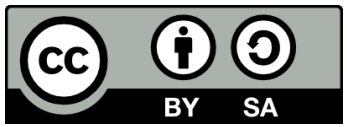
Latent Variable Methods Course

Learning from data

Instructor: Kevin Dunn
kevin.dunn@connectmv.com
http://connectmv.com

Revision: 268:adfd compiled on 15-12-2011

Copyright, sharing, and attribution notice



This license allows you:

- **to share** - to copy, distribute and transmit the work
- **to adapt** - but you must distribute the new result under the same or similar license to this one
- **commercialize** - you *are allowed* to create commercial applications based on this work
- **attribution** - you must attribute the work as follows:
  - "Portions of this work are the copyright of ConnectMV", *or*
  - "This work is the copyright of ConnectMV"

We appreciate:

- if you let us know about **any errors** in the slides
- **any suggestions to improve the notes**
- telling us if you use the slides, especially commercially, so we can inform you of major updates
- emailing us to ask about different licensing terms

All of the above can be done by writing us at

`courses@connectmv.com`

If reporting errors/updates, please quote the current revision number: `268:adfd`

Objectives for this class

1. Combine and learn from a variety of data sources
2. Track variation during a batch and how it affects product quality

We will

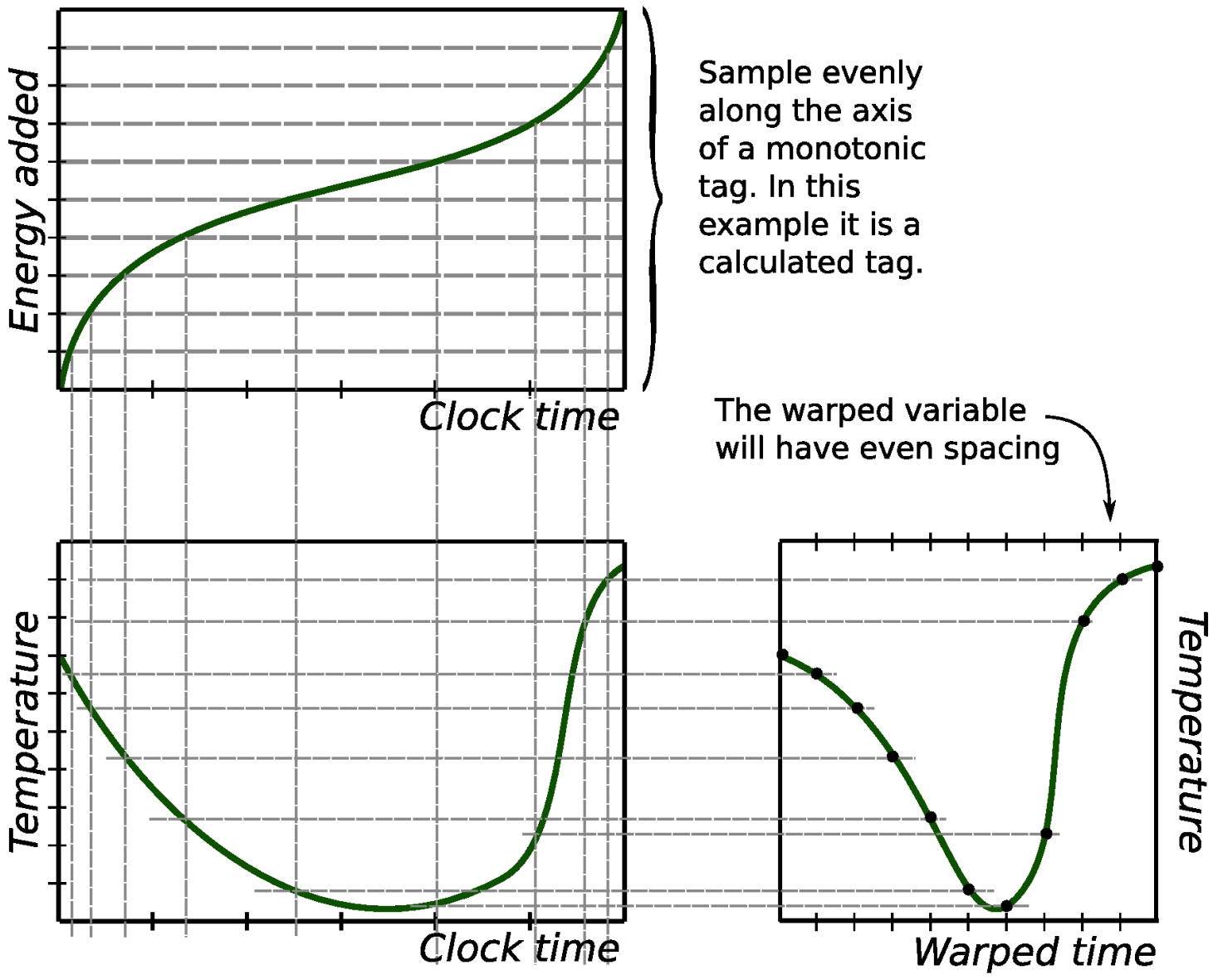
- recap the alignment concept that was badly explained last class
- introduce multiblock methods
- come back to batch monitoring
- end off with a case study that combines all these concepts

Recap: Alignment with an indicator variable

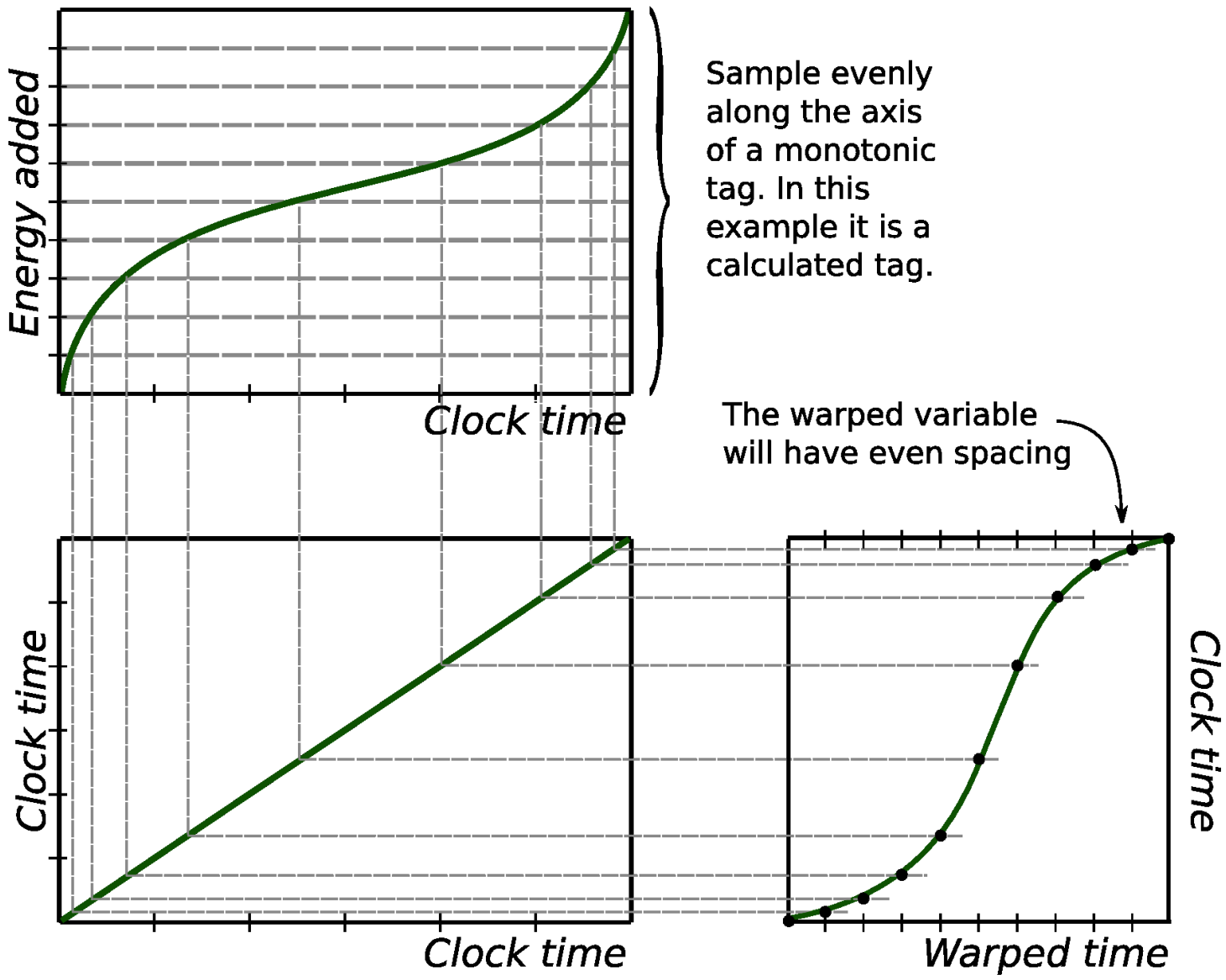
Read this slides in conjunction with the figures on the next 2 slides

- Choose a monotonic indicator variable to align against. For example:
  - reaction completion
  - a calculated variable
  - see other examples from last class
- Sample evenly along the y-axis of this tag
- Project across and down this monotonic tag onto all other tags, e.g. the temperature tag
- Resample these other tags at the new time points
- Notice how the tag has been time-warped
- We can also resample the "clock time" variable:
  - this creates a new "trajectory" called warped time
  - there is a warped time trajectory for each batch
  - include this in the unfolded $\mathbf{X}$ matrix as a new tag

Recap: Alignment with an indicator variable



Alignment recap



Multiblock methods

The main concept

Divide your variables into blocks to get

- better model interpretation
- easier monitoring and improved fault detection

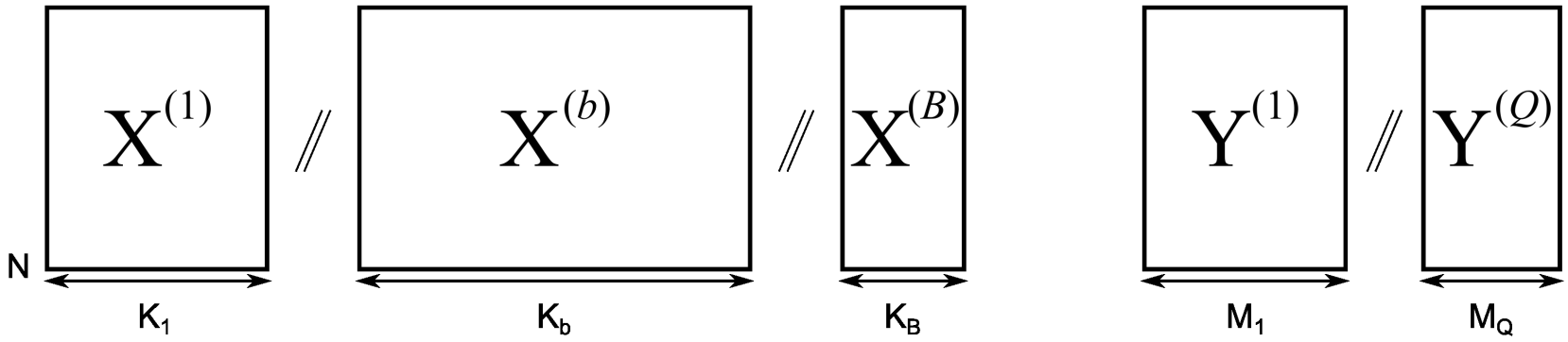
Why do this?: we'd like to understand the relationships between several groups of possibly related datasets

Sometimes called "data fusion".

References

- Original concept: Wold *et al.*, 1987 conference paper
- Improved fault detection: MacGregor *et al.*, AIChE Journal
- Equivalence of MBPCA and PBPLS to PCA and PLS (very important paper): Westerhuis, Kourti and MacGregor
- Process monitoring example with MB methods: Qin *et al.*
- Good overview of all multiblock methods: Smilde, Westerhuis and de Jong, 2003

Notation



- Multiple $\mathbf{X}$ and $\mathbf{Y}$ blocks are available
- There is only one consistent dimension: N = observations
- We will only consider the case of one $\mathbf{Y}$ block ($M_1 = M$)
  - $\mathbf{Y}$ will contain the usual quality variables
- We can have in $\mathbf{X}^{(b)}$ for example:
  - raw material properties (e.g. one block per material)
  - NIR or UV-VIS spectra from each observation
  - Unfolded batch data
  - Measurements from each unit operation

Key point: you can have duplicated variables between blocks

Terminology and concepts

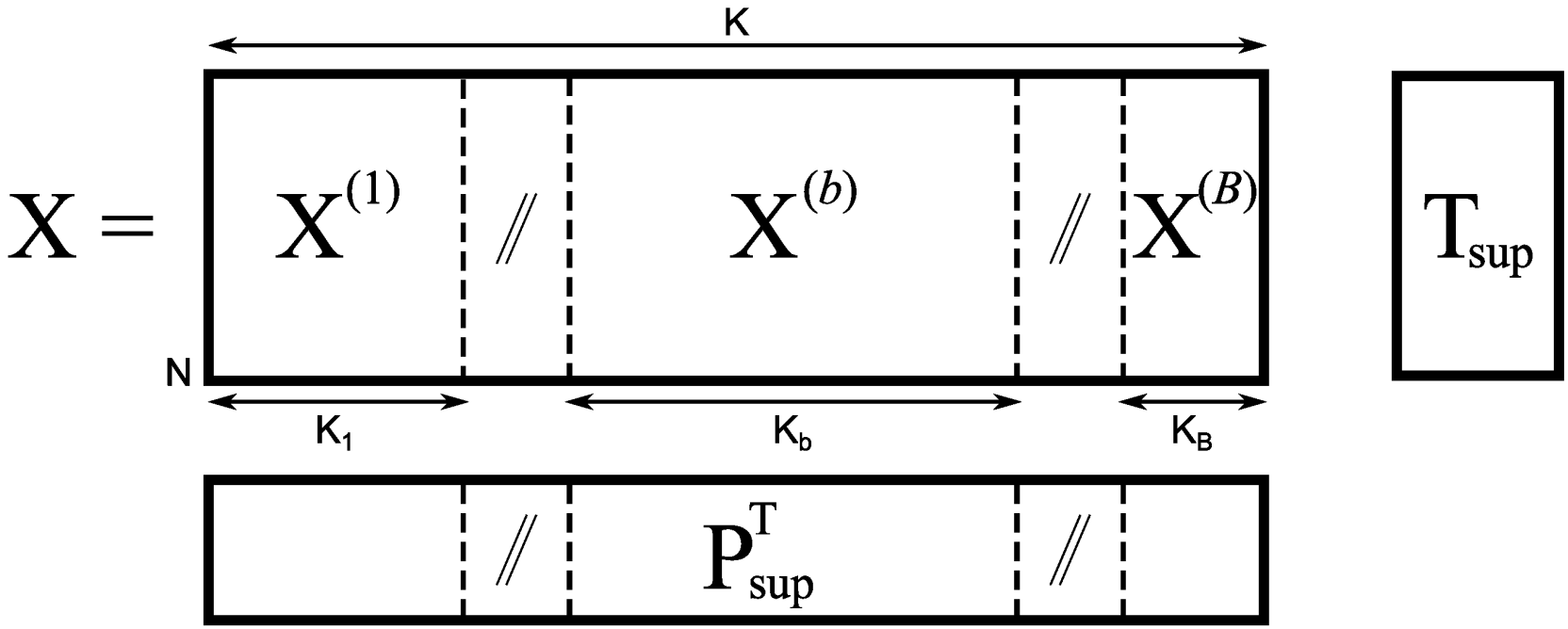
- Only have **X** blocks: multiblock PCA
- Add one or more **Y** blocks: then it becomes multiblock PLS

- Each block has: scores, loadings, SPE, T^2, weights, VIP, R^2

- We also have a "super-level" or "super-model" that summarizes the blocks

SUM-PCA approach

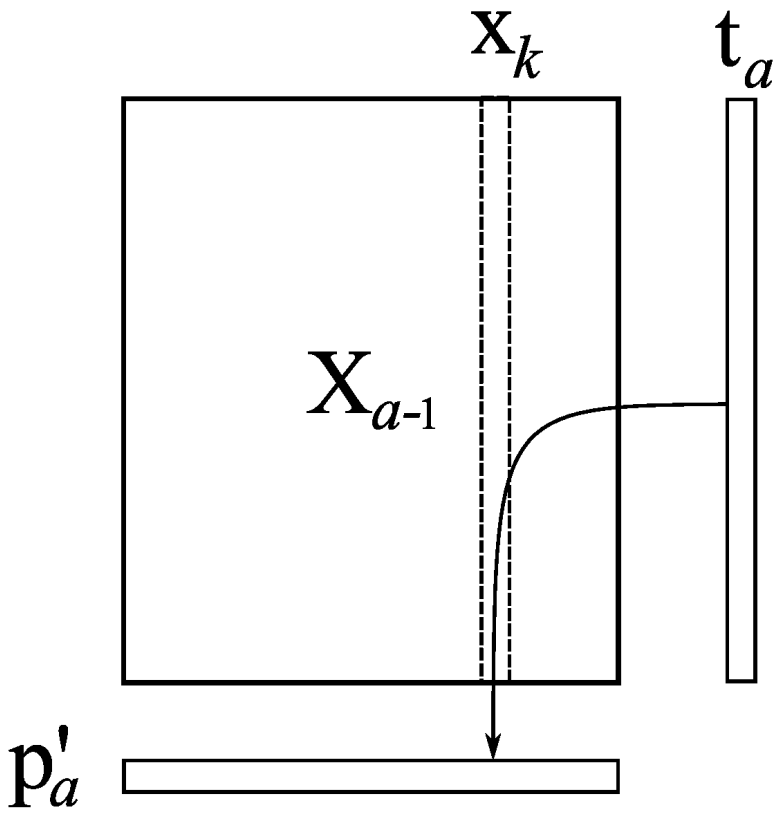
Crude approach: push all blocks together and build PCA model.



