情報処理学会論文誌 Vol.57 No.4 1303–1318 (Apr. 2016)

# 時系列データにおけるレグ振動解析

今村 誠[1] 中村 隆顕[1,a)] 柴田 秀哉[1] 平井 規郎[1] 北上 眞二[2] 撫中 達司[3]

受付日 2015年7月3日, 採録日 2016年1月12日

**概要**：ビル，プラント，工場などの設備保全業務では，故障兆候や劣化を検出するために，与えられた振幅値とウインドウサイズに対して，その振幅を超える上下変動の頻度を算出したいというニーズがある．Fink らは，このニーズに応える技術として，雑音など変動幅の小さな局所的な上下変動を含む時系列における大域的な上下変動を検索するレグ検索を提案したが，上昇/下降などの単独のレグを扱うにとどまっていた．本論文では，従来のような単独のレグだけでなく，上昇レグと下降レグが交代出現する頻度を表現するレグ振動数を定式化するとともに，高速にレグ振動数を求めるアルゴリズムを提案する．提案アルゴリズムの計算オーダは，時系列データの長さを $n$，ウインドウサイズを $w$ とするとき，ナイーブな手法の計算オーダが $O(n(w!)^2)$ であるのに対して，$O(nw)$ になる．次いで，設備保全の現場への適用評価に提案手法の有用性を示す．さらに，特性の異なる 6 種類のデータを用いて，ウインドウサイズと振幅へのレグ振動解析処理時間の依存性を評価することにより，現実的な処理性能を持つことを示す．

**キーワード**：時系列，状態監視保全，異常検知，特徴抽出，振動解析

# Leg Vibration Analysis for Time Series

Makoto Imamura[1] Takaaki Nakamura[1,a)] Hideya Shibata[1] Norio Hirai[1]
Shinji Kitagami[2] Tatsuji Munaka[3]

Received: July 3, 2015, Accepted: January 12, 2016

***Abstract:*** The calculation of the frequency of variations in sensor data is necessary in order to detect a sign of failure or deterioration for facility maintenance in a building, a plant or a factory needs to. Fink et al. proposed a leg search method to find a global trend in time-series including small variations such as noise. However, their method treats only single leg so that it can find an upward or downward trend but can't calculate the frequency of variations. In this paper, we propose leg vibration analysis that can calculate the frequency of variations in time-series that include an upward trend and a downward trend appear alternately and iteratively. We show an algorithm whose calculation order is linear to window size. In contrast, the calculation order of a naive algorithm is the factorials of the square of window size. We also show that our proposed method has usefulness and practical time performance by its application to real facility management business and an evaluation of the dependency of execution time on window size and amplitude with 6 data that have different behavior.

***Keywords:*** time series, condition-based maintenance, anomaly detection, feature extraction, vibration analysis

[1] 三菱電機株式会社情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation, Kamakura, Kanagawa 247–8501, Japan
[2] 三菱電機ビルテクノサービス株式会社
Mitsubishi Electric Building Techno-Service Co., Ltd., Arakawa, Tokyo 116–0002, Japan
[3] 東海大学情報通信学部
School of Information and Telecommunication Engineering, Tokai University, Minato, Tokyo 108–8619, Japan
[a)] Nakamura.Takaaki@dy.MitsubishiElectric.co.jp

## 1. はじめに

ビル，プラント，工場などの設備保全業務では，故障兆候や劣化を検出するために，不規則に変動するセンサ時系列に対して，与えられた振幅を超える上下変動の頻度を算出したいというニーズがある．

不規則に変動する時系列の上下変動を扱った従来技術と

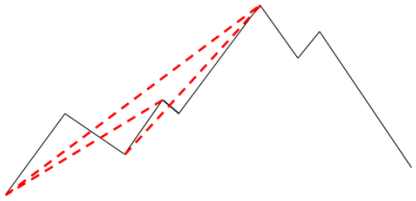


図 1 レグの例
**Fig. 1** Examples of leg.

しては，データ工学分野では，Fink ら [1] が提案したレグがある．レグは，**図 1** の点線に示すように，不規則な上下変動があっても全体として上昇，ないし下降している部分列のことをいう．

また，材料強度力学の分野では，不規則に変動する時系列データから，疲労や亀裂の発生に寄与するひずみの振幅とその繰返し数を求めるレインフロー法 [2], [3], [4] が提案されている．3 章でより詳細に説明するが，レインフロー法は，レグと同等の概念を扱っている．

Fink やレインフロー法が，個々のレグをそれぞれ独立に扱っている，あるいは，直前のレグとの関係のみを扱っているのに対して，本論文では，指定した振幅と時間内に上昇下降するレグが交互に出現する数を解析することを目標とする．そして，ある一定以上の振幅を持つ上下変動の振動的な振舞いを表現する新たな指標としてレグ振動数を定義し，レグ振動数を計算するアルゴリズムを提案する．

データ工学分野では，時系列のパターンを記述し，その記述を満たすパターンを検索できるようにするストリーム処理 [5], [6] が提案されている．ストリーム処理は，レグ振動数を計算する手段として利用できる可能性があるので，ストリーム処理との関連性についても考察する．

本論文の構成は，以下のとおりである．2 章で，レグ振動数を定義する．3 章で，本論文で扱う応用課題，技術課題，および，従来技術との差異を述べる．4 章では，レグ振動数を計算するアルゴリズムについて述べる．5 章では，設備保全の現場への適用評価により，提案手法の実用性を示す．さらに，6 種類の特性の異なるデータを対象として，レグ振動数算出の速度性能を評価することにより，提案手法が実用的な速度性能を持つことを示す．6 章では，ストリーム処理との関連性を考察し，7 章でまとめを述べる．

## 2. レグ振動数

本章では，不規則に変動する時系列のレグ振動数を定義する．最初に，1 つの上下変動を表現する部分列としてレグを定義する．次に，振幅と振動数を定義するための基本概念として，レグ振動列を定義する．そして，レグ振動列を用いて，与えられた振幅 $a$ に対する部分列のレグ振動数を定義する．最後に，与えられたウインドウサイズ $w$ と振幅 $a$ に対するレグ振動数 $F$ を定義する．

### 2.1 レグ

【定義 1】時系列 $X$

$m$ 個の実数の順序付けられた列（リスト）$X = [x_1, \cdots, x_m]$ を，時系列と呼ぶ．時系列 $X$ の $i$ 番目の値 $x_i$ を $X[i]$ と記す．添え字 $i$ は，1 から $m$ の整数値をとる変数である．添え字 $i$ を時点と呼ぶ．また，$m$ を時系列の長さ $\mathrm{length}(X)$ と呼ぶ．

【定義 2】部分列 $l$

時系列 $X$ の $p$ 番目から $q$ 番目の値を抽出して得られるリスト $l = [x_p, x_{p+1}, \ldots, x_q]$ を $X$ の部分列と呼ぶ．部分列 $l$ の開始時点 $p$ を $\mathrm{start}(l)$ と記載し，終了時点 $q$ を $\mathrm{end}(l)$ と記載する．また，部分列の開始時点と終了時点を明示したい場合には，部分列 $l$ を $X[p{:}\,q]$ と記載する．

図 1 の点線に示す部分列のように，すべての時点の値が開始時点の値よりも大きいが終了時点の値より小さく，かつ，開始時点と終了時点が時系列における極値になっている場合，その部分列を上昇レグと呼ぶ．

【定義 3】上昇レグ $l$，下降レグ $l$，レグ $l$

(1) 上昇レグ $l$

部分列 $l = X[p{:}\,q]$ が，以下の 3 つの条件を満たすならば，$l$ を上昇レグと呼ぶ．ただし，$X[p-1]$ や $X[q+1]$ が存在しない場合は，(2)，(3) は条件に含めない．

$$\forall i.\ p \le i \le q \quad X[p] < X[i] < X[q] \tag{1}$$
$$X[p-1] \ge X[p] \tag{2}$$
$$X[q] \ge X[q+1] \tag{3}$$

(2) 下降レグ $l$

同様に，部分列 $l = X[p{:}\,q]$ が，以下の 3 つの条件を満たすならば，$l$ を下降レグと呼ぶ．

$$\forall i.\ p \le i \le q \quad X[p] > X[i] > X[q] \tag{4}$$
$$X[p-1] \le X[p] \tag{5}$$
$$X[q] \le X[q+1] \tag{6}$$

(3) レグ $l$

部分列 $l$ が，上昇レグ，または，下降レグならば，$l$ をレグと呼ぶ．レグは部分列でもあるので，部分列と同様に，start と end を定義できる．

Fink は，単調増加，または単調減少する部分列をレグと呼び，上記で定義したレグを拡張レグと呼んでいるが，本論文では拡張レグしか用いないので，表記を簡潔にするために，単にレグと呼ぶ．

【定義 4】レグの振幅 amp，符号 sign

$l = X[p{:}\,q]$ をレグとする．レグの振幅 amp とレグの符号 sign を以下で定義する．

$$\mathrm{amp}(l) \equiv \mathrm{abs}(X[q] - X[p])$$
$$\mathrm{sign}(l) \equiv \mathrm{sign}(X[q] - X[p])$$

### 2.2 レグ振動列の振幅と振動数

【定義 5】レグ振動列 $s$

$l_1, l_2, \ldots, l_n$ をレグ，$a$ を正の実数とする．以下の 3 条件を満たすとき，レグの系列 $s = [l_1, l_2, \ldots, l_n]$ を振幅 $a$ のレグ振動列と呼ぶ．

$$1 \le i \le n-1 \text{ に対して，} \mathrm{end}(l_i) \le \mathrm{start}(l_{i+1}) \tag{7}$$

$$1 \le i \le n \text{ に対して，} \mathrm{amp}(l_i) \ge a \tag{8}$$

$$\mathrm{sign}(l_i) \times \mathrm{sign}(l_{i+1}) < 0 \tag{9}$$

すなわち，レグの振幅がある値より大きく，その符号が異なるレグが交互に並んでいるレグの系列を，レグ振動列と呼ぶ．また，レグ振動列の符号 sign，開始時点 start，終了時点 end，末尾レグ last，長さ length を，レグ振動列の先頭レグ $l_1$ と末端レグ $l_n$ を用いて以下のように定義する．

$$\mathrm{sign}(s) \equiv \mathrm{sign}(l_1)$$

$$\mathrm{start}(s) \equiv \mathrm{start}(l_1)$$

$$\mathrm{end}(s) \equiv \mathrm{end}(l_n)$$

$$\mathrm{last}(s) \equiv l_n$$

$$\mathrm{length}(s) \equiv n$$

【定義 6】レグ振動列 $s$ の振動数 $F$，振幅 amp

$s = [l_1, l_2, \ldots, l_n]$ をレグ振動列とするとき，レグ振動列 $s$ の振動数 $F$ と振幅 amp を以下で定義する．

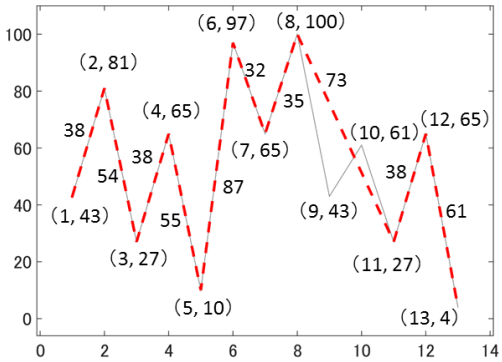
$$F(s) \equiv \mathrm{sign}(s) \times \mathrm{length}(s) \tag{10}$$

$$\mathrm{amp}(s) \equiv \min_i \mathrm{amp}(l_i) \quad \text{ただし，} 1 \le i \le n \tag{11}$$

