文章编号: 1002-0411(2001) 02-183-06

多元轨迹同步化问题的改进型 DTW 算法

高 翔[1] 王 纲[2] 赵立杰[2] 马纪虎[1]
(1. 中国科学院沈阳自动化研究所 110015; 2. 沈阳化工学院 110021)

摘 要: 针对间歇生产过程故障诊断中各批次数据轨迹时间长度不一致, 导致应用 MPCA/MPLS 准确建模及故障诊断失效的问题, 本文将动态时间错位(DTW) 算法用于所观测的数据轨迹时间长度的同步化; 并在此基础上进一步提出了递推式和粗格子点两种 DTW 改进算法. 仿真结果表明, 两种 DTW 改进算法能够有效地同步化各批次轨迹, 而且可以推广到在线实时应用. *

关键词: 动态时间错位 动态规划 间歇过程 故障诊断
中图分类号: TP13 文献标识码: B

IMPROVED DTW ALGORITHMS FOR SYNCHRONIZATION OF BATCH TRAJECTORIES

GAO Xiang[1] WANG Gang[2] ZHAO Li-jie[2] MA Ji-hu[1]
(1. *Shenyang Institute of Automation, Chinese Academy of Sciences* 110015;
2. *Shenyang Institute of Chemical Technology* 110021)

Abstract: Modeling and diagnosing using MPCA/MPLS will lose efficiency because of different time length of data trajectories in batch process fault diagnosis. The Dynamic Time Warping (DTW) is used to synchronize the time length of data trajectories observed. Both improved DTW, recursive and coarse-grid points, are proposed. The simulations show that the two methods can not only synchronize data trajectories successfully, but also be applied to on-line fault diagnosis.

Keywords: dynamic time warping, dynamic programming, batch process, fault diagnosis

1 引言 (Introduction)

动态时间错位(Dynamic Time Warping) 是一种灵活的、定常的模式匹配方案, 用于每对模式的匹配; 它能够局部地转移、压缩和扩展局部模式, 使模式内的相似特征能够匹配. 它较多地应用于语音识别领域中间隔词和粘连词的辨识(Myers, C. et al., 1980[1]).

为了能够利用大量易得的现场数据监控间歇过程, 保证产品质量和进行故障诊断, Nomikos and MacGregor (1994)[2] 提出采用多方向主元分析(MPCA) 和多方向潜隐结构投影(MPLS) 方法, 但在他们的方法中一个重要的强假设是所有批次都具有相同的时间长度且同步. 但是在间歇过程(batch process) 中, 由于初始加料时组分的变化, 以及由于季节变化冷却水热交换能力的不同, 都会影响反应速率, 导致各批次的反应时间长度不一致. Lakshminarayanan, S. et al. (1996)[3] 提出将所有批次轨迹都扩至最长, 即简单地在其它短轨迹后人为加上最后点测量值. 此外, 还有按最短时间长度保留各批次轨迹并截断其余数据点的方法等. 但是, 这些方法都忽略了局部模式特征, 因此处理过的轨迹模式不同于原轨迹模式. Anthanassios Kassidas et al. (1998)[4] 提出用 DTW 方法解决轨迹同步化问题, 我们认为这是一种有效的方法.

本文针对间歇生产过程故障诊断中遇到的多元轨迹不同步问题, 深入地对采样点多、反应时间长的间歇过程如何同步化进行了研究, 并且在原有 DTW 方法基础上提出了两种改进的 DTW 算法: 递推式算法解决了同步化时的数据存储难题; 而粗格子点算法实现了快速同步化.

2 DTW 的基本原理(The basic principle of DTW)

设 $T(t\times N)$ 和 $R(r\times N)$ 为两条多元轨迹, t 和 r 为采样次数, N 为变量个数; DTW 运用动态规划

* 收稿日期: 2000-08-13

的原理,非线性地错位两条轨迹,排列相似事件,以获得最短距离.