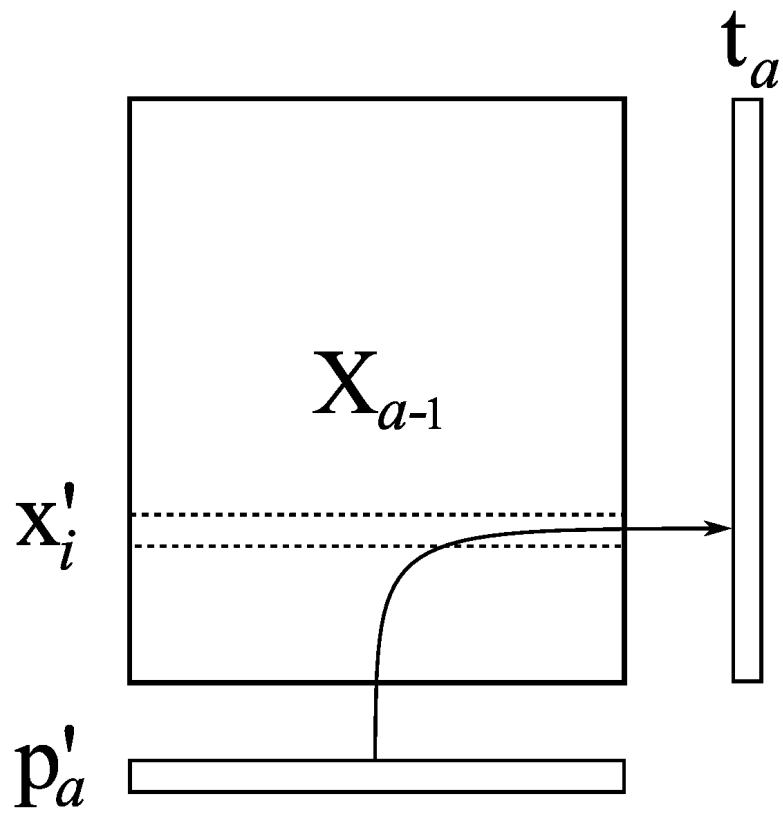
- Investigate loadings, R^2, *etc* separately for each block
- Block loadings, $P^{(b)}$, will not be orthogonal
- The super scores (the usual PCA scores) simply explain variation for entire **X**
- No guarantee that each block will contribute to superscores

NIPALS review

Before we proceed, let's recap the NIPALS algorithm for PCA



Loadings are regression coefficients (slopes) when regressing columns in $\mathbf{X}$ onto $\mathbf{t}_a$



Scores are regression coefficients (slopes) when regressing rows in $\mathbf{X}$ onto $\mathbf{p}_a$

Consensus PCA (CPCA)

Left blank for you to draw the diagram

Consensus PCA steps

1. Let $t_a^{(s)}$ be any column from any block
2. Regress column from $\mathbf{X}_a^{(b)}$ onto $\mathbf{t}_a^{(s)}$ to obtain block loadings

$$\mathbf{p}_a^{(b)} = \mathbf{X}^{(b)T}\mathbf{t}_a^{(s)} / \mathbf{t}_a^{(s)T}\mathbf{t}_a^{(s)}$$

3. Normalize $\mathbf{p}_a^{(b)}$ to unit length
4. Calculate block's score: $\mathbf{t}_a^{(b)} = \mathbf{X}^{(b)}\mathbf{p}_a^{(b)} \cdot \frac{1}{\sqrt{K_b}}$
   - weight $\sqrt{K_b}$ prevents blocks with many terms (variables) in the above linear combination from creating large score values, $\mathbf{t}_a^{(b)}$
5. Assemble block scores: $\mathbf{T}_a^{[s]} = \left[\mathbf{t}_a^{(1)} \ \ldots \ \mathbf{t}_a^{(b)} \ \ldots \ \mathbf{t}_a^{(B)}\right]$

Consensus PCA steps

6. Regress columns in $\mathbf{T}_a^{[s]}$ onto the superscore, $\mathbf{t}_a^{(s)}$ to calculate the super-level's loading:

$$\begin{aligned} \mathbf{p}_a^{[s]} &= \mathbf{T}_a^{[s]T}\mathbf{t}_a^{(s)} / \left(\mathbf{t}_a^{(s)T}\mathbf{t}_a^{(s)}\right) \\ & \quad (B \times N)(N \times 1) \end{aligned}$$

7. Normalize $\mathbf{p}_a^{[s]}$ to unit length
8. Regress rows in $\mathbf{T}_a^{[s]}$ onto $\mathbf{p}_a^{[s]}$ to get the super-scores $\mathbf{t}_a^{(s)}$:

$$\begin{aligned} \mathbf{t}_a^{(s)} &= \mathbf{T}_a^{[s]}\mathbf{p}_a^{[s]} / \left(\mathbf{p}_a^{[s]T}\mathbf{p}_a^{[s]}\right) \\ & \quad (N \times B)(B \times 1) \end{aligned}$$

denominator is usually = 1.0

9. Not converged? return back to step 2.
10. Converged? deflate each block *with the superscore*

$$\mathbf{X}_a^{(b)} = \mathbf{X}_a^{(b)} - \mathbf{t}_a^{(s)}\mathbf{p}_a^{(b)T}$$

Consensus PCA (CPCA)

- $\mathbf{t}_a^{(s)}\mathbf{p}_a^{(b)}$ = block prediction from the *superscore*, $\mathbf{t}_a^{(s)}$, not the block's score
- $\mathbf{t}_a^{(s)}$ was calculated from the assembled scores, $\mathbf{T}_a^{[s]}$
- $\mathbf{t}_a^{(s)}$ is just a weighted sum of these block scores (step 8): called the *consensus score*
- **Each entry in the superloading shows how much of block b is used in the consensus score**
- If a block behaves differently from the others, then its entry in $\mathbf{p}_a^{(s)}$ will be small
- Deflation by $\mathbf{t}_a^{(s)}$ removes the superscore information, not the block-score information.
  - We get non-orthogonal block scores, but orthogonal superscores

Computational simplification

Westerhuis, Kourti and MacGregor (1998) showed we don't need to calculate CPCA as just described.

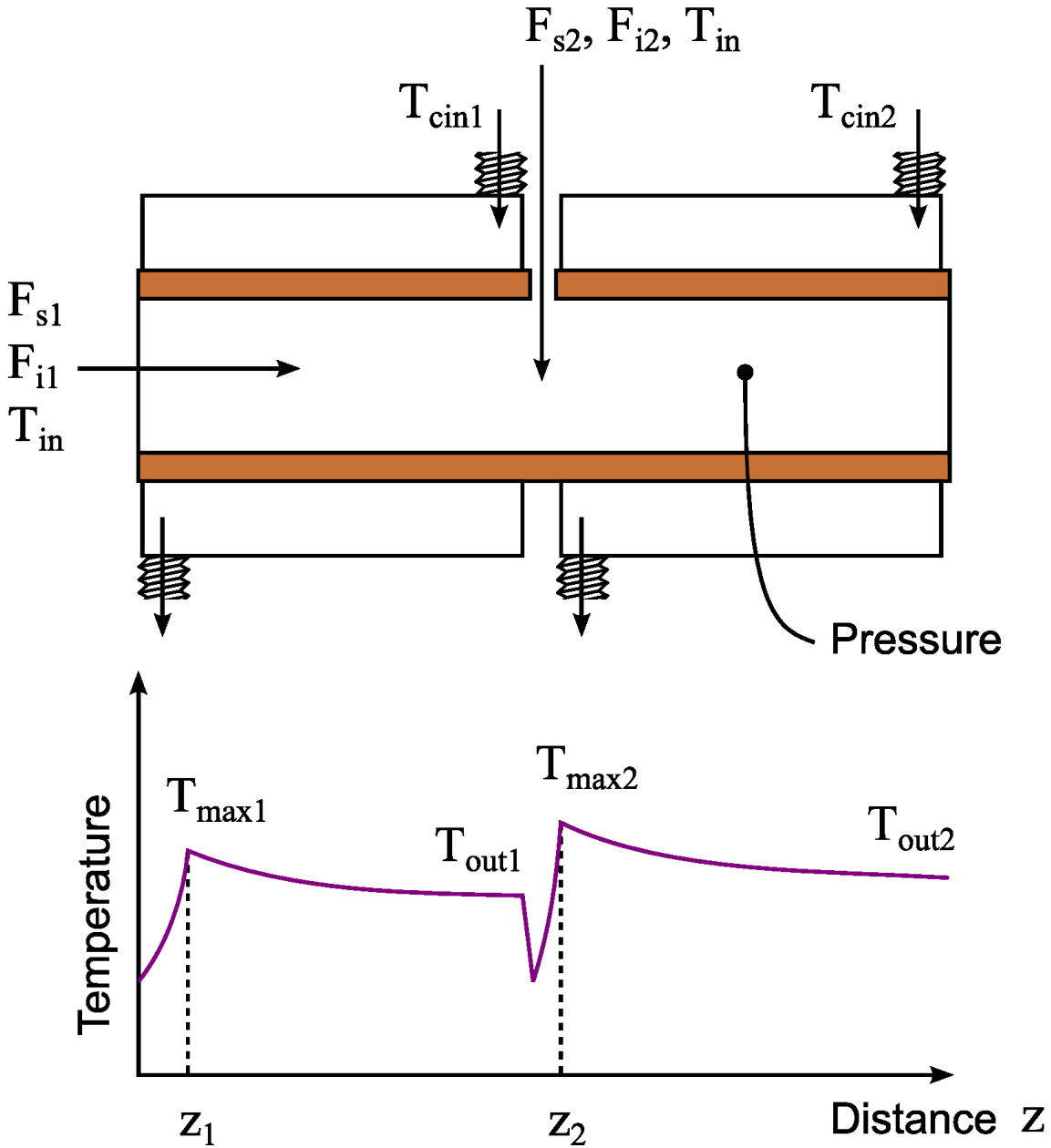
Much easier approach:

- Preprocess the data from each block as normal
- Post divide each block by $\sqrt{K_b}$ and assemble:

$$\mathbf{X} = \left[\frac{\mathbf{X}^{(1)}}{\sqrt{K_1}}, \frac{\mathbf{X}^{(2)}}{\sqrt{K_2}}, \ldots, \frac{\mathbf{X}^{(B)}}{\sqrt{K_B}} \right]$$

  - Same idea as block-scaling (covered earlier in the course)
- Calculate PCA in the usual way on $\mathbf{X}$ to obtain:
  - scores will be identical to CPCA super scores, $\mathbf{t}_1^{(s)}, \mathbf{t}_2^{(s)}, \ldots, \mathbf{t}_A^{(s)}$
  - then follow steps 2, 3, 4, 5 and 6 from above
  - results will be identical to the full approach

In-class example



In-class example

Load the LDPE data set and create a 2-block PCA model:

1. "Zone 1" block
   - Inlet temperature
   - Pressure
   - All other variables ending in "1"
2. 'Zone 2" block
   - Inlet temperature
   - Pressure
   - All other variables ending in "2"

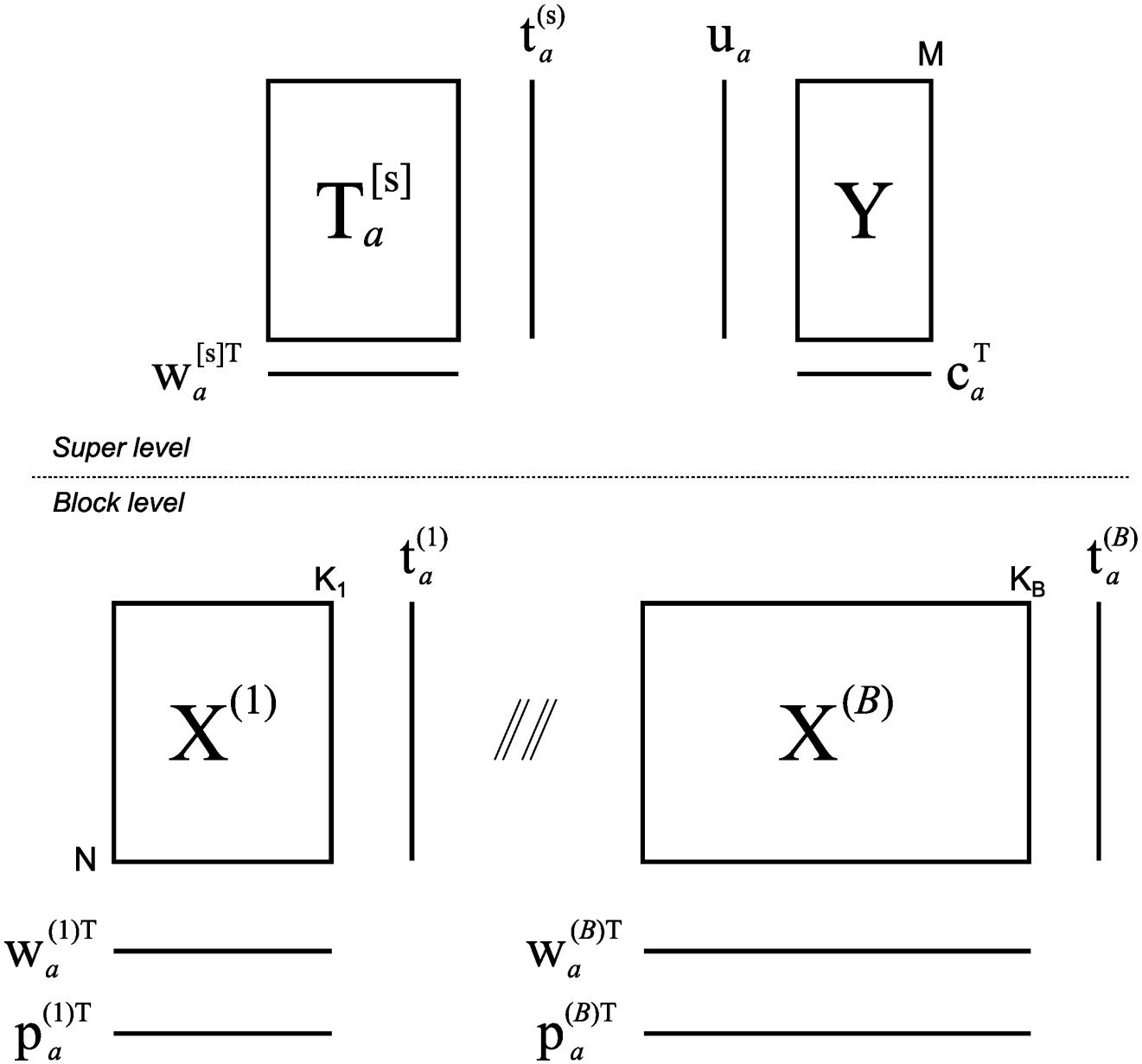
Build a multiblock CPCA model, using cross-validation to determine A:

- examine scores for each block, and the superblock
- examine the loadings bi-plot for each block
- example the SPE time-series for each block and the superblock

What can go into each block?