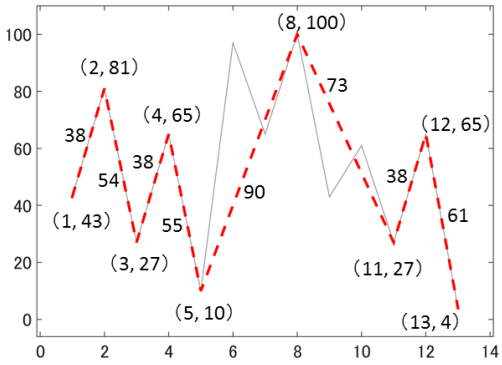
**図 2** に，レグ振動列の例を示す．図中の (a) から (d) では，実線で 13 の時点からなる部分列を示し，点線でレグ振動列（点線）を示す．また，(a) において，(1, 43) や (2, 81) は，部分列の時点と値の対を示している．(1, 43) と (2, 81) を結ぶ点線の部分がレグであり，点線の肩にレグの振幅の値 38 を示す．レグ振動列 (a) は，10 個のレグからなり，上昇レグから始まっているので，符号は +，レグ振動列 (a) の長さは 10，よって，振動数は 10 である．また，レグ振動列 (a) を構成するレグの振幅は，順に，38，54，38，55，87，32，35，73，38，61 なので，これらの最小値である 32 がレグ振動列 (a) の振幅になる．同様に，レグ振動列 (b)，(c)，(d) の長さは，順に，8，3，2 であり，符号は，順に +，−，+ なので，レグ振動数は，順に，8，−3，2 となる．また，振幅は，順に，38，71，90 になる．

### 2.3 振幅 $a$ に対する部分列のレグ振動数

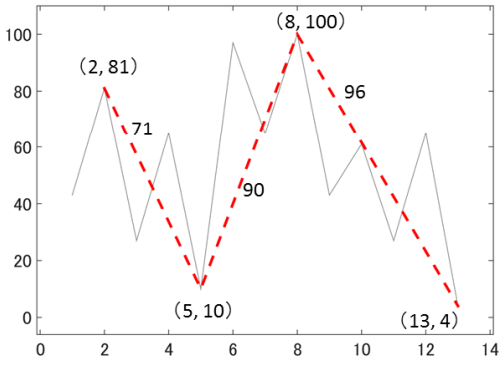
部分列には，振幅や振動数が異なる複数のレグ振動列が存在するので，部分列のレグ振動列を定義するためには，複数のレグ振動列を扱うしくみが必要になる．レグの上下変動の頻度は，注目する振幅 $a$ に応じて異なるので，以下では，「部分列に含まれる振幅 $a$ 以上のレグ振動列の集合」という概念を用いて，部分列のレグ振動数を定義する．



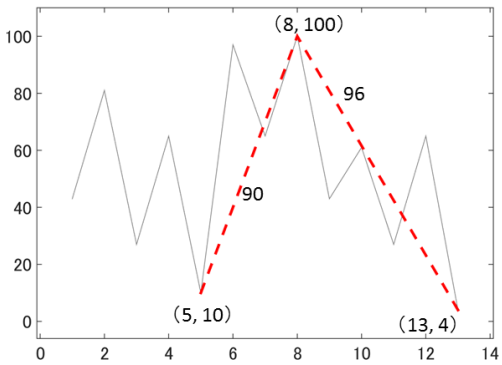

(a) 振動数10，振幅32のレグ振動列




(b) 振動数8，振幅38のレグ振動列




(c) 振動数−3，振幅71のレグ振動列




(d) 振動数2，振幅90のレグ振動列

図 2 レグ振動列の例

**Fig. 2** Examples of leg vibration sequence.

【定義 7】振幅 $a$ に対する部分列のレグ振動数
$X[p{:}\,q]$ を部分列，$a$ を振幅とするとき，部分列のレグ振動数 $F_a(X[p{:}\,q])$ を以下で定義する．

$$F_a(X[p{:}\,q]) \equiv \mathrm{sign}(s_{\max}) \times \mathrm{length}(s_{\max})$$
$$\text{such that } s_{\max} = \mathrm{argmax}_{s \in S(X[p:q],a)}\, \mathrm{length}(s) \quad (12)$$

ただし，$S(X[p{:}\,q], a)$ は，部分列 $X[p{:}\,q]$ に含まれる振幅 $a$ 以上のレグ振動列 $s$ の集合とする．また，$\mathrm{argmax}_{s \in S(X[p:q],a)}$ $\mathrm{length}(s)$ は，「集合 $S(X[p{:}\,q], a)$ の要素であるレグ振動列の中で，長さ length が最大のレグ振動列」を意味する．

**図 3** を用いて，「集合 $S(X[p{:}\,q], a)$ における長さが最大のレグ振動列」の直観的な意味を説明する．図 3 の (a)，(b)，(c)，(d) は，図 2 で示した部分列に含まれる振幅が 38 のレグ振動列の例である．レグ振動列 (a)，(b)，(c)，(d) は，振幅は同じだが，長さは，順に，8，8，6，6 と異なる．この中で，長さが最大なものは 8 なので，レグ振動列 (a) と (b) が最大の長さを持つ振動列になる．定義 7 は，レグ振動列 (a)，(b)，(c)，(d) を「部分列に含まれる振幅 38 以上のすべてのレグ振動列の集合」に置き換えて，同様の操作をすることを意味する．

図 3 の例でみたように，長さが最大のレグ振動列は複数存在する可能性があるので，符号付きのレグ振動数を定義できることを示すには，長さが最大のレグ振動列の符号は互いに等しいことを示す必要がある．

【補題：最長レグ振動列の符号の同一性】レグ振動列集合 $S(X[p{:}\,q], a)$ において最大の長さを持つレグ振動列は，互いに同符号である．

【証明】「レグ振動列 $s = [l_1, l_2, \ldots, l_n]$，$u = [m_1, m_2, \ldots, m_n]$ をレグ振動列集合 $S(X[p{:}\,q], a)$ における最大の長さ $n$ を持ち，かつ，符号が異なる」と仮定すると矛盾することを示す．ここで $s$ の符号を正，$u$ の符号を負としても一般性を失わない．$s$ の符号は正なので，$l_1$ は上昇レグであり，$u$ の符号は負であるから，$m_1$ は下降レグである．

$s$ の先頭レグ $l_1$ と $u$ の先頭レグ $m_1$ の時区間 $[\mathrm{start}(l_1), \mathrm{end}(l_1)]$ と $[\mathrm{start}(m_1), \mathrm{end}(m_1)]$ は，「交差する」，「一方が他方に含まれる」，そして，「一方の後に他方が出現する」のいずれかが成立する．以下，いずれの場合も矛盾することを示す．

最初に，「$l_1$ と $m_1$ が交差する場合」は，条件 1-1「$\mathrm{start}(l_1) < \mathrm{start}(m_1) < \mathrm{end}(l_1) < \mathrm{end}(m_1)$」，または，条件 1-2「$\mathrm{start}(m_1) < \mathrm{start}(l_1) < \mathrm{end}(m_1) < \mathrm{end}(l_1)$」が成立する．条件 1-1 が成立する場合は，$l_1$ は上昇レグなので，$X[\mathrm{start}(m_1)] < X[\mathrm{end}(l_1)]$ が成立する．一方，$m_1$ は下降レグなので，$X[\mathrm{start}(m_1)] > X[\mathrm{end}(l_1)]$ が成立する．これは，矛盾である．条件 1-2 の場合も同様に矛盾を導くことができる．

次に，「レグ $l_1$ と $m_1$ において，一方が他方に含まれる場合」は，条件 2-1「$\mathrm{start}(l_1) < \mathrm{start}(m_1) < \mathrm{end}(m_1) < \mathrm{end}(l_1)$」，

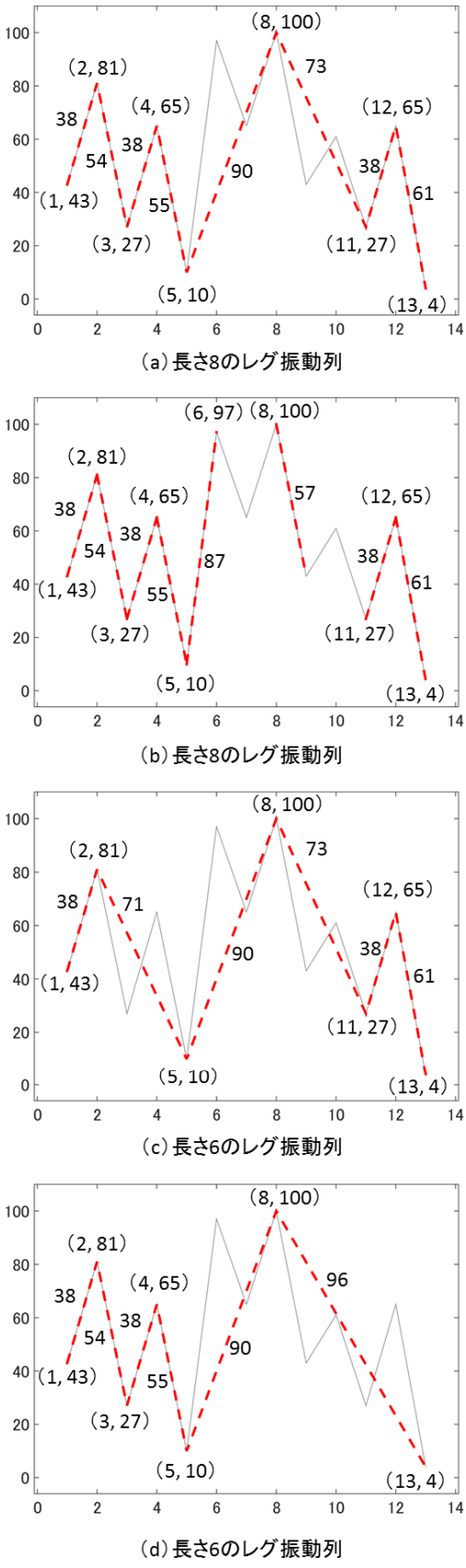



**図 3** 同じ振幅で長さが異なるレグ振動列
**Fig. 3** Leg vibration sequences whose length are different but whose amplitude are equal.

または，条件 2-2「$\mathrm{start}(m_1) < \mathrm{start}(l_1) < \mathrm{end}(l_1) < \mathrm{end}(m_1)$」が成立する．上記で，レグの定義（定義 3）から端点は極値であるから，「上昇レグの開始時点は下降レグの開始時点にならない．また，逆も成立する」，「下降レグの終了時点は上昇レグの終了時点にならない．また，逆も成立する」ので，不等号記号は，$\leq$ でなく $<$ になることを用いている．

