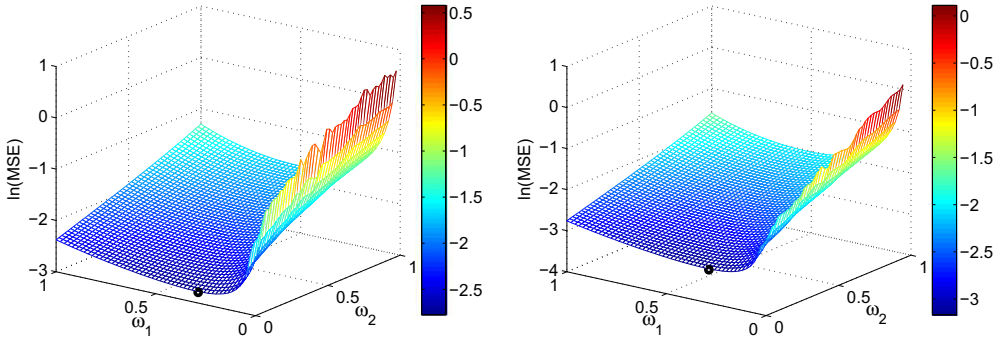
Next we examine the optimal weightings for double-EWMA controllers in rejecting DD-WN, DD-RW and DD-IMA disturbances. In these cases, we search for the optimal ω_1 and ω_2. The results of double-EWMA controllers are shown in Figs. 4–6. The optimal weightings obtained by the simulations (shown as circles in Figs. 4–6) all agree with the theoretical values derived in Section 4. In addition, the simulation results confirm that the double-EWMA weightings providing the optimal state estimation performance also provide optimal control performance.

EWMA controllers are also applied to control processes with deterministic drifts. In this example, the drifting slope is quite small ($\delta/\sigma = 0.01$), and the disturbance is reset to the initial state every 500 runs so the deterministic drift never becomes the dominant part. Therefore, the optimal performance obtained from EWMA controllers (both gain and intercept updating) is almost the same as the optimal performance obtained from double-EWMA controllers for the above case study. For processes with larger drift-



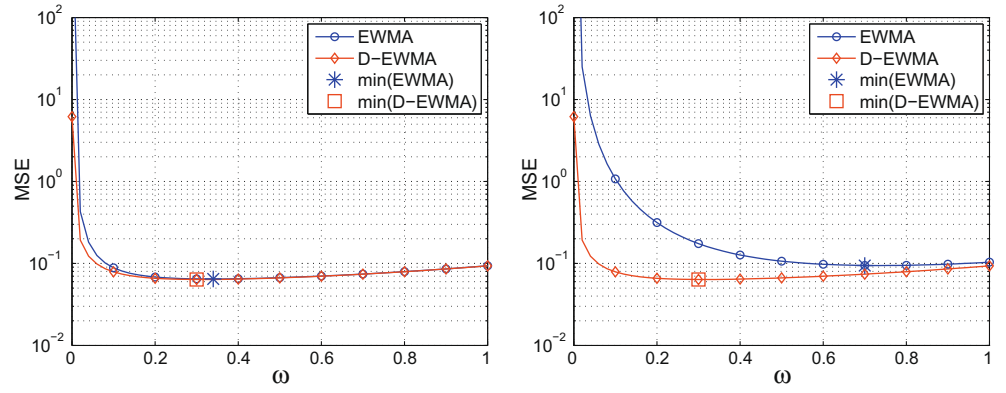
(a) double-EWMA-I (b) double-EWMA-G

Fig. 5. Effects of double-EWMA weightings for DD-RW disturbances (minimum MSE, indicated by the circles, obtained when $\omega_1 = 1$ and $\omega_2 = 0$).



(a) double-EWMA-I (b) double-EWMA-G

Fig. 6. Effects of double-EWMA weightings for DD-IMA disturbances (minimum MSE, indicated by the circles, obtained when $\omega_1 = 1 - \lambda = 0.3$ and $\omega_2 = 0$).



(a) $\delta/\sigma = 0/02$ (b) $\delta/\sigma = 0/2$

Fig. 7. Performance of EWMA and double-EWMA for DD-IMA disturbances (minimum MSE, indicated by the star for EWMA and square for d-EWMA).

ing slopes, the performance difference between EWMA and double-EWMA controllers becomes more apparent. As EWMA controller does not account for the deterministic drift explicitly, there will be a steady-state bias in the process output when it is applied to control a process with deterministic drift disturbance. However, such bias can be reduced by increasing the EWMA weighting as shown in [5], and the optimal tuning is the tread-off between tracking deterministic drift and rejecting stochastic noise. In Fig. 7 we compare the performance of EWMA-I and double-EWMA-I (with $\omega_2 = 0$) controllers when the process disturbance is DD-IMA. Fig. 7a is the case with slow process drift ($\delta/\sigma = 0.02$) and Fig. 7b is the case with fast process drift ($\delta/\sigma = 0.2$). It can be seen that for process with relatively slow drift, EWMA controller is sufficient, but for process with relatively large drift, the benefit of double-EWMA is apparent.

It is worth noting that Figs. 3–7 indicate the performances of EWMA and d-EWMA controllers do not degrade much in a relatively large neighborhood of the optimal tuning, except for DD-WN disturbance. As deterministic drift with pure white noise disturbance is rare in practice, these simulation results suggest that in practice the EWMA controllers can always be tuned to be slightly conservative to handle model/plant mismatch and other type of disturbances, with little sacrifice on the control performance. In addition, for processes with deterministic drifts, as long as the drift is not significant, a EWMA controller can obtain satisfactory control performance which simplifies controller implementation and tuning. Last but not least, when a double-EWMA is necessary to address the more apparent deterministic drift, the weighting corresponding to the deterministic drift (i.e., ω_2) should be kept small if a reliable estimate of the deterministic drift can be obtained from historical data, as the other EWMA weighting (i.e., ω_1) can effectively address stochastic drifts.

6. Conclusions and discussion

In this work, we systematically study the stability and optimal tuning of two types of EWMA controllers, intercept adaptation and gain adaptation, in terms of rejecting different disturbances that are encountered in semiconductor manufacturing industry.

By formulating the closed-loop system controlled by EWMA controllers into the proposed state-space representations, different types of disturbances are viewed as the driving functions to the closed-loop system, which straightforwardly explains the observations that the stability conditions of both EWMA-G and EWMA-I controllers are independent from the types of disturbances. In addition, the proposed formulation reveals that the different system responses of EWMA-I and EWMA-G controllers of the unstable closed-loop system are due to their different output equations (linear vs. nonlinear).

For sensitivity analysis, we propose a general framework to separate the state estimation from the feedback control law, and determine the optimal tuning based on the state estimation performance. Such decoupling greatly simplifies the analysis. By formulating different disturbances into state-space representations, we use the Kalman filter formulation to derive the optimal state estimator for different disturbances. We show that for process disturbances without a deterministic drift trend, an EWMA controller with optimal weighting provides the optimal performance, while for process disturbances with deterministic drift, a double-EWMA controller with optimal weightings provides the optimal performance. The derived optimal conditions are verified by simulation examples, which also suggest that in general the EWMA controller can be tuned to be slightly conservative without sacrificing the control performance noticeably. In addition, a EWMA controller can be implemented for a process with deterministic drift as long as the drift slope is not significant.

It is worth noting that metrology delay will affect both the stability region and optimal tuning parameters for the EWMA controllers. The effect of metrology delay itself is an active research area and many results are available [7,15,16]. Generally speaking, with increasing metrology delay, the stability regions shrink for both EWMA controllers, and the EWMA weighting should be tuned to be less aggressive.

Acknowledgement

Financial support from NSF is gratefully acknowledged by JW under Grant CBET-0853983 and QPH under Grant CBET-0853748.

Appendix A. Optimal D-EWMA weighting for DD-WN disturbance

When $\omega_1 = \omega_2 = 0$, the double-EWMA estimate become

$$\xi_1[n+1] = \xi_1[n] + \xi_2[n] \quad (57)$$
$$\xi_2[n+1] = \xi_2[n] \quad (58)$$
$$\hat{z}[n+1] = \xi_1[n+1] + \xi_2[n+1] \quad (59)$$

The optimal estimate obtained from the Kalman filter, i.e., Eqs. (51) and (52), can be expended to the following:

$$\hat{\theta}_1[n+1] = \hat{\theta}_1[n] \quad (60)$$
$$\hat{\theta}_2[n+1] = \hat{\theta}_1[n] + \hat{\theta}_2[n] \quad (61)$$
$$\hat{z}[n+1] = \hat{\theta}_1[n] + \hat{\theta}_2[n] \quad (62)$$

comparing Eqs. (57)–(59) with Eqs. (60)–(62), we see that

$$\xi_2[n+1] = \hat{\theta}_1[n] \quad (63)$$
$$\xi_1[n+1] = \hat{\theta}_2[n] \quad (64)$$
$$\hat{z}[n+1] = \xi_1[n+1] + \xi_2[n+1] = \hat{\theta}_1[n] + \hat{\theta}_2[n] \quad (65)$$

In other words, the optimal estimate of a DD-WN disturbance is obtained from a double-EWMA estimate with $\omega_1 = \omega_2 = 0$.

Appendix B. Optimal D-EWMA weighting for DD-IMA disturbance

When $\omega_1 = 1 - \lambda$ and $\omega_2 = 0$, the double-EWMA estimate become

$$\xi_1[n+1] = (1-\lambda)z[n] + \lambda(\xi_1[n] + \xi_2[n]) = (1-\lambda)z[n] + \lambda\hat{z}[n] \quad (66)$$
$$\xi_2[n+1] = \xi_2[n] \quad (67)$$
$$\hat{z}[n+1] = \xi_1[n+1] + \xi_2[n+1] = (1-\lambda)z[n] + \lambda\hat{z}[n] + \xi_2[n] \quad (68)$$

Comparing the above double-EWMA estimates with the corresponding estimate from the Kalman filter, i.e., Eq. (56), we observe that,

$$\xi_1[n+1] = \hat{\theta}_2[n+1] - \hat{\theta}_1[n+1] \quad (69)$$
$$\xi_2[n+1] = \hat{\theta}_1[n+1] = \delta \quad (70)$$
$$\hat{z}[n+1] = \xi_1[n+1] + \xi_2[n+1] = \hat{\theta}_2[n+1] \quad (71)$$

In other words, the optimal estimate obtained from the Kalman filter is equivalent to the double-EWMA estimates with $\omega_1 = 1 - \lambda$ and $\omega_2 = 0$.

References

[1] E. Sachs, A. Hu, A. Ingolfsson, Run by run process control: combining SPC and feedback control, IEEE Trans. Semicond. Manuf. 8 (1) (1995) 26–43.

[2] J. Moyne, E. del Castillo, A.M. Hurwitz, Run-to-Run Control in Semiconductor Manufacturing, CRC Press LLC, Boca Raton, FL, 2001.
[3] E. del Castillo, A. Hurwitz, Run-to-run process control: literature review and extensions, J. Qual. Technol. 29 (2) (1997) 184–196.
[4] D. Boning, W. Moyne, T. Smith, J. Monyne, R. Telfeyan, A. Hurwitz, S. Shellman, J. Taylor, Run by run control of chemical mechanical polishing, IEEE Trans. Semicond. Manuf. 19 (4) (1996) 307–316.
[5] A. Ingolfsson, E. Sachs, Stability and sensitivity of an EWMA controller, J. Qual. Technol. 25 (4) (1993) 271–287.
[6] E. Del Castillo, Some properties of EWMA feedback quality adjustment schemes for drifting disturbances, J. Qual. Technol. 33 (2) (2001) 153–166.
[7] R.P. Good, S.J. Qin, On the stability of MIMO EWMA run-to-run controllers with metrology delay, IEEE Trans. Semicond. Manuf. 19 (1) (2006) 78–86.
[8] N. Patel, G. Miller, C. Guinn, A. Sanchez, S. Jenkins, Device dependent control of chemical mechanical polishing of dielectric films, IEEE Trans. Semicond. Manuf. 13 (3) (2000) 331–343.
[9] J. Wang, Q.P. He. A new run-to-run method for oxide cmp processes, in: Proceedings of SPIE's International Symposium on Advanced Microelectronic Manufacturing, vol. 5755, 2005, pp. 9–17.
[10] A. Chen, R. Guo, Age-based double EWMA controller and its application to CMP processes, IEEE Trans. Semicond. Manuf. 14 (1) (2001) 11–19.
[11] J. Wang, Q.P. He, A new bayesian approach for fast disturbance detection and classification in microelectronics manufacturing, IEEE Trans. Semicond. Manuf. 20 (2) (2007) 126–136.
[12] S. Butler, J. Stefani, Supervisory run-to-run optimizing controller for linear and nonlinear semiconductor processes, IEEE Trans. Semicond. Manuf. 7 (2) (1994) 193–201.
[13] J. Wang, Q.P. He, S.J. Qin, C. Bode, M. Purdy, Recursive least squares estimation for run-to-run control with metrology delay and its application to sti etch process, IEEE Trans. Semicond. Manuf. 18 (2) (2005) 309–319.
[14] J. Wang, Q.P. He. EWMA run-to-run controllers with gain updating: stability and sensitivity analysis, in: Proceedings of 2008 American Control Conference, Seattle, WA, 2008, pp. 2872–2877.
[15] A.J. Su, C.C. Yu, B.A. Ogunnaike, On the interaction between measurement strategy and control performance in semiconductor manufacturing, J. Process Control 18 (2008) 166–176.
[16] A.V. Prabhu, T.F. Edgar, Performance assessment of run-to-run EWMA controllers, IEEE Trans. Semicond. Manuf. 20 (4) (2007) 381–385.

Computers and Chemical Engineering xxx (2017) xxx–xxx

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Dynamic latent variable analytics for process operations and control

Yining Dong[a,b], S. Joe Qin[a,b,c,*]

[a] Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA
[b] The Chinese University of Hong Kong, Shenzhen, 2001 Longxiang Ave, Longgang, Shenzhen, Guangdong 518172, China
[c] Mork Family Department of Chemical Engineering and Materials Science, University of Southern California, Los Angeles, CA 90089, USA

ARTICLE INFO

Article history:
Received 11 July 2017
Received in revised form 20 October 2017
Accepted 25 October 2017
Available online xxx

Keywords:
Dynamic PCA
Dynamic PLS
Dynamic CCA
Dynamic latent variable modeling
Process monitoring
Dynamic feature extraction

ABSTRACT

After introducing process data analytics using latent variable methods and machine learning, this paper briefly review the essence and objectives of latent variable methods to distill desirable components from a set of measured variables. These latent variable methods are then extended to modeling high dimensional time series data to extract the most dynamic latent time series, of which the current values are best predicted from the past values of the extracted latent variables. We show with an industrial case study how real process data are efficiently and effectively modeled using these dynamic methods. The extracted features reveal hidden information in the data that is valuable for understanding process variability.

1. Introduction

The McKinsey Global Institute (MGI) (Manyika et al., 2011) states that analyzing large data sets is a key basis of competitiveness, productivity growth, and innovations. The availability of massive amount of data has prompted many disciplines to reexamine their traditional views, such as statistics, management science, econometrics, and engineering. As a result, a new discipline known as data science is forthcoming, which is concerned with deriving knowledge and information from massive data. Several examples show that applying effective analytics to huge amount of data can produce powerful results. The Google's flu prediction is such an example (Ginsberg et al., 2009), which could predict the spread of the winter flu outbreak in 2009 in the United States. The authors took 50 million most common searches on the web and compared them to the Center for Disease Control (CDC) data on the spread of winter flu from 2003 to 2008. They then screened through 150 million models to discover 45 features that had high correlation with the data from CDC. In addition, the resulting model could predict the flu spread nearly in real time, while CDC's data took weeks to compile. Although this data analytic approach is new to the area of process systems engineering (PSE), the application is analogous to inferential sensors (Tham et al., 1991; Qin and McAvoy, 1992).

The recent success of neural networks as a promising approach to artificial intelligence (AI) has made it a forefront for engineering and technology development. This is, however, at least the third time for neural networks to become an active area of AI, with the first one in the 60s and the second one in the late 80s (Hinton, 1989). The process systems engineering (PSE) area participated actively in the second wave of neural networks, as is well documented in the Proceedings of the Fourth Chemical Process Control Conference with a session on *Learning Systems, Adaptive and AI Control* (Hopfield, 1990; Venkatasubramanian, 1991; Stephanopoulos, 1991; Morris et al., 1991). Unfortunately, the PSE field has virtually opted out from AI and machine learning since the mid-90s. What has crystallized in PSE is the inferential estimation of product properties and perhaps model predictive control based on multi-layer neural networks (Tham et al., 1991; Åström and McAvoy, 1992; Qin and Badgwell, 2003).

Nevertheless, the area of AI did not stop its progress. Being one of the early pioneers in back-propagation learning, G.E. Hinton in Hinton and Salakhutdinov (2006) developed deep learning techniques that outperformed the alternatives in image recognition by employing many layers to form a deep network structure (Bengio et al., 2007). The Google's DeepMind team (Silver et al., 2016) developed AlphaGo based on deep learning and tree search to beat the best Go players in the world. It is fair to say that, the recent advancements of neural networks are not vastly different from their predecessors, other than the employment of the deep network structure and massive data. Statistical machine learning (SML) is a

* Corresponding author.
E-mail address: sqin@usc.edu (S.J. Qin).

https://doi.org/10.1016/j.compchemeng.2017.10.029
0098-1354/

rich area that produces statistically-sound machine learning theory and techniques, including support vector machines (SVM) (Vapnik, 2013), variable selection techniques such as Lasso and elastic net (Zou and Hastie, 2005), the kernel method by Freund and Schapire (1999). SML establishes a theoretical basis for machine learning methods to control the errors in inference (National Research Council, 2013). For an overview of machine learning techniques and their potential implication in PSE, one overview is provided in Lee et al. (2017).

Industrial process data are massive and high dimensional due to the complexity of process operations and control. Although these measurements are high dimensional, the measured variables usually do not act independently due to process operation requirements and physical constraints. These data are often highly collinear and correlated, making traditional regression methods such as least squares unreliable due to ill-conditioning. Regularized least squares methods such as ridge regression can be tuned to achieve reliable prediction with the bias and variance trade-off. However, these models are not easily interpretable.

To analyze these high-dimensional and correlated data effectively, latent variables methods (LVM), including principal component analysis (PCA), projection to latent structures (PLS), and canonical correlation analysis (CCA) are preferred choices. For brevity of the paper, we will not provide a detailed review of the latent variable methods in process applications. Interested readers should refer to MacGregor and Kourti (1995), Wise and Gallagher (1996), and Qin (2003).

In the remainder of this paper we offer a brief introduction to the essence and objectives of latent variable analytics. We then present dynamic latent variable methods for the modeling of multidimensional time series data for prediction, diagnosis, and feature analysis. The methods are demonstrated on an industrial process data set to extract features that are best predicted by their past and are easily visualized. The features are attributed to process malfunctions or anomalies that need to be eliminated. In concluding the paper we adopt an open mindset towards embracing the power of new machine learning techniques that have enjoyed tremendous development in recent years.

2. Data analytics using latent variables

In this section we review the traditional latent variable methods that are effective static process data analytics (PDA). First we give the context in which the process and quality data are collected and monitored. Then we illustrate the objectives of each LVM and comment on their advantages and shortcomings. Lastly we give an analogy of latent-variable modeling that extracts components according to the objectives to that of a distillation process that separates chemical components.

2.1. Latent variable modeling methods

Popular latent variable methods include PCA, PLS, CCA, and their extensions. The common characteristics of these methods are their ability to reduce dimensions and concentrate relevant features in a reduced-dimensional space. The objective of PCA is to represent a set of correlated variables with a limited number of latent variables that are most representative of the original variables. Without any prior requirement it is natural to require that the latent variables capture the largest variation in the original data and, therefore, the residuals will be minimal. The extracted latent variables or principal components (PC) can be easily visualized in a low dimensional space and interpreted with process knowledge behind the observed data. In these models, the measured data are observations of the underlying latent variables that are not directly measured.

Let $\boldsymbol{x} \in \mathbb{R}^M$ denote a sample vector of M variables. Assuming that there are N observations for each variable, a data matrix $\mathbf{X}$ is composed with N rows (observations) as follows

$$\mathbf{X} = [\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \cdots \quad \boldsymbol{x}_N]^T$$

It is a usual practice that variables are scaled to zero mean and unit variance. Principal component analysis extracts a direction or subspace of the largest variance in the M dimensional measurement space. For an arbitrary vector direction $\mathbf{p}$ such that $\|\mathbf{p}\|^2 = 1$, the projection of $\mathbf{X}$ on to this direction is $\mathbf{t} = \mathbf{Xp}$. The PCA objective is to maximize the variance of the projection, that is,

$$\max \mathbf{t}^T\mathbf{t} = \mathbf{p}^T\mathbf{X}^T\mathbf{Xp} \tag{1}$$

The solution to the above problem with $\|\mathbf{p}\|^2 = 1$ as a constraint can be obtained using a Lagrange multiplier as follows

$$\mathbf{X}^T\mathbf{Xp} = \lambda\mathbf{p} \tag{2}$$

which implies that $\mathbf{p}$ is the eigenvector corresponding to the largest eigenvalue of $\mathbf{X}^T\mathbf{X}$, which would be the sample covariance matrix of $\mathbf{X}$ if divided by $(N-1)$. The vector $\mathbf{p}$ is known as the loading vector for the first principal component. After the first component is extracted, it is removed from the data matrix and the same eigen-decomposition procedure is iterated, that is,

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{t}_i\mathbf{p}_i^T \tag{3}$$

The data matrix $\mathbf{X} = \mathbf{X}_1$ and is decomposed as follows,

$$\mathbf{X} = \sum_{i=1}^{l} \mathbf{t}_i\mathbf{p}_i^T + \mathbf{X}_{l+1} \tag{4}$$

If the data matrix $\mathbf{X}$ contains highly correlated columns, it will take fewer than M components to exhaust the variance in the residual $\mathbf{X}_{l+1}$. The eigenvalues correspond to the variance of the extracted PC scores, $\mathbf{t}_i$, which is extracted one after another.

Partial least-squares methods find a latent structure between two data matrices, $\mathbf{X}$ and $\mathbf{Y}$, collected from input variables and output variables, such that the respective score vectors

$$\mathbf{t} = \mathbf{Xw}$$
$$\mathbf{u} = \mathbf{Yq}$$

have maximized covariance. Mathematically, this is expressed as

$$\max_{\mathbf{t},\mathbf{u}} J = \mathbf{t}^T\mathbf{u} \tag{5}$$

subject to the constraint that the weighting vectors $\mathbf{w}$ and $\mathbf{q}$ have unit norm,

$$\|\mathbf{w}\|^2 = 1$$
$$\|\mathbf{q}\|^2 = 1$$

The solution to this problem can also be achieved by using Lagrange multipliers, which lead to an eigen-problem related to the two data matrices. Deflations and iterations are necessary to extract all significant latent variables one after another. Due to the use of a covariance objective function in PLS, PLS usually requires multiple latent variables even for a single output variable in Y. One arguable advantage of requiring multiple LVs is that the PLS method exploits variance of the input while trying to interpret the output. This is, in fact, trying to achieve two objectives at once, which can make both objectives compromised. For instance, there is usually a subspace of the PLS latent subspace that is orthogonal to the output. Although that subspace contains significant variability of the input data space, it is irrelevant to the output. This is the motivation of several recent efforts to develop orthogonalized PLS (Trygg and Wold, 2002) and concurrent PLS methods (Qin and Zheng, 2013).

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

Another latent variable objective is the canonical correlation analysis objective developed by Hotelling (1936), which maximizes the correlation between two sets of latent vectors **t** and **u**,

$$\max_{\mathbf{t},\mathbf{u}} J = \frac{\mathbf{t}^T\mathbf{u}}{\|\mathbf{t}\|\|\mathbf{u}\|} \tag{6}$$

which is actually the cosine of the angle between the latent vectors. The solution to this problem is an eigen-vector solution of $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}$. An advantage of the CCA method is that it has maximized efficiency in predicting the output **Y** using variations in **X**. To predict a single output, CCA requires only one latent variable to interpret all variations in the output data. However, due to the inverses of the covariance matrices in the CCA solution, it is sensitive to collinearity among the variables. Therefore, some form of regularization is necessary to make the method insensitive to collinear data. Another issue is that CCA does not pay attention to representing the input variances, other than extracting the portion that is useful in predicting the output. This makes CCA uninterested in exploiting the input variance structure, especially in the input subspace that is orthogonal to the output. The recently developed concurrent CCA (Zhu et al., 2016, 2017) combines CCA and PCA to achieve two objectives, this is, to exploit the variance structure of the input while trying to predict the output efficiently.

The CCA objective in Eq. (6) can be equivalently achieved with the following least squares objective,

$$\min J = \|\mathbf{u} - b\mathbf{t}\|^2 = \|\mathbf{Yq} - b\mathbf{Xw}\|^2 \tag{7}$$

This objective, of course, does not have a minimum unless the weights are constrained in the norm. By restricting $\|\mathbf{Yq}\|^2 = 1$ and $\|\mathbf{Xw}\|^2 = 1$, we have the following Theorem.

Theorem 1. *The least squares objective,*

$$\min J = \|\mathbf{Yq} - b\mathbf{Xw}\|^2$$

subject to $\|\mathbf{Yq}\|^2 = 1$ *and* $\|\mathbf{Xw}\|^2 = 1$ *is equivalent to the canonical correlation analysis objective in Eq.* (6)

The proof of Theorem 1 is given in Appendix A. The least squares objective Eq. (7) gives an explicit interpretation for CCA being a latent variable regression method.

The aforementioned methods exploit latent structured relations among the variables that are linear and static. They form the foundation for extensions to dynamic latent variable modeling. Since all methods have clear objectives factor by factor, they can be interpreted as distilling data into one component after another, with specific objectives and intentions.

2.2. Data distillation vs. data mining

Data mining is a popular term that represents data explorative analysis techniques to find patterns or features from data of social, economic, medical, biological, text, or engineering systems. Many data mining techniques are in the category of unsupervised learning and clustering, with the hope to find or extract interesting features from massive data. For engineering and manufacturing processes, nearly all units, equipment, and subsystems have dedicated purposes to perform certain functions, and data from such processes are required to conform to the design and operational specifications. Therefore, to analyze data from these engineering processes, one often has prior knowledge on what features to explore from the data based on engineering and operational principles.

To better characterize methods that can extract these prespecified features of interest, we use the term *data distillation* to refer to the class of methods that extract certain features from data with a clear order of relevance to the objectives. For example, principal component analysis is a method to distill data into components with a descending order of the variance magnitudes. The variance of the features is clearly defined in the objective of the PCA algorithm. Canonical correlation analysis rank-orders the components by their canonical correlations, from high to low, which are also clearly defined in the CCA objective. In the next section of dynamic data distillation, the components are extracted and rank-ordered in terms of the predictability of the current values from the past ones, which will be explicitly defined in the objective functions.

2.3. Process data analytics and monitoring

The process and quality data considered for process data analytics can be illustrated in Fig. 1, where the hierarchical data structure is shown. At the bottom level are the equipment sensor measurements that can be in milliseconds. At the process level are regularly sampled process control data. The product quality measurements come in all forms and often irregularly sampled. The top level is the customer feedback data that can go from customer service channels to social network complaints. The advantages of the data driven latent structure modeling methods, such as PCA and PLS, are that they can be used to detect abnormal changes in process operations from real time data due to the dimension-reduction capability, ease of visualization, and ease of interpretation. The related fault diagnosis methods have been intensively studied and applied successfully in many industrial processes, e.g. chemicals, iron and steel, polymers, and semiconductor manufacturing.

Process data are often categorized into process input and output data, quality output data, and indirect (e.g., vibration signals and images) types of data, as shown in Fig. 1. The typical procedure of the multivariate process monitoring is given as follows and can be found in, e.g., MacGregor et al. (1994), Qin (2012), Chiang et al. (2000), and Cinar et al. (2007).

- Collection of normal data with good coverage of the operating regions
- Fault data cases can be useful, but not required
- Latent variable methods (PCA, PLS, etc.) to model the data
- Fault detection indices and control limits, such as the Hotelling's T^2 and the squared prediction error indices
- Fault diagnosis and troubleshooting, such as reconstruction-based fault identification and contribution analysis.

3. Dynamic latent variable analytics

Since a vast amount of process data are collected in the form of time series, with sampling intervals from seconds to milliseconds, dynamics or time correlations are often strong among the data. These dynamics make static data analytics inadequate, but they should be modeled appropriately so that they can be useful for prediction and monitoring. Considering the large dimensional time series data that are both cross-correlated and auto-correlated over time, it is necessary to develop dynamic extensions of the latent variables methods such that their current values are best predicted by their past data, using a reduced number of dynamic latent variables. The extracted data of these dynamic latent variables are referred to as *principal time series* with reduced dimensions. In this section we present the dynamic-inner PCA algorithm (Dong and Qin, 2017), dynamic-inner PLS algorithm (Dong and Qin, 2015), and a novel dynamic-inner CCA algorithm.