2.1 对称式算法和非对称式算法(Symmetric algorithm and asymmetric algorithm)

设 i 和 j 分别为 T 和 R 轨迹上的关于时间的坐标,DTW 在 $t\times r$ 网格中建立 K 个点的 F^* 序列:

$$F^* = \{c(1),c(2),\cdots,c(k),\cdots,c(K)\}$$
$$\max(t,r) \leqslant K \leqslant t + r, c(k) = [i(k),j(k)] \tag{1}$$

$c(k)$ 为网格中表示 i 和 j 匹配的每一点,DTW 算法寻出最佳路径,匹配两轨迹的各个向量,以使其间的总距离为最短. 式(1)为对称式算法.

如我们将 R 的采样时间点投影至 T 上,就得到了非对称算法:

$$F^* = \{c(1),c(2),\cdots,c(j),\cdots,c(r)\},$$
$$c(j) = (i(j),j) \tag{2}$$

应用对称式 DTW 算法,可以保留局部特征,但由于同步化后轨迹伸长,故轨迹不能保证预定的时间长度;非对称式 DTW 算法的优点在于可以按时间长度同步化,但由于存在轨迹压缩,同步化后容易丢失数据点. 文[4]采用对称式 DTW 算法搜索出最佳路径之后,对于与参考轨迹中一点对应的待同步化轨迹中的 n 个点,进行向量平均,平均后的 n 个点视为一点. 这样,既解决了时间长度问题,又避免了数据失真.

2.2 局部约束和全局约束(Local constraints and global constraints)

始终点约束:

$$c(1,1) = (1,1), c(k) = (t,r) \tag{3}$$

局部约束:

$$i(k+1) \geqslant i(k), j(k+1) \geqslant j(k) \tag{4}$$

全局约束:

$$M \geqslant |t - r| \tag{5}$$

图 2 示出 Sakoe 和 Chiba(1978)[5] 提出的全局约束,可以避免线性路径的较大偏差.

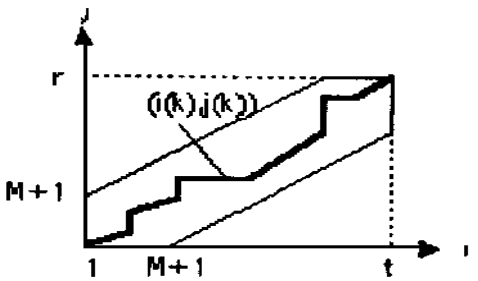


图 1 Sakoe-Chiba 全局约束

Fig. 1 Sakoe-Chiba global constraints

2.3 轨迹间最短距离(The shortest distance between trajectories)

文[5]提出 DTW 算法中,使两轨迹间的距离为最小的公式:

$$D(t,r) = \frac{\sum_{k=1}^{E} d[i(k),j(k)]^* w(k)}{N(w)} \tag{6}$$

其中 $D(t,r)$ 为两轨迹间的标准总距离;$d[i(k),j(k)]$ 为在轨迹 T 的 $i(k)$ 向量和轨迹 R 的 $j(k)$ 向量之间的带加权的局部平方距离.

$$d(i(k),j(k)) = \{T[i(k),:] - R[j(k),:]\}^* W^* \{T[i(k),:] - R[j(k),:]\}^{\mathrm{T}} \tag{7}$$

式(7)中,W 为一正定的权矩阵,反映了各被测变量的相对重要性;一般说来,W 选为对角阵,各 $W_{ii}, i = 1, 2\cdots N$ 的选择依靠过程知识或特殊要求,也可简单地选为单位阵. 文[4]提出一种方法计算量较大,本文提出一种经验公式:将参考轨迹中变量曲线升降次数较多的 W_{ii} 设为零. 其它的 W_{ii} 选取方法为:其变量曲线中 q_i 的初值 $q_{i0} = 1$,以后单调性每变换一次(不计尖峰),$q_i = q_i + 1$.

$$W_{ii} = \frac{\sum_{i=1}^{N} q_i}{q_i} \tag{8}$$

而式(6)中,$N(w)$ 是 $w(k)$ 的函数,为标准化因子. 由文[4],$N(w)$ 为恒值,对称式算法 $N(w)$ 为 $t+r$;而非对称算法为 $r\cdot w(k)$ 为关于 $d(i,j)$ 的权函数,由文[4],式(10)中 $w(k)$ 均为 1.