条件 2-1 の場合は，$\mathrm{amp}(m_1) \geq a$ であることから，$l_{1-1} = X[\mathrm{start}(l_1){:}\,\mathrm{start}(m_1)]$ と $l_{1-2} = X[\mathrm{end}(m_1){:}\,\mathrm{end}(l_1)]$ は，振幅 $a$ 以上の上昇レグになる．したがって，$[l_{1-1}, m_1, l_{1-2}, l_2, \ldots, l_n]$ は，長さ $n+2$ の振幅 $a$ 以上のレグ振動列となるが，$s$ と $u$ が最大の長さ $n$ を持つことに矛盾する．同様に，条件 2-2 の場合も矛盾を導くことができる．

最後に，「レグ $l_1$ と $m_1$ において，一方の出現後に，もう一方が出現する場合」は，条件 3-1「$\mathrm{end}(l_1) \leq \mathrm{start}(m_1)$」，または，条件 3-2「$\mathrm{end}(m_1) \leq \mathrm{start}(l_1)$」が成立する．条件 3-1 を満たす場合は，$[l_1, m_1, m_2, \ldots, m_n]$ は，長さ $n+1$ のレグ振動列となるので，$s$ と $u$ が最大の長さ $n$ を持つことに矛盾する．同様に，条件 3-2 の場合も矛盾を導くことができる．

以上のとおり，レグ $l_1$ とレグ $m_1$ が異符号と仮定すると矛盾するので，レグ $l_1$ とレグ $m_1$ は同符号でなければならない．したがって，レグ振動列の符号の定義から，レグ振動列 $s$ と $u$ は，同符号である． □

### 2.4 ウインドウサイズ $w$，振幅 $a$ に対する時系列のレグ振動数

2.3 節により，部分列に対するレグ振動列を定義することができた．したがって，ウインドウサイズ $w$ が与えられれば，時点 $t$ ごとに部分列 $X[t{:}\,t+w-1]$ が決まることに注意すれば，時系列 $X$ に対するレグ振動数 $F_{X,a,w}(t)$ を以下のように定義できる．時点 $t$ ごとにレグ振動数が定まるので，時系列 $X$ に対するレグ振動数は，時点 $t$ の関数になる．

【定義 8】時系列 $X$ のレグ振動数 $F_{X,a,w}(t)$

$X$ を時系列，$a$ を振幅，$w$ をウインドウサイズ，$t$ を時点，とするとき，レグ振動数 $F_{X,a,w}(t)$ を下記で定義する．

$$F_{X,a,w}(t) \equiv \mathrm{sign}(s_{\max}(t)) \times \mathrm{length}(s_{\max}(t))$$

$$\text{such that } s_{\max}(t) = \mathrm{argmax}_{s \in S(X[t:t+w-1],a)}\, \mathrm{length}(s)$$

なお，ウインドウサイズとは，時系列 $X$ が与えられたとき，時点 $t$ ごとに時点 $t$ から開始する部分列を切り出す際の部分列の長さのことである．**図 4** に，ウインドウサイズを 4 とした場合に，時点 $t$ ごとに，時系列から長さ 4 の部分列を切り出した例を示す．

## 3. 課題

### 3.1 応用課題

ビル，プラント，工場など向けの設備保全業務では，「温度偏差が $X$℃ 以上である状態が $Y$ 分以上継続し，かつ，弁開度の偏差が $Z$%以上であると，異常の可能性がある」といった時系列データのウインドウの特徴量で構成される論理制約式で表現される条件をチェックしたいというニーズがある [7], [8]．このニーズに応えるために，設備保全業務の要求分析に基づき，時系列のウインドウの特徴量で構成される条件式を宣言的に記述し，その条件を満たす異常を示す時刻を検索するシステムが開発されている [8]．このようなシステムを広く現場に適用するためには，機器や設備のセンサの振舞いに応じた特徴抽出が重要になる．これらの特徴量の中で，最も典型的なものは，不規則に変動する時系列データにおける上下変動の検出である．というのは，機器や設備が最終的に故障した場合には，値がある閾値を超えるといった検知しやすいデータ挙動を示すのに対して，故障の兆候や機器の劣化の場合は，凸片状のピークや上下振動といったデータの上下変動パターンが正常挙動にまぎれて時折出現する場合も多いからである．

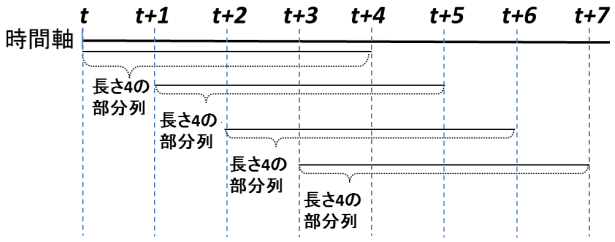



図 4 ウインドウサイズ
**Fig. 4** Window size.

典型的な上下変動パターンとしては，制御系のハンチング現象がある．ハンチング現象とは，制御の目標値の上下を出力値が振動する現象である．たとえば，空調機では，正常時は，部屋の実測温度はある一定時間後に設定温度に収束するが，センサや制御系に異常がある場合には，設定温度より高い温度と低い温度の間を振動する場合がある．したがって，空調機の異常監視では，機器性能や部屋の熱容量に応じて，ある一定の時間内に，指定した温度差以上の上下変動を監視する [7]．この異常監視ルールは，「$w$ 分のウインドウサイズで，$a$℃ 以上の上昇下降するレグが $F$ 回以上交互に出現するならば，異常とする」のような論理式で表現できる．また，ハンチング以外でも，配管の詰まりがある場合の圧力の急激な上昇下降や，電気機器が劣化した場合の電流値のピークの頻出などのデータの振舞いも，ある一定のウインドウサイズ内の振幅とレグ振動数で表現することができる．

### 3.2 技術課題

本論文の目的は，3.1 節で述べた異常監視ルールに必要な特徴抽出を実現するために，時刻 $t$ ごとに値が定まる時系列 $X$，振幅 $a$，ウインドウサイズ $w$ が与えられたとき，時刻 $t$ から開始する長さ $w$ の部分列に対して，振幅 $a$ 以上

の上下変動の頻度を対応させるレグ振動数 $F_{X,a,w}(t)$ を形式的に定義するとともに，レグ振動数を求める実用的な処理性能を持つアルゴリズムを得ることである．

最初の課題である「レグ振動数 $F_{X,a,w}(t)$ の形式的定義」については，2 章で述べたとおりである．レグは従来から知られた概念であるが，レグ振動列やレグ振動数は本論文で新たに導入した概念である．

2 章で述べたレグ振動数の定義をそのままアルゴリズムに翻訳すると，レグ系列の集合 $S(X[t:t+w-1],a)$ を求める処理は，図 3 で示したようにレグの選択の任意性があるために，組合せ爆発を起こす．すなわち，このアルゴリズムは非常に大きな計算量を要する．したがって，その処理は非常に遅くなる．そこで，第 2 の課題は，レグ振動数を求める実用的な処理性能を持つアルゴリズムを得ることである．レグ振動数を求めるアルゴリズムについては，4 章で述べる．

### 3.3 従来技術との差異

不規則に変動する時系列の振動の度合いを定量化する従来手法として，レインフロー法 [2], [3], [4] と平均クロスカウント法 [9] がある．前者のレインフロー法は，レグの振動という概念は持たないが，符号の異なる 2 つのレグの連続出現を扱っているという点で類似している．また，後者の平均クロスカウント法は，レグという概念は持たないが，与えられた振幅 $a$ に対する上下変動の頻度を扱っているという点で類似している．

**(1) レインフロー法**

時系列の振動の度合いを定量化する従来手法としては，材料強度力学の分野で知られているレインフロー法がある．レインフロー法は，不規則に変動する時系列において，極値をとる時点ごとに，その時点から開始するサイクルと呼ばれる部分列を切り出し，サイクルの振幅を計算する方法を提供する．材料強度力学の分野では，このサイクルの振幅の累積値を用いて，材料の疲労損傷の度合いを算出することにより，材料の寿命を予測している．

レインフロー法では，屋根を流れる雨水の流れの比喩を用いて，サイクルを切り出す手続きを定義している．手続きは，「流れの基本法則」と「流れの停止条件」からなる．**図 5** を用いて，レインフロー法による部分列の切り出し方法を説明するが，雨水の流れの比喩を説明しやすくするために，図 5 では，縦軸を時間，横軸を時系列の値とした．

(i) 流れの基本法則

部分列を屋根，極値を軒と見なす．

(a) 水は，軒から屋根の上を時間が進む方向に流れる．

(b) 次の軒に到達しても，流れの停止条件を満たすまでは，下段の屋根に流れ落ちて，その流れを継続する．

(ii) 流れの停止条件

(a) 右向きの流れの場合，流れを開始した軒の値よりも，左側（値が小さい）に軒があれば，流れを停止する．左向きの流れの場合は，右側に軒があれば，流れを停止する．

(b) 上の軒からの流れが，屋根にすでに水が流れていた場合は，流れを停止する．

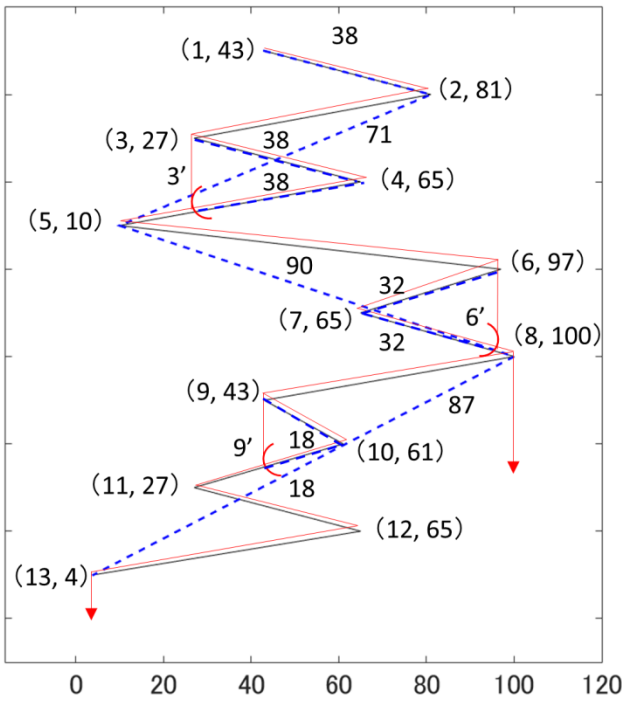



**図 5** レインフロー法
**Fig. 5** Rain flow method.

たとえば，軒 1 から開始する右向きの流れは，軒 3 が軒 1 よりも左にあるので，(ii)(a) の条件により，軒 2 で停止する．したがって，軒 1 から軒 2 の部分列がサイクルになり，その振幅は 38（81 − 43）である．また，軒 5 から開始する右向きの流れは，軒 7 は軒 5 より左にはないので，流れを継続し，軒 8 まで流れる．軒 5 から開始するサイクルの振幅は 90（100 − 10）になる．また，軒 7 から開始する右向きの流れは，軒 5 からの流れが軒 6 から落ちてくるので，6' の時点からすでに水が流れているので，(ii)(b) の条件により停止する．軒 7 から開始するサイクルの振幅は 32（97 − 65）になる．