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

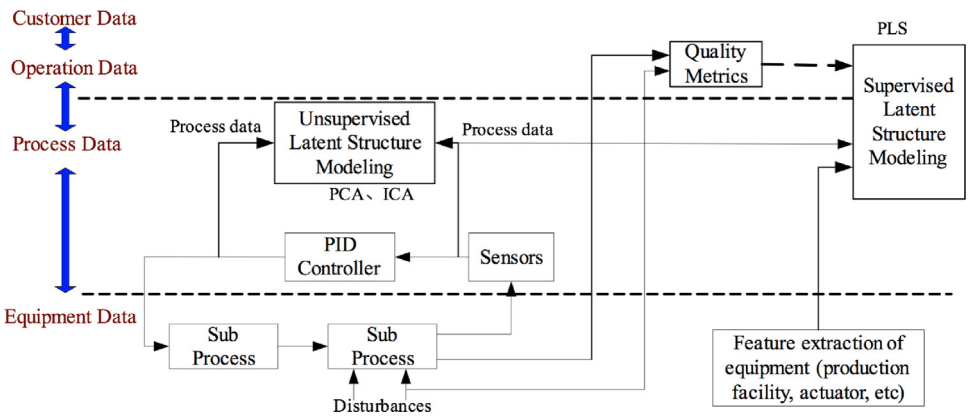


Fig. 1. Process and quality data collected under a process and control hierarchy.

3.1. PCA with dynamic latent variables

Dynamic-inner principal component analysis (DiPCA) builds most dynamic relations of the inner latent variables that have maximized auto-covariance. DiPCA extracts one latent variable after another that is a linear combination of the original variables with which the current values are in a sense best predicted from their past values. As a consequence, the residuals after extracting the most predictable latent variables from the data will be least predictable and, in the limiting case, tend to be white noise. The method overcomes drawbacks of existing dynamic PCA methods that simply perform static PCA on augmented data with time lags (e.g., Ku et al., 1995).

The advantages of the DiPCA algorithm that extracts principal time series are that (i) the dynamic components can be predicted from their past data as known information, so the uncertainty is on the prediction errors only; (ii) the extracted dynamic components can highlight useful dynamic features for data interpretation and diagnosis, which are otherwise difficult to observe from the original data; and (iii) the prediction errors after all dynamics are effectively extracted are essentially not time-correlated and can be further modeled as static data with traditional PCA method.

In general, we wish to extract dynamics in a latent variable t_k so that the current value can be predicted from the past using the following auto-regressive (AR) model,

$$t_k = \beta_1 t_{k-1} + \cdots + \beta_s t_{k-s} + r_k \tag{8}$$

with the latent variable as a linear combination of the original variables $t_k = \boldsymbol{x}_k^T \mathbf{w}$. The prediction from the dynamic inner model is

$$\begin{aligned} \hat{t}_k &= \boldsymbol{x}_{k-1}^T \mathbf{w}\beta_1 + \cdots + \boldsymbol{x}_{k-s}^T \mathbf{w}\beta_s \\ &= \begin{bmatrix} \boldsymbol{x}_{k-1}^T & \cdots & \boldsymbol{x}_{k-s}^T \end{bmatrix} (\boldsymbol{\beta} \otimes \mathbf{w}) \end{aligned} \tag{9}$$

where $\boldsymbol{\beta} = [\beta_1 \ \beta_2 \ \cdots \ \beta_s]^T$ and $\mathbf{w}$ are constrained to be unit norm without loss of generality. The objective of the dynamic inner PCA algorithm is to maximize the covariance between the extracted data and the prediction, that is

$$\max_{\mathbf{w},\boldsymbol{\beta}} \quad \frac{1}{N} \sum_{k=s+1}^{s+N} \mathbf{w}^T \boldsymbol{x}_k \begin{bmatrix} \boldsymbol{x}_{k-1}^T & \cdots & \boldsymbol{x}_{k-s}^T \end{bmatrix} (\boldsymbol{\beta} \otimes \mathbf{w}) \tag{10}$$

for $(N+s)$ observations. Denoting the data matrix as

$$\mathbf{X} = [\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \cdots \quad \boldsymbol{x}_{N+s}]^T$$

and forming the following data matrices,

$$\begin{aligned} &\mathbf{X}_i = [\boldsymbol{x}_i \quad \boldsymbol{x}_{i+1} \quad \cdots \quad \boldsymbol{x}_{N+i-1}]^T \text{ for } i = 1, 2, \ldots, s+1 \\ &\mathbf{Z}_s = [\mathbf{X}_s \quad \mathbf{X}_{s-1} \quad \cdots \quad \mathbf{X}_1] \end{aligned} \tag{11}$$

Dong and Qin (2017) reformulate the objective Eq. (10) of DiPCA as

$$\begin{aligned} &\max_{\mathbf{w},\boldsymbol{\beta}} \ \mathbf{w}^T \mathbf{X}_{s+1}^T \mathbf{Z}_s (\boldsymbol{\beta} \otimes \mathbf{w}) \\ &\text{s.t. } \|\mathbf{w}\| = 1, \|\boldsymbol{\beta}\| = 1 \end{aligned} \tag{12}$$

where the scalar $1/N$ is omitted. The complete DiPCA algorithm derived in Dong and Qin (2017) is summarized in Appendix B.

This DiPCA can also be viewed as a whitening filter applied to the data. After all DiPCA components are extracted, the prediction errors are essentially white as virtually all the dynamic relationships in data are extracted. An important notion of this whitening filter is that it has a reduced number of latent variables compared to the number of variables in that data, and is appropriate for modeling the common case of highly collinear data from real world problems. This solution is different from a full dimensional vector autoregressive (VAR) model that requires to invert a covariance matrix, which can be ill-conditioned with highly correlated data. Furthermore, the DiPCA latent variables have a clear objective and can provide useful features for data based interpretation, visualization, and diagnosis.

3.2. PLS with dynamic latent variables

The PLS algorithm performs regression with inter-related variables by projecting the data to a lower dimensional latent space one dimension at a time. This approach not only avoids direct inversion of a potentially ill-conditioned matrix in ordinary least squares, it also provides a way to trade off between the model prediction variance and bias by selecting an appropriate number of latent variables.

The objective of PLS only focuses on static relations in the input and output data. To build a dynamic inner PLS model, the objective should be changed to extracting a dynamic latent relation such as,

$$u_k = \beta_1 t_k + \beta_2 t_{k-1} + \cdots + \beta_s t_{k-s+1} + r_k \tag{13}$$

with the latent variables related to data as follows

$$\begin{aligned} u_k &= \boldsymbol{y}_k^T \mathbf{q} \\ t_k &= \boldsymbol{x}_k^T \mathbf{w} \end{aligned}$$

For each factor, the inner model prediction should be

$$\begin{aligned} \hat{u}_k &= \boldsymbol{x}_k^T \mathbf{w}\beta_1 + \cdots + \boldsymbol{x}_{k-s+1}^T \mathbf{w}\beta_s \\ &= \begin{bmatrix} \boldsymbol{x}_k^T & \cdots & \boldsymbol{x}_{k-s+1}^T \end{bmatrix} (\boldsymbol{\beta} \otimes \mathbf{w}) \end{aligned}$$

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

The dynamic inner PLS (DiPLS) algorithm from Dong and Qin (2015) maximizes the covariance between the latent scores u_k and its prediction $\hat{u}_k$ as follows,

$$\max_{\mathbf{q},\mathbf{w},\boldsymbol{\beta}} \frac{1}{N+1}\sum_{k=s}^{s+N}\mathbf{q}^T\boldsymbol{y}_k\left[\boldsymbol{x}_k^T \quad \cdots \quad \boldsymbol{x}_{k-s+1}^T\right](\boldsymbol{\beta}\otimes\mathbf{w}) \tag{14}$$

This objective to solve for the model vectors **q, w** and $\boldsymbol{\beta}$ clearly contains latent dynamics, while retaining outer projections of the input and output data to the latent variable dimension. For the special case of $s=1$, DiPLS reduces to the static PLS.

For $(N+s)$ observations of input and output data we form the following data matrices

$$\mathbf{X}_i = [\boldsymbol{x}_i \quad \boldsymbol{x}_{i+1} \quad \cdots \quad \boldsymbol{x}_{N+i}]^T \text{ for } i = 1, 2, \ldots, s$$
$$\mathbf{Z}_s = [\mathbf{X}_s \quad \mathbf{X}_{s-1} \quad \cdots \quad \mathbf{X}_1]$$
$$\mathbf{Y}_s = [\boldsymbol{y}_s \quad \boldsymbol{y}_{s+1} \quad \cdots \quad \boldsymbol{y}_{s+N}]^T$$

The objective of DiPLS can be represented as

$$\begin{aligned}\max_{\mathbf{q},\mathbf{w},\boldsymbol{\beta}} \quad & \mathbf{q}^T\mathbf{Y}_s^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) \\ \text{s.t.} \quad & \|\mathbf{w}\| = 1, \|\mathbf{q}\| = 1, \|\boldsymbol{\beta}\| = 1\end{aligned} \tag{15}$$

Lagrange multipliers are used to solve this optimization problem, which yields the DiPLS algorithm (Dong and Qin, 2015) as given in Appendix C.

3.3. CCA with dynamic latent variables

The DiPCA and DiPLS algorithms build inherent dynamics in the latent variables with explicit projections from the data space to the latent space. However, their objective functions, as illustrated in Section 2 for the static counterparts, are not as efficient as the CCA objective in maximizing prediction with the least latent dimensions. To obtain a principal time series that can be best predicted from its past values, we propose a dynamic-inner CCA (DiCCA) algorithm that maximizes the correlation.

3.3.1. DiCCA objective

Mathematically, we wish to ensure that the dynamic latent variable t_k is best predicted by $\hat{t}_k$. This is done by maximizing the correlation between t_k and $\hat{t}_k$, which is represented as

$$\max \frac{\sum_{k=s+1}^{s+N} t_k\hat{t}_k}{\sqrt{\sum_{k=s+1}^{s+N} t_k^2}\sqrt{\sum_{k=s+1}^{s+N}\hat{t}_k^2}} \tag{16}$$

It can be shown that when restricting $\sum_{k=s+1}^{s+N} t_k^2 = 1$ and $\sum_{k=s+1}^{s+N}\hat{t}_k^2 = 1$, maximizing (16) is equivalent to minimizing $\sum_{k=s+1}^{s+N}(t_k-\hat{t}_k)^2$, the residual sum of squares of the prediction model under these constraints. Therefore, with the same prediction model and matrix notation as in Eqs. (9) and (11), the objective (16) can be rewritten as

$$\max \frac{\mathbf{w}^T\mathbf{X}_{s+1}^T(\mathbf{X}_s\mathbf{w}\beta_1 + \cdots + \mathbf{X}_1\mathbf{w}\beta_s)}{||\mathbf{X}_{s+1}\mathbf{w}||||\mathbf{X}_s\mathbf{w}\beta_1 + \cdots + \mathbf{X}_1\mathbf{w}\beta_s||} \tag{17}$$

which can be reformulated as the following DiCCA objective function

$$\begin{aligned}\max_{\mathbf{w},\boldsymbol{\beta}} \quad & J = \mathbf{w}^T\mathbf{X}_{s+1}^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) \\ \text{s.t.} \quad & \|\mathbf{X}_{s+1}\mathbf{w}\| = 1, \|\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w})\| = 1\end{aligned} \tag{18}$$

where $\mathbf{X}_i$'s and $\mathbf{Z}_s$ are defined in Eq. (11).

3.3.2. Extracting one dynamic correlation component

To solve the optimization problem in (18), Lagrange multipliers are applied. Define

$$\begin{aligned}L \quad &= \mathbf{w}^T\mathbf{X}_{s+1}^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) + \frac{1}{2}\lambda_1(1-\mathbf{w}^T\mathbf{X}_{s+1}^T\mathbf{X}_{s+1}\mathbf{w}) \\ &\quad + \frac{1}{2}\lambda_2(1-(\boldsymbol{\beta}\otimes\mathbf{w})^T\mathbf{Z}_s^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}))\end{aligned}$$

Making use of the following identities

$$(\boldsymbol{\beta}\otimes\mathbf{w}) = (\boldsymbol{\beta}\otimes\mathbf{I})\mathbf{w} = (\mathbf{I}\otimes\mathbf{w})\boldsymbol{\beta}$$

and taking derivatives of L with respective to **w** and $\boldsymbol{\beta}$ and set them to zero respectively, we have

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{w}} \quad &= \mathbf{X}_{s+1}^T\mathbf{Z}_s\left(\boldsymbol{\beta}\otimes\mathbf{w}\right) + (\boldsymbol{\beta}\otimes\mathbf{I})^T\mathbf{Z}_s^T\mathbf{X}_{s+1}\mathbf{w} \\ &\quad -\lambda_1\mathbf{X}_{s+1}^T\mathbf{X}_{s+1}\mathbf{w} - \lambda_2(\boldsymbol{\beta}\otimes\mathbf{I})^T\mathbf{Z}_s^T\mathbf{Z}_s\left(\boldsymbol{\beta}\otimes\mathbf{I}\right)\mathbf{w} = 0\end{aligned} \tag{19}$$

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{\beta}} \quad &= (\mathbf{I}\otimes\mathbf{w})^T\mathbf{Z}_s^T\mathbf{X}_{s+1}\mathbf{w} \\ &\quad -\lambda_2(\mathbf{I}\otimes\mathbf{w})^T\mathbf{Z}_s^T\mathbf{Z}_s\left(\mathbf{I}\otimes\mathbf{w}\right)\boldsymbol{\beta} = 0\end{aligned} \tag{20}$$

Pre-multiplying Eq. (20) by $\boldsymbol{\beta}^T$ and using the constraint in Eq. (18), we have $J=\lambda_2$. Pre-multiplying Eq. (19) by $\mathbf{w}^T$ and refer to the constraint, we have $2J-\lambda_1-\lambda_2=0$, leading to $\lambda_1=J$. Therefore, we let $\lambda_1=\lambda_2=\lambda$. In addition, defining

$$\mathbf{T}_s = \mathbf{Z}_s(\mathbf{I}\otimes\mathbf{w}) = [\mathbf{X}_s\mathbf{w} \quad \mathbf{X}_{s-1}\mathbf{w} \quad \cdots \quad \mathbf{X}_1\mathbf{w}] = [\mathbf{t}_s \quad \mathbf{t}_{s-1} \quad \cdots \quad \mathbf{t}_1] \tag{21}$$

$$\hat{\mathbf{X}}_{s+1} = \mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{I}) = \sum_{i=1}^{s}\beta_i\mathbf{X}_{s-i+1} \tag{22}$$

we have

$$\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) = \sum_{i=1}^{s}\beta_i\mathbf{X}_{s-i+1}\mathbf{w} = \hat{\mathbf{X}}_{s+1}\mathbf{w} = \hat{\mathbf{t}}_{s+1} \tag{23}$$

where

$$\begin{aligned}&\mathbf{t} = \mathbf{X}\mathbf{w} \in \mathbb{R}^{s+N} \\ &\mathbf{t}_i = \mathbf{X}_i\mathbf{w} \in \mathbb{R}^N, \text{ for } i = 1, 2, \ldots, s+1\end{aligned} \tag{24}$$

Eq. (20) can be rewritten as

$$\mathbf{T}_s^T\mathbf{X}_{s+1}\mathbf{w} = \lambda\mathbf{T}_s^T\mathbf{T}_s\boldsymbol{\beta}$$

or,

$$(\mathbf{T}_s^T\mathbf{T}_s)^{-1}\mathbf{T}_s^T\mathbf{X}_{s+1}\mathbf{w} = \lambda\boldsymbol{\beta} \tag{25}$$

Similarly, Eq. (19) can be reorganized as follows

$$\mathbf{X}_{s+1}^T\hat{\mathbf{X}}_{s+1}\mathbf{w} + \hat{\mathbf{X}}_{s+1}^T\mathbf{X}_{s+1}\mathbf{w} = \lambda(\mathbf{X}_{s+1}^T\mathbf{X}_{s+1} + \hat{\mathbf{X}}_{s+1}^T\hat{\mathbf{X}}_{s+1})\mathbf{w}$$

or,

$$(\mathbf{X}_{s+1}^T\mathbf{X}_{s+1} + \hat{\mathbf{X}}_{s+1}^T\hat{\mathbf{X}}_{s+1})^{+}(\mathbf{X}_{s+1}^T\hat{\mathbf{X}}_{s+1} + \hat{\mathbf{X}}_{s+1}^T\mathbf{X}_{s+1})\mathbf{w} = \lambda\mathbf{w} \tag{26}$$

where $()^+$ denotes the Moore–Penrose pseudo-inverse. Eqs. (26) and (25) imply that **w** is the eigenvector of $(\mathbf{X}_{s+1}^T\mathbf{X}_{s+1} + \hat{\mathbf{X}}_{s+1}^T\hat{\mathbf{X}}_{s+1})^{+}(\mathbf{X}_{s+1}^T\hat{\mathbf{X}}_{s+1} + \hat{\mathbf{X}}_{s+1}^T\mathbf{X}_{s+1})$ corresponding to the largest eigenvalue. However, since $\hat{\mathbf{X}}_{s+1}$ depends on $\boldsymbol{\beta}$ and therefore $\boldsymbol{\beta}$ and **w** are coupled together, there is no analytical

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

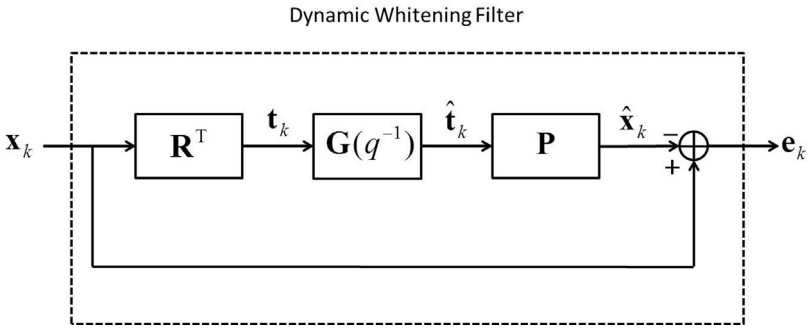


Fig. 2. Dynamic whitening filter structure by DiCCA.

solution to the optimization problem (18). Eqs. (26) and (25) can be reorganized as follows using Eqs. (24) and (23),

$$\begin{aligned}\lambda\mathbf{w} &= \left(\mathbf{X}_{s+1}^T\mathbf{X}_{s+1} + \hat{\mathbf{X}}_{s+1}^T\hat{\mathbf{X}}_{s+1}\right)^+ \left(\mathbf{X}_{s+1}^T\hat{\mathbf{t}}_{s+1} + \hat{\mathbf{X}}_{s+1}^T\mathbf{t}_{s+1}\right)\\ &= \left(\begin{bmatrix}\mathbf{X}_{s+1}\\ \hat{\mathbf{X}}_{s+1}\end{bmatrix}^T\begin{bmatrix}\mathbf{X}_{s+1}\\ \hat{\mathbf{X}}_{s+1}\end{bmatrix}\right)^+\begin{bmatrix}\mathbf{X}_{s+1}\\ \hat{\mathbf{X}}_{s+1}\end{bmatrix}^T\begin{bmatrix}\hat{\mathbf{t}}_{s+1}\\ \mathbf{t}_{s+1}\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{X}_{s+1}\\ \hat{\mathbf{X}}_{s+1}\end{bmatrix}^+\begin{bmatrix}\hat{\mathbf{t}}_{s+1}\\ \mathbf{t}_{s+1}\end{bmatrix}\end{aligned} \tag{27}$$

$$\lambda\boldsymbol{\beta} = (\mathbf{T}_s^T\mathbf{T}_s)^{-1}\mathbf{T}_s^T\mathbf{t}_{s+1} \tag{28}$$

It is clear from Eq. (28) that $\boldsymbol{\beta}$ depends on $\mathbf{w}$ implicitly through $\mathbf{t}_i$. Also, $\boldsymbol{\beta}$ is proportional to the least squares solution of the AR model parameters of time series $\{t_k\}_{k=1,2,\ldots}$. Since $\mathbf{w}$ is an eigenvector based on Eq. (26), the norm of $\mathbf{w}$ is scaled to one. Therefore, we can find $\boldsymbol{\beta}$ to be the least squares solution from Eq. (28). The core DiCCA algorithm is summarized as follows.

1. Initialize $\mathbf{w}$ with a column of the identity matrix.
2. Calculate $\mathbf{w}$, $\boldsymbol{\beta}$ by iterating the following relations until convergence.
   $\mathbf{t} = \mathbf{X}\mathbf{w}$ and form $\mathbf{t}_i$ from $\mathbf{t}$ according to Eq. (24).
   Form $\mathbf{T}_s = [\mathbf{t}_s \cdots \mathbf{t}_1]$.
   $\boldsymbol{\beta} = (\mathbf{T}_s^T\mathbf{T}_s)^{-1}\mathbf{T}_s^T\mathbf{t}_{s+1}$,
   $\hat{\mathbf{X}}_{s+1} = \sum_{i=1}^{s}\beta_i\mathbf{X}_{s-i+1}$,
   $\hat{\mathbf{t}}_{s+1} = \sum_{i=1}^{s}\beta_i\mathbf{t}_{s-i+1}$,
   $\mathbf{w} = \begin{bmatrix}\mathbf{X}_{s+1}\\ \hat{\mathbf{X}}_{s+1}\end{bmatrix}^+\begin{bmatrix}\hat{\mathbf{t}}_{s+1}\\ \mathbf{t}_{s+1}\end{bmatrix}$ and $\mathbf{w} := \mathbf{w}/\|\mathbf{w}\|$
3. Calculate $J = \lambda = \frac{\hat{\mathbf{t}}_{s+1}^T\mathbf{t}_{s+1}}{\|\hat{\mathbf{t}}_{s+1}\|\|\mathbf{t}_{s+1}\|}$.

To extract the next dynamic latent variable, the same iteration procedure can be applied to the deflated matrices of $\mathbf{X}_{s+1}$ and $\mathbf{Z}_s$, which will be discussed in the next subsection.

3.3.3. Deflation

After the loading vector $\mathbf{w}$ and latent scores $\mathbf{t}$ are obtained from the iteration procedure, $\mathbf{X}$ is deflated as

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T \tag{29}$$

where the loading vector $\mathbf{p}$ is defined as

$$\mathbf{p} = \mathbf{X}^T\mathbf{t}/\mathbf{t}^T\mathbf{t} \tag{30}$$

The deflated matrix $\mathbf{X}$ is then used to repeat the same iteration procedure to extract the next latent variable. The deflation procedure leads to desirable geometric properties of DiCCA algorithm, which is analyzed in Dong and Qin (2018). For instance, the score vectors $\mathbf{t}$ from different latent variables are orthogonal.

3.4. DiCCA model relations

After obtaining the latent variable t_k, an AR model can be built to describe the dynamics in t_k as

$$t_k = \alpha_1 t_{k-1} + \cdots + \alpha_s t_{k-s} + \varepsilon_k \tag{31}$$

The solution to estimate $\boldsymbol{\alpha}$ is ordinary least squares, which coincidentally is already solved in the iterative algorithm as $\boldsymbol{\beta}$. Therefore, there is no need to fit another AR model.

Compared to other dynamic data modeling algorithms such as DiPLS (Dong and Qin, 2015), a re-estimation of $\boldsymbol{\beta}$ has to be done after the outer model projection. The extraction of the latent variables and dynamic modeling of the latent variables are achieved simultaneously in DiCCA, because DiCCA employs consistent outer modeling and inner modeling objectives. This is a unique property of DiCCA and makes it a more efficient dynamic modeling algorithm than the others.

3.4.1. DiCCA model with l components

The DiCCA algorithm extracts latent time series one by one with descending predictability or R^2 values. After l latent time series are extracted, the next latent time series extracted will have a R^2 value close to 0, which implies that there are little or no dynamics left in the residuals. The orthogonality of the latent scores guarantees that the number of latent time series required to extract all dynamics is fewer than the number of variables, which will be shown later in the paper. Mathematically, by using $t_k^{(j)}$ to denote the jth latent score at time k, and β_{ji} for $i = 1, 2, \ldots s$ to denote the AR coefficients for the jth latent score, we have the prediction model for each score as

$$\begin{aligned}\hat{t}_k^{(j)} &= (\beta_{j1}q^{-1} + \beta_{j2}q^{-2} + \cdots + \beta_{js}q^{-s})t_k^{(j)}\\ &= G_j(q^{-1})t_k^{(j)}\end{aligned} \tag{32}$$

where q^{-1} is the backward shift operator. By combining l prediction models together, we can obtain a prediction model for the latent score vector $\boldsymbol{t}_k = [t_k^{(1)} \quad t_k^{(2)} \quad \cdots \quad t_k^{(l)}]^T$ as

$$\begin{aligned}\hat{\boldsymbol{t}}_k &= \mathbf{G}(q^{-1})\boldsymbol{t}_k\\ &= \mathrm{diag}(G_1(q^{-1}), G_2(q^{-1}), \ldots, G_l(q^{-1}))\boldsymbol{t}_k\end{aligned} \tag{33}$$

3.4.2. DiCCA model relations

DiCCA has a similar model structure as DiPLS. Assuming the number of latent variables is chosen as l in the DiCCA model and defining the following matrices,

$$\mathbf{T} = [\mathbf{t}^{(1)} \quad \mathbf{t}^{(2)} \quad \cdots \quad \mathbf{t}^{(l)}]$$

$$\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_l]$$

$$\mathbf{P} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_l]$$

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

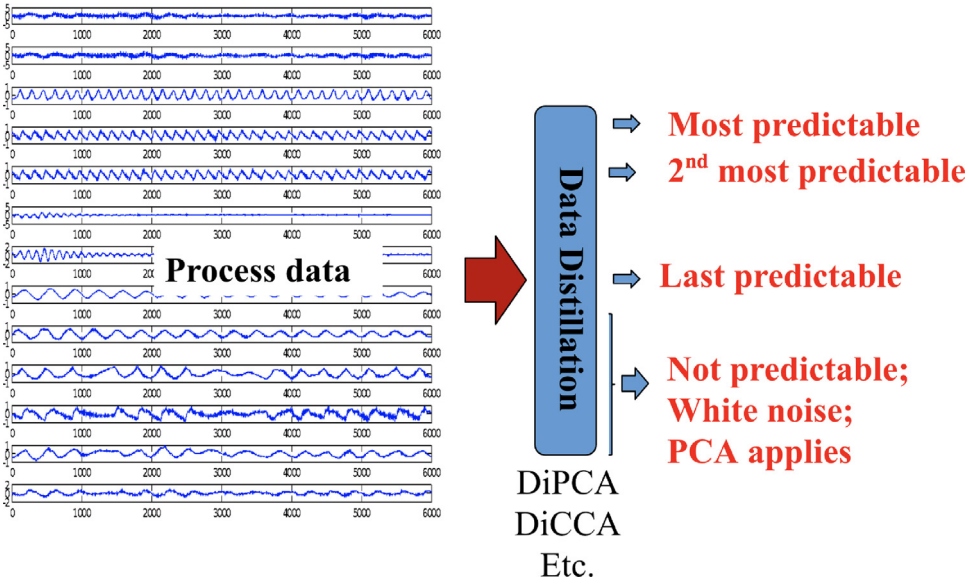


Fig. 3. Dynamic latent variable analytics is interpreted as a process of distilling dynamic latent components one after another, with the objective to maximize the covariance or correlation between the component and the prediction from its past.

