# 基于传递熵的 MPCA 间歇过程监测方法①

赵化良

(北京化工大学 信息科学与技术学院, 北京 100029)

**摘 要**: 传统统计分析方法忽略了变量间作用关系, 而传递熵可以有效地表达变量间作用关系, 因此提出了一种基于传递熵的 MPCA 间歇过程监测方法. 利用传递熵表达变量间的作用关系, 在计算传递熵时采用非参数核密度估计法, 利用该方法不依赖于数据先验分布知识的特点来处理非高斯分布的过程数据, 通过构建传递熵矩阵, 结合滑动窗, 实现对间歇过程变量间信息传递的动态表达, 最后对传递熵矩阵进行多向主元分析方法(MPCA)建模, 实现间歇过程监测. 通过青霉素发酵的仿真, 结果表明与传统多变量统计过程控制(MSPC)方法作对比, 本文监测方法能更及时准确地监测到过程异常.

**关键词**: 间歇过程; 传递熵矩阵; 核密度估计; 多向主元分析; 在线监测

## MPCA Online Monitoring Based on Transfer Entropy for Batch Process

ZHAO Hua-Liang

(Beijing University of Chemical Technology, Beijing 100029, China)

**Abstract**: The traditional statistical analysis methods ignore the relations between variables. Transfer entropy could express relations between variables effectively. So this paper proposes an MPCA online monitoring method based on entropy transfer for batch process. The transfer entropy is adopted to describe the complex relations between process variables. The non-parametric kernel density estimation method which does not depend on the prior distribution of data is utilized to calculate transfer entropy to deal with the non-Gauss distribution of the process data. By constructing the transfer entropy matrix combined with the sliding window to achieve the expression of dynamic information transfer between process variables, the MPCA model is then established based on these matrices for detecting faults of batch process. The simulation results show that, compared with the traditional MSPC method, the proposed method can timely identify the faults with better accuracy.

**Key words**: batch processes; transfer entropy matrix; kernel density estimation; multiway principal component analysis; online monitorin

间歇过程是过程工业的一个重要的分支, 特别是在医药和食品等领域占据很大的比重[1]. 不同于连续生产方式, 间歇过程是周期性的, 在生产周期内各变量间具有较强的非线性和动态性, 其过程的产物的质量和安全性受到各方面因素的影响. 为了减少经济损失和杜绝安全事故, 对间歇过程的故障采取及时的监测和诊断是十分必要的.

为了定量研究间歇过程数据的复杂统计特性, 多变量统计过程控制(Multivariable Statistical Process Control, MSPC) 方法得到了广泛的研究. Nomikos 和 MacGregor[2] 最早提出应用多向主元分析方法(Multiway PCA, MPCA)建立多元统计模型对间歇过程进行监测, 此后基于数据驱动的间歇过程监测方法如雨后春笋般不断涌现. 为了改善因 MPCA 自身线性化特点而无法处理复杂的非线性系统的缺陷, Lee 等[3]提出了核 MPCA 方法, 通过将原始非线性数据映射到高维近似线性空间进行建模分析, 得到了良好的效果. 然而在实际中, 间歇过程数据往往不能完全满足 MPCA

① 收稿时间:2015-05-20;收到修改稿时间:2015-07-02

假定的服从高斯分布, 于是 Yoo[4]提出了多向独立成分分析(Multiway ICA)方法. 针对过程变量的非线性问题, Zhang Y W 等[5]将核技术与 ICA 相结合并应用到间歇过程的在线监测和故障诊断领域. 为了更好地体现出间歇过程的局部特征和提高统计模型的精度, 谢晓庆等[6]研究了基于时段和过渡区域的 KICA 间歇过程监测方法. 间歇过程故障的发生往往伴随着变量间作用关系的改变, 以上的统计方法忽略了过程变量间的作用关系, 因此不能及时监测出变化缓慢的故障.