2.4 运用动态规划求解最佳路径(The solution of the optimal path using dynamic programming)

文[1]提出两条规则:(I)设 F^* 为 $t\times r$ 网格中的最佳全局路径,如 F^* 通过 (i,j) 点,那么通过 (i,j) 点的局部最佳路径必为 F^* 的一部分;(II)通过 (i,j) 点的最佳路径只依赖其前的网格点.

最佳路径有且仅有一条,文中给出了寻优的思路. 设 $D_A(i,j)$ 为从 $(1,1)$ 出发到 (i,j) 的最短累积距离的分子部分:

$$D_A(i,j) = \min_F \sum_{k=1}^{K} d[i(k),j(k)]^* w(k)$$
$$F^* = \arg_F \min[D_A(t,r)] \tag{9}$$

因 $N(w)$ 为一常量,因此只考虑 $D_A(i,j)$,就可计算最短距离.

由文[5]的局部无约束条件,并且由规则(I),点 $(i-1,j)$,$(i-1,j-1)$,$(i,j-1)$ 是至点 (i,j) 最短距离的可能的三个父节点;由规则(II),这最短距离

不会被其后的节点影响. 考虑到本文将要使用的对称式算法和非对称式算法相结合的方法, $D_A(i,j)$ 可由下面的递推公式得到[4]:

$$D_A(i,j) = \min\begin{Bmatrix} D_A(i-1,j) + d(i,j) \\ D_A(i-1,j-1) + d(i,j) \\ D_A(i,j-1) + d(i,j) \end{Bmatrix} \tag{10}$$

$$D_A(1,1) = d(1,1) \tag{11}$$

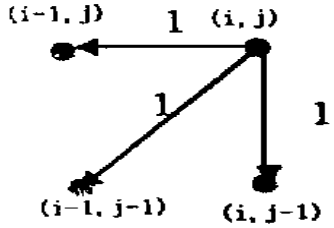


图 2　递推法搜索

Fig. 2　Recursive searching

3　递推式 DTW 算法和粗格子点 DTW 算法 (Recursive DTW algorithm and coarse DTW algorithm)

文[4]用 DTW 算法对工业乳剂聚合过程的 120 个采样点、31 个良好批次进行同步化, 该文选取的采样次数较少. 然而事实上我们在按工业现场现有的采样频率进行数据分析时, 对于采样次数很多、数据矩阵维数激增的情况, 会出现数据存储和运算速度等问题, 影响同步化工作的完成. 本文提出的两种改进 DTW 算法很好地解决了这类问题.

3.1　递推式算法(The recursive DTW algorithm)

在同步化工作中, 由于 $D_A(i,j)$ 的计算是反复地进行的, 只有将 $D_A(i,j)$ 的可能前点的值 $D_A(i-1,j)$、$D_A(i-1,j-1)$ 和 $D_A(i,j-1)$ 存储起来, 才能完成计算. 其步骤为: 在正向计算 $D_A(i,j)$ 后, 反向应用式(10), 寻找这三个点中哪一个是点 (i,j) 的前点, 并记录下来, 共有式(1)中 K 种匹配模式对应关系. 才能对待同步化的轨迹进行压缩、扩展和平移, 这也需要记忆点 (i,j) 的三个可能前点值. 如果逐个存储并正向计算每一个 $D_A(i,j)$ 值的结果, 形成一个距离矩阵 $D_A(t\times r)$, 再一起反向寻优, 如果运算过程中采样点多, 数据矩阵的维数会激增, 导致计算 $D_A(t\times r)$ 阵维数巨大, 运算将面临存储空间危机.