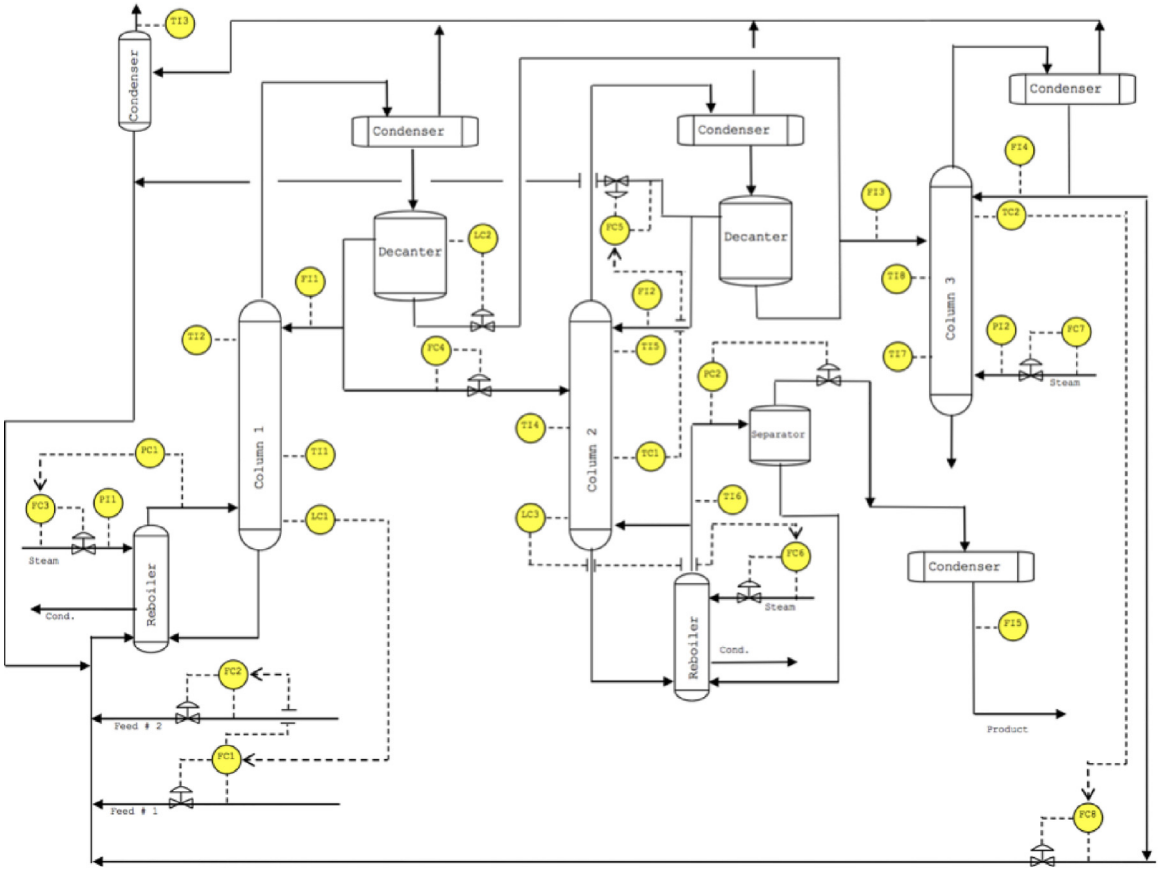


Fig. 4. Process schematic diagram from the Eastman Chemical Company.

By iterating (29), we have the following relations

$$\mathbf{X}^{(l+1)} = \mathbf{X} - \sum_{i=1}^{l} \mathbf{t}^{(i)}\mathbf{p}_i^T = \mathbf{X} - \mathbf{T}\mathbf{P}^T$$

or equivalently

$$\mathbf{X} = \mathbf{X}^{(l+1)} + \mathbf{T}\mathbf{P}^T \tag{34}$$

Based on the orthogonal DiCCA properties in Dong and Qin (2017),

$$\mathbf{T} = \mathbf{X}\mathbf{R} \tag{35}$$

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

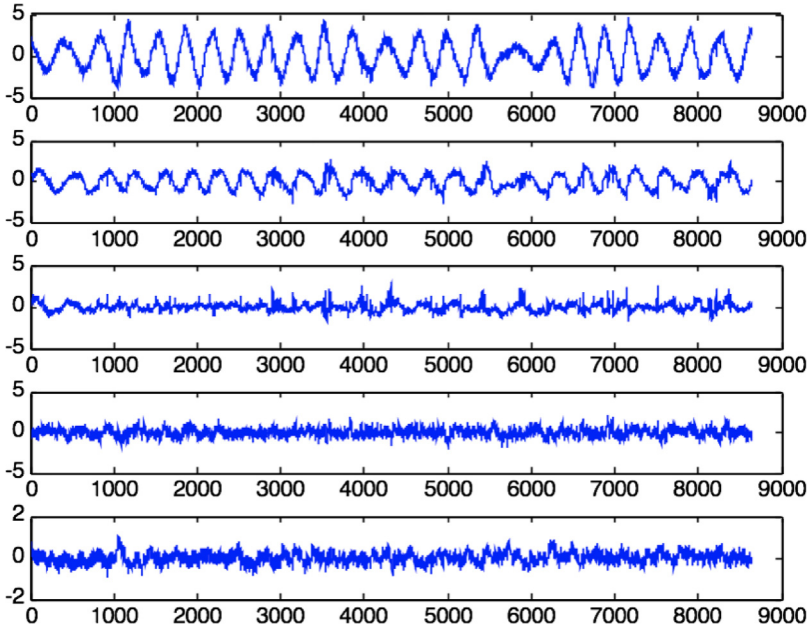


Fig. 5. Plots of five dynamic principal components using DiPCA.

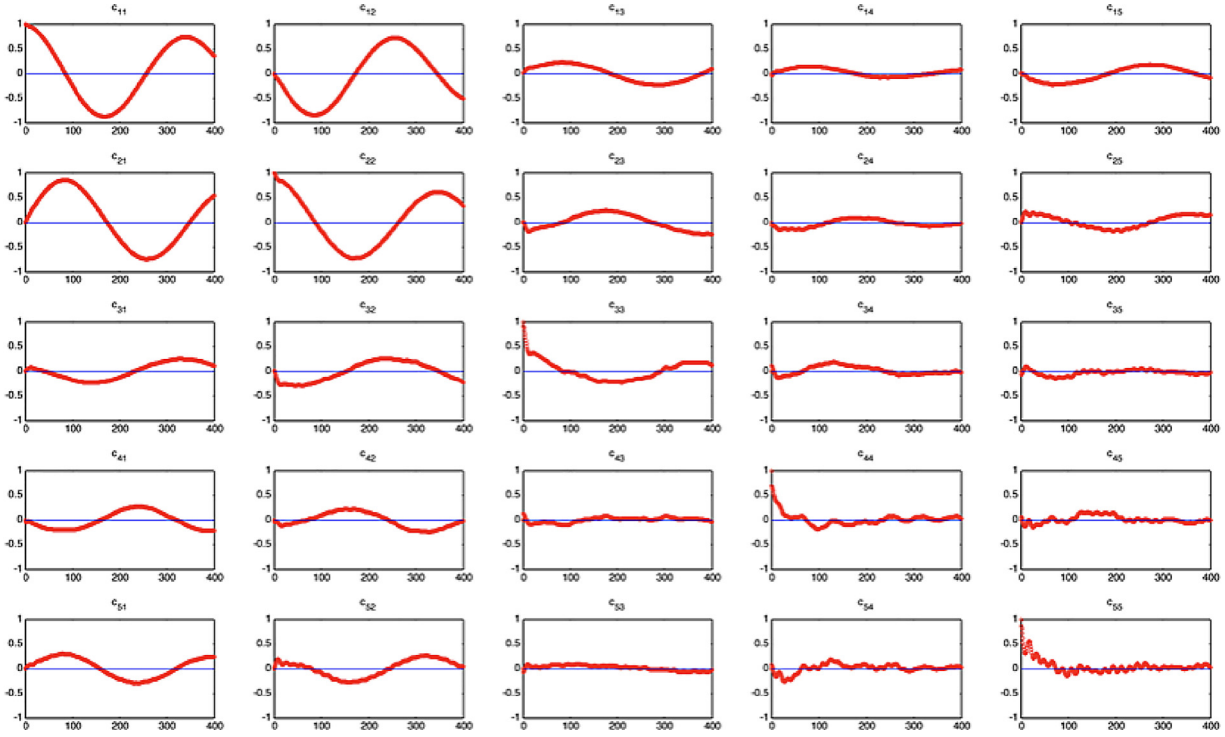


Fig. 6. Auto-correlation and cross-autocorrelation for five DiPCA PCs.

where

$$\mathbf{R} = \mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1} \tag{36}$$

Therefore, for a given vector $\boldsymbol{x}_k$, the score vector can be calculated as

$$\boldsymbol{t}_k = \mathbf{R}^T\boldsymbol{x}_k$$

and $\mathbf{x}_k$ can be decomposed as

$$\boldsymbol{x}_k = \mathbf{P}\boldsymbol{t}_k + \tilde{\boldsymbol{x}}_k \tag{37}$$

The decomposition in (37) gives the partition of the space formed by current data. Furthermore, to explore the relations between the past data and current data, relation (33) can be utilized to derive the one step prediction error,

$$\boldsymbol{e}_k = \boldsymbol{x}_k - \mathbf{P}\hat{\boldsymbol{t}}_k = \boldsymbol{x}_k - \mathbf{P}\mathbf{G}(q^{-1})\boldsymbol{t}_k \tag{38}$$

When the number of dynamic latent variables l is selected to extract all dynamics in the data, there will be little or no dynamics left in $\boldsymbol{e}_k$. Further PCA modeling of $\boldsymbol{e}_k$ will be appropriate if it is desired to extract the covariance structure in the prediction error.

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

In addition, since there is little or no dynamics left in $\mathbf{e}_k$ after the removal of the prediction error, DiCCA can be interpreted as a dynamic whitening filter which removes all the dynamics in the data. Mathematically, based on (33) and (38), the dynamic whitening filter can be written as

$$\boldsymbol{e}_k = (\mathbf{I} - \mathbf{PG}(q^{-1})\mathbf{R}^T)\boldsymbol{x}_k \tag{39}$$

The block diagram of DiCCA dynamic whitening filter is shown in Fig. 2.

3.5. Interpretation as dynamic data distillation

With the objective of maximizing covariance or correlation between the latent time series and its prediction from the past, DiPCA and DiCCA performs dynamic data distillation from all measured data such that the extracted dynamic components co-vary or correlate the most with their past. These most predictable dynamic components are referred to as principal time series that are latent in the original data. The prediction errors of the data after the first predicted component is extracted are then used to extract the second most predictable latent component, until all significant dynamic components are extracted. This procedure is analogous to a multi-stage binary distillation process, with each stage separating the most dynamic component from others, and the leftovers after all components are extracted are essentially un-correlated in time, resembling white noise. Fig. 3 illustrates how these dynamic latent variable analytics distill dynamic latent components one after another, with the objective to maximize the covariance or correlation between the component and its prediction from the past data. High dimensional time series data are considered as a mixture of a number of dynamic latent components, which are not measured directly, and static errors. DiPCA and DiCCA distill the multi-dimensional data into dynamic components in descending order of covariance or correlation, respectively.

4. Dynamic feature extraction for industrial process data

The case study is on a set of data from the Eastman Chemical Company, with Fig. 4 showing the process schematic diagram. The Eastman Chemical Company had identified a need to diagnose a common oscillation of about two hours (about 340 samples/cycle). Early work (Thornhill et al., 2003; Xia et al., 2005) has focused on the diagnosis of this oscillation. Using the Granger causality analysis, five process variables are identified to have strong oscillations (Yuan and Qin, 2014). These five variables with strong oscillations are selected in this work to illustrate how the dynamic data and features can be modeled using DiPCA and DiCCA.

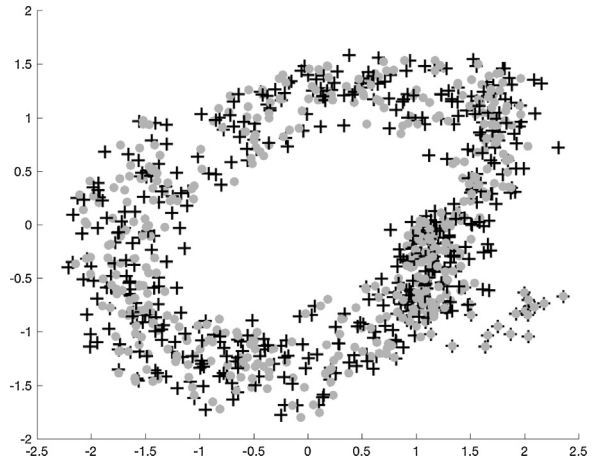


Fig. 7. The first two DiPCA PCs and the predictions from their past scores using DiPCA. The circular shape shows covarying oscillations at the same frequency.

Using DiPCA on the five process variables leads to five dynamic PCs as shown in Fig. 5. The auto-regression order of the dynamics is chosen as 21, which makes the prediction errors of the dynamic principal components essentially white. Fig. 6 depicts the auto-correlation and cross-autocorrelation for the five dynamic PCs. It is clear that the first two PCs are very oscillatory, while the third one is still somewhat oscillatory and co-varies with the first two PCs. To visualize how the DiPCA model predicts the PCs, the first two DiPCA PCs and the predictions from their past scores are shown in Fig. 7. The circular shape shows the co-varying oscillations at the same frequency.

Next, DiCCA is used to model the five process variables, which leads to five dynamic PCs as shown in Fig. 8. The order of the dynam-

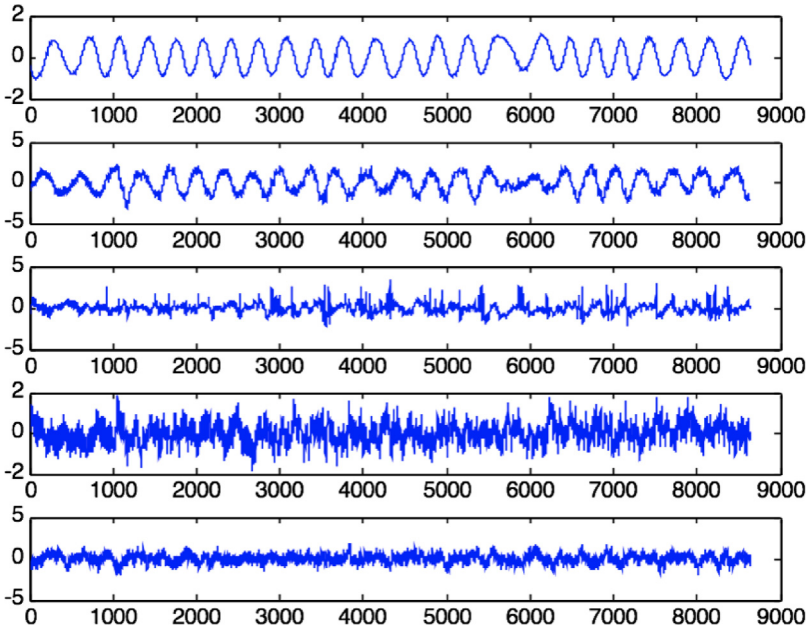


Fig. 8. Plots of five dynamic principal components using DiCCA.

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

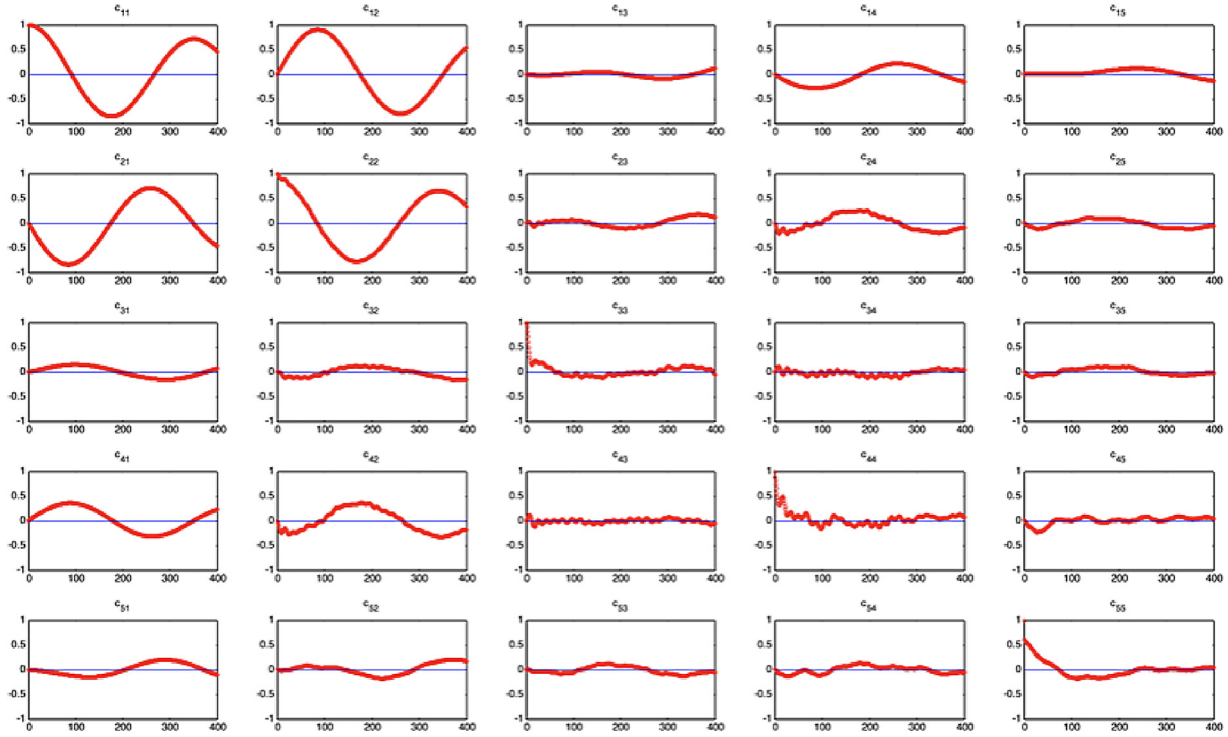


Fig. 9. Auto-correlation and cross-autocorrelation for five DiCCA PCs. The third PC is little correlated to the first two PCs.

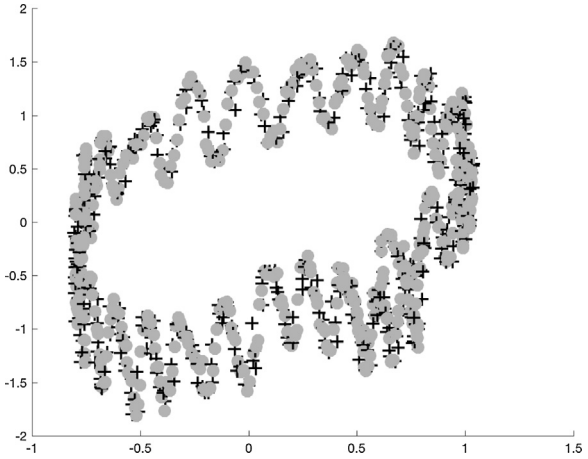


Fig. 10. The first two DiCCA PCs and the predictions from their past scores. The big circular shape shows the co-varying oscillations at the same frequency. In addition, there is a clear smaller oscillation with higher frequency due to the second PC.

ics is chosen as 22, which is chosen such that the errors predicted with the dynamic PCs are essentially white. Fig. 9 depicts the auto-correlation and cross-autocorrelation for the five DiCCA PCs. It is clear that the first two PCs are very oscillatory, while the third one is little correlated to the first two PCs.

To visualize how the DiCCA model predicts the PCs, the first two DiCCA PCs and the predictions from their past scores are shown in Fig. 10. While the big circular shape shows co-varying oscillations at the same frequency, there is a clear smaller oscillation with higher frequency that is best captured by the second PC. This feature is not observed at all using DiPCA analysis. The DiCCA scatterplot has clear but faster oscillations on top of the large circular shape, indicating that there is another high frequency oscillation component. This frequency is likely caused by another control loop with valve stiction. Dong and Qin (2018) apply causality analysis and contribution analysis to pinpoint to the causes of these oscillations. The fact that DiCCA detects a new feature makes it better than DiPCA in extracting dynamic features.

5. Conclusions

Process data analytics have been applied in chemical process operations for decades. Although latent variable methods have been successfully applied to static data analysis, the methods presented in this paper appear to be the first class of methods that maximizing the dynamic content of the projected latent scores. The principal dynamic latent variables are most predictable components in the whole data space, and therefore have the least prediction errors. In the industrial data case from Tennessee Eastman studied in this paper, the sustained dynamic oscillatory content is undesirable in process operations. Further diagnostic analysis reveals what they are and where they come from Dong and Qin (2018). By fixing the causes the excessive oscillations can be eliminated.

With the advancement of analytics in other sectors of industries and business operations, there appears to be much more research opportunities in the future. While physical and chemical sciences develop principles which enable mechanistic models to be established for process understanding, data analytics provide real and up-to-date information that reflects changes and uncertainty in the operation, and provide a reliable source of information to detect emerging situations.

Prediction, visualization, and interpretation of massive data with latent variables are powerful to deal with high dimensional and highly correlated data. The benefit of data analytics is to turn data into knowledge and support effective operations and decision-making, which help push beyond the traditional boundaries.

Acknowledgments

Funding for the work in this paper from the following sources is gratefully acknowledged: the Natural Science Foundation of China

under Grant 61490704, the Fundamental Research Program of the Shenzhen Committee on Science and Innovations (20160207, 20170155), and the Texas-Wisconsin-California Control Consortium. The authors are grateful to the Eastman Chemical process control team, in particular John Cox, for providing the industrial process data analyzed in this work.

Appendix A. Proof of Theorem 1

The least squares solution for b to the objective $\min \| \mathbf{Yq} - b\mathbf{Xw} \|^2$ is

$$b = (\mathbf{w}^T\mathbf{X}^T\mathbf{Xw})^{-1}\mathbf{w}^T\mathbf{X}^T\mathbf{Yq}$$

Applying the constraint $\| \mathbf{Xw} \|^2 = \mathbf{w}^T\mathbf{X}^T\mathbf{Xw} = 1$, the solution can be simplified as $b = \mathbf{w}^T\mathbf{X}^T\mathbf{Yq}$. Replacing b in the least squares objective and utilizing $\| \mathbf{Yq} \|^2 = 1$ and $\| \mathbf{Xw} \|^2 = 1$, we have

$$\begin{aligned} J &= (\mathbf{Yq} - b\mathbf{Xw})^T(\mathbf{Yq} - b\mathbf{Xw}) \\ &= \left\|\mathbf{Yq}\right\|^2 - b\mathbf{w}^T\mathbf{X}^T\mathbf{Yq} - b\mathbf{q}^T\mathbf{Y}^T\mathbf{Xw} + b^2\left\|\mathbf{Xw}\right\|^2 \\ &= 1 - (\mathbf{w}^T\mathbf{X}^T\mathbf{Yq})^2 - (\mathbf{w}^T\mathbf{X}^T\mathbf{Yq})^2 + (\mathbf{w}^T\mathbf{X}^T\mathbf{Yq})^2 \\ &= 1 - (\mathbf{w}^T\mathbf{X}^T\mathbf{Yq})^2 \end{aligned}$$

Therefore, minimizing $\| \mathbf{Yq} - b\mathbf{Xw} \|^2$ is equivalent to maximizing $\mathbf{w}^T\mathbf{X}^T\mathbf{Yq}$ under the constraints $\| \mathbf{Yq} \|^2 = 1$, $\| \mathbf{Xw} \|^2 = 1$, which is equivalent to maximizing

$$J = \frac{\mathbf{q}^T\mathbf{Y}^T\mathbf{Xw}}{\|\mathbf{Yq}\|\|\mathbf{Xw}\|}$$

Appendix B. DiPCA algorithm

1. Scale $\mathbf{X}$ to zero-mean and unit-variance. Initialize $\mathbf{w}$ to a random unit vector.
2. Extracting latent variables. Iterate the following relations until convergence.

$$\mathbf{t} = \mathbf{Xw} \quad \text{and form } \mathbf{t}_i \text{ from } \mathbf{t} \text{ similar to the formation of } \mathbf{X}_i \text{ for } i = 1, 2, \ldots, s+1$$

$$\boldsymbol{\beta} = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_s]^T\mathbf{t}_{s+1}$$

$$\mathbf{w} = \sum_{i=1}^{s} \beta_i(\mathbf{X}_{s+1}^T\mathbf{t}_i + \mathbf{X}_i^T\mathbf{t}_{s+1})$$

$$\mathbf{w} := \mathbf{w}/\|\mathbf{w}\|$$

$$\boldsymbol{\beta} := \boldsymbol{\beta}/\|\boldsymbol{\beta}\|$$

3. Deflation. Deflate $\mathbf{X}$ as

$$\mathbf{X} := \mathbf{X} - \mathbf{tp}^T; \ \mathbf{p} = \mathbf{X}^T\mathbf{t}/\mathbf{t}^T\mathbf{t}$$

4. Return to Step 2 to extract the next latent variable, until l latent variables are extracted.
5. Dynamic inner modeling. Build a VAR model for latent score. Then, $\mathbf{T}_{s+1}$ is predicted as

$$\hat{\mathbf{T}}_{s+1} = \bar{\mathbf{T}}_s\hat{\boldsymbol{\Theta}}$$

6. Static modeling of prediction errors. Perform traditional PCA on the prediction error matrix $\mathbf{E}_{s+1}$

$$\mathbf{E}_{s+1} = \mathbf{X}_{s+1} - \hat{\mathbf{T}}_{s+1}\mathbf{P}^T = \mathbf{T}_r\mathbf{P}_r^T + \mathbf{E}_r$$

Appendix C. DiPLS algorithm

1. Scale $\mathbf{X}$ and $\mathbf{Y}$ to zero-mean and unit-variance. Initialize $\boldsymbol{\beta}$ with $[1, 0, \ldots, 0]'$, and $\mathbf{u}_s$ as some column of $\mathbf{Y}_s$.
2. Outer modeling. Iterate the following relations until convergence achieved.

$$\mathbf{w} = \sum_{i=1}^{s} \beta_i\mathbf{X}_{s+1-i}^T\mathbf{u}_s; \ \mathbf{w} := \mathbf{w}/\|\mathbf{w}\|$$

$$\mathbf{t} = \mathbf{Xw} \quad \text{and} \quad \text{form } \mathbf{t}_i \text{ from } \mathbf{t} \text{ for } i = 1, \ldots, s$$

$$\mathbf{q} = \mathbf{Y}_s^T\sum_{i=1}^{s} \beta_i\mathbf{t}_{s+1-i}; \ \mathbf{q} := \mathbf{q}/\|\mathbf{q}\|$$

$$\mathbf{u}_s = \mathbf{Y}_s\mathbf{q}$$

$$\boldsymbol{\beta} = [\beta_1 \quad \beta_2 \quad \cdots \quad \beta_s] = [\mathbf{t}_s \quad \mathbf{t}_{s-1} \quad \cdots \quad \mathbf{t}_1]^T\mathbf{u}_s; \ \boldsymbol{\beta} := \boldsymbol{\beta}/\|\boldsymbol{\beta}\|$$

3. Inner modeling. Build a linear model between $\mathbf{t}_s, \mathbf{t}_{s-1}, \ldots, \mathbf{t}_0$ and $\mathbf{u}_s$ by least squares.

$$\mathbf{u}_s = \alpha_1\mathbf{t}_s + \alpha_2\mathbf{t}_{s-1} + \cdots + \alpha_s\mathbf{t}_1 + \mathbf{r}_s$$

and calculate the predicted $\hat{\mathbf{u}}_s$ as follows

$$\hat{\mathbf{u}}_s = \alpha_1\mathbf{t}_s + \alpha_2\mathbf{t}_{s-1} + \cdots + \alpha_s\mathbf{t}_1$$

4. Deflation. Deflate $\mathbf{X}$ and $\mathbf{Y}$ as

$$\mathbf{X} := \mathbf{X} - \mathbf{tp}^T; \ \mathbf{p} = \mathbf{X}^T\mathbf{t}/\mathbf{t}^T\mathbf{t}$$

$$\mathbf{Y}_s := \mathbf{Y}_s - \hat{\mathbf{u}}_s\mathbf{q}^T$$

5. Return to Step 2 until enough latent variables are extracted.

References

Åström, K.J., McAvoy, T.J., 1992. Intelligent control. J. Process Control 2 (3), 115–127.
Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., 2007. Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems., pp. 153–160.
Chiang, L.H., Russell, E.L., Braatz, R.D., 2000. Fault Detection and Diagnosis in Industrial Systems. Springer Science & Business Media.
Cinar, A., Palazoglu, A., Kayihan, F., 2007. Chemical Process Performance Evaluation. CRC Press.
Dong, Y., Qin, S.J., 2015. Dynamic-inner partial least squares for dynamic data modeling. IFAC-PapersOnLine 48 (8), 117–122.
Dong, Y., Qin, S.J., 2017. A novel dynamic PCA algorithm for dynamic data modeling and process monitoring. J. Process Control (in press).
Dong, Y., Qin, S.J., 2018. Dynamic-inner canonical correlation and causality analysis for high dimensional time series data. Submitted to the 10th IFAC Symposium on Advanced Control of Chemical Processes.
Freund, Y., Schapire, R.E., 1999. Large margin classification using the perceptron algorithm. Mach. Learn. 37 (3), 277–296, http://dx.doi.org/10.1023/A:1007662407062.
Ginsberg, J., Mohebbi, M.H., Patel, R.S., Brammer, L., Smolinski, M.S., Brilliant, L., 2009. Detecting influenza epidemics using search engine query data. Nature 457 (7232), 1012–1014.
Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.
Hinton, G.E., 1989. Connectionist learning procedures. Artif. Intell. 40 (1–3), 185–234.
Hopfield, J., 1990. Dynamics and neural network computation. Int. J. Quantum Chem. 38 (S24), 633–644.
Hotelling, H., 1936. Relations between two sets of variates. Biometrika 28 (3/4), 321–377.
Ku, W., Storer, R.H., Georgakis, C., 1995. Disturbance detection and isolation by dynamic principal component analysis. Chemom. Intell. Lab. Syst. 30 (1), 179–196.
Lee, J.H., Shin, J., Realff, M.J., 2017. Machine learning: Overview of the recent progresses and implications for the process systems engineering field.