- Raw material properties
  - have one block per raw material
- NIR or UV-VIS spectra ($K^{(b)}$ will be large)
- Unfolded batch trajectories
- Features extracted from batch trajectories
- Data from sequential operations
  - data from each step/operation/phase in its own block
  - could be hard to ensure consistency from row to row
- Large PCA and PLS models. E.g. petroleum refinery
  - distillation column's data
  - fractionator's data
  - FCCU data
- Judges: one block per judge
  - each judge block contains the same columns (attributes)
- Lagged variables
  - e.g. a variable and all its lag per block
  - or, all variables at a particular lag in each block

Multiblock PLS (MBPLS) concept



MBPLS concept

We won't go through the detailed arrow pushing diagrams:

1. Start with an initial guess for $\mathbf{u}_a$
2. Perform a CPCA cycle through all the $\mathbf{X}^{(b)}$ blocks and this $\mathbf{u}_a$
3. Assemble each block's scores, $\mathbf{t}_a^{(b)}$, into $\mathbf{T}_a^{[s]} = \left[\mathbf{t}_a^{(1)} \ \ldots \ \mathbf{t}_a^{(B)}\right]$
4. Do a single NIPALS cycle for PLS between $\mathbf{T}_a^{[s]}$ and $\mathbf{Y}$ for
   - super scores, $\mathbf{t}_a^{(s)}$
   - super weights, $\mathbf{w}_a^{[s]}$, a $B \times 1$ vector
   - $\mathbf{Y}$-space loadings: $\mathbf{c}_a$, a $M \times 1$ vector
   - $\mathbf{Y}$-space scores: $\mathbf{u}_a$
5. Repeat from step 2 until convergence for the a^{th} component
6. Then deflate ... (next slide)

Once all components calculated, predict $\hat{\mathbf{Y}} = \mathbf{t}_1^{(s)}\mathbf{c}_1 + \ \ldots \ + \mathbf{t}_A^{(s)}\mathbf{c}_A$

MBPLS deflation

There are 2 choices to deflate each block:

1. using the block's own score and loading

$$\mathbf{X}^{(b)} = \mathbf{X}^{(b)} - \mathbf{t}_a^{(b)} \mathbf{p}_a^{(b)T}$$

   - induces orthogonal scores and loadings at the block level
   - super scores, $\mathbf{t}_a^{(s)}$, will not be orthogonal

2. using the super score and the block's loading

$$\mathbf{X}^{(b)} = \mathbf{X}^{(b)} - \mathbf{t}_a^{(s)} \mathbf{p}_a^{(b)T}$$

   - block level scores and loadings not orthogonal
   - super scores are orthogonal

Using the MBPLS model in the future

1. Center and scale new data, $\mathbf{x}_{\text{new}}^{(b)}$, according to each block's preprocessing
2. Calculate block score $= t_{a,\text{new}}^{(b)} = \mathbf{x}_{\text{new}}^{(b)T} \mathbf{w}_a^{(b)} \cdot \frac{1}{K_b}$
3. Assemble the block score vector: $\mathbf{t}_{a,\text{new}}^{[s]} = \left[t_{a,\text{new}}^{(1)}, \ \ldots \ , t_{a,\text{new}}^{(B)} \right]$
4. Calculate the super score: $t_{a,\text{new}}^{(s)} = \mathbf{t}_{a,\text{new}}^{[s]} \mathbf{w}_a^{[s]}$
5. Deflate each block: $\mathbf{x}_{\text{new}}^{(b)} = \mathbf{x}_{\text{new}}^{(b)} - t_{a,\text{new}}^{(s)} \mathbf{p}_a^{(b)}$ using *superscore*
6. Repeat from step 2 for all components $a = 1, 2, \ldots A$
7. Predict: $\hat{\mathbf{y}}_{\text{new}} = t_{1,\text{new}}^{(s)} \mathbf{c}_1 + \ \ldots \ + t_{A,\text{new}}^{(s)} \mathbf{c}_A$

Also calculate SPE and T^2 for each block, and for the super level

Which deflation to use for MBPLS

Method 1

- Removes all variation in $\mathbf{t}_a^{(b)}$ from $\mathbf{X}^{(b)}$
- Also, $\mathbf{t}_a^{(b)} w_{a,b}^{[s]}$ is the portion from block b used to predict $\mathbf{Y}$
- If $\mathbf{w}_{a,b}^{[s]} \approx 0$ (small super weight for block b for component a), then $\mathbf{t}_a^{(b)}$ has not predictive ability for $\mathbf{Y}$
- Once removed (deflated), it cannot be used in subsequent components
- One advantage though: the block scores tend to be more directly related to $\mathbf{Y}$

Method 2

- Removes from $\mathbf{X}^{(b)}$ the variation in $\mathbf{t}_a^{(s)}$
- Variation in $\mathbf{t}_a^{(s)}$ is used to explain $\mathbf{Y}$

Actual calculation for MBPLS

Westerhuis, Kourti and MacGregor (1998) showed we don't need to calculate MBPLS as just described.

Easier approach:

- Preprocess the data from each block as normal
- Post divide each block by $\sqrt{K_b}$ and assemble:

$$\mathbf{X} = \left[\frac{\mathbf{X}^{(1)}}{\sqrt{K_1}}, \frac{\mathbf{X}^{(2)}}{\sqrt{K_2}}, \ldots, \frac{\mathbf{X}^{(B)}}{\sqrt{K_B}} \right]$$

- Calculate PLS in the usual way on $\mathbf{X}$ and $\mathbf{Y}$ to obtain:
  - scores are identical to MBPLS super scores, $\mathbf{t}_1^{(s)}, \mathbf{t}_2^{(s)}, \ldots, \mathbf{t}_A^{(s)}$
  - back-calculate the block weights, loadings and scores
  - then calculate the block SPE and T^2
  - also calculate the super weights
  - results will be identical to the full approach

Is all this complexity worth it?

Given the above derivations (especially if this is the first time seeing it), one can rightly ask whether this is worth it.

- Consensus PCA can be calculated from ordinary PCA
- Multiblock PLS can be calculated from ordinary PLS
- This implies the predictive performance will be identical

Advantages are:

- better interpretation
- separate monitoring and fault detection for each block, since
  - each block has its own SPE, T^2, weights, loadings, VIP, R^2
  - super level: has SPE, T^2, weights, VIP, R^2

Better interpretation from multiblock models

FMC features example

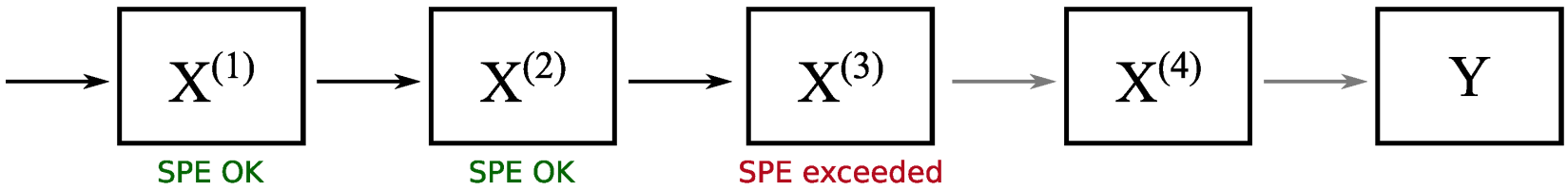
Better monitoring from multiblock models

The problem: contribution plots from a single PCA or PLS model often identify too many variables

- Complex systems with sub parts are split into blocks
- Even single units can be subdivided
  - film extruder: melt zone, extrusion zone, casting, roll-up
  - distillation column: bottom, feed and top trays, reboiler, condenser
- Monitor the SPE's from each block, and the super block's SPE and T^2
- When SPE limit is exceeded, only show contributions for the block where the limit is exceeded

Sequential monitoring with multiblock models

Many processes consist of sequential steps. *Example*: 4 sequential operations are used to produce the final product; lab values are measured at the end. Two weeks from start to end.



- Use data from each stage to calculate block's SPE and T^2
- Proceed to the next stage if they are below the limit
- One also obtains a prediction of $\mathbf{Y}$ after each stage
  - the prediction accuracy should improve after each stage
- If limit exceeded: use contributions, and judge the risk/cost of continuing
  - previous bad observations will help determine and understand this risk

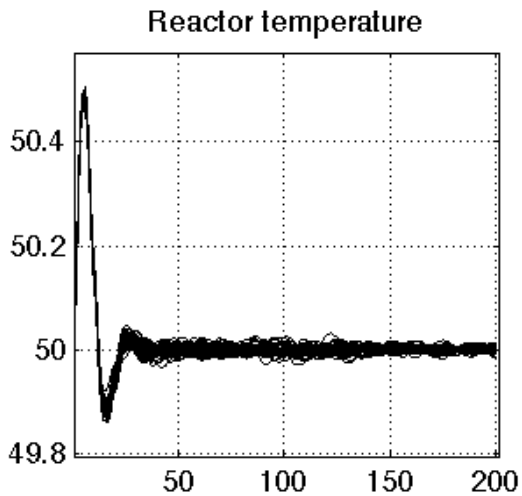
Batch PLS example: SBR

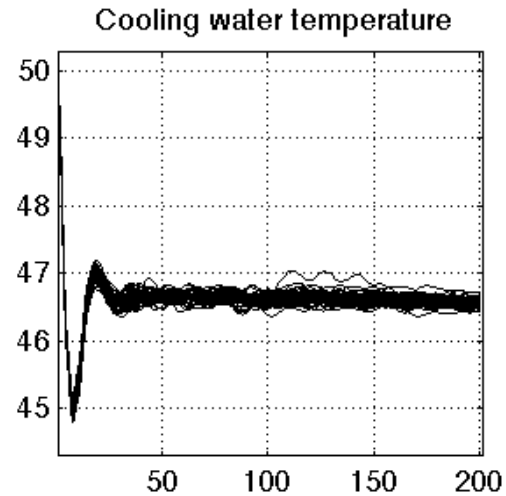
- Simulated data from *first principles* mechanistic model for styrene butadiene rubber[1]
- Simulations are useful to make sure models identify what we expect
- Simulation contained mostly "normal operating conditions"
  - 2 problematic batches were simulated
  - the same fault, but starting at different times
- **Y**-space quality variables:
  1. Composition
  2. Particle size
  3. Branching
  4. Cross linking
  5. Polydispersity

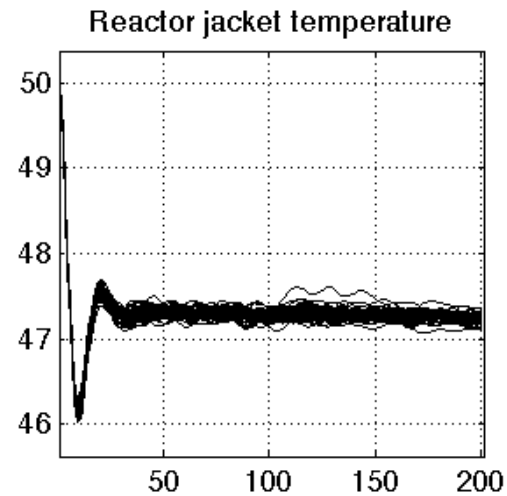
[1] More details can be found in Paul Nomikos' PhD thesis

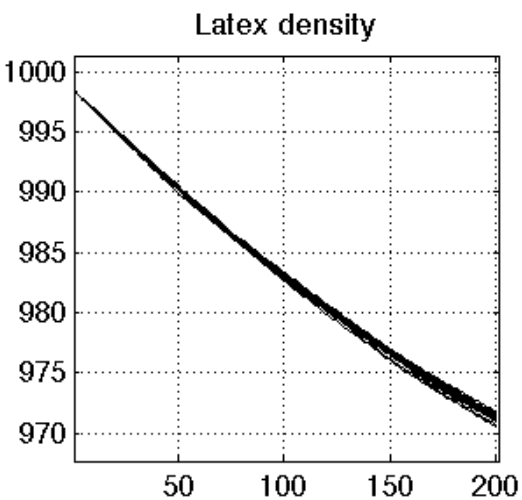
SBR: raw data

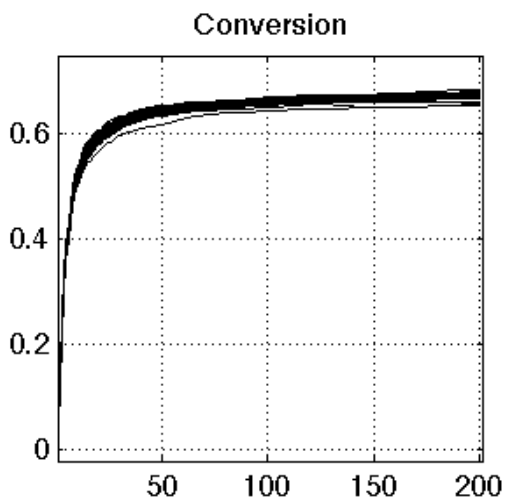
- Batches data: $N = 53$; Tags: $K = 6$; Time steps: $J = 200$

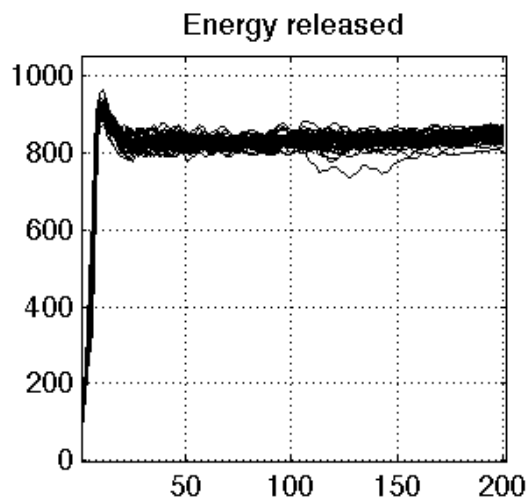












SBR: build model

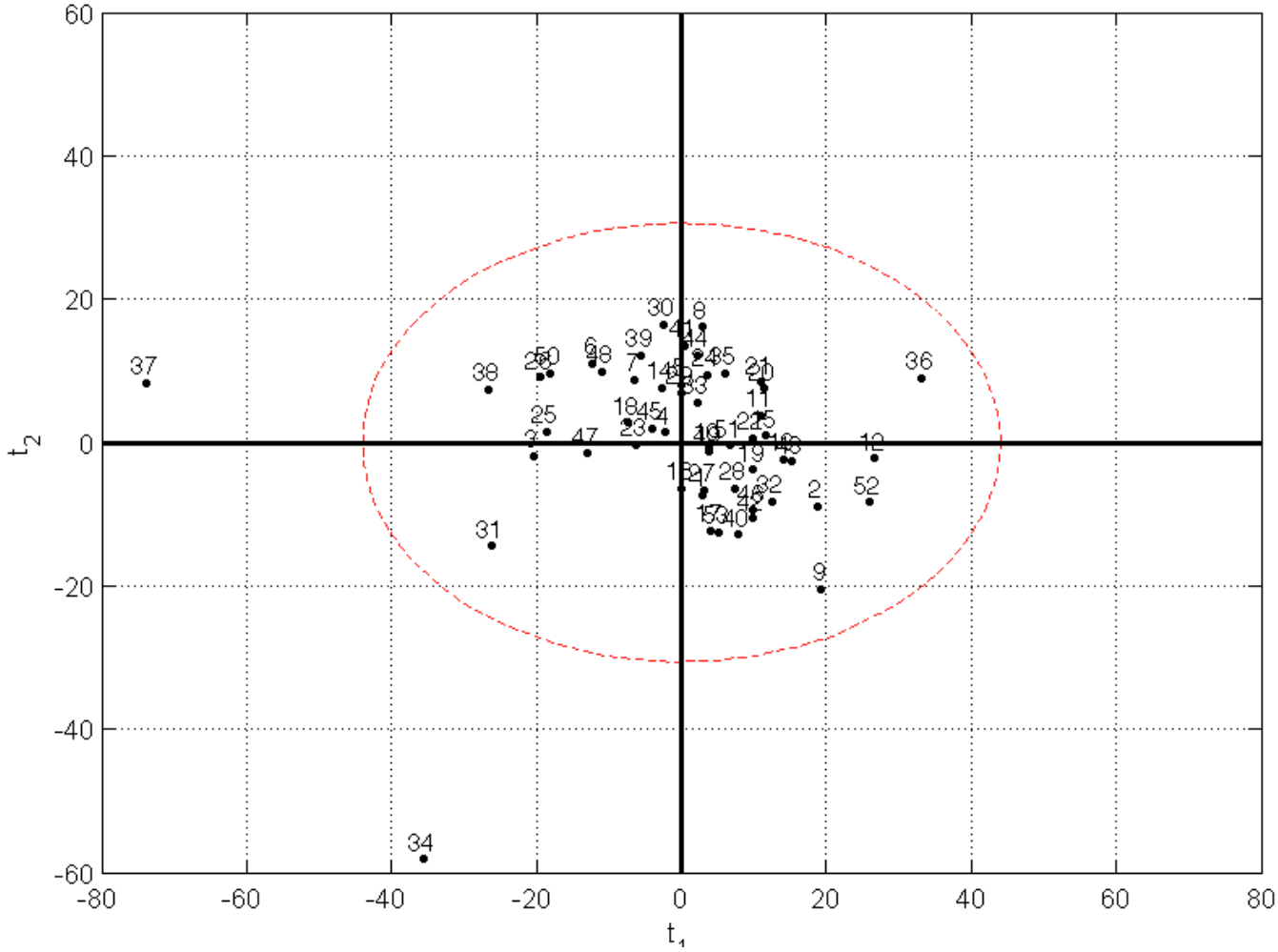
Approach:

- Normally I would start with a PCA on the **X**-space trajectories to understand the trajectory relationships
- Then a PCA on the **Y**-space quality variables to see if there are unusual batches
- In this data set: both these PCA models give the same interpretation as PLS
- So we only show the PLS results here.