レグ振動解析の概念と比較すると，流れの停止条件 (ii)(a) は，レグ定義（定義 3 の式 (1)）と同じなので，レインフロー法はレグを扱っているといえる．また，流れの停止条件 (ii)(b) は，長さ 2 のレグ振動列において，レグ振動列の振幅を，レグ振動列を構成するレグの振幅の小さい方の値で定義している点（定義 6 の式 (11)）と似ている．以下，類似の意味を説明する．たとえば，図 5 の時点 4 から時点 3' の部分列（以下，サイクル 4 と呼ぶ）は，流れの停止条件 (ii)(b) によって，サイクルとなる例である．3' が極値でないので，サイクル 4 はレグではないが，サイクルの振幅は，時点の値の差の絶対値で定義されるので，サイクル 4

の振幅は 38 である．レグ振動解析では，時点 3 から時点 4 が上昇レグ（以下，レグ 3 と呼ぶ），時点 4 から 5 が下降レグで，この 2 つのレグの交代出現は，長さ 2 のレグ振動列となり，その振幅は 38 である．すなわち，振幅値が等しくなるという点で，流れの停止条件 (ii)(b) は，長さ 2 のレグ振動列の振幅の定義と似ているといえる．レインフロー法では，レグ 3 とサイクル 4 のように，振幅は同じだが流の方向が反対のサイクルの連続出現をフルサイクルと呼んでおり，長さ 2 のレグ振動列に類似の概念を定義していると解釈できる．

しかし，レインフロー法では，極値をとる時点ごとにレグが一意に決まるので，図 2 に示したようにレグ振動数のように与えられた振幅に応じて，異なるレグを選択することができない．たとえば，レインフロー法では，図 3 の (c) のように，時点 2 から開始するレグとして振幅が最大となる「時点 2 から 5 までのレグ」を選択する．しかし，レグ振動解析では，図 3 (a) や (b) のように，与えられた振幅に応じてレグを選択する点が異なる．以上を総合すると，レインフロー法では，与えられた振幅 $a$ に対して，上下変動の振動の頻度を定量化することはできないことが分かる．

### (2) 平均クロスカウント法

平均クロスカウント法とは，振幅 $a$ が与えられたとき，部分列の平均値からの偏差が $a/2$ を超える点とクロスする点の数を数えるという方法である．平均クロスカウント法は，Mike らが提案する異常検知方式 [9] において，異常の度合いを定量化する素性のうちの 1 つとして用いられている．

たとえば，**図 6** に示すように，振幅 38 に対して，図 3 と同じ部分列の振動数を平均クロスカウント法により求めると，−3 になる．レグ振動数では，図 3 で示したように振幅 38 に対する振動数は 8 であるから，平均クロスカウント法で求めた振動数 −3 は，より短くなっている．上記の例に見られるように，平均クロスカウント法では，部分列全体の平均との偏差を基準とするので，レグ振動数と比較すると，部分列の中での局所的な上下変動を感度良くとらえることができないと考えられる．この仮説については，5.1 節で実データにより検証する．

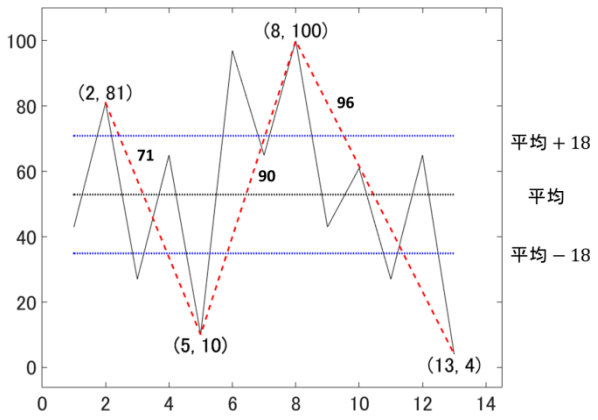



図 6 平均クロスカウント法
**Fig. 6** Mean cross count method.

## 4. レグ振動数を求めるアルゴリズム

### (1) ナイーブなアルゴリズム

3.1 節の定義を手続きに翻訳することにより，時系列 $X$，振幅 $a$，ウインドウサイズ $w$ が与えられたときに，レグ振動数 $F_{X,a,w}(t)$ を求めるナイーブなアルゴリズム CalclegFreq_naive（**図 7**）を得ることができる．アルゴリズムの基本部分は，振幅 $a$ 以上のレグの集合を構成する GetLeg（**図 8**）と，レグ振動列集合を構成する GetLegSeq_naive（**図 9**）である．

図 7 を用いて，アルゴリズム CalclegFreq_naive を説明する．CalclegFreq_naive は，最初に，GetLeg で振幅 $a$ 以上のレグからなるレグ集合 $L$ を構成する（1 行目）．次いで，時系列 $X$ の時点 $t$ ごとの for 文（2〜7 行目）によって，時点 $t$ のレグ振動数 $F[t]$ を求める．以下，for 文内の処理を説明する．

まず，GetLegSeq_naive により，$t$ から始まる振幅 $a$ のすべてのレグ振動列からなる集合 $S_t$ を構成する（3〜4 行目）．次に，$S_t$ における最長のレグ振動列を選択し（5 行目），最長のレグ振動列の符号と長さから時点 $t$ におけるレ

**Algorithm: CalclegFreq_naive(*X, a, w*)**
**Input:** *X*: 時系列，*a*: 振幅，*w*: ウインドウサイズ
**Output:** *F*: レグ振動数

```
1  L ← GetLeg(X, a, w)
2  for t ← 1 to n                    // n: 時系列 X の長さ
3    L_tw ← {l ∈ L | start(l) ≥ t and end(l) ≤ t + w − 1}
4    S_t ← GetLegSeq_naive({ }, [ ], L_tw )
5    l_max ← argmax_{l∈St} length(l)
6    F[t] ← sign(l_max) · length(l_max)
7  end for
8  return F
```

図 7 レグ振動数計算（ナイーブ方式）
**Fig. 7** Longest leg vibration sequence algorithm (naive).

**Algorithm:** GetLeg (***X, a, w***)
**Input:** *X*: 時系列，*a*: 振幅，*w*: ウインドウサイズ
**Output:** *L*: レグ振動列集合

```
1  L = {}
2  for t ← 1 to n − w + 1
3    for e ← 2 to w
4      l_te ← X[t, t + e − 1]
5      if l_te はレグ and amp(l_te) ≥ a
6        L ← L ∪ {l_te}
7      end if
8    end for
9  end for
10 return L
```

図 8 サブルーチン GetLeg
**Fig. 8** Subroutine GetLeg.

**Algorithm: GetLegSeq_naive (*S*, *s* , *L*)**

**Input:** *S*:レグ振動列集合, *s*: レグ振動列, *L*:レグ集合
**Output:** $S_{\text{out}}$ : レグ振動列集合

```
1   if L == 空集合 return S
2   else
3     for each l_last ∈ L
4       s_next ← [s | l_last]    // 系列 s の後ろに l_last を追加
5       S_next ← S ∪ {s_next}    // s_next を集合 S に追加
6       L_next ← {1 ∈ L |start(l) ≥ end(l_last) and
7                   sign(l) · sign(l_last) < 0}
8       S_out ←GetLegSeq_naive(S_next, s_next, L_next)
9     end for
10  end
```

図 9 サブルーチン GetLegSeq_naive
**Fig. 9** Subroutine GetLegSeq_naive.

グ振動数 $F[t]$ を求める（6 行目）．

次に，図 8 を用いて，サブルーチン GetLeg を説明する．時系列 $X$ の長さを $n$ とする．GetLeg は，時系列 $X$ の時点を先頭から $n-w+1$ まで順に時点 $t$ を選択・処理していく for 文からなる（2〜8 行目）．for 文では，時点 $t$ から開始する長さ $2 \le e \le w$ の部分列 $X[t{:}\,t+e-1]$ がレグの条件（定義 3）を満たしており，かつ，振幅が $a$ 以上である場合には，レグ集合 $L$ に追加していく（4〜7 行目）．

図 9 を用いて，サブルーチン GetLegSeq_naive を説明する．GetLegSeq_naive は，レグ集合 $L$ の要素であるレグ $l$ を順に取り出し，そのレグから始まるレグ振動列を求める処理を反復し（3〜9 行目），すべてのレグ $l$ に対して処理を完了したら，レグ振動列集合 $S$ を返す（1 行目）．

3 行目では，レグ集合 $L$ から任意のレグ $l_{\text{last}}$ を選択し，4 行目では，レグ振動列 $s$ の後ろに $l_{\text{last}}$ を追加し，$s_{\text{next}} = [s\,|\,l_{\text{last}}]$ を得る．5 行目では，レグ振動列集合 $S$ に $s_{\text{next}}$ を追加し $S_{\text{next}}$ を構成し，6〜7 行目では，レグ $l_{\text{last}}$ より後ろに出現し，かつ，符号が異なるレグの集合 $L_{\text{next}}$ を構成する．8 行目では，再帰的に，GetLegSeq_naive$(S_{\text{next}}, s_{\text{next}}, L_{\text{next}})$ を呼ぶ．再帰的に，GetLegSeq_naive を呼ぶごとに，中間的に生成されたレグ振動列 $s$ の後ろに新たなレグを追加したレグ振動列 $s_{\text{next}}$ が得られるので，最終的に，すべてのレグ振動列を含む集合 $S$ を構成できる．

最後に，ナイーブなレグ振動数アルゴリズムの計算量について述べる．まず，GetLeg の計算量は，たかだか $n$ 回と $w$ 回の 2 重の for 文による反復処理なので，$O(nw)$ である．次に GetLegSeq_naive は，レグ集合 $L$ の要素数を $k$ とすると，GetLegSeq_naive は，$k$ 回の繰返しごとに，GetLegSeq_naive を再帰的に呼び出すが，呼び出しごとに $L$ の要素数は少なくとも 1 減少するので，全体の計算量は $O(k!)$ となる．レグ集合 $L$ の要素数 $k$ はたかだか $w(w-1)/2$ なので，GetLegSeq_naive の計算量は，$O(w^2!)$ になる．よって，CalclegFleq_naive 全体の計算量は $O(n \cdot (w^2!))$ である．

### (2) 最左レグ振動列

ナイーブなレグ振動数アルゴリズムの計算量が大きいのは，手続き GetLegSeq_naive において，レグ集合 $L$ から振幅 $a$ 以上のレグ振動列を全解探索する処理の計算量が大きいことが原因であった．レグ振動数を求めるためには，最長のレグ振動列をすべて求める必要はなく，最長なレグ振動列の長さだけが分かればよいことに注目すると，少なくとも 1 つ最長なレグ振動列を高速に見つけることができればよいことが分かる．実は，下記で定義する最左レグ振動列は，「部分列に含まれる振幅 $a$ のレグ振動列の中で最大の長さを持つこと」を証明できる．