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

Presented at the FOCAPO/Chemical Process Control Conference, Jan. 8–14, 2017, Tucson, AZ.
MacGregor, J.F., Kourti, T., 1995. Statistical process control of multivariate processes. Control Eng. Pract. 3 (3), 403–414.
MacGregor, J.F., Jaeckle, C., Kiparissides, C., Koutoudi, M., 1994. Process monitoring and diagnosis by multiblock PLS methods. AIChE J. 40 (5), 826–838.
Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H., 2011. Big Data: The Next Frontier for Innovation, Competition, and Productivity, MGI Report.
Morris, A., Montague, G., Tham, M., Aynsley, M., Di Massimo, C., Lant, P., 1991. Towards improved process supervision-algorithms and knowledge based systems. Chemical Process Control CPC-IV, CACHE and AIChE, New York.
National Research Council, 2013. Frontiers in Massive Data Analysis. The National Academies Press, Washington, DC.
Qin, S.J., 2003. Statistical process monitoring: basics and beyond. J. Chemom. 17 (8–9), 480–502.
Qin, S.J., Badgwell, T.A., 2003. A survey of industrial model predictive control technology. Control Eng. Pract. 11 (7), 733–764.
Qin, S., McAvoy, T., 1992. A data-based process modeling approach and its applications. In: Dynamics and Control of Chemical Reactors, Distillation Columns and Batch Processes (DYCORD+'92): Selected Papers from the 3rd IFAC Symposium, Maryland, USA, 26–29 April 1992. Elsevier, pp. 93.
Qin, S.J., Zheng, Y., 2013. Quality-relevant and process-relevant fault monitoring with concurrent projection to latent structures. AIChE J. 59 (2), 496–504.
Qin, S.J., 2012. Survey on data-driven industrial process monitoring and diagnosis. Annu. Rev. Control 36 (2), 220–234.
Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., 2016. Mastering the game of Go with deep neural networks and tree search. Nature 529 (7587), 484–489.
Stephanopoulos, G., 1991. Towards the intelligent controller: formal integration of pattern recognition with control theory. Preprints of the 4th International Conference on Chemical Process Control, 17–22.
Tham, M.T., Montague, G.A., Morris, A.J., Lant, P.A., 1991. Soft-sensors for process estimation and inferential control. J. Process Control 1 (1), 3–14.
Thornhill, N.F., Cox, J.W., Paulonis, M.A., 2003. Diagnosis of plant-wide oscillation through data-driven analysis and process understanding. Control Eng. Pract. 11 (12), 1481–1490.
Trygg, J., Wold, S., 2002. Orthogonal projections to latent structures (O-PLS). J. Chemom. 16 (3), 119–128.
Vapnik, V., 2013. The Nature of Statistical Learning Theory. Springer Science & Business Media.
Venkatasubramanian, V., 1991. Recall and generalization performances of neural networks for process fault diagnosis. Proc. CPC IV, 647–664.
Wise, B.M., Gallagher, N.B., 1996. The process chemometrics approach to process monitoring and fault detection. J. Process Control 6 (6), 329–348.
Xia, C., Howell, J., Thornhill, N.F., 2005. Detecting and isolating multiple plant-wide oscillations via spectral independent component analysis. Automatica 41 (12), 2067–2075.
Yuan, T., Qin, S.J., 2014. Root cause diagnosis of plant-wide oscillations using granger causality. J. Process Control 24 (2), 450–459.
Zhu, Q., Liu, Q., Qin, S.J., 2016. Concurrent canonical correlation analysis modeling for quality-relevant monitoring. IFAC-PapersOnLine 49 (7), 1044–1049.
Zhu, Q., Liu, Q., Qin, S.J., 2017. Concurrent quality and process monitoring with canonical correlation analysis. J. Process Control (in press).
Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. J. R. Stat. Soc.: Ser. B: Stat. Methodol. 67 (2), 301–320.

Please cite this article in press as: Dong, Y., Qin, S.J., Dynamic latent variable analytics for process operations and control. Computers and Chemical Engineering (2017), https://doi.org/10.1016/j.compchemeng.2017.10.029

Journal of Process Control 67 (2018) 1–11

Contents lists available at ScienceDirect

Journal of Process Control

journal homepage: www.elsevier.com/locate/jprocont

A novel dynamic PCA algorithm for dynamic data modeling and process monitoring

Yining Dong[a,c], S. Joe Qin[a,b,c,*]

[a] Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA
[b] Mork Family Department of Chemical Engineering and Material Science, University of Southern California, Los Angeles, CA 90089, USA
[c] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, 2001 Longxiang Blvd., Longgang, Shenzhen, Guangdong, China

ARTICLE INFO

Article history:
Received 18 October 2016
Received in revised form 1 April 2017
Accepted 8 May 2017
Available online 12 June 2017

Keywords:
Dynamic PCA
Dynamic data modeling
Process monitoring
Fault detection

ABSTRACT

Principal component analysis (PCA) has been widely applied for data modeling and process monitoring. However, it is not appropriate to directly apply PCA to data from a dynamic process, since PCA focuses on variance maximization only and pays no attention to whether the components contain dynamics or not. In this paper, a novel dynamic PCA (DiPCA) algorithm is proposed to extract explicitly a set of dynamic latent variables with which to capture the most dynamic variations in the data. After the dynamic variations are extracted, the residuals are essentially uncorrelated in time and static PCA can be applied. The new models generate a subspace of *principal time series* that are most predictable from their past data. Geometric properties are explored to give insight into the new dynamic model structure. For the purpose of process monitoring, fault detection indices based on DiPCA are developed based on the proposed model. Case studies on simulation data, data from an industrial boiler process, and the Tennessee Eastman process are presented to illustrate the effectiveness of the proposed dynamic models and fault detection methods.

1. Introduction

Industrial process data are becoming massive and increasingly valuable assets for decision making in process operations, process control and monitoring. Since process measurements are often highly correlated, latent variable methods, such as principal component analysis (PCA) and partial least squares (PLS), are effective analytic tools for data modeling and process monitoring [1–5]. In PCA, the objective is to extract latent variables from the data such that the variance of the extracted latent variables is maximized. By applying PCA, the measurement space can be decomposed into a principal subspace with maximized variability and a residual subspace. Fault detection statistics are developed for each subspace for process monitoring.

One major shortcoming for PCA is the lack of focus on time dependence, i.e., the structure of autocorrelation in the data is not exploited. However, measurements from industrial processes are often both cross-correlated and autocorrelated. Several problems can arise when applying static PCA to dynamic data directly. Since static PCA is unable to extract dynamic relationships from the data, autocorrelation and cross-correlation are mixed together, which makes it difficult for traditional PCA to reveal what type of relations among the measured variables. Furthermore, autocorrelations invalidate the statistical properties for fault detection methods developed for traditional PCA. Directly applying traditional fault detection methods may lead to misleading results.

Several methods have been developed to deal with measurements from dynamic processes. Negiz et al. [6] proposed a time series based approach, where a time series model is identified first. Then, the time series model is used to make predictions, and the prediction errors are used for process monitoring. However, this approach considers univariate time series models for simplicity. Even though it can be extended to multivariate time series, it does not retain the dimension reduction feature that is available in PCA to deal with cross-correlations.

A straightforward dynamic PCA method was proposed by Ku et al., which augments the data matrix with a number of lagged measurements [7]. Static PCA is performed on the augmented data matrix to extract latent variables. However, this method has several disadvantages. First, both the dimension of the loading vectors and the number of parameters increase dramatically as the number of lags increases. Second, no explicit representation can be derived between the latent variable and the original measurement vari-

* Corresponding author at: Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA.
E-mail address: sqin@usc.edu (S.J. Qin).

https://doi.org/10.1016/j.jprocont.2017.05.002
0959-1524/

ables. Third, the extracted latent variables have no attention on the dynamic content of the data; only variance is focused on, which makes it difficult to interpret the model and explore the underlying correlation structure. Alternative dynamic PCA algorithms were proposed to deal with various cases, such as nonlinearity and batch processes [8–10]. However, augmented data matrices are used in all of these algorithms. Therefore, they share the same limitations as Ku et al. [7].

Recently, Li et al. proposed a dynamic latent variable (DLV) modeling algorithm [11]. It first uses an auto-regressive PCA algorithm to extract the latent variables so that the maximum auto-covariance is achieved. Then, a vector autoregressive (VAR) model, which is referred as an *inner model*, is built for the latent variables to represent the dynamic relationships. This algorithm provides an explicit representation of the inner dynamic relationships, and a compact model to represent the data structure. However, when extracting the latent variables, only the auto-covariance at one time lag is maximized in this method; the auto-covariance at other time lags are ignored. This makes the extraction of latent variables inconsistent with the dynamic modeling of latent variables.

A structured dynamic PCA algorithm was proposed by Li et al. as an improvement, where the objective function is to maximize the variance of a weighted sum of lagged latent variables [12]. Then a VAR model is built to capture the dynamic relationships of the latent variables. However, the VAR model of the inner dynamic relationships is not consistent with the maximum variance in the objective function. Collinear static relationships can dominate a latent variable that satisfies the maximum-variance, which fails to extract dynamic relationships in the latent variables.

Inspired by dynamic-inner partial least squares (DiPLS) [13] developed recently, a dynamic-inner principal component analysis (DiPCA) is proposed in this paper. While DiPLS is suitable for dynamic data modeling between two sets of variables, DiPCA extracts one or more latent variables that are linear combinations of one set of variables and have maximized auto-covariance. In the complement, the residuals after extracting the most predictable latent variables from the data will, in the limiting case, tend to be white noise. In this sense, the DiPCA algorithm extracts a set of principal time series from a multi-dimensional time series. The residual with little or no auto-covariance can then be treated as static data with traditional PCA methods for additional monitoring and analysis.

Although there are many versions of dynamic PCA algorithms in the literature, and some of them are used for process monitoring, most of them follow the approach by extending the data with time-lagged variables. These approaches immediately increase the dimensions of the data matrix, making them hardly a dimension reduction approach. Further, the static PCA objective in these methods does not even guarantee that the extracted principal components are dynamic. We listed other prior work in Table 1 for comparison, which were developed in our group as precursors to this current work. The interpretability of data analytic methods is always a desirable feature to possess. In this regard none of the existing methods can match the current work.

The remainder of this paper is organized as follows. Section 2 reviews the traditional PCA algorithm and gives the objective function of the DiPCA algorithm. Section 3 presents the proposed DiPCA algorithm. Geometric properties and relations of DiPCA are discussed in Section 4. Section 5 derives fault detection indices and proposes process monitoring schemes based on DiPCA model. Case studies on simulation data, boiler process data, and Tennessee Eastman Process data are presented to show the effectiveness of the proposed algorithm in Section 6. The final section gives conclusions.

2. PCA and DiPCA objective functions

Let $\mathbf{x} \in \mathbb{R}^m$ denote a sample vector of m variables. Assuming that there are n samples for each variable, a data matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, with each row representing a sample, can be used to build a PCA model. The objective of PCA is to extract a direction $\mathbf{p}$ of the largest variance in the m dimensional measurement space, which can be expressed as

$$\begin{aligned} \max_{\mathbf{p}} \quad & \mathbf{p}^T\mathbf{X}^T\mathbf{X}\mathbf{p} \\ \text{s.t.} \quad & \|\mathbf{p}\| = 1 \end{aligned} \tag{1}$$

The solution to this optimization problem is obtained as follows using a Lagrange multiplier,

$$\mathbf{X}^T\mathbf{X}\mathbf{p} = \lambda\mathbf{p} \tag{2}$$

where $\lambda = \mathbf{p}^T\mathbf{X}^T\mathbf{X}\mathbf{p}$ matches the objective in (1). This implies that $\mathbf{p}$ is the eigenvector of $\mathbf{X}^T\mathbf{X}$ corresponding to the largest eigenvalue. After the loading vector $\mathbf{p}$ is obtained, the latent score vector $\mathbf{t}$ can be calculated as $\mathbf{t} = \mathbf{X}\mathbf{p}$. To extract the next principal component, the same optimization problem can be solved on the deflated matrix:

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T \tag{3}$$

It is clear from the objective of PCA that only static cross-correlations among the variables are extracted by PCA. If the data are collected from a dynamic process, traditional PCA will leave the dynamics unmodeled, making the principal components and even residuals having unmodeled dynamics. Having dynamic content in a component means that its future can be predictable from its past to a certain extent. The uncertainty will be on the prediction error only, not the entire component. Therefore, ignoring dynamic information in the analysis makes the model not representative of the data when they are used for further analysis and process monitoring. The consequence is that, while the false alarm rate can be controlled with a larger-than-necessary control limit, it leads to an

Table 1
Comparisons of DiPCA with other methods.

Method	Focusing on dynamics	Explicit dynamic components	Interpretability	Dimension reduction
DiPCA (this work)	Yes	Yes	Easy	Yes, up to the dimension of the variable space
PCA with augmented lagged data [7]	No	No	Hardly any. There is no guarantee that the extracted principal components are even dynamic	No. The number of principal components can be greater than the dimension of the variable space
DLV [11]	No	Yes	Some, but static correlation can dominate a dynamic principal component	Yes, up to the dimension of the variable space
Structured DLV [12]	No	Yes	Some, but static correlation can dominate a dynamic principal component	Yes, up to the dimension of the variable space

overly large missed alarm rate. This restricts the applicability of PCA and makes it unsuitable for dynamic data modeling.

The objective of this paper is to develop a dynamic PCA objective that is consistent with its inner dynamic VAR relation, which is most predictable by its past. While this seems to be a natural and useful component to derive from the data, no prior work has been found in the literature. This approach is referred to as dynamic inner PCA in this paper. In general, the dynamics in latent variables that predict the current from the past can be expressed as

$$t_k = \beta_1 t_{k-1} + \cdots + \beta_s t_{k-s} + r_k \tag{4}$$

with the latent variable as a linear combination of the original variables

$$t_k = \mathbf{x}_k^T \mathbf{w} \tag{5}$$

where $\left\|\boldsymbol{\beta}\right\| = 1$, $\|\mathbf{w}\| = 1$, $\mathbf{x}_k$ is the sample vector at time k, and $\mathbf{w}$ is the weight vector. If the number of time lags, s, is chosen long enough such that the residual r_k is essentially white noise for each latent variable, the prediction from the dynamic inner model is

$$\begin{aligned}\hat{t}_k &= \mathbf{x}_{k-1}^T \mathbf{w}\beta_1 + \cdots + \mathbf{x}_{k-s}^T \mathbf{w}\beta_s \\ &= [\mathbf{x}_{k-1}^T \cdots \mathbf{x}_{k-s}^T](\boldsymbol{\beta} \otimes \mathbf{w})\end{aligned} \tag{6}$$

where $\boldsymbol{\beta} = [\beta_1\ \beta_2 \cdots \beta_s]^T$ and $\boldsymbol{\beta} \otimes \mathbf{w}$ is the Kronecker product. Based on the above dynamic inner model, the objective function for extracting the latent variables should maximize the covariance between t_k and $\hat{t}_k$. Let $\mathbf{x}_k$ be the sample vector at time k, $k = 1, 2, \ldots, N+s$. Therefore, the objective is to maximize

$$\frac{1}{N}\sum_{k=s+1}^{s+N} \mathbf{w}^T \mathbf{x}_k [\mathbf{x}_{k-1}^T \cdots \mathbf{x}_{k-s}^T](\boldsymbol{\beta} \otimes \mathbf{w}) \tag{7}$$

This objective leads to the dynamic inner PCA algorithm to be derived in the next section.

3. Dynamic inner PCA Algorithm

3.1. Extracting One Dynamic Component

Denote the data matrix as

$$\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_{N+s}]^T$$

and form the following data matrices from $\mathbf{X}$,

$$\begin{aligned}\mathbf{X}_i &= [\mathbf{x}_i \quad \mathbf{x}_{i+1} \quad \cdots \quad \mathbf{x}_{N+i-1}]^T \quad \text{for} \quad i = 1, 2, \cdots, s+1 \\ \mathbf{Z}_s &= [\mathbf{X}_1 \quad \mathbf{X}_2 \quad \cdots \quad \mathbf{X}_s]\end{aligned}$$

The objective in (7) can be rewritten as

$$\begin{aligned}\max_{\mathbf{w},\boldsymbol{\beta}} \quad & \mathbf{w}^T \mathbf{X}_{s+1}^T \mathbf{Z}_s (\boldsymbol{\beta} \otimes \mathbf{w}) \\ \text{s.t.} \quad & \|\mathbf{w}\| = 1,\ \left\|\boldsymbol{\beta}\right\| = 1\end{aligned} \tag{8}$$

where s is the dynamic order of the model. The dimension of $\mathbf{w}$ is the same as the number of variables, which does not increase with the dynamic order of the model. After the weighting vector $\mathbf{w}$ is extracted, the latent score t_k is calculated as $t_k = \mathbf{x}_k^T \mathbf{w}$. It is clear that the most co-varying dynamic relationship is extracted between t_k and $t_{k-1}, \ldots, t_{k-s}$ by the objective function. Therefore, an explicit dynamic model is built between the latent variables. Compared to other dynamic PCA algorithms, the objective function of DiPCA leads to the extraction of only dynamic latent relations. The residuals after all dynamic components are extracted contain little dynamic information and, therefore, can be analyzed using static PCA further if so desired. Lagrange multipliers are used to solve the optimization in (8). Define

$$J = \mathbf{w}^T \mathbf{X}_{s+1}^T \mathbf{Z}_s (\boldsymbol{\beta} \otimes \mathbf{w}) + \frac{1}{2}\lambda_\beta (1 - \boldsymbol{\beta}^T \boldsymbol{\beta}) + \lambda_w (1 - \mathbf{w}^T \mathbf{w}) \tag{9}$$

where

$$(\boldsymbol{\beta} \otimes \mathbf{w}) = (\boldsymbol{\beta} \otimes \mathbf{I})\mathbf{w} = (\mathbf{I} \otimes \mathbf{w})\boldsymbol{\beta}$$

Taking derivatives with respective to $\mathbf{w}$ and $\boldsymbol{\beta}$ and set them to zero, we have

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{w}} &= \mathbf{X}_{s+1}^T \mathbf{Z}_s (\boldsymbol{\beta} \otimes \mathbf{I})\mathbf{w} + (\boldsymbol{\beta} \otimes \mathbf{I})^T \mathbf{Z}_s^T \mathbf{X}_{s+1}\mathbf{w} \\ &\quad -2\lambda_w \mathbf{w} = 0\end{aligned} \tag{10}$$

$$\frac{\partial J}{\partial \boldsymbol{\beta}} = (\mathbf{I} \otimes \mathbf{w})^T \mathbf{Z}_s^T \mathbf{X}_{s+1}\mathbf{w} - \lambda_\beta \boldsymbol{\beta} = 0 \tag{11}$$

Define $\mathbf{G}_\beta = \mathbf{X}_{s+1}^T \mathbf{Z}_s (\boldsymbol{\beta} \otimes \mathbf{I})$, the above equations can be simplified as

$$\frac{\partial J}{\partial \mathbf{w}} = \mathbf{G}_\beta \mathbf{w} + \mathbf{G}_\beta^T \mathbf{w} - 2\lambda_w \mathbf{w} = 0 \tag{12}$$

$$\frac{\partial J}{\partial \boldsymbol{\beta}} = (\mathbf{I} \otimes \mathbf{w})^T \mathbf{Z}_s^T \mathbf{X}_{s+1}\mathbf{w} - \lambda_\beta \boldsymbol{\beta} = 0 \tag{13}$$

Pre-multiplying $\mathbf{w}^T$ to (12) and $\boldsymbol{\beta}^T$ to (13) and using the fact that these vectors are unit norm, we have

$$\begin{aligned}2\lambda_w &= \mathbf{w}^T (\mathbf{G}_\beta + \mathbf{G}_\beta^T)\mathbf{w} = 2J \\ \lambda_\beta &= \boldsymbol{\beta}^T (\mathbf{I} \otimes \mathbf{w})^T \mathbf{Z}_s^T \mathbf{X}_{s+1}\mathbf{w} = J\end{aligned} \tag{14}$$

The above results imply that λ_w and λ_β are both equal to the maximum value of J, and $\mathbf{w}$ is the eigenvector of $\mathbf{G}_\beta + \mathbf{G}_\beta^T$ corresponding to the largest eigenvalue. However, since $\boldsymbol{\beta}$ and $\mathbf{w}$ are coupled together, there is no analytical solution to the optimization problem (8). Defining the latent scores $\mathbf{t}$ as follows,

$$\mathbf{t} = \mathbf{X}\mathbf{w}$$

and denoting

$$\mathbf{t}_i = \mathbf{X}_i \mathbf{w}, \quad \text{for } i = 1, 2, \ldots, s+1$$

as scores for the sub-matrices, (13) can be rewritten as follows,

$$\lambda_\beta \boldsymbol{\beta} = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_s]^T \mathbf{t}_{s+1} \tag{15}$$

which indicated that $\boldsymbol{\beta}$ depends on $\mathbf{w}$ implicitly through its dependence on $\mathbf{t}_i$, for $i = 1, 2, \ldots, s+1$, entirely. Furthermore, (12) can be re-organized as follows,

$$2\lambda_w \mathbf{w} = \sum_{i=1}^{s} \beta_i (\mathbf{X}_{s+1}^T \mathbf{t}_i + \mathbf{X}_i^T \mathbf{t}_{s+1}) \tag{16}$$

Therefore, the following iterative method is proposed to solve the problem.

(1) Initialize $\mathbf{w}$ with unit vector.
(2) Calculate $\mathbf{w}$, $\boldsymbol{\beta}$ by iterating the following relations until convergence.

$$\begin{aligned}\mathbf{t} &= \quad \mathbf{X}\mathbf{w} \text{ and form } \mathbf{t}_i \text{ from } \mathbf{t} \text{ for } i = 1, 2, \ldots, s+1 \\ \boldsymbol{\beta} &= \quad [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_s]^T \mathbf{t}_{s+1} \\ \mathbf{w} &= \quad \sum_{i=1}^{s} \beta_i (\mathbf{X}_{s+1}^T \mathbf{t}_i + \mathbf{X}_i^T \mathbf{t}_{s+1}) \\ \mathbf{w} &:= \quad \mathbf{w}/\|\mathbf{w}\| \\ \boldsymbol{\beta} &:= \quad \boldsymbol{\beta}/\left\|\boldsymbol{\beta}\right\|\end{aligned}$$

• Calculate $J = \sum_{i=1}^{s} \beta_i \mathbf{t}_i^T \mathbf{t}_{s+1}$.

X
s+1
Z
s

3.2. Deflation

After the weight vector $\mathbf{w}$ and latent scores $\mathbf{t}$ are obtained from the above algorithm, $\mathbf{X}$ is deflated as

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T \tag{17}$$

where the loading vector $\mathbf{p}$ is defined as

$$\mathbf{p} = \mathbf{X}^T\mathbf{t}/\mathbf{t}^T\mathbf{t} \tag{18}$$

Form $\mathbf{X}_i$ from $\mathbf{X}$ for $i = 1, 2, \ldots, s+1$ and repeat the same algorithm to extract the next latent variable.

Remark 1. As in DiPLS [13], an alternative approach could be conceived, where the predicted $\hat{\mathbf{t}}_{s+1}$ is used to deflate $\mathbf{X}_{s+1}$ as

$$\begin{aligned} \mathbf{X}_{s+1} &:= \mathbf{X}_{s+1} - \hat{\mathbf{t}}_{s+1}\mathbf{q}^T \\ \hat{\mathbf{t}}_{s+1} &= \alpha_1\mathbf{t}_1 + \alpha_2\mathbf{t}_2 + \cdots + \alpha_s\mathbf{t}_s \\ \mathbf{q} &= \mathbf{X}_{s+1}^T\hat{\mathbf{t}}_{s+1}/\hat{\mathbf{t}}_{s+1}^T\hat{\mathbf{t}}_{s+1} \end{aligned} \tag{19}$$

While the deflation of other matrices in $\mathbf{Z}_s$ remains the same as (17). However, this approach cannot be applied to DiPCA because we need to deflate both $\mathbf{Z}_s$ and $\mathbf{X}_{s+1}$ the same way to ensure that the next dynamic inner model is still an auto-regressive model. Therefore, it is imperative to deflate the way as shown in (17) to ensure that the subsequent latent variables are derived from consistently deflated residual matrices. The approach depicted in (19) no longer extracts the auto-regressive time series beyond the first latent time series.

3.3. DiPCA Model with l Components

The DiPCA main algorithm extracts the latent time series as the dynamic latent scores one by on in descending order of predicted covariation. These scores are orthogonal as will be shown later in this paper, which are very convenient for interpretation, but there can be some degree of correlation among them at different time lags. There is no reason to believe that these interactions are strong given that they are orthogonal without time lags, but one has the option to predict all latent scores simultaneously to account for these minor interactions. Let $\check{\mathbf{t}}_j, j = 1, 2, \ldots, l$ denote the *jth* latent component score vector and let $\mathbf{T} = [\check{\mathbf{t}}_1\ \check{\mathbf{t}}_2 \cdots \check{\mathbf{t}}_l]^T$. Form $\mathbf{T}_i$ from $\mathbf{T}$ for $i = 1, 2, \ldots, s+1$ in the same way as forming $\mathbf{X}_i$ from $\mathbf{X}$. A VAR model can be built to represent the dynamic relations between $\mathbf{T}_{s+1}$ and $\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_s$ as follows.

$$\begin{aligned} \mathbf{T}_{s+1} &= \mathbf{T}_1\boldsymbol{\Theta}_s + \mathbf{T}_2\boldsymbol{\Theta}_{s-1} + \cdots + \mathbf{T}_s\boldsymbol{\Theta}_1 + \mathbf{V} \\ &= \bar{\mathbf{T}}_s\boldsymbol{\Theta} + \mathbf{V} \end{aligned} \tag{20}$$

where $\bar{\mathbf{T}}_s = [\mathbf{T}_1\ \mathbf{T}_2 \cdots \mathbf{T}_s]$ and $\boldsymbol{\Theta} = [\boldsymbol{\Theta}_s\boldsymbol{\Theta}_{s-1} \cdots \boldsymbol{\Theta}_1]$. The least squares estimate for $\boldsymbol{\Theta}$ is

$$\hat{\boldsymbol{\Theta}} = (\bar{\mathbf{T}}_s^T\bar{\mathbf{T}}_s)^{-1}\bar{\mathbf{T}}_s^T\mathbf{T}_{s+1} \tag{21}$$

Once $\hat{\boldsymbol{\Theta}}$ is obtained, $\mathbf{T}_{s+1}$ can be predicted as

$$\hat{\mathbf{T}}_{s+1} = \bar{\mathbf{T}}_s\hat{\boldsymbol{\Theta}} \tag{22}$$

which can further be used to calculate the prediction of $\mathbf{X}_{s+1}$ as

$$\hat{\mathbf{X}}_{s+1} = \hat{\mathbf{T}}_{s+1}\mathbf{P}^T \tag{23}$$

where $\mathbf{P} = [\mathbf{p}_1\mathbf{p}_2 \cdots \mathbf{p}_l]$ is the loading matrix with each $\mathbf{p}_i$ defined in (18).

With the prediction from past data, the prediction errors (PE) from DiPCA can be calculated as follows.

$$\mathbf{E}_{s+1} = \mathbf{X}_{s+1} - \hat{\mathbf{T}}_{s+1}\mathbf{P}^T \tag{24}$$

After the principal dynamic components are extracted, traditional PCA can be performed on the prediction error as follows,

$$\mathbf{E}_{s+1} = \mathbf{T}_r\mathbf{P}_r^T + \mathbf{E}_r \tag{25}$$

Therefore, $\mathbf{X}_{s+1}$ is decomposed as

$$\mathbf{X}_{s+1} = \hat{\mathbf{T}}_{s+1}\mathbf{P}^T + \mathbf{T}_r\mathbf{P}_r^T + \mathbf{E}_r \tag{26}$$

where the first term on the right hand side of (26) is a prediction using past data, while the other terms are projections of data involving the current data. The procedure of DiPCA modeling can be summarized in Table 2 as the DiPCA Algorithm.

Remark 2. It is possible that the iteration converges to a local maximum. This will lead to a smaller value of J. To avoid local maxima, initialize $\mathbf{w}$ randomly and perform multiple trials. The optimal $\mathbf{w}$ and $\boldsymbol{\beta}$ correspond to the largest value of J.

Remark 3. The inner model dynamics is parameterized with an autoregressive model. It is known in system identification that AR models can approximate more general dynamics such as autoregressive moving average (ARMA) dynamics. If it is desirable to parametrize the inner dynamics as an ARMA model, a subsequent model-reduction can be implemented to obtain an ARMA model or state space model [14]. If there exists integrating dynamics, it is likely to be extracted among the first few dynamic components. One has the option to model the integrating dynamics a priori or do it as part of DiPCA.