从信息论的角度, 由于间歇过程各正常批次中变量间的信息作用关系是相对稳定的, 但在故障批次中, 这种关系会发生显著的变化, 所以可以利用传递熵(Transfer Entropy, TE)来有效地表达这种信息作用关系. 传递熵概念最早由 Schreiber[7,8]于 2000 年提出后, 得到了大量的研究和应用. 得益于传递熵不仅能够定量度量变量间的线性和非线性关系, 而且能够指出变量间影响关系的方向性, 同时其结果易于从过程机理上进行解释, 传递熵被应用于刻画复杂系统内部的动态非线性特征[9], 尤其在计算神经科学方面得到了重要应用[10,11]. Samuel R 等[12]利用传递熵来定量定向地度量供热通风与空气调节(HVAC)系统的传感器模组间的交互信息, 实现了对 HVAC 系统的故障监测和诊断. Bauer M 等[13]基于化工过程历史数据将传递熵应用于过程的扰动传播, 通过设置特定的算法参数, 实现了对过程扰动根事件的确定, 并与过程机理相吻合, 验证了传递熵方法的可行性. Duan P 等[14]进一步将传递熵概念进行了拓展, 得到微分传递熵, 并将其成功应用于化工过程故障传播路径上的直接因果关系和间接因果关系的判定方面, 并从过程机理方面得到了很好的解释. 以上方法都是利用传递熵来确定系统变量间的拓扑关系, 实现对系统整体的信息传递的表达.

本文应用传递熵理论与核密度估计法并结合滑动窗来构建传递熵矩阵, 这些矩阵数据表达了间歇过程变量在时间维度上的动态信息传递, 提出了基于传递熵的 MPCA 间歇过程监测方法, 通过统计分析过程变量间的信息传递变化, 达到故障监测的目的.

## 1 传递熵方法

### 1.1 传递熵理论

传递熵是信息熵理论一个较新的分支, 它是用来刻画系统中时间序列间信息传递量的重要工具. 本文利用传递熵来表达间歇过程变量间的信息传递关系.

设 $X_n$ 和 $Y_n$ 为两个在 $n$ 时刻具有 $x_n$ 和 $y_n$ 离散状态的时间序列, 且 $X_n$ 和 $Y_n$ 分别可以近似为 $k$ 阶和 $l$ 阶的稳态马尔科夫过程, 那么从 $Y_n$ 到 $X_n$ 的传递熵定义式[7]如下:

$$T_{Y\to X}=\sum_{u_n}p(u_n)\log\frac{p\left(x_{n+1}\mid x_n^{(k)},y_n^{(l)}\right)}{p\left(x_{n+1}\mid x_n^{(k)}\right)} \quad (1)$$

其中 $T_{Y\to X}$ 表示 $Y$ 到 $X$ 的传递熵(Transfer Entropy), $u_n=(x_{n+1},x_n,y_n^{(l)})$, $p(u_n)$ 表示状态 $x_{n+1}$ 和序列 $x_n^{(k)}$, $y_n^{(l)}$ 同时出现的概率; $p(x_{n+1}\mid x_n^{(k)},y_n^{(l)})$ 表示在 $n$ 时刻, 已知 $x_n^{(k)}$, $y_n^{(l)}$ 的前提下, $x_{n+1}$ 的条件概率; $p(x_{n+1}\mid x_n^{(k)})$ 表示 $x_n^{(k)}$ 已知前提下 $x_{n+1}$ 的条件概率. 当 $x_n$ 在某个时刻的状态完全由自身的历史状态决定时, 传递熵为零. 本文取 2 为对数底, 此时传递熵单位为 *bit*, 为了计算简便取 $l$=$k$=1.