PLS results:

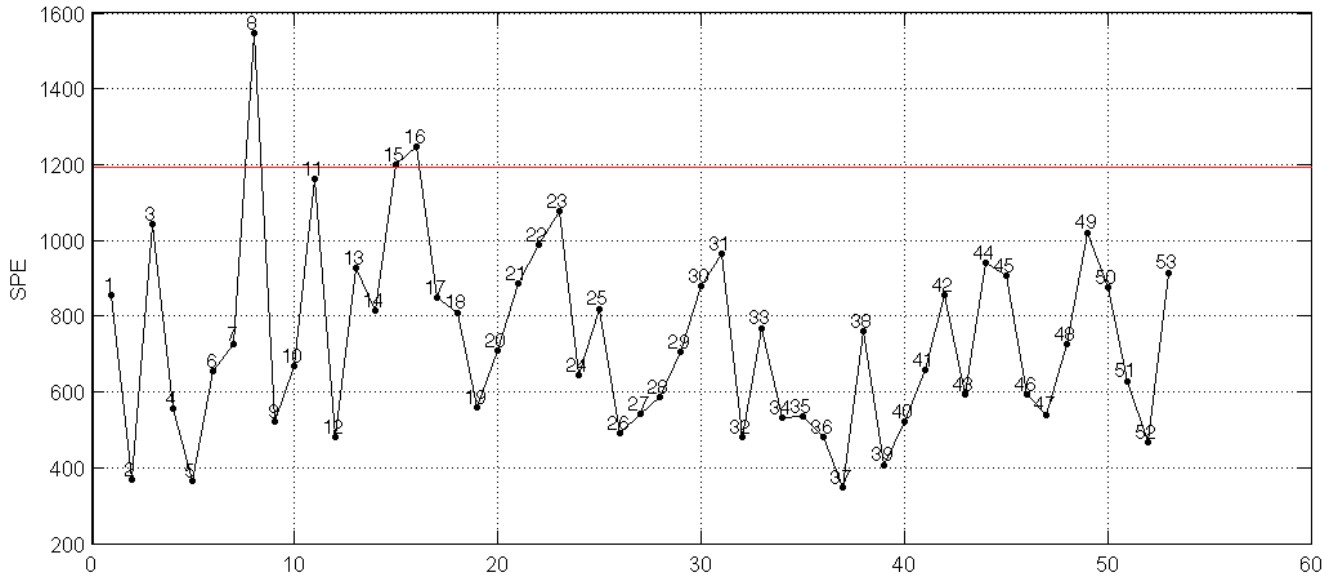
- Start with 2 to 3 components: *just to see what's going on*
- $R^2_{X,1} = 24.5\%$ and $R^2_{X,2} = 12.7\%$
- $R^2_{Y,1} = 65.3\%$ and $R^2_{X,2} = 6.9\%$
- Next: scores, weights, SPE, T^2 ... all the usual PLS tools

SBR: PLS score plot



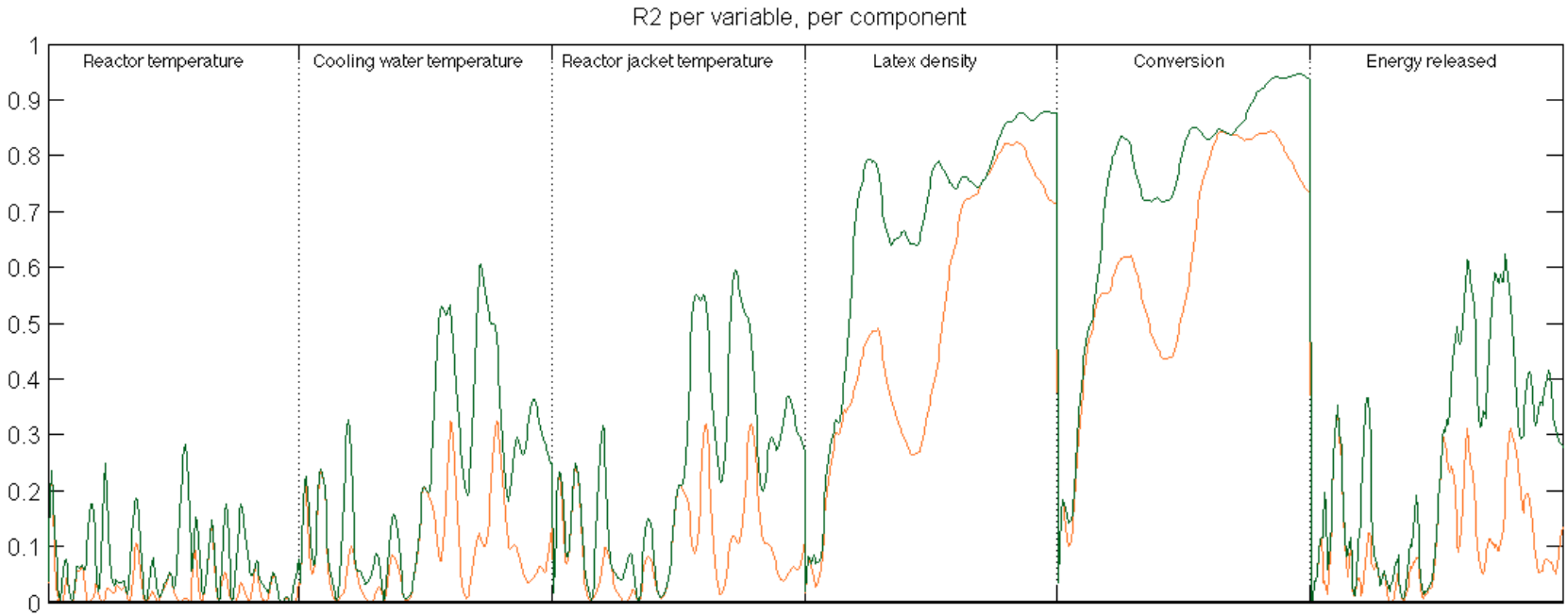
- Batches 34 and 37 were in fact the unsuccessful batches! This shows promise.

SBR: check SPE



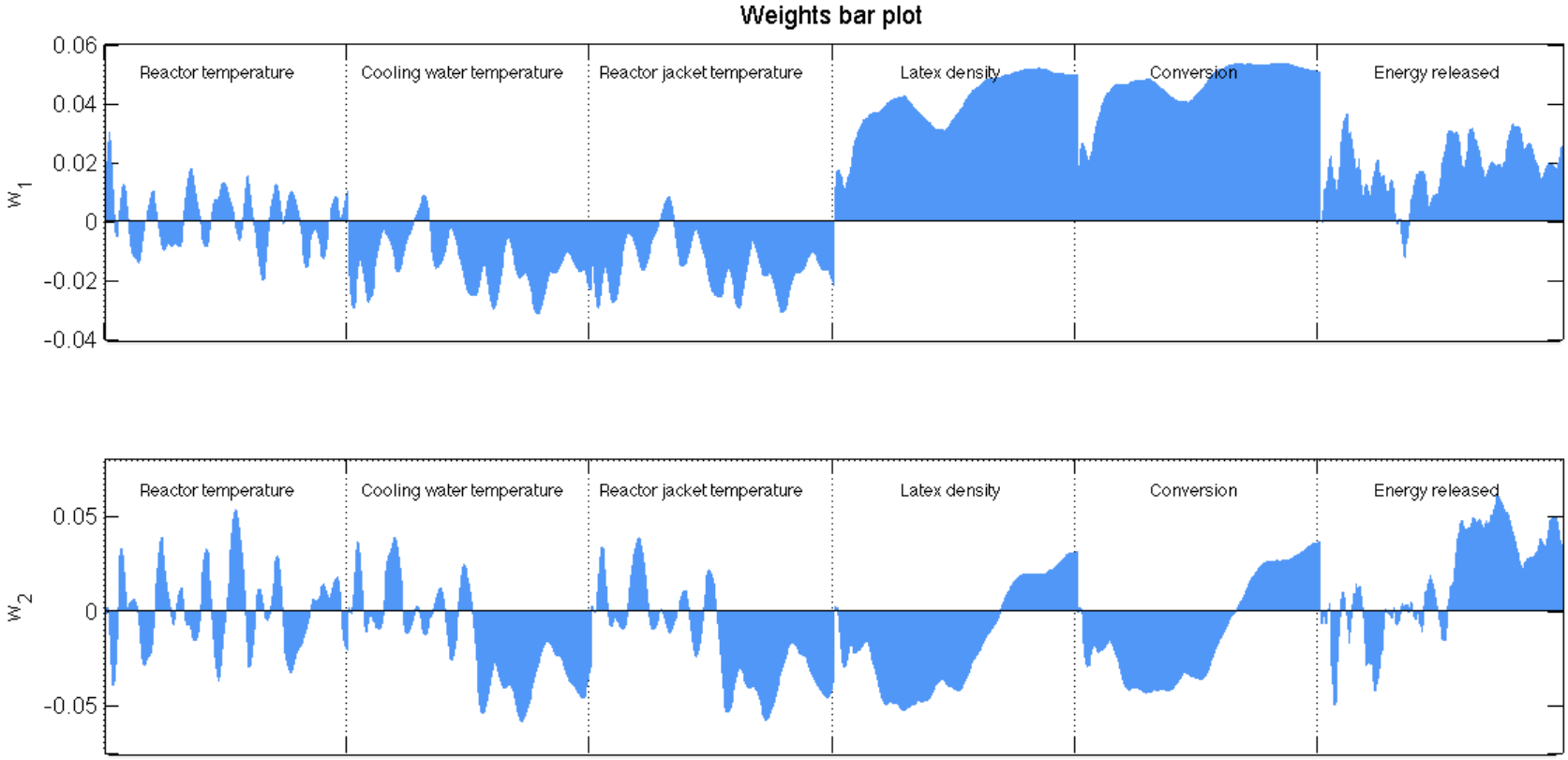
- No problems picked up. This is the overall SPE, using data from the *entire* batch.

SBR: understand R^2 breakdown in the **X**-space



- LV1 and 2: latex density and conversion dominate the model
- R^2 is low at start because all batches are similar initially
  - after centering and scaling there is just noise at the start.

SBR: PLS weights

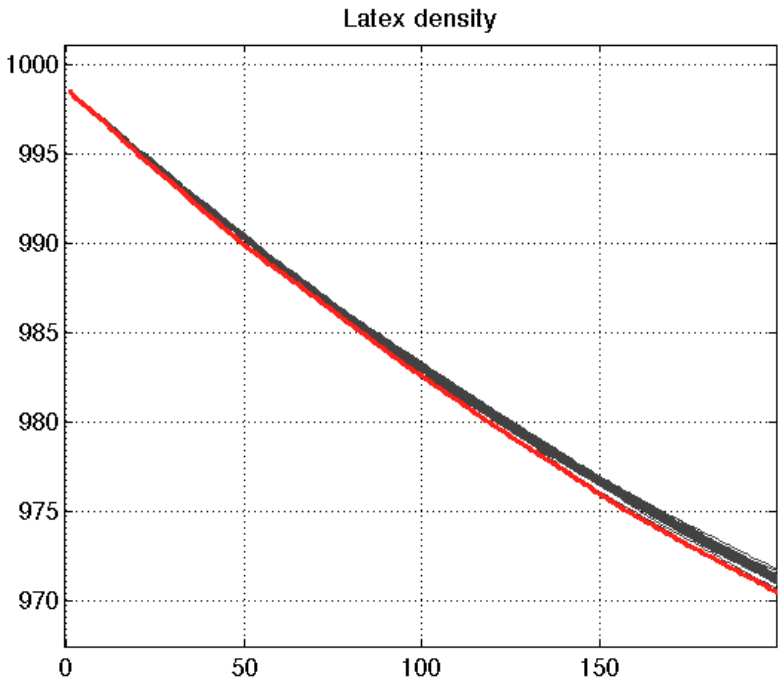


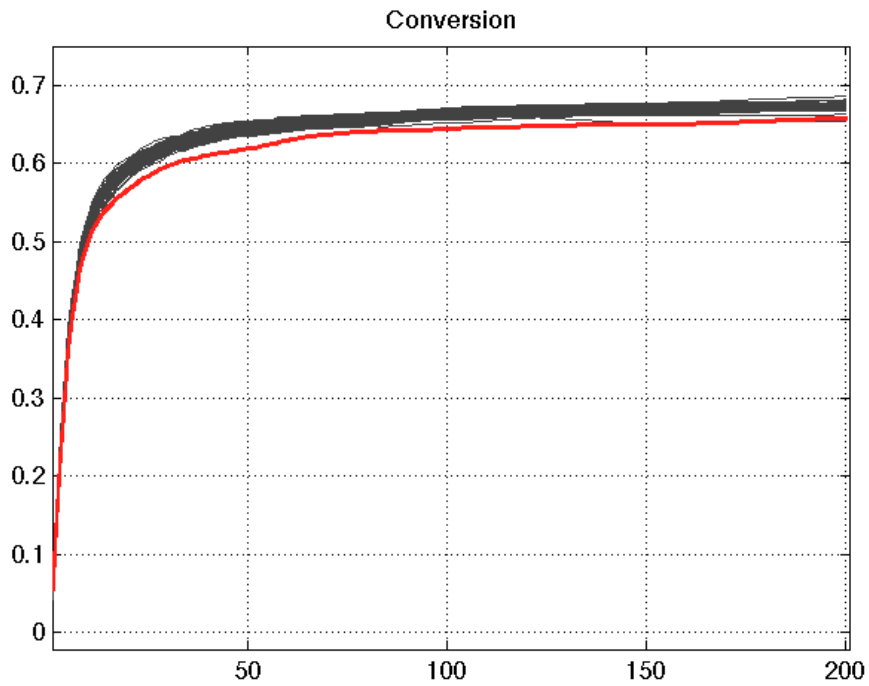
From the above we can infer that:

- batch 37 had low t_1 because of
  - **below average latex density** *throughout the batch*
  - **below average conversion** *throughout the batch*