3.4. Determination of model parameters

In DiPCA modeling, three parameters need to be determined: dynamic order s, the number of dynamic latent variables l, and the number of static latent variables. First, assuming s is determined, then l can be chosen such that 95 percent of auto-covariance are captured by the first l dynamic latent variables. Therefore, l is viewed as a function of s, which can be written as $l = l(s)$. To determine the optimal s, a DiPCA model is built based on the training the data first. Then applying the model to the validation dataset allows us to obtain the prediction error matrix $\mathbf{E}_{s+1}^V$ of the validation data matrix. According to the previous analysis, little dynamic relationships are left in $\mathbf{E}_{s+1}^V$. Therefore, the sample crosscorrelation of any two variables in $\mathbf{E}_{s+1}^V$ should be close to 0, except when $lag = 0$. The calculation of the corresponding confidence bounds can be found in [15]. When all the pairs of variables are considered, the total violations of the confidence bounds can be obtained for any $(s, l(s))$. The parameter $(s, l(s))$ corresponding to the minimum violations is determined to be optimal. To determine the number of static components, cumulative percentage of variance (CPV) is applied [16].

4. DiPCA geometric properties and relations

4.1. DiPCA geometric properties

The DiPCA algorithm has a different structure from static PCA and other dynamic PCA algorithms. Therefore, it is important to understand the geometric properties of the DiPCA projections and

Table 2
DiPCA Algorithm.

1.	Scale $\mathbf{X}$ to zero-mean and unit-variance. Initialize $\mathbf{w}$ to a random unit vector.
2.	Extracting latent variables. Iterate the following relations until convergence. $\mathbf{t} = \mathbf{X}\mathbf{w}$ and form $\mathbf{t}_i$ from $\mathbf{t}$ similar to the formation of $\mathbf{X}_i$ for $i = 1, 2, \ldots, s+1$ $\boldsymbol{\beta} = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_s]^T \mathbf{t}_{s+1}$ $\mathbf{w} = \sum_{i=1}^{s} \beta_i (\mathbf{X}_{s+1}^T \mathbf{t}_i + \mathbf{X}_i^T \mathbf{t}_{s+1})$ $\mathbf{w} := \mathbf{w}/\|\mathbf{w}\|$ $\boldsymbol{\beta} := \boldsymbol{\beta}/\left\|\boldsymbol{\beta}\right\|$
3.	Deflation. Deflate $\mathbf{X}$ as $\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T$; $\mathbf{p} = \mathbf{X}^T\mathbf{t}/\mathbf{t}^T\mathbf{t}$
4.	Return to Step 2 to extract the next latent variable, until l latent variables are extracted.
5.	Dynamic inner modeling. Build a VAR model for latent scores based on (21) and (22). Then, $\mathbf{T}_{s+1}$ is predicted as $\hat{\mathbf{T}}_{s+1} = \bar{\mathbf{T}}_s \hat{\boldsymbol{\Theta}}$
6.	Static modeling of prediction errors. Perform traditional PCA on the prediction error matrix $\mathbf{E}_{s+1}$ $\mathbf{E}_{s+1} = \mathbf{X}_{s+1} - \hat{\mathbf{T}}_{s+1}\mathbf{P}^T = \mathbf{T}_r\mathbf{P}_r^T + \mathbf{E}_r$

how the data space is partitioned. To explore the DiPCA geometric properties with j latent variables being extracted, we use a subscript to denote the succession from one latent variable to the next as follows.

$$\mathbf{X}_{j+1} = \mathbf{X}_j - \check{\mathbf{t}}_j \mathbf{p}_j^T \quad \text{with} \quad \mathbf{p}_j = \mathbf{X}_j^T \check{\mathbf{t}}_j / \check{\mathbf{t}}_j^T \check{\mathbf{t}}_j \tag{27}$$

From (27) and the relations in DiPCA algorithm, we can obtain the following lemma.

Lemma 1. *Suppose $i < j$. We have the following relationships among the residual matrices and loading vectors.*

- $\mathbf{X}_j = \mathbf{H}\mathbf{X}_{i+1}$
- $\mathbf{X}_j = \mathbf{X}_{i+1}\mathbf{G}$
- $\mathbf{X}_j\mathbf{w}_i = 0 i < j$
- $\mathbf{X}_j^T \check{\mathbf{t}}_i = 0 \; i < j$
- $\mathbf{w}_i^T \mathbf{p}_i = 1$

where $\mathbf{H}$ *and* $\mathbf{G}$ *are some matrices formed by the DiPCA latent vectors and loadings and are given in the Appendix* A.

With the above Lemma, the following Theorem is given to summarize the orthogonal properties of the DiPCA latent scores and loadings.

Theorem 1. *In the DiPCA algorithm, denoting* $\bar{\mathbf{X}}_s = \sum_{d=1}^{s} \beta_d \mathbf{X}_d$, *then*

$$\mathbf{w} \propto [\mathbf{X}_{s+1}^T \bar{\mathbf{X}}_s + \bar{\mathbf{X}}_s^T \mathbf{X}_{s+1}]\mathbf{w}$$

which means that $\mathbf{w}$ *is the eigenvector of the above symmetric matrix corresponding to the largest eigenvalue. In addition, for Components i, j in the model, the following relations hold.*

- $\mathbf{w}_i^T \mathbf{w}_j = 0 \; \forall i \neq j$
- $\check{\mathbf{t}}_i^T \check{\mathbf{t}}_j = 0 \; \forall i \neq j$
- $\mathbf{w}_i^T \mathbf{p}_j = 0 \; \forall i < j$

where $\check{\mathbf{t}}_i$ and $\check{\mathbf{t}}_j$ denote the score vectors of Components i and j, respectively, to be different from the subscripts that denote time-lagged vectors or matrices. The proof of above theorem can be found in Appendix B.

4.2. DiPCA model relations

Assuming the number of latent variables is chosen as l in the DiPCA model and collecting all latent score vectors and model vectors into the following matrices,

$$\mathbf{T} = \begin{bmatrix} \check{\mathbf{t}}_1 & \check{\mathbf{t}}_2 & \cdots & \check{\mathbf{t}}_l \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_l \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_l \end{bmatrix}$$

we have the following relation by iterating (27),

$$\mathbf{X}_{l+1} = \mathbf{X}_1 - \sum_{j=1}^{l} \check{\mathbf{t}}_j \mathbf{p}_j^T = \mathbf{X} - \mathrm{T}\mathrm{P}^T \tag{28}$$

or

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{X}_{l+1} \tag{29}$$

In many cases, it is desirable to express the scores in terms of the original data for the convenience of model calculations. Post-multiplying (29) by $\mathbf{W}$ and using Relation 3. of Lemma 1, we can obtain

$$\begin{aligned} \mathbf{X}\mathbf{W} &= \mathbf{T}\mathbf{P}^T\mathbf{W} + \mathbf{X}_{l+1}\mathbf{W} \\ &= \mathbf{T}\mathbf{P}^T\mathbf{W} \end{aligned}$$

or

$$\mathbf{T} = \mathrm{X}\mathrm{R} \tag{30}$$

where

$$\mathbf{R} = \mathrm{W}(\mathbf{P}^T\mathbf{W})^{-1} \tag{31}$$

Pre-multiplying (31) by $\mathbf{P}^T$, we can obtain

$$\mathbf{P}^T\mathbf{R} = \mathbf{P}^T\mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1} = \mathbf{I} \tag{32}$$

Therefore,

$$(\mathbf{P}\mathbf{R}^T)(\mathbf{P}\mathbf{R}^T) = \mathbf{P}\mathbf{R}^T \tag{33}$$

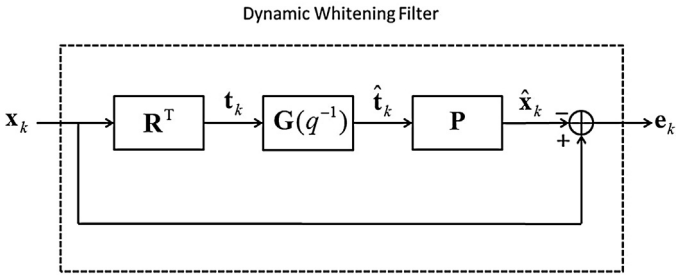


Fig. 1. DiPCA dynamic whitening filter structure.

which is idempotent. Relation (31) and (33) are the same as the relations of **R** and **P** matrices of PLS given in [17,18]. (29) can be rewritten as

$$\mathbf{X} = \mathbf{XRP}^T + \mathbf{X}_{l+1}$$

For a given vector $\mathbf{x}_k$ that would appear as a row in **X**, the score vector is calculated from (30) as follows

$$\mathbf{t}_k = \mathbf{R}^T\mathbf{x}_k \tag{34}$$

and $\mathbf{x}_k$ can be decomposed as

$$\mathbf{x}_k = \mathbf{P}\mathbf{t}_k + \tilde{\mathbf{x}}_k \tag{35}$$

The decomposition as (35) gives the partition of the space formed by current data. To explore how the past data are related to current data, (24) should be used to derive the sample relation for the time k as follows.

$$\mathbf{e}_k = \mathbf{x}_k - \mathbf{P}\hat{\mathbf{t}}_k$$

where $\mathbf{e}_k$ is the one step ahead prediction error and $\mathbf{t}_k$ can be predicted using the result in (22) as

$$\hat{\mathbf{t}}_k = \sum_{i=1}^{s}\mathbf{\Theta}_i^T\mathbf{t}_{k-i} = \mathbf{G}(q^{-1})\mathbf{t}_k$$

where $\mathbf{G}(q^{-1}) = \sum_{i=1}^{s}\mathbf{\Theta}_{s+1-i}^T q^{-i}$ is the transfer function of the dynamic model and q^{-1} is the backward shift operator. Therefore,

$$\mathbf{x}_k = \mathbf{P}\mathbf{G}(q^{-1})\mathbf{t}_k + \mathbf{e}_k \tag{36}$$

With the additional static PCA on $\mathbf{e}_k$, for a sample vector $\mathbf{x}_k$, we have

$$\mathbf{x}_k = \mathbf{P}\mathbf{G}(q^{-1})\mathbf{t}_k + \mathbf{P}_r\mathbf{t}_{k,r} + \mathbf{e}_{k,r} \tag{37}$$

We call $\mathbf{t}_k$ dynamic latent variables, and $\mathbf{t}_{k,r}$ static latent variables.

4.3. Dynamic whitening filter

DiPCA can also be viewed as a preprocessing filter that applied to $\mathbf{x}_k$ before PCA modeling. Since there are possible dynamic relationships in the data, directly applying traditional PCA to $\mathbf{x}_k$ is not appropriate. After DiPCA is first applied to the data $\mathbf{x}_k$, the prediction errors are essentially white, which makes them suitable to use PCA to model the latent structure in $\mathbf{e}_k$. Therefore, DiPCA can be interpreted as a whitening filter which removes all the dynamic relationships in data. Mathematically, based on (34) and (36), the dynamic whitening filter can be written as

$$\mathbf{e}_k = [\mathbf{I} - \mathbf{P}\mathbf{G}(q^{-1})\mathbf{R}^T]\mathbf{x}_k$$

The block diagram of DiPCA dynamic whitening filter is shown in Fig. 1.

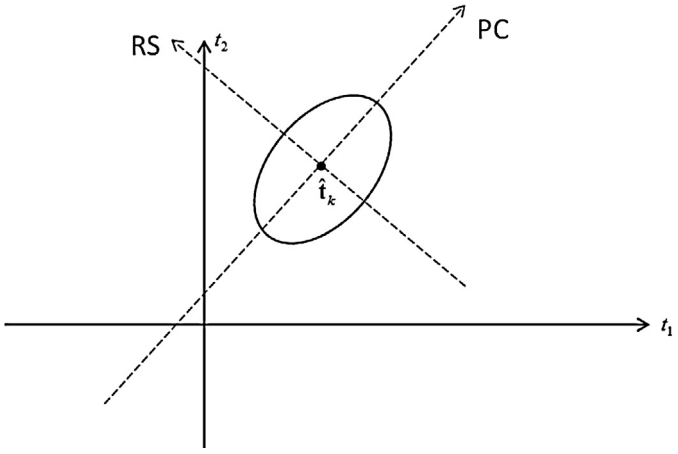


Fig. 2. Confidence region of $\hat{\mathbf{t}}_k$ indicated by φ.

5. Fault detection based on DiPCA

In process monitoring based on PCA, squared prediction error (SPE) and Hotelling's T^2 are typically used for detecting abnormal situations. SPE monitors the variations in the residual subspace, and Hotelling's T^2 monitors variations in the principal component subspace [19,20]. Process monitoring based on DiPCA can be divided into two parts: the monitoring of dynamic relationships and the monitoring of static relationships. Dynamic latent scores, dynamic residuals, static latent scores, and static residuals can be calculated respectively as follows.

$$\begin{aligned}
&\mathbf{t}_k = \mathbf{R}^T\mathbf{x}_k\\
&\mathbf{v}_k = \mathbf{t}_k - \hat{\mathbf{t}}_k;\ \hat{\mathbf{t}}_k = \sum_{i=1}^{s}\mathbf{\Theta}_i^T\mathbf{t}_{k-i}\\
&\mathbf{e}_k = \mathbf{x}_k - \mathbf{P}\hat{\mathbf{t}}_k\\
&\mathbf{t}_{r,k} = \mathbf{P}_r^T\mathbf{e}_k\\
&\mathbf{e}_{r,k} = (\mathbf{I} - \mathbf{P}_r\mathbf{P}_r^T)\mathbf{e}_k
\end{aligned} \tag{38}$$

It is clear from (37) that process monitoring should be performed on $\hat{\mathbf{t}}_k$, $\mathbf{t}_{r,k}$ and $\mathbf{e}_{r,k}$ respectively. However, since $\hat{\mathbf{t}}_k$ is dynamic and could even be unstationary, monitoring $\hat{\mathbf{t}}_k$ can result in high false alarm rate. Therefore, the monitoring of $\hat{\mathbf{t}}_k$ can be performed through $\mathbf{v}_k$. Since $\mathbf{v}_k$ can be cross-correlated, it is appropriate to build another PCA model on $\mathbf{v}_k$ and construct a combined index to monitor $\mathbf{v}_k$ [21]. The combined index for $\mathbf{v}_k$ is defined as

$$\begin{aligned}
&\varphi_v = \mathbf{v}_k^T\mathbf{\Phi}_v\mathbf{v}_k = \frac{T_v^2}{\chi_v^2} + \frac{Q_v}{\delta_v^2}\\
&\mathbf{\Phi}_v = \frac{\mathbf{P}_v\mathbf{\Lambda}_v^{-1}\mathbf{P}_v^T}{\chi_v^2} + \frac{\mathbf{I} - \mathbf{P}_v\mathbf{P}_v^T}{\delta_v^2}
\end{aligned} \tag{39}$$

where $\mathbf{\Lambda}_v = \frac{1}{N-1}\mathbf{V}^T\mathbf{V}$, T_v^2 and Q_v are the fault detection indices for PCA based fault detection on $\mathbf{v}_k$, and χ_v^2 and δ_v^2 are the corresponding control limits as in [21]. In this way, the control region of the prediction $\hat{\mathbf{t}}_k$ can also be indicated by the control region of the combined index φ_v as Fig. 2.

In this figure, the elliptical area indicates the control region of the combined index φ_v, which is also the control region centered around the prediction $\hat{\mathbf{t}}_k$.

The monitoring of $\mathbf{e}_k$ can be performed as traditional PCA based process monitoring, where T_r^2 and Q_r are defined as follows,

$$T_r^2 = \mathbf{t}_{k,r}^T \mathbf{\Lambda}_r^{-1} \mathbf{t}_{r,k} = \mathbf{e}_k^T \mathbf{P}_r \mathbf{\Lambda}_r^{-1} \mathbf{P}_r^T \mathbf{e}_k$$

$$Q_r = \left\| \mathbf{e}_{r,k} \right\|^2 = \mathbf{e}_k^T (\mathbf{I} - \mathbf{P}_r \mathbf{P}_r^T) \mathbf{e}_k$$

where $\mathbf{\Lambda}_r = \frac{1}{N-1} \mathbf{T}_r^T \mathbf{T}_r$. Control limits for T_r^2 and Q_r can be determined based on the results in [21], for instance.

6. Case studies

Two case studies are presented in this section. The first case study is performed on simulation data, and the second case study is performed on Tennessee Eastman Process (TEP) data. The effectiveness of the proposed DiPCA modeling and process monitoring algorithms are demonstrated in both cases.

6.1. Simulation Data

In this simulation, $\mathbf{t}_k$ is generated from a VAR(1) process, and $\mathbf{x}_k$ is generated from a latent variable model as

$$\begin{cases} \mathbf{t}_k = \mathbf{c} + \mathbf{A}\mathbf{t}_{k-1} + \mathbf{v}_k \\ \mathbf{x}_k = \mathbf{P}\mathbf{t}_k + \mathbf{e}_k \end{cases}$$

$$\mathbf{A} = \begin{pmatrix} 0.5205 & 0.1022 & 0.0599 \\ 0.5367 & -0.0139 & 0.4159 \\ 0.0412 & 0.6054 & 0.3874 \end{pmatrix}, \mathbf{c} = \begin{pmatrix} 0.5205 \\ 0.5367 \\ 0.0412 \end{pmatrix}$$

$$\mathbf{P} = \begin{pmatrix} 0.4316 & 0.1723 & -0.0574 \\ 0.1202 & -0.1463 & 0.5348 \\ 0.2483 & 0.1982 & 0.4797 \\ 0.1151 & 0.1557 & 0.3739 \\ 0.2258 & 0.5461 & -0.0424 \end{pmatrix}$$

where $\mathbf{e}_k \in \mathbb{R}^5 \sim N([0, 0.1^2])$, and $\mathbf{v}_k \in \mathbb{R} \sim N([0, 1^2])$. 3000 data points are generated. First 1000 data points are used as a training dataset to train the DiPCA model. The next 1000 data points are used as a validation dataset to determine the dynamic order s and the number of dynamic latent variables l. The last 1000 data points are used as a testing dataset to evaluate the proposed fault detection indices.

By using the method described in Section 3, $s = 1$ and $l = 3$ are selected. Figs. 3 and 4 show the autocorrelation and crosscorrelation of the original variables $\mathbf{x}_k$ and the dynamic latent variables $\mathbf{t}_k$ respectively. It is clear from both figures that strong dynamic relationships exist in $\mathbf{x}_k$ and $\mathbf{t}_k$. After DiPCA modeling, the autocorrelation and crosscorrelation of the prediction error $\mathbf{e}_k$ and the innovation $\mathbf{v}_k$ are plotted in Fig. 5 and 6. The results show that all the autocorrelation and crosscorrelation coefficients are close to zero except at lag 0. This indicates that dynamic relationships are removed from $\mathbf{x}_k$ and $\mathbf{t}_k$, and only static relationships remain in $\mathbf{e}_k$ and $\mathbf{v}_k$, which can be further modeled by PCA.

To test the performance of the fault detection indices, the following two types of faults are added to the test dataset. Since there is no residual subspace for $\mathbf{v}_k$, φ reduces to T_v^2.

1 $\mathbf{t}_k := \mathbf{t}_k + [0\ 5\ 0]^T$, $k > 500$. In this case, the fault occurs in the dynamic components and is detected by φ and T_r^2 indices. However, since this fault does not affect the residual part of the static relationships, it is not detected Q_r index. This can be seen from Fig. 7.
2 When $k > 500$,

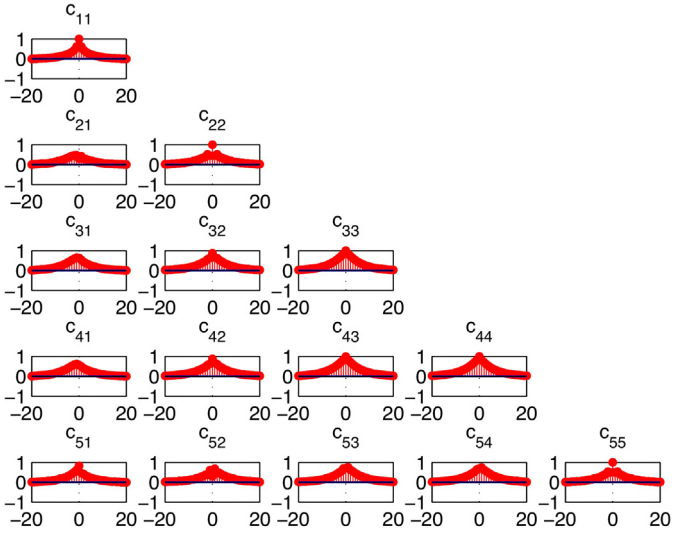


Fig. 3. Autocorrelation and crosscorrelation of $\mathbf{x}_k$ of simulation data.

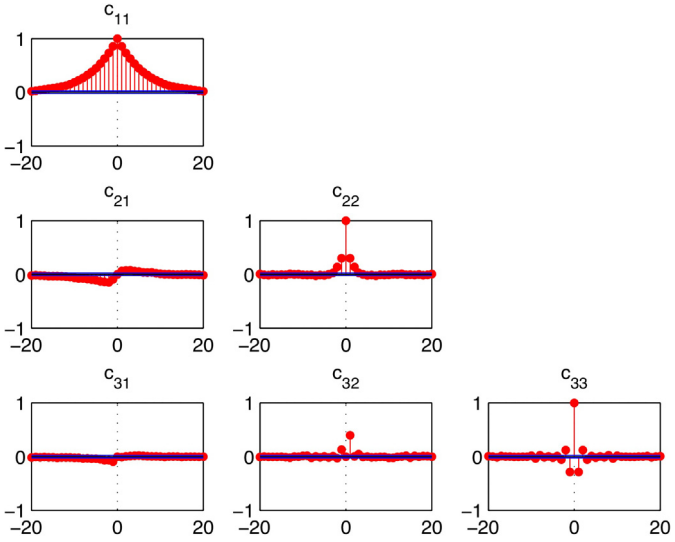


Fig. 4. Autocorrelation and crosscorrelation of $\mathbf{t}_k$ of simulation data.

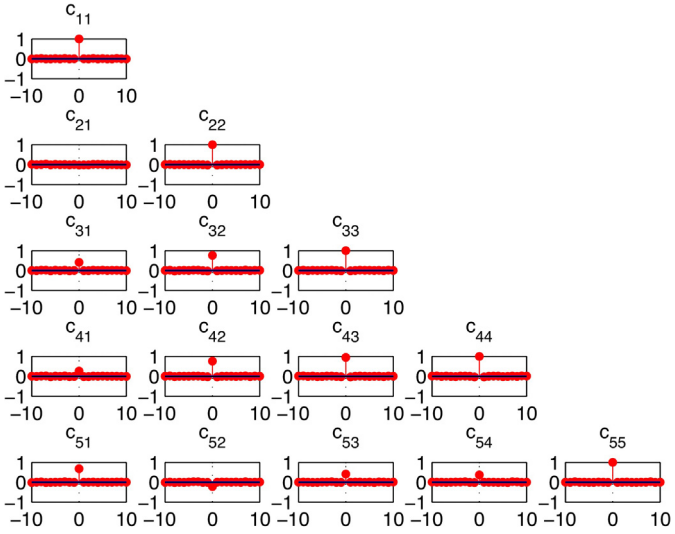


Fig. 5. Autocorrelation and crosscorrelation of $\tilde{\mathbf{x}}_k$ of simulation data.

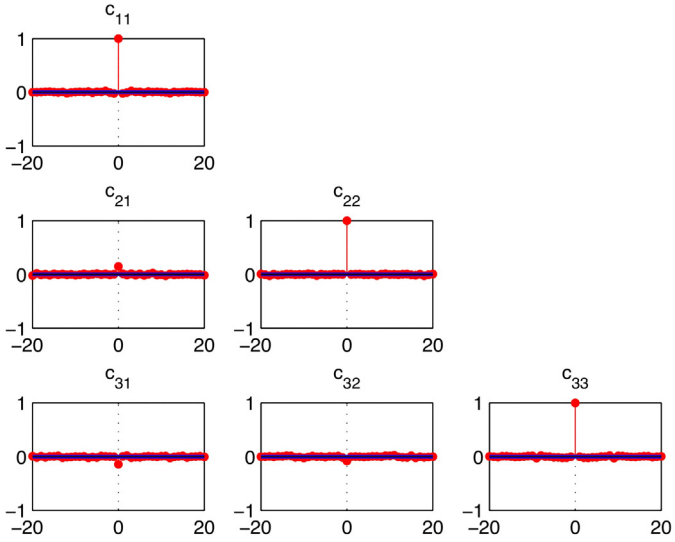


Fig. 6. Autocorrelation and crosscorrelation of $\mathbf{v}_k$ of simulation data.

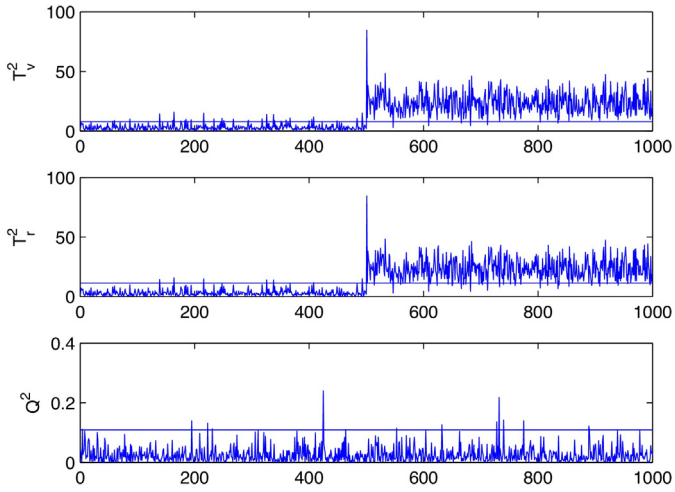


Fig. 7. Fault detection results of fault 1 on simulation data.

$$\mathbf{x}_k := \mathbf{x}_k + 5 * [0.0455 \quad 0.3877 \quad -0.8365 \quad 0.2911 \quad 0.2513]^T$$

In this case, the fault occurs in the residual part of the static relationships. Therefore, this fault is only detected by Q index, other indices remain unaffected as shown in Fig. 8.

3 $\mathbf{A} := 1.2 * \mathbf{A}$, other matrices in the model remain the same. In this case, the VAR model is unstable. 500 data points are used to train the DiPCA model, 100 data points are used as test dataset to check the false alarm rate of three fault detection indices. The results are shown in Fig. 9. It is clear from the figure that T_d^2 has a high false alarm rate when the process is unstable. Therefore, it should not be used for fault detection purpose.

6.2. Data from an industrial boiler

Four sensors with 430 samples were collected from a boiler process under normal operating conditions. 330 samples are used to build the DiPCA model, and the remaining samples are used for testing. $s = 2$ and $l = 2$ are selected according to the method described

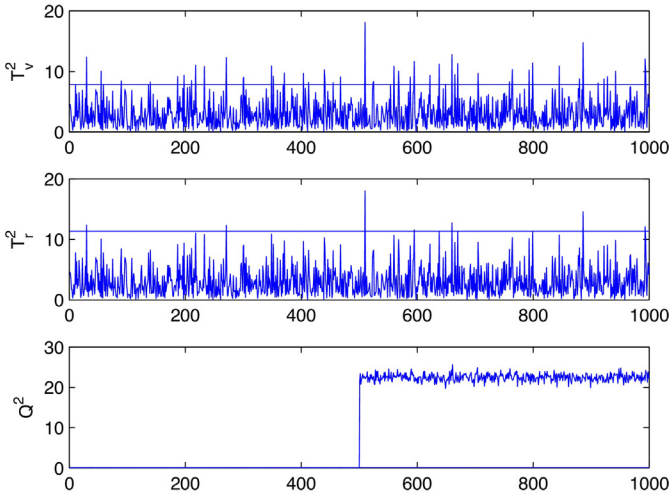


Fig. 8. Fault detection results of fault 2 on simulation data.

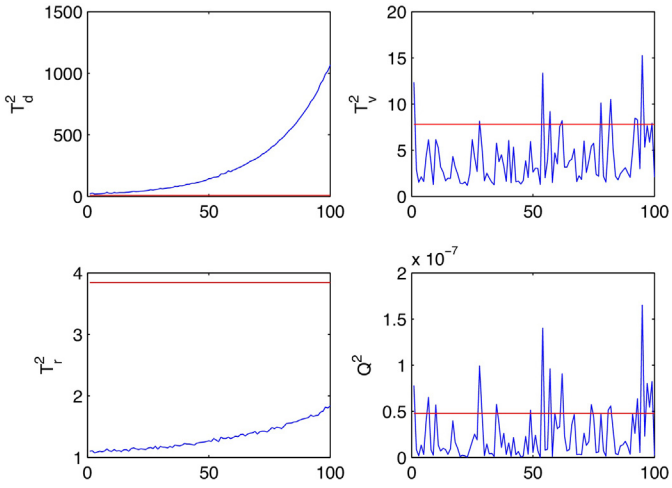


Fig. 9. Fault detection results of faulty data with unstable dynamics.

in Section 3.4. In this example, there is no residual subspace for $\mathbf{v}_k$, therefore, the combined index φ_v reduces to T_v^2. Fig. 10 shows the crosscorrelations of $\mathbf{e}_k$, from which we can see that negligible dynamics remain in the prediction error $\mathbf{e}_k$ after DiPCA modeling.

Figs. 11 and 12 show the crosscorrelation of $\mathbf{t}_k$ and $\mathbf{v}_k$ respectively. It is obvious that strong dynamic relationships exist in $\mathbf{t}_k$, while only static relationships remain in $\mathbf{v}_k$ after a VAR model is built for $\mathbf{t}_k$.