为了计算传递熵, 需要对观测数据采取有效的概率密度估计, 核密度估计方法作为非参数密度估计技术中的重要方法, 得到了广泛的应用. 它的优点是不依赖于数据分布的先验知识, 可以有效解决数据的非高斯分布问题. 设时间序列的概率密度函数是 $p(x_i^{(k)})$, $\hat{p}(x_i^{(k)})$ 是 $p(x_i^{(k)})$ 的估计, 由文献[8]得到式(1)的估计值 $\hat{T}_{Y\to X}$ 可由下式计算:

$$\hat{T}_{Y\to X}=\frac{1}{N}\sum_{n=1}^{N}\log\frac{\hat{p}(x_{n+1},x_n^{(k)},y_n^{(l)})\hat{p}(x_n^{(k)})}{\hat{p}(x_n^{(k)},y_n^{(l)})\hat{p}(x_{n+1},x_n^{(k)})} \quad (2)$$

其中联合概率密度函数的估计[7]如下:

$$\hat{p}_r(x_n,y_n)=\frac{1}{N}\sum\nolimits_{n'}\Theta\left(\left\|\begin{pmatrix}x_n-x_{n'}\\ y_n-y_{n'}\end{pmatrix}\right\|-r\right) \quad (3)$$

核函数 Θ 定义为(Θ(x>0)=1, Θ(x<0)=0), |·|表示最大距离, $r$ 为核半径, 可见联合概率密度函数是序列 $X_n$ 和 $Y_n$ 之间几何距离的函数.

### 1.2 改进的传递熵方法

为了描述变量间信息作用关系的时序动态性, 本文采用滑动窗来动态表达变量间的传递熵的变化.

滑动窗是通过窗口宽度和移动步长来逐步更新当前数据子空间的采样数据, 对于正常工况的间歇过程数据 $X(I\times K\times J)$(其中 $I$ 代表批次, $K$ 代表时间, $J$ 代表被测变量), 利用滑动窗技术将 $X$ 划分为连续的数据子空间 $X_L(I\times SW\times J)$, $SW$ 为窗口宽度, 内含 $SW$ 个采样数据, $L$ 为窗口移动步长, 内含 $L$ 个采样数据, ($L$<$SW$). 这两

个值的大小要合适, *SW* 太小时滑动窗内样本容量过小将不利于核密度估计的性能, 继而对传递熵的计算产生影响; *L* 太大将不能及时跟踪过程的变化.

在进行监测时, 若需要每隔 *p* 小时对过程进行一次状态监测, 过程采样间隔为 *q* 小时, 那么取 *L*=*p*/*q*. 可知对过程异常状态监测的最小识别时间是 *p* 小时.

经过以上处理后原过程变量数据就可以表示为一系列数据子空间的集合形式:

$$X(I\times K\times J)=\{X_{0L}(I\times SW\times J)X_{1L}(I\times SW\times J)\ X_{2L}(I\times SW\times J)\cdots X_{nL}(I\times SW\times J)\} \quad (4)$$

其中 *n* 为滑动窗口在时间轴上移动的次数.

在滑动窗口的同时需要不断计算过程变量之间的传递熵值, 于是每个滑动窗口将得到一个 *J*×*J* 的传递熵矩阵, 其表达式如下所示:

$$te=\begin{bmatrix} te_{11} & te_{12} & \cdots & te_{1m} \\ te_{21} & te_{22} & \cdots & te_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ te_{m1} & te_{m2} & \cdots & te_{mm} \end{bmatrix} \quad (5)$$

其中, $te_{ij}(i,j=1,2\cdots m)$表示序号为 *i* 的变量对于序号为 *j* 的变量的传递熵; 变量对自身的传递熵取值设为 $te_{mm}=NULL(m=1,2\cdots n)$; 易知传递熵矩阵为非对称矩阵, 即有 $te_{ij}\neq te_{ji}(i,j=1,2\cdots m)$.

在整个批次内最终得到 *n*+1 个这样的传递熵矩阵, 这些矩阵的变化实现了对间歇过程总体变量间的信息相互作用的动态描述, 然后将这些矩阵数据进行 MPCA 建模分析.