【定義 9】最左レグ振動列

$X$ を時系列，$a$ を振幅（正の実数値），$w$ をウインドウサイズ，$t$ を時点とする．また，部分列 $X[t{:}\,t+w-1]$ 内にある振幅 $a$ 以上の拡張レグの集合を $L$ とする．

まず，レグ集合 $L$ の中で，終了時点が最も早いレグを $m_1$ とする．続いて，$m_i$ より後ろにあるレグの中で，「符号が $m_i$ と異なり，かつ，終了時点が最も早いもの」を $m_{i+1}$ とする．すなわち，以下のように再帰的に $m_{i+1}$ を選択する．

$$
\begin{aligned}
&m_{i+1} = \text{argmin}_{l \in L_i} \text{end}(l) \\
&\text{ただし，} L_i \equiv \{l \in L \,| \\
&\qquad\qquad \text{start}(l) \ge \text{end}(m_i) \ \text{and} \\
&\qquad\qquad \text{sign}(l) \cdot \text{sign}(m_i) < 0\} \quad \cdots\text{< 条件 1>}
\end{aligned}
$$

この操作を順に適用し得られたレグの系列 $[m_1, m_2, \ldots, m_n]$ を部分列 $X[t, t+w-1]$ における最左レグ振動列と呼ぶ．

最左レグ振動列の例としては，図 3 の (b) がある．(a) から (d) はいずれも振幅 38 のレグ振動列であるが，振幅 38 を超えるレグを順に選択して，前のレグと符号が異なり，かつ，終了時点が最も左にあるレグを選択しているからである．

【定理：最左レグ振動列の最長性】$S(X, a, w, t)$ を，部分列 $X[t, t+w-1]$ に含まれる振幅 $a$ 以上のレグ振動列の集合とする．部分列 $X[t, t+w-1]$ における最左レグ振動列は，$S(X, a, w, t)$ において，最大の長さを持つレグ振動列である．

【証明】$s$ を最左レグ振動列 $s = [l_1, l_2, \ldots, l_n]$ とし，その長さを $n$ とする．$u$ を $S(X, a, w, t)$ における最大の長さを持つ任意のレグ振動列 $u = [k_1, k_2, \ldots, k_m]$ とし，その長さを $m$ とする．$n < m$ と仮定すると，矛盾することを示す．

まず，レグ $l_1$ と $k_1$ は，同符号でなければならないことを示す．なぜなら，レグ $l_1$ と $k_1$ が異符号とすると，$s$ が最左であることと，補題と同様の論法を用いれば，$[l_1, k_1, k_2, \ldots, k_m]$ は長さ $m+1$ のレグ振動列となるので，$u$ が最大の長さを持つことに反するからである．

$l_1$ と $k_1$ は同符号であり，かつ，$s$ が最左であることから，$\text{end}(l_1) \le \text{end}(k_1) \le \text{start}(k_2)$ が成り立つ．したがって，

| Algorithm: CalclegFreq_leftMost (*X, a, w*) | |
|---|---|
| **Input:** | *X*: 時系列, *a*: 振幅, *w*: ウインドウサイズ |
| **Output:** | *F*: レグ振動数 |
| 1 | For $t \leftarrow 1$ to $n$ // *n*: 時系列 *X* の長さ |
| 2 | $s_{\max} \leftarrow \text{GetLegSeq\_leftMost}([\,], t, t+w-1, X)$ |
| 3 | $F[t] \leftarrow \text{sign}(s_{\max}) \cdot \text{length}(s_{\max})$ |
| 4 | end for |
| 5 | return *F* |

図 10 レグ振動数計算（最左レグ振動列方式）
**Fig. 10** Longest leg vibration sequence algorithm (left most).

$[l_1, k_2, \ldots, k_m]$ は，長さ $m$ のレグ振動列になる．同様に，$s$ が最左であることから，$\text{end}(l_2) \leq \text{end}(k_2) \leq \text{start}(k_3)$ であるから，$[l_1, l_2, k_3, \ldots, k_m]$ は，長さ $m$ のレグ振動列にある．$n < m$ と仮定すると，上記の操作を $n$ 回繰り返すことができるので，$[l_1, \ldots, l_n, k_{n+1}, \ldots, k_m]$ は，振幅 $a$ 以上のレグからなるレグ振動列になる．したがって，部分列 $X[\text{end}(l_n){:}\,\text{end}(k_m)]$ において，レグ $k_{n+1}$ と同符号の最左レグが存在することとなり，$s$ が最左レグ振動列であることに矛盾する．よって，定理は証明された． □

### (3) 最左レグ振動列アルゴリズム

(2) の定義を手続きに翻訳することにより，ナイーブなアルゴリズムより高速な最左レグ振動列アルゴリズム CalclegFreq_leftMost を得ることができる．

**図 10** を用いて，CalclegFreq_leftMost の動作を説明する．時系列 $X$ の時点 $t$ ごとに（1 行目），長さ 0 のレグ振動列 [] を引数として，GetLegSeq_leftMost の呼び出しにより，開始時点 $t$ から終了時点 $t+w-1$ までのウインドウ内での最左レグ振動列 $s_{\max}$ を求め（2 行目），$s_{\max}$ レグ振動数を求める処理（3 行目）を繰り返す．

**図 11** を用いて，GetLegSeq_leftMost の動作を説明する．

1 から 3 行目は，停止条件であり，$t$ が $t_{\text{end}}$ 以上の場合には，その時点までに得られているレグ振動列 $s$ を返す．

4 から 8 行目は，$t$ から開始する部分列は，最左レグ振動列の要素にならない場合の分岐であり，$\text{GetLegSeq\_leftMost}(s, t+1, t_{\text{end}}, X)$ を再帰的に呼び出す．6 行目は，$X[t_{\text{next}}] - X[\text{t}]$ が 0 の場合には，端点が唯一の最大値または最小値になるというレグの条件を満たさない場合を示す．7 行目は，最左レグ振動列の < 条件 1> を満たさない場合を示す．以下，$(X[t_{\text{next}}] - X[t])$ を *diff* とする．

9 から 18 行目では，*diff* の符号が変わるか，または，符号が変わらずに $\text{abs}(diff)$ が $a$ を超えるまで，$t_{\text{next}}$ を増やしていく．

*diff* の符号が変わることにより While 文を抜けた場合は，$t \leq \tau \leq t_{\text{next}} - 1$ とするとき，$\tau$ から開始する部分列は，最左レグ振動列の構成要素にはならない．なぜなら，補題と同様の論法を用いると，$\tau$ から開始する部分列で，最左レグ振動列の < 条件 1> を満たす符号を持つレグは，振幅が $a$ 以上にならないからである．19 から 20 行目は，*diff* の符号が変わった場合の処理であり，再帰的に，$\text{GetLegSeq}_{\text{leftMost}}(s, t_{\text{next}} - 1, t_{\text{end}}, X)$ を呼び出す．

| Algorithm: GetLegSeq_leftMost ($s$, $t$, $t_{\text{end}}$, $X$) | |
|---|---|
| **Input:** | *s*:レグ振動列, *t*:開始時点,$t_{\text{end}}$:終了時点,*X*:時系列 |
| **Output:** | *s*:レグ振動列 |
| 1 | if $t \geq t_{end}$ |
| 2 | return $s$ |
| 3 | end |
| 4 | $t_{\text{next}} \leftarrow t+1$ |
| 5 | $diff \leftarrow X[t_{\text{next}}] - X[t]$ |
| 6 | if $diff == 0$ |
| 7 | $\|\|$ ($s$ ~=[] & $\text{sign}(diff) == \text{sign}(\text{last}(s))$ // t から開始するレグは，最左レグ振動列の要素でない |
| 8 | $s \leftarrow \text{GetLegSeq\_leftMost}(s, t+1, t_{\text{end}}, X)$ |
| 9 | else |
| 10 | if $s$ ~= [] |
| 11 | $\text{next\_sign} \leftarrow \text{sign}(\text{last}(s)) * -1$ |
| 12 | else |
| 13 | $\text{next\_sign} \leftarrow \text{sign}(diff)$ |
| 14 | end |
| 15 | while $\text{sign}(diff) == \text{next\_sign} \,\|\|\, \text{abs}(diff) < a$ |
| 16 | $t_{\text{next}} \leftarrow t_{\text{next}} + 1$ |
| 17 | $diff \leftarrow X[t_{\text{next}}] - X[t]$ |
| 18 | end |
| 19 | if $\text{sign}(diff) \sim = \text{next\_sign}$ // τ から開始するレグは，最左列レグ振動列の要素でない // 但し，$t \leq \tau \leq t_{\text{next}} - 1$ |
| 20 | $s \leftarrow \text{GetLegSeq\_leftMost}(s, t_{\text{next}}, t_{\text{end}}, X)$ |
| 21 | else // $t$ から開始するレグは，構成要素である |
| 22 | $abs\_diff\_pre \leftarrow \text{abs}(X[t_{\text{next}} - 1] - X[t])$ |
| 23 | while $\text{sign}(diff) == \text{next\_sign}$ |
| 24 | & $\text{abs}(diff) > abs\_diff\_pre$ |
| 25 | $abs\_diff\_pre \leftarrow \text{abs}(diff)$ |
| 26 | $t_{\text{next}} \leftarrow t_{\text{next}} + 1$ |
| 27 | $diff \leftarrow X[t_{\text{next}}] - X[t]$ |
| 28 | end |
| 29 | $l_{\text{next}} \leftarrow X[t{:}\, t_{\text{next}} - 1]$ |
| 30 | $s_{\text{next}} \leftarrow [s|l_{\text{next}}]$ // 系列 s の末尾に$l_{\text{next}}$を追加 |
| 31 | $s \leftarrow \text{GetLegSeq\_leftMost}(s_{\text{next}}, t_{\text{next}} - 1, t_{\text{end}}, X)$ |
| 32 | end |

図 11 サブルーチン GetLegSeq_leftMost
**Fig. 11** Subroutine GetLegSeq_leftMost.