Fig. 13 shows the control region for dynamic components at different time stamps. In Fig. 13, the points with solid or hollow circles indicate $\hat{\mathbf{t}}_k$ and the points marked with asterisks indicate $\mathbf{t}_k$. The ellipses indicate the control region that is centered around the predicted scores. In process monitoring, a sample is considered normal if $\mathbf{t}_k$ is inside the control region centered around $\hat{\mathbf{t}}_k$, and faulty if $\mathbf{t}_k$ is outside the control region. We can see from Fig. 13 that all $\mathbf{t}_k$ are within the control region when no fault occurs. In addition, the control region varies with time. This is the main difference from static PCA where the control region does not change over time.

6.3. TEP data

The Tennessee Eastman Process was developed to provide a realistic simulation of an industrial process for the evaluation of monitoring methods [22]. The process contains 12 manipulated variables and 41 measured variables. The measured variables con-

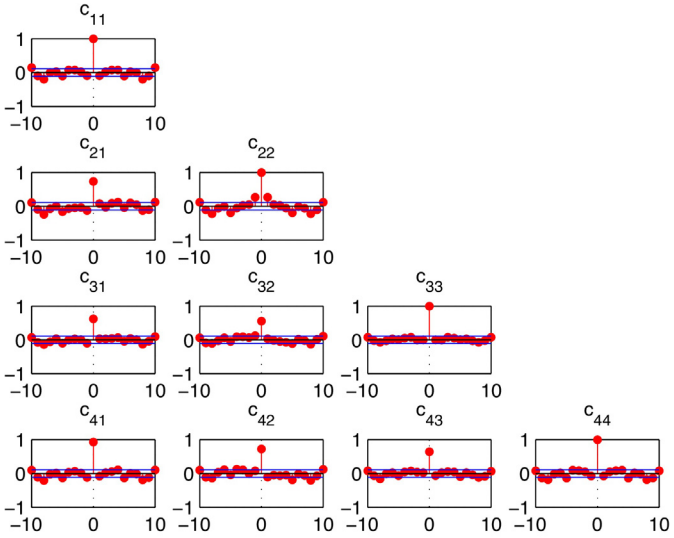


Fig. 10. Crosscorrelation of $\mathbf{e}_k$ of data from an industrial boiler.

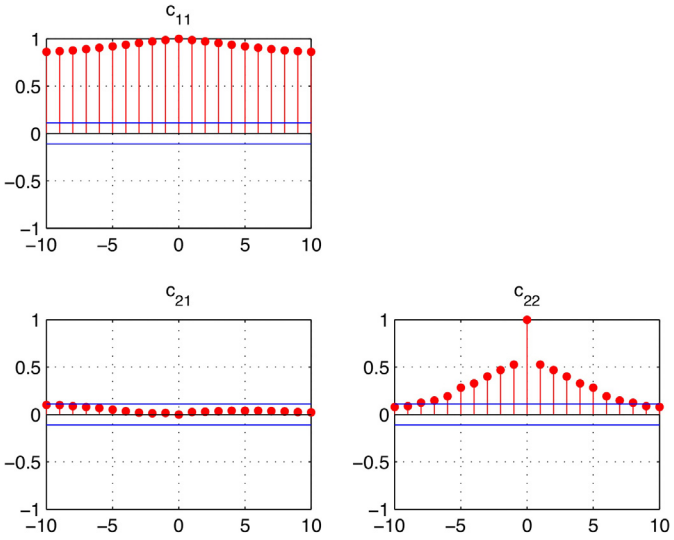


Fig. 11. Crosscorrelation of $\mathbf{t}_k$ of data from an industrial boiler.

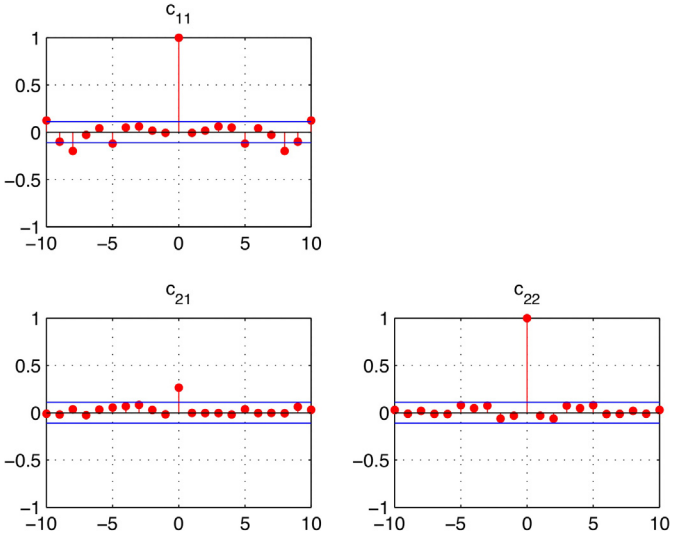


Fig. 12. Crosscorrelation of $\mathbf{v}_k$ of data from an industrial boiler.

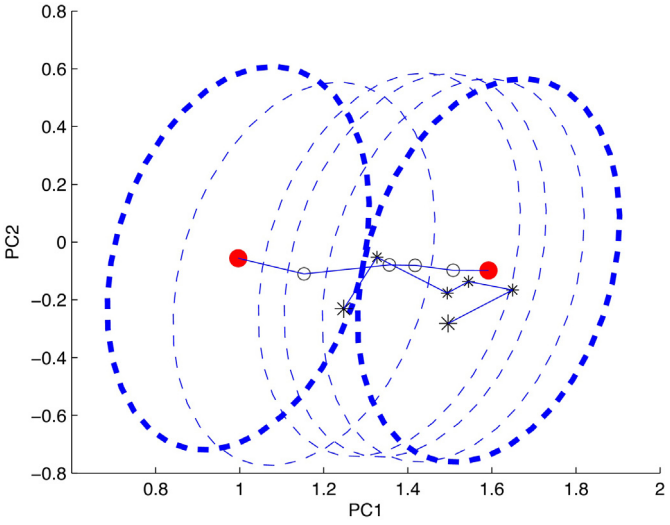


Fig. 13. Control regions for dynamic components at different time stamps.

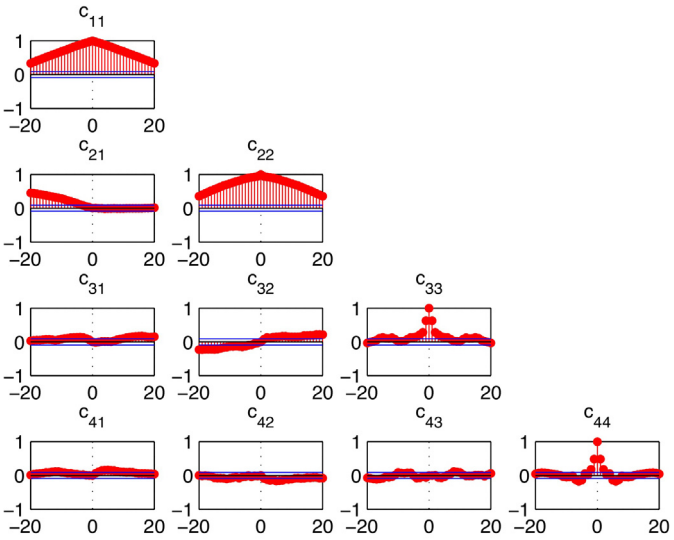


Fig. 14. Autocorrelation and crosscorrelation of $\mathbf{t}_k$ of TEP data.

tain 22 process variables sampled every 3 min, and 19 quality variables sampled with dead time and time delays. In this case study, 22 process variables XMEAS(1-22) and 11 manipulated variables XMV(1-11) are collected as $\mathbf{X}$. 480 data points are used as training dataset. By using the parameter determination method described earlier, $s=3$ and $l=13$ are selected. Fig. 14 shows the autocorrelation and crosscorrelation of the first four dynamic latent variables $\mathbf{t}_k$, and Fig. 15 shows the autocorrelation and crosscorrelation of the innovations $\mathbf{v}_k$. From Fig. 14, we can see that there are obvious autocorrelation and crosscorrelations in $\mathbf{t}_k$. After dynamic modeling, little dynamic crosscorrelations are left in $\mathbf{v}_k$, which can be seen from Fig. 15.

The false alarm rates of φ_v, T_r^2 and Q_r indices are 5.54%, 6.58%, and 9.82% respectively. Table 3 shows the fault detection rates of three fault detection indices for 15 known faults in TEP. The fault detection results show that most of the faults can be detected successfully by the proposed fault detection indices, except for faults IDV(3), IDV(9) and IDV(15). The types of disturbances of these three faults are listed in Table 4. As discussed in [23], these three faults are hard to detect for all the popular process monitoring methods.

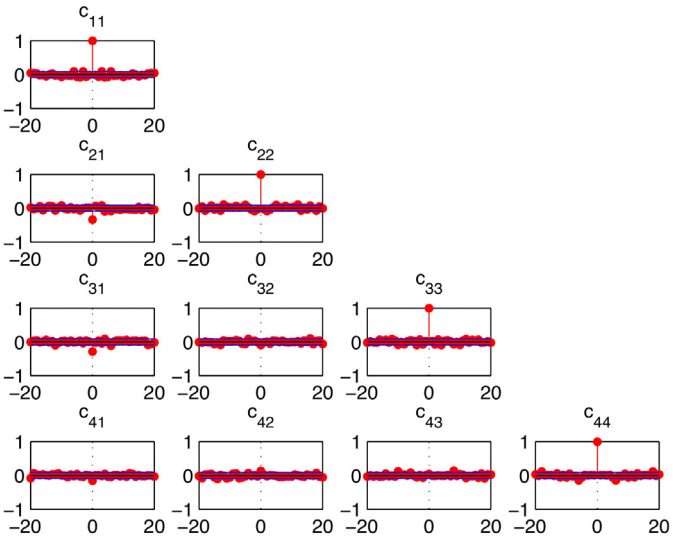


Fig. 15. Autocorrelation and crosscorrelation of $\mathbf{v}_k$ of TEP data.

Table 3
Fault detection rate based on the DiPCA model.

Fault type	φ_v	T_r^2	Q_r
IDV(1)	100	99.50	100
IDV(2)	99.00	98.62	97.87
IDV(3)	6.65	7.53	12.55
IDV(4)	97.49	100	27.73
IDV(5)	22.08	22.33	97.74
IDV(6)	100	99.37	100
IDV(7)	83.56	100	88.33
IDV(8)	95.86	94.10	95.98
IDV(9)	7.53	8.41	13.17
IDV(10)	15.18	13.93	77.16
IDV(11)	76.66	88.83	42.53
IDV(12)	95.23	95.61	99.00
IDV(13)	94.86	92.35	96.74
IDV(14)	100	100	100
IDV(15)	7.15	8.28	13.43

Table 4
Description of undetected disturbances. These disturbances have little impact on the overall process behavior due to feedback control.

Type	Description
IDV(3)	D feed temperature (step)
IDV(9)	D feed temperature (random variation)
IDV(15)	Condenser cooling water valve (sticking)

7. Conclusions

In this paper, a dynamic inner PCA algorithm is developed for dynamic data modeling by maximizing the covariance between the component and the prediction from its past values. In the proposed method, a dynamic latent variable model is extracted first to capture the most auto-covarying dynamics in the data. The captured dynamic components contain the self-predictable variations of the data, while the residuals after extracting the dynamic components are essentially uncorrelated, least predictable and can be treated using static PCA. A cross-validation method is given to determine the order and the number of dynamic latent variables.

Geometric properties of DiPCA method are derived to give insight into the DiPCA model structure. The DiPCA objective guarantees that static variations in the data are left in the residuals after extracting dynamic components. In this case, the DiPCA model can be interpreted as a whitening filter with a dynamic latent structure, where the dimension of the dynamic latent space is generally smaller than the dimension of the original variable space. This feature is not available in other time series modeling or the Kalman filter approach.

For the purpose of process monitoring, three fault detection indices for the dynamic latent model and the subsequent static PCA model are derived. After filtering out the principal dynamics, the residual data are essentially white and therefore appropriate for PCA based monitoring. Since the whitening filter leaves white noise in the residuals, the overall residual space after DiPCA filtering has the same dimension as the original measurement space, but with deflated variance-covariance. The extracted dynamic principal components provide clear dynamic visualization and monitoring. Case studies on simulation data, boiler process data, and the TEP data demonstrated that the proposed DiPCA modeling method provides a unique way to separate dynamic variations with latent variables from static variations. Further, fault detection indices are made reliable after separating the dynamic variations from static ones.

Acknowledgements

This work is supported by funds from the National Natural Science Foundation of China (61490704), The Fundamental Research Program of the Shenzhen Committee on Science and Innovations (Ji20160207), and the Texas-Wisconsin-California Control Consortium.

Appendix A. Proof of Lemma 1

1 From (27) we have

$$\begin{aligned}\mathbf{X}_j &= \mathbf{X}_{j-1} - \check{\mathbf{t}}_{j-1}\mathbf{p}_{j-1}^T \\ &= \mathbf{X}_{j-1} - \check{\mathbf{t}}_{j-1}\check{\mathbf{t}}_{j-1}^T\mathbf{X}_{j-1}/\check{\mathbf{t}}_{j-1}^T\check{\mathbf{t}}_{j-1} \\ &= (\mathbf{I} - \check{\mathbf{t}}_{j-1}\check{\mathbf{t}}_{j-1}^T/\check{\mathbf{t}}_{j-1}^T\check{\mathbf{t}}_{j-1})\mathbf{X}_{j-1}\end{aligned} \tag{A.1}$$

It is clear that this is a recursive relation for $\mathbf{X}_j$. Iterating this relation until $\mathbf{X}_{i+1}$ appears, Relation 1. in Lemma 1 will be resulted with

$$\mathbf{H} = (\mathbf{I} - \check{\mathbf{t}}_{j-1}\check{\mathbf{t}}_{j-1}^T/\check{\mathbf{t}}_{j-1}^T\check{\mathbf{t}}_{j-1})\cdots(\mathbf{I} - \check{\mathbf{t}}_{i+1}\check{\mathbf{t}}_{i+1}^T/\check{\mathbf{t}}_{i+1}^T\check{\mathbf{t}}_{i+1}) \tag{A.2}$$

2 In (A.1) if $\check{\mathbf{t}}_j$ is replaced by $\check{\mathbf{t}}_j = \mathbf{X}_j\mathbf{w}_j$, we have

$$\begin{aligned}\mathbf{X}_j &= \mathbf{X}_{j-1} - \check{\mathbf{t}}_{j-1}\mathbf{p}_{j-1}^T = \mathbf{X}_{j-1} - \mathbf{X}_{j-1}\mathbf{w}_{j-1}\mathbf{p}_{j-1}^T \\ &= \mathbf{X}_{j-1}(\mathbf{I} - \mathbf{w}_{j-1}\mathbf{p}_{j-1}^T)\end{aligned} \tag{A.3}$$

Again, this is another recursive relation for $\mathbf{X}_j$. Iterating this relation until $\mathbf{X}_{i+1}$ appears, Relation 2. of Lemma 1 will be resulted.

3 From Relation 1. in Lemma 1 and (A.1) we have

$$\begin{aligned}\mathbf{X}_j\mathbf{w}_i &= \mathbf{H}\mathbf{X}_{i+1}\mathbf{w}_i = \mathbf{H}(\mathbf{I} - \check{\mathbf{t}}_i\check{\mathbf{t}}_i^T/\check{\mathbf{t}}_i^T\check{\mathbf{t}}_i)\mathbf{X}_i\mathbf{w}_i \\ &= \mathbf{H}(\mathbf{I} - \check{\mathbf{t}}_i\check{\mathbf{t}}^T/\check{\mathbf{t}}_i^T\check{\mathbf{t}}_i)\check{\mathbf{t}}_i = 0\end{aligned} \tag{A.4}$$

which proves Relation 3. of the Lemma 1.

4 From Relation 2. and (A.1) again, we have

$$\check{\mathbf{t}}_i^T\mathbf{X}_j = \check{\mathbf{t}}_i^T\mathbf{X}_{i+1}\mathbf{G} = \check{\mathbf{t}}_i^T(\mathbf{I} - \check{\mathbf{t}}_i\check{\mathbf{t}}_i^T/\check{\mathbf{t}}_i^T\check{\mathbf{t}}_i)\mathbf{X}_i\mathbf{G} = 0 \tag{A.5}$$

using the same reasoning.

5 Using the expression for $\mathbf{p}_i$, Relation 5. of Lemma 1 can be easily shown as follows.

$$\mathbf{w}_i^T\mathbf{p}_i = \mathbf{w}_i^T\mathbf{X}_i^T\check{\mathbf{t}}_i/\check{\mathbf{t}}_i^T\check{\mathbf{t}}_i = \check{\mathbf{t}}_i^T\check{\mathbf{t}}_i/\check{\mathbf{t}}_i^T\check{\mathbf{t}}_i = 1 \tag{A.6}$$

Appendix B. Proof of Theorem 1

To show relations 1. and 2. for $i \neq j$, it suffices to show them for the case of $i<j$, since the case of $i>j$ can be shown by symmetry. Using the expression for $\mathbf{w}_j$ we have,

$$\mathbf{w}_j = c\sum_{d=1}^{s}\beta_{d,j}(\mathbf{X}_{s+1,j}^T\check{\mathbf{t}}_{d,j} + \mathbf{X}_{d,j}^T\check{\mathbf{t}}_{s+1,j}) \tag{B.1}$$

where c is a proportional constant. Since $\mathbf{X}_{d,j}$ and $\mathbf{X}_{s+1,j}$ are sub-matrices of $\mathbf{X}_j$, $\mathbf{w}_i^T\mathbf{X}_j^T = 0$ implies $\mathbf{w}_i^T\mathbf{X}_{d,j}^T = 0$ and $\mathbf{w}_i^T\mathbf{X}_{s+1,j}^T = 0$, which further implies $\mathbf{w}_i^T\mathbf{w}_j = 0$. This is obvious from Relations 3. of Lemma 1. Furthermore, using Relation 4. of Lemma 1,

$$\check{\mathbf{t}}_i^T\check{\mathbf{t}}_j = \check{\mathbf{t}}_i^T\mathbf{X}_j\mathbf{w}_j = 0 \tag{B.2}$$

Using Relation 3. of Lemma 1 again,

$$\mathbf{w}_i^T\mathbf{p}_j = \mathbf{w}_i^T\mathbf{X}_j^T\check{\mathbf{t}}_j/\check{\mathbf{t}}_j^T\check{\mathbf{t}}_j = 0 \tag{B.3}$$

References

[1] J.V. Kresta, J.F. MacGregor, T.E. Marlin, Multivariate statistical monitoring of process operating performance, Can. J. Chem. Eng. 69 (1) (1991) 35–47.
[2] B.M. Wise, N. Ricker, D. Veltkamp, B.R. Kowalski, A theoretical basis for the use of principal component models for monitoring multivariate processes, Process Control Qual. 1 (1) (1990) 41–51.
[3] S. Joe Qin, Statistical process monitoring: basics and beyond, J. Chemom. 17 (8–9) (2003) 480–502.
[4] J.F. MacGregor, C. Jaeckle, C. Kiparissides, M. Koutoudi, Process monitoring and diagnosis by multiblock PLS methods, AIChE J. 40 (5) (1994) 826–838.
[5] D. Zhou, G. Li, S.J. Qin, Total projection to latent structures for process monitoring, AIChE J. 56 (1) (2010) 168–178.
[6] A. Negiz, E.S. Lagergren, A. Cinar, Statistical quality control of multivariable continuous processes, American Control Conference, vol. 2 (1994) 1289–1293.
[7] W. Ku, R.H. Storer, C. Georgakis, Disturbance detection and isolation by dynamic principal component analysis, Chemom. Intel. Lab. Syst. 30 (1) (1995) 179–196.
[8] J. Chen, K.-C. Liu, On-line batch process monitoring using dynamic PCA and dynamic PLS models, Chem. Eng. Sci. 57 (1) (2002) 63–75.
[9] N. Lu, Y. Yao, F. Gao, F. Wang, Two-dimensional dynamic PCA for batch process monitoring, AIChE J. 51 (12) (2005) 3300–3304.
[10] W. Lin, Y. Qian, X. Li, Nonlinear dynamic principal component analysis for on-line process monitoring and diagnosis, Comput. Chem. Eng. 24 (2) (2000) 423–429.
[11] G. Li, B. Liu, S.J. Qin, D. Zhou, Dynamic latent variable modeling for statistical process monitoring, in: Proc. IFAC World Congress, Milan, Italy, 2011, pp. 12886–12891.
[12] G. Li, S.J. Qin, D. Zhou, A new method of dynamic latent-variable modeling for process monitoring, IEEE Trans. Ind. Electron. 61 (11) (2014) 6438–6445.
[13] Y. Dong, S.J. Qin, Dynamic-inner partial least squares for dynamic data modeling, IFAC-Papers OnLine 48 (8) (2015) 117–122.
[14] L. Ljung, System Identification, Wiley Online Library, 1999.
[15] G.E. Box, G.M. Jenkins, G.C. Reinsel, Time Series Analysis: Forecasting and Control, 734, John Wiley & Sons, 2011.
[16] S. Valle, W. Li, S.J. Qin, Selection of the number of principal components: the variance of the reconstruction error criterion with a comparison to other methods, Ind. Eng. Chem. Res. 38 (11) (1999) 4389–4401.
[17] H. Martens, T. Naes, K. Norris, Multivariate Calibration by Data Compression, 2001.
[18] I.S. Helland, On the structure of partial least squares regression, Commun. Stat. Simul. Comput. 17 (2) (1988) 581–607.
[19] J.E. Jackson, A User's Guide to Principal Components, 587, John Wiley & Sons, 2005.
[20] J.E. Jackson, G.S. Mudholkar, Control procedures for residuals associated with principal component analysis, Technometrics 21 (3) (1979) 341–349.
[21] H.H. Yue, S.J. Qin, Reconstruction-based fault identification using a combined index, Ind. Eng. Chem. Res. 40 (20) (2001) 4403–4414.
[22] J.J. Downs, E.F. Vogel, A plant-wide industrial process control problem, Comput. Chem. Eng. 17 (3) (1993) 245–255.
[23] S. Yin, S.X. Ding, A. Haghani, H. Hao, P. Zhang, A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process, J. Process Control 22 (9) (2012) 1567–1581.

Journal of Process Control 68 (2018) 64–72

Contents lists available at ScienceDirect

Journal of Process Control

journal homepage: www.elsevier.com/locate/jprocont

Regression on dynamic PLS structures for supervised learning of dynamic data

Yining Dong [a], S. Joe Qin [a,b,*,1]

[a] Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, United States
[b] Mork Family Department of Chemical Engineering and Materials Science, University of Southern California, Los Angeles, CA 90089, United States

ARTICLE INFO

Article history:
Received 18 October 2016
Received in revised form 26 March 2018
Accepted 21 April 2018

Keywords:
Dynamic partial least squares
Data-driven modeling

ABSTRACT

Partial least squares (PLS) regression is widely used to capture the latent relationship between inputs and outputs in static system modeling. Several dynamic PLS algorithms have been proposed to capture the characteristics of dynamic data. However, none of these algorithms provides an explicit expression for the dynamic inner and outer models. In this paper, a dynamic inner PLS algorithm is proposed for dynamic data modeling. The proposed algorithm provides an explicit dynamic inner model that is ensured in deriving the outer model. Several examples are presented to demonstrate the effectiveness of the proposed algorithm.

1. Introduction

Partial least squares or projection to latent structures (PLS) has been developed to model inter-related variables with a latent structure between two blocks of data, i.e., an input block and an output block. A set of latent variables with a reduced dimension is extracted in PLS that removes collinearity among the original input variables. A notable feature of PLS is that it is not sensitive to data collinearity in input and output compared to its predecessor, canonical correlation analysis (CCA) [1] when they are used as a regression method. The first work that gives the idea of PLS is [2], which bears its connection to principal component analysis (PCA). However, it is not until [3] the method began to become known to the chemometrics community for multivariate calibration [4]. The tutorial work of [5] makes the method popularized, while [6] is among the first to give a comprehensive account of PLS model properties. The work by Li et al. [7] further reveals the geometric properties of the PLS projection and its variants. In addition to its ability to predict the output as a regression method, PLS tries to interpret the variance structure in the input data that co-vary with the output data, which makes it a supervised data analytic tool [8]. However, the two objectives are not always achievable simultaneously. It can happen that a small variance direction in the input space is most useful to predict the output [7]. In this situation a large portion of variance in the input is unexplained in PLS. A recent work by [9] provides a concurrent PLS modeling and monitoring scheme for the input and output data in the latent space that is suitable to monitor both input space anomalies and output space anomalies simultaneously.

However, to model dynamics in the data, PLS has to be extended, since the original PLS achieves a static model relation only. PLS usually involves an outer model that serves to reduce the dimension of the input and output space to derive latent factors and an inner model that serves to predict output latent scores from input scores. In static PLS, neither the inner model nor the outer model involves any dynamic relations. A straightforward method is proposed by [10,11], where a number of lagged inputs are included in the augmented input matrix. Finite impulse response (FIR) inner models are built between inputs and outputs after outer models converge. The disadvantage of this method is that it does not give an explicit representation of the latent relationship. The augmented loading matrix is even larger in dimension than the original input space and is difficult to interpret.

The work in [12] proposes a modified PLS modeling algorithm. It provides a compact representation as no lagged variables are included in the outer model. Data are pre-filtered with the hope that dynamic components in the inputs are removed. A dynamic inner model and a static outer model are built between filtered inputs and outputs. Lakshminarayanan et al. [13] propose a similar method by building dynamic inner relationship between input scores and out-

* Corresponding author at: The Mork Family Department of Chemical Engineering and Materials Science, University of Southern California, Los Angeles, CA 90089, United States.
E-mail address: sqin@usc.edu (S.J. Qin).
[1] Work on this paper was done when the author was on leave from the University of Southern California.

https://doi.org/10.1016/j.jprocont.2018.04.006
0959-1524/

put scores, but the outer model is not compatible with the inner model since it ignores the dynamics in the data entirely.

Recently, [14] proposed a dynamic PLS method by utilizing a weighted combination of lagged input data as the input to the algorithm. An inner model is built between output scores and a weighted combination of lagged input scores. This gives a compact model and a compatible inner and outer model. However, the inner model is not explicit and difficult to interpret.

In this paper, a dynamic inner PLS (DiPLS) algorithm is proposed. The proposed algorithm involves an explicit dynamic model and outer model that is compatible with the inner dynamic model structure. An iterative method, derived by applying the Lagrange multiplier technique, is proposed to maximize the covariance between the output latent scores and the predicted scores from input scores with a dynamic inner model relation. The resulting dynamic model makes the output dynamically related to the latent variables, while the latent variables are a projection of the input variables to a lower-dimensional subspace. More importantly, the dynamic latent variables are extracted with the supervision of the output data, making it radically different from dynamic factor models. The explicit and compatible model representation makes it easy to interpret the results.

The remaining sections of the paper are organized as follows. Section 2 reviews the traditional PLS objective function and presents a dynamic PLS objective that is dictated by a dynamic inner model. Section 3 presents the proposed DiPLS algorithm based on the new dynamic PLS objective. Section 4 presents the geometric properties of DiPLS and its model relationships. Section 5 presents several examples to show the effectiveness of the DiPLS algorithm. Section 6 gives conclusions.

2. Dynamic partial least squares formulations

2.1. The PLS objective

A complete form of PLS is first given by [3] to perform regression with collinear input variables, which is common for routine operation data. The PLS algorithm performs regression with collinear input variables by projecting to a lower dimensional latent space one dimension at a time. This is effectively a version of conjugate gradient methods for the linear regression problem. This approach not only avoids direct inversion of a potentially ill-conditioned matrix in ordinary least squares, it also provides a way to trade off between the model prediction variance and bias by selecting an appropriate number of latent variables through cross-validation. PLS extracts a pair of latent variables from inputs and outputs such that their covariance is maximized. First, loading vectors for inputs and outputs are used to generate the latent variables. Then a linear inner model is built between input scores and output scores. The input scores are used to deflate the input matrix, while the estimated output scores calculated from the inner linear model are used to deflate the output matrix.