## 2　基于传递熵的MPCA建模故障监测方法

变成四维数据 *X*(*I*×*J*×*J*×*K*), 其中 *K*=*SW*+*nL*, 首先将每个时刻的传递熵矩阵 *J*×*J* 展开成 $1\times\dot{J}$ 的向量 ($\dot{J}=J\times J$), 然后将三维数据 $X(I\times\dot{J}\times K)$ 按变量展开为二维矩阵 $X(KI\times\dot{J})$, 然后对其进行 PCA 建模, 模型如下:

$$\boldsymbol{X}=\sum_{r=1}^{R}\boldsymbol{t}_r\boldsymbol{p}_r+\sum_{i=R+1}^{JK}\boldsymbol{t}_i\boldsymbol{p}_i=\boldsymbol{T}_r\boldsymbol{P}_r^{\mathrm{T}}+\boldsymbol{E} \quad (6)$$

其中, *Pr* 为主元负载矩阵, *Tr* 为主元得分矩阵, *E* 为残差矩阵, 主元个数 *R* 由累计方差贡献率>85%确定.

### 2.1 利用训练数据样本计算 $T^2$ 和 SPE 控制限

建立模型后计算 $T^2$ 和 *SPE* 统计量控制限, $T^2$ 统计量的控制限由 *F* 分布按下式进行计算:

$$T_{R,n,\alpha}^2=\frac{R(\mathrm{n}-1)}{n-R}F_{R,n-1,\alpha} \quad (7)$$

其中, *n*、*R* 分别为样本个数和主元个数, $F_{R,n-1,\alpha}$ 是置信水平为 $\alpha$ 的自由度为 *R* 和 *n*-1 的 *F* 分布的临界值.

*SPE* 统计量的控制限由下式进行计算:

$$SPE_{k,\alpha}=g_k\chi_{h_k,\alpha}^2 \quad (8)$$

$$g_k=\frac{v_k}{2m_k} \quad (9)$$

$$h_k=\frac{2m_k^2}{v_k} \quad (10)$$

其中, $v_k$ 与 $m_k$ 分别为 *k* 时刻 *SPE* 统计量的方差与均值, $\chi^2_{2m_k^2/v_k}$ 对应自由度为 $2m_k^2/v_k$ 的置信水平为 $\alpha$ 的卡方分布临界值.

离线建模的具体步骤如下所述:

1) 计算单批次的三维 $J\times J\times K$ 传递熵矩阵;

2) 按照步骤 1)的方法计算 *I* 个批次的传递熵矩阵,最终得到四维数据 $X(I\times J\times J\times K)$;

3) 将 *I* 批次 *K* 时刻的 $J\times J$ 矩阵依次展开成 $1\times\dot{J}$ 的向量, 得到三维数据 $X(I\times\dot{J}\times K)$;

4) 继续将步骤 3)中的三维矩阵按变量展开成二维矩阵 $X(KI\times\dot{J})$.

5) 使用数据 *X* 进行 PCA 建模, 利用公式(7)和(8)计算 *SPE* 和 $T^2$ 统计量控制限.

### 2.2 检测数据样本

当离线建立好基于传递熵的 MPCA 模型后, 在线监测时计算各时刻 *K* 的 $T^2$ 和 *SPE* 值:

$$T_{k,i}^2=t_{k,i}\times S_k^{-1}\times t_{k,i}^T \quad (11)$$

$$t_{k,i}=X_{k,i}P_k \quad (12)$$

$$S_k=diag\{\lambda_{k,1},\lambda_{k,2},\cdots,\lambda_{k,\mathrm{R}}\} \quad (13)$$

$$SPE_{k,i}=\left(x_{k,i}-\hat{x}_{k,i}\right)\left(x_{k,i}-\hat{x}_{k,i}\right)^T \quad (14)$$

其中, $x_{k,i}$ 是第 *i* 批次 *k* 时刻的过程数据, $t_{k,i}$ 为 $x_{k,i}$ 的主成分向量.