一方，*diff* の符号が変わらずに $\text{abs}(diff)$ が $a$ を超えることにより While 文を抜けた場合は，$t$ から始まる振幅 $a$ 以上の最左レグが存在する．23 から 29 行目は，レグの右端の時点を求めることにより，$t$ から始まる最左レグ $l_{\text{next}}$ を求める．0 行目で，$s$ の末尾に $l_{\text{next}}$ を追加したレグ振動列を $s_{\text{next}}$ を構成して，31 行目で，再帰的に $\text{GetLegSeq\_leftMost}(s_{\text{next}}, t_{\text{next}}+1, t_{\text{end}}, X)$ を呼び出す．

最後に，最左レグ方式のレグ振動数アルゴリズムの計算量について述べる．GetLegSeq_leftMost では，$t$ から $t+w-1$ まで，1 時点ずつ増やしながら後戻りなく処理をしているので，計算オーダは $O(w)$ になる．すなわち，ウインドウサイズ $w$ の部分列の最左レグ振動列を求める処理は $O(w)$ になる．したがって，CalclegFreq_leftMost は，時系列 $X$

の長さ $n$ の for 文で GetLegSeq_leftMost を呼び出す処理なので，計算オーダは，$O(nw)$ になる．

## 5. 評価

### 5.1 実データに対するレグ振動数計算の評価実験

提案方式をビル空調機の設備保全業務に適用することにより，実用性を評価する．また，従来技術との比較として，レインフロー法は振動数に相当する概念を定義できないので，平均クロスカウント法との異常判定の精度を比較する．

**(1) 設備保全現場適用による業務改善効果**

**表 1** に，評価環境のハードウェア，ソフトウェアを示す．対象データは，空調機の室内温度で，3 年分，サンプリング周期 1 分で，総点数は 1,578,239 である．実験対象とする設備の保全ルールとしては，現場での要求分析ヒアリングにより得られた「ハンチング異常制約」[8] を対象とした．ハンチング異常制約とは，「室内温度が $X$ 分間で $Y$℃以上の振幅で上下する変動が $Z$ 回存在する時間」を異常として検出するというルールである．このルールを現場では，「トレンドのばたつきをみる」と呼んでいたが，レグ振動数は，現場の分析者のヒアリングに基づき，「トレンドのばたつき」を定量的に表現した概念である．レグ振動解析では，$X$ はウインドウサイズ，$Y$ は振幅，$Z$ はレグ振動数の絶対値 $Z$ に対応する．$X$，$Y$ などのパラメータは設備ごとに異なるが，振幅が大きいほど高い警告レベルとすることで，振幅に応じた設備保全を実施できる．この実験では，レグ振動数の絶対値 $Z$ が 4 以上を異常と定義し，ウインドウサイズ $X$ を 30 分，振幅 $Y$ は 1.0℃，2.0℃，4.0℃の警告レベルで監視する業務を想定した．レグ振動数の絶対値が 4 というのは，センサデータが上昇，下降，上昇，下降と 2 往復することを意味しており，評価対象とした設備保全の現場で，データのばたつきの条件として用いられていた基準値である．

各々の温度幅でレグ振動数の絶対値が 4 以上となった時点の数は，順に，1901 点（0.12%），454 点（0.029%），69 点（0.0044%）である．なお，括弧の中は，時点の中での異常となった時点の数の比率である．また，処理時間は，各々，0.612 秒，0.554 秒，0.489 秒であり，実用上ストレスのない時間で結果を得ることができた．

著者らが関わっている設備保全の現場では，定期的な設備診断作業は，「大量の実データから集約値（平均，最大値，分散など）が異常を示すセンサデータから，人手で詳細にチェックすべき特徴的なデータを抽出する下分析作業」と，「ドメイン知識を用いて，特徴的なデータやその関連データ分析することにより，設備診断レポートを作成する本分析作業」からなっていた．この下分析作業では，ウインドウサイズと振幅値に応じたトレンドのばたつきを目視でチェックする作業を人手で実施していたが，データのばたつきを定量化するレグ振動数を提示するデータ分析支援ツールにより，この作業を自動化できた．この結果，下分析作業時間を 85%短縮するとともに，トータルな分析時間を 60%削減することができた．

**表 1** 評価環境
**Table 1** Environment for evaluation.

| | |
|---|---|
| CPU | Intel Core i7-3770，3.4GHz(4 Core) |
| メモリ | 16GB |
| OS | Windows 7 Professional SP1 (64bit) |
| 実装言語 | Visual C++ 2012 |

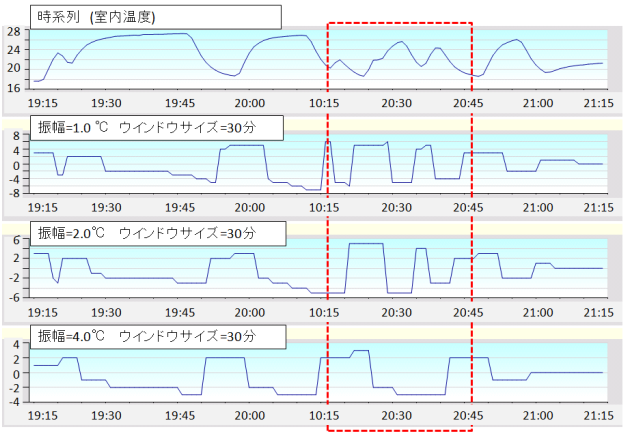



**図 12** 実データのレグ振動数の例
**Fig. 12** Trend graphs of experimental data.

また，設備や機器設置条件などにより，異常と判断する振幅や振動数が変わる場合もあるが，分析者が振幅や振動数をパラメータとして自由に設定できるので，設備や機器設置条件に応じた分析がしやすくなるという効果も得られた．以下，**図 12** と**図 13** を用いて，振幅を変化させたときに，実データのレグ振動数の事例を示す．

図 12 は，分析支援ツールが表示したスナップショットである．上から順に，分析対象の温度データ，振幅 1℃ のレグ振動数，振幅 2℃ のレグ振動数，振幅 4℃ のレグ振動数のトレンドグラフである．図中で，点線で示した 30 分の時区間では，各々のレグ振動数は，6，−5，そして，2 である，振幅に応じて振動数が異なっていることが分かる．警告の条件はレグ振動数の絶対値 4 以上であるから，この 30 分の時区間では，1.0℃，2.0℃ レベルの警告が発生しているが，4.0℃ レベルの警告は発生していないことが分かる．

図 13 に，図 12 の点線で囲んだ 30 分の時区間における最左レグ振動列を示す．振幅に応じて，選択されるレグが異なっていることが分かる．設備保全業務の担当者は，警告に対応するレグ振動列を参照することにより，ウインドウサイズ中でのより詳細な異常な変動の時間帯が分かるので，異常の原因調査に役立てることができる．

**(2) 従来技術との精度比較**

**表 2** に，振幅が 2℃ の場合に，レグ振動解析と平均クロスカウント法との精度の比較結果を示す．評価の現場で

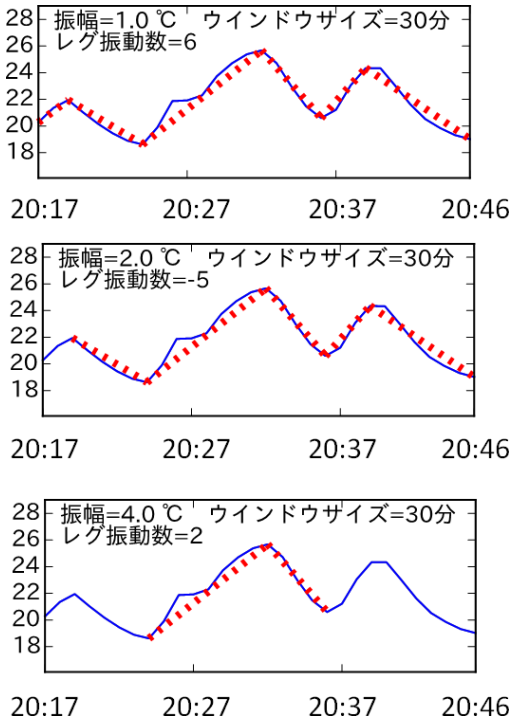



図 13 レグ振動列

**Fig. 13** Leg variation sequences.

表 2 レグ振動解析と平均クロスカウント法の比較

**Table 2** The comparison between leg analysis and mean cross count.

| アルゴリズム | 適合率 | 再現率 |
|---|---|---|
| レグ振動解析(振幅 2℃) | 100% | 100% |
| 平均クロスカウント法(振幅 2℃) | 100% | 13.5% |
| 平均クロスカウント法(振幅 1.8℃) | 99.4% | 15.3% |
| 平均クロスカウント法(振幅 1.6℃) | 96.4% | 17.2% |
| 平均クロスカウント法(振幅 1.4℃) | 86.1% | 19.0% |
| 平均クロスカウント法(振幅 1.2℃) | 74.4% | 21.5% |
| 平均クロスカウント法(振幅 1℃) | 65.0% | 23.8% |

用いているレグ振動数の絶対値が 4 以上のものを正解としたとき，平均クロスカウント法で振幅を 2℃ から 0.2℃ 刻みで 1℃ まで変化させた場合の適合率と再現率を示している．振幅 2℃ のレグ振動数を正解としているので，レグ振動解析の適合率は，振幅 2℃ の適合率と再現率は 100%とした．平均クロスカウント法は，振幅（2℃）の場合は適合率が 100%であるが，再現率は 13.5%にとどまっている．また，振幅を小さくすることにより，再現率を向上させることができるが，振幅を 1℃ にしても再現率は 23.8%にしか達しない．再現率が低いということは異常を見逃す率が高いということなので，再現率 100%に近い精度が要求される設備保全の現場への適用は難しい．平均クロスカウント法は，指定する値からの偏差の振動を監視する場合には有用だが，レグの上下変動の振動を監視する場合には，適さないと考える．また，3.3 節 (2) で，「平均クロスカウント法では，局所的な上下変動を感度良くとらえることができない」という仮説を述べたが，この仮説を実データでも確認できた．

一方，レグ振動解析では，与えられた振幅以上の上下変動を漏れなく検知することを目的としているので，振幅を小さくした場合は，雑音のようなランダムかつ小さな変動も漏れなく検知する．たとえば，上記にあげた室温の 2℃ というのは十分大きな値であるが，振幅 0.2℃ の上下変動を検知する場合には，雑音を拾う可能性がある．一方，平均クロスカウント法では，与えられたウインドウ幅の平均をとるので，雑音を除去できるメリットがある．

### 5.2 速度評価

#### (1) 評価方針

レグ振動数を現実に適用する際に，対象データに対するドメイン知識がない場合には，様々な振幅とウインドウサイズでレグ振動数を計算することにより，異常検知に適した振幅とウインドウサイズを求める必要がある．レグ振動数の計算は，振幅が小さいほど遅くなり，また，ウインドウサイズが大きいほど遅くなるが，実用的なレグ振動計算アルゴリズムは，ウインドウサイズと振幅が変わっても，実用的な時間で計算が終了しなければならない．

実用的な時間目標を決めるための利用シーンとして，1 台の PC サーバで，夜間バッチ処理で，1 万種程度の 1 分周期 1 日分の時系列（データ長 1,440）に対して，振幅とウインドウ数の組合せ 100 パターンのレグ振動数を計算し，翌日にその計算結果を利用するという運用を想定する．すなわち，1 万程度の時系列のレグ振動数を 8 時間（一晩，28,800 秒）で計算できる性能を目標とすると，1 レグ振動数あたりの目標実行時間は 0.0288 秒になる．

レグ振動数の計算時間は，振幅を固定しウインドウサイズを変化させた場合のウインドウサイズ依存性と，ウインドウサイズを固定し振幅を変化させた場合の振幅依存性を評価する．また，4 章 (1) に示したナイーブな方式と最左レグ方式の実行時間を比較することにより，最左レグ方式の効果を示すとともに，速度の目標値の達成有無を検証する．なお，定理「最左レグ振動列の最長性」により，ナイーブな方式と最左レグ方式の結果は一致することを証明済であるが，検証の目的で比較評価データに対して一致することも確認した．