Consider the input matrix $\mathbf{X}$ and output matrix $\mathbf{Y}$, the objective of PLS is

$$\begin{aligned} &\max \quad \mathbf{q}^T\mathbf{Y}^T\mathbf{X}\mathbf{w} \\ &\text{s.t.} \quad \|\mathbf{w}\| = 1, \|\mathbf{q}\| = 1 \end{aligned} \tag{1}$$

where $\mathbf{w}$ and $\mathbf{q}$ are input and output weights, respectively. The solution to this optimization problem with the Lagrange multiplier technique is given as follows

$$\begin{aligned} &\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}\mathbf{w} = \lambda_w\mathbf{w} \\ &\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y}\mathbf{q} = \lambda_q\mathbf{q} \end{aligned} \tag{2}$$

which indicates λ_w and λ_q are eigenvalues and $\mathbf{w}$ and $\mathbf{q}$ are eigenvectors of the corresponding matrices. After the outer model converges, latent score vectors $\mathbf{t}$ and $\mathbf{u}$ can be calculated as $\mathbf{t}=\mathbf{X}\mathbf{w}$, and $\mathbf{u}=\mathbf{Y}\mathbf{q}$. An inner model can be built between $\mathbf{u}$ and $\mathbf{t}$ by a simple linear regression as follows

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T \tag{3}$$

$$\mathbf{Y} := \mathbf{Y} - b\mathbf{t}\mathbf{q}^T \tag{4}$$

where $b=\mathbf{u}^T\mathbf{t}/(\mathbf{t}^T\mathbf{t})$ is obtained by regressing $\mathbf{u}$ to $\mathbf{t}$. This process is iterated until enough factors are extracted. Details for the PLS algorithm can be found in [6,5].

2.2. Dynamic inner PLS objective

It is clear from the objective and procedure of PLS that only static relations in the input and output are extracted by PLS. To build dynamic PLS (DPLS) models, a straightforward approach is to extend the input matrix with time-lagged inputs, as proposed in [11] in a nonlinear dynamic PLS scheme. While this DPLS approach is simple, it is difficult to interpret the extracted latent factors, and the dynamic model tends to have excessive parameters.

An alternative approach proposed by [12,13] keeps the outer model the same as that in static PLS, but builds a dynamic inner model between $\mathbf{u}$ and $\mathbf{t}$. This approach is inconsistent between the outer model treatment and inner model treatment, as the static outer model via the static objective has no intention to extract dynamics in the latent scores $\mathbf{u}$ and $\mathbf{t}$. It is possible that $\mathbf{u}$ and $\mathbf{t}$ are statically uncorrelated but dynamically correlated, making this approach fail.

Therefore, the outer model of a dynamic PLS should be extracted based on the same dynamic inner relation, such as

$$u_k = \beta_0 t_k + \beta_1 t_{k-1} + \cdots + \beta_s t_{k-s} + r_k$$

with the latent variables related to original variables as follows

$$\begin{aligned} u_k &= \mathbf{y}_k^T\mathbf{q} \\ t_k &= \mathbf{x}_k^T\mathbf{w} \end{aligned}$$

where $\mathbf{x}_k$ and $\mathbf{y}_k$ are the input and output vectors at time k. For each factor, the inner model prediction should be

$$\begin{aligned} \hat{u}_k &= \mathbf{x}_k^T\mathbf{w}\beta_0 + \mathbf{x}_{k-1}^T\mathbf{w}\beta_1 + \cdots + \mathbf{x}_{k-s}^T\mathbf{w}\beta_s \\ &= [\mathbf{x}_k^T \quad \mathbf{x}_{k-1}^T \cdots \mathbf{x}_{k-s}^T](\boldsymbol{\beta}\otimes\mathbf{w}) \end{aligned}$$

where $\boldsymbol{\beta}=(\beta_0\beta_1\cdots\beta_s)^T$ and $\boldsymbol{\beta}\otimes\mathbf{w}$ is the Kronecker product. The outer model that is consistent with the above inner dynamic model should maximize the covariance between u_k and $\hat{u}_k$, that is, to maximize

$$\frac{1}{N}\sum_{k=s}^{N+s}\mathbf{q}^T\mathbf{y}_k[\mathbf{x}_k^T \quad \mathbf{x}_{k-1}^T\cdots\mathbf{x}_{k-s}^T](\boldsymbol{\beta}\otimes\mathbf{w}) \tag{5}$$

This objective is the dynamic inner PLS (DiPLS) objective that is first proposed in [15], which gives rise to the DiPLS algorithm.

Let $\mathbf{x}_k$ and $\mathbf{y}_k$ be the input and output vectors at time k and $N+s+1$ samples of input and output are collected to form the following matrices

$$\begin{aligned} \mathbf{X} &= [\mathbf{x}_0 \quad \mathbf{x}_1 \quad \cdots \quad \mathbf{x}_{s+N}]^T \in \Re^{m\times(N+s+1)} \\ \mathbf{Y} &= [\mathbf{y}_0 \quad \mathbf{y}_1 \quad \cdots \quad \mathbf{y}_{s+N}]^T \in \Re^{p\times(N+s+1)} \end{aligned}$$

Form the following data matrices using the samples,

$$\mathbf{X}_i = [\mathbf{x}_i \quad \mathbf{x}_{i+1} \quad \cdots \quad \mathbf{x}_{i+N}]^T, \quad \text{for} \quad i = 0, 1, 2, \ldots, s \tag{6}$$

$$\mathbf{Z}_s = [\mathbf{X}_s \quad \mathbf{X}_{s-1} \quad \cdots \mathbf{X}_0]$$
$$\mathbf{Y}_s = [\mathbf{y}_s \quad \mathbf{y}_{s+1} \quad \cdots \quad \mathbf{y}_{s+N}]^T$$

The objective of DiPLS in (5) can be reformulated as

$$\begin{aligned} \max \quad & J = \mathbf{q}^T\mathbf{Y}_s^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) \\ \text{s.t.} \quad & \|\mathbf{w}\| = 1, \|\mathbf{q}\| = 1, \left\|\boldsymbol{\beta}\right\| = 1 \end{aligned} \tag{7}$$

where s is the impulse response order of the model. The dimension of $\mathbf{w}$ is the same as the number of input variables. Note that the matrix $\mathbf{Z}_s$ contains $\mathbf{X}_s$, which makes the inner model to include a direct contribution from $\mathbf{x}_k$ to $\mathbf{y}_k$. If $s=0$ so that $\mathbf{Y}_s$ is related to $\mathbf{X}_s$ only, DiPLS reduces to the static PLS.

3. Dynamic inner PLS algorithms

Lagrange multipliers are used to solve this optimization in (7). Define

$$\begin{aligned} L = \quad & \mathbf{q}^T\mathbf{Y}_s^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) + \frac{1}{2}\lambda_q(1-\mathbf{q}^T\mathbf{q}) \\ & + \frac{1}{2}\lambda_\beta(1-\boldsymbol{\beta}^T\boldsymbol{\beta}) + \frac{1}{2}\lambda_w(1-\mathbf{w}^T\mathbf{w}) \end{aligned} \tag{8}$$

where

$$(\boldsymbol{\beta}\otimes\mathbf{w}) = (\boldsymbol{\beta}\otimes\mathbf{I})\mathbf{w} = (\mathbf{I}\otimes\mathbf{w})\boldsymbol{\beta} \tag{9}$$

Taking derivatives with respective to $\mathbf{q}$, $\mathbf{w}$, $\boldsymbol{\beta}$ and setting the results to zero leads to the following results [15]

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{q}} &= \mathbf{Y}_s^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) - \lambda_q\mathbf{q} = 0 \\ \frac{\partial L}{\partial \mathbf{w}} &= (\boldsymbol{\beta}\otimes\mathbf{I})^T\mathbf{Z}_s^T\mathbf{Y}_s\mathbf{q} - \lambda_w\mathbf{w} = 0 \\ \frac{\partial L}{\partial \boldsymbol{\beta}} &= (\mathbf{I}\otimes\mathbf{w})^T\mathbf{Z}_s^T\mathbf{Y}_s\mathbf{q} - \lambda_\beta\boldsymbol{\beta} = 0 \end{aligned} \tag{10}$$

Multiplying $\mathbf{q}^T$, $\mathbf{w}^T$ and $\boldsymbol{\beta}^T$ on the left of the above three equations, respectively, and making use of (9), it is easy to show that

$$\lambda_q = \lambda_w = \lambda_\beta = J$$

That is, all of the multipliers are equal. Eq. (10) gives the solution for $\mathbf{q}$, $\mathbf{w}$ and $\boldsymbol{\beta}$ with a unknown proportional constant. Since these vectors are unit norm, we find them as follows.

$$\mathbf{q} = \mathbf{Y}_s^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}); \quad \mathbf{q}{:=}\mathbf{q}/\|\mathbf{q}\| \tag{11}$$

$$\mathbf{w} = (\boldsymbol{\beta}\otimes\mathbf{I})^T\mathbf{Z}_s^T\mathbf{Y}_s\mathbf{q}; \quad \mathbf{w}{:=}\mathbf{w}/\|\mathbf{w}\| \tag{12}$$

$$\boldsymbol{\beta} = (\mathbf{I}\otimes\mathbf{w})^T\mathbf{Z}_s^T\mathbf{Y}_s\mathbf{q}; \quad \boldsymbol{\beta}{:=}\boldsymbol{\beta}/\left\|\boldsymbol{\beta}\right\| \tag{13}$$

The latent variables of input and output can be calculated as

$$\begin{aligned} \mathbf{X}\mathbf{w} &= \mathbf{t} \in \Re^{N+s+1} \\ \mathbf{Y}\mathbf{q} &= \mathbf{u} \in \Re^{N+s+1} \end{aligned} \tag{14}$$

We further denote $\mathbf{t}_i$ and $\mathbf{u}_s$ as follows

$$\begin{aligned} \mathbf{t}_i &= [t_i \quad t_{i+1} \quad \cdots \quad t_{i+N}]^T = \mathbf{X}_i\mathbf{w}, \quad i = 0, 1, \ldots, s \\ \mathbf{u}_s &= [u_s \quad u_{s+1} \quad \cdots \quad u_{s+N}]^T = \mathbf{Y}_s\mathbf{q} \end{aligned} \tag{15}$$

where $\mathbf{X}_i$ is defined in (6). The Kronecker expressions (11)–(13) can be simplified using (14) and (15) as follows

$$\mathbf{w} = (\boldsymbol{\beta}\otimes\mathbf{I})^T\mathbf{Z}_s^T\mathbf{Y}_s\mathbf{q} = \sum_{i=0}^{s}\beta_i\mathbf{X}_{s-i}^T\mathbf{Y}_s\mathbf{q} = \mathbf{X}_\beta^T\mathbf{u}_s$$
$$\mathbf{q} = \mathbf{Y}_s^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) = \mathbf{Y}_s^T\sum_{i=0}^{s}\beta_i\mathbf{X}_{s-i}\mathbf{w} = \mathbf{Y}_s^T\sum_{i=0}^{s}\beta_i\mathbf{t}_{s-i}$$
$$\boldsymbol{\beta} = [\mathbf{t}_s \quad \mathbf{t}_{s-1} \quad \cdots \quad \mathbf{t}_0]^T\mathbf{u}_s$$

where

$$\mathbf{X}_\beta = \sum_{i=0}^{s}\beta_i\mathbf{X}_{s-i} \tag{16}$$

The inner model describing the dynamic relationship between input scores and output scores should, therefore, be built between $\mathbf{u}_s$ and $\mathbf{t}_s, \mathbf{t}_{s-1}, \cdots \mathbf{t}_0$ by minimizing

$$\left\|\mathbf{u}_s - \sum_{i=0}^{s}\hat{\beta}_i\mathbf{t}_{s-i}\right\|^2$$

Denoting $\hat{\boldsymbol{\beta}} = \begin{bmatrix}\hat{\beta}_0 & \hat{\beta}_1 & \cdots & \hat{\beta}_s\end{bmatrix}^T$ and $\mathbf{T}_s = [\mathbf{t}_s \quad \mathbf{t}_{s-1} \quad \mathbf{t}_0]$, the least squares solution is

$$\hat{\boldsymbol{\beta}} = (\mathbf{T}_s^T\mathbf{T}_s)^{-1}\mathbf{T}_s^T\mathbf{u}_s \tag{17}$$

and the model estimate

$$\hat{\mathbf{u}}_s = \sum_{i=0}^{s}\hat{\beta}_i\mathbf{t}_{s-i} = \mathbf{T}_s\hat{\boldsymbol{\beta}} \tag{18}$$

The loading vector $\mathbf{p}$ for $\mathbf{X}$ can be derived as

$$\mathbf{p} = \mathbf{X}^T\mathbf{t}/\mathbf{t}^T\mathbf{t}$$

Delation of $\mathbf{X}$ and $\mathbf{Y}_s$ are performed as

$$\begin{aligned} \mathbf{X}{:=}\mathbf{X} - \mathbf{t}\mathbf{p}^T \\ \mathbf{Y}_s{:=}\mathbf{Y}_s - \hat{\mathbf{u}}_s\mathbf{q}^T \end{aligned} \tag{19}$$

Subsequent factors can be obtained from these residuals by repeating the same procedure. The algorithm of DiPLS modeling is summarized in Appendix A.

3.1. DiPLS with one output only

If there is only one output, the traditional PLS is known as PLS1. In PLS1, the loading vector $\mathbf{q}=1$, and the output score $\mathbf{u}=\mathbf{y}$. Therefore, the iteration procedure reduces to one-step calculation of the loading vector $\mathbf{w}$ and input score $\mathbf{t}$. We denote DiPLS in the case of only one output to be DiPLS1. Similar to PLS1, $\mathbf{q}=1$ and $\mathbf{u}_s=\mathbf{y}_s$ in DiPLS1. Using (13), it is clear that $\boldsymbol{\beta}$ should be a unit vector in the direction $(\mathbf{I}\otimes\mathbf{w})^T\mathbf{Z}_s^T\mathbf{y}_s$, that is,

$$\beta_i = \mathbf{y}_s^T\mathbf{X}_{s-i}\mathbf{w}$$

From (12) we have

$$\mathbf{w} \propto \sum_{i=0}^{s}\beta_i\mathbf{X}_{s-i}^T\mathbf{y}_s = \sum_{i=0}^{s}\mathbf{X}_i^T\mathbf{y}_s\mathbf{y}_s^T\mathbf{X}_i\mathbf{w}$$

where "$\propto$" denotes that both sides of it are proportional. Define

$$\mathbf{y}_{s,i}^T = \begin{bmatrix}\mathbf{0}_i^T & \mathbf{y}_s^T & \mathbf{0}_{s-i}^T\end{bmatrix}$$

where $\mathbf{0}_i \in \mathbb{R}^i$ is a zero vector. It is easy to show that

$$\mathbf{X}_i^T\mathbf{y}_s = \mathbf{X}^T\mathbf{y}_{s,i}, \quad \text{for} \quad i = 0, 1, \ldots, s$$

Therefore,

$$\mathbf{w} \propto \sum_{i=0}^{s}\mathbf{X}^T\mathbf{y}_{s,i}\mathbf{y}_{s,i}^T\mathbf{X} = \mathbf{X}^T\bar{\mathbf{Y}}_s\bar{\mathbf{Y}}_s^T\mathbf{X}$$

where

$$\bar{\mathbf{Y}}_s = \begin{bmatrix}\mathbf{y}_{s,0} & \mathbf{y}_{s,1} & \cdots & \mathbf{y}_{s,s}\end{bmatrix}$$

Now we can see that $\mathbf{w}$ is the eigenvector of $\mathbf{X}^T\bar{\mathbf{Y}}_s\bar{\mathbf{Y}}_s^T\mathbf{X}$ corresponding to the largest eigenvalue. After $\mathbf{w}$ is calculated, $\hat{\boldsymbol{\beta}}$ can be obtained using (17). The portion of the output explained by DiPLS1 factor is

$$\hat{\mathbf{y}}_s = \hat{\mathbf{u}}_s = \sum_{i=0}^{s}\hat{\beta}_i\mathbf{t}_{s-i}$$

and the residual of $\mathbf{y}_s$ is

$$\mathbf{y}_s := \mathbf{y}_s - \hat{\mathbf{u}}_s$$

3.2. DiPLS with auto-regressive dynamics

When it is desirable to include the auto-regression of the output, the objective of DiPLS can be modified as

$$\begin{aligned}\max \quad & \mathbf{q}^T\mathbf{Y}_s\left[\left(\sum_{i=1}^{s}\alpha_i\mathbf{Y}_{s-i}\right)\mathbf{q} + \left(\sum_{j=0}^{s}\beta_j\mathbf{X}_{s-j}\right)\mathbf{w}\right] \\ \text{s.t.} \quad & \|\mathbf{w}\|^2 = 1, \|\mathbf{q}\|^2 = 1, \left\|\boldsymbol{\beta}\right\|^2 + \|\boldsymbol{\alpha}\|^2 = 1\end{aligned} \tag{20}$$

where α's are the coefficients of different lags of $\mathbf{Y}$. Denote

$$\begin{aligned}\mathbf{V}_s &= [\mathbf{Y}_{s-1} \quad \mathbf{Y}_{s-2} \quad \cdots \quad \mathbf{Y}_0] \\ \boldsymbol{\alpha} &= [\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_s]^T\end{aligned}$$

Lagrange multipliers can be used to solve the optimization problem in (20). Define

$$\begin{aligned}L = \; & \mathbf{q}^T\mathbf{Y}_s^T\left[\mathbf{V}_s(\boldsymbol{\alpha}\otimes\mathbf{q}) + \mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w})\right] + \frac{1}{2}\lambda_q(1-\mathbf{q}^T\mathbf{q}) \\ & + \frac{1}{2}\lambda_w(1-\mathbf{w}^T\mathbf{w}) + \frac{1}{2}\lambda(1-\boldsymbol{\alpha}^T\boldsymbol{\alpha}-\boldsymbol{\beta}^T\boldsymbol{\beta})\end{aligned} \tag{21}$$

Taking derivatives with respective to $\mathbf{q}$, $\mathbf{w}$, $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$, and setting the results to zero leads to the following results [15]

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{q}} &= [\mathbf{Y}_s^T\mathbf{V}_s(\boldsymbol{\alpha}\otimes\mathbf{I}) + (\boldsymbol{\alpha}\otimes\mathbf{I})^T\mathbf{V}_s^T\mathbf{Y}_s]\mathbf{q} + \mathbf{Y}_s^T\mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) - \lambda_q\mathbf{q} = 0 \\ \frac{\partial L}{\partial \mathbf{w}} &= (\boldsymbol{\beta}\otimes\mathbf{I})^T\mathbf{Z}_s^T\mathbf{Y}_s\mathbf{q} - \lambda_w\mathbf{w} = 0 \\ \frac{\partial L}{\partial \boldsymbol{\beta}} &= (\mathbf{I}\otimes\mathbf{w})^T\mathbf{Z}_s^T\mathbf{Y}_s\mathbf{q} - \lambda\boldsymbol{\beta} = 0 \\ \frac{\partial L}{\partial \boldsymbol{\alpha}} &= (\mathbf{I}\otimes\mathbf{q})^T\mathbf{V}_s^T\mathbf{Y}_s\mathbf{q} - \lambda\boldsymbol{\alpha} = 0\end{aligned} \tag{22}$$

In addition, defining

$$\begin{aligned}\mathbf{Y}_\alpha &= \mathbf{V}_s(\boldsymbol{\alpha}\otimes\mathbf{I}) = \sum_{i=1}^{s}\alpha_i\mathbf{Y}_{s-i} \\ \mathbf{u}_\alpha &= \mathbf{V}_s(\boldsymbol{\alpha}\otimes\mathbf{q}) = \mathbf{Y}_\alpha\mathbf{q} \\ \mathbf{t}_\beta &= \mathbf{Z}_s(\boldsymbol{\beta}\otimes\mathbf{w}) = \mathbf{X}_\beta\mathbf{w} \\ \mathbf{U}_s &= [\mathbf{u}_s \quad \cdots \quad \mathbf{u}_1] = \mathbf{V}_s(\mathbf{I}\otimes\mathbf{q})\end{aligned}$$

(22) can be further simplified as

$$\begin{aligned}&\mathbf{Y}_s^T\mathbf{u}_\alpha + \mathbf{Y}_s^T\mathbf{t}_\beta + \mathbf{Y}_\alpha^T\mathbf{u}_s - \lambda_q\mathbf{q} = 0 \\ &\mathbf{X}_\beta^T\mathbf{u}_s - \lambda_w\mathbf{w} = 0 \\ &\begin{bmatrix}\mathbf{U}_s^T \\ \mathbf{T}_s^T\end{bmatrix}\mathbf{u}_s = \lambda\begin{bmatrix}\boldsymbol{\alpha} \\ \boldsymbol{\beta}\end{bmatrix}\end{aligned} \tag{23}$$

where $\mathbf{q}$, $\mathbf{w}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ are unit vectors. Eq. (23) leads to an iterative solution procedure to solve (20) with proper initialization of $\mathbf{u}_s$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

The inner model is subsequently built as an ARX model of $\mathbf{u}_s$,

$$\mathbf{u}_s = \sum_{i=1}^{s}\hat{\alpha}_i\mathbf{u}_{s-i} + \sum_{i=0}^{s}\hat{\beta}_i\mathbf{t}_{s-i} + \mathbf{r}_s \tag{24}$$

Then the prediction of output score $\hat{\mathbf{u}}_s$ can be calculated using $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ which are formed from (24) using least squares.

4. DiPLS geometric properties and relations

4.1. DiPLS geometric properties

The DiPLS algorithm is different from the more traditional dynamic PLS algorithms, which simply augment the input data with time lagged values. Therefore, it is important to understand the geometric properties of the DiPLS projections and how the data space is partitioned when a dynamic inner model as well as a dynamic outer model is employed. While the DiPLS objective (7) maximizes the dynamic covariance between a latent variable of $\mathbf{Y}$ and a latent variable of $\mathbf{X}$, the actual DiPLS algorithm in Appendix A has embedded in it important geometric properties such as orthogonality to ensure numerical stability and robustness. To explore the DiPLS geometric properties extracting j latent variables (LV), we use a subscript j to denote the iteration from one LV to another as follows.

$$\mathbf{X}_{j+1} = \mathbf{X}_j - \mathbf{t}_j\mathbf{p}_j^T \quad \text{with} \quad \mathbf{p}_j = \mathbf{X}_j^T\mathbf{t}_j/\mathbf{t}_j^T\mathbf{t}_j \tag{25}$$

$$\mathbf{Y}_{s,j+1} = \mathbf{Y}_{s,j} - \hat{\mathbf{u}}_j\mathbf{q}_j^T \tag{26}$$

From (25) and the relations in Appendix A, we can obtain the following lemma.

Lemma 1. *Suppose $i<j$. We have the following relationships among the residual matrices and loading vectors.*

(1) $\mathbf{X}_j = \mathbf{H}\mathbf{X}_{i+1}$
(2) $\mathbf{X}_j = \mathbf{X}_{i+1}\mathbf{Z}$
(3) $\mathbf{X}_j\mathbf{w}_i = 0 \; \forall \; i<j$
(4) $\mathbf{X}_j^T\mathbf{t}_i = 0 \quad \forall \quad i<j$
(5) $\mathbf{w}_i^T\mathbf{p}_i = 1$

Proof

1. From (25) we have

$$\begin{aligned}\mathbf{X}_j &= \mathbf{X}_{j-1} - \mathbf{t}_{j-1}\mathbf{p}_{j-1}^T \\ &= \mathbf{X}_{j-1} - \mathbf{t}_{j-1}\mathbf{t}_{j-1}^T\mathbf{X}_{j-1}/\mathbf{t}_{j-1}^T\mathbf{t}_{j-1} \\ &= (\mathbf{I} - \mathbf{t}_{j-1}\mathbf{t}_{j-1}^T/\mathbf{t}_{j-1}^T\mathbf{t}_{j-1})\mathbf{X}_{j-1}\end{aligned} \tag{27}$$

   It is clear that this is a recursive relation for $\mathbf{X}_j$. Iterating this relation until $\mathbf{X}_{i+1}$ appears, Lemma 1 is proven with

$$\mathbf{H} = (\mathbf{I} - \mathbf{t}_{j-1}\mathbf{t}_{j-1}^T/\mathbf{t}_{j-1}^T\mathbf{t}_{j-1})\cdots(\mathbf{I} - \mathbf{t}_{i+1}\mathbf{t}_{i+1}^T/\mathbf{t}_{i+1}^T\mathbf{t}_{i+1})$$

2. In Step 1 if $\mathbf{t}_j$ is replaced by $\mathbf{t}_j = \mathbf{X}_j\mathbf{w}_j$, we have

$$\begin{aligned}\mathbf{X}_j &= \mathbf{X}_{j-1} - \mathbf{t}_{j-1}\mathbf{p}_{j-1}^T = \mathbf{X}_{j-1} - \mathbf{X}_{j-1}\mathbf{w}_{j-1}\mathbf{p}_{j-1}^T \\ &= \mathbf{X}_{j-1}(\mathbf{I} - \mathbf{w}_{j-1}\mathbf{p}_{j-1}^T)\end{aligned}$$

   Again, this is another recursive relation for $\mathbf{X}_j$. Iterating this relation until $\mathbf{X}_{i+1}$ appears, Relation (2) of Lemma 1 is obtained.
3. From Relation (1) in Lemma 1 and (27) we have

$$\begin{aligned}\mathbf{X}_j\mathbf{w}_i &= \mathbf{H}\mathbf{X}_{i+1}\mathbf{w}_i = \mathbf{H}(\mathbf{I} - \mathbf{t}_i\mathbf{t}_i^T/\mathbf{t}_i^T\mathbf{t}_i)\mathbf{X}_i\mathbf{w}_i \\ &= \mathbf{H}(\mathbf{I} - \mathbf{t}_i\mathbf{t}_i^T/\mathbf{t}_i^T\mathbf{t}_i)\mathbf{t}_i = 0\end{aligned}$$

   which proves Relation (3) of the Lemma.
4. From Relation (2) and (27) again, we have

$$\mathbf{t}_i^T\mathbf{X}_j = \mathbf{t}_i^T\mathbf{X}_{i+1}\mathbf{Z} = \mathbf{t}_i^T(\mathbf{I} - \mathbf{t}_i\mathbf{t}_i^T/\mathbf{t}_i^T\mathbf{t}_i)\mathbf{X}_i\mathbf{Z} = 0$$

   using the same reasoning.
5. Using the expression for $\mathbf{p}_i$, Relation (5) of Lemma 1 can be easily shown as follows.

$$\mathbf{w}_i^T\mathbf{p}_i = \mathbf{w}_i^T\mathbf{X}_i^T\mathbf{t}_i/\mathbf{t}_i^T\mathbf{t}_i = \mathbf{t}_i^T\mathbf{t}_i/\mathbf{t}_i^T\mathbf{t}_i = 1$$

   with the above Lemma we give the following theorem to summarize some orthogonal properties of the DiPLS latent scores and loadings.

Theorem 1. *For the DiPLS algorithm in Appendix A, the following relations hold.*

(1) $\mathbf{w}_i^T\mathbf{w}_j = 0 \quad \forall i \neq j$
(2) $\mathbf{t}_i^T\mathbf{t}_j = 0 \quad \forall i \neq j$
(3) $\mathbf{w}_i^T\mathbf{p}_j = 0 \quad \forall i < j$

Proof. To show relations (1) and (2) for $i \neq j$, it suffices to show them for the case of $i<j$, since the case of $i>j$ can be shown by symmetry. Using the expression for $\mathbf{w}_j$ in Appendix A we have,

$$\mathbf{w}_j = c\sum_{d=0}^{s}\beta_{d,j}\mathbf{X}_{s-d,j}^T\mathbf{u}_{s,j}$$

where c is a proportional constant. Since $\mathbf{X}_{s-d,j}$ is a sub-matrix of $\mathbf{X}_j$, $\mathbf{w}_i^T\mathbf{X}_j^T = 0$ implies $\mathbf{w}_i^T\mathbf{X}_{s-d,j}^T = 0$, which further implies $\mathbf{w}_i^T\mathbf{w}_j = 0$. This is obvious from Relations (3) of Lemma 1. Furthermore, using Relation (4) of Lemma 1,

$$\mathbf{t}_i^T\mathbf{t}_j = \mathbf{t}_i^T\mathbf{X}_j\mathbf{w}_j = 0$$

Using Relation (3) of Lemma 1 again,

$$\mathbf{w}_i^T\mathbf{p}_j = \mathbf{w}_i^T\mathbf{X}_j^T\mathbf{t}_j/\mathbf{t}_j^T\mathbf{t}_j = 0$$

□

4.2. DiPLS model relations

Assuming the number of latent variables is chosen as l in the DiPLS model and denoting

$$\begin{aligned}\mathbf{T} &= \begin{bmatrix}\mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_l\end{bmatrix} \\ \hat{\mathrm{U}}_s &= \begin{bmatrix}\hat{\mathrm{u}}_{s,1} & \hat{\mathrm{u}}_{s,2} & \cdots & \hat{\mathrm{u}}_{s,l}\end{bmatrix} \\ \mathbf{W} &= \begin{bmatrix}\mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_l\end{bmatrix} \\ \mathbf{P} &= \begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_l\end{bmatrix} \\ \mathbf{Q} &= \begin{bmatrix}\mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_l\end{bmatrix}\end{aligned}$$

we have the following relations by iterating (25) and (26),

$$\mathbf{X}_{l+1} = \mathbf{X}_1 - \sum_{j=1}^{l}\mathbf{t}_j\mathbf{p}_j^T = \mathbf{X} - \mathrm{TP}^T \tag{28}$$

$$\mathbf{Y}_{s,l+1} = \mathbf{Y}_{s,1} - \sum_{j=1}^{l}\hat{\mathrm{u}}_{s,j}\mathbf{q}_j^T = \mathrm{Y_s} - \hat{\mathrm{U}}_\mathrm{s}\mathrm{Q}^T \tag{29}$$

or

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{X}_{l+1} \tag{30}$$

$$\mathbf{Y}_s = \hat{\mathrm{U}}_\mathrm{s}\mathrm{Q}^T + \mathbf{Y}_{s,l+1} \tag{31}$$

The DiPLS model estimations for $\mathbf{X}$ and $\mathbf{Y}$ are

$$\hat{\mathrm{X}} = \mathbf{T}\mathbf{P}^T \tag{32}$$

$$\hat{\mathrm{Y}}_s = \hat{\mathrm{U}}_s\mathbf{Q}^T \tag{33}$$

Since the scores $\mathbf{t}_j$ of the DiPLS model are related to the residuals $\mathbf{X}_j$ by $\mathbf{w}_j$, it is desirable to express the scores in terms of the original data for the convenience of model calculations. Post-multiplying (30) by $\mathbf{W}$ and using Relation (3) of Lemma 1, we can obtain

$$\begin{aligned}\mathbf{X}\mathbf{W} &= \mathbf{T}\mathbf{P}^T\mathbf{W} + \mathbf{X}_{l+1}\mathbf{W} \\ &= \mathbf{T}\mathbf{P}^T\mathbf{W}\end{aligned}$$

or

$$\mathbf{T} = \mathrm{XR} \tag{34}$$

where

$$\mathbf{R} = \mathrm{W}(\mathbf{P}^T\mathbf{W})^{-1} \tag{35}$$

This relation is the same as the PLS $\mathbf{R}$ matrix given in [16–18].