通过如下步骤实现在线监测:

1) 采样得到实时过程数据, 对其进行预处理, 计算滑动窗口内变量间的 *J*×*J* 传递熵矩阵;

2) 将步骤 1)中的矩阵展开成 $1\times\dot{J}$ 的向量, 并按公式(11)和(14)计算 *SPE* 和 $T^2$ 统计量;

3) 判断该时刻 *SPE* 和 $T^2$ 统计量是否超过对应控制限, 若均未超过, 判断工况正常; 否则判断工况异常.

## 3 实验分析

青霉素具有极高的医用价值, 故对其进行有效的故障监测具有重大的意义. 本文数据来源于 Pensim2.0 仿真平台[15], 产生 $I$=20 批次的正常工况数据, 表 1 是数据的初始条件范围, 标注星号的变量为本文选取的监测变量, 此外还包括青霉素浓度和产热量共计 10 个关键过程变量.

设置发酵过程总反应时间为 400h, 采样间隔 0.01h, 组成三维数据矩阵 $X$(20×10×40000). 通过多次对比实验, 取 $L$=50, 即移动步长 0.5h, $r$=0.12, $SW$=500, 窗宽为 5h, 能达到较好的监测效果.

表 1 青霉素发酵过程变量

| 初始条件 | 变化范围 | 监测变量 |
|---|---|---|
| 溶氧浓度(mol/L) | 1.1-1.2 | * |
| CO2 浓度(g/L) | 0.5-0.6 | * |
| 基质浓度(g/L) | 13.0-16.0 | * |
| 菌体浓度(g/L) | 0.09-0.15 | * |
| 通风率(L/h) | 8.2-8.8 | * |
| 搅拌功率(W) | 26-33 | * |
| 底物流加速度(L/h) | 0.038-0.043 | * |
| 培养基体积(L) | 100-105 | * |
| 底物流温度(K) | 296-297 | |
| 发酵罐温度(K) | 298-299 | |
| pH 值 | 4.8-5.2 | |

间歇过程原始三维数据经传递熵方法处理后得到四维传递熵数据矩阵 $X$(20×10×10×790), 其中 10×10 为一个传递熵矩阵, 且每个批次有 790=(40000-500)/50 个传递熵矩阵, 将每个 10×10 个矩阵按行展开成 1×100 的向量, 于是四维数据矩阵被展开成三维数据矩阵 $X$(20×100×790), 再继续按变量展开成二维数据矩阵 $X$(15800×100), 经标准化后进行 PCA.

引入新的故障进行在线监测实验, 与 MPCA 方法和 KICA 方法[6]做对比, 验证本文方法的可行性和有效性, 故障数据描述见表 2.

表 2 青霉素发酵过程故障描述

| 故障编号 | 过程描述 | 故障类型 | 故障位置(h) |
|---|---|---|---|
| 1 | 搅拌功率 | 斜坡+10% | 250-270 |
| 2 | 搅拌功率 | 斜坡 +5% | 160-180 |
| 3 | 通风率 | 斜坡+10% | 160-180 |
| 4 | 通风率 | 斜坡 +5% | 320-340 |

MPCA、KICA 方法和本文方法针对故障 1 的监测效果如图 1、图 2 和图 3 所示. 从图中可知, MPCA 和 KICA 方法发现故障的时间分别是 $t$=263h 和 $t$=262.5h, 均在故障产生之后的 10h 以后, 未能及时发现过程异常情况; 本文方法在 $t$=251h 及时发现了故障, 具有良好的及时性.