Confirmed in the raw data, and contribution plot for batch 37 ... next

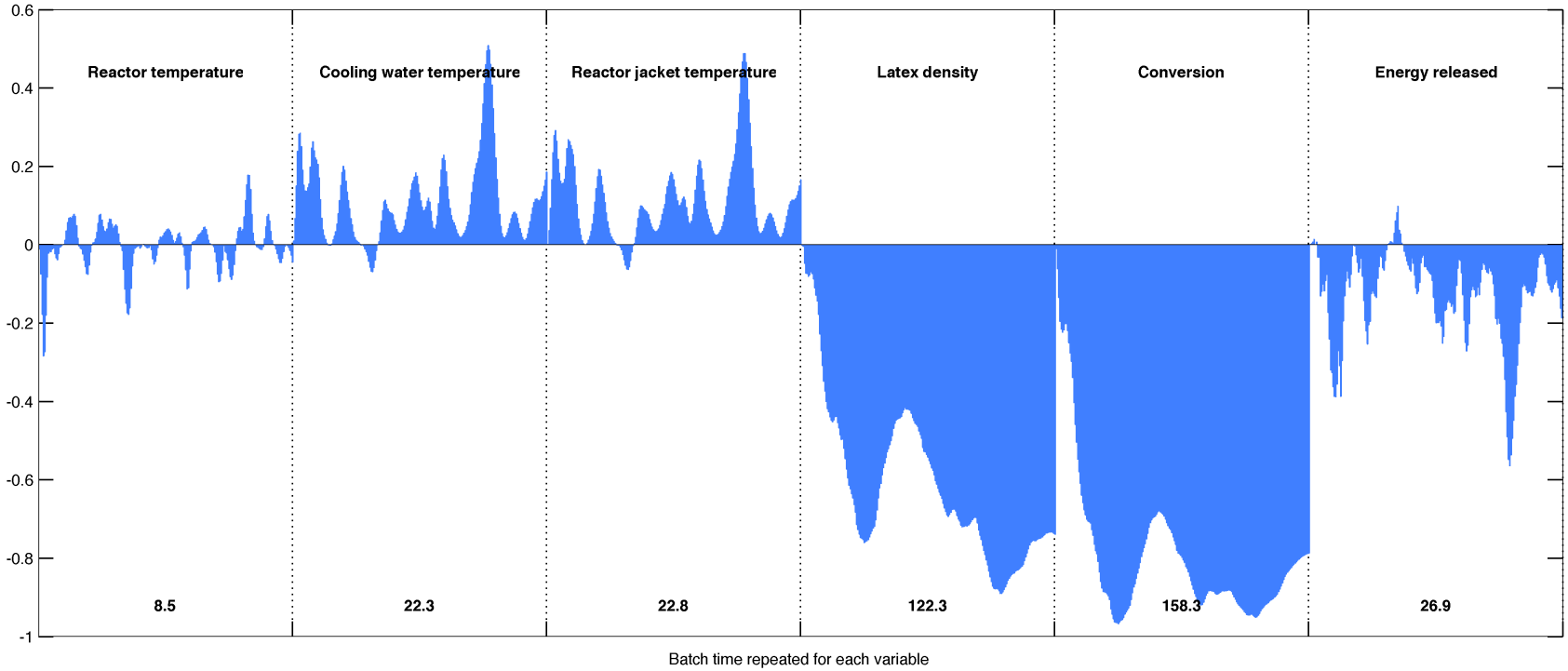
SBR: raw data for batch 37 (to confirm)





- Confirmed our interpretation with the raw data
- **True cause** (from simulation): 30% greater organic impurity in butadiene feed, from the start of the batch

SBR: contribution plot for batch 37

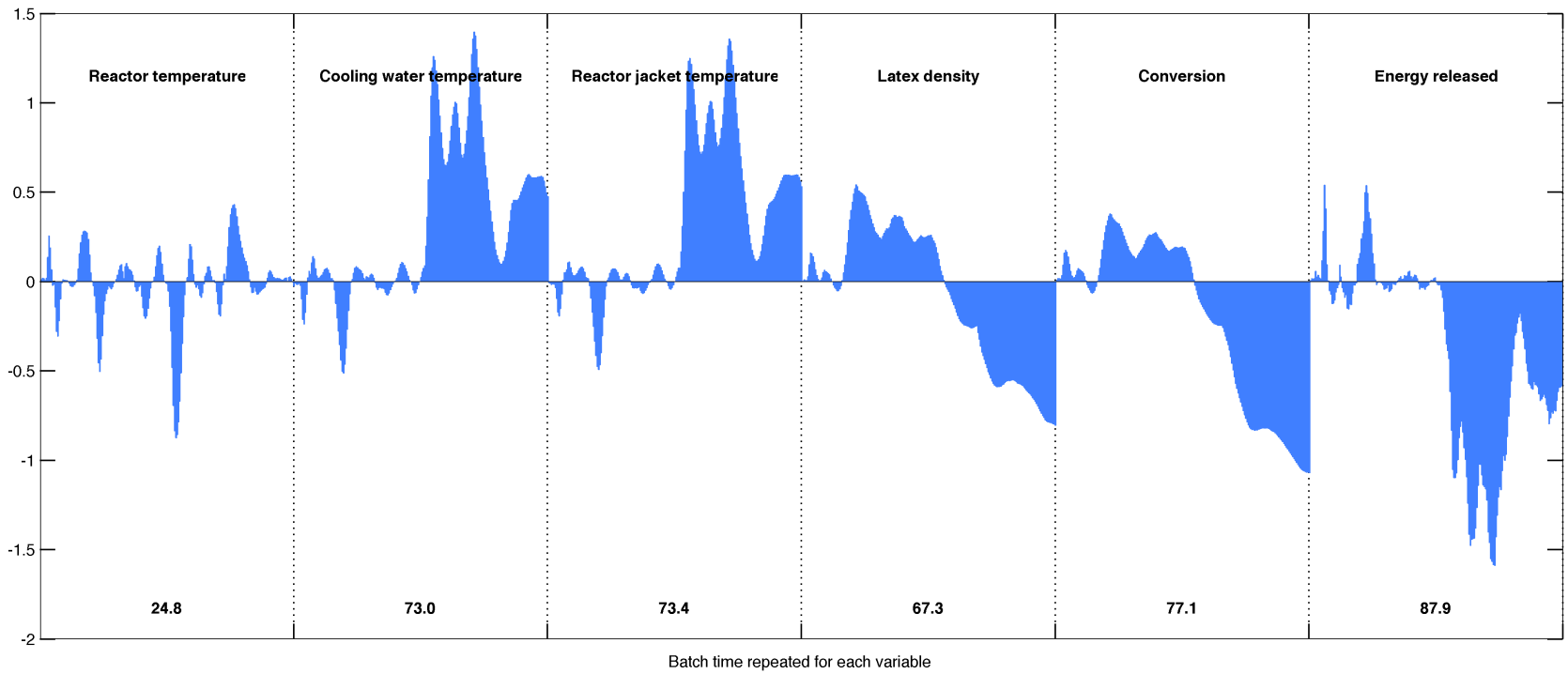


- Contributions highest for the latex density and conversion, as expected.

SBR: investigate batch 34

Batch 34 had high t_2:

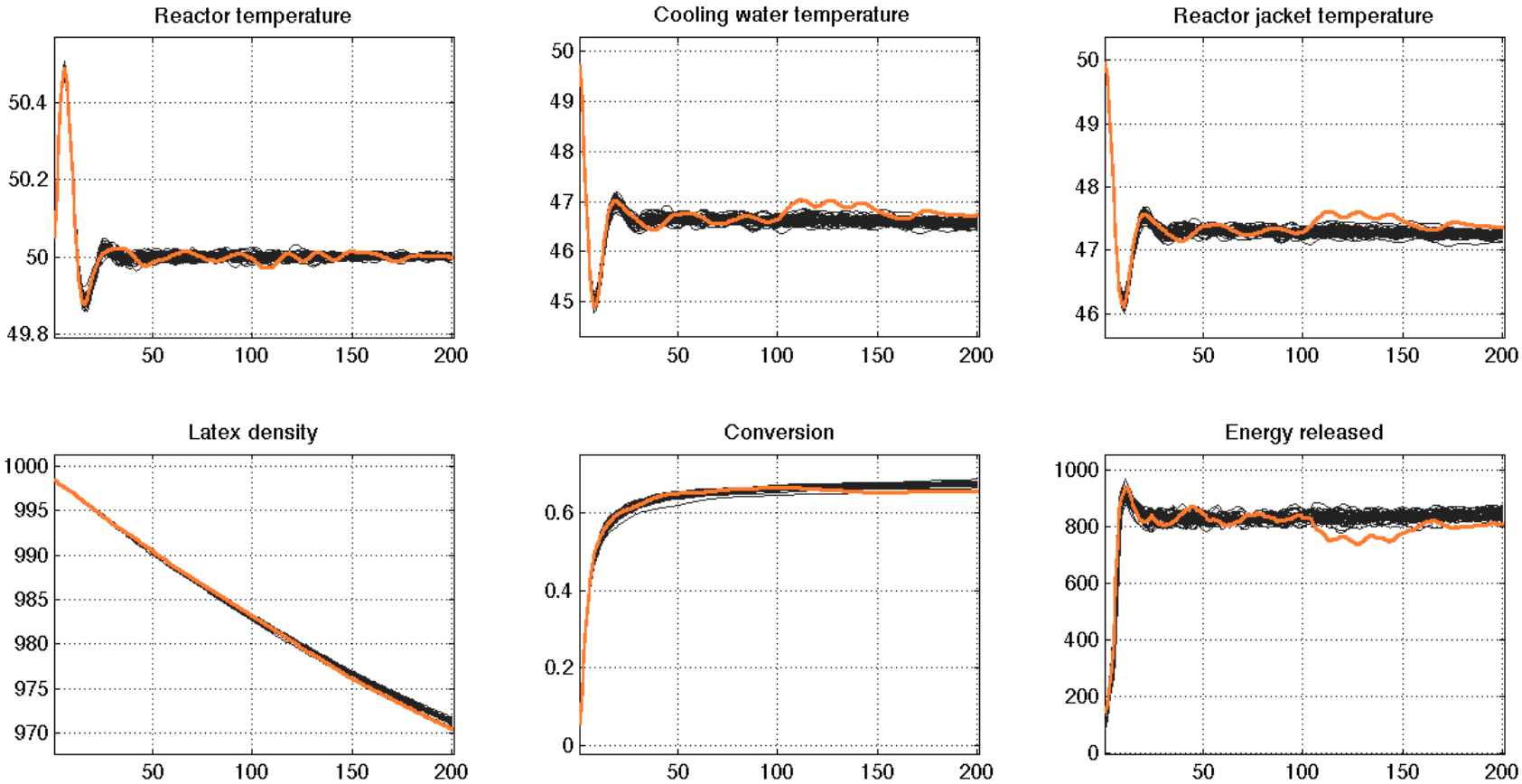
- From weights plot for w_2 (earlier): we expect the problem to be due to cooling water, jacket temperature, and below average energy released in last half of the batch
- Contribution plot confirms this:



This affected the density and conversion as well.

SBR: investigate batch 34

Raw data for this batch is highlighted



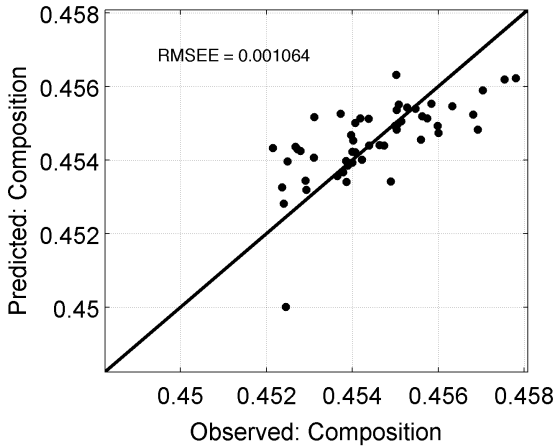
Confirms the problem occurred. Same problem as before, 30% greater organic impurity in butadiene feed, but only midway during the batch progress.

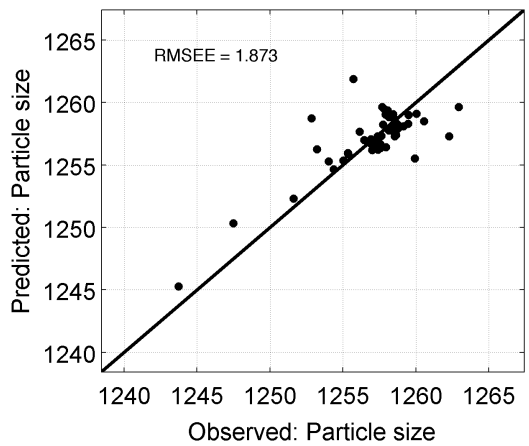
Interesting observation

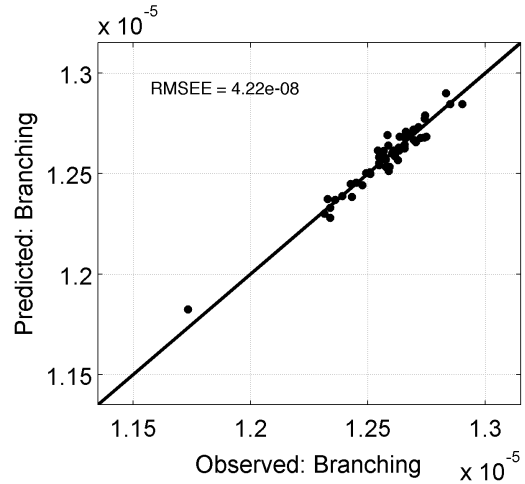
- The same fault occurred in batch 34 and 37.
- But they show up in different locations in the score plot
- Because the *time when the fault occurred* is different

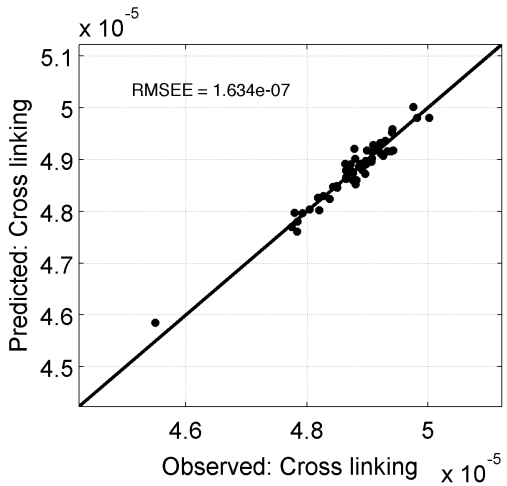
SBR: predictions from the model

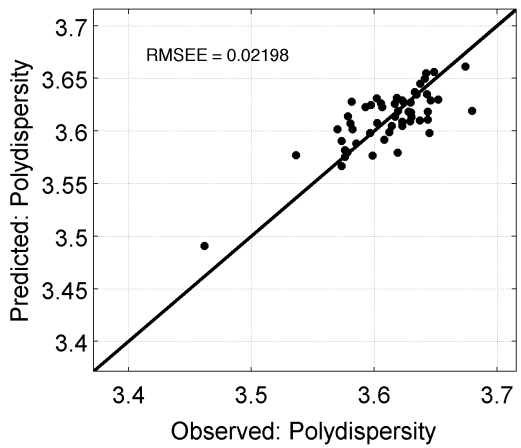
We also get predictions from the batch PLS model for the 5 quality variables:











Notice that batches 34 and 37 have poor quality attributes

Batch monitoring

Two types of monitoring

1. Off-line, post-batch monitoring
   - Use all the data after the batch is complete: score plots, SPE plots, contribution plots for new data, in the usual way
   - Allows for early release of the batch to the next stage. Don't have to wait for lab results if the batch is multivariately inside the control limits
   - *We have already covered the material for this*
   - Risk: don't just use the SPE and scores at the end of the batch: *it is also how you go to the end* that matters

2. On-line monitoring[2]
   - real-time detection of problems as a new batch progresses
   - many high value batch systems run in the order of weeks
   - save money if we detect and correct these problems before the batch end