本文针对这种情况, 给出了解决该类问题的递推式算法: 由于 $D_A(i,j)$ 的计算只依赖其前的三个点, 所以每次只需计算 $D_A(t\times r)$ 的一行(算法中均考虑式(5), 以减少计算量):

(1) $i=1$, 首先计算 $D_A(i-1,:)$, $IP=1$, $JP=1$;

(2) $i\leftarrow(i+1)$, 利用 $D_A(i-1,:)$ 的结果, 计算 $D_A(i,:)$;

(3) 在 $D_A(i-1,:)$ 和 $D_A(i,:)$ 带状窗口内即时寻优, 始点为 (IP,JP), 终点为 (IE,JE). 其中, $IE=IP+1$, JE 的确定需满足:

$$D_A(IE,JE) = \min[D_A(IE,JP), D_A(IE,JP+1)\cdots D_A(IE,r)] \tag{12}$$

找出最佳路径, 并记录各坐标点.

(4) 删除行 $D_A(i-1,:)$, $IP\leftarrow IE$, $JP\leftarrow JE$.

(5) 重复(2)到(4)的步骤, 直至 $i=t$.

(6) $i=t$ 时, 路径终点若为 (t,r), 搜索结束; 否则, 路径终点若为 (t,p), 且 $p<r$, 则 (t,p) 至 (t,r) 的直线上各点均为最佳路径上的点, 搜索结束.

上述方法实质上采用的是递推分段寻优, 由(4)可知, 每一个窗口终点的纵坐标 JE 不小于其始点的纵坐标 JP, $D_A(IE,JE)$ 是点 (IE,JE) 到点(1,1)累积最短距离, 而 $D_A(IP,JP)$ 是点 (IP,JP) 到点(1,1)的累积最短距离. 因此, 由规则(I), 从点 (IP,JP) 至点 (IE,JE) 的路径必为 F^* 的一部分; 且对于每一窗口, 终点有且仅有一个, 这是符合最优路径唯一性的, 同一纵坐标下路径的其它点属于下一窗口(如图 3).

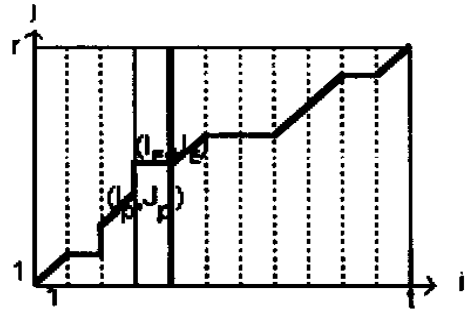


图 3　递推式算法示意图

Fig. 3　The graph of recursive algorithm

这样, 计算过程中只需要存储两行 $(2\times r)$ 个 D_A 值, 不需要存储 $t\times r$ 个 D_A 值, 递推算法节省了存储空间.

在轨迹同步化时结合非对称算法使轨迹时间长度为 r. 设同步化之前轨迹为 T, 同步化后为 NT, 若 F^* 的各点横坐标对纵坐标的映射是一一映射, 则有:

$$NT(j,:) = T(i,:) \tag{13}$$

否则, 若有 n 个点的纵坐标相同, 那么:

$$NT(j,:) = \frac{1}{n}\sum_{\eta=i}^{i+n-1} T(\eta,:) \qquad (14)$$

3.2 粗格子点算法 (The coarse algorithm)

对于采样频率已确定、且采样点多,数据矩阵维数较大的情况,为了降低计算的复杂度,加快运算速度,运用动态规划原理有关 $t\times r$ 网格路径优化达的粗格子点法能够有效地解决数据矩阵维数激增情况下多元轨迹的同步化问题. 该方法适应于反应时间长、变化缓慢的间歇过程.