#### (2) 評価条件

実験では，評価対象データとして，振舞いが異なる 6 種類の時系列を対象とした．具体的には，Keogh らが公開している時系列データセット [10] と，ノイズを混入させた正弦波形とした（図 **14**）．これらの時系列は，長さ 5,000〜10,000 であるが，最左レグ方式の実行時間はデータ長に線形であるため，目標実行時間は，実験データに対しては，3.5〜7 倍の約 0.1〜0.2 秒になる．

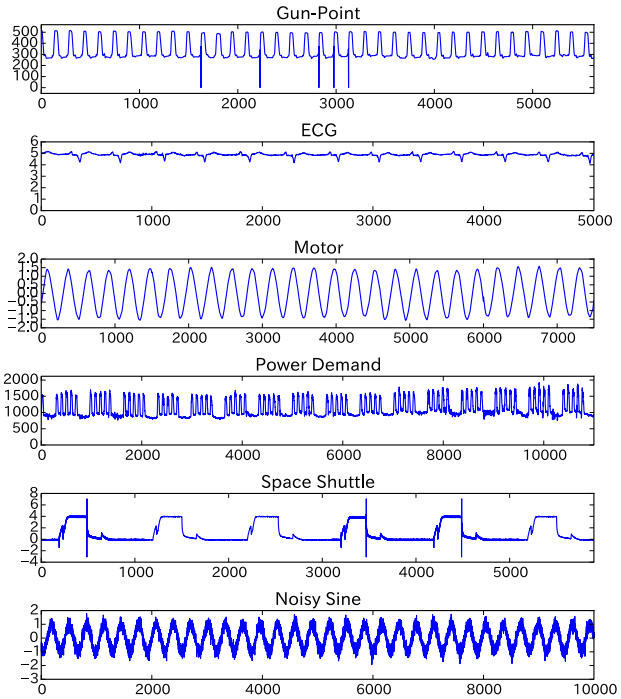



図 14 実験データのトレンドグラフ
**Fig. 14** The trend graphs of experimental data.

表 3 実験データの諸元
**Table 3** The specification of experimental data.

| データ名称 | データ長 | 基準振幅 | 基準ウインドウサイズ |
|---|---|---|---|
| Gun-Point | 5,625 | 258.1 | 120 |
| ECG | 5,000 | 0.522 | 20 |
| Motor | 7,500 | 1.570 | 300 |
| power demand | 11,000 | 582.5 | 50 |
| Space Shuttle（マロッタバルブ） | 5,901 | 5.080 | 200 |
| Noisy Sine | 10,000 | 1.855 | 300 |

ウインドウサイズ依存性の評価では，レグの振動をとらえるのに適切と目視で判断したウインドウサイズ（基準ウインドウサイズ）の 1/30〜30 倍の範囲のウインドウサイズで実験した．振幅依存性の評価では，実験データの値域の半分を基準値（基準振幅）として，その基準値を 1/50〜2 倍の範囲の振幅で実験した．

**表 3** に，評価対象データのデータ長，基準振幅，そして，基準ウインドウサイズを示す．

### (3) ウインドウサイズ依存性

**図 15** に，ウインドウサイズ依存性の実行時間をデータごとに示す．横軸がウインドウサイズ，縦軸が実行時間（単位：秒，対数目盛）である．実行時間は，ナイーブなアルゴリズムで 10 回，最左レグ振動列アルゴリズムで 100 回測定した平均値である．

ナイーブなアルゴリズム（図中の naive）では，4 章 (1) で述べたとおり，実行時間がウインドウサイズに対して指数的に増加することが確認できる．そのため，ウインドウサイズ依存性の測定を途中で打ち切らざるをえなかった．一方，最左レグ振動列アルゴリズム（図中の left-most）は，実行時間の増加が緩やかであり，高速化の効果を確認できた．また，最左レグ振動列アルゴリズムの実行時間は，1 日分程度のウインドウサイズ（1,440）であっても，最大で 0.2 秒程度（Noisy Sine）であり，実用的な時間で実行できることが確認できた．

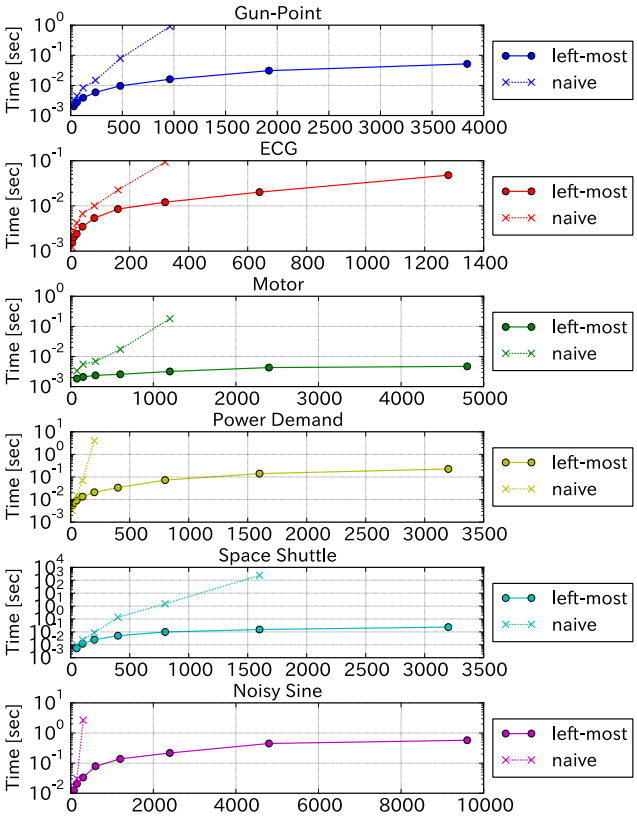



図 15 実行時間のウインドウサイズ依存性
**Fig. 15** Dependency of the execution time on window size.

### (4) 振幅依存性

**図 16** に，振幅依存性の実行時間をデータごとに示す．グラフの縦軸と横軸は，図 15 と同様である．こちらも，実行時間は，ナイーブなアルゴリズムで 10 回，最左レグ振動列アルゴリズムで 100 回測定した平均値である．この評価では，ウインドウサイズを基準ウインドウサイズの 1〜16 倍に設定して実験した．

4 章 (1) で述べたとおり，ナイーブなアルゴリズムの計算量は，レグ集合に含まれるレグ数に依存する．振幅が小さくなるに従いレグ数は増加するため，ナイーブなアルゴリズムの実行時間が振幅に従って増加していることが確認できる．一方，最左レグ方式の場合は，ウインドウサイズが一定であれば，振幅が変わっても実行時間もおおむね一定であることが確認できた．

実行時間は，マロッタバルブの時系列（Space Shuttle）

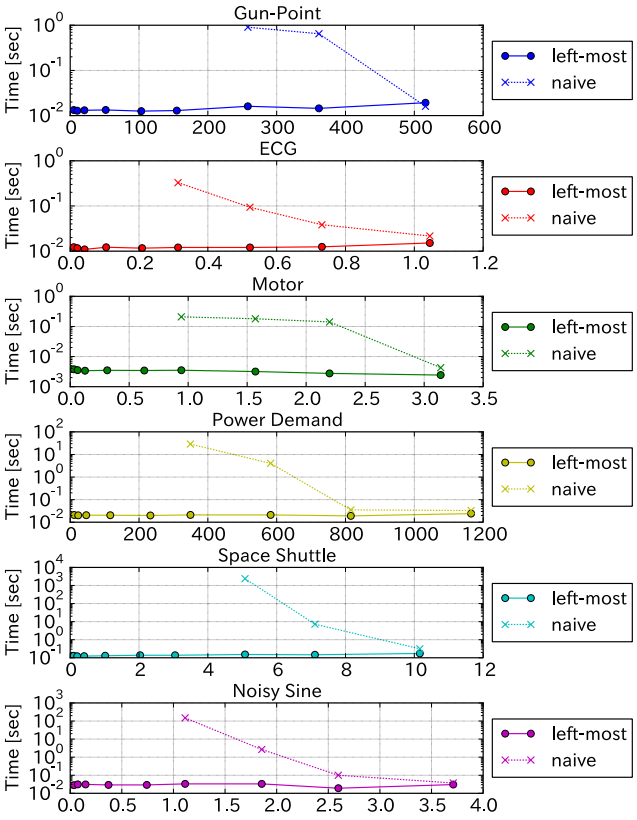



図 16 実行時間の振幅依存性

**Fig. 16** Dependency of the execution time on amplitude.

以外は，0.014〜0.056 秒であり，実用的な時間といえる．マロッタバルブの時系列も，(3) で示したとおり，基準ウインドウサイズ付近では，0.02 秒前後で実行できており，実用的な時間といえる．

## 6. ストリーム処理との関係に関する考察

本研究の関連として，データ工学分野のデータストリームの問合せ処理がある．以下では，データストリーム問合せ処理を説明した後，本論文の提案方式と比較する．

### 6.1 データストリーム問合せ処理

データストリーム問合せでは，データストリームが満たすべきパターンを記述する問合せ言語を提供し，その問合せを受理するオートマトンを構成することにより，問合せを評価する．

データストリームの問合せ言語として代表的なものに，SASE+ [5] や Esper [6] がある．SASE+ では，問合せとして，データストリームに対する選択，結合，集約，そして，クリーネ閉包を含む制約が記述できる．Esper も同様の問合せ言語を提供するが，パターンにマッチする系列が複数ある場合に，系列をすべて求めるかあるいは，最初の 1 つを求めるかなどを制御する every 演算子を提供している．レグ振動解析と関連の深い記述例としては，Balkesen ら [11] の論文にある**図 17** のパターンがあげられる．以下

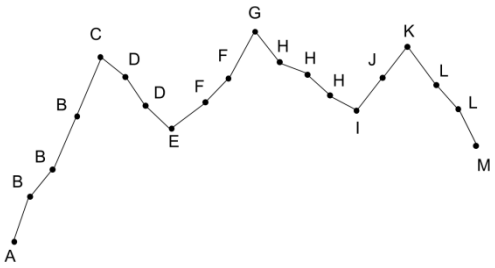



図 17 データストリーム問合せで記述できるパターン例（文献 [11] からの引用）

**Fig. 17** A sample pattern described by data stream query language (Refered from Ref. [11]).

に，図 17 のパターンを記述する問合せの例を示す．

PATTERN (A B* C D* E F* G H* I J* K L* M) (13)

DEFINE B AS (B.price ≥ PREV(B.price)) (14)

C AS (C.price > PREV(C.price)) (15)

D AS (D.price ≤ PREV(D.price)) (16)

E AS (E.price < PREV(E.price) AND E.price > A.price)) (17)

F AS (F.price ≥ PREV(F.price)) (18)

G AS (G.price ≥ PREV(G.price) AND G.price > C.price) (19)

H AS (H.price ≤ PREV(H.price)) (20)

I AS (I.price < PREV(I.price) AND I.price > A.price) (21)

J AS (J.price ≥ PREV(J.price)) (22)

K AS (K.price < PREV(K.price) AND K.price < G.price) (23)

L AS (L.price ≤ PREV(L.price)) (24)