[2] Reference: Paul Nomikos' PhD thesis

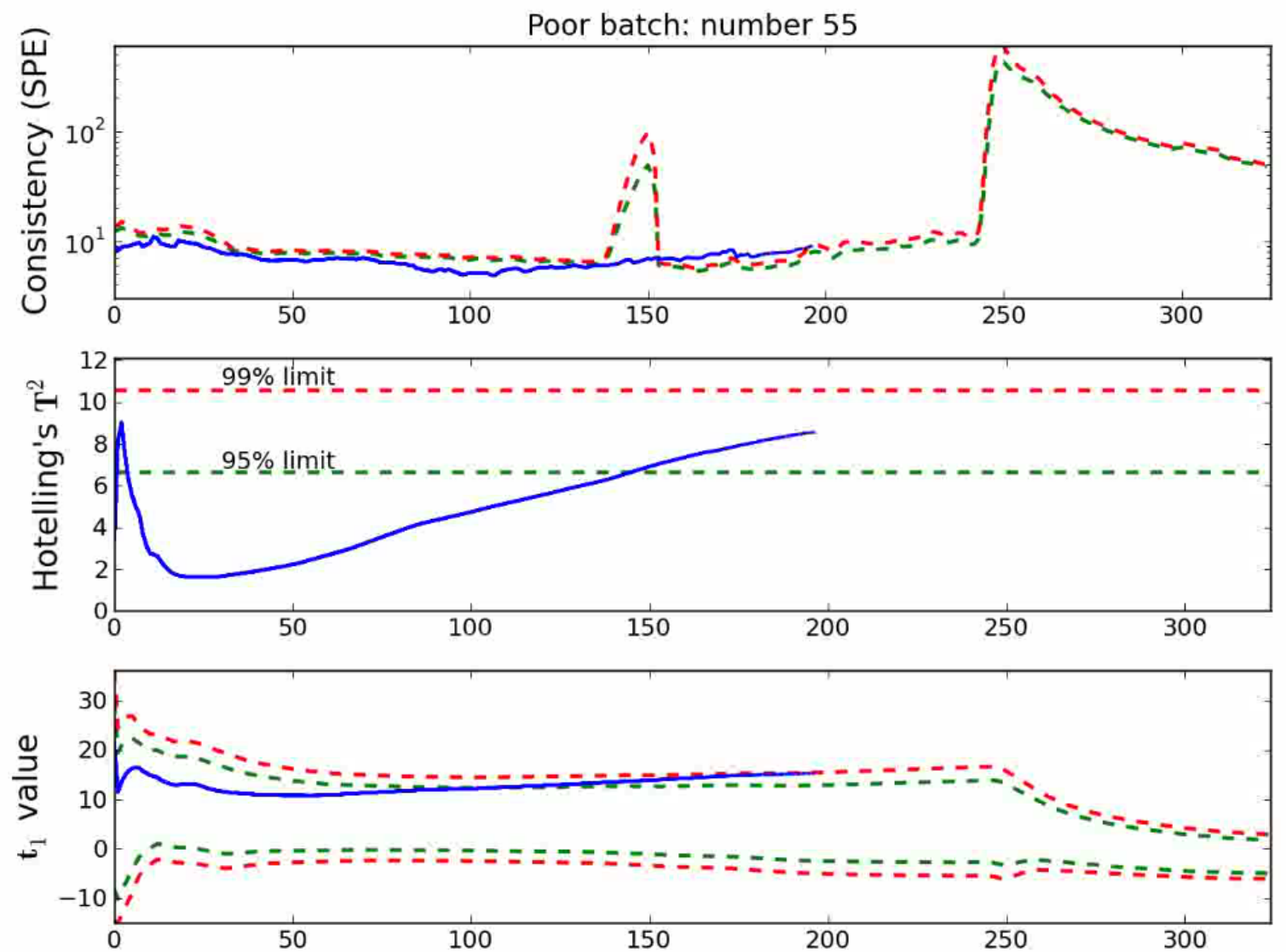
Principle of real-time monitoring (and prediction)

- While the batch progresses, at time step j, try to get the best estimate of the scores at the **end of the batch**, $\hat{\mathbf{t}}_{j,\text{new}} = \boldsymbol{\tau}_j$
- $\hat{\mathbf{x}}_{j,\text{new}} = \mathbf{P}_j \boldsymbol{\tau}_j$ $\leftarrow$ predicted trajectory at time j
- $\mathbf{e}_{j,\text{new}} = \mathbf{x}_{j,\text{new}} - \hat{\mathbf{x}}_{j,\text{new}}$ $\leftarrow$ only a $K \times 1$ vector
- $\text{SPE}_{j,\text{new}} = \mathbf{e}_{j,\text{new}}^T \, \mathbf{e}_{j,\text{new}}$ $\leftarrow$ SPE at time j
- This is called the *instantaneous* SPE
- $\mathbf{e}_{1:j,\text{new}} = \mathbf{x}_{1:j,\text{new}} - \mathbf{P}_{1:j} \boldsymbol{\tau}_j$ $\leftarrow$ a $jK \times 1$ vector
- SPE calculated using data from start to time j: called the *evolving* SPE
- Evolving SPE gets closer and closer to final SPE as $j \to J$
- For batch PLS, we get a prediction: $\hat{\mathbf{y}}_{j,\text{new}} = \boldsymbol{\tau}_j^T \mathbf{C}$

Our real time monitoring and predictions hinge on the ability to calculate the estimated end-point score, $\hat{\mathbf{t}}_{j,\text{new}} = \boldsymbol{\tau}_j$

Demonstration of batch monitoring

3 monitoring videos: good, poor, and a batch with a problem in the middle



Time-varying monitoring limits

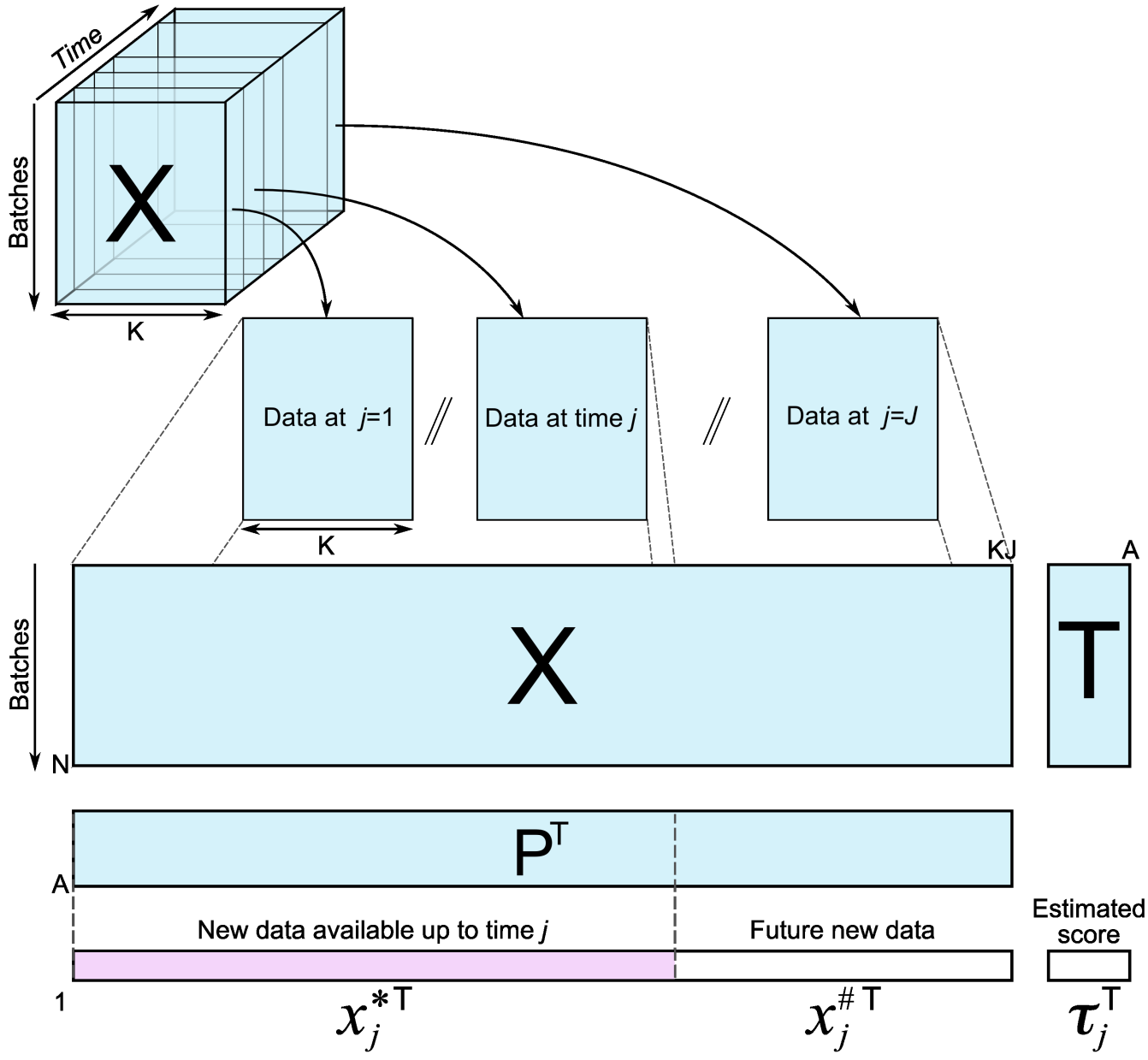
Limits for SPE and the scores vary with time[3]

- **SPE limits**
  - $\text{SPE}_j \sim g\chi^2(h)$ ← follows an approximate χ^2 distribution
  - $g = \dfrac{v}{2m}$ = premultiplier
  - $h = \dfrac{2m^2}{v}$ = degrees of freedom of $\chi^2(h)$
  - $m = \text{mean}(\text{SPE}_j)$
  - $v = \text{var}(\text{SPE}_j)$

  Use SPE values at time step j on all the good batches to estimate g and h
- **Score limits**
  - Assume $t_{a,j}$ to be normally distributed, though a t-distribution is more correct
  - Estimate mean and variance at time j from good batches

[3] Derivations in Nomikos and MacGregor paper

Real-time monitoring of a new batch



How to handle the missing future data

How to estimate the end-score: $\hat{\mathbf{t}}_{j,\text{new}} = \boldsymbol{\tau}_j$?

1. Fill future value with zeros
   - implies rest of batch runs at the average trajectory
2. Current deviations approach
   - mean centered and scaled deviation at time j is copied and pasted forward
   - implies current deviations persist (MPC assumption)
3. Missing data handling
   - Use one of the many missing data handling methods for PCA/PLS
     - score limits tend to have have variability at start, but quickly stabilize
     - single component projection, SCP: poor, but simple choice
     - projection to model plane, PMP: improves SCP somewhat
     - conditional mean replacement (CMR) or trimmed score regression (TSR) are better (good)

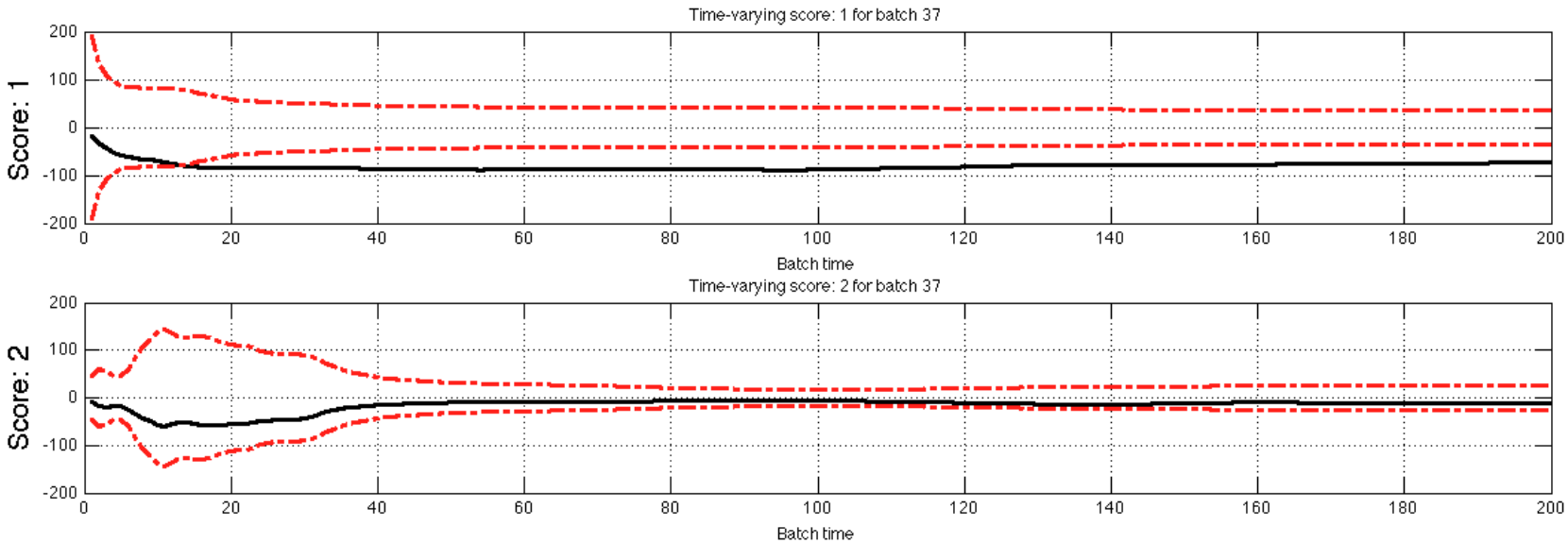
How to handle the missing future data

From a monitoring perspective:

- doesn't really matter too much which missing data method is used
- the control limits are a function of the method chosen

More details: Comparing different missing data approaches for on-line monitoring and trajectory prediction (García-Muñoz *et al.*)

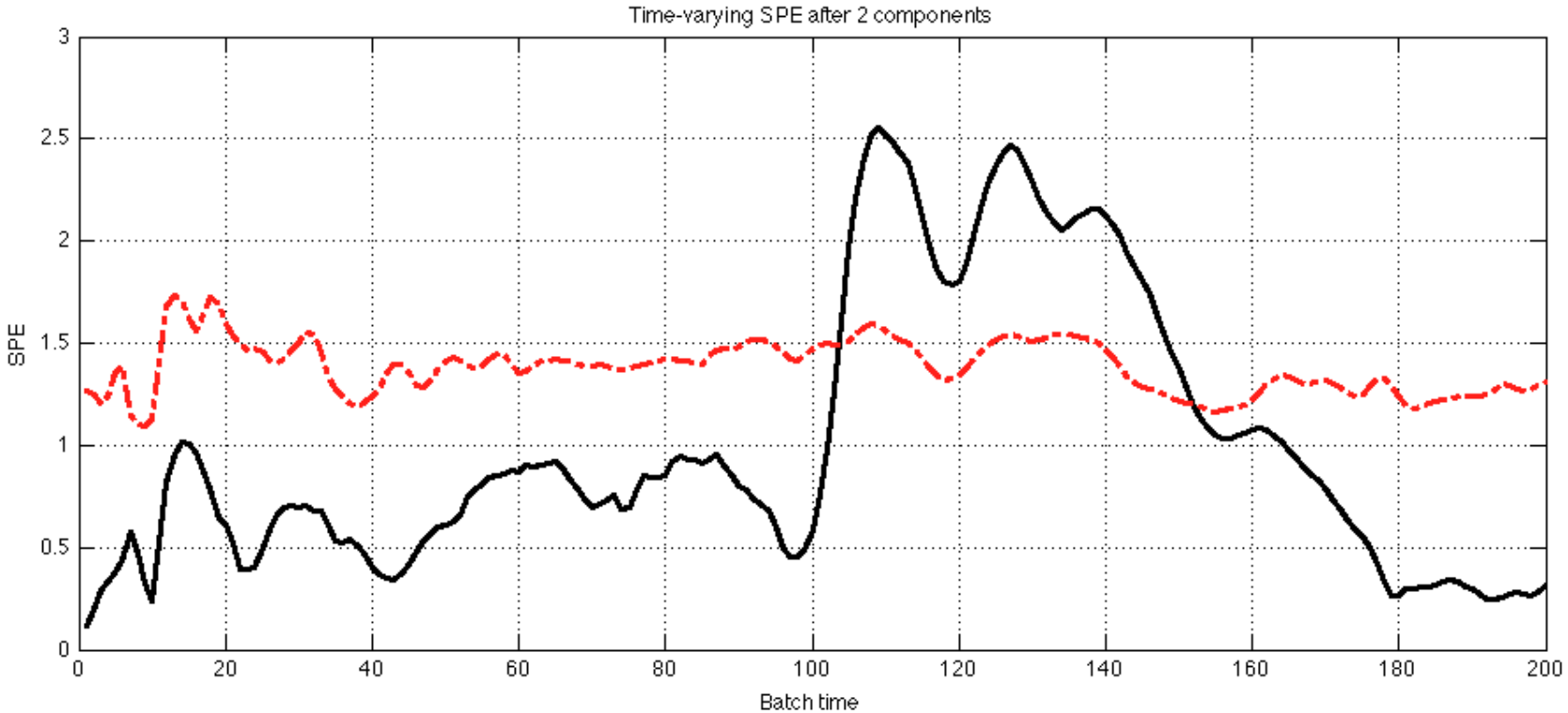
SBR: scores *over time* for batch 37



- Highlights *when* the problem occurred: right at the start
  - Was due to an impurity in the feed: consumed reactant and lowers latex density and conversion
- SPE was within limits throughout the batch