将参考轨迹 R 和待同步化轨迹 T 的时间长度平均分成 λ 等份,粗格子点划分后 R_{CR} 和 T_{CR} 为原轨迹每等份中各点的平均值;对于不能整除的情况,余数 α 也可作为一份. 对 R_{CR} 和 T_{CR} 同步化(其对称式算法可正向计算距离矩阵 $D_A(t\times r)$,再一起反向寻优,非对称算法如式(13)、(14)),得到同步化后的轨迹 NT_{CR},再将其插值拟合为 NT.

在插值拟合的过程中,不能只考虑将轨迹 NT_{CR} 的插值,这是因为考虑到 NT 的始末要和 T 对齐,而不是与 T_{CR} 对齐. 由于同时考虑到插值拟合数据点的疏密程度的不同,这样轨迹 NT 的插值分成三段:初始段 NT_A、中段 NT_B 和末段 NT_C.

(1) NT_A 的确定:

$$NT_A(1,:) = T(1,:)$$
$$NT_A(i+1,:) = T(1,:) + NT_{CR}(1,:) - T(1,:)*2/\lambda*i$$
$$i\in(1,\lambda/2-1) \text{ 且为正整数} \qquad (15)$$

(2) NT_B 为:

(i) 在没有余数的情况下,将 NT_{CR} 的各点,进行非线性样条插值;

(ii) 若有余数 α,上述 NT_{CR} 各点(不包括余数的那一点)非线性插值后构成 NT_{B1},余数 α 的那一份线性插入($\lambda+\alpha$)/2−1 的整数个点构成 NT_{B2};

$$NT_{B2}(i,:) = NT_{CR}(r_{CR},:) + i^*2/(\lambda+\alpha) * [NT_{CR}(r_{CR},:) - NT_{CR}(r_{CR}-1,:)]$$
$$i\in(1,\lambda/2+\alpha/2-1) \text{ 且为正整数} \qquad (16)$$

其中,r_{CR} 为粗格子点后数据矩阵的维数. 这样,情况(ii)时,NT_B 由两部分构成:NT_{B1} 和 NT_{B2}.

(3) NT_C 的确定:

(i) 若无余数:

$$NT_C(i,:) = NT_{CR}(r_{CR},:) + i^*2/\lambda* [T(t,:) - NT_{CR}(r_{CR},:)]$$
$$i\in(1,\lambda/2-1) \text{ 且为正整数} \qquad (17)$$

(ii) 若有余数 α:

$$NT_C(i,:) = NT_{CR}(r_{CR},:) + i^*2/\alpha* [T(t,:) - NT_{CR}(r_{CR},:)]$$
$$i\in(1,\alpha/2-1) \text{ 且为正整数} \qquad (18)$$

其中,t 为待同步化矩阵 T 的维数.

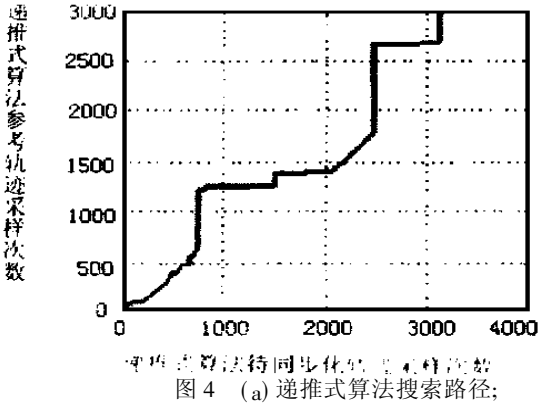
4 仿真实例(The example of simulation)

我们对某大型化工厂实际 PVC 聚合反应过程,采用本文提出的改进型 DTW 算法,进行了同步化研究;通过多次测量,参考轨迹的采样时间长度定为 3000 次. 本文对其中长度为 3160 次的轨迹进行了同步化仿真研究.