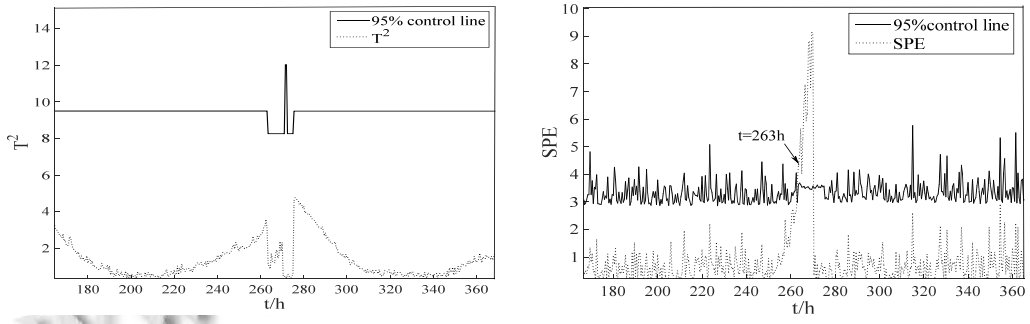


图 1 MPCA 方法

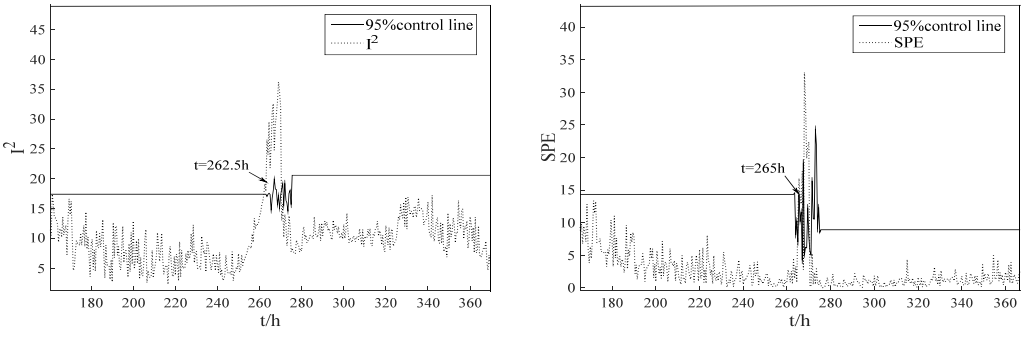


图 2 KICA 方法

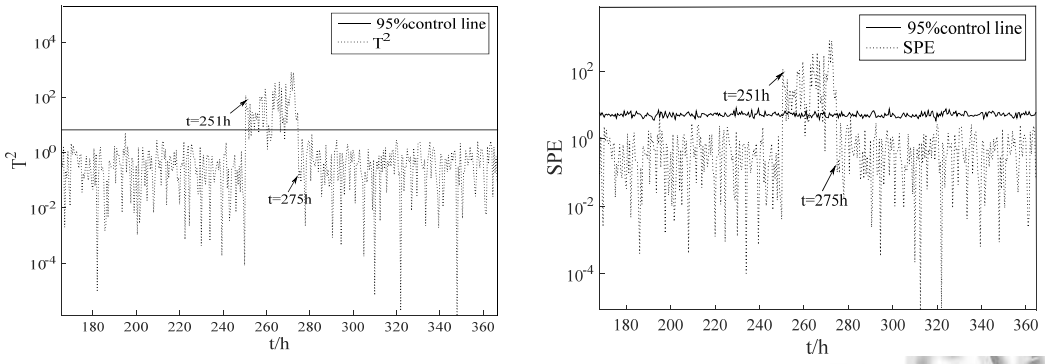


图 3　本文方法

需要注意的是图 3 中按本文的方法 *SPE* 和 $T^2$ 值在 *t*=275h 后恢复到正常范围, 恰好是设置的故障结束后的一个窗宽的时间(*t*=5h), 这是因为此时滑动窗内刚好不再含有故障时刻的历史数据.

使用仿真软件 Pensim2.0 生成 40 个故障批次(故障 1 到故障 4 各 10 个批次), 表 3 为对比 MPCA 方法、KICA 方法与本文提出的方法在不同故障下的平均故障监测率. 从表中可以看出, KICA 方法与 MPCA 方对故障 1、2、3、4 的监测率很低, 而本文的方法平均监测率均在 70%以上, 经过传递熵处理后的间歇过程数据更好地表达了原数据中信息相互作用的特征.