SBR example: bad batch 34

Simulation introduced impurity in feed midway, during the batch



We will use the software to diagnose the contributions

Case study: multiblock batch PLS model

This case study will introduce a number of concepts, by example. We will see:

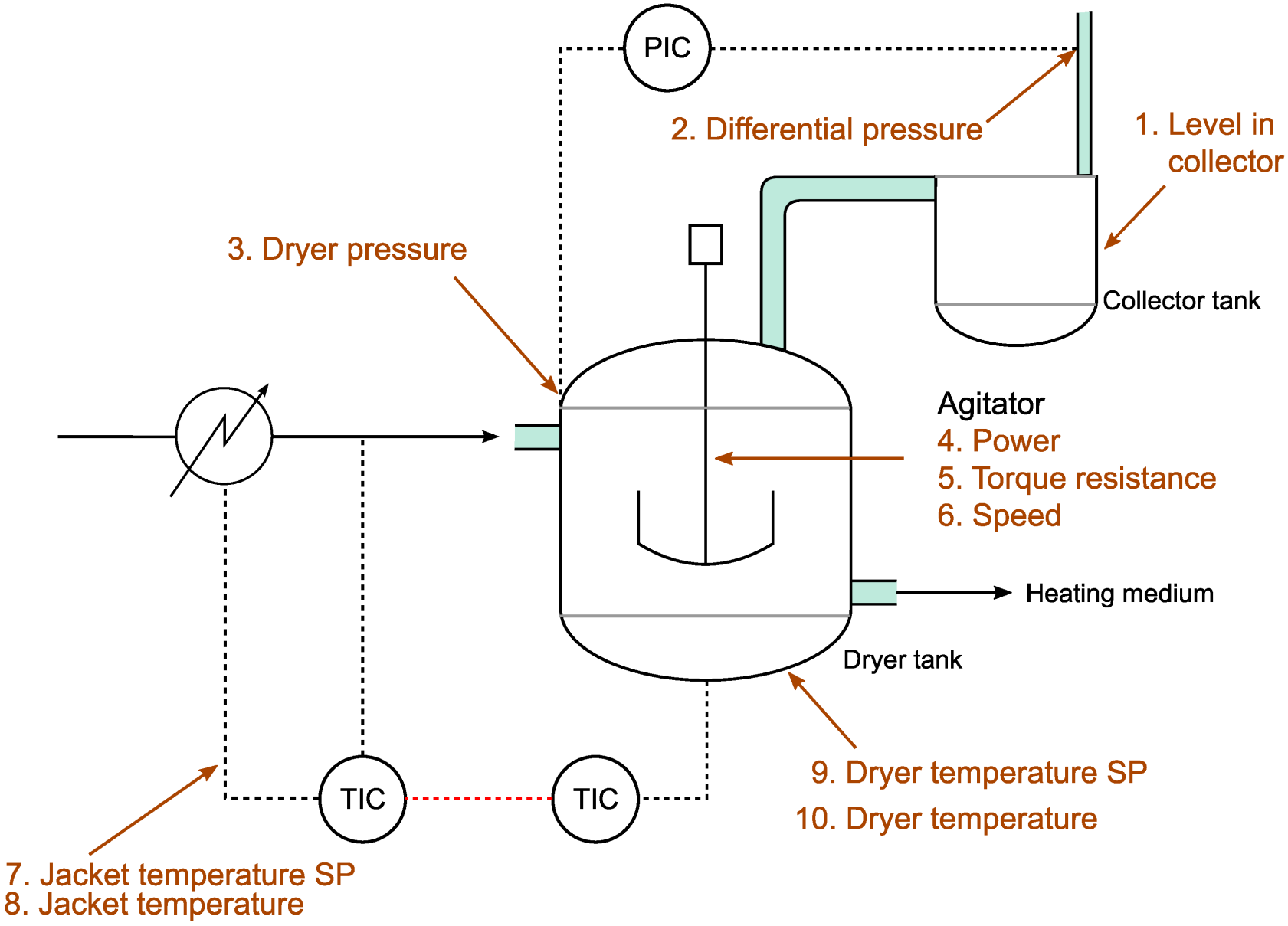
1. Multivariate characterization of product quality
2. Effect of initial conditions on product quality
3. Alignment of batch trajectories
4. Troubleshoot problems: poor product quality
5. Predictions of final quality attributes
6. Stagewise batch monitoring

Case study: multiblock batch PLS model

This case study is as complex as it gets:

- Multiblock:
  - $\mathbf{X}^{(1)}$: initial conditions (chemistry information)
  - $\mathbf{X}^{(2)}$: alignment information
  - $\mathbf{X}^{(3)}$: batch trajectories
  - $\mathbf{Y}$: quality attributes
- $\mathbf{X}^{(3)}$ contains batch trajectories
- We will work up to a multiblock PLS model for the quality predictions

Process background

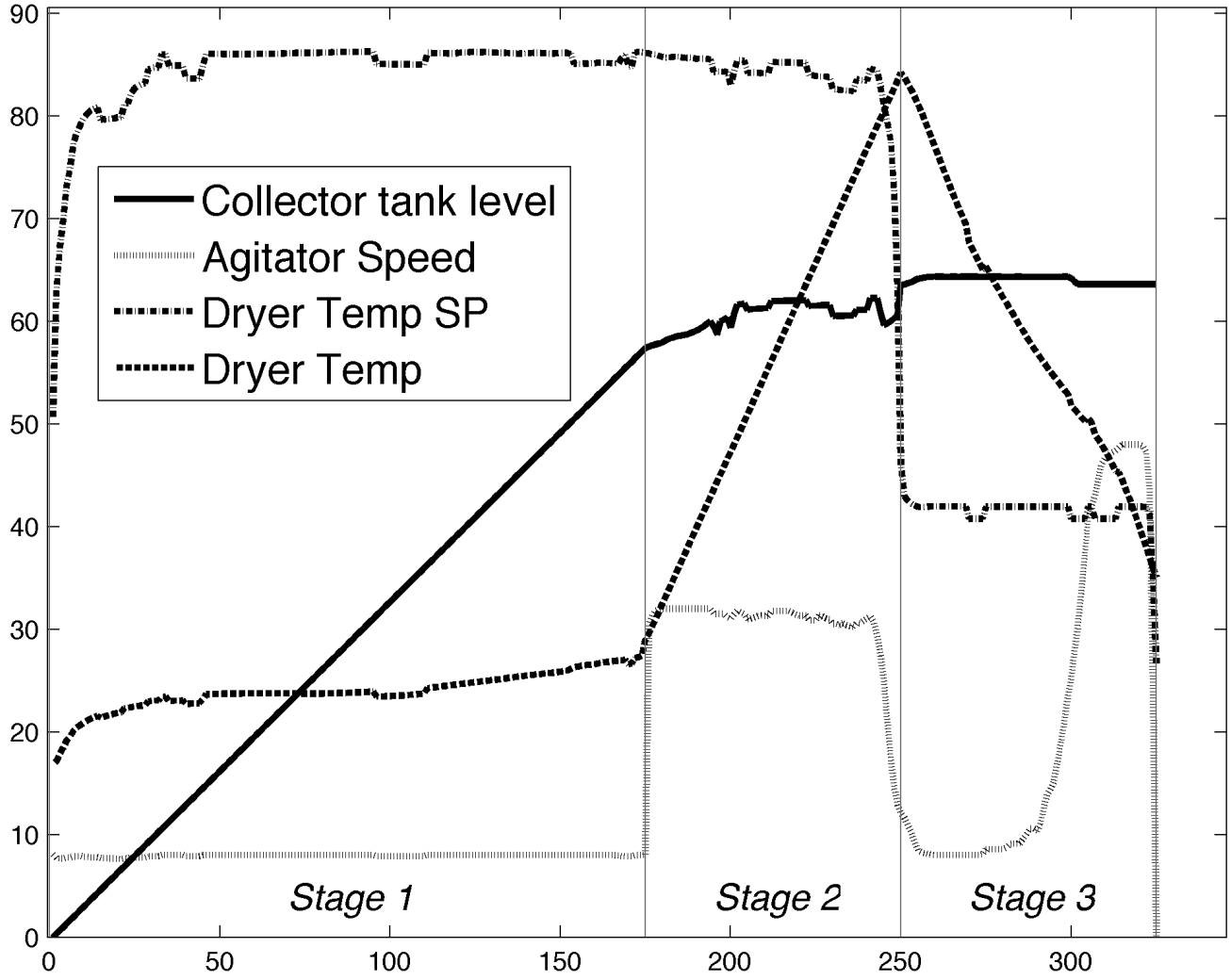


Process background

- Agricultural chemical production
- Wet “cake” (solid with embedded solvent) is charged to system and dried
- The solvent is collected in an external, side tank
- Chemical changes occur in the solid phase during drying
- 3 phases in the recipe (more details later)
- Operators can adjust some parameters

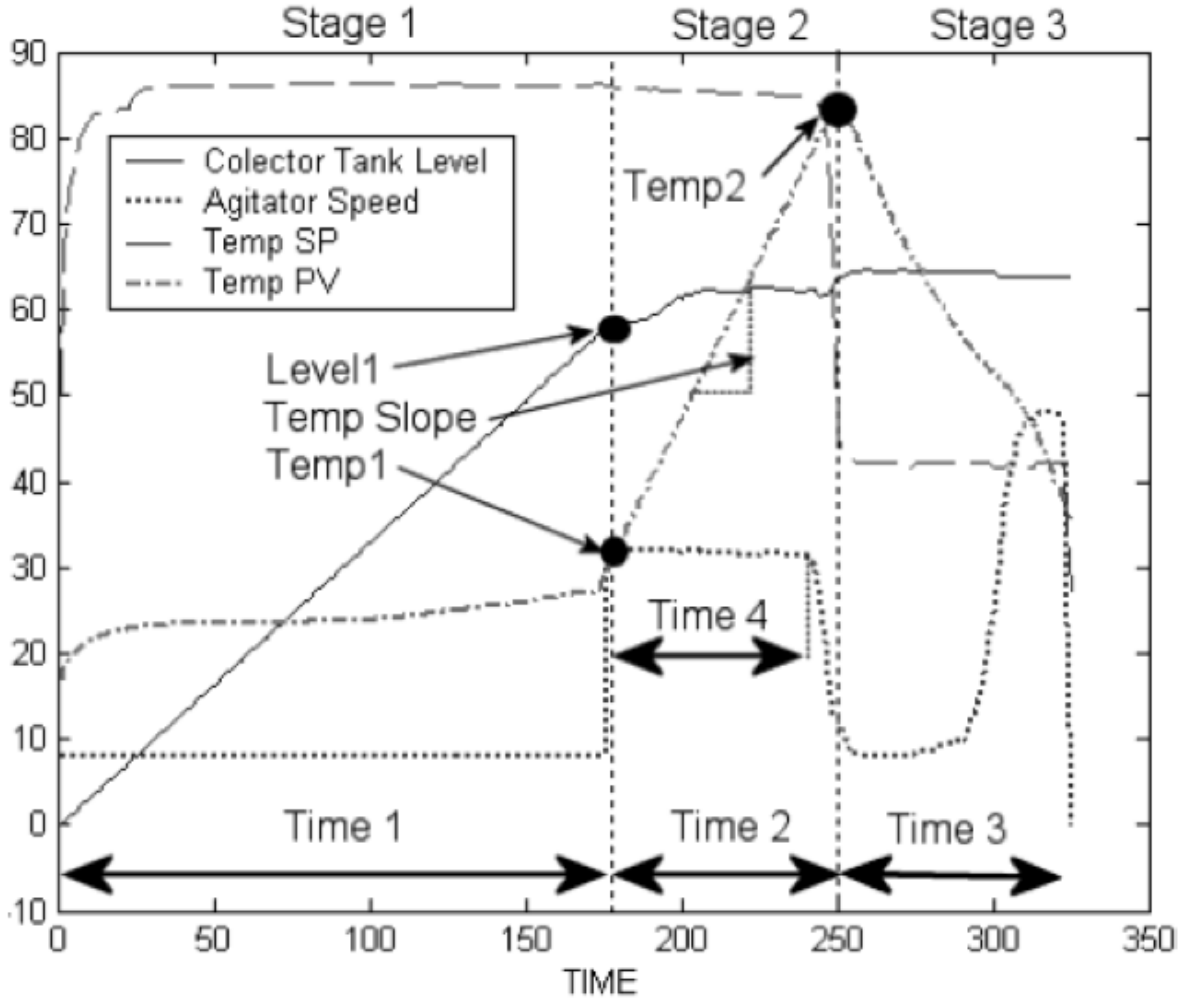
About the batch trajectories

- 10 trajectories measured per batch
- 3 phases: solvent collection, temperature ramp, cooling down