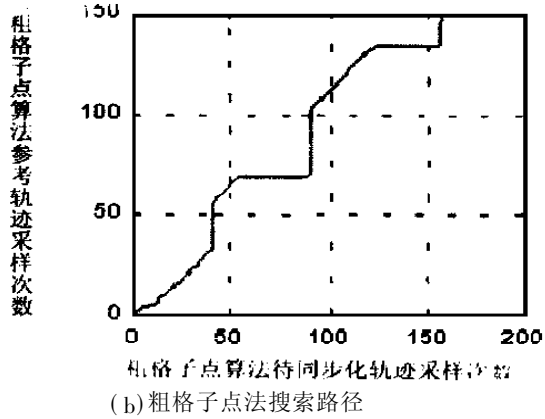
首先,由于轨迹中各变量的工程单位不尽相同,所以在同步化前有必要将所测数据矩阵 X 量化成均值为零,方差为 1 的矩阵.

$$x_i = \frac{x_i - E(x_i)}{Var(x_i)} \qquad (19)$$

x_i 为 X 的采样时刻为 i 的包含各变量值的行向量,$E(x_i)$ 为各变量均值的行向量,$Var(x_i)$ 为其方差.



(a) 递推式算法搜索路径;

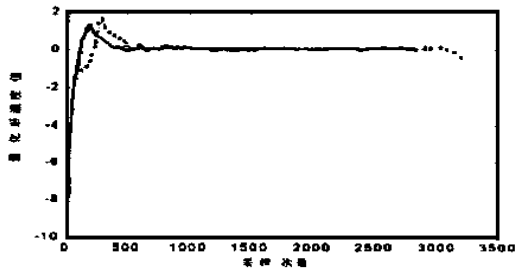


(b) 粗格子点法搜索路径

Fig. 4 (a) The searching path of recursive algorithm (b) The searching path of coarse algorithm

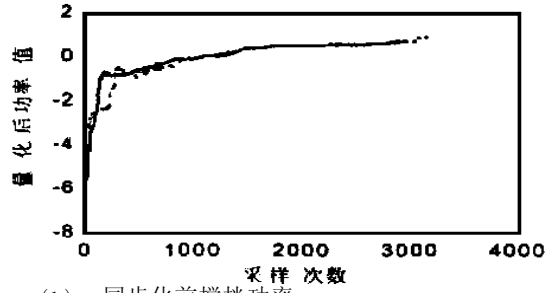
由式(5),设 $M = \hat{u}_t - r\hat{u} + m_0$,$m_0$ 需按具体情况人为选定. 在递推式算法中,$m_0 = 400$;而在粗格子

点算法中, m_0= 20. 从图 4 中可以看出, 搜索路径不相似, 这是数据处理方法不相同的缘故. 图 5 给出 10 个变量中的釜内温度和搅拌功率的用递推式算法同步化曲线结果. 图 6 则对应地给出了用粗格子点算法同步化曲线结果. 图中黑色线为参考轨迹, 绿色线为待同步化轨迹和同步化后轨迹. 仿真结果表明, 两种方法都能很好地同步化批次轨迹. 我们在 CPU 为 PII450, 内存为 64M 的计算机上进行仿真, 可以见到: 递推式算法同步该轨迹用时 12 分钟, 其它轨迹最多用时 40 多分钟; 而用粗格子点算法不超过一分钟, 证明粗格子点算法能够快速同步化.