表 3　平均故障监测率

| 故障编号 | FCM MPCA(%) | KICA(%) | the proposed method(%) |
|---|---|---|---|
| 1 | 39.75 | 41 | 83.25 |
| 2 | 20.75 | 18.75 | 80.75 |
| 3 | 46.5 | 51.75 | 79.75 |
| 4 | 25.25 | 29.5 | 72.5 |

## 4　结语

本文提出一种基于传递熵的 MPCA 间歇过程监测方法. 该方法首先利用基于核函数的传递熵方法解决了数据的非高斯分布问题; 然后, 结合滑动窗, 传递熵矩阵能够及时地反映过程的动态信息传递变化, 为后续 MPCA 建模提供很好的数据基础; 最后, 建立 MPCA 监测模型. 通过对比实验表明, 本文所提方法能够及时发现间歇过程异常, 具有较强的监测能力.

## 参考文献

1 杨志才.化工生产中的间歇过程—原理、工艺及设备.北京: 化学工业出版社,2001.

2 Nomikos P, MacGregor JF. Monitoring batch processes using multiway principal component analysis. American Institute of Chemical Engineers Journal, 1994, 40(8): 1361–1375.

3 Lee JM, Yoo CK, Lee IB. Fault detection of batch process using multiway kernel principal component analysis. Computers and Chemical Engineering, 2004, 28(9): 1837–1847.

4 Yoo CK, Lee JM, Vanrolleghem PA, Lee IB. On-line monitoring of batch process using multiway independent component analysis. Chemometrics and Intelligent Laboratory Systems, 2004, 71(2): 151–163.

5 Zhang YW, Qin SJ. Fault detection of nonlinear processes using multiway kernel independent component analysis. Industrial and Engineering Chemistry Research, 2007, 46: 7780–7787.

6 谢晓庆,王建林,赵利强,于涛.基于时段及过渡区域的 KICA 间歇过程监测方法.计算机与应用化学,2014,31(10): 1250–1256.

7 Schreiber T. Measuring information transfer.Physical Review Letters, 2000, 85(2): 461–464.

8 Kaiser A, Schreiber T. Information transfer in continuous processes. Physica D: Nonlinear Phenomena, 2002, 166(1): 43–62.

9 Lizier JT, Prokopenko M, Zomaya AY. A framework for the local information dynamics of distributed computation in complex systems. Guided Self-Organization: Inception, Series Emergence, Complexity and Computation. Springer Berlin Heidelberg. 2014, 9: 115–158.

10 Vicente R, Wibral M, Lindner M, et al. Transfer entropy-a model-free measure of effective connectivity for the neurosciences. Journal of Computational Neuroscience, 2011, 30(1): 45–67.

11 Ito S, Hansen ME, Heiland R, et al. Extending transfer entropy improves identification of effective connectivity in a spiking cortical network model. PloS One, 2011, 6(11): e27431.

12 West SR, Guo Y, Wang XR, et al. Automated fault detection and diagnosis of HVAC subsystems using statistical machine learning.12th International Conference of the International Building Performance Simulation Association. 2011.

13 Bauer M, Cox JW, Caveness MH, Downs JJ, Thornhill NF. Finding the direction of disturbance propagation in a chemical process using transfer entropy. IEEE Trans. on Control Systems Technology, 2007, 15 (1).

14 Duan P, Yang F, Chen T, et al. Direct causality detection via the transfer entropy approach. IEEE Trans. on Control Systems Technology, 2013, 21(6): 2052–2066.

15 Birol G, Ündey C, Cinar A. A modular simulation package for fed-batch fermentation: penicillin production. Computers & Chemical Engineering, 2002, 26(11): 1553–1565.