M AS (M.price ≤ PREV(M.price) AND M.price < E.price AND M.price < I.price) (25)

MAXLENGTH 100

上記の記述では，式 (13) は，パターン全体を正規表現で記述し，式 (14) から (25) で式 (13) を構成する B，C などの記号が満たす条件を記述している．たとえば，式 (13), (14) は，「時点 B の価格は，値が単調増加する系列（以下，価格は省略する）である」という制約を表現している．式 (15) は，「時点 C は前の時点よりも小さい」を表現している．式 (13) から (16) を合わせると，「時点 C で，極大値をとる」ことを表現できる．同様に，式 (13) と，式 (15) から式 (18) と合わせると，「時点 E で，時点 A よりも大きな値を持つ極小値をとる」という制約を表現できる．同様

に，式 (13) と，式 (18) から式 (24) により，「時点 G で極大値をとる」，「時点 I で，時点 A よりも大きな値を持つ極小値をとる」，「時点 K で，時点 G より小さな極大値をとる」，「時点 M で，時点 E や時点 I より小さな極小値をとる」という制約を記述できる．

### 6.2 本論文の提案方式との比較

本節では，最初に，レグ振動解析とデータストリーム問合せ言語との関係を述べる．次に，レグ振動数と最左レグ振動列がデータストリーム問合せ言語により記述可能かどうかについて考察する．最後に，レグ振動解析におけるデータストリーム問合せ言語の活用について考察する．

**(1) レグ振動解析とデータストリーム問合せ言語との関係**

6.1 節で述べたように，データストリーム問合せ言語では，時系列のパターンを記述して，その記述を満たすパターンを検索することができる．一方，本論文の主な主張点は，レグ振動列のパターンの記述方式とその検索方式ではなく，『補題「最長レグ振動数の符号の同一性」を証明することにより，部分列に対して，符号付きのレグ振動数が定義可能であることを示したこと』，さらに，『定理「最左レグ振動列の最長性」を証明することにより，部分列のレグ振動数（定義 7）と，「部分列における最左レグ振動列（定義 9）の長さと符号の積」とが一致することを示し，その帰結として，部分列のウインドウサイズを $w$ とするとき，部分列のレグ振動数を求める計算量 $O(w)$ のアルゴリズムが得られることを示したこと』にある．また，これらの導出では，オートマトンを受理するパーザのストレッジや処理の共有や並列化などの最適化手法ではなく，レグ振動列が満たす不等式の性質を用いた推論に基づいている．したがって，本論文の提案内容は，データストリーム問合せ言語との独立の関係にあると考える．

**(2) レグ振動数の記述可能性について**

次いで，本論文の主な最初の主張点である定義 7 のレグ振動数の記述可能性について考察する．SASE+ や Esper などのデータストリーム問合せ言語を用いて，「与えられた振幅より大きく，かつ符号の異なるレグが反復出現するパターン」を定義できるので，本論文のレグ振動列（定義 5）を表現することができる．すなわち，レグ振動数の定義（定義 7）にあるレグ振動列集合 $S(X[p\!:\!q], a)$ を求める問合せを表現できる．しかし，個々のパターンに関する制約は記述できるが，パターンの集合に関する制約を記述できないので，「パターンにマッチしたレグ振動列の集合の中で長さが最大なレグ系列」という制約は表現できない．また，本論文の主な 2 つ目の主張点である『部分列のレグ振動数（定義 7）と，「部分列における最左レグ振動列（定義 9）の長さと符号の積」とが一致することを示すことはできない．

**(3) 最左レグ振動列の記述可能性について**

SASE+ は，パターンの記述に加えて，問合せパターンにマッチしない時点をスキップする skip-till-next-match などのパターンマッチを制御するための機構を提供している．SASE+ が提供するパターン制御機構や Esper の every 演算子を用いて，最左レグ振動列のパターンを記述できるかどうかは自明ではないが，必要に応じて新たな制御機構を問合せ言語に組み込むことにより，記述可能になると考える．

**(4) データストリーム問合せ言語の活用**

本節 (3) により，最左レグ振動列を求める実装手段として，データストリーム問合せ言語を用いるという選択肢があることが分かった．著者らは，「アルゴリズムのカスタマイズ性」や「他機能との連携容易性」などの要請から，C 言語により最左レグ振動列を求める処理を実装した．アルゴリズムのカスタマイズ例としては，「振幅ごとのレグ振動数の分布を求める」，「複数の振幅に対するレグ振動数の同時計算により，重複計算を避けることで処理を高速化」，「部分列 $X[t, t+w-1]$ のレグ振動数計算の中間結果を部分列 $X[t+1, t+w]$ のレグ振動数計算に用いることにより，重複計算を避けることで処理を高速化」などがある．C 言語による実装は，探索の足切りやデータ構造の工夫など，計算対象の特殊性を用いてアルゴリズムを高速化できるという利点がある．一方，データストリーム問合せ言語は，汎用の記述を提供するので，アドホックな問合せに柔軟に対応できるというメリットがある．レグ振動解析に関して，処理速度や開発効率の点から，データストリーム問合せ言語による実装と，C 言語による実装とを比較することは，今後の課題である．

## 7. おわりに

本論文では，雑音などの変動幅の小さな局所的な上下変動を含む時系列の大域的な上下変動の頻度を計算するレグ振動解析を提案した．本論文の貢献は，下記の 3 点である．

(1) Fink らが扱った単独のレグだけでなく，上昇傾向のレグと下降傾向のレグが交代出現するレグ振動列という新たな概念を導入した．また，「補題：最長レグ振動列の符号の同一性」を証明することにより，与えられた振幅 $a$ と部分列に対して，符号付きレグ振動数を定義可能であることを示した．

(2)「定理：最左レグ振動列の最長性」を証明することにより，最左レグ振動列を求めればレグ振動数を計算できることを示した．その結果に基づき，ウインドウサイズを $w$ とするとき，部分列のレグ振動数を求める計算量 $O(w)$ のアルゴリズムを導出した．

(3) 設備保全の現場への適用評価により，提案方式の有用性を確認した．また，特性の異なる 6 種類のデータに対して，レグ振動解析のウインドウサイズと振幅に対する実行時間

の依存性を評価することにより，レグ振動解析が実データに対して実用的な時間で処理可能であることを示した．

## 参考文献


[1] Fink, E. and Kevin B.P.: Indexing of Compressed Time series, *DATA MINING IN TIME SERIES DATABASES*, pp.43–65, World Scientific (2004).

[2] 遠藤達雄，松石正典，光永公一，小林角市，高橋清文：「Rain Flow Method」の提案とその応用，九州工業大学研究報告 (1974)，入手先 ⟨http://www-it.jwes.or.jp/qa/details.jsp?pg_no=0040020170⟩.

[3] Rychlik, I.: A new definition of the rainflow cycle counting method, *International Journal of fatigue*, Vol.9, No.2, pp.119–121 (1987).

[4] Musallam, M. and Johnson, C.M.: An efficient implementation of the rainflow counting algorithm for life consumption estimation, *IEEE Trans. Reliability*, Vol.61, No.4, pp.978–986 (2012).

[5] Agrawal, J., Diao, Y., Gyllstrom, D. and Immerman, N.: Efficient pattern matching over event streams, *Proc. 2008 ACM SIGMOD International Conference on Management of Data*, pp.147–160 (2008).

[6] EsperTech Inc.: EsperTech Event Series Intelligence (2015), available from ⟨http://www.espertech.com/⟩.

[7] 木下栄蔵，金尾　毅，柴本　覚，宮坂房千加，臼井清治，原　英嗣：ビル空調のエネルギー・環境・設備のための統計解析，5.2 運転データによる異常検知診断，pp.208–236，オーム社 (2006).

[8] 今村　誠，竹内丈志，北上眞二，菅野幹人，撫中達司：設備異常診断用の時系列データ検索言語 TPQL，電気学会論文誌 C（電子・情報・システム部門誌），Vol.134, No.1, pp.156–167 (2014).

[9] Jones, M., Nikovski, D., Imamura, M. and Hirata, T.: Anomaly Detection in Real-Valued Multidimensional Time Series, *The 2nd ASE International Conference on Big Data Science and Computing* (2015), available from ⟨http://ase360.org:8080/bitstream/handle/123456789/56/⟩.

[10] Keogh, E., Zhu, Q., Hu, B., Hao, Y., Xi, X., Wei, L. and Ratanamahatana, C.A.: The UCR Time Series Classification/Clustering Homepage (2011), available from ⟨http:// www.cs.ucr.edu/~eamonn/time_series_data/⟩.

[11] Balkesen, C., Dindar, N., Wetter, M. and Tatbul, N.: RIP: run-based intra-query parallelism for scalable complex event processing, *Proc. 7th ACM International Conference on Distributed Event-based Systems*, pp.3–14 (2013).



**今村 誠** （正会員）

1986 年 3 月京都大学大学院工学研究科数理工学専攻修士課程修了．同年 4 月三菱電機（株）入社．情報技術総合研究所勤務．博士（情報科学）．データ分析，構造化文書処理，自然言語処理，CALS/EC システム等の研究開発に従事．情報処理学会平成 18 年度論文賞，（社）日本電機工業会平成 13 年度電機工業技術功績者発達賞，平成 16 年度山下記念研究賞（情報学基礎）等を受賞．電気学会，人工知能学会各会員．



**中村 隆顕** （正会員）

2002 年 3 月九州大学大学院システム情報科学府情報理学専攻修士課程修了．同年 4 月三菱電機（株）入社．情報技術総合研究所勤務．データ分析，データ検索，データベース技術の研究開発に従事．



**柴田 秀哉**

2008 年 3 月京都大学大学院情報学研究科数理工学専攻修士課程修了．同年 4 月三菱電機（株）入社．情報技術総合研究所勤務．データ検索，データ圧縮，データマイニング等の研究開発に従事．電子情報通信学会会員．



**平井 規郎**

1991 年 3 月東京工業大学大学院理工学研究科情報科学専攻課程修了．同年 4 月三菱電機（株）入社．情報技術総合研究所勤務．データ検索，データ分析，データベース技術の研究開発に従事．データベース学会会員．

**北上 眞二** （正会員）

1983 年 3 月富山大学大学院電気工学科修士課程修了．同年三菱電機（株）入社．主に，データベース技術，M2M サービスシステム技術，省エネ制御技術の研究に従事．2013 年 3 月東北大学大学院情報科学研究科博士後期課程修了．博士（情報科学）．現在，三菱電機ビルテクノサービス（株）において，ビル向け遠隔サービス技術の開発に従事．電気学会，日本工学教育協会各会員．

**撫中 達司**

1986 年三菱電機入社．オペレーティングシステム（OS），ネットワークシステム，ネットワークセキュリティの研究開発に従事．2015 年より東海大学情報通信学部組込みソフトウエア工学科教授．製造業等の産業用システムへの IoT 適用に関する研究開発に従事．第 21 回電気通信普及財団賞テレコムシステム技術賞受賞（2006 年），第 62 回電機工業技術功労表彰奨励賞受賞（2012 年）．博士（工学）．IEEE Senior Member，電子情報通信学会会員，本会シニア会員．