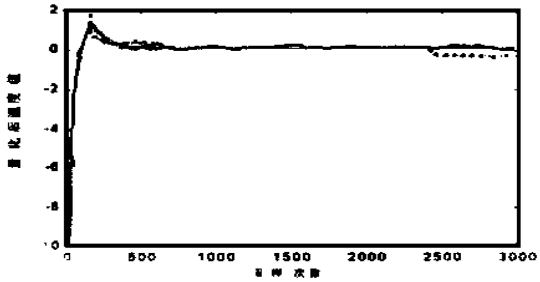
(a)　同步化前釜内温度

The rector temperature before synchronization



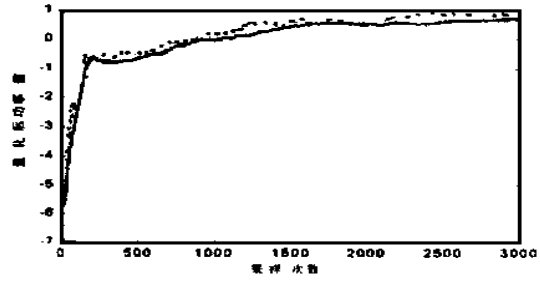
(b)　同步化前搅拌功率

The stirred power before synchronization



(c)　同步化后釜内温度

The rector temperature after synchronization

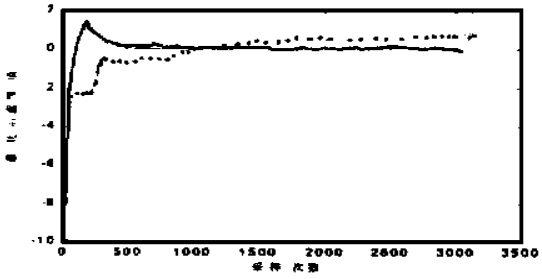


(d)　同步化后搅拌功率

The stirred power after synchronization

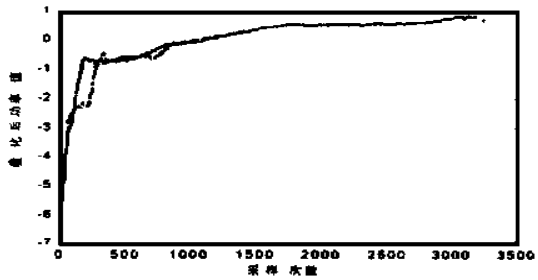
图 5　递推式算法同步化前后比较

Fig. 5　The comparison of DTW in recursive algorithm



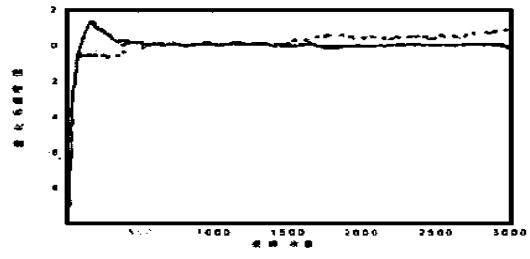
(a)　同步化前釜内温度

The rector temperature before synchronization



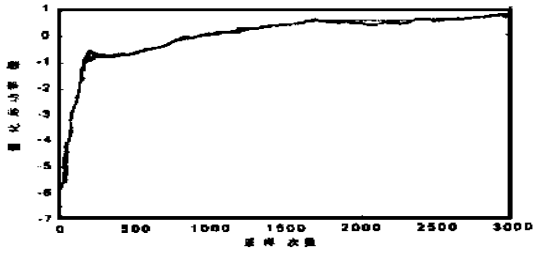
(b)　同步化前搅拌功率

The stirred power before synchronization



(c)　同步化后釜内温度

The rector temperature after synchronization



(d)　同步化后搅拌功率

The stirred power after synchronization

图 6　粗格子点算法同步化前后比较

Fig. 6　The comparison of DTW in coarse algorithm

5 结论(Conclusion)