The DiPLS projection of $\mathbf{X}$ can be derived from (32) as follows.

$$\hat{\mathrm{X}} = \mathbf{X}\mathbf{R}\mathbf{P}^T \tag{36}$$

For a given input vector $\mathbf{x}_k$, which acts as a row of $\mathbf{X}$, the DiPLS scores, input estimate, and input residual are, respectively,

$$\mathbf{t}_k = \mathbf{R}^T\mathbf{x}_k \tag{37}$$

$$\hat{\mathbf{x}}_k = \mathbf{P}\mathbf{R}^T\mathbf{x}_k = \mathbf{P}\mathbf{t}_k \tag{38}$$

$$\tilde{\mathbf{x}}_k = (\mathbf{I} - \mathrm{PR}^T)\mathbf{x}_k \tag{39}$$

where $\tilde{\mathbf{x}}_k$ is the residual of the DiPLS projection.

From (35) it is easy to see that

$$\mathbf{P}^T\mathbf{R} = \mathbf{P}^T\mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1} = \mathbf{I} \tag{40}$$

Therefore,

$$(\mathbf{P}\mathbf{R}^T)(\mathbf{P}\mathbf{R}^T) = \mathbf{P}\mathbf{R}^T \tag{41}$$

which is idempotent. Since $\mathbf{PR}^T$ is not symmetric in general, $\mathbf{PR}^T$ is an oblique projector, similar to the PLS projections shown in [7],

$$\Pi_{P|R^\perp} = \mathbf{PR}^T \tag{42}$$

$$\Pi_{R^\perp|P} = \mathbf{I} - \mathrm{PR}^T \tag{43}$$

where $\Pi_{P|R^\perp}$ is a projector onto the subspace $Span\{\mathbf{P}\}$ along the subspace $Span\{\mathbf{R}\}^\perp$. Although the DiPLS projections share the same expressions as the PLS projections, the DiPLS partitions of the data space are radically different from the PLS partitions.

The DiPLS output estimate is less conventional, but can be derived from (33) by making use of the jth inner model in Appendix A,

$$\hat{\mathbf{u}}_{s,j} = \sum_{i=0}^{s} \hat{\beta}_{i,j}\mathbf{t}_{s-i,j} = b_j(q^{-1})\mathbf{t}_{s,j}$$

where $b_j(q^{-1}) = \sum_{i=0}^{s}\hat{\beta}_{i,j}q^{-i}$ is the transfer function of the jth inner model and q^{-1} the backward shift operator. Denoting further

$$\mathbf{B}(q^{-1}) = diag\{b_1(q^{-1}), b_2(q^{-1}), \ldots, b_l(q^{-1})\}$$

Equation (33) can be expressed as follows,

$$\hat{\mathrm{Y}}_s = \hat{\mathrm{U}}_s\mathbf{Q}^T = \mathbf{T}_s\mathbf{B}(q^{-1})\mathbf{Q}^T = \mathbf{X}_s\mathbf{R}\mathbf{B}(q^{-1})\mathbf{Q}^T \tag{44}$$

where $\mathbf{T}_s$ is defined similarly to $\mathbf{X}_s$. The estimated $\hat{\mathbf{y}}_k$, which takes a row of $\hat{\mathrm{Y}}_s$, for a given input vector $\mathbf{x}_k$ or $\mathbf{t}_k$ is

$$\hat{\mathbf{y}}_k = \mathbf{Q}\mathbf{B}(q^{-1})\mathbf{t}_k = \mathbf{Q}\mathbf{B}(q^{-1})\mathbf{R}^T\mathbf{x}_k \tag{45}$$

Note that (38) and (44) jointly give the DiPLS latent variable model. The estimate of the output is related to the input by a transfer matrix.

4.3. Exhausting DiPLS residuals

A major attractive feature of PLS is its ability to deal with highly collinear input data in $\mathbf{X}$. The objective of PLS is to extract latent variables that are most relevant to exploring the output variations while effectively exploring the variance in the input space. In the case that the rank of $\mathbf{X}$ is less than the number of input variables, it is known that the number of PLS factors does not go beyond the rank of $\mathbf{X}$ [19]. In other words all variations in $\mathbf{X}$ are represented with the number of LV's up to the rank of $\mathbf{X}$. The above feature, however, is not available for traditional dynamic PLS that simply augments lagged inputs to form an extended input matrix, as in this case the extended input matrix usually has a much larger dimension than the original data matrix. Fortunately, this feature is preserved in DiPLS, since dynamic inner relations are built in the model. We summarize this feature of DiPLS in the following theorem.

Theorem 2. *Assuming rank*$(\mathbf{X}) = r < m$*, where* m *is the number of columns of* $\mathbf{X}$*, then, after* r *DiPLS factors are extracted, the input residuals* $\mathbf{X}_{r+j} = 0$ *for* $j \geq 0$.

Proof. Based on Relation (1) of Lemma 1, it is sufficient to show that $\mathbf{X}_{r+1} = 0$.

After r DiPLS factors are extracted, all latent scores are collected in $\mathbf{T}_r = [\mathbf{t}_1\mathbf{t}_2\cdots\mathbf{t}_r]$. By iterating the DiPLS outer model relations, we have

$$\begin{aligned}\mathbf{X}_{r+1} &= \mathbf{X}_r - \mathbf{t}_r\mathbf{p}_r^T = (\mathbf{I} - \mathbf{t}_r\mathbf{t}_r^T/\mathbf{t}_r^T\mathbf{t}_r)\mathbf{X}_r \\ &= \cdots\cdots \\ &= (\mathbf{I} - \mathbf{t}_r\mathbf{t}_r^T/\mathbf{t}_r^T\mathbf{t}_r)\cdots(\mathbf{I} - \mathbf{t}_1\mathbf{t}_1^T/\mathbf{t}_1^T\mathbf{t}_1)\mathbf{X}\end{aligned}$$

Using the orthogonality of $\mathbf{t}_j$ in Theorem 1, it is straightforward to show that the above relation is equivalent to

$$\mathbf{X}_{r+1} = (\mathbf{I} - \mathbf{T}_r(\mathbf{T}_r^T\mathbf{T}_r)^{-1}\mathbf{T}_r^T)\mathbf{X}$$

which implies that $\mathbf{X}_{r+1}$ is the projection of $\mathbf{X}$ on to $span(\mathbf{T}_r)^\perp$.

This theorem has several implications in practice. First, DiPLS needs no more than m factors to exhaust all variabilities in the data. This is a drastic improvement over the conventional dynamic PLS where the input matrix is augmented with lagged data, which usually requires a large number of LVs to explore variabilities in the data. Second, when the input data are not fully excited, which is typical for routinely collected operational data, DiPLS needs only up to the rank of $\mathbf{X}$ to extract all variabilities in the data. Since the number of LV's is closely related to and upper-bounded by the rank of $\mathbf{X}$, the extracted latent factors are fewer than those extracted by traditional DPLS and are easy to interpret.

4.4. Eigenvector and singular vector interpretations

As in the static PLS, where the loadings can be interpreted as eigenvectors or singular vectors of some matrices, the DiPLS loadings also have eigenvector and singular vector interpretations. Focusing on the DiPLS algorithm in Appendix A and using (16), the outer model relations in Appendix A lead to

$$\mathbf{w} \propto \mathbf{X}_\beta^T\mathbf{u}_s = \mathbf{X}_\beta^T\mathbf{Y}_s\mathbf{q} \tag{46}$$

$$\mathbf{q} \propto \mathbf{Y}_s^T\sum_{i=0}^{s}\beta_i\mathbf{X}_{s-i}\mathbf{w} = \mathbf{Y}_s^T\mathbf{X}_\beta\mathbf{w} \tag{47}$$

where "$\propto$" denotes that the quantities on both sides differ by a proportional constant. The above relations can be rearranged as follows,

$$\mathbf{w} \propto \mathbf{X}_\beta^T\mathbf{Y}_s\mathbf{Y}_s^T\mathbf{X}_\beta\mathbf{w}$$

$$\mathbf{q} \propto \mathbf{Y}_s^T\mathbf{X}_\beta\mathbf{X}_\beta^T\mathbf{Y}_s\mathbf{q}$$

which indicates that $\mathbf{w}$ and $\mathbf{q}$ are indeed eigenvectors of the respective matrices. Since the DiPLS objective aims to maximize the co-variation, these eigenvectors are the ones that correspond to the largest eigenvalues of the respective matrices. Consequently, from (46) it is straightforward to show that $\mathbf{w}$ and $\mathbf{q}$ are the left and right singular vectors of $\mathbf{X}_\beta^T\mathbf{Y}_s$ corresponding to the largest singular value. This interpretation is analogous to that given in [20] for static PLS.

5. Case studies

In this section, three sets of data are simulated, each corresponding to a scenario. In Scenario 1, both input $\mathbf{X}$ and $\mathbf{Y}$ are generated from a static model. In Scenario 2, input $\mathbf{X}$ is generated from a static model, and output $\mathbf{Y}$ is generated from a dynamic model. In Scenario 3, input $\mathbf{X}$ is generated from a dynamic model, and output $\mathbf{Y}$ is generated from a static model. The advantages of DiPLS over traditional PLS is demonstrated in these three basic examples.

5.1. Scenario 1

$\mathbf{X}$ and $\mathbf{Y}$ are generated from a static process.

$$\mathbf{x}_k = \mathbf{P}\mathbf{t}_k + \mathbf{e}_k$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k$$

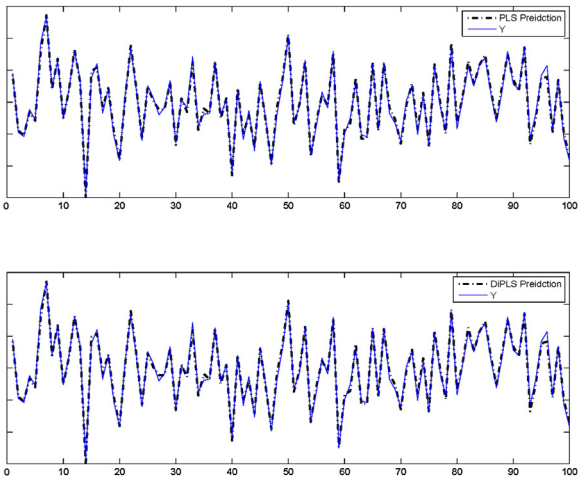


Fig. 1. Prediction result of DiPLS and PLS for Scenario 1.

Table 1
β_0, β_1 for each factor in Scenario1.

Value	Factor 1	Factor 2	Factor 3
β_0	0.9998	−0.9590	−0.9978
β_1	−0.217	0.2833	0.0670

$$\mathbf{P} = \begin{pmatrix} 0.5765 & 0.2856 & 0.1614 \\ 0.3660 & 0.0458 & 0.9060 \\ 0.5889 & 0.4645 & 0.9942 \\ 0.3572 & 0.3450 & 0.7396 \\ 0.4036 & 0.6851 & 0.2262 \end{pmatrix}, \quad \mathbf{C} = \begin{pmatrix} 0.7451 \\ 0.4928 \\ 0.7320 \\ 0.4738 \\ 0.5652 \end{pmatrix}^T$$

where $\mathbf{e}_k \in \mathbb{R}^5 \sim N([0, 0.5^2])$, and $\mathbf{v}_k \in \mathbb{R} \sim N([0, 0.5^2])$, $\mathbf{t}_k \in \mathbb{R}^3 \sim N([0, 2^2])$.

1000 data points are generated. The first 500 data points are used as training dataset to train the model, and the next 400 data points are used as the development dataset to select the parameter, and the last 100 data points are used as test dataset to evaluate the prediction result. The optimal parameters determined by cross validation is $s = 0$ and the number of components is 3. $s = 0$ indicates that DiPLS reduces to traditional PLS, which is consistent with the static model of inputs and outputs.

Fig. 1 shows the prediction results of DiPLS and PLS, from which we can see DiPLS gives the same result as PLS. This is consistent with our analysis. If we increase s to 1, the values of β_0, β_1, which are the weights of time lag 0 and 1, are listed in Table 1. We can tell from the table that $\beta_0^2 \gg \beta_1^2$ for each iteration, which implies the input data with lag 1 has little impact in the outer model building. DiPLS reduces to PLS even though excess time lags are included.

5.2. Scenario 2

X is generated from a dynamic process, while **Y** is generated from a static process.

$$\mathbf{t}_k = \mathbf{A}_1\mathbf{t}_{k-1} + \mathbf{A}_2\mathbf{t}_{k-2} + \mathbf{f}_k$$

$$\mathbf{x}_k = \mathbf{P}\mathbf{t}_k + \mathbf{e}_k$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k$$

$\mathbf{f}_k \in \mathbb{R}^3 \sim N([0, 0.5^2])$ where , P and C are the same as Scenario 1.

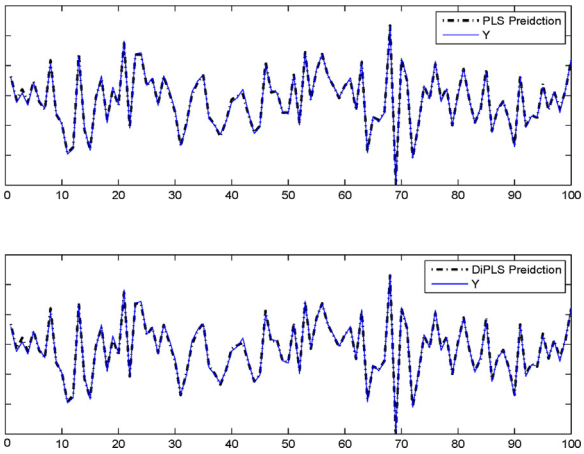


Fig. 2. Prediction result of DiPLS and PLS for Scenario 2.

Table 2
$\beta_0, \beta_1, \beta_3, \beta_4$ for each factor in Scenario 2.

Value	Factor 1	Factor 2	Factor 3
β_0	0.9940	0.9156	−0.9060
β_1	0.0821	0.3233	−0.1166
β_2	0.0691	0.0690	0.3554
β_3	0.0197	−0.2290	0.1980

$$\mathbf{A}_1 = \begin{pmatrix} 0.6767 & 0.5809 & 0.9315 \\ 1.2812 & -0.5343 & -1.6000 \\ -1.5083 & 0.9991 & 0.7529 \end{pmatrix}$$

$$\mathbf{A}_2 = \begin{pmatrix} 0.7155 & -0.0652 & 1.1192 \\ 1.1132 & -0.5371 & -0.1691 \\ -0.5571 & -1.0748 & 0.2330 \end{pmatrix}$$

We use the same number of data points for the training dataset, development dataset, and testing dataset for scenario 2 as scenario 1. The optimal parameters determined by cross validation is $s = 0$ and the number of factors is 3. The reason that $s = 0$ is because DiPLS considers the covariance between the input and the output, not the covariance of the input or the output. Therefore, s is only determined by the relationship between input and output. This example shows that, even though the input variables contain dynamics, the DiPLS model relation is static as long as the input–output relation is static. This example also indicates that the dynamics left in the input residuals should be further modeled dynamically if so interested. The prediction results of DiPLS and PLS are shown in Fig. 2. From the figure, we can see DiPLS gives the same result as PLS, which is consistent with the analysis. When s is increased to 3, the values of β_0, β_1, β_2, β_3 are listed in Table 2. We can tell from the table that β_0 is much larger than the square of other β's, therefore, the input data with no lag dominates the DiPLS result. DiPLS performs like a PLS model even though excessive lagged input data are included.

5.3. Scenario 3

X is generated from a static process, while **Y** is generated from a dynamic process.

$$\mathbf{x}_k = \mathbf{P}\mathbf{t}_k + \mathbf{e}_k$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{C}_2\mathbf{x}_{k-1} + \mathbf{v}_k$$

where **P**, **C** are the same as Scenario 1.

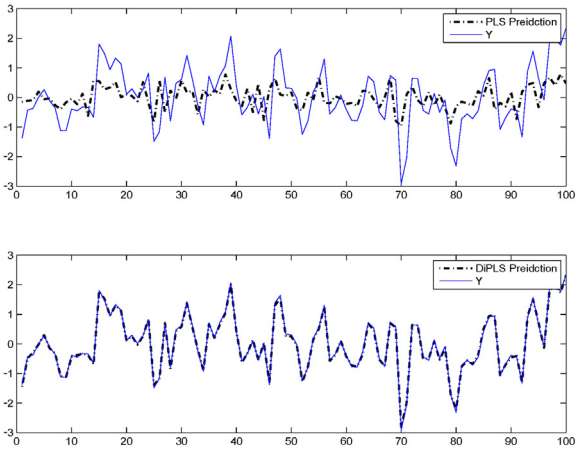


Fig. 3. Prediction result of DiPLS and PLS for Scenario 3.

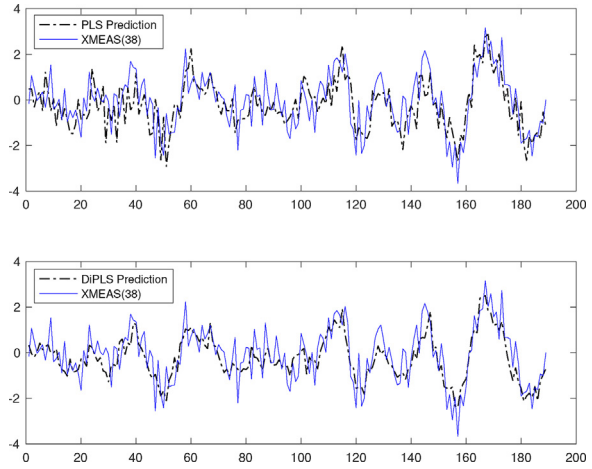


Fig. 4. Prediction result of DiPLS and PLS for TEP data.

Table 3
$\beta_0, \beta_1, \beta_3, \beta_4$ for each factor in Scenario 3.

Value	Factor 1	Factor 2	Factor 3	Factor 4
β_0	0.4251	0.1102	0.4528	0.9514
β_1	0.9009	−0.9850	0.8808	0.2832
β_2	−0.0707	0.1069	−0.1062	−0.0694
β_3	−0.0517	−0.0789	−0.0888	−0.0996

$$\mathbf{C}_2 = \begin{pmatrix} 1.9939 & 0.7728 & 1.0146 & 1.1563 & 1.2307 \end{pmatrix}$$

The same number of data points as in Scenario 1 are used as training dataset, development dataset, testing dataset. The optimal parameters determined by cross validation is $s = 1$ and the number of components is 4. Since $\mathbf{y}_k$ is related to both $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$, $s = 1$ is consistent with the dynamic structure between $\mathbf{X}$ and $\mathbf{Y}$. The prediction results of DiPLS and PLS are shown in Fig. 3. From Fig. 3, we can see when there are dynamics between inputs and outputs, DiPLS gives much better results than PLS. When s is increased to 3, the values of β_0, β_1, β_2, β_3 are listed in Table 3 for 4 latent factors.

We can see the squares of β_2 and β_3 are much smaller than β_0 and β_1 in general. Therefore, the input data with a lag of 2 and a lag of 3 will have little impact on the result. The DiPLS models built with $s = 1$ and $s = 3$ are similar.

5.4. Tennessee Eastman Process

The Tennessee Eastman Process (TEP) was developed to provide a realistic simulation of an industrial process for the evaluation of monitoring and control methods [21]. The process contains 12 manipulated variables and 41 measured variables. The measured variables contain 22 process variables sampled every 3 minutes, and 19 quality variables sampled with dead time and time delays. In this case study, 22 process variables XMEAS(1–22) and 11 manipulated variables XMV(1–11) are used as input, XMEAS(38), which is the calculated data from the analyzer, is used as output. The sampling rate for XMEAS(38) is 0.25 h and dead time is 0.25 h. 400 data points are used as the training dataset, 80 data points are used as the development dataset, and 960 data points are used as the testing dataset. Data are pre-shifted to compensate the 0.25 h delay. The optimal parameters determined by cross validation is $s = 5$, and the number of components is 3. The prediction results of DiPLS and PLS is shown in Fig. 4.

Note that for XMEAS(38), there is only one measured value every five data points due to the lower sampling rate, and the subsequent four values are artificial. Therefore, only 1/5 of the 960 data points are compared. Fig. 4 shows that DiPLS captures trends in the data better than PLS. Table 4 gives the mean squared error (MSE) of PLS and DiPLS predictions. We can see that DiPLS method gives about 40% improvement over the PLS method. Therefore, this case study shows that DiPLS performs better than PLS in TEP process modeling.

Table 4
MSE of PLS and DiPLS predictions on TEP data.

	PLS	DiPLS
MSE	0.9198	0.5503

6. Conclusions

In this article, a dynamic inner PLS modeling method is proposed for dynamic process and quality data modeling. The proposed method gives an explicit representation of the dynamic latent structures, by enforcing dynamic inner model structures when the outer models are derived. DiPLS reduces to traditional PLS and gives the same results as PLS if a particular latent factor has no dynamic correlations. This feature makes DiPLS a general modeling approach that can capture instantaneous as well as time-lagged input–output correlations. The dynamic inner models can also accommodate the case of time delays. Cross-validation is used in this paper to determine the optimal number of lags and the number of latent factors. Case studies on simulation data and Tennessee Eastman Process show that, when dynamic correlations are present in the data, it is more efficient to model the data with DiPLS. In addition, the DiPLS model does not directly augment the data matrix with a number of lags in order to capture dynamics in the data, making the outer model projections explicit and the compact number of latent factors easy to interpret. The use of DiPLS models for process monitoring and soft sensors will be studied in the future.

Acknowledgements

This work is supported by funds from the National Natural Science Foundation of China (61490704) and the Fundamental Disciplinary Research Program of the Shenzhen Committee on Science and Innovations (20160207, 20170155).

Appendix A. The basic DiPLS algorithm

1. Scale $\mathbf{X}$ and $\mathbf{Y}$ to zero-mean and unit-variance. Initialize $\boldsymbol{\beta}$ with $[1, 0, \ldots, 0]^T$, and $\mathbf{u}_s$ as some column of $\mathbf{Y}_s$.

2. Outer modeling. Iterate the following relations until convergence achieved.

$$\mathbf{w} = \sum_{i=0}^{s} \beta_i \mathbf{X}_{s-i}^T \mathbf{u}_s; \mathbf{w} := \mathbf{w} / \|\mathbf{w}\|$$

$$\mathbf{t} = \mathbf{X}\mathbf{w} \quad \text{Form} \quad \mathbf{t}_{s-i} \quad \text{from} \quad \mathbf{t} \quad \text{for } i = 0, 1, \ldots, s$$

$$\text{and} \quad \mathbf{T}_s = [\mathbf{t}_s \quad \mathbf{t}_{s-1} \quad \cdots \quad \mathbf{t}_0]$$

$$\mathbf{q} = \mathbf{Y}_s^T \sum_{i=0}^{s} \beta_i \mathbf{t}_{s-i}; \mathbf{q} := \mathbf{q} / \|\mathbf{q}\|$$

$$\mathbf{u}_s = \mathbf{Y}_s \mathbf{q}$$

$$\boldsymbol{\beta} = [\beta_0 \quad \beta_1 \cdots \beta_s] = [\mathbf{t}_s \quad \mathbf{t}_{s-1} \cdots \mathbf{t}_0]^T \mathbf{u}_s$$

$$\boldsymbol{\beta} := \boldsymbol{\beta} / \left\|\boldsymbol{\beta}\right\|$$

3. Inner modeling. Calculate $\boldsymbol{\beta}$ using (17) and

$$\hat{\mathbf{u}}_s = \mathbf{T}_s \hat{\boldsymbol{\beta}} = \mathbf{T}_s (\mathbf{T}_s^T \mathbf{T}_s)^{-1} \mathbf{T}_s \mathbf{u}_s$$

4. Deflation. Deflate $\mathbf{X}$ and $\mathbf{Y}$ as

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T; \quad \mathbf{p} = \mathbf{X}^T \mathbf{t} / \mathbf{t}^T \mathbf{t}$$

$$\mathbf{Y}_s := \mathbf{Y}_s - \hat{\mathbf{u}}_s \mathbf{q}^T$$

5. Repeat to Step 2 until enough latent variables are extracted.

References

[1] H. Hotelling, Relations between two sets of variates, Biometrika 28 (3/4) (1936) 321–377.
[2] H. Wold, Estimation of principal components and related models by iterative least squares, J. Multivar. Anal. (1966) 391–420.
[3] S. Wold, A. Ruhe, H. Wold, W. Dunn III, The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses, SIAM J. Sci. Stat. Comput. 5 (3) (1984) 735–743.
[4] H. Martens, T. Naes, Multivariate Calibration, John Wiley Sons, New York, 1989.
[5] P. Geladi, B.R. Kowalski, Partial least-squares regression: a tutorial, Anal. Chim. Acta 185 (1986) 1–17.
[6] A. Höskuldsson, PLS regression methods, J. Chemom. 2 (3) (1988) 211–228.
[7] G. Li, S.J. Qin, D. Zhou, Geometric properties of partial least squares for process monitoring, Automatica 46 (2010) 204–210.
[8] J.F. MacGregor, C. Jaeckle, C. Kiparissides, M. Koutoudi, Process monitoring and diagnosis by multiblock PLS methods, AIChE J. 40 (1994) 826–838.
[9] S.J. Qin, Y. Zheng, Quality-relevant and process-relevant fault monitoring with concurrent projection to latent structures, AIChE J. 59 (2) (2013) 496–504.
[10] N.L. Ricker, The use of biased least-squares estimators for parameters in discrete-time pulse-response models, Ind. Eng. Chem. Res. 27 (2) (1988) 343–350.
[11] S.J. Qin, T. McAvoy, Nonlinear FIR modeling via a neural net PLS approach, Comput. Chem. Eng. 20 (2) (1996) 147–159.
[12] M.H. Kaspar, W.H. Ray, Dynamic PLS modelling for process control, Chem. Eng. Sci. 48 (20) (1993) 3447–3461.
[13] S. Lakshminarayanan, S.L. Shah, K. Nandakumar, Modeling and control of multivariable processes: dynamic PLS approach, AIChE J. 43 (9) (1997) 2307–2322.
[14] G. Li, B. Liu, S.J. Qin, D. Zhou, Quality relevant data-driven modeling and monitoring of multivariate dynamic processes: the dynamic T-PLS approach, IEEE Trans. Neural Netw. 22 (12) (2011) 2262–2271.
[15] Y. Dong, S.J. Qin, Dynamic-inner partial least squares for dynamic data modeling, IFAC-PapersOnLine 48 (8) (2015) 117–122.
[16] H. Martens, T. Naes, Multivariate calibration by data compression, in: P. Williams, K. Norris (Eds.), Near-Infrared Technology in the Agricultural and Food Industries, American Association of Cereal Chemists, St. Paul, MN, 1987, pp. 57–87.
[17] I.S. Helland, On the structure of partial least squares regression, Commun. Stat. Simul. Comput. 17 (1988) 581–607.
[18] A. Höskuldsson, PLS regression methods, J. Chemom. 2 (3) (1988) 211–228.
[19] S.J. Qin, Recursive PLS algorithms for adaptive data modeling, Comput. Chem. Eng. 22 (4–5) (1998) 503–514.
[20] M.H. Kaspar, W.H. Ray, Partial least squares modelling as successive singular value decompositions, Comput. Chem. Eng. 17 (10) (1993) 985–989.
[21] J.J. Downs, E.F. Vogel, A plant-wide industrial process control problem, Comput. Chem. Eng. 17 (3) (1993) 245–255.