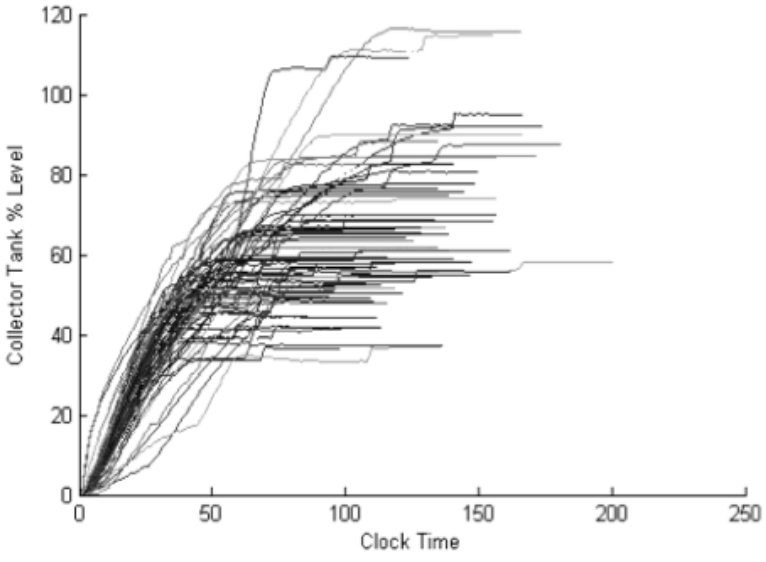
Aligning the trajectories

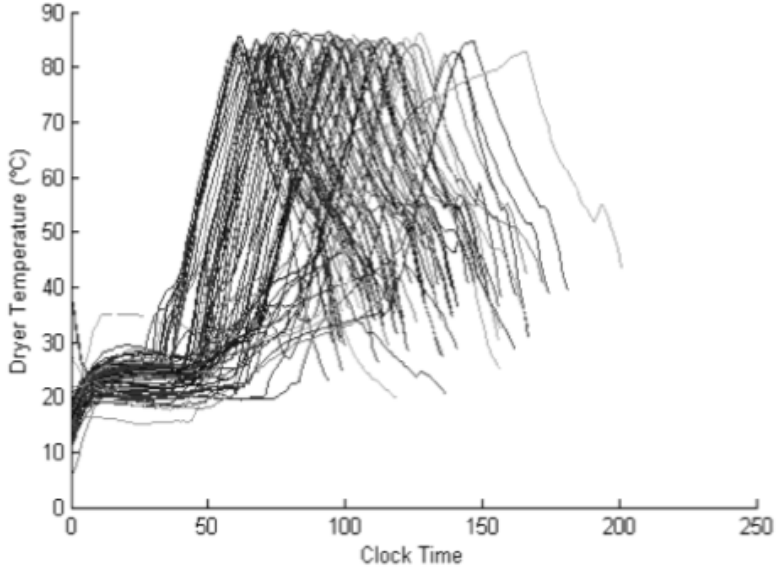
- Done within each phase

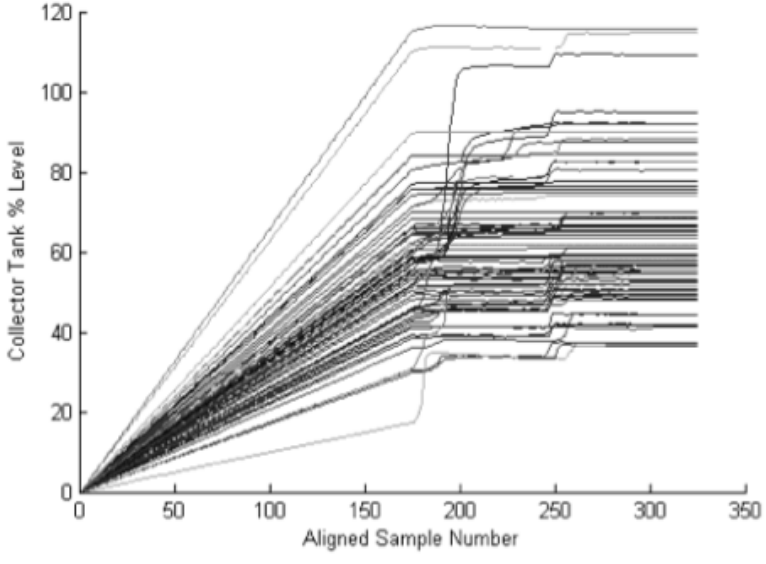


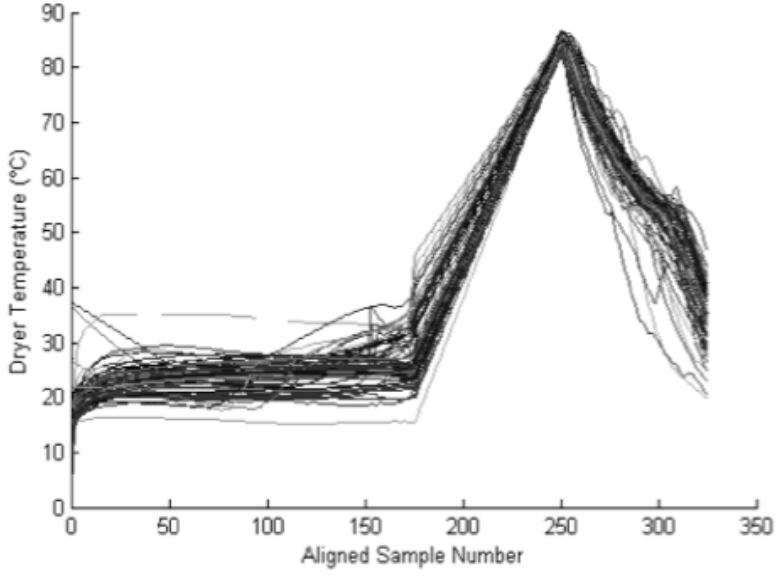
- Transfer alignment information to $\mathbf{Z}_{\text{op}} = \mathbf{X}^{(2)}$

Aligning the trajectories



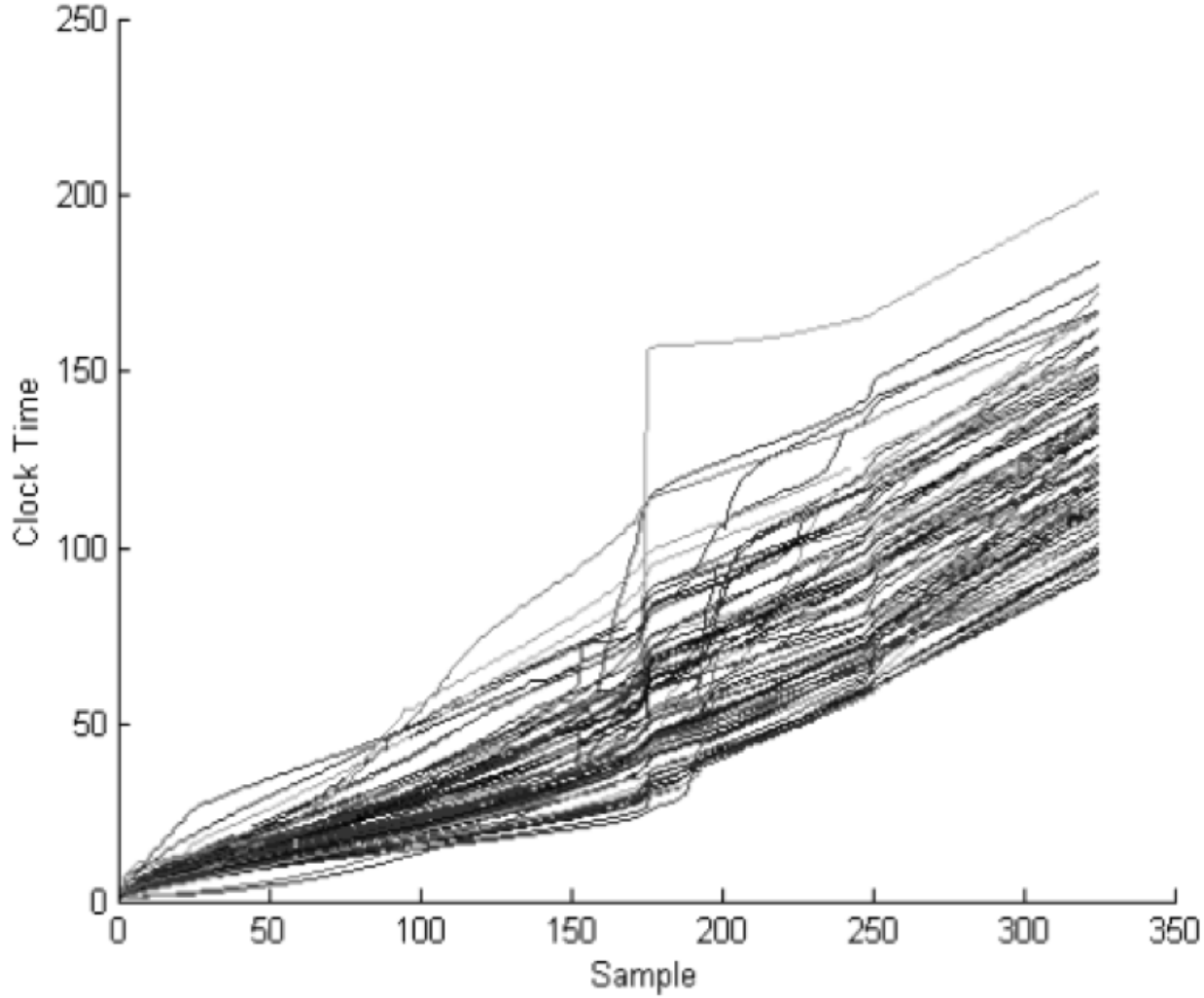






Aligning the trajectories

- Include time-distortion variable as a trajectory



Multiblock: what goes in $\mathbf{Z}$

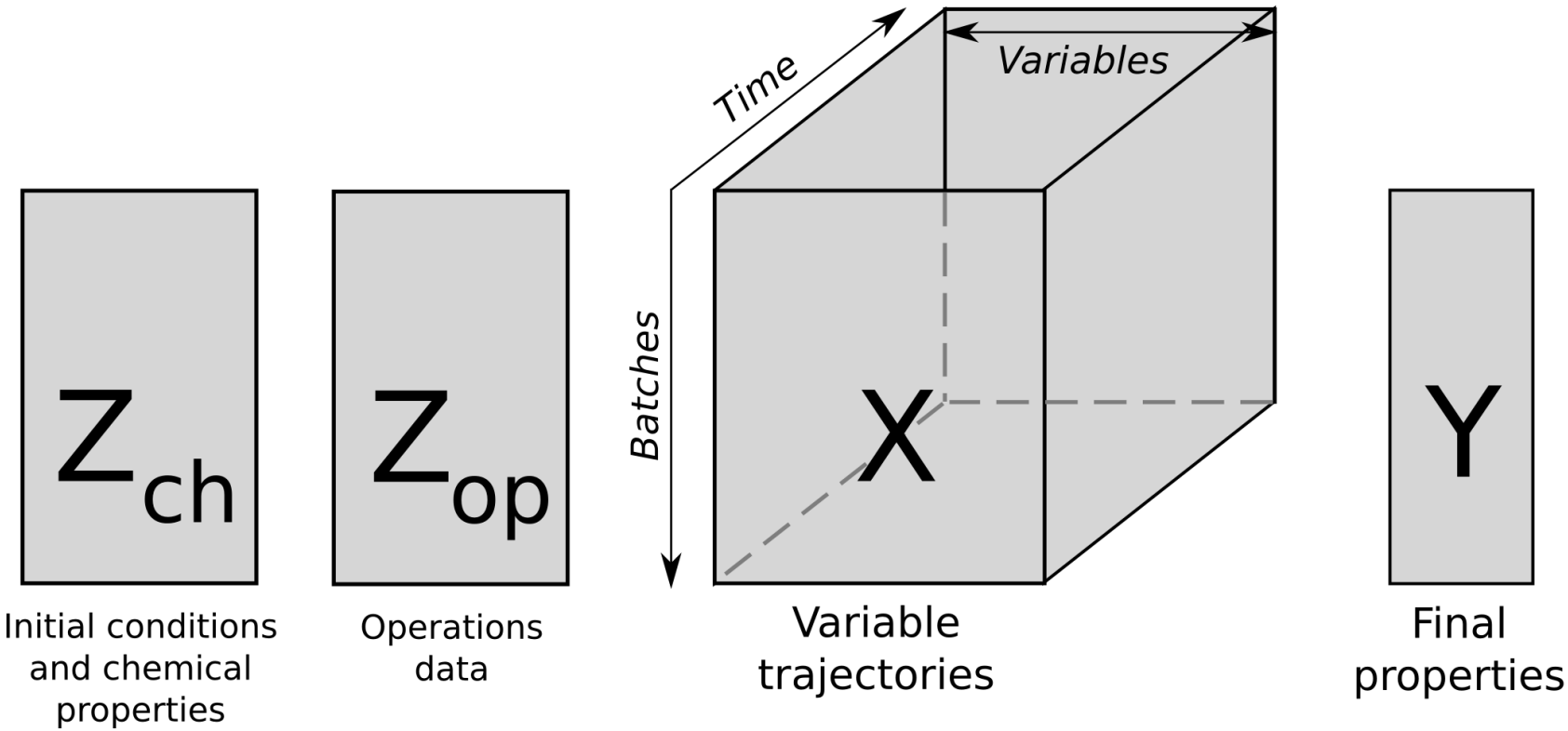
Our $\mathbf{Z}$ blocks are just $\mathbf{X}^{(b)}$ blocks in the multiblock algorithm.

Use any relevant information that is constant over the batch:

- Feed (raw material) properties and supplier code
- Feed composition (e.g. from the supplier's certificate)
- Set up time
- Summary of any upstream operations on the raw material
  - summary of raw values
  - PCA or PLS scores from upstream units
- recipe information
- alignment summary (warping factors)
- operator identifiers or shift identifier
- Properties after adding materials, but before starting the batch
  - pH, NIR spectra, temperature
- ambient conditions
- idle times between phases of the batch

A more complete analysis for product quality

- $\mathbf{Z}_{\text{chem}} = \mathbf{X}^{(1)}$: chemical properties of the cake
- $\mathbf{Z}_{\text{op}} = \mathbf{X}^{(2)}$: alignment information
- $\mathbf{X} = \mathbf{X}^{(3)}$: batch trajectories, including the time warping trajectory



Characterizing product quality: understanding the **Y** space

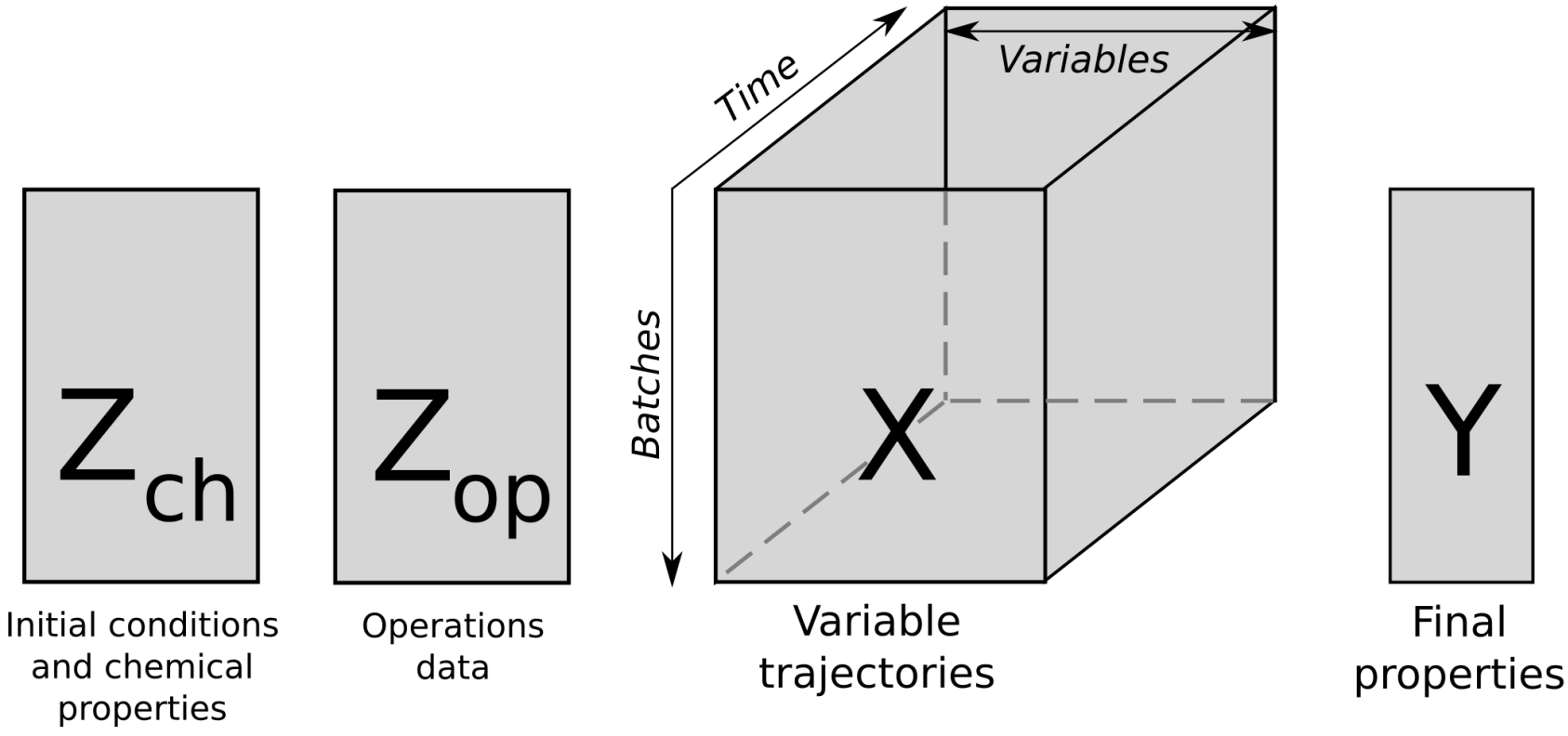
- Product quality is a multivariate property
- One should start with a PCA model of **Y**
- Look at this in the software

Batch disposition:

- Good batches: labeled 1 to 33
- Abnormal batches: labeled 34 to 61
- High residual solvent batches: labeled 62 to 71

Effect of initial conditions on product quality

- Investigate the chemistry effect: $\mathbf{Z}_{\text{chem}}$ effect on $\mathbf{Y}$
- Weight of wet cake



Multiblock PLS model

Create the following blocks:

- Timing block: all features related to timing in the batch
- Temperatures: all temperature related features
- Chemistry block: `Z1`, `Z2`, ... `Z11` and `WgtCake`
- Impeller block: power, torque and agitator
- Pressure block: pressures and tank level variables
- **Y**-block: all `Yi` tags, including `SolventConc`