运用DTW对多元轨迹同步化,为进一步准确地应用MPCA/MPLS方法建模和故障诊断提供了条件.用递推式算法和粗格子点算法能够对采样点多、数据矩阵维数大的情况实施同步化.递推式算法能够解决在该种情况下数据存储的难题,本文的想法是在两采样点之间的窗口进行寻优,即边计算,边寻优,而不是计算完后整体反向寻优,并且给出了判断窗口内最优路径的终点的方法;而粗格子点算法在保留了轨迹局部特征的同时,降低了矩阵的维数,从而节约了同步化时间.本文在将参考轨迹和待同步化轨迹平均分成λ等份的同时,考虑了有余数的情况,用每一等份的均值进行粗格子点法的同步化.在对同步化后粗格子点轨迹插值拟合时,考虑了与原待同步化轨迹首尾数据对齐,同时算法中对有无余数也作了讨论.我们认为本文方法不仅可以有效解决间歇过程多元轨迹同步化问题,从而提高建模和诊断的准确度,且可以推广到实时在线应用.

参 考 文 献(Reference)

1 Myers C, *et al*. Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition. IEEE Trans. on Acoustics, speech and Signal Processing, ASSP1984, **32**(2): 263

2 Nomikos P, J F MacGregor. Monitoring Batch Processes Using Multiway Principal Component Analysis. AIChE J. 1994, 40, 1361

3 Lakshminarayanan S, *et al*. Monitoring Batch Processes using Multivariate Statistical Tools: Extensions and Practical Issues. IFAC Triennial World Congress, San Francisco, 1996

4 Athanassios Kassidas, *et al*. Synchronization of Batch Trajectories Using Dynamic Time Warping. AIChE J. 1998, 44, 864

5 Sakoe H, S. Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition, IEEE Trans. On Acoustics, Speech and Signal Process. ASSP1978, **26**(1), 43

作者简介

高 翔(1967–),男,博士生.研究领域为故障诊断、动态规划、统计过程控制等.

王 纲(1956–),男,教授.研究领域为复杂工业过程建模控制和优化、故障诊断、统计过程控制等.

(上接第178页)

表**1** 续表 算法的迭代过程

变量 / 步骤	x1 G1(x1)	x2 G2(x2)	x3 G3(x3)	x4 G4(x4)	x5 G5(x5)	x6 G6(x6)	x7 G7(x7)	x8 G8(x8)	x9 G9(x9)	x10 G10(x10)	约束G(X)
x10	2 0.998	2 0.995	2 0.978	2 0.96	3 0.984	2 0.996	2 0.990	2 0.998	1 0.98	2 0.986	0.872
x11	2 0.998	2 0.995	2 0.978	3 0.992	3 0.984	2 0.996	2 0.990	2 0.998	1 0.98	2 0.986	0.901
x12	2 0.998	2 0.995	2 0.978	3 0.992	3 0.984	2 0.996	2 0.990	2 0.998	2 0.999	2 0.986	0.919
x13	2 0.998	2 0.995	3 0.997	3 0.992	3 0.984	2 0.996	2 0.990	2 0.998	2 0.999	2 0.986	0.937
x14	2 0.998	2 0.995	3 0.997	3 0.992	3 0.984	2 0.996	2 0.990	2 0.998	2 0.999	3 0.998	0.948
x15	2 0.998	2 0.995	3 0.997	3 0.992	4 0.996	2 0.996	2 0.990	2 0.998	2 0.999	3 0.998	0.961
x16	2 0.998	2 0.995	3 0.997	3 0.992	4 0.996	2 0.996	3 0.999	2 0.998	2 0.999	3 0.998	0.969
x17	2 0.998	2 0.995	3 0.997	4 0.998	4 0.996	2 0.996	3 0.999	2 0998	2 0.999	3 0.998	0.973
x18	2 0.998	3 0.999	3 0.997	4 0.998	4 0.996	2 0.996	3 0.999	2 0.998	2 0.999	3 0.998	0.980

参 考 文 献(References)

1 曹晋华,程侃,1985,可靠性数学引论,北京科学出版社

作者简介

赵中奇(1957–),博士,副教授.研究领域为控制科学与控制工程.

潘德惠(1928–),博士导师.研究领域为管理科学与工程,控制科学与